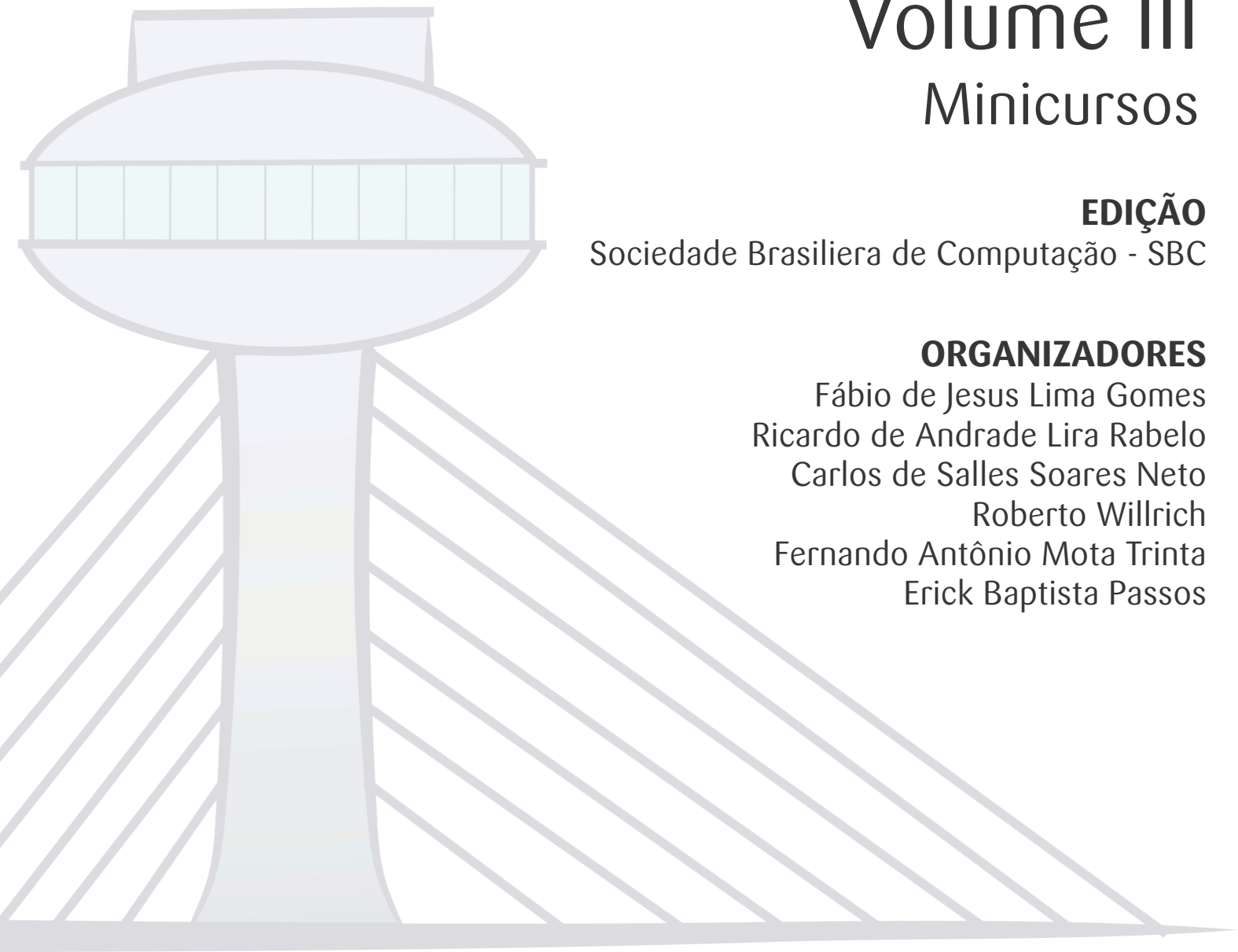
WebMedia2016

22º Simpósio Brasileiro de Sistemas Multimídia e Web

ANAIS

Volume III

Minicursos



EDIÇÃO

Sociedade Brasiliera de Computação - SBC

ORGANIZADORES

Fábio de Jesus Lima Gomes
Ricardo de Andrade Lira Rabelo
Carlos de Salles Soares Neto
Roberto Willrich
Fernando Antônio Mota Trinta
Erick Baptista Passos

Teresina - Piauí

08 a 11 de novembro

Promoção

Realização

Patrocínio

WebMedia2016

XXII Simpósio Brasileiro de Sistemas Multimídia e Web

De 08 a 11 de novembro de 2016

Teresina – PI – Brasil

ANAIS (volume 3)
Minicursos

Organizadores

Fábio de Jesus Lima Gomes (IFPI)

Ricardo de Andrade Lira Rabelo (UFPI)

Carlos de Salles Soares Neto (UFMA)

Roberto Willrich (UFSC)

Fernando Antonio Mota Trinta (UFC)

Erick Baptista Passos (IFPI)

Realização

Sociedade Brasileira de Computação (SBC)

Em cooperação com

ACM/SIGWEB e ACM/SIGMM

Organização

Instituto Federal do Piauí (IFPI)

Universidade Federal do Piauí (UFPI)

Universidade Estadual do Piauí (UESPI)

FICHA CATALOGRÁFICA
Serviço de Processamento Técnico - IFPI
Biblioteca Dr. Francisco Montojos

S621a Simpósio Brasileiro de Sistemas Multimídia e Web (Vol. 3): Minicursos (22.,2016: Teresina, PI).
Anais do XXII Simpósio Brasileiro de Sistemas Multimídia e Web (Vol. 3): Minicursos, 8 a 11 de novembro, 2016, Teresina, Piauí.
263p.
E-book.
ISBN: 978-85-7669-333-8.

Evento promovido pela Sociedade Brasileira de Computação (SBC), Porto Alegre, RS/ Organização: UFPI, IFPI, UESPI.

1. TV Digital. 2. Computação Ubíqua e Móvel. 3. Web e Redes Sociais. 4. Multimídia. I. Título.

CDD 006.7

XXII WebMedia

Simpósio Brasileiro de Sistemas Multimídia e Web
8 a 11 de novembro de 2016
Teresina, Piauí, Brasil

Comitês

Coordenação Geral

Fábio de Jesus Lima Gomes (IFPI) – *Coordenador Geral*
Ricardo de Andrade Lira Rabelo (UFPI) – *Vice-coordenador Geral*
Carlos de Salles Soares Neto (UFMA) – *Coordenador Adjunto*
Roberto Willrich (UFSC) – *Presidente do Comitê de Programa*

Comitê Organizador

Fábio de Jesus Lima Gomes (IFPI)
Ricardo de Andrade Lira Rabelo (IFPI)
Carlos Giovanni Nunes de Carvalho (UESPI)
Harilton da Silva Araújo (Faculdade Estácio de Sá - CEUT)
Rodrigo Augusto Rocha Souza Baluz (UESPI - Parnaíba)

Coordenadores dos Minicursos

Fernando Antonio da Mota Trinta (UFC) – *Coordenador Geral*
Erick Baptista Passos (IFPI) – *Coordenador Local*

Comitê de seleção de minicursos

Adriano Pereira (UFMG)
Alessandra Macedo (USP)
Erick Baptista Passos (IFPI)
Fernando Antonio Mota Trinta (UFC)
Flávio Sousa (UFC)
Marco Cristo (UFAM)
Marcio Maia (UFC)
Mário Teixeira (UFMA)
Renato Bulcão (UFG)
Rudinei Goularte (USP)
Tiago Maritan (UFPB)
Tatiana Tavares (UFPEL)
Vania Vidal (UFC)
Windson Viana (UFC)

Coordenação da Comissão Especial de Sistemas Multimídia e Web

Carlos de Salles Soares Neto (UFMA) – *Coordenador*
Manoel Carvalho Marques Neto (IFBA) – *Vice-coordenador*

Comitê Gestor

Adriano C. Machado Pereira (UFMG)
Celso Alberto Saibel Santos (UFES)
Fábio de Jesus Lima Gomes (IFPI)
Fernando Antonio Mota Trinta (UFC)
Guido Lemos de Souza Filho (UFPB)
José Valdeni de Lima (UFRGS)
Luiz Fernando Gomes Soares (PUC-Rio) – *in memoriam*
Marco Antônio Pinheiro de Cristo (UFAM)
Maria da Graça Campos Pimentel (USP)
Roberto Willrich (UFSC)
Valter Roesler (UFRGS)

Organização

Instituto Federal do Piauí (IFPI)
Universidade Federal do Piauí (UFPI)
Universidade Estadual do Piauí (UESPI)

Promoção

SBC – Sociedade Brasileira de Computação

Patrocinadores

NIC.br – Núcleo de Informação e Coordenação do Ponto BR
CGI.br – Comitê Gestor da Internet no Brasil
CAPES – Coordenação de Aperfeiçoamento de Pessoal de Nível Superior
CNPq – Conselho Nacional de Desenvolvimento Científico e Tecnológico
FAPEPI – Fundação de Amparo à Pesquisa do Estado do Piauí
Infoway Soluções em Informática Ltda

Apoio

Delta TIC's – Instituto de Tecnologias, Inovação e Ciências do Delta
Faculdade Estácio – Centro de Ensino Unificado de Teresina (CEUT)
Instituto Delta – Instituto de Pesquisa, Desenvolvimento e Empreendedorismo em Tecnologia da Informação

WebMedia 2016 é realizado em cooperação com a ACM SIGMM e ACM SIGWEB

ACM SIGMM – Association for Computing Machinery Special Interest Group on Multimedia
ACM SIGWEB – Association for Computing Machinery Special Interest Group on Hypertext and the Web

Prefácio

Uma característica comum a boa parte dos simpósios nacionais organizados pela Sociedade Brasileira de Computação (SBC) é a realização de minicursos de curta duração relacionados aos temas de interesse de cada simpósio. Este também é o caso do XXII Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia 2016), onde os minicursos tem a duração de 4 horas e permitem que ouvintes recebam informações sobre novas tecnologias e tópicos atuais de pesquisa em áreas correlatas ao evento. Com isto, minicursos aparecem como uma excelente oportunidade para familiarização dos congressistas com novos temas de pesquisa que podem vir a ser úteis em suas vidas profissionais.

Para o Webmedia 2016, o processo de seleção de minicursos foi feito a partir de uma chamada pública divulgada na lista eletrônica de emails da SBC, bem como ampla divulgação no site oficial do evento e em redes sociais. Foram recebidas 16 (dezesseis) propostas de minicursos, avaliadas por comitê composto de professores com conhecimento nos temas abordados. Cada minicurso foi avaliado por pelo menos dois avaliadores, que pontuaram notas para quesitos como relevância para o evento, expectativa do público, atualidade e conteúdo de cada minicurso. Ao final, 8 (oito) propostas foram selecionadas, cujos conteúdos abordados constituem os capítulos deste livro.

O primeiro capítulo aborda o tema de documentos multimídia interativos, especialmente nas questões da manutenção da consistência durante o ciclo de vida destes documentos. Os autores apresentam os requisitos impostos por tais documentos para preservar sua consistência, bem como as principais soluções existentes na literatura para atacar o problema. Também são apresentados desafios relevantes de pesquisa, indicando o caminho para futuros trabalhos relacionados ao tema.

O Capítulo dois apresenta os principais conceitos que têm sido geralmente utilizados para avaliação de acessibilidade e usabilidade em Rich Internet Applications (RIAs). É apresentada uma visão geral das perspectivas, desde normas e padrões de qualidade existentes, até quais são recursos existentes para implementação de aplicações Web.

O Capítulo três aborda o tema de *crowdsourcing*, no qual multidões de usuários contribuem para a soluções de problemas de forma colaborativa. Os autores abordam desde conceitos fundamentais ao tema, até cenários propícios para sua utilização, apresentando também casos reais de aplicação da abordagem, especialmente àqueles voltados ao tema de multimídia.

O quarto capítulo é relacionado aos ecossistemas de software (ECOS). São apresentados os principais conceitos e estratégias de ECOS, bem como a organização de mecanismos e ferramentas para modelar e analisar o desenvolvimento de plataformas para web, redes sociais e multimídia. Ao final, casos reais são utilizados para explorar cenários de decisão e recursos apresentados.

Em seguida, o capítulo cinco apresenta o ESPIM (*Experience Sampling and Programmed Intervention Method*), um método que combina procedimentos da psicologia, ESM (*Experience Sampling Method*) e computação ubíqua, para promover melhorias em coleta de dados e intervenções em configurações naturalistas. O minicurso apresenta os conceitos fundamentais da ESM e resultados relevantes da derivação deste método, bem como o sistema ESPIM, suas funcionalidades e detalhes de desenvolvimento.

O capítulo seis explora o tema de Internet das Coisas (IoT), onde propõe-se que di-

versos equipamentos (geladeiras, ar-condicionados, dentre outros) sejam integrados, permitindo a troca de informação e a coordenação de ações de modo a viabilizar a criação de espaços inteligentes como casas, ou mesmo grandes cidades. Os autores apresentam o tema por meio de uma abordagem prática, onde a plataforma Arduíno e o framework Node.js são utilizados para viabilizar cenários de IoT.

O sétimo capítulo discorre sobre computação cognitiva e multimídia, onde as oportunidades de integração entre os dois temas são explorados pelos autores. Computação Gognitiva engloba um conjunto de metodologias, modelos, arquiteturas e algoritmos que permitem a um sistema computacional se comportar e interagir empregando cognição para solução de problemas, tomada de decisões e previsões. O minicurso intitulado *Multimedia for Decision-Making (MM4DM)* debate sobre como a área de multimídia pode tirar proveito dos processos de tomada de decisão na era da computação cognitiva.

Por fim, o oitavo capítulo apresenta uma introdução ao GStreamer, um dos mais conhecidos frameworks para processamento multimídia. O curso utiliza um estudo de caso para abordar de forma prática o uso do GStreamer, sua arquitetura e modelo de programação. Também são apresentados elementos nativos do GStreamer para manipulação de amostras de áudio e vídeo, permitindo que pessoas com conhecimento básico em linguagem C possam criar aplicações simples, bem como explorar características avançadas do framework.

Esperamos que este livro seja útil para todos aqueles interessados e praticantes da área de Sistemas Multimídia e Web.

Teresina, novembro de 2016.

Fernando Antonio Mota Trinta (UFC)
Erick Baptista Passos (IFPI)
Coordenadores dos Minicursos do WebMedia'2016

Sumário

Capítulo 1

Desafios da Verificação de Consistência de Documentos Multimídia Interativos

Joel dos Santos[1 2] e Débora Muchaluat-Saade[2]

[1] Escola de Informática & Computação - CEFET/RJ
[2] Laboratório MídiaCom - Dep. de Ciência da Computação - Universidade Federal Fluminense

Abstract

A multimedia document may be specified manually or automatically generated, instantiated, adapted or has its content and structure dynamically edited. Therefore, changes in a document specification may occur from its creation *to its* execution*. Such different phases and changes performed over a document caracterize its life cycle. It is important to maintain multimedia document consistency along its life cycle, which means that document execution should always follow guidelines expressed at* creation *time. Different works in the literature present approaches for addressing this issue by validating multimedia documents along different phases of a document life cycle. This work discusses challenges and approaches for multimedia document consistency checking along its life cycle. There will be a discussion about key solutions presented in the literature, challenges and directions for future research in this topic.*

Resumo

Um documento multimídia pode ser especificado manualmente ou gerado automaticamente, instanciado, adaptado ou ter seu conteúdo e estrutura dinamicamente editada. Portanto, mudanças na especificação de um documento podem ocorrer desde sua criação *até sua* execução*. Estas diferentes fases e mudanças realizadas em um documento caracterizam o seu ciclo de vida. É importante manter a consistência de um documento multimídia ao longo do seu ciclo de vida, o que significa que a execução de um documento deveria sempre seguir diretrizes expressas em tempo de* criação*. Diferentes trabalhos na literatura apresentam abordagens para resolver esse problema através da validação de documentos multimídia em diferentes fases ao longo do seu ciclo de vida. Este capítulo*

tem como principal objetivo discutir os desafios e abordagens para a verificação de consistência de documentos multimídia ao longo de seu ciclo de vida. Serão discutidas as principais soluções apresentadas na literatura e apresentados os desafios e pontos em aberto que podem nortear pesquisas futuras.

1.1. Introdução

Atualmente, observamos uma grande difusão de diferentes dispositivos portáteis equipados com câmeras, microfones, sensores, displays e com grande capacidade de processamento e armazenamento. Com isso, sistemas multimídia tornam-se cada vez mais presentes em nosso dia-a-dia. Podemos ser tanto produtores de conteúdo multimídia, dada a capacidade de captura de informação de tais dispositivos, quanto consumidores, dada a capacidade de exibição dos mesmos.

Tais conteúdos representam unidades de informação consumidas por seres humanos, como um texto, uma imagem, um áudio, um vídeo e mesmo efeitos sensoriais, como movimento ou vibração. Estas informações, denominadas mídias, são apresentadas de forma organizada no tempo e espaço segundo a descrição contida em um documento multimídia. Um documento multimídia, portanto, descreve o conjunto de mídias que serão por ele consideradas - também chamado de o conteúdo de um documento - e relações entre elas para determinar como tais mídias serão apresentadas no tempo, espaço ou ambos. Relações em um documento podem levar em consideração ocorrências de eventos, como interação do espectador (incluindo reconhecimento de gestos), o resultado de uma computação (por exemplo por um script auxiliar) ou uma *query* (por exemplo a um servidor externo) realizada durante a execução do documento ou mesmo a chegada do espectador a alguma posição geográfica.

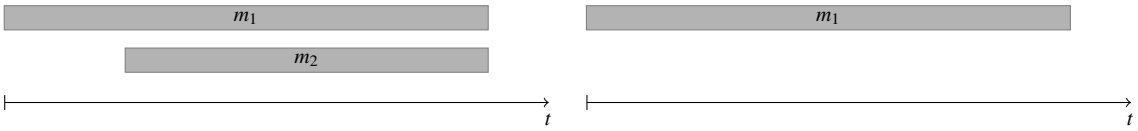
Quando executada, a descrição contida em um documento resulta em um arranjo particular das mídias no tempo e espaço, chamado de uma apresentação multimídia. Diferente de um documento, uma apresentação é o resultado que é efetivamente apresentado ao espectador.

Atualmente, novas facilidades estão disponíveis para a apresentação de documentos multimídia, dentre as quais podemos destacar: a personalização de uma apresentação para um dado espectador [Laborie et al. 2011]; a divisão de uma apresentação em múltiplos dispositivos (por exemplo um aparelho de TV e uma segunda tela) [Sarkis et al. 2014]; novos dispositivos de exibição (como telas *touch* ou com sensores de movimento embutidos) melhorando a interface da apresentação com o espectador e permitindo novas formas de interação (como gestos, *multi touch* e *force touch*); edição dinâmica de uma apresentação (por exemplo através de anotações [Teixeira et al. 2012] ou edição ao vivo [Soares et al. 2012]); e a geração dinâmica do conteúdo de documentos multimídia de acordo com uma busca feita pelo espectador (caso frequente em documentos web).

Tendo em vista tais facilidades, um documento multimídia descreve um conjunto de possíveis diferentes apresentações. Este conjunto é restringido de acordo com o contexto do espectador de forma a escolher a apresentação mais adequada a ele. O contexto do espectador contém informação a respeito do seu dispositivo de exibição, preferências de exibição, e tudo mais que possa influenciar na apresentação do documento. Apesar

de um documento resultar em diferentes apresentações, elas não são completamente diferentes umas das outras, visto que devem seguir uma mesma especificação contida em um documento. Exemplos de adaptação de um documento ao contexto do espectador são ajustar uma apresentação ao tamanho da tela do dispositivo de exibição, a supressão de um áudio da apresentação dada a indisponibilidade de um canal de áudio, etc.

Após um documento ser moldado de acordo com o contexto do espectador, este pode ser ainda modificado de acordo com a interação do espectador, resultando assim na apresentação final, isto é, aquela efetivamente apresentada ao espectador. A interação do espectador pode acontecer, por exemplo, via gestos, reconhecimento de fala, seleção, *multi touch* e *force touch*. Como exemplo, considere um documento onde uma mídia m_2 é apresentada dado que o espectador tenha interagido com a mídia m_1. A Figura 1.1 apresenta duas possíveis apresentações obtidas a partir de tal documento, ambas diferindo pela ocorrência ou não de interação do espectador.



(a) Espectador interagiu com m_1 (b) Espectador não interagiu com m_1

Figura 1.1: Possíveis apresentações de acordo com interação do espectador

A descrição contida em um documento multimídia é usualmente textual, seguindo alguma linguagem de autoria. No paradigma declarativo, linguagens de autoria provêem construções com um alto grau de abstração para declarar o conjunto de mídias em um documento e as relações entre elas. A ideia de uma linguagem declarativa é separar a descrição contida em um documento das especificidades de sua execução. O nível de abstração das construções providas por uma linguagem varia de acordo com o modelo de sincronização por ela seguido. Na Seção 1.2 serão apresentados diferentes modelos de sincronização encontrados na literatura.

Um outro benefício da autoria declarativa, no cenário multimídia, é facilitar a tarefa de criação de documentos multimídia. Tal objetivo é importante visto que documentos multimídia podem ser utilizados em diferentes áreas, como web, TV digital e IPTV; e podem ser criados por autores com diferentes perfis, como desenvolvedores e produtores de conteúdo.

A especificação contida em um documento pode variar ao longo de seu ciclo de vida, isto é, desde sua criação até sua execução. Ela varia de relações de sincronização expressas em uma linguagem de autoria em alto nível de abstração até eventos de execução de baixo nível e modificações incrementais que podem ser feitas sobre um documento. Portanto, é importante garantir que um documento permaneça consistente à sua especificação ao longo do seu ciclo de vida, isto é, às diretrizes expressas durante sua criação.

A consistência de um documento ao longo do seu ciclo de vida pode ser mantida através de validação. A ideia é combinar a especificação de um documento com propriedades representando as diretrizes e validá-las de forma a garantir que qualquer modificação produzida em um dado passo do ciclo de vida não torne o documento inconsistente.

Este capítulo tem por objetivo discutir a verificação de consistência de documentos multimídia declarativos ao longo do seu ciclo de vida, seus desafios e diferentes abordagens. Serão apresentadas as principais soluções encontradas na literatura e apresentados os desafios e pontos em aberto que podem nortear pesquisas futuras.

O restante deste capítulo está estruturado da seguinte forma.

A Seção 1.2 apresenta os diferentes modelos de sincronização espaço-temporal utilizados na autoria de documentos multimídia. Cada modelo define uma abordagem para a especificação da sincronização entre objetos de mídia ao longo da execução de um documento multimídia. Tais abordagens serão usadas para a classificação das soluções sobre verificação de consistência de documentos apresentadas na literatura.

A Seção 1.3 discute as diferentes formas de realizar a validação de documentos multimídia, dentre as quais destacam-se a validação estrutural e comportamental. A primeira visa validar a forma como um documento é descrito e a segunda visa validar o comportamento de um documento quando executado. A validação comportamental, por sua vez, pode ser dividida em três categorias, dependendo do tipo de validação realizada. São elas: a validação temporal, a validação espacial e a validação espaço-temporal.

A Seção 1.4 apresenta as diferentes etapas pelas quais passa um documento, juntamente com a classificação de tais etapas em um ciclo de vida. A apresentação do ciclo de vida é feita juntamente com a discussão das etapas onde a validação de um documento é importante e a apresentação de como padrões e trabalhos publicados na literatura se encaixam nesse ciclo.

A Seção 1.5 apresenta trabalhos publicados na literatura relacionados com a validação de documentos multimídia, juntamente com a indicação da etapa do ciclo de vida onde atuam. Ainda, as soluções apresentadas são classificadas de acordo com o tipo de validação que provêem.

A Seção 1.6 apresenta um exemplo de documento multimídia juntamente com a verificação de sua consistência.

A Seção 1.7 discute os desafios de pesquisa relacionados à validação de documentos multimídia. Esses desafios são problemas em aberto, ainda não tratados pelos trabalhos relacionados e que podem motivar novas pesquisas na área.

Por fim, a Seção 1.8 conclui este capítulo.

1.2. Modelos de Sincronização Temporal

Um modelo de sincronização temporal especifica como são definidas as relações espaço-temporais entre as mídias que compõem um documento. Construções comuns em linguagens declarativas são: nós, fragmentos de nós (âncoras ou simplesmente fragmentos), elos e composições. Nós são entidades que representam mídias em um documento, sendo a representação do seu conteúdo abstraída em sua representação no documento. Um nó pode representar um texto, áudio, vídeo, imagem, script, um programa, etc. O conteúdo de um nó é composto de um conjunto de unidades de informação dependentes do tipo de mídia (quadros de um vídeo, amostras de um áudio, pixels em uma imagem, caracteres em um texto, etc). Um fragmento de um nó representa uma parte, um subconjunto, de seu

conteúdo. Elos, por sua vez, são usados para representar relações entre nós. Finalmente, composições, por sua vez, representam conjuntos de nós e/ou elos ou outras composições, e também são usadas para representar relações entre nós.

Nesta seção, os conceitos acima serão utilizados na descrição das diferentes classes de modelos de sincronização temporal. As seguintes classes serão apresentadas: baseado em eixos temporais, baseado em restrições, baseado em sincronização hierárquica, baseado em especificação formal, baseado em eventos e baseado em scripts.

Para ilustrar os diferentes modelos de sincronização temporal, a Figura 1.2 apresenta um exemplo de organização temporal de um conjunto de mídias. Neste exemplo temos um vídeo, um áudio, uma imagem e um texto relacionados no tempo. Este exemplo foi retirado de [Boll 2001].

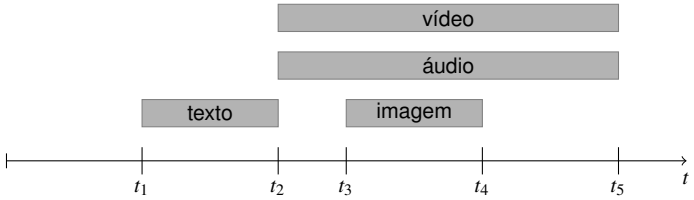


Figura 1.2: Exemplo de cenário temporal

A Figura 1.2 descreve a apresentação de um texto, seguido pela apresentação, em paralelo, de um vídeo e um áudio. Com um atraso de 1 minuto ($t_3 = t_2 + 1$) após o término do texto, uma imagem é apresentada. As seções a seguir apresentam as diferentes classes de modelos de sincronização temporal e como cada modelo representa o exemplo descrito.

1.2.1. Baseado em Eixos Temporais

Modelos de sincronização baseados em eixos temporais, ou baseado em linha de tempo, mapeiam o início ou o fim da apresentação de um nó em um eixo temporal. Este eixo pode ser global, de forma que todo nó é mapeado no mesmo eixo temporal, ou pode ser virtual, de forma que nós podem ser mapeados em diferentes eixos. Exemplos de sistemas que usam este tipo de sincronização temporal são softwares comerciais como Apple iMovie ou Apple Final Cut. A representação do exemplo da Figura 1.2 usando um modelo de sincronização baseado em eixo temporal é apresentada a seguir.

```
1  texto [t1, t2]
2  vídeo [t2, t5]
3  áudio [t2, t5]
4  imagem [t3, t4]
```

1.2.2. Baseado em Restrições

Modelos de sincronização baseados em restrições, definem restrições sobre a ordem temporal de um conjunto de nós. Modelos de sincronização baseados em restrições podem ser divididos em dois tipos: baseados em intervalos e em pontos de referência.

A *sincronização baseada em intervalos* representa a duração de um nó como um intervalo. A sincronização entre dois intervalos é definida usando treze relações básicas

apresentadas por Allen em [Allen 1983], ou sete se as relações inversas forem desconsideradas. Uma outra abordagem é utilizar as vinte e nove relações definidas por Wahl em [Wahl and Rothermel 1994]. A Figura 1.3 apresenta as sete relações de Allen.

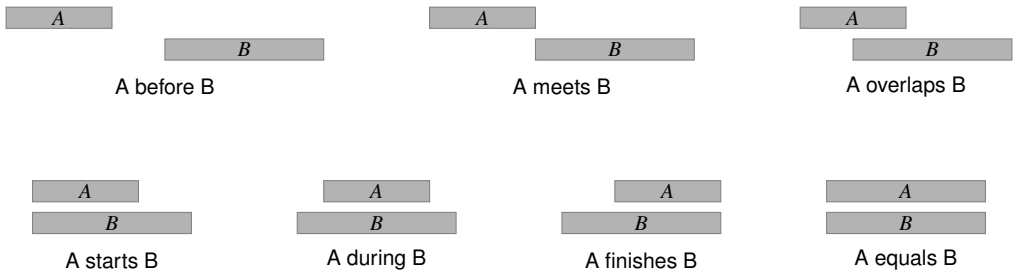


Figura 1.3: Relações de Allen entre intervalos temporais

Um exemplo de sistema que usa este tipo de sincronização temporal é o sistema de autoria Madeus [Jourdan et al. 1998]. A representação do exemplo na Figura 1.2 usando um modelo de sincronização baseado em intervalos é apresentada a seguir, onde assume-se que as mídias texto e imagem possuem duração associada.

```
1  vídeo equals áudio
2  texto meets vídeo
3  texto meets imagem delay 1min
```

Note que, apesar de esse tipo de sincronização prover uma definição simples das relações entre nós (intervalos), um modelo de sincronização baseado em intervalos não permite a especificação de relacionamentos entre fragmentos de um nó. Esse tipo de relacionamento é feito de forma indireta através de atrasos ou subdividindo um nó de acordo com seus fragmentos.

A *sincronização baseada em pontos de referência* usa pontos de referência para definir a sincronização entre nós. Pontos de referência podem ser o início ou fim da apresentação de um nó ou de um de seus fragmentos. O relacionamento entre nós (ou fragmentos) é feito através da especificação de restrições sobre os pontos de referência, criando pontos de sincronização. Nós (ou fragmentos) participando de um mesmo ponto de sincronização são iniciados ou parados de uma única vez, quando o ponto de sincronização é alcançado durante a apresentação do documento. A representação do exemplo na Figura 1.2 usando um modelo de sincronização baseado em pontos de referência é apresentada a seguir. Assim como no caso anterior, assume-se que as mídias texto e imagem possuem duração associada.

```
1  Point t2:
2      texto ends and vídeo starts and áudio starts and imagem starts delay 1 min
3  Point t5:
4      vídeo ends and áudio ends
```

Dado que seja possível a representação de fragmentos de um nó, pontos de referência podem ser criados para definir sincronização para tais fragmentos. Assim, a sincronização baseada em pontos de referência apresenta maior flexibilidade que a baseada em intervalos. Um exemplo de sistema que usa sincronização baseada em pontos de referência é o sistema Firefly [Buchanan and Zellweger 1992, Buchanan and Zellweger 2005].

1.2.3. Baseado em Sincronização Hierárquica

Modelos baseados em sincronização hierárquica descrevem a sincronização entre nós usando duas composições temporais principais: composição sequencial e paralela. Nessa abordagem, a sincronização do documento é definida por uma árvore de composições temporais, representando a apresentação sequencial ou paralela do seus nós internos. É possível ainda a definição de um atraso para o início de um nó específico. Exemplos de uso de modelo baseado em sincronização hierárquica são as linguagens declarativas SMIL (Synchronized Multimedia Integration Language) [W3C 2008b] e MPEG-4 XMT (eXtensible MPEG-4 Textual) [ISO/IEC 2005]. A representação do exemplo na Figura 1.2 usando um modelo baseado em sincronização hierárquica é apresentada a seguir.

```
1  seq{
2      texto
3      par{
4          vídeo
5          áudio
6          imagem begin 1min
7      }
8  }
```

Um modelo baseado em sincronização hierárquica provê uma abordagem simples para a sincronização de nós. Entretanto, assim como na sincronização baseada em intervalos, não é possível a criação de relacionamentos entre fragmentos dos nós, sendo, este tipo de sincronização, feito de forma indireta.

1.2.4. Baseado em Especificação Formal

Modelos de sincronização baseados em especificação formal são aqueles que usam algum tipo de formalismo para representar nós e especificar a sincronização entre eles. Esse tipo de sincronização se beneficia do grande número de ferramentas para modelos formais disponíveis para apresentar o documento descrito. Dois tipos comuns de modelos de sincronização baseados em especificação formal são: sincronização baseada em redes de Petri temporizadas [Peterson 1981] e baseada em *statecharts* [Harel 1987].

A *sincronização baseada em redes de Petri temporizadas* usa redes de Petri, onde lugares possuem duração e representam os nós do documento e transições são utilizadas para a sincronização entre lugares. Apesar de permitir a definição de qualquer tipo de relação de sincronização, esse tipo de modelo também não permite a criação de relacionamentos entre fragmentos dos nós. Exemplos de sistemas que utilizam sincronização baseada em redes de Petri são os sistemas Trellis [Furuta and Stotts 2001], caT (*Context-aware Trellis*) [Na and Furuta 2001] e HTSPN (*Hierarchical Time Stream Petri Net*) [Willrich et al. 2001]. Uma representação do exemplo na Figura 1.2 usando um modelo de sincronização baseado em redes de Petri é apresentada na Figura 1.4.

A *sincronização baseada em statecharts* utiliza statecharts para representar os nós e sua sincronização temporal. Nessa abordagem estados representam mídias, enquanto transições definem as relações de sincronização entre estados. Uma transição ainda define se estados filhos são apresentados simultaneamente ou não. Um exemplo de sincronização baseada em statecharts é apresentada em [de Oliveira et al. 2001]. A representação do exemplo na Figura 1.2 usando um modelo de sincronização baseado em statecharts é apresentada na Figura 1.5.

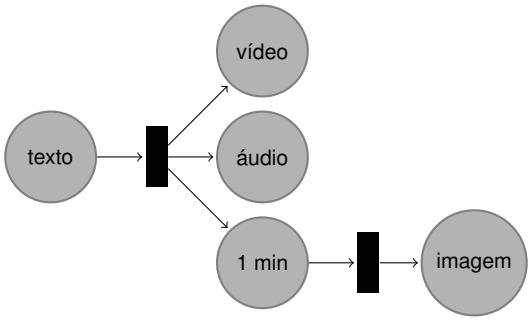


Figura 1.4: Representação baseada em Redes de Petri

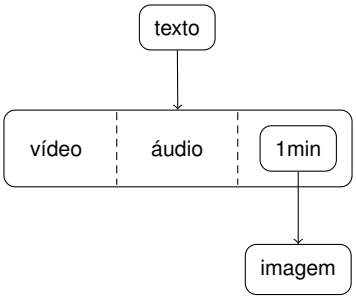


Figura 1.5: Representação baseada em Statecharts

Na Figura 1.5, os estados do *statechart* representam nós do documento. Ainda, um estado é utilizado para representar o atraso de 1 minuto para o início da imagem. Neste exemplo, a composição de estados, como visto para o vídeo, o áudio e o atraso de 1 minuto, indica que tais mídias (e atraso) iniciarão sua execução em paralelo. As transições entre estados indicam a sequência de execução das mídias representadas em cada estado.

1.2.5. Baseado em Eventos

Modelos de sincronização baseados em eventos permitem a criação de relacionamentos entre eventos que acontecem durante a execução do documento, como eventos de apresentação do conteúdo de um nó, eventos de seleção (interação do espectador, por exemplo) ou eventos de atribuição (mudança do valor de uma variável, por exemplo). Exemplos de modelos baseados em eventos são os modelos NCM (*Nested Context Model*) [Soares et al. 2000], que foi usado como base para a linguagem de autoria NCL (*Nested Context Language*) [ITU 2009], e Labyrinth [Díaz et al. 2001]. A representação do exemplo na Figura 1.2 usando um modelo de sincronização baseado em eventos é apresentada a seguir, onde assume-se que as mídias texto e imagem possuem duração associada.

```
1 onBegin document, start texto
2 onEnd texto, start vídeo and start áudio and start imagem delay 1min
3 onEnd video, end áudio
```

1.2.6. Baseado em Scripts

Modelos de sincronização baseados em scripts usam uma descrição textual para a sincronização dos nós. Um script (descrição textual) define um conjunto de passos para a

obtenção da apresentação desejada. Exemplos de uso de modelo de sincronização baseado em scripts são a linguagem HTML5 [W3C 2014], onde a sincronização entre mídias é definida através de scripts JavaScript, e a linguagem Flash [Adobe 2010]. A representação do exemplo na Figura 1.2 usando um modelo de sincronização baseado em scripts é apresentada a seguir. No exemplo, os atrasos são especificados em milissegundos.

```
1   function imagemEnd() {
2       imagem.stop();
3   }
4
5   function imagemStart() {
6       imagem.start();
7       setInterval(imagemEnd, (t4−t3)∗60000);
8   }
9
10  function vídeoEnd() {
11      vídeo.stop();
12      áudio.stop();
13  }
14
15  function textoEnd() {
16      texto.stop();
17      vídeo.start();
18      áudio.start();
19      setInterval(imagemStart, 60000);
20      setInterval(vídeoEnd, (t5−t2)∗60000);
21  }
22
23  function textoStart() {
24      texto.start();
25      setInterval(textoEnd, (t2−t1)∗60000);
26  }
27
28  textoStart();
```

Um modelo de sincronização é dito mais expressivo que um outro modelo, quando é capaz de representar apresentações mais elaboradas que o outro modelo. Scripts são muito expressivos, entretanto, uma desvantagem do uso de scripts é o fato de o autor ter de definir todos os detalhes para a sincronização dos nós através de programação, o que já é abstraído em uma abordagem declarativa.

No paradigma de programação declarativo, modelos de sincronização baseados em eventos são uma abordagem bastante expressiva, além de apresentarem um alto nível de abstração, para a autoria de documentos multimídia. Através de eventos, é possível descrever as mesmas apresentações passíveis de serem descritas com outros modelos [Blakowski and Steinmetz 1996]. Uma exceção, entretanto, é o caso de modelos baseados em restrições. Dado que, em geral, modelos de sincronização baseados em eventos definem relações causais entre eventos, a representação de restrições entre mídias, quando possível, é complexa de ser feita usando somente eventos.

Em geral, a validação de modelos de sincronização baseados em restrições é feita sobre o próprio conjunto de restrições. Por outro lado, a validação de modelos baseados em eventos (e consequentemente nos demais) é feita sobre uma representação dos estados do documento ao longo de sua execução. Tal classificação é explorada na Seção 1.5 ao apresentar as soluções disponíveis na literatura.

Nesse contexto, um modelo híbrido, isto é, suportando eventos e restrições, surge como uma solução interessante a ser explorada, possibilitando a validação de uma grande

gama de documentos multimídia declarativos.

1.3. Tipos de Validação de Documentos Multimídia

Como discutido na Seção 1.1, um documento multimídia descreve um conjunto de possíveis apresentações. Posteriormente, durante sua execução, uma única apresentação é escolhida, conforme o contexto do espectador. A apresentação resultante, portanto, representa a execução de um dado documento.

Considerando tanto sua criação quanto sua execução, um documento pode ser considerado de dois pontos de vista distintos, um representando sua especificação e outro representando sua execução. No primeiro, são consideradas as construções em uma linguagem de autoria específica utilizadas na criação de um documento. No segundo, são considerados os comportamentos resultantes de tais construções quando o documento é executado.

Da maneira análoga, inconsistências podem ser consideradas de dois pontos de vista distintos, quando certas propriedades sobre um documento não são satisfeitas. Tais propriedades são classificadas em *estruturais* e *comportamentais*. As seções a seguir discutem tais propriedades e seu uso para a validação de um documento.

1.3.1. Propriedades Estruturais

Desde sua criação, a linguagem XML (*Extensible Markup Language*) [W3C 2008a] tem sido utilizada como uma sintaxe concreta para linguagens multimídia. Elementos XML são usados para representar mídias e relações entre elas. Exemplos de linguagens declarativas baseadas em XML são HTML5 [W3C 2014], NCL [ITU 2009] e SMIL [W3C 2008b]. NCL é parte do padrão brasileiro de TV digital [ABNT 2011] e padrão ITU para serviços IPTV [ITU 2009]. SMIL é padrão para apresentações multimídia na web [W3C 2008b].

Regras sintáticas definidas pela gramática de uma linguagem de autoria multimídia induzem um conjunto de propriedades estruturais que um documento deve seguir, de forma a ser considerado estruturalmente consistente. Trabalhos publicados na literatura [Araújo et al. 2008, Neto et al. 2011] identificam as seguintes propriedades estruturais:

- A estrutura léxica e sintática de um documento deve ser bem formada e estar de acordo com a gramática da linguagem de autoria utilizada. Por exemplo, quando uma linguagem baseada em XML é utilizada, as *tags* XML [W3C 2008a] devem ser corretamente fechadas e estarem presentes no espaço de nomes (*namespace*) da linguagem em uso.
- Todo elemento em um documento deve conter somente elementos filhos válidos e com a cardinalidade correta.
- Todo elemento em um documento deve conter somente atributos válidos, sendo que os atributos obrigatórios devem estar definidos.
- Todo identificador, quando for o caso, deve ser único.
- Atributos com valores relacionados devem seguir as restrições definidas pela linguagem de autoria. Por exemplo, NCL define os atributos *type* e *subtype* para o

elemento *transition* [ITU 2009]. Para cada diferente valor do atributo *type*, NCL define um conjunto de valores que o atributo *subtype* pode assumir. Um outro exemplo em NCL são os atributos *component* e *interface*, onde um dado elemento referenciado pelo atributo *interface* deve ser um elemento filho daquele referenciado pelo atributo *component*.

- Referências entre elementos devem seguir as restrições definidas pela linguagem de autoria. Por exemplo, NCL define atributos que podem referenciar somente um tipo ou um grupo específico de elementos. No elemento *media*, o atributo *descriptor* pode referenciar apenas elementos *descriptor*, enquanto o atributo *refer* pode referenciar apenas outros elementos *media*.

- Elementos dentro de uma composição não podem referenciar elementos fora da mesma composição. Por exemplo, em SMIL a composição *par* define o atributo *endsync*, o qual pode fazer referência a um componente interno a composição, determinando que toda a composição irá terminar sua apresentação quando o componente referenciado terminar. Neste caso, o atributo *endsync* de uma composição não pode fazer referência a um componente externo a ela.

- Uma composição não pode criar um *loop* de aninhamento, isto é, uma composição não pode conter a si mesma diretamente ou através de outras composições. NCM [Soares and Rodrigues 2005] e caT [Na and Furuta 2001] permitem o reúso de composições (contextos e switches em NCM e lugares em caT). Em ambos, composições não podem aninhar a si mesmas, caso contrário o documento multimídia seria inconsistente.

- Um elemento não pode criar um *loop* de reúso, isto é, um elemento não pode reusar a si mesmo diretamente ou através de outros elementos. Por exemplo, um elemento NCL não pode fazer referência, através de seu atributo *refer* a si mesmo.

Inconsistências estruturais surgem quando um documento não segue as propriedades estruturais. A validação estrutural de um documento, portanto, é importante, visto que uma ferramenta de exibição de um documento multimídia pode não conseguir ler ou mesmo executar um documento com inconsistências estruturais.

1.3.2. Propriedades Comportamentais

Propriedades comportamentais são usadas para verificar a existência de inconsistências na apresentação resultante da execução de um documento. O conjunto de relações definido em um documento é comumente separado em dois eixos, o espacial e o temporal. Chamaremos o conjunto de relações em cada eixo como o leiaute do documento. Desta forma, dizemos que um documento define um leiaute temporal e um leiaute espacial.

No leiaute temporal de um documento, mídias são organizadas no tempo seja através de valores absolutos ou em relação a outras mídias ou ocorrências de eventos, como interação do espectador. No leiaute espacial, mídias são organizadas comumente em relação à tela do dispositivo de exibição, a outras mídias ou em canais predefinidos. É possível ainda que relações espaciais definam o posicionamento de uma mídia em relação a uma ou mais mídias declaradas no documento.

É usual que mídias sejam posicionadas relativamente à tela, em valores absolutos (*pixels*) ou relativos (porcentagem). Apesar da posição inicial das mídias ser definida de forma estática, é possível que seu posicionamento seja alterado ao longo da execução. Algumas linguagens de autoria multimídia permitem a mudança da posição de mídias em resposta à ocorrência de eventos na apresentação do documento. Tais mudanças podem considerar o movimento de mídias, por exemplo modificando seus atributos *left*/*top*, e/ou o redimensionamento de mídias, por exemplo modificando seus atributos *width*/*height*. A modificação de atributos espaciais de uma mídia pode ser instantânea ou incremental ao longo de um intervalo temporal.

É possível que o leiaute espaço-temporal resultante da execução de um documento não represente a expectativa do autor ao criar um documento. Tal quebra de expectativa pode ocorrer pelo uso indevido de construções da linguagem de autoria utilizada ou pela falta de relações no documento criado. Diretrizes do autor, portanto, representam comportamentos esperados em um documento e devem ser definidas junto da criação do documento. Tais diretrizes têm por objetivo evitar incompatibilidades entre "o que o autor quer" e "o que o autor percebe". Trabalhos publicados na literatura [Santos et al. 1998, de Oliveira et al. 2001, Júnior et al. 2012, dos Santos 2016] identificam as seguintes diretrizes gerais:

- Toda mídia de um documento deve ser apresentada durante a execução do documento.
- Uma mídia de um documento deve terminar sua apresentação.
- A execução de um documento como um todo deve terminar. A execução do documento termina se a apresentação de todos as mídias do documento terminam e não há *loops* de execução (por exemplo, uma mídia reiniciando sua apresentação toda vez que termina).
- Duas mídias distintas não devem usar um mesmo recurso de apresentação (posição na tela ou um canal de áudio, por exemplo) simultaneamente, evitando sua sobreposição.

Apesar de diferentes trabalhos publicados na literatura identificarem tais comportamentos como esperados [Santos et al. 1998, de Oliveira et al. 2001, Júnior et al. 2012, dos Santos 2016], eles podem não condizer com a intenção do autor. Por exemplo, se um documento representa um jogo, o autor pode querer que este nunca termine sua execução. Entretanto, se pensarmos em um documento representando um comercial interativo, este deverá terminar obedecendo o tempo limite a ele alocado pela emissora.

As diretrizes acima citadas representam diretrizes gerais a serem aplicadas a qualquer documento. Ainda, o autor de um documento pode definir diretrizes específicas para o documento sendo criado. Por exemplo, ele(a) pode definir que duas mídias A e B devem ser apresentadas uma após a outra durante a apresentação do documento. Tais diretrizes são identificadas como diretrizes definidas pelo autor [dos Santos 2012, dos Santos et al. 2013, dos Santos 2016].

Durante a criação de um documento, usualmente o autor de um documento tenta verificar se um documento segue suas diretrizes através de simulação e/ou execução do documento. Neste processo, o autor executa um documento diversas vezes, assumindo o papel do espectador, e observa o resultado obtido em cada execução, verificando se o mesmo se adequa ou não às suas expectativas. Esse processo, entretanto, usualmente não é *efetivo*, já que diversas execuções seriam necessárias para a verificação de comportamentos indesejáveis, e pode ser *incompleta*, já que um documento pode possuir infinitas diferentes execuções. Ainda, no caso onde um documento é gerado automaticamente dentro de uma cadeia de produção, a simulação de um documento, caso possível, pode ser muito complexa de ser realizada.

1.3.3. Validação Estrutural e Comportamental

Inconsistências estruturais surgem quando um documento não segue as propriedades apresentadas na Seção 1.3.1. Por outro lado, inconsistências comportamentais surgem quando a combinação das relações declaradas em um documento e/ou diretrizes do autor é inconsistente, como apresentado na Seção 1.3.2.

A validação estrutural de um documento multimídia é importante uma vez que um documento estruturalmente inconsistente pode não conseguir ser executado ou mesmo lido. Ainda, a validação estrutural deve ser feita antes da validação comportamental, uma vez que inconsistências estruturais podem impossibilitar a validação comportamental do documento.

O tipo de validação comportamental provido depende do tipo de relações e propriedades suportadas pela ferramenta de validação. Nesta seção, para ilustrar os diferentes tipos de validação comportamental, serão apresentados dois pequenos exemplos de um documento d que apresenta duas mídias A e B. Para cada exemplo, as posições de A e B são definidas em valores absolutos. Cada figura representa um exemplo, onde retângulos tracejados representam a tela do dispositivo de exibição em um dado momento e retângulos sólidos representam a região da tela onde uma mídia é apresentada. Setas entre duas telas representam um salto no tempo e setas entre regiões representam um movimento no espaço. No caso de o movimento ser contínuo em um intervalo, a duração do intervalo é indicada junto a seta.

1.3.3.1. Caso 1

O exemplo do Caso 1 descreve um leiaute espacial estático, isto é, A e B não mudam sua posição ao longo do tempo. Este caso é um exemplo puramente temporal, onde A é apresentada (imediatamente) antes de B no tempo. A Figura 1.6 apresenta o leiaute espaço-temporal percebido pelo espectador do documento d.

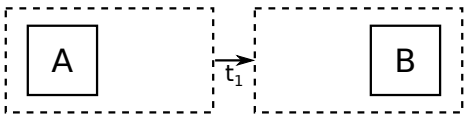


Figura 1.6: Leiaute espaço-temporal do caso 1

Considerando o exemplo acima, a maior parte das diretrizes do autor podem ser descritas através de propriedades temporais. Por exemplo, o autor pode querer garantir que, para este documento, A seja apresentado antes de B, o que pode ser representado pela fórmula A *before* B. Ainda, o autor pode querer garantir que as mídias A e B não sejam apresentadas ao mesmo tempo, o que pode ser representado pela fórmula $not(A$ *together* $B)$.

Apesar de ser um exemplo puramente temporal, o autor pode ainda expressar propriedades espaciais. Por exemplo, o autor pode querer garantir que A e B sejam apresentados em posições laterais, representado pela fórmula A *sideof* B; ou que ambas tenham o mesmo tamanho, representado pela fórmula A *samesize* B. Dado que o leiaute espacial desse exemplo é estático, a validação espacial pode ser feita sobre a posição inicial das mídias do documento.

1.3.3.2. Caso 2

O exemplo do Caso 2 envolve a mudança ao longo do tempo do leiaute espacial de um documento multimídia. Neste exemplo, A tem uma posição fixa e B move-se ao longo da tela, mudando sua posição de maneira incremental ao longo de t_1 unidades de tempo. A Figura 1.7 apresenta o leiaute espaço-temporal percebido pelo espectador do documento d.

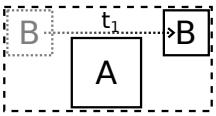


Figura 1.7: Leiaute espaço-temporal do caso 2

A validação do leiaute espaço-temporal do exemplo acima não é tão simples quanto validar o leiaute espacial e, em paralelo, validar o leiaute temporal. Por exemplo, suponha que o autor deseje garantir que, em algum ponto, enquanto se move pela tela, B irá se sobrepor a A. Tal verificação requer verificar se, em pelo menos um momento durante a execução do documento, A e B irão se sobrepor. Tal propriedade deve ser codificada pela composição de propriedades temporais e espaciais, tal como *somepoint*$(A$ *overlap* $B)$.

Nos exemplos apresentados anteriormente, (i) o leiaute espacial e temporal são independentes ou (ii) o leiaute espacial é dependente do temporal. Como visto acima, o exemplo apresentado no segundo caso é chamado de um leiaute espaço-temporal.

A validação puramente temporal atua somente no eixo temporal, verificando a consistência das relações temporais entre mídias. Por outro lado, a validação puramente espacial atua somente no eixo espacial, verificando a consistência das relações espaciais entre mídias. A validação espaço-temporal atua em ambos os eixos simultaneamente, considerando o caso onde mídias mudam sua posição ao longo do tempo.

1.4. Ciclo de Vida de Documentos Multimídia

A especificação contida em um documento multimídia pode variar desde sua criação até sua execução. Tal especificação pode variar de construções em um alto nível de abstração, seguindo uma linguagem de autoria, até eventos de execução em baixo nível de abstração. Diferentes trabalhos relacionados a documentos multimídia publicados na literatura comumente assumem um ciclo de vida composto pelas etapas de *concepção*, *armazenamento* e *execução* (trabalhos relacionados à autoria de documentos, como [Muchaluat-Saade 2003, Bulterman et al. 2013]) ou composto pelas etapas de *concepção*, *adaptação* e *execução* (trabalhos relacionados à adaptação de documentos, como [Lemlouma and Layaïda 2004, Na and Furuta 2001]).

Com base em tais abordagens, foi definido o ciclo de vida [dos Santos 2016] descrito na Figura 1.8. O ciclo apresentado é uma generalização de tais abordagens visando a identificação dos passos onde a validação de um documento é necessária. Ele foi projetado de modo que abordagens como [Sarkis et al. 2014, Lemlouma and Layaïda 2004, Laborie et al. 2011, Teixeira et al. 2012, Soares et al. 2012] possam ser representadas. Tais abordagens representam novas facilidades de apresentação de documentos multimídia, como a personalização da apresentação para um dado espectador [Lemlouma and Layaïda 2004, Laborie et al. 2011], a divisão da apresentação em múltiplos dispositivos [Sarkis et al. 2014] e a edição dinâmica da apresentação (por exemplo via anotação [Teixeira et al. 2012] ou edição ao vivo [Soares et al. 2012]).

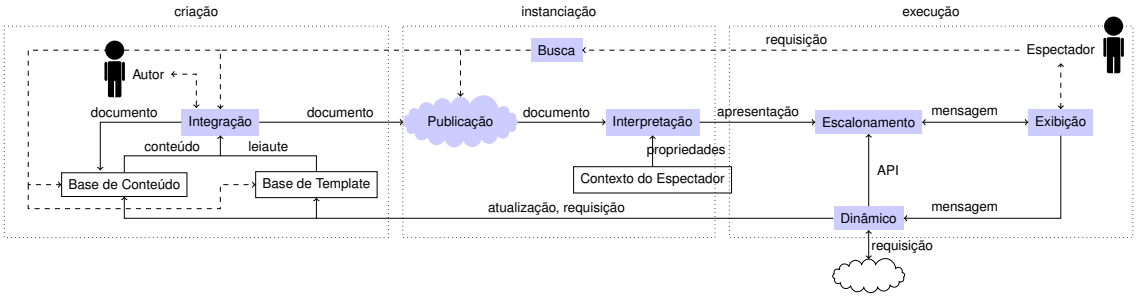


Figura 1.8: Ciclo de Vida de Documentos Multimídia

De acordo com a Figura 1.8, o ciclo de vida de um documento multimídia é composto de três fases principais: *criação*, *instanciação* e *execução*. Tais fases são representadas por retângulos pontilhados.

Cada fase é subdividida em passos (*Integração*, *Publicação*, *Interpretação*, *Escalonamento*, *Exibição*, *Dinâmico*) representados por retângulos pintados ou por uma nuvem pintada (como é o caso da *Publicação*). Armazenamento de dados é representado por retângulos vazios (como é o caso da *Base de Conteúdo*, *Base de Template* e *Contexto do Espectador*). Fluxos de dados são representados por linhas sólidas, enquanto linhas tracejadas representam uma informação sendo trocada com o *Espectador* ou com o *Autor*.

Referente à fase de *criação*, é importante ressaltar três aspectos. (i) Documentos são também considerados como conteúdo, isto é, podem também ser armazenados na base de conteúdo e usados na criação de novos documentos. (ii) Um template é considerado

como um leiaute genérico que é instanciado no passo de *Integração* para um determinado conteúdo. Assim, um template pode representar desde construções em uma determinada linguagem de autoria até especificações parciais de um documento que devem ser preenchidas com mídias específicas. (iii) Um documento pode ser criado tanto pelo autor quanto por um processo de geração automática. No primeiro caso, é possível que o autor use templates para facilitar o processo de criação de um documento, como discutido em [Damasceno et al. 2014]. No segundo caso, documentos são criados de acordo com uma requisição feita pelo espectador, combinando conteúdos existentes com templates predefinidos. É importante ressaltar que ainda assim o autor tem um importante papel na criação de tais templates.

De acordo com uma requisição feita pelo *Espectador*, no passo de *Integração* serão tomadas uma das seguintes ações: (i) serão organizados conteúdos disponíveis na *Base de Conteúdo* de acordo com o leiaute descrito em um template na *Base de Templates* ou (ii) se a requisição resultar em um documento já armazenado na *Base de Conteúdo*, o passo de *Integração* irá repassar tal documento adiante sem criar um novo. É importante notar que quando um novo documento é criado no passo de *Integração*, o documento resultante pode ser tanto guardado na *Base de Conteúdo* para uso futuro quanto usado como entrada para o passo de *Publicação*.

O passo de *Publicação* representa a transmissão[1] do documento para o espectador, de acordo com a requisição recebida. Um documento publicado é instanciado, no passo de *Interpretação*, em uma apresentação de acordo com o contexto do espectador. O contexto do espectador é composto de propriedades expressando (mas não limitadas a): (i) informação sobre o espectador, como idioma, localização, gênero, idade, etc; (ii) características do(s) dispositivo(s) de exibição do espectador, como tamanho de tela, tipos de mídias cuja exibição é suportada, disponibilidade de rede, etc; e (iii) preferências do usuário, como predisposição à interação, tipos de mídias preferidas, etc. Portanto, o documento executado deveria ser o mais adequado às características e preferências do espectador e dispositivos de exibição disponíveis.

O passo de *Escalonamento* mantém a sincronização da apresentação, exibindo cada conteúdo no tempo, espaço e dispositivo correto. O passo de *Exibição* atua como uma interface entre a apresentação e o espectador, repassando eventos de interação do espectador de volta para o passo de *Escalonamento*. Durante a execução, em relação a ocorrências de eventos (como por exemplo interação do espectador), o passo *Dinâmico* acessa APIs disponíveis no passo de *Escalonamento* para edição da apresentação. Exemplos de tais APIs são a API DOM (Document Object Model) [W3C 2000], a API SMIL DOM [W3C 1999], os Comandos de Edição ao Vivo de NCL [Soares et al. 2006] e a API Web Animations [W3C 2014]. Este passo pode ainda trocar informações com outros pares na rede, bem como armazenar novos conteúdos na *Base de Conteúdo* e novos templates na *Base de Templates*, ou requisitar tais informações destas bases.

[1]É por isso que o passo de *Publicação* é representado por uma nuvem, diferentemente dos demais passos.

1.4.1. Casos de Uso

Usualmente, ao se referir ao ciclo de vida de documentos multimídia, a literatura o divide em dois lados: o lado *servidor/difusor* e o lado *cliente/espectador*. No lado *cliente* (ou *espectador*), ficam os passos relacionados à execução do documento. O cliente pode ser tão complexo quanto possível - por outro lado tão simples quanto possível - relativo aos passos por ele realizados. O lado *servidor* (ou *difusor*) fica com os passos restantes. Quanto mais simples é o cliente, mais complexo é o servidor e vice-versa.

Uma divisão comum em ambientes de TV digital e IPTV [ETSI 2008, ATSC 2009, ARIB 2014, ITU 2009] e ainda na web, é usar o passo de *Publicação* como a divisão entre os lados servidor e cliente. O trabalho apresentado em [Lemlouma and Layaïda 2004] apresenta uma divisão diferente, seguindo a ideia de um cliente mais simples, onde somente o passo de *Exibição* está no cliente. Em tal cenário, um documento é criado e sua execução é tratada pelo servidor. O cliente recebe amostras das mídias a serem apresentadas, para um dado momento, e comunica eventos da apresentação e interação do espectador de volta para o servidor.

Esta seção discute trabalhos que implementam partes do ciclo de vida apresentado na Figura 1.8. Os trabalhos apresentados cobrem diferentes áreas de pesquisa em multimídia e inspiraram o ciclo de vida aqui apresentado. Os parágrafos a seguir apresentam tais trabalhos junto com uma discussão sobre os passos que eles implementam.

O trabalho em [Sarkis et al. 2014] apresenta uma abordagem para a divisão de uma aplicação multimídia em duas telas. Este trabalho se encaixa no ciclo de vida apresentado na Figura 1.8 como segue. No passo de *Integração*, elementos do documento são anotados com informação sobre a tela onde serão apresentados, de acordo com a requisição recebida. O documento anotado é enviado, no passo de *Publicação*, para ambos os clientes identificados na requisição recebida. A seguir, na etapa de *Interpretação*, um dado cliente divide o documento mantendo somente os elementos a serem por ele apresentados visíveis e "escondendo" os demais. Durante a execução, o passo *Dinâmico* recebe a interação do espectador com a apresentação e mantém comunicação entre ambas as telas de forma que o documento como um todo mantenha sua sincronização.

O trabalho apresentado em [Laborie et al. 2011] propõe uma abordagem para a adaptação no cliente de documentos multimídia. Tal abordagem implementa o passo de *Interpretação* de forma que um dado documento seja traduzido em um modelo abstrato, como um conjunto de objetos e relações espaço-temporais entre eles. Relações neste modelo são representadas por restrições. Além das relações, restrições são também utilizadas para representar diretrizes do autor. O processo de adaptação combina restrições do documento com outras representando o *Contexto do Espectador*. Dado que o conjunto de restrições resultante desta combinação seja satisfatível, o documento pode ser adaptado sem precisar de modificações. Caso contrário, relações no documento devem ser modificadas até que o conjunto de restrições seja satisfatível.

Ambos os trabalhos apresentados em [Teixeira et al. 2012] e [Soares et al. 2012] atuam no passo *Dinâmico*. O trabalho em [Teixeira et al. 2012] propõe uma abordagem para a anotação de documentos e o trabalho em [Soares et al. 2012] propõe uma abordagem para a adaptação no lado cliente de um documento.

Em [Teixeira et al. 2012], o passo *Dinâmico* é implementado como uma extensão do middleware cliente de forma a permitir ao espectador inserir texto, som ou imagens em uma apresentação. A abordagem proposta captura eventos de interação do espectador para criar anotações sobre o conteúdo sendo apresentado. As anotações produzidas são sincronizadas com o conteúdo principal sendo apresentado através de um documento NCL. Apesar de não ser explorado no artigo, o documento resultante, isto é, contendo as anotações criadas pelo espectador, pode ser enviado de volta para a *Base de Conteúdo* para que seja provido para outros espectadores em futuras visualizações do mesmo conteúdo ou mesmo compartilhado com outros espectadores assistindo ao mesmo conteúdo.

Em [Soares et al. 2012], o passo *Dinâmico* é implementado usando um script Lua que, reagindo a eventos de interação do espectador, dispara requisições para a *Base de Conteúdo* de forma a obter novos vídeos relacionados ao conteúdo sendo apresentado em um dado momento. Tal abordagem cria um novo documento com o novo conteúdo a ser incluído na apresentação, comparando a nova e a atual versão da apresentação para determinar as mudanças a serem realizadas visando atualizar a apresentação. Em seguida são enviados comandos de edição ao vivo de NCL [Soares et al. 2006] para realizar as modificação ao longo da execução.

1.4.2. Validação ao Longo do Ciclo

Visando manter a consistência de um documento ao longo do seu ciclo de vida, a validação é importante em diferentes etapas.

Inicialmente, é importante garantir que conteúdos e template(s) combinados no passo de *Integração* resultem em um documento consistente. Isto pode ser obtido realizando a validação do documento no passo de *Integração*. Tal validação leva em consideração propriedades associadas aos conteúdos e templates utilizados, assim como propriedades relacionadas ao documento sendo criado. Estas propriedades relacionadas ao documento definem o que nos referimos como diretrizes do autor. Validação no passo de *Integração* deve ser capaz de suportar diferentes paradigmas de autoria de documentos multimídia (relações, eventos, composições, etc). É importante mencionar, entretanto, que algumas linguagens multimídia declarativas provêem a possibilidade de se utilizar linguagens de script auxiliares (como Lua junto com NCL e JavaScript junto com HTML) para implementar facilidades não suportadas pela linguagem em uso. A validação de tais scripts, entretanto, foge ao escopo da validação da linguagem declarativa em uso.

De acordo com o paradigma utilizado no passo de *Integração* é possível validar o documento ao mesmo tempo em que ele é descrito. Por exemplo, em uma abordagem baseada em restrições, como em [Jourdan et al. 1998], relações podem ser validadas enquanto são criadas. Por outro lado, em uma abordagem baseada em eventos, tal como [dos Santos et al. 2015], a validação pode ter de esperar a criação de um conjunto de relações antes de ser realizada. Nesse ponto do ciclo de vida do documento, a validação tem um papel *preventivo*, investigando se um documento pode ser instanciado e, uma vez apresentado, não leva a erros de execução. No caso de um erro ser encontrado, a ferramenta de validação pode tanto (i) "corrigir" automaticamente o documento ou (ii) dar um *feedback* ao autor para que este possa "corrigir" o documento sendo criado. Ainda, a validação neste ponto também é *parcial*, visto que o conteúdo (ou mesmo o leiaute) de

um documento pode mudar dinamicamente durante execução.

Após a fase de *criação*, a validação de um documento ao longo do seu ciclo de vida deve sempre retornar uma apresentação executável, ainda que através de alguma forma automática de "correção de erro". Erros após a fase de criação podem interromper a apresentação do documento requisitado. Apesar de *feedbacks* serem importantes para que o autor possa corrigir novas versões do documento, tal prática não ajuda na correção do documento sendo apresentado. Ainda, o tempo de resposta da validação tem um papel importante, para evitar que o espectador espere um longo tempo para começar a apresentação do documento (*Interpretação*) ou trave durante sua execução (*Escalonamento*).

Durante o processo de instanciação no passo de *Interpretação*, é importante garantir que um documento criado possa ser instanciado em uma apresentação levando em conta o contexto do espectador. Isso pode ser realizado validando a combinação tanto de propriedades associadas ao documento quanto outras expressando o contexto do espectador. De acordo com a forma como o documento é descrito e dado que mais de uma apresentação pode ser instanciada a partir de um documento, a validação do documento pode ser vista como uma forma de prover personalização e ajudar a adaptação do documento, escolhendo a instância de apresentação mais adequada ao contexto do espectador. Ainda, dado que tanto a adaptação quando a validação podem ser realizadas usando as mesmas técnicas, nos referimos a adaptação de um documento como um caso especial de validação.

Uma vez que um documento esteja sendo executado, ele pode evoluir seja por interação do espectador quanto por mudanças incrementais feitas no passo *Dinâmico*. Similar à escrita colaborativa [Sun et al. 1998], nesse ponto, é importante prover algum tipo de *preservação de intenção*, para evitar que a apresentação se torne incoerente em relação às diretrizes do autor. Da mesma forma como feito no passo de *Interpretação*, na etapa de edição *Dinâmica*, a validação deve ser capaz de combinar propriedades representando características da apresentação com outras representando as mudanças incrementais. Tal validação deve sempre resultar em uma apresentação executável, caso contrário, uma mudança incremental deve ser ignorada para evitar que a apresentação se torne inconsistente.

É importante salientar que diretrizes do autor são propriedades a serem preservadas ao longo do ciclo de vida de um documento. Devido a personalização (no passo de *Interpretação*) ou mudanças incrementais (no passo *Dinâmico*), relações expressando o leiaute do documento podem ser modificadas. Diretrizes do autor representam uma forma de guiar tais modificações de forma a prover *preservação de intenção* [Sun et al. 1998] ao longo do ciclo de vida.

1.5. Soluções Propostas na Literatura

A literatura é rica em trabalhos sobre a validação de documentos multimídia. As soluções aqui apresentadas são classificadas de acordo com o método aplicado na validação de um documento. O primeiro grupo de trabalhos [Felix 2004, Gaggi and Bossi 2011, Bouyakoub and Belkhir 2011, Júnior et al. 2012, King et al. 2004, dos Santos 2016] realiza a validação de um documento investigando seu estado ao longo da sua execução. Tal abordagem pode ser feita analisando a alcançabilidade de estados, ou pela aplicação de axiomas sobre o estado do documento. O segundo grupo de trabalhos

[Bertino et al. 2005, Laborie et al. 2011, Elias et al. 2006, dos Santos 2016] realiza a validação de um documento checando a consistência de um conjunto de restrições.

1.5.1. Trabalhos Referentes a Alcançabilidade de Estados

Felix em [Felix 2004] apresenta uma abordagem para a validação temporal de documentos NCL [ITU 2009] através da aplicação de técnicas de *model-checking*. Ele apresenta uma notação para a descrição das características temporais de NCL, que segue um modelo baseado em eventos, a qual é transformada em uma representação similar a uma máquina de estados indicando o leiaute temporal de um documento. A transformação cria uma máquina de estados para cada mídia em um documento e uma "máquina de estados de sincronização" representando cada relação declarada no documento. Tais máquinas de sincronização são usadas para relacionar a ocorrência de eventos nas máquinas representando as mídias de um documento. A validação de um documento NCL é então realizada sobre um autômato resultante da combinação de tais máquinas de estado, o qual representa o leiaute temporal do documento. A validação é feita através de fórmulas em lógica temporal definidas pelo autor.

Gaggi e Bossi em [Gaggi and Bossi 2011] definem uma semântica formal para a linguagem SMIL através de um conjunto de regras de inferência inspiradas na lógica de Hoare. SMIL segue um modelo baseado em sincronização hierárquica e as regras descrevem o estado do documento antes e depois da execução de uma construção da linguagem SMIL. Assim, na fase de autoria, a estrutura de um documento SMIL é enriquecida com assertivas expressando propriedades temporais. Uma outra aplicação da semântica definida é o conceito de equivalência entre construções. Tal conceito garante que dois conjuntos de construções da linguagem SMIL podem ser substituídos em um documento sem mudar a apresentação resultante. A validação de um documento é realizada pela aplicação de axiomas, também definidos na semântica proposta, que verificam se uma dada construção, ou um conjunto delas, corretamente modifica o estado do documento. Caso contrário, a solução apresenta o problema encontrado para que o autor possa corrigi-lo.

Bouyakoub e Belkhir em [Bouyakoub and Belkhir 2011] apresentam uma ferramenta de autoria incremental para documentos SMIL, chamada *SMIL Builder*. Sempre que o autor realiza alguma mudança no documento, SMIL Builder verifica se tal modificação tornará o leiaute temporal do documento inconsistente. Em caso positivo, a mudança é rejeitada e um relato do erro é apresentado ao autor. Tanto a autoria incremental quanto a verificação de consistência são suportadas pelo modelo H-SMIL-Net [Bouyakoub and Belkhir 2008]. H-SMIL-Net estende Redes de Petri de forma a poder representar de forma completa conceitos temporais da linguagem SMIL. Inconsistências são detectadas seja por construção, por exemplo, quando uma transição tem mais arcos do que possível, ou por verificação, no modelo, das transições disparadas.

Júnior et al. em [Júnior et al. 2012] usam uma abordagem dirigida a modelos para a validação de documentos NCL. A validação é realizada transformando um documento NCL em uma estrutura de Kripke. Esta transformação é feita em duas etapas. Na primeira etapa o documento é representado em uma linguagem chamada FIACRE, como um conjunto de componentes e processos (representando o comportamento de um componente). A segunda etapa transforma a representação em FIACRE para uma estrutura de Kripke.

A validação usa uma ferramenta de *model-checking* e fórmulas em lógica temporal representando as propriedades a serem validadas. A validação espacial é brevemente discutida em [Júnior et al. 2012] e é realizada sobre a posição inicial das mídias declaradas no documento. Como apresentado na Tabela 1.1, esta abordagem cobre tanto as dimensões temporal e espacial, entretanto a dimensão espacial é estática.

King et al. em [King et al. 2004] definem extensões para a linguagem SMIL permitindo que autores possam descrever como o leiaute espaço-temporal deve reagir a um evento. Mudanças na posição e tamanho são descritas por um conjunto de expressões, as quais podem levar em consideração o estado do documento. O trabalho apresenta uma abordagem para calcular durante a execução do documento o valor de tais expressões e, portanto, fazer a mudança no leiaute espacial de acordo com a expressão. Tal abordagem, apesar de não ter por objetivo validar o leiaute temporal, ou espaço-temporal, de um documento, pode ser considerada um caso especial de validação, onde o leiaute da aplicação é recalculado dada a passagem do tempo ou a ocorrência de um evento.

dos Santos et al. em [dos Santos 2016] apresentam uma abordagem para a validação de documentos multimídia, especificados tanto em NCL quanto em SMIL, tendo como base um modelo formal chamado *SHM*. Uma das técnicas de validação propostas em [dos Santos 2016], chamada *RWT*, captura o comportamento de um documento representado no modelo *SHM* em termos do seu estado ao longo da sua execução. A validação proposta é baseada em uma teoria de reescrita $\mathscr{R}_{RWT}$, capaz de representar o comportamento de documentos multimídia ao longo do tempo e espaço. A teoria $\mathscr{R}_{RWT}$, induz um sistema de transição $\mathscr{S}_{RWT}$ onde os estados representam o estado do documento em um dado instante e transições representam a passagem do tempo ou uma interação do espectador. A validação tanto de propriedades temporais quanto espaciais é realizada através *model-checking* sobre $\mathscr{S}_{RWT}$. É importante destacar que em [dos Santos 2016] a validação de um documento pode ser realizada com base em um conjunto de propriedades comuns, bem como em propriedades construídas pelo autor. Tal facilidade é provida através de um conjunto de propriedades atômicas que podem ser combinadas formando as diretrizes desejadas pelo autor.

1.5.2. Trabalhos Referentes a Validação de Restrições

Bertino et al. em [Bertino et al. 2005] propõem um modelo de autoria baseado em restrições. Um documento neste modelo consiste de um conjunto de tópicos, onde cada tópico é composto de mídias semanticamente relacionadas. O sistema automaticamente agrupa mídias em tópicos de acordo com as restrições definidas pelo autor. O processo de geração da apresentação é responsável por três tarefas principais, são elas: checagem de consistência, geração da estrutura da apresentação e geração dos tópicos. O sistema adiciona no conjunto de restrições, outras restrições que embora não tenham sido definidas explicitamente, são consequência das restrições definidas pelo autor. A checagem de consistência é então realizada sobre esse conjunto de restrições. Se uma inconsistência é obtida, o sistema aplica técnicas de relaxamento para reduzir o conjunto de restrições em um conjunto consistente. Quando a redução automática não é possível, a revisão do autor é requisitada. O processo de criação da estrutura da apresentação cria um grafo dirigido que representa a estrutura da apresentação. Cada vértice do grafo representa um tópico e arestas representam uma conexão entre os tópicos. Após esta etapa, o sistema associa

mídias aos tópicos, cria o leiaute espacial e a sequência temporal de mídias para cada tópico.

Laborie et al. em [Laborie et al. 2011] apresentam uma abordagem de adaptação do leiaute de um documento de acordo com o dispositivo de exibição. No artigo, a linguagem SMIL é utilizada para demonstrar a abordagem proposta. A abordagem proposta cria uma descrição abstrata do documento como um conjunto de objetos e restrições representando relações temporais e espaciais entre objetos. É ainda considerado um perfil contendo restrições do dispositivo de exibição e preferências do usuário. De acordo com o conjunto de potenciais execuções de um documento $\mathscr{M}_s$ dado pela representação abstrata do documento, e o conjunto de potenciais execuções $\mathscr{M}_p$ dado pelo perfil, o processo de adaptação calcula a interseção $\mathscr{M}_s \cap \mathscr{M}_p$ para determinar se alguma adaptação é necessária. No caso de o documento precisar ser adaptado, o objetivo da solução proposta é alterar relações declaradas no documento de forma que este fique aderente ao perfil e que a distância (do comportamento) em relação ao documento original seja mínima.

Elias et al. em [Elias et al. 2006] também propõem um modelo de autoria baseado em restrições. A abordagem proposta define dois operadores *TEMPORAL* e *SPATIAL*, para modelar relações temporais e espaciais, respectivamente. Cada operador permite que o autor defina um valor de prioridade. Para manter a consistência do conjunto de restrições, sempre que necessário, restrições são removidas de acordo com seu valor de prioridade. No caso de duas restrições inconsistentes apresentarem o mesmo valor de prioridade, técnicas de relaxamento são aplicadas para determinar a restrição a ser removida. A checagem de consistência é feita achando a árvore geradora mínima T do conjunto de restrições. Restrições que criam ciclos são removidas para manter a natureza acíclica de T. A checagem de completude é feita buscando todas as mídias que são alcançadas a partir da primeira mídia. Se essa busca retorna o conjunto de vértices de T, então todas as mídias são alcançadas direta ou indiretamente a partir da mídia inicial. Caso contrário, o autor tem que definir restrições para tornar o conjunto de restrições completo. Com o uso do operador *SPATIAL*, é possível determinar se duas mídias A e B se sobrepõem. Cada restrição espacial é associada a um intervalo, de forma que uma restrição espacial deve ser satisfeita dentro deste intervalo. Apesar de o trabalho apresentado em [Elias et al. 2006] não definir atributos espaciais (posição e tamanho) em função do tempo, ele representa um exemplo de restrições parametrizadas pelo tempo.

dos Santos et al. em [dos Santos 2016] em sua abordagem baseada no modelo formal *SHM* provêem uma segunda técnica de validação, chamada *SMT*, que captura o comportamento de um documento em termos de intervalos e ocorrências de eventos. Intervalos temporais e regiões espaciais representando fragmentos do documento e relações espaço-temporais entre tais intervalos/regiões são descritas através de fórmulas em SMT (*Satisfiability Modulo Theories*) [De Moura and Bjørner 2011]. A validação tanto de propriedades temporais quanto espaciais é realizada através de um *solver* SMT. Assim como na técnica anteriormente descrita, a validação pode ser realizada com base em um conjunto de propriedades comuns, bem como em propriedades construídas pelo autor.

1.5.3. Comparação entre as Soluções Propostas

A seção anterior apresentou algumas soluções propostas na literatura para a validação de documentos multimídia. Tais soluções foram classificadas de acordo com o método aplicado na validação e o tipo de validação realizada (temporal e/ou espacial). A Tabela 1.1 resume tais soluções de acordo com as três principais fases do ciclo de vida onde atuam (*criação*, *instanciação* e *execução*).

	Temporal	Temporal + Espacial	Espaço-temporal	Alcance de Estados	Validação de Restrições	Criação	Instanciação	Execução
[Felix 2004]	✓			✓		✓		
[Gaggi and Bossi 2011]	✓			✓		✓		
[Bouyakoub and Belkhir 2011]	✓			✓		✓		
[Júnior et al. 2012]		✓		✓		✓		
[King et al. 2004]			✓	✓				✓
[Bertino et al. 2005]		✓			✓	✓	✓	
[Laborie et al. 2011]		✓			✓	✓	✓	
[Elias et al. 2006]			✓		✓	✓		
[dos Santos 2016]			✓	✓	✓	✓		✓

Tabela 1.1: Comparação entre as soluções propostas na literatura

Como pode ser visto na Tabela 1.1, as soluções apresentadas em [Felix 2004, Gaggi and Bossi 2011, Bouyakoub and Belkhir 2011] apresentam uma abordagem puramente temporal, onde a validação de um documento é realizada investigando seu estado ao longo de sua execução. Em [Felix 2004], a validação é feita analisando a alcançabilidade de estados, em [Bouyakoub and Belkhir 2011] a validação é feita verificando a consistência da rede de Petri representando um documento e em [Gaggi and Bossi 2011] analisando se o estado do documento muda de acordo com alguns axiomas. A validação do leiaute espacial do documento não é discutida nesses trabalhos.

As soluções apresentadas em [Júnior et al. 2012, Bertino et al. 2005, Laborie et al. 2011] cobrem tanto as dimensões temporais e espaciais. No primeiro, a validação de um documento é realizada via *model-checking* e, no segundo e terceiro trabalhos a validação é realizada pela checagem de consistência de um conjunto de restrições. Entretanto, em todos esses trabalhos, a dimensão espacial é estática, visto que restrições espaciais não mudam ao longo do tempo. Ainda, raciocinar sobre tempo e espaço é feito como dois problemas separados.

Finalmente, as soluções propostas em [Elias et al. 2006, dos Santos 2016] provêem uma validação verdadeiramente espaço-temporal, dado que relações espaciais podem variar ao longo do tempo. Apesar de [King et al. 2004] suportar mudanças no leiaute

espacial ao longo do tempo, tal abordagem não tem por objetivo suportar a validação do documento. Em [Elias et al. 2006] a validação é realizada pela checagem de consistência de um conjunto de restrições. Em [dos Santos 2016], ambas as técnicas são suportadas.

As soluções apresentadas nesta seção atuam em diferentes etapas do ciclo de vida de um documento (Figura 1.8), como visto na Tabela 1.1. Grande parte das soluções atuam na etapa de *criação*, como esperado, uma vez que propõem ferramentas ou abordagens para facilitar a autoria de documentos. Ambas as soluções [Bertino et al. 2005, Laborie et al. 2011] levam em consideração o contexto do espectador para prover adaptação do documento antes de sua execução, portanto atuando na etapa de *instanciação*. A validação provida por tais trabalhos, entretanto, possui um papel preventivo. Eles investigam se um documento é executável e se nenhuma inconsistência pode surgir durante sua execução. Tais soluções, entretanto, não levam em consideração o caso em que um documento é editado dinamicamente. É importante salientar também que apesar de tais propostas focarem na adaptação, elas poderiam ser utilizadas também na etapa de criação para validar um documento sendo criado.

As soluções apresentadas em [King et al. 2004, dos Santos 2016] são os únicos trabalhos que atuam na etapa de *execução*. Em [King et al. 2004], é proposta uma forma de calcular, durante a execução, o leiaute espacial de um documento. Em [dos Santos 2016], são providas duas técnicas de validação, onde ambas atuam na etapa de *criação*. Ainda, um protótipo da validação usando restrições para a validação de um documento durante sua execução também é apresentado.

1.6. Exemplo de Verificação de Consistência

Esta seção apresenta um documento multimídia para exemplificar a validação discutida ao longo deste capítulo. O documento de exemplo, chamado “Roteiro do Dia” é um documento especificado com a linguagem NCL [ITU 2009] e está disponível para *download* no Clube NCL[2].

Este documento é um exemplo de programa de televisão interativo. O programa em questão faz uma visita turística a uma cidade, apresentando seus principais pontos turísticos. No programa, a cidade do Rio de Janeiro é apresentada, sendo visitados os seguintes pontos turísticos: Central do Brasil, Copacabana, Jardim Botânico e a Gafieira Estudantina.

O espectador pode escolher, ao início da apresentação do documento se está disposto ou não a interagir. Estando disposto, ao fim da visita a um ponto turístico, o espectador escolhe o próximo ponto a ser visitado. Assim, uma visita turística personalizada é apresentada. Caso o espectador não esteja disposto a interagir, um roteiro de visita escolhido previamente pelos autores do programa é apresentado.

Nesta seção, será utilizada como exemplo de solução para a validação do documento acima, a API aNaa (API NCL de Autoria e Análise) [dos Santos 2012, dos Santos 2016]. A Figura 1.9 apresenta aNaa4Web, uma interface web construída sobre aNaa para utilizar a validação por ela provida.

[2]http://clube.ncl.org.br

Figura 1.9: Interface inicial da ferramenta aNaa4Web

Uma vez escolhido o documento a ser validado, a ferramenta apresenta em sua barra superior, o identificador e o caminho para o arquivo principal do documento. A ferramenta ainda identifica as mídias que possuem uma duração implícita (áudio ou vídeo) para que o usuário indique a duração em segundos de cada. A Figura 1.10 apresenta a interface da ferramenta após a escolha do documento "Roteiro do Dia".

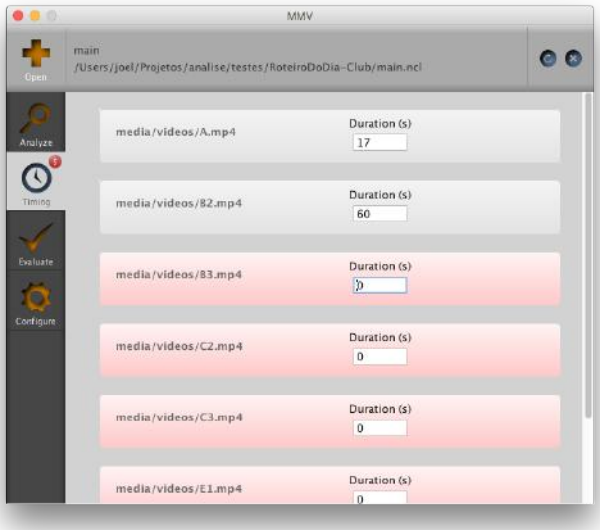


Figura 1.10: Interface para indicação da duração das mídias

Na Figura 1.10 é possível identificar o contador apresentado sobre o botão "Timing". Este contador indica o número de mídias para as quais a duração ainda não foi identificada. Ainda, para cada mídia sem duração, uma campo é apresentado onde o usuá-

rio poderá indicar a duração da mídia. Uma vez indicada a duração, o campo muda da cor vermelha para a cinza e o contador é decrementado.

Ao clicar no botão "Analyze", um menu é apresentado onde o usuário pode escolher se quer realizar a validação estrutural ou comportamental de um documento. Inicialmente será apresentado o uso da ferramenta para a validação estrutural do documento. A validação estrutural é realizada com base nas propriedades estruturais apresentadas na Seção 1.3.1. Cada propriedade é instanciada para a linguagem NCL, gerando um conjunto de propriedades a serem validadas. O resultado da validação é apresentado na Figura 1.11.

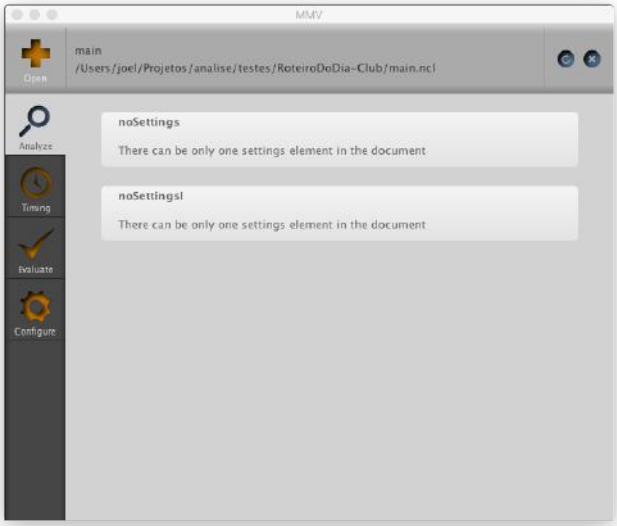


Figura 1.11: Resultado da Validação Estrutural

Note que, duas mensagens de erro são apresentadas, ambas relacionadas ao fato de existirem duas mídias do tipo *settings* no documento. Uma mídia do tipo *settings* é uma mídia que representa um conjunto de variáveis globais em um documento e, de acordo com a especificação da linguagem NCL [ABNT 2011], deve ser única.

Uma vez corrigido o documento, pode ser realizada sua validação comportamental. A API aNaa provê uma validação espaço-temporal do comportamento do documento, seguindo uma abordagem baseada na alcançabilidade de estados. Como apresentado na Seção 1.5.1, o documento a ser validado é representado no modelo formal *SHM* como uma teoria de reescrita. Esta teoria induz um sistema de transição representando as possíveis execuções do documento Roteiro do Dia, onde cada estado representa o estado das mídias declaradas no documento e as transições representam uma progressão temporal ou interação do espectador. A interação, nesse caso, representa a escolha da próxima atração turística a ser visitada.

A validação comportamental é realizada através *model-checking* com base nas propriedades comportamentais gerais apresentadas na Seção 1.3.2. Cada propriedade é instanciada, para cada mídia do documento, em um comando em lógica temporal e validada sobre o sistema de transição representando as possíveis execuções do documento.

A Figura 1.12 apresenta o resultado da validação comportamental do documento Roteiro do Dia.

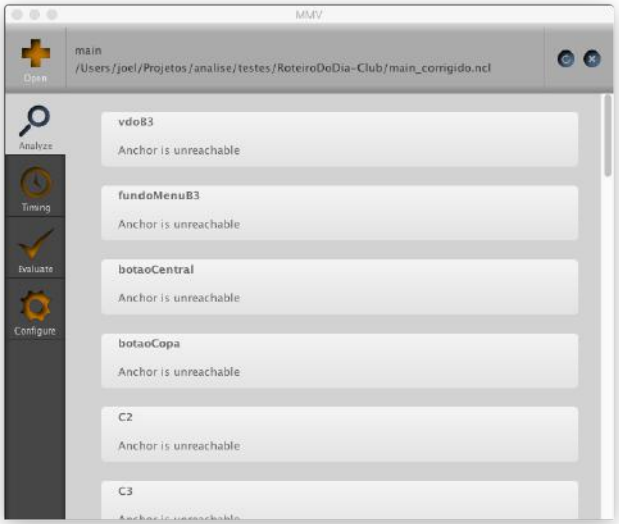


Figura 1.12: Resultado da Validação Comportamental

É possível notar que a ferramenta identifica diversos *warnings* no documento. Tais *warnings* estão relacionados ao fato de que diferentes mídias podem não ser apresentadas, de acordo com a interação do espectador.

Apesar da API aNaa suportar a validação de propriedades criadas pelo autor, a interface aNaa4Web só prevê a validação das propriedades gerais apresentadas na Seção 1.3.2. Uma segunda ferramenta construída tendo como base a API aNaa, chamada NCL-Tester [Barreto et al. 2016] permite que o autor crie um conjunto de propriedades específicas a um documento e realize sua validação. A interface do NCL-Tester é apresentada na Figura 1.13.

NCL-Tester define um conjunto de propriedades temporais que podem ser utilizadas como base para a criação de testes pelo autor. Para cada propriedade instanciada, o autor indica as mídias a serem consideradas. A validação das propriedades criadas pode ser feita em separado ou em conjunto.

No exemplo apresentado na Figura 1.13 a ferramenta NCL-Tester foi utilizada para criar um teste sobre o documento "Roteiro do Dia". O teste criado especifica que a apresentação do vídeo *vdoB3* (apresenta a cidade e chama o espectador a interagir) deve preceder a apresentação do vídeo *C2* (apresenta a Central do Brasil).

O teste criado é do tipo *meets*, sendo baseado na relação de Allen [Allen 1983] de mesmo nome. Na aba criada para o teste, a mídia *vdoB3* foi arrastada para a posição da mídia mestre do teste enquanto a mídia *C2* foi arrastada para a posição escrava. O teste é então traduzido para uma propriedade em lógica temporal e enviado para a API aNaa para que ela possa realizar a validação da propriedade. O resultado do teste é apresentado no painel do lado direito da tela. Além disso, a ferramenta indica visualmente o resultado do

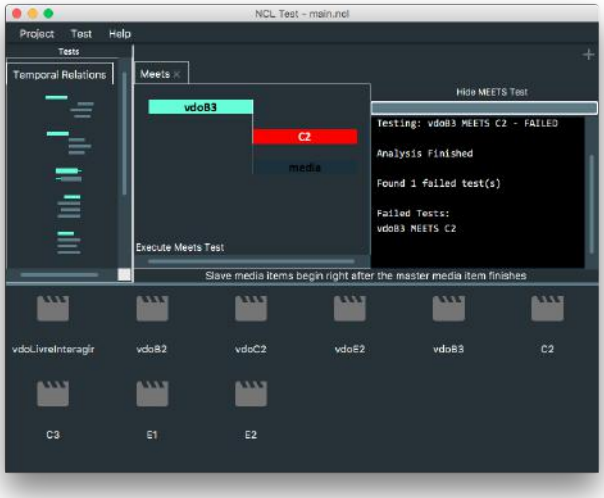


Figura 1.13: Interface do NCL-Tester [Barreto et al. 2016]

teste, alterando a cor da mídia escrava para verde, caso o teste passe, ou vermelho, caso falhe. O teste apresentado na Figura 1.13 falha, pois é possível que o vídeo *C2* não seja executado após o vídeo *vdoB3* devido a interação feita pelo espectador.

Esta seção apresentou um exemplo de documento NCL e sua validação usando uma solução presente na literatura. A Seção a seguir discute desafios de pesquisa ainda em aberto relacionados à validação de documentos multimídia.

1.7. Desafios de Pesquisa

Foram apresentadas na Seção 1.5 diferentes soluções para a validação de documentos multimídia disponíveis na literatura. Apesar da existência de diferentes soluções, algumas facilidades relacionadas à validação ainda são desafios em aberto.

Um primeiro desafio está relacionado à utilização da validação nas diferentes etapas do ciclo de vida de um documento. Grande parte dos trabalhos focam na validação durante a fase de *criação*. A validação durante as fases de *instanciação* e *execução* ainda não são muito exploradas na literatura. Além disso, uma abordagem para a definição de diretrizes pelo autor, que estejam disponíveis em tempo de execução, ainda não foi apresentada.

Documentos multimídia podem ser criados através do uso de templates de documentos [dos Santos and Muchaluat-Saade 2012, Neto et al. 2012, Deltour and Roisin 2006]. Um template representa uma especificação parcial de um documento que deve ser completada com mídias específicas escolhidas pelo autor. Visto que um template pode ser utilizado para a criação de uma gama de documentos compartilhando características comuns especificadas no template, realizar a validação de um template representa uma forma de garantir que documentos criados a partir deste template sejam consistentes.

Templates abstraem o conteúdo de um documento representando um conjunto de mídias semanticamente relacionadas como um componente do template. Relações podem

então ser definidas (i) entre componentes (conjuntos) ou (ii) sobre um mesmo componente (conjunto). Como um exemplo do primeiro caso, suponha um template definindo uma relação onde sempre que uma mídia do conjunto A é selecionada (componente foto), uma outra mídia do conjunto B (componente descrição) é apresentada. Um caso real seria uma aplicação com fotos e suas descrições, por exemplo. Como um exemplo do segundo caso, suponha um template definindo uma relação sobre um conjunto de mídias (componente foto), onde assim que uma mídia na posição i termina sua apresentação, começa a apresentação de uma mídia na posição $i+1$. Um caso real seria uma aplicação que apresenta um conjunto de fotos em sequência.

Um desafio para a validação de templates, portanto, é a representação destes dois casos. No caso em que uma relação é definida entre dois conjuntos, cada conjunto pode ser pensado como uma única mídia. No caso em que uma relação é definida sobre um mesmo conjunto, entretanto, a abordagem anterior não pode ser utilizada, sendo necessário instanciar esse template para um determinado número de mídias. Tal diferença de representação se apresenta como um grande desafio para a validação de templates.

Documentos multimídia podem ter parte de seu conteúdo obtido sob demanda quando executado. Nesse caso, atrasos na obtenção de tais conteúdos podem impactar na apresentação de um documento. Ainda, é possível que a plataforma onde um documento é executado não seja capaz de manter uma exata sincronização entre mídias, por exemplo, adicionando um pequeno atraso no início de apresentação de cada mídia. Tais características dependem da plataforma onde um documento é executado. É possível que um documento consistente torne-se inconsistente durante sua execução, devido a tais características. Um desafio de pesquisa é pensar em como representar tais características, simulando um mau funcionamento da plataforma de exibição, visando checar a robustez de um documento. Isto significa que o documento possui relações capazes de corrigir pequenos erros na sincronização do documento, mesmo quando alguma inconsistência é inserida pela plataforma de exibição.

Similar ao caso acima, é a validação de documentos multimídia distribuídos. No caso em que diferentes dispositivos são utilizados para apresentar um mesmo documento, devem ser levados em consideração, durante a validação, eventuais atrasos de comunicação entre os dispositivos responsáveis pela exibição de um documento. Exemplos de exibição de um documento em múltiplos dispositivos são aplicação com segunda tela, e aplicações mulsemídia (*multiple sensorial media*) [Ghinea et al. 2014] onde o espectador recebe estímulos sensoriais para outros sentidos além de visão e audição, já presentes em documentos multimídia. Tais estímulos são produzidos pelos chamados atuadores, que devem ser controlados em conjunto como em uma aplicação em múltiplos dispositivos.

Por fim, um outro desafio de pesquisa é no caso de aplicações mulsemídia. O desafio, nesse caso é como representar estímulos sensoriais e sensores, de forma a poder-se validar um documento que leve em consideração tais componentes.

1.8. Conclusão

Um documento multimídia descreve um conjunto de mídias e como estas devem ser apresentadas, tanto no tempo, quanto no espaço. Quando executada, essa descrição induz um arranjo particular das mídias no tempo e espaço, chamado de uma apresentação multimí-

dia. A descrição contida em um documento é comumente textual, utilizando uma linguagem de autoria. Diferentes linguagens de autoria, bem como trabalhos publicados na literatura provêem formas de se adaptar um documento ao contexto do leitor. Por exemplo, é possível reorganizar mídias na tela do dispositivo de exibição de acordo com seu tamanho. Algumas linguagens provêem ainda a possibilidade de editar dinamicamente um documento ao longo da sua apresentação. Isso significa que relações espaço-temporais em um documento podem ser modificadas enquanto o documento é executado.

Desde sua criação até sua execução, um documento passa por um conjunto de etapas, chamadas aqui de o ciclo de vida de um documento multimídia. Visto que a especificação contida em um documento pode variar ao longo do seu ciclo de vida, é importante verificar, ao longo de cada etapa, se modificações realizadas em um documento o transformam em algo diferente do que foi especificado em tempo de criação. Assim, dizemos que um documento multimídia é consistente, se este se mantém aderente a um conjunto de diretrizes do autor. A verificação da consistência de um documento é feita através de sua validação.

Diferentes tipos de validação podem ser feitos sobre um documento. Neste capítulo foram discutidas a validação estrutural e comportamental de um documento. A validação comportamental foi ainda classificada em três tipos: puramente temporal, temporal e espacial analisadas em separado e espaço-temporal. Este capítulo apresentou ainda o ciclo de vida de um documento multimídia composto de etapas onde a validação de um documento se faz necessária. Foram ainda apresentadas soluções propostas na literatura e como cada solução se encaixa no ciclo apresentado e tipo de validação provida. Uma das soluções apresentadas foi utilizada para apresentar um exemplo de verificação de consistência, demonstrando seu uso para validar um documento em diferentes etapas do seu ciclo de vida. Por fim, foram discutidos alguns desafios de pesquisa relacionados à validação de documentos multimídia ainda em aberto.

Referências

[ABNT 2011] ABNT (2011). Digital terrestrial television - data coding and transmission specification for digital broadcasting - part 2: Ginga-ncl for fixed and mobile receivers - xml application language for application coding. ABNT NBR 15606-2:2011 standard.

[Adobe 2010] Adobe (2010). *ActionScript references and documentation*. Adobe Systems Incorporated.

[Allen 1983] Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843.

[Araújo et al. 2008] Araújo, E. C., Azevedo, R. G. A., and Neto, C. S. S. (2008). Ncl-validator: um processo para validação sintática e semântica de documentos multimídia ncl. In *Jornada de Informática do Maranhão*. in portuguese.

[ARIB 2014] ARIB (2014). Data coding and transmission specification for digital broadcasting. Standard B24 v6.1.

[ATSC 2009] ATSC (2009). Advanced common application platform (acap). Document A/101A.

[Barreto et al. 2016] Barreto, F., Tamaki, D., dos Santos, J. A. F., and Muchaluat-Saade, D. C. (2016). Ncl-tester: Ferramenta gráfica para criação de testes temporais para documentos ncl. In *Proceedings of the 22th Brazilian Symposium on Multimedia and the Web*, WebMedia '16. ACM.

[Bertino et al. 2005] Bertino, E., Ferrari, E., Perego, A., and Santi, D. (2005). A constraint-based approach for the authoring of multi-topic multimedia presentations. In *IEEE International Conference on Multimedia and Expo*, pages 578–581, Amsterdam, Netherlands. IEEE Computer Society.

[Blakowski and Steinmetz 1996] Blakowski, G. and Steinmetz, R. (1996). A media synchronization survey: Reference model, specification and case studies. *Journal on Selected Areas in Communications*, 14(1):5–35.

[Boll 2001] Boll, S. (2001). *ZYX - Towards flexible multimedia document models for reuse and adaptation*. PhD thesis, Vienna University of Technology.

[Bouyakoub and Belkhir 2008] Bouyakoub, S. and Belkhir, A. (2008). H-smil-net: A hierarchical petri net model for smil documents. In *Proceedings of the Tenth International Conference on Computer Modeling and Simulation*, UKSIM '08, pages 106–111, Washington, DC, USA. IEEE Computer Society.

[Bouyakoub and Belkhir 2011] Bouyakoub, S. and Belkhir, A. (2011). Smil builder: An incremental authoring tool for smil documents. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, 7(1):2:1–2:30.

[Buchanan and Zellweger 1992] Buchanan, M. C. and Zellweger, P. T. (1992). Specifying temporal behavior in hypermedia documents. In *Proceedings of the ACM conference on Hypertext*, pages 262–271. ACM.

[Buchanan and Zellweger 2005] Buchanan, M. C. and Zellweger, P. T. (2005). Automatic temporal layout mechanisms revisited. *ACM Transactions on Multimedia Computing, Communications and Applications*, 1(1):60–88.

[Bulterman et al. 2013] Bulterman, D., Cesar, P., and Guimaraes, R. L. (2013). Socially-aware multimedia authoring: Past, present, and future. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, 9(1s):35.

[Damasceno et al. 2014] Damasceno, J. R., dos Santos, J. A. F., and Muchaluat-Saade, D. C. (2014). Editec - a graphical editor for hypermedia composite templates. *Multimedia Tools and Applications*, 70(2):1167–1198.

[De Moura and Bjørner 2011] De Moura, L. and Bjørner, N. (2011). Satisfiability modulo theories: Introduction and applications. *Communications of the ACM*, 54(9):69–77.

[de Oliveira et al. 2001] de Oliveira, M., Turine, M., and Masiero, P. (2001). A statechart-based model for hypermedia applications. *ACM Transactions on Information Systems (TOIS)*, 19(1):52.

[Deltour and Roisin 2006] Deltour, R. and Roisin, C. (2006). The limsee3 multimedia authoring model. In *Proceedings of the 2006 ACM symposium on Document engineering*, pages 173–175. ACM.

[Díaz et al. 2001] Díaz, P., Aedo, I., and Panetsos, F. (2001). Modeling the dynamic behavior of hypermedia applications. *IEEE Transactions on Software Engineering*, 27(6):550–572.

[dos Santos 2012] dos Santos, J. A. F. (2012). Multimedia and hypermedia document validation and verification using a model-driven approach. Master's thesis, UFF.

[dos Santos 2016] dos Santos, J. A. F. (2016). *Multimedia Document Validation Along its Life Cycle*. PhD thesis, Universidade Federal Fluminense.

[dos Santos et al. 2013] dos Santos, J. A. F., Braga, C., and Muchaluat-Saade, D. C. (2013). Automating the analysis of ncl documents with a model-driven approach. In *Proceedings of the 19th Brazilian Symposium on Multimedia and the Web*, WebMedia '13, pages 193–200, New York, NY, USA. ACM.

[dos Santos et al. 2015] dos Santos, J. A. F., Braga, C., and Muchaluat-Saade, D. C. (2015). A rewriting logic semantics for ncl. *Science of Computer Programming*, 107–108:64 – 92.

[dos Santos and Muchaluat-Saade 2012] dos Santos, J. A. F. and Muchaluat-Saade, D. C. (2012). Xtemplate 3.0: spatio-temporal semantics and structure reuse for hypermedia compositions. *Multimedia Tools and Applications*, 61(3):645–673.

[Elias et al. 2006] Elias, S., Easwarakumar, K., and Chbeir, R. (2006). Dynamic consistency checking for temporal and spatial relations in multimedia presentations. In *Proceedings of the 2006 ACM symposium on Applied computing*, pages 1380–1384, Dijon, France. ACM.

[ETSI 2008] ETSI (2008). Digital video broadcasting (dvb); multimedia home platform (mhp) specification 1.0.3. ETSI ES 201 812 v1.1.2.

[Felix 2004] Felix, M. F. (2004). *Formal Analysis of Software Models Oriented by Architectural Abstractions*. PhD thesis, Pontifícia Universidade Católica do Rio de Janeiro. in Portuguese.

[Furuta and Stotts 2001] Furuta, R. and Stotts, P. D. (2001). Trellis: a formally-defined hypertextual basis for integrating task and information. *Coordination Theory and Collaboration Technology*, pages 341–367.

[Gaggi and Bossi 2011] Gaggi, O. and Bossi, A. (2011). Analysis and verification of smil documents. *Multimedia Systems*, 17(6):487–506.

[Ghinea et al. 2014] Ghinea, G., Timmerer, C., Lin, W., and Gulliver, S. R. (2014). Mulsemedia: State of the art, perspectives, and challenges. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 11(1s):17.

[Harel 1987] Harel, D. (1987). Statecharts: A visual formalism for complex systems. *Science of computer programming*, 8(3):231–274.

[ISO/IEC 2005] ISO/IEC (2005). Information technology - coding of audio-visual objects - part 11: Scene description and application engine. ISO/IEC 14496-11:2005.

[ITU 2009] ITU (2009). Nested context language (ncl) and ginga-ncl for iptv services. http://www.itu.int/rec/T-REC-H.761-200904-S. ITU-T Recommendation H.761.

[Jourdan et al. 1998] Jourdan, M., Layaida, N., Roisin, C., Sabry-Ismail, L., and Tardif, L. (1998). Madeus, an authoring environment for interactive multimedia documents. In *Proceedings of the 6th ACM International Conference on Multimedia*, pages 267–272. ACM.

[Júnior et al. 2012] Júnior, D. P., Farines, J.-M., and Koliver, C. (2012). An approach to verify live ncl applications. In *Proceedings of the 18th Brazilian Symposium on Multimedia and the Web*, WebMedia '12, pages 223–232, New York, NY, USA. ACM.

[King et al. 2004] King, P., Schmitz, P., and Thompson, S. (2004). Behavioral reactivity and real time programming in xml: functional programming meets smil animation. In *Proceedings of the 2004 ACM symposium on Document engineering*, pages 57–66. ACM.

[Laborie et al. 2011] Laborie, S., Euzenat, J., and Layaïda, N. (2011). Semantic adaptation of multimedia documents. *Multimedia tools and applications*, 55(3):379–398.

[Lemlouma and Layaïda 2004] Lemlouma, T. and Layaïda, N. (2004). Context-aware adaptation for mobile devices. In *IEEE International Conference on Mobile Data Management*, pages 106–111.

[Muchaluat-Saade 2003] Muchaluat-Saade, D. C. (2003). *Relations in Hypermedia Authoring Languages: Improving Reuse and Expressiveness*. PhD thesis, Pontifícia Universidade Católica do Rio de Janeiro.

[Na and Furuta 2001] Na, J. and Furuta, R. (2001). Dynamic documents: authoring, browsing, and analysis using a high-level petri net-based hypermedia system. In *Proceedings of the 2001 ACM Symposium on Document engineering*, pages 38–47. ACM.

[Neto et al. 2012] Neto, C. d. S. S., Soares, L. F. G., and de Souza, C. S. (2012). Tal-template authoring language. *Journal of the Brazilian Computer Society*, 18(3):185–199.

[Neto et al. 2011] Neto, J. R. C., Santos, R. C. M., Neto, C. S. S., and Teixeira, M. M. (2011). Método de validação estrutural e contextual de documentos ncl. In *Proceedings of the 17th Brazilian Symposium on Multimedia and the Web*, pages 1–8. in Portuguese.

[Peterson 1981] Peterson, J. L. (1981). *Petri Net Theory and Modeling of systems*. Prentice-Hall.

[Santos et al. 1998] Santos, C., Soares, L., de Souza, G., and Courtiat, J. (1998). Design methodology and formal validation of hypermedia documents. In *Proceedings of the sixth ACM International Conference on Multimedia*, pages 39–48, Bristol, United Kingdom. ACM.

[Sarkis et al. 2014] Sarkis, M., Concolato, C., and Dufourd, J.-C. (2014). The virtual splitter: Refactoring web applications for themultiscreen environment. In *Proceedings of the 2014 ACM Symposium on Document Engineering*, DocEng '14, pages 139–142, New York, NY, USA. ACM.

[Soares et al. 2012] Soares, L. F. G., Neto, C. S. S., and Sousa, J. G. (2012). Architecture for hypermedia dynamic applications with content and behavior constraints. In *ACM symposium on Document engineering*, pages 217–226. ACM.

[Soares and Rodrigues 2005] Soares, L. F. G. and Rodrigues, R. F. (2005). Nested context model 3.0 part 1 - ncm core. Technical report, Informatics Department, PUC-Rio, Rio de Janeiro.

[Soares et al. 2006] Soares, L. F. G., Rodrigues, R. F., Costa, R. R., and Moreno, M. F. (2006). Nested context language 3.0 part 9 - ncl live editing commands. Technical report, Informatics Department, PUC-Rio, Rio de Janeiro.

[Soares et al. 2000] Soares, L. F. G., Rodrigues, R. F., and Muchaluat-Saade, D. C. (2000). Modeling, authoring and formatting hypermedia documents in the hyperprop system. *Multimedia Systems*, 8(2):118–134.

[Sun et al. 1998] Sun, C., Jia, X., Zhang, Y., Yang, Y., and Chen, D. (1998). Achieving convergence, causality preservation, and intention preservation in real-time cooperative editing systems. *ACM Trans. Comput.-Hum. Interact.*, 5(1):63–108.

[Teixeira et al. 2012] Teixeira, C. A. C., Melo, E. L., Freitas, G. B., Santos, C. A. S., and Pimentel, M. d. G. C. (2012). Discrimination of media moments and media intervals: sticker-based watch-and-comment annotation. *Multimedia Tools and Applications*, 61(3):675–696.

[W3C 1999] W3C (1999). Synchronized multimedia integration language (smil) document object model. http://www.w3.org/TR/SMIL/smil-DOM.html. World-Wide Web Consortium Recommendation.

[W3C 2000] W3C (2000). Document object model (dom) level 2 core specification. World-Wide Web Consortium Recommendation DOM-Level-2-Core-20001113.

[W3C 2008a] W3C (2008a). Extensible markup language (xml) 1.0 (fifth edition). World-Wide Web Consortium Recommendation.

[W3C 2008b] W3C (2008b). Synchronized multimedia integration language - smil 3.0 specification. http://www.w3c.org/TR/SMIL3. World-Wide Web Consortium Recommendation.

[W3C 2014] W3C (2014). Html5: A vocabulary and associated apis for html and xhtml. World-Wide Web Consortium Candidate Recommendation.

[W3C 2014] W3C (2014). Web animations 1.0. http://www.w3.org/TR/web-animations/. World-Wide Web Consortium Working Draft.

[Wahl and Rothermel 1994] Wahl, T. and Rothermel, K. (1994). Representing time in multimedia systems. In *Proceedings of International Conference on Multimedia Computing and Systems*. IEEE Computer Society Press.

[Willrich et al. 2001] Willrich, R., de Saqui-Sannes, P., Sénac, P., and Diaz, M. (2001). Design and management of multimedia information systems. pages 380–411.

Biografia Resumida dos Autores

Dr. Joel dos Santos possui graduação em engenharia de telecomunicações (2009), mestre (2012) e doutor (2016) em Ciência da Computação pela Universidade Federal Fluminense (UFF). É professor da Escola de Informática e Computação (EIC) do Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (CEFET/RJ), atuando em cursos relacionados à Ciência da Computação nos níveis Técnico, Graduação e Pós-Graduacão. Atua desde 2006 em áreas relacionadas a multimídia, dentre as quais destacam-se: Autoria multimídia, Multissesorial mídia e TV digital. Pelo laboratório MídiaCom (UFF) participou do desenvolvimento da Linguagem XTemPlate 3.0 e das APIs aNa e aNaa. Durante intercâmbio na Universität Ulm, Alemanha, tive a oportunidade de trabalhar em projeto de pesquisa no OMI (*Institut für Organization und Management von Informationssytemen*). Como parte do programa Ciência sem Fronteiras trabalhei como pesquisador convidado do grupo Tyrex no Inria (*Institut National de Recherche en Informatique et en Automatique*) em Grenoble, França.

Dr. Débora Christina Muchaluat Saade possui graduação em Engenharia de Computação (1992), mestrado em Informática (1996) e doutorado em Informática pela Pontifícia Universidade Católica do Rio de Janeiro (2003). Desde 2002, é professora da Universidade Federal Fluminense. Integrou o corpo docente do Departamento de Engenharia de Telecomunicações até maio de 2009 e desde então faz parte do corpo docente do Instituto de Computação. É bolsista de produtividade DT do CNPq e foi jovem cientista pela FAPERJ. Tem experiência nas áreas de Ciência da Computação e Engenharia de Telecomunicações, com ênfase em Teleinformática, atuando principalmente nos seguintes temas: redes de computadores, redes sem fio, sistemas multimídia e hipermídia distribuídos, linguagens de autoria hipermídia, televisão digital interativa e telemedicina. Participou do desenvolvimento da linguagem NCL - Nested Context Language - adotada como padrão ABNT NBR 15606-2 no middleware GINGA do Sistema Brasileiro de TV Digital e como recomendação internacional do ITU-T H.761 para serviços IPTV.

Capítulo 2

Avaliação de Acessibilidade e Usabilidade em RIA

Renata Pontin M. Fortes[1], Humberto Lidio Antonelli[1] e André de Lima Salgado[1]

[1]Instituto de Ciências Matemáticas e de Computação – Universidade de São Paulo (ICMC/USP)

Resumo

A popularidade de aplicações de Internet atingiu escalas abrangentes. Como consequência, uma grande diversidade de soluções foram criadas baseadas em funcionalidades Web. Aplicações de Internet Ricas (RIAs) é um termo importante adotado para avanços tecnológicos em software desenvolvidos para a Web, e que confere a aplicações Web meios para prover aos usuários uma experiência similar à experiência de uso de desktops. RIAs possuem, usualmente, capacidades mais amplas em comparação com aplicações tradicionais de hipertexto, especialmente considerando elementos interativos de interfaces. Novas possibilidades que emergiram a partir de RIAs foram essenciais para suportar aspectos de Web 2.0, como a participação e colaboração. Como entre outras aplicações, o desenvolvimento de RIAs usáveis e acessíveis é aspecto valioso e fundamental para as equipes de desenvolvimento. Algumas das novas funcionalidades de interação RIA disponíveis na Web ainda não são acessíveis por usuários com deficiências ou mobilidade reduzida. Por esta razão, este minicurso visa apresentar alguns dos principais conceitos usualmente aplicados para avaliação de usabilidade e acessibilidade RIA; ele abrange perspectivas sobre referenciais práticos e teóricos, a partir de Padrões de Qualidade até implementação de conteúdos para aplicações Web. A abordagem deste minicurso cobriu práticas sobre principais técnicas de codificação RIA, e roteiros para métodos de avaliação de acessibilidade e usabilidade como Avaliação Heurística e Testes com Usuários. Este minicurso foi desenvolvido para novatos e profissionais que desejam especializar suas habilidades para o desenvolvimento de aplicações RIA usáveis e acessíveis.

Palavras-chave: RIA, Acessibilidade, Usabilidade, Roteiro, Avaliação

Abstract

Popularity of Internet applications has reached wide scales. In consequence, a wide diversity of solutions has been created based on Web features. Rich Internet Application (RIA) is a important term adopted for technological advances in software developed for Web, which refers to Web applications aimed at providing users with a desktop similar experience. RIAs usually have wider capabilities in comparison with traditional hypertext applications, specially regarding interactive elements of their interfaces. New possibilities emerged from RIA were essential to support relevant aspects of Web 2.0, such as participation and collaboration. As among other applications, developing accessible and usable RIAs is a valuable and fundamental aspect for development teams. Some RIAs new interaction features available on Web are still not accessible and usable for people with disabilities or reduced mobility. For this reason, this mini-course aimed at presenting some of the main concepts usually used on accessibility and usability evaluation of RIA; it is an overview of perspectives about practices and theoretical references, from recent Standards for Quality up to the implementation resources of Web applications. The approach of this mini-course covered practices on main RIA coding techniques, and a Guide for methods of usability and accessibility evaluation as Heuristic Evaluation and Test with Users. Moreover, this mini-course was developed aiming newcomers and professionals that want to specialize their skills on the development and evaluation of usable and accessible RIAs.

Keywords: RIA, Accessibility, Usability, Guide, Evaluation

2.1. Introdução

A popularização do acesso à Internet possibilitou que grande parte das tarefas diárias (compras, movimentações financeiras, comunicação entre as pessoas, etc.) pudessem ser desempenhadas com auxílio de aplicações Web, tornando-as indispensáveis na rotina de várias pessoas. Assim, o projeto de interfaces para sites e aplicações da Web também tornou-se essencial no desenvolvimento de qualquer produto que apoie a diversidade de tarefas realizadas na Internet. A necessidade de que, na Web, interfaces sejam usáveis e acessíveis, se consolida e proporciona cada vez mais pessoas exigindo uma experiência que lhes satisfaça e atenda suas demandas de eficiência e efetividade, durante a navegação e realização de suas tarefas do cotidiano. No contexto da importância de aplicações Web para o cotidiano dos usuários, estudos têm mostrado opções de enriquecimento de interfaces para aplicações na Web, na direção de propiciar interações semelhantes às existentes em ambientes *Desktop*, por meio de *Rich Internet Applications* (RIA, que pode ser traduzido como Aplicações de Internet Rica) [Almeida and Baranauskas 2012], [Fernandes et al. 2013], [Carvalho et al. 2016].

Este minicurso aborda um tema de grande relevância técnica e social para profissionais e pesquisadores envolvidos com o desenvolvimento de aplicações Web em geral, e em especial, de RIAs. É cada vez mais importante promover o uso de princípios de *design* inclusivo no desenvolvimento de sistemas interativos -– e em particular para sistemas Web. Desenvolver sistemas Web acessíveis e usáveis é importante para atingir uma parcela significativa de usuários que possuam alguma deficiência (estimados em cerca de 23,9% da população brasileira, segundo o IBGE (2010) e para atender a requisitos legais.

Realizar avaliações de diversos tipos, priorizando as demandas da variedade de aplicações Web e seus domínios, é reconhecida como atividade fundamental para atingir resultados efetivos no desenvolvimento de sistemas que todas as pessoas realmente possam usar de maneira satisfatória [ISO/IEC 25066 2016]. O desafio de identificar os recursos necessários para acompanhar a evolução tecnológica que emergiu com as aplicações Web, reconhecidas por terem interfaces "ricas", se mostra com grande potencial para condução de estudos a fim de prover aprofundamentos no tema [Bozzon et al. 2006], [Fernandes et al. 2012]. Ao passo que novas tecnologias e possibilidades são implementadas para aplicações Web, torna-se evidente a necessidade de atualização e/ou criação de métodos apropriados para avaliação de tais tecnologias [Doush et al. 2013], [Fernandes 2013], [Watanabe et al. 2012].

O objetivo deste minicurso é disseminar conhecimento sobre as práticas, geralmente utilizadas, para avaliação de usabilidade e acessibilidade, que podem ser aplicadas a interfaces RIA, cosiderando-se o uso de recursos de Tecnologia Assistiva (TA). Sobretudo, são apresentadas as práticas relacionadas a planejamento e organização de testes com usuários, a planejamento e organização de inspeções com especialistas por Avaliação Heurística (AH) e a práticas de revisão de *guidelines* próprias para aplicações ricas (RIA). Nesse sentido, este minicurso abordou como foco principal a utilização correta da especificação WAI-ARIA para desenvolvimento de RIAs mais usáveis e acessíveis, bem como o uso de metodologias para a avaliação do produto desenvolvido em relação a diferentes perfis de usuários.

Este capítulo apresentou uma introdução ao conteúdo preparado para todo o minicurso desenvolvido. As seções seguintes foram organizadas como segue: *Termos e Definições* (Seção 2.2), *Planejamento de Avaliação de Interfaces RIA* (Seção 2.3), *Técnicas de Avaliação de Acessibilidade para RIA* (Seção 2.4) e, ao final, *Conclusões* (Seção 2.5), seção na qual apresentamos uma breve discussão sobre os conhecimentos vistos.

2.2. Termos e Definições

Os conceitos apresentados neste minicurso tornaram-se cada vez mais sólidos como requisitos fundamentais para o *design* e avaliação de uma "boa" interação de usuários com interfaces Web e diferencial profissional para desenvolvedores Web, seguindo a crescente evolução de tecnologias e recursos na Web.

Nas próximas subseções são descritos: princípios fundamentais de interação, com o objetivo de fornecer conceitos básicos sobre Interação Humano-Computador (IHC); definições sobre *User Experience* (UX), usabilidade, acessibilidade e características de soluções RIA. Em seguida, apresentamos o tópico sobre Avaliação de interfaces, detalhando principais tipos de métodos: baseados em usuários e baseados em inspeção.

2.2.1. Princípios Fundamentais de Interação

Este minicurso foi idealizado para ser ministrado a profissionais que desejam aprofundamento em conhecimentos relacionados à área de *design* de interfaces Web usáveis e acessíveis e, também, para estudantes interessados em especializar seus estudos nesta área. Para tanto, entende-se ser necessário fornecer conceitos fundamentais de IHC para que profissionais possam revisá-los e estudantes possam conhecê-los. Estes conceitos não

são específicos das áreas de *Web design* (projeto de Web), mas são amplamente utilizados em diversas áreas do conhecimento como ferramenta para melhorar a compreensão sobre o que deve ser observado na interação entre pessoas e interfaces diversas.

A partir da experiência prévia dos autores com ministração de cursos e aulas sobre conteúdos relacionados a IHC e recursos Web, os autores priorizam, neste minicurso, o conteúdo sobre princípios fundamentais de interação, popularizados por Norman [2013]. Norman apresentou tais princípios na primeira versão de seu livro, então intitulado "*Psicologia do Dia a Dia*" e, posteriormente, intitulado "*Design do Dia a Dia*" [Norman 1988], [Norman 2013].

Os conteúdos apresentados neste minicurso foram retirados da versão revisada do livro "*Design do Dia a Dia*", publicado no ano de 2013. A escolha deste material ocorreu pela relevância da publicação e do autor na área de IHC, bem como da importância do autor na área de desenvolvimento Web, sendo sócio de um dos principais grupos de consultoria da área de interfaces Web, o *Nielsen and Norman Group*[1]. Adicionalmente, destaca-se a necessidade de difundir os conteúdos recém apresentados por Norman (note a data de publicação deste minicurso) durante a versão revisada de seu livro [Norman 2013].

Neste contexto, abordamos o conceito de *Significantes*, apresentado por Norman em "*Living with Complexity*" (Vivendo com a complexidade) [Norman 2010] e destacado pelo mesmo autor na versão revisada de "Design do Dia a Dia", a fim de esclarecer confusões que, segundo Norman, surgiram no mundo do *design* sobre o uso de outro termo importante em princípios de interação: *Affordances*.

Os sete **Princípios Fundamentais de Interação** apresentados por Norman são os seguintes: *Descoberta*, *Feedback*, *Modelo Conceitual*, *Affordances*, *Significantes*, *Mapeamento* e *Restrições*. A seguir, apresentamos breve descrição e exemplificação para cada um destes princípios.

(1) Descoberta - O primeiro princípio destacado por Norman é chamado *Descoberta*. O princípio de *Descoberta* refere-se à necessidade que usuários possuem de, quando interagindo pela primeira vez com uma nova interface, compreender como ela funciona.

O princípio de *Descoberta* é o estado de compreensão do usuário sobre como uma interface funciona, o que ela é capaz de fazer e quais operações que o usuário pode executar com ela [Norman 2013, p. 10].

(2) Feedback - O princípio de *Feedback* pode ser traduzido em Português por parecer (ou retorno). Para profissionais e estudantes de áreas relacionadas à computação, este termo é comumente utilizado a fim de descrever saídas esperadas para algum processamento. No contexto apresentado por Norman, *Feedback* refere-se à existência contínua de informações sobre os resultados das ações conduzidas pelo usuário na interface.

Preferencialmente, o *Feedback* deve aparecer o quão imediato possível após a ação executada pelo usuário na interface [Norman 2013, p. 72]. A Figura 2.1 é uma ilustração

[1] `nngroup.com`

sobre a ocorrência de *Feedback* após a execução da ação do usuário digitar "WWW" e na interface em questão, é então apresentada essa sequencia de caracteres.

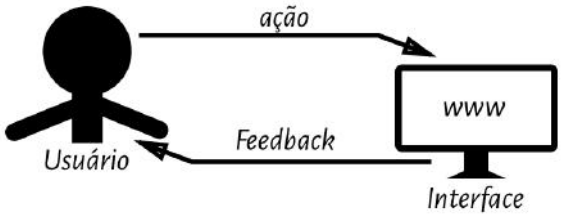


Figura 2.1. Ilustração sobre *Feedback* ocorrendo após a execução de uma ação pelo usuário com a interface.

(3) Modelo Conceitual - O princípio seguinte é chamado por Norman de *Modelo Conceitual*. *Modelo Conceitual* representa uma explicação de como a interface funciona. Segundo Norman, tais explicações são geralmente simplificadas. Quando tais modelos estão na mente do usuário, são chamados *Modelos Mentais*.

Repare que modelos conceituais são explicações existentes, independente de estarem na mente dos usuários [Norman 2013, p. 25-26]. A Figura 2.2 apresenta uma amostra da diversidade de ícones encontrados no site Flaticon[2] a partir da busca pelo termo "menu", em setembro de 2016. Essa variedade de ícones pode refletir a variedade de modelos mentais de usuários quanto ao padrão de funcionamento de menus. Nestes casos, o importante é que os *designers* (profissionais do *design*) conheçam o público de usuários da interface em desenvolvimento a fim de identificar a predominância de *Modelos Mentais* entre os mesmos.

Figura 2.2. Exemplo da diversidade de ícones existentes para representação de menu em software.

(4) Affordances - Dentre os princípios, segundo Norman, o termo de mais destaque entre profissionais da área de *design* foi *Affordances*. Em seu livro, Norman mostrou que *Affordances* são relacionamentos possíveis entre as propriedades da interface e do usuário em questão, ressaltando que o termo vem da palavra *afford* (do Inglês permitir, "ser para") [Norman 2013, p. 11].

Por exemplo, se determinados usuários se deparam com alguma barreira visual que lhes impeça a visualização de uma aplicação Web, então, para esses usuários interagindo com esta aplicação, não existe o *Affordance* "estar visível" da aplicação, sendo necessário o

[2]Licenciado por Flaticon - www.flaticon.com - "*Designed by the authors of this minicourse from Flaticon*"

desenvolvimento de outras possibilidades de interação (por exemplo, código acessível a leitores de tela).

De maneira similar, quando uma interface possui propriedades que impedem algum tipo de relacionamento com as propriedades dos usuários, ocorre a existência de *Anti-affordances* - previnem determinadas interações [Norman 2013, p. 11]. Alguns óculos estereoscópicos possuem lentes diferentes para cada lado dos óculos, cada uma exercendo um diferente tipo de filtro para a percepção do usuário; os impedimentos causados por tais filtros são exemplos de *Anti-affordances*.

(5) Significantes - O termo *Significantes* (do Inglês *Signifiers*) foi introduzido por Norman na versão revisada de seu livro [Norman 2013]. Norman entendeu que era preciso introduzir um termo que auxiliasse a comunidade de *designers* a evitar confusões com o uso do termo *Affordances*.

Segundo Norman, os *designers* precisavam de um termo para designar quais partes uma interface podem ser clicadas, tocadas, etc., e por erro utilizavam o termo *Affordances*. Neste sentido, Norman relatou que alguns *designers* acabavam utilizando afirmações erradas como "coloquei um *Affordance* aqui para indicar onde o usuário poderia digitar o texto". Portanto, para satisfazer a necessidade de um termo que referisse ao local onde a interação deve ocorrer, ele introduziu o termo *Significantes* (em Inglês *Signifiers*).

Assim, *Significantes* indicam onde a ação do usuário deve ocorrer [Norman 2013, p. 13-14]. Norman ressaltou que, durante o processo de *design*, *Significantes* são mais importantes que *Affordances* [Norman 2013, p. 19]. No exemplo mostrado na Figura 2.3, é possível verificar que a aplicação RIA (Google Docs[3]) apresenta uma paleta de cores, na qual cada cor significa onde a ação de seleção de cor (*Significantes*) deve ocorrer para seleção da cor específica para texto à esquerda.

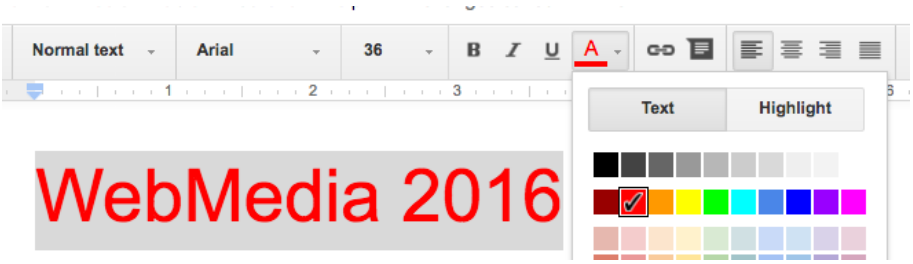


Figura 2.3. Imagem do editor de textos Google Docs, ferramenta de seleção de cores de texto.

(6) Mapeamento - O princípio de Mapeamento refere-se ao relacionamento entre diferentes elementos da interface, assim como a correspondência de causa-efeito entre eles. Um exemplo prático de *Mapeamento* está no mecanismo de rolagem de alguns *mouses* que podem variam a direção (para cima ou para baixo) da rolagem de páginas Web conforme o sistema operacional utilizado, causando problemas de mapeamento para usuários que migram entre sistemas de padrões de mapeamento distintos (vide Figura 2.4).

[3] www.google.com/docs/about/

Mapeamentos podem ser observados também em aplicações que gerenciam casas inteligentes, para que usuários controlem de maneira satisfatória a iluminação de um ambiente específico a partir de seu celular ou *tablet* [Norman 2013, p. 20-21].

Figura 2.4. Foto de um *mouse* para exemplificar *Mapeamento* segundo seu mecanismo de rolagem.

(7) Restrições - O princípio *Restrições* pode ser explicado a partir de quatro vertentes: *Restrições Físicas*, *Restrições Culturais*, *Restrições Semânticas* e *Restrições Lógicas*.

Restrições Físicas implicam na impossibilidade de determinadas interações devido a limites físicos da interface. Por exemplo, usuários podem identificar facilmente qual a conexão do computador eles devem conectar o cabo de vídeo e qual devem conectar o cabo de rede, por causa da impossibilidade física de conectar cabo de rede na entrada de vídeo (e vice-versa) [Norman 2013, p. 125].

Restrições Culturais implicam em questões diferentes entre culturas diferentes [Norman 2013, p. 128-129]. Por exemplo, algumas culturas religiosas podem preferir aplicações de recomendações de refeições que não contenham recomendações sobre determinados alimentos, ou que apresentem indicações para casos em que ocorra a existência de tais alimentos.

Restrições Semânticas variam as possibilidades de ações de acordo com o significado da situação ou tempo [Norman 2013, p. 129-130]. Como exemplo, consideremos carros totalmente autônomos, estes ainda precisam do farol para iluminar a pista? É possível que, nestes casos, o valor semântico dos faróis mude: ao invés de iluminar o caminho (necessário para carros guiados por motoristas humanos), passam a servir para alertar motoristas de carros próximos, sobre a presença do carro autônomo.

Por fim, as *Restrições Lógicas* estão relacionadas com tentativas e erros. Quando usuários estão em dúvida sobre o efeito de suas ações com diferentes elementos da interface, eles podem optar por realizar diversas tentativas. A cada tentativa errada, o número de alternativas diminuirá de maneira lógica [Norman 2013, p. 130]. Ressalta-se que isto não implica na possibilidade de construir interfaces com inúmeras possibilidades de dúvidas, pois o usuário poderá ficar insatisfeito com a aplicação e, até, deixar de usá-la.

2.2.2. *User Experience* - UX

A definição para o termo Experiência de Usuário (do Inglês ***U**ser **Ex**perience* - UX) ainda é bastante discutida na literatura [Bevan et al. 2016]. Um dos trabalhos bem aceitos na literatura mostrou UX como:

> *- a experiência envolvendo os sentimentos dos usuários, assim como os aspectos afetivos e hedônicos relacionados com sua **interação** com interfaces* [Law et al. 2009].

A definição de UX foi apresentada neste minicurso devido à importância do termo na área, e à proximidade deste minicurso com práticas atuais de mercado. De fato, para compreender e considerar UX, tem-se que observar essencialmente as interações dos usuários, mas as questões emocionais dos usuários ampliam sobremaneira o escopo dos possíveis atributos para se avaliar.

O foco deste minicurso foi direcionado aos conceitos relativos a Usabilidade e Acessibilidade, a partir dos quais, tem-se uma perspectiva mais delimitada e relacionada com as tecnologias utilizadas em RIAs.

2.2.3. Usabilidade e Acessibilidade

A **usabilidade** é compreendida como um dos principais atributos de qualidade de interfaces. A ISO/IEC 25066 (2016) apresenta a seguinte definição para usabilidade:

> "*O grau em que um produto pode ser usado por **usuários** específicos para atingir **objetivos** específicos com **efetividade**, **eficiência** e **satisfação** em um **contexto específico de uso***".

Neste contexto, podemos observar com mais detalhes esta definição, a partir dos seguintes aspectos: **usuário** se refere à "*pessoa que interage com*" a interface; **objetivo** se refere à "*saída desejada*"; **efetividade** se refere a "*acurácia e a completude com as quais usuários alcançam objetivos*"; **eficiência** se refere à efetividade e adequada utilização de recursos; **satisfação** se refere a "*ausência de desconforto, e atitudes positivas*" a medida que os usuários utilizam a interface; **contexto específico de uso** se refere à combinação entre perfil dos usuários, tarefas e equipamentos envolvidos e; **tarefas** foram apresentadas como "*atividades requisitadas para atingir um objetivo*"[4].

A **acessibilidade** é mostrada pela ISO/IEC 25066 (2016) a partir de características similares às de usabilidade, porém visando pessoas de uma população com maior diversidade de características e capacidades. Trabalhos recentes mostraram que a acessibilidade é um conceito essencial que visa garantir o acesso e uso a todos os usuários, independente de suas limitações. Assim, acessibilidade na Web corresponde a possibilitar que qualquer usuário, utilizando qualquer agente (software ou hardware que recupera e apresenta conteúdo Web) possa entender e interagir com o conteúdo Web.

[4]Conteúdo disponibilizado *online* pelo portal ISO no endereço: `https://www.iso.org/obp/ui/#iso:std:iso-iec:25066:ed-1:v1:en`

Acessibilidade incorpora ainda a ideia de que todas as pessoas têm o direito de serem incluídas na sociedade, independente de deficiências, localização geográfica, barreiras de linguagem ou outra barreira [Thatcher et al. 2002].

Outra definição para acessibilidade bastante aceita na área foi mostrada pela *Web Accessibility Initiative* (WAI), organização criada pelo W3C, na qual a acessibilidade na Web consiste em [W3C 2005]:

> "*websites e softwares que atendam às diferentes necessidades, preferências e situações dos usuários. Especificamente, a acessibilidade permite que pessoas com deficiência possam perceber, compreender, navegar e* ***interagir*** *com a Web, e assim podem contribuir com a mesma*."

2.2.4. *Rich Internet Applications*

A Web foi elaborada com o objetivo de oferecer um repositório multiplataforma e interoperável de dados e informações. Para permitir que o conteúdo pudesse ser acessado por quaisquer conjuntos de software e hardware que os usuários estivessem operando, foram especificadas a linguagem de marcação para os conteúdos *HyperText Markup Language* (HTML) e o *HyperText Transport Protocol* (HTTP), como um mecanismo para transmitir arquivos textuais entre diferentes plataformas e sistemas operacionais conectados em uma rede [Berners-Lee 1990].

Desde sua criação, a Web tem passado por um constante processo evolutivo, proporcionando espaços mais dinâmicos e flexíveis de interação, de modo que usuários sem conhecimentos técnicos da área de computação também participem e modifiquem os conteúdos disponíveis. A partir desses novos usuários, surgiu o conceito de Web 2.0, que estabelece que o seu conteúdo não é mais proprietário e disponibilizado por pequenos grupos de pessoas. Esse conteúdo deve ser decidido por toda a população presente na Web, que passou a desempenhar co-autoria, além de consumir os serviços e conteúdos disponibilizados [van Wamelen and de Kool 2008].

A capacidade de contribuir com o conteúdo Web e interagir com os demais usuários da Internet mudou drasticamente o panorama da Web em um curto período de tempo, deixando de ser um ambiente passivo, caracterizado por páginas estáticas, para se tornar um ambiente ativo e passível de rápidas atualizações, repleto de páginas dinâmicas e interativas [Gehtland et al. 2006]. Novas aplicações Web, tais como *blogs*, *Social Bookmarking*, *wikis*, *podcasts*, *RSS feeds*, serviços online como *Facebook*[5] e *Gmail*[6] proporcionaram melhorias expressivas sobre os sites estáticos. Com isso, a Web tornou-se um dos principais canais de informação e comunicação entre as pessoas.

As características de maior interatividade são, muitas vezes, implementadas utilizando tecnologias abertas de Internet, tais como o HTML, CSS, JavaScript, DOM, AJAX, entre outras, nas chamadas *Rich Internet Applications* (RIA). Vale observar que HTML é a linguagem de marcação padrão da Web, responsável pela representação dos dados e estrutura dos conteúdos disponibilizados; CSS é a linguagem de folhas de estilo para definir-se

[5]http://www.facebook.com
[6]http://www.gmail.com

a formatação e layout das páginas Web; JavaScript é uma linguagem de programação que propicia a lógica de processamento dos dados (consistências, validação por ex.) no lado do cliente. As tecnologias DOM e AJAX, por serem especialmente importantes no escopo deste minicurso, serão descritas adiante.

Para Lawton (2008), a principal finalidade de RIAs é proporcionar uma experiência de acesso às aplicações Web de forma semelhante à que ocorre com as aplicações *desktop*. Segundo [Hooshmand et al. 2016], a utilização das características de RIA tornou-se padrão no desenvolvimento aplicações Web modernas.

Segundo Mesbah, Deursen e Roest (2012), alguns dos motivos para ampla adoção de RIAs são:

- não requerem a instalação no lado do cliente;
- todo mundo utiliza a versão mais recente;
- possibilidade de interação a partir de qualquer lugar com acesso à Internet, e;
- novas oportunidades de construção colaborativa e de comunidades.

RIAs combinam funcionalidades de interfaces do usuário de aplicações *desktop* com o amplo alcance do desenvolvimento de aplicações Web, além do melhor da interatividade e comunicação multimídia [Casteleyn et al. 2014]. O resultado final é uma aplicação que oferece uma experiência de usuário mais intuitiva, ágil e eficaz. Atualmente, parte significativa dos *websites* são RIAs ou apresentam componentes (*widgets*) que implementam as características de uma RIA. Os editores colaborativos, redes sociais e acompanhamento ao vivo de elementos são alguns exemplos populares de RIAs [Hooshmand et al. 2016].

As novas possibilidades de interação enriquecidas, que surgiram a partir de RIAs, são essenciais para apoiar os princípios relevantes da Web 2.0, como a participação e colaboração. Por outro lado, existe uma preocupação crescente sobre a acessibilidade e usabilidade dessas aplicações, uma vez que esses novos recursos de interação disponíveis na Web nem sempre são acessíveis ou são usados de maneira eficiente por todas as pessoas [Mori et al. 2011],[Casteleyn et al. 2014].

O aumento da interatividade tem exigido cada vez mais dos agentes de usuários (navegadores Web) e de recursos de TA, uma vez que a interação em RIAs deve se manter acessível aos usuários com deficiências ou com limitações temporárias [Machado Neto et al. 2014]. As alterações e atualizações na estrutura de uma aplicação Web geradas dinamicamente pelo JavaScript, no entanto, nem sempre são percebidas pelos usuários que interagem por meio de recursos de TA [Almeida and Baranauskas 2012],[Carvalho et al. 2016].

Tradicionalmente, tais recursos de TA apresentam informações, considerando que a página Web possui uma estrutura linearizada de conteúdo; porém as RIAs, muitas vezes, não possuem uma estrutura linearizada, permitindo que a estrutura da página Web (inclusive a árvore gerada por meio do *Document Object Model* – DOM) seja modificada durante a interação [Doush et al. 2013],[Fernandes et al. 2013].

A estrutura de dados "árvore", gerada pelo DOM, é utilizada pelos navegadores Web para representar toda a estrutura da página Web, ou seja, ela corresponde à estrutura de dados referente aos elementos e conteúdos HTML que são armazenados em RAM, durante a renderização de uma página no *browser*/navegador.

Assim, quando essa árvore é alterada com o uso do JavaScript, altera-se também a apresentação da página Web, seja a sua estrutura ou seu conteúdo. Essa alteração é uma das características de RIAs, que se não implementada de maneira correta pode ocasionar problemas de acessibilidade e usabilidade, em especial para os usuários de recursos de TA[7].

2.2.5. Avaliação

A literatura apresenta diversos métodos para a avaliação de usabilidade e/ou acessibilidade de interfaces. Segundo a ISO/IEC 25066 (2016), tais métodos podem ser agrupados em duas categorias: *Métodos Baseados em Usuários* e *Métodos Baseados em Inspeção*.

As subseções seguintes apresentam os exemplos de métodos abordados neste minicurso, segundo sua respectiva categoria.

2.2.5.1. Métodos Baseados em Usuários

A ISO/IEC 25066 (2016) apresenta *Métodos Baseados em Usuários* como métodos que envolvem a participação de uma amostra representativa de usuários reais da interface. Testes envolvem a observação de usuários reais na realização de tarefas pré-estabelecidas a partir da interface considerada na avaliação. Para tanto, técnicas e ferramentas adicionais podem ser consideradas, como o protocolo *Think Aloud* (do Inglês "pensando em voz Alta"), ferramentas de gravação de vídeo e de *Eye-tracking* (do Inglês "Perseguindo o Olhar") [Ericsson and Simon 1980], [Dix et al. 2003], [Preece et al. 2015].

Este minicurso preparou a apresentação de um modelo genérico para planejamento e condução de testes com usuários, apresentados na Seção 2.3.2. Este modelo de planejamento genérico foi definido, considerando-se os principais cuidados relativos a realização de testes com usuários com deficiência [Freire et al. 2013].

2.2.5.2. Métodos Baseados em Inspeção

De acordo com a ISO/IEC 25066 (2016), *Métodos Baseados em Inspeção* envolvem o julgamento de inspetores com base em critérios pré-determinados. A norma mostra que estes inspetores podem variar de especialistas em usabilidade, usuários finais do sistema a outros tipos de profissionais.

Este minicurso aborda dois dos métodos de inspeção de usabilidade e/ou acessibilidade mais populares na literatura: Avaliação Heurística e Revisão de *Guidelines* [Følstad et al. 2012, Petrie and Power 2012, Martins et al. 2014]. A seguir estão descritos cada um destes métodos.

[7]https://developer.mozilla.org/

(a) Avaliação Heurística – O método de Avaliação Heurística (AH) foi desenvolvido por Nielsen e Molich (1990) . A AH envolve o julgo de avaliadores especialistas em usabilidade sobre a conformidade entre a interface avaliada e um conjunto de princípios amplos sobre usabilidade (ditos heurísticas). Inicialmente, Nielsen e Molich desenvolveram um conjunto de 9 heurísticas que, posteriormente, foram refinadas e reorganizadas em 10 heurísticas por Nielsen em trabalhos posteriores [Nielsen 1994a], [Nielsen 1994b]. No Apêndice A encontra-se a lista das 10 heurísticas de Nielsen.

(b) Revisão de *Guidelines* – O método de Revisão de *Guidelines* envolve a verificação de conformidade entre a interface avaliada e um conjunto de diretrizes (do Inglês *Guidelines*) sobre usabilidade e/ou acessibilidade. Ao contrário de heurísticas, *guidelines* representam situações conhecidas e específicas que, caso não atendidas, ocorrem na existência de problemas de usabilidade e/ou acessibilidade [Paz and Pow-Sang 2016].

Neste minicurso, abordaremos um conjunto de diretrizes da *Accessible Rich Internt Applications* (WAI-ARIA 1.0)[8] que busca promover a acessibilidade de RIAs em complemento a *Web Content Accessibility Guidelines* (WCAG 2.0)[9]. Por meio da WAI-ARIA, é possível aumentar a semântica do conteúdo que está sendo apresentado visualmente (por exemplo, identificar que uma lista de itens está sendo usada como um menu ou que uma lista de links está sendo usada como uma lista em formato de árvore), bem com identificar regiões da página Web em que ocorrem atualizações automáticas ou por meio da interação do usuário (por exemplo, validações automáticas, atualizações do placar de um jogo, galeria de fotos apresentadas em um *widget*, entre outros) de maneira que o navegador Web possa avisar a API de acessibilidade sobre as atualizações dinâmicas no momento em que ocorrem.

Manter a API de acessibilidade atualizada, especialmente sobre as atualizações constantes que ocorrem em RIAs, é extremamente importante do ponto de vista dos usuários que utilizam recursos de TA para interagir com a aplicação, uma vez que sua interação pode ser seriamente prejudicada.

2.3. Planejamento de Avaliação de Interfaces RIA

A organização de avaliações de interfaces RIA possui peculiaridades que as tornaram mais sensíveis que a organização de avaliações em sistemas Web tradicionais. Em todo contexto de avaliações de interfaces, é importante conhecer as características do perfil de usuários que deseja-se abranger [Preece et al. 2015, p. 265]. Nestes casos, usuários diferem entre si, porém características em comum podem ser levantadas por técnicas apropriadas de compreensão do perfil de usuário (como questionários e entrevistas). Entretanto, RIAs podem ser ainda mais sensíveis que aplicações tradicionais devido às amplas possibilidade de personalização de conteúdo do DOM feita pelos usuários (mais detalhes apresentados na Seção 2.4) [Doush et al. 2013],[Fernandes et al. 2013].

As diferenças entre usuários do grupo desejado são ressaltadas em RIAs pela sua própria natureza adaptativa às preferências e usos dos usuários [Doush et al. 2013],[Fer-

[8] https://www.w3.org/TR/wai-aria/
[9] https://www.w3.org/TR/WCAG20/

nandes et al. 2013]. Assim, é intuitivo compreender que avaliações de interfaces RIA com base em pouca automatização precisam, consequentemente, de organizações bem estruturadas para minimizar a perda de informações importantes sobre usabilidade e acessibilidade da aplicação. Diante da necessidade de elaboração de um planejamento para avaliação de interfaces RIA, foi utilizada a perspectiva prática organizacional. Nesse contexto, preparou-se dois roteiros (passo-a-passo) de atividades para a organização de dois dos principais métodos de avaliação: **AH** e **Testes com usuários** [Følstad et al. 2012], [Martins et al. 2014]. As seções seguintes apresentam cada um desses modelos, e respectivas descrições.

2.3.1. Roteiro para Preparação de Avaliações Heurísticas

Esta seção apresenta uma proposta de roteiro, cujo objetivo é didático, para organização de inspeções de interfaces por AH. O passo-a-passo foi desenvolvido inicialmente a partir da tabela "Atividades do método de avaliação heurística" apresentada por Barbosa e Da Silva (2010, p. 318). Em seguida, essa tabela foi adaptada para contemplar abordagens adicionais apresentadas na literatura [Nielsen 1995],[Preece et al. 2015].

O roteiro está estruturado nas seguintes etapas: *Pré-inspeção* - Tabela 2.1; *Inspeção* - Tabela 2.2 e *Pós-inspeção* - Tabela 2.3. Para cada etapa, preparou-se uma tabela explicativa contendo o nome da etapa, os passos identificados para a organização da mesma segundo a literatura consultada e a finalidade principal de cada passo elencado. Cabe ressaltar que os passos nas linhas hachuradas e marcados com um asterisco (*), à frente de sua descrição, são considerados pelos autores como de maior importância.

Tabela 2.1: Passos da Etapa Pré-inspeção para organização de AHs e descrição de suas finalidades.

Etapa 1 – Pré-inspeção	**Documentação das características dos usuários:** visa o conhecimento sobre os usuários.
	Documentação das características da aplicação: visa o conhecimento sobre a aplicação.
	Documentação de cenário: visa o conhecimento sobre usuários utilizando a aplicação.
	(*)Descrição das tarefas sob avaliação: visa o conhecimento sobre tarefas abordadas durante a avaliação.
	Descrição de escopo da aplicação: visa o conhecimento das partes da aplicação que serão envolvidas na avaliação.
	(*)Convite a avaliadores: recomenda-se múltiplos avaliadores (pelo menos 3) especialistas no domínio da aplicação ou em IHC.
	(*)Definição do(s) observador(es): podem ser profissionais já envolvidos no projeto, com conhecimento sobre a aplicação.
	Definição de canal de comunicação: listas de e-mails, telefones, etc., para comunicação entre avaliadores e organizadores.
	(*)Agendamento da avaliação: definição de data, horário e local para a avaliação segundo disponibilidade dos avaliadores.
	Preparação e verificação de aparato: preparar local da avaliação e verificar funcionamento de todo o aparato necessário (computadores, rede, etc.)
	(*)Definição do conjunto de heurísticas: escolher o conjunto de heurísticas mais adequado à aplicação. Recomenda-se envolver os avaliadores neste passo.

Tabela 2.2: Passos da Etapa Inspeção para organização de AHs e descrição de suas finalidades.

Etapa 2 – Inspeção	**Compreensão de escopo aplicação:** observadores disponíveis para sanar possíveis dúvidas dos avaliadores quanto a aplicação. Avaliadores percorrem a interface para compreender seu funcionamento.
	(*)Inspeção: inspeção da interface segundo conjunto de heurísticas considerado. Cada problema levantado pode estar relacionado a uma ou mais heurísticas do conjunto considerado.
	Anotações por observadores: observadores podem anotar os problemas mencionados pelos avaliadores, deixando-os mais dedicados à inspeção.

Tabela 2.3: Passos da Etapa Pós-inspeção para organização de AHs e descrição de suas finalidades.

Etapa 3 – Pós-inspeção	**(*)Revisão colaborativa de existência de problemas:** avaliadores se reúnem para discutir a existência dos problemas levantados durante a inspeção.
	(*)Atribuição colaborativa de severidade: avaliadores discutem e definem juntos a severidade de cada problema levantado.
	Definição de sugestões: os avaliadores listam as sugestões de melhorias na interface, citando os respectivos problemas a corrigir.
	(*)Elaboração de relatório final: avaliadores elaboram o relatório final de avaliação contendo a lista de problemas (descrição, heurística(s) afetada(s), elemento da interface envolvido, severidade, etc.) e soluções levantadas (quando identificadas).
	Manutenção do canal de comunicação: manter a comunicação entre organizadores e avaliadores após a avaliação para possíveis questionamentos e explicações.

Informações detalhadas sobre a execução de AHs podem ser encontradas no portal *Nielsen and Norman Group*[10]. A seção seguinte apresenta roteiro similar voltado para o planejamento e organização de sessões de avaliação por Testes com Usuários.

2.3.2. Roteiro para preparação de Testes com Usuários

Assim como feito para o roteiro de preparação de AHs, os autores deste minicurso prepararam um roteiro passo-a-passo, com finalidade didática, para a organização de testes com usuários. Similarmente, o roteiro de organização de testes com usuários foi elaborado com base na tabela de atividades de testes com usuários apresentada por Barbosa e Da Silva (2010, p. 342) . Foi também realizada uma adaptação, para contemplar experiências prévias dos autores com organização de testes com usuários, e também a partir de referências adicionais obtidas na literatura [Preece et al. 2015],[Usability.gov 2016a], [Usability.gov 2016b], [Nielsen 2001].

O roteiro sobre Testes com Usuários também está organizado em três sessões:

[10]https://www.nngroup.com/articles/how-to-conduct-a-heuristic-evaluation/

Pré-teste, *Teste* e *Pós-teste*. Assim como o roteiro para AHs, apresentamos uma tabela explicativa contendo nome da etapa, passos para a organização da mesma segundo a literatura consultada e descrição da finalidade principal de cada passo elencado pelos autores. Estas tabelas também apresentam passos marcados com um asterisco (*) a frente, significando os passos considerados pelos autores como de maior importância em cada etapa. Assim, preparamos as seguintes tabelas: Tabela 2.4 apresenta conteúdo da etapa *Pré-teste*; Tabela 2.5 apresenta conteúdo da etapa *Teste* e; Tabela 2.3 apresenta conteúdo da etapa *Pós-teste*.

Tabela 2.4: Passos da Etapa Pré-teste para organização de Testes com Usuários e descrição de suas finalidades.

Etapa 1 – Pré-teste	**(*)Descrição das tarefas sob avaliação:** visa o conhecimento sobre tarefas abordadas durante a avaliação.
	(*)Definição do(s) observador(es): podem ser profissionais já envolvidos no projeto, com conhecimento sobre a aplicação e sobre IHC.
	Treinamento e formação do(s) observador(es): análises de custo benefícios podem indicar se formação de observadores dentro da empresa é mais viável comparado à contratação de observadores disponíveis no mercado.
	(*)Definição de perfil dos participantes: Conhecimento sobre os usuários.
	(*)Agendamento individual da avaliação: Definição de data, horário e local para a avaliação segundo disponibilidade dos usuários.
	Definição de métricas de avaliação: Taxa de sucesso, taxa de erros, tempo necessário para completar a tarefa e/ou satisfação dos usuários.
	Descrição de escopo da aplicação: Conhecimento das partes da aplicação que serão envolvidas na avaliação.
	Definição de questões de teste: O que se deseja responder com os testes.
	Documentação de cenário: Conhecimento sobre usuários utilizando a aplicação.
	(*)Revisão de ética: Atender à regulamentações sobre aspectos éticos de pesquisas envolvendo pessoas.
	Preparação e verificação de aparato: Preparar local da avaliação e verificar funcionamento de todo o aparato necessário (computadores, rede, etc.)
	Definição de canal de comunicação: Listas de e-mails, telefones, etc., para comunicação entre participantes e organizadores.

Tabela 2.5: Passos da Etapa Teste para organização de Testes com Usuários e descrição de suas finalidades.

Etapa 2 – Teste	**(*)Verificação de Consentimento:** documentar ciência e concordância dos usuários sobre participação nos testes.
	Acompanhamento de cuidado e conforto: garantir que todos os participantes recebam o conforto e cuidado digno, não expondo usuários à situações prejudiciais de nenhuma natureza.
	Configuração de software de apoio: configurar softwares de apoio caso utilizados, como eye-trackers, gravadores de voz, câmeras, gravação de tela, etc.
	(*)Teste piloto: realizar testes pilotos com um número reduzido de usuários a fim de obter indícios sobre a efetividade e eficiência dos testes, evitando custos desnecessários.

	(*)Teste: condução dos testes como planejados, considerando possíveis adaptações levantadas durante o teste piloto.
	(*)Anotações por observador(es): observadores devem tomar notas sobre os problemas de usabilidade ocorridos durante as interações dos participantes com a aplicação nos testes.

Tabela 2.6: Passos da Etapa Pós-teste para organização de Testes com Usuários e descrição de suas finalidades.

Etapa 3 – Pós-teste	**Questionários pós-avaliação:** coleta de informações adicionais sobre satisfação e/ou outras características da interação.
	Análise de software de apoio: análise dos dados coletados por software de apoio. Deve ser feita por profissionais com conhecimento sobre IHC e sobre a ferramenta.
	Análise de correlação entre dados: analisar correlação entre dados de diferentes fontes (notas de observadores e software de apoio).
	(*)Elaboração de relatório final: observadores elaboram o relatório final de avaliação contendo a lista de problemas observados e soluções possíveis (quando identificadas).
	Manutenção do canal de comunicação: manter a comunicação entre organizadores e avaliadores após a avaliação para possíveis questionamentos e explicações.

Informações detalhadas sobre a execução de Testes com Usuários podem ser encontradas no portal do governo Americano, *Usability.gov*[11]. É importante também destacar que os cuidados durante a realização dos testes com usuários com deficiências devem ser criteriosamente considerados [Freire et al. 2013].

A seção seguinte descreve técnicas para avaliações de interface RIA que dependem de menor participação de usuários, por estarem especialmente relacionadas a aspectos tecnológicos.

2.4. Técnicas de Avaliação de Acessibilidade para RIA

De acordo com o relatório de setembro/2016 do SecuritySpace[12], mais de 66% de todos os *websites* são implementados utilizando JavaScript. Entretanto, o desenvolvimento dessas aplicações, quando não são consideradas as questões de acessibilidade e usabilidade, afeta dramaticamente a capacidade das pessoas com deficiência acessarem o conteúdo da Web [Steen-Hansen and Fagernes 2015].

As aplicações de interfaces gráficas mais antigas utilizavam APIs de acessibilidade próprias que interoperavam com recursos de TA, a fim de solucionar ou reduzir minimamente os diversos problemas de acesso e utilização dessas aplicações. Contudo, as implementações atuais de RIAs, que utilizam puramente DHTML[13], não suportam a utilização de tais APIs. Para tanto, o W3C, por meio da *Web Accessibility Initiative*

[11] `https://www.usability.gov/how-to-and-tools/methods/running-usability-tests.html`

[12] `http://www.securityspace.com/s_survey/data/man.201608/techpen.html`

[13] *Dynamic* (dinâmico em Português) HTML, ou DHTML, é a união das tecnologias: HTML, JavaScript e uma linguagem de apresentação, como folhas de estilo, aliada a um (DOM), para permitir que uma página

(WAI), tem trabalhado na implementação de recomendações para desenvolvimento de RIAs acessíveis.

Nesse sentindo, uma das principais atividades para melhorar a acessibilidade de *websites* é a utilização de métodos que revelem problemas e barreiras de acessibilidade para que possam ser resolvidos durante o desenvolvimento. Existem métodos que podem ser utilizados, incluindo métodos de inspeção, realizados por especialistas, ou testes envolvendo usuários com deficiência, conforme descritos na Seção 2.3.

Os métodos de inspeção manual por especialistas têm um papel importante no processo de avaliação de *websites*. Para tanto, este minicurso orienta os desenvolvedores na inspeção de aplicações Web, tendo como base o conjunto de documentos que integram a *Accessible Rich Internet Applications* (WAI-ARIA). Esse conjunto de documentos define uma maneira para tornar as aplicações Web e seus respectivos conteúdos mais acessíveis a pessoas com deficiência, e visa auxiliar, especialmente, a criação de conteúdo dinâmico e controles avançados nas interfaces, as quais são desenvolvidas com Ajax, HTML, JavaScript ou tecnologias relacionadas.

A especificação técnica da WAI-ARIA é uma recomendação oficial do W3C desde março/2014, voltada principalmente para desenvolvedores de *browsers*, recursos de TA e outros agentes do usuário; desenvolvedores de tecnologias Web; e desenvolvedores de ferramentas de avaliação de acessibilidade. Um dos objetivos para se adotar a WAI-ARIA, durante o desenvolvimento das páginas Web, é fornecer semânticas adequadas para elementos dinâmicos (*widgets*), de modo a torná-los acessíveis, utilizáveis e interoperáveis com o uso de recursos de TA. Por meio dessa especificação, é possível identificar os tipos de *widgets* e suas estruturas, fornecendo uma ontologia das funções correspondentes que podem ser incorporadas ao conteúdo.

Em outras palavras, os elementos da linguagem de marcação utilizada que estejam definidos por meio de determinado *role* podem ser interpretados como elementos específicos, independentemente de qualquer semântica herdada da linguagem hospedeira de execução, como por exemplo do HTML. A Figura 2.5 ilustra o funcionamento de um acesso a uma RIA que inclui a marcação da WAI-ARIA.

Como mostrado na Figura 2.5, é possível observar que o relacionamento entre o navegador Web e os recursos de TA ocorre por meio de uma API de acessibilidade. Esse relacionamento possibilita o compartilhamento das informações comuns (*roles*, estados, eventos notificações, relações de informações e descrições) dos dados da aplicação que precisam ser processados quando o usuário requer meios especiais para acesso.

O navegador Web é encarregado de transmitir as informações relevantes para essa API de acessibilidade, conforme o usuário interage com a aplicação Web, e elas podem ser usadas por qualquer recurso de TA para transmitir o conteúdo de maneira acessível ao usuário.

Nesse contexto, para que de fato a acessibilidade possa ser alcançada, a especificação ARIA define um conjunto de atributos que os elementos HTML que compõem a implementação de uma determinada *widget* deve incluir. Esses atributos adicionam dados

Web seja modificada dinamicamente na própria máquina cliente, sem necessidade de novos acessos ao servidor web.

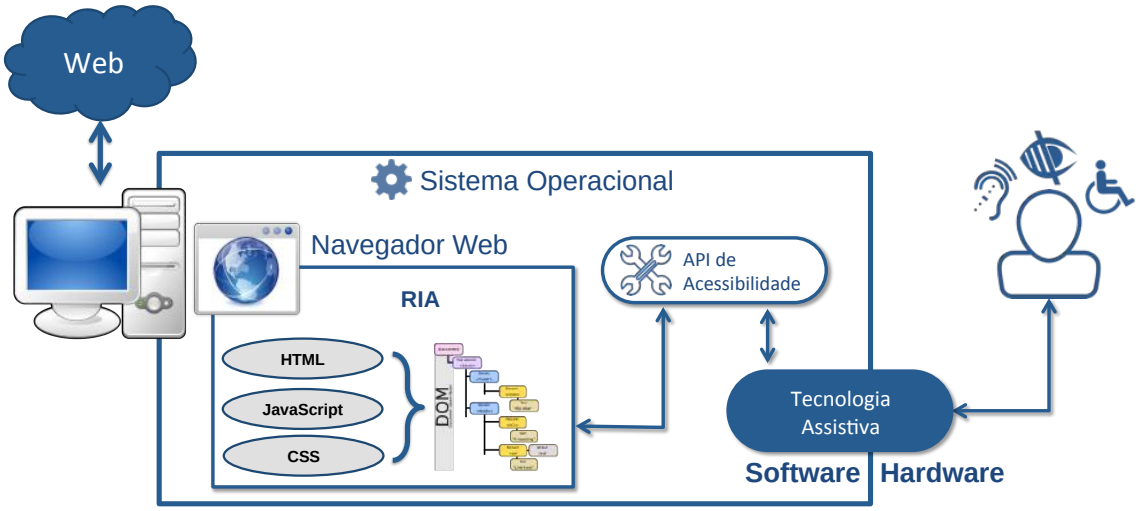


Figura 2.5. Esquema de interação em RIAs por pessoas com deficiência

semânticos aos elementos de uso genérico (tais como e) e disponibilizam informações sobre o comportamento das *widgets* aos agentes de usuário e recursos de TA.

Esse conjunto de atributos da WAI-ARIA compreende as seguintes extensões de tecnologias Web que devem ser incorporadas às tecnologias utilizadas em RIAs, atualmente [W3C 2014]:

- ***Role:*** permite que o desenvolvedor anote um determinado elemento HTML com informações semânticas sobre o seu comportamento como um componente de interface por meio do atributo `role`. É importante ressaltar que este minicurso não tem como finalidade a descrição de toda a especificação WAI-ARIA. Portanto, será introduzida a contextualização e apresentação do principais conceitos, abordando apenas alguns dos tipos de *roles*. A lista completa está disponível na especificação da WAI-ARIA, em que é possível encontrar a finalidade de cada *role*, as propriedades e estados aplicáveis a ela, bem como sua relação com as demais. Segundo especificação da WAI-ARIA, os *roles* são categorizados da seguinte maneira:
  - ***Roles* Abstratos** – são utilizados apenas para definição de conceitos abstratos da taxonomia, servindo como base para construção de todos os demais *roles*. Portanto, os desenvolvedores não devem fazer uso desses *roles*;
  - ***Roles* de componentes de interface (*widgets*)** – são usados para definir componentes de interface de usuário (por exemplo, `alert`, `menuitem`, `tooltip` e `treeitem`) ou partes de componentes compostos de interface de usuário (por exemplo, `combobox` e `treegrid`). Além disso, alguns desses *roles* podem ser classificados como regiões de conteúdo "ao vivo", em que ocorre atualização automática de conteúdo como resultados de eventos externos, ou seja, não depende da interação do usuário. Exemplos são: marquee, status e *timer*.;
  - ***Roles* de estrutura de documento** – são utilizados para definir estruturas que organizam uma página Web. Exemplos: `article`, `heading` e `note`;

  - ***Roles* de pontos de referência (*landmarks*)** – são regiões em uma página Web compostas por pontos importantes, as quais um usuário pode querer acessá-las rapidamente. Exemplos são: `main`, `navigation` e `search`;

- **Estados** e **Propriedades**: são usados para declarar atributos de um elemento. Assim, os recursos de TA e sistemas operacionais são capazes de manipular o elemento, mesmo quando ele é alterado de forma assíncrona, por vezes, sem a interação do usuário. Os atributos utilizados para representar estados e propriedades se referem a funcionalidades similares, uma vez que ambos proveem informações específicas sobre um objeto e fazem parte da natureza de uma *role*. De acordo com a especificação WAI-ARIA, os estados e propriedades são tratados como atributos prefixados com `aria-`. No entanto, os dois conceitos (estados e propriedades) são mantidos separados, visto que os valores de propriedades são menos prováveis de serem alterados dinamicamente em comparação com os valores de estados, os quais são frequentemente alterados em função da interação dos usuários.

A seguir é apresentado um exemplo de implementação em HTML do *widget Tab*, utilizando os atributos da especificação WAI-ARIA. Assim, os diferenciais dessa implementação que obedecem á especificação WAI-ARIA estão na inclusão dos atributos: `role`, `aria-controls` e `aria-labelledby`.

A apresentação visual permanecerá a mesma caso os atributos da WAI-ARIA não sejam definidos, porém o uso desses atributos melhora a interação dos usuários de recursos de TA, uma vez que fornecem informações adicionais que não estariam acessíveis sem a WAI-ARIA.

Além disso, é importante ressaltar que apenas a especificação dos atributos no código HTML não garante a acessibilidade, visto que deve-se também implementar os controles de interação dinâmicos em uma linguagem de *script*, conforme descrito na especificação WAI-ARIA.

```
1  <!-- Tabs em que cada link no topo abre uma tab de conteúdos
       em baixo -->
2  <ul class="tablist" role="tablist">
3    <li id="tab1" class="tab" aria-controls="panel1" role="tab"
       >Frutas</li>
4    <li id="tab2" class="tab" aria-controls="panel2" role="tab"
       >Vegetais</li>
5  </ul>
6
7  <div id="panel1" aria-labelledby="tab1" role="tabpanel">
8    <h3>Frutas</h3>
9    <ul>
10     <li>Maça</li>
11     <li>Laranja</li>
12     <li>Banana</li>
13   </ul>
14 </div>
15
```

```
16 <div id="panel2" aria-labelledby="tab2" role="tabpanel">
17   <h3>Vegetais</h3>
18   <ul>
19     <li>Batata</li>
20     <li>Cenoura</li>
21     <li>Cebola</li>
22   </ul>
23 </div>
```

A implementação de RIAs, seja na totalidade do *website* ou em partes específicas, requer que os desenvolveres definam o comportamento dos componentes dinâmicos, dadas as múltiplas possibilidades de interação por parte do usuário. Além disso, é necessário associar os componentes de interface com um atributo *role*, que mapeie uma funcionalidade semelhante para a qual cada componente foi projetado, e atribuir os respectivos estados e propriedades que são requisitos dessa *role*, utilizando a linguagem de *script* apropriada.

Todos os elementos dinâmicos devem ter um esquema bem definido de navegação por teclado (por meio da gerência de foco na aplicação Web), considerando-se a possibilidade de utilização de TA. Finalmente, esses elementos dinâmicos devem ser incluídos na estrutura geral da página Web, considerando a região em que o usuário interagiu, e deverão ser apresentadas em meio aos demais elementos existentes [W3C 2014].

Vale destacar que o processo de desenvolvimento de RIAs é bastante especializado, possibilitando uma grande variedade de recursos que podem ser explorados nas diversas formas de interação da Web 2.0. A adição dos atributos da especificação WAI-ARIA nos elementos da página Web não implica em alterações visuais. Assim, testar RIAs por observação pode ser frustrante. No entanto, há algumas técnicas que podem ser utilizadas.

O uso de um ou mais recursos de TA que tenham dependência da marcação WAI-ARIA para apresentar o conteúdo de uma RIA pode ser utilizado para verificar sua acessibilidade. Os sistemas operacionais atualmente oferecem, em sua maioria, um leitor de tela nativo, porém existem outras alternativas gratuitas, como por exemplo, o NVDA[14]. Ademais, existem diversas soluções comerciais de leitores de tela que oferecem versões limitadas, mas gratuitas para testes, como o JAWS[15].

O uso de leitores de tela, no entanto, pode ser lento a princípio e cansativo a longo prazo. Em geral, os leitores de tela oferecem um recurso para habilitar a saída textual dos texto que seria pronunciado pelo sintetizador de voz, tornando-se possível analisar o conteúdo sem a necessidade de ficar ouvindo a leitura repetidas vezes. Por exemplo, no NVDA, basta selecionar o menu *Ferramentas* e depois habilitar a opção *Exibidor de fala*. Outra estratégia é utilizar um software para emulação de leitores de tela, como Fangs[16], disponível para o navegador Mozilla Firefox.

[14] http://www.nvaccess.org

[15] http://www.freedomscientific.com/Products/Blindness/JAWS

[16] https://addons.mozilla.org/pt-br/firefox/addon/fangs-screen-reader-emulator/

Cabe ressaltar que, antes de verificar se os recursos de TA estão apresentando o seu conteúdo corretamente, é altamente recomendada a verificação se o código-fonte atende aos padrões de linguagem de marcação, folhas de estilo e *scripts*. A WAI-ARIA proporciona orientações gerais que auxiliam na implementação acessível de RIAs, mas se mostra desprovida de formas para se garantir o cumprimento de suas especificações de implementação tecnológica [Watanabe 2014]. Portanto, o desenvolvedor deve ser responsável pela condução de inspeções manuais e testes com usuários para avaliação da acessibilidade e usabilidade das aplicações em conformidade com as recomendações da especificação WAI-ARIA.

Recomendações gerais para uso da especificação WAI-ARIA

O entendimento claro de quando e como adicionar os atributos definidos da especificação WAI-ARIA é tão importante quanto compreender quais são os *roles*, propriedades e estados existentes, uma vez que nem sempre é necessário o uso desses atributos na página Web. Para tanto, o W3C [W3C 2015] definiu cinco recomendações básicas para utilização:

Regra 1: Dar preferência ao uso de elementos ou atributos nativos da linguagem HTML, os quais apresentem a semântica e o comportamento desejado, em vez de utilizar os atributos da WAI-ARIA. Contudo, existem algumas exceções em que o uso apenas do HTML não é possível:

- o elemento ou atributo definido pela linguagem HTML ainda não está implementado por navegadores Web ou esse não é suportado por recursos de TA;
- as restrições do *design* visual excluem a utilização de um elemento nativo em particular, pois esse não pode ser personalizado como desejado;
- o recurso desejado não está disponível no HTML.

Regra 2: Não alterar a semântica de elementos nativos do HTML, a menos que seja estritamente necessário. Entretanto, se um elemento HTML estático for usado como base de um elemento interativo com a marcação WAI-ARIA adequada, será necessário adicionar *scripts* com todo o comportamento dinâmico desejado, incluindo a interação apenas pelo teclado.

Regra 3: Todos os controles interativos da especificação WAI-ARIA devem ser usáveis pelo teclado. Em outras palavras, em uma *widget* que permite ao usuário clicar, arrastar, tocar, rolar, entre outras maneiras de interação, deve-se fornecer a mesma capacidade de interação nessa *widget* apenas com o uso do teclado. Para isso, é necessário a criação de *scripts* que respondam aos pressionamentos de teclas ou combinações de teclas, quando aplicável.

Regra 4: Não utilizar `role=presentation` ou o estado `aria-hidden="true"` em um elemento visível e que pode receber foco, uma vez que isto pode resultar em um estado de foco no "nada". Exemplo de mau uso:

```
1 |     <button aria-hidden="true">press me</button>
```

Regra 5: Todos os elementos interativos devem ter um nome acessível. As API de acessibilidade de cada plataforma fornecem a propriedade de nome acessível ou equivalentes, de modo que esse é concatenado com a função do elemento de interface de usuário e apresentado pelo recurso de TA. Um exemplo é a identificação de um determinado campo de formulário por meio do elemento *label*, conforme a seguir:

```
1    <label for="nomeusu">Usuário</label>
2    <input type="text" id="nomeusu">
```

2.5. Conclusões

Este minicurso abordou um tema de grande relevância técnica e social para profissionais e pesquisadores envolvidos com o desenvolvimento de aplicações Web em geral, e em especial, de RIAs. Desenvolver sistemas Web acessíveis e usáveis é importante para atingir uma parcela significativa de usuários com deficiência. Para tanto, foram apresentados os principais conceitos que têm sido geralmente utilizados para avaliação de acessibilidade e usabilidade em RIA - uma visão geral das perspectivas, desde Normas / Padrões de Qualidade até os recursos para implementação de aplicações Web.

A experiência dos autores no campo do ensino mostrou-lhes que o desenvolvimento de interfaces RIA usáveis e acessíveis é desafio também no campo do ensino. A recente existência dos conteúdos RIA ainda gera incertezas quanto à aplicabilidade de métodos tradicionais de avaliação de interface para avaliação das mesmas. A grande possibilidade de adaptação de tecnologias RIA para preferências individuais de usuários, como mostrado ao longo deste minicurso, reforça a sensibilidade de tais tecnologias à qualidade da organização de métodos de avaliação pouco automatizáveis.

Assim, neste minicurso, os autores organizaram roteiros práticos para dois dos métodos mais tradicionais entre os de avaliação de usabilidade e acessibilidade a fim de assegurar a boa organização dos mesmos por praticantes, também objetivando maximizar o aproveitamento e performance de tais métodos. Tais roteiros foram desenvolvidos com base em uma compilação de recomendações identificadas na literatura. Assim, este minicurso foi oportunidade de difundir a importância de avanços no desenvolvimento e validação de roteiros similares para a literatura da área.

2.6. Agradecimentos

Agradecemos ao suporte dado pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) e a Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), processo n. 2015/09493-5.

Referências

[Almeida and Baranauskas 2012] Almeida, L. D. A. and Baranauskas, M. C. C. (2012). Accessibility in rich internet applications: People and research. In *Proceedings of the 11th Brazilian Symposium on Human Factors in Computing Systems*, IHC '12, pages 3–12, Porto Alegre, Brazil, Brazil. Brazilian Computer Society.

[Barbosa and da Silva 2010] Barbosa, S. D. J. and da Silva, B. S. (2010). *Interação humano-computador*. Elsevier.

[Berners-Lee 1990] Berners-Lee, T. (1990). Information management: A proposal. W3C archive.

[Bevan et al. 2016] Bevan, N., Carter, J., Earthy, J., Geis, T., and Harker, S. (2016). New ISO Standards for Usability, Usability Reports and Usability Measures. In Kurosu, M., editor, *Human-Computer Interaction. Theory, Design, Development and Practice*, number 9731 in Lecture Notes in Computer Science, pages 268–278. Springer International Publishing. DOI: 10.1007/978-3-319-39510-4_25.

[Bozzon et al. 2006] Bozzon, A., Comai, S., Fraternali, P., and Carughi, G. T. (2006). Capturing ria concepts in a web modeling language. In *Proceedings of the 15th International Conference on World Wide Web*, WWW '06, pages 907–908, New York, NY, USA. ACM.

[Carvalho et al. 2016] Carvalho, L. P., Ferreira, L. P., and Freire, A. P. (2016). Accessibility evaluation of rich internet applications interface components for mobile screen readers. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, SAC '16, pages 181–186, New York, NY, USA. ACM.

[Casteleyn et al. 2014] Casteleyn, S., Garrig'os, I., and Maz'on, J.-N. (2014). Ten years of rich internet applications: A systematic mapping study, and beyond. *ACM Trans. Web*, 8(3):18:1–18:46.

[Dix et al. 2003] Dix, A., Finlay, J., Abowd, G. D., and Beale, R. (2003). *Human Computer Interaction*. Pearson Education Limited, 3rd edition.

[Doush et al. 2013] Doush, I. A., Alkhateeb, F., Maghayreh, E. A., and Al-Betar, M. A. (2013). The design of {RIA} accessibility evaluation tool. *Advances in Engineering Software*, 57:1 – 7.

[Ericsson and Simon 1980] Ericsson, K. A. and Simon, H. A. (1980). Verbal reports as data. *Psychological Review*, 87(3):215–251.

[Fernandes 2013] Fernandes, N. (2013). Towards Web Accessibility Repair. In *Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility*, W4A '13, pages 9:1–9:2, New York, NY, USA. ACM.

[Fernandes et al. 2013] Fernandes, N., Batista, A. S., Costa, D., Duarte, C., and Carriço, L. (2013). Three Web Accessibility Evaluation Perspectives for RIA. In *Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility*, W4A '13, pages 12:1–12:9, New York, NY, USA. ACM.

[Fernandes et al. 2012] Fernandes, N., Costa, D., Neves, S., Duarte, C., and Carriço, L. (2012). Evaluating the Accessibility of Rich Internet Applications. In *Proceedings of the International Cross-Disciplinary Conference on Web Accessibility*, W4A '12, pages 13:1–13:4, New York, NY, USA. ACM.

[Følstad et al. 2012] Følstad, A., Law, E., and Hornbæk, K. (2012). Analysis in practical usability evaluation: A survey study. In *Proceedings of the SIGCHI Conference on*

Human Factors in Computing Systems, CHI '12, pages 2127–2136, New York, NY, USA. ACM.

[Freire et al. 2013] Freire, A. P., de Lara, S. M. A., and de Mattos Fortes, R. P. (2013). Avaliação Da Acessibilidade De Websites Por Usuários Com Deficiência. In *Proceedings of the 12th Brazilian Symposium on Human Factors in Computing Systems*, IHC '13, pages 348–351, Porto Alegre, Brazil, Brazil. Brazilian Computer Society.

[Gehtland et al. 2006] Gehtland, J., Galbraith, B., and Dion, A. (2006). *Pragmatic Ajax: A Web 2.0 Primer*. Pragmatic Bookshelf, USA, 1 ed. edition.

[Hooshmand et al. 2016] Hooshmand, S., Mahmud, A., Bochmann, G. V., Faheem, M., Jourdan, G.-V., Couturier, R., and Onut, I.-V. (2016). D-ForenRIA: Distributed Reconstruction of User-Interactions for Rich Internet Applications. In *Proceedings of the 25th International Conference Companion on World Wide Web*, WWW '16 Companion, pages 211–214, Republic and Canton of Geneva, Switzerland. ACM Press.

[IBGE 2010] IBGE (2010). Censo demográfico 2010. Technical report, Rio de Janeiro, RJ.

[ISO/IEC 25066 2016] ISO/IEC 25066 (2016). ISO/IEC 25066:2016(en) Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Common Industry Format (CIF) for Usability — Evaluation Report.

[Law et al. 2009] Law, E. L.-C., Roto, V., Hassenzahl, M., Vermeeren, A. P., and Kort, J. (2009). Understanding, Scoping and Defining User Experience: A Survey Approach. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 719–728, New York, NY, USA. ACM.

[Lawton 2008] Lawton, G. (2008). New ways to build rich internet applications. *Computer*, 41(8):10–12.

[Machado Neto et al. 2014] Machado Neto, O. J., Cunha, B. C., Pimentel, M. G., and Fortes, R. (2014). Capítulo - Tecnologia Assistiva: pesquisas, oportunidades e exemplos de recursos de tecnologia assistiva multimidia interativos. In *Livro de minicursos do XX Simpósio Brasileiro de Sistemas Multimídia em Web, João Pessoa, PB*, pages 1–18, João Pessoa, PB. SBC.

[Martins et al. 2014] Martins, A. I., Queirós, A., Silva, A. G., and Rocha, N. P. (2014). Usability Evaluation Methods: A Systematic Review. *Human Factors in Software Development and Design*, page 250.

[Mesbah et al. 2012] Mesbah, A., van Deursen, A., and Roest, D. (2012). Invariant-based automatic testing of modern web applications. *IEEE Transactions on Software Engineering*, 38(1):35–53.

[Mori et al. 2011] Mori, G., Buzzi, M. C., Buzzi, M., Leporini, B., and Penichet, V. M. R. (2011). *Collaborative Editing for All: The Google Docs Example*, pages 165–174. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Nielsen 1994a] Nielsen, J. (1994a). Enhancing the Explanatory Power of Usability Heuristics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '94, pages 152–158, New York, NY, USA. ACM.

[Nielsen 1994b] Nielsen, J. (1994b). Heuristic evaluation. In NIELSEN, J.; MACK, R. L., editor, *Usability inspection methods*, pages 25 –62.

[Nielsen 1995] Nielsen, J. (1995). How to conduct a heuristic evaluation. Nielsen Norman Group: Evidence-Based User Experience Research, Training, and Consulting. Url: `https://www.nngroup.com/articles/how-to-conduct-a-heuristic-evaluation/`. Acessado em 2 de setembro de 2016.

[Nielsen 2001] Nielsen, J. (2001). Usability metrics. Nielsen Norman Group: Evidence-Based User Experience Research, Training, and Consulting. Url: `https://www.nngroup.com/articles/usability-metrics/`. Acessado em 2 de setembro de 2016.

[Nielsen and Molich 1990] Nielsen, J. and Molich, R. (1990). Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 249–256. ACM.

[Norman 1988] Norman, D. A. (1988). *The psychology of everyday things*. Basic books.

[Norman 2010] Norman, D. A. (2010). *Living with complexity*. MIT press.

[Norman 2013] Norman, D. A. (2013). *The design of everyday things: Revised and expanded edition*. Basic books.

[Paz and Pow-Sang 2016] Paz, F. and Pow-Sang, J. A. (2016). A systematic mapping review of usability evaluation methods for software development process. *International Journal of Software Engineering and Its Applications*, 10(1):165–178.

[Petrie and Power 2012] Petrie, H. and Power, C. (2012). What do users really care about?: a comparison of usability problems found by users and experts on highly interactive websites. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2107–2116. ACM.

[Preece et al. 2015] Preece, J., Sharp, H., and Rogers, Y. (2015). *Interaction design: beyond human-computer interaction*. John Wiley & Sons.

[Steen-Hansen and Fagernes 2015] Steen-Hansen, L. and Fagernes, S. (2015). Achieving accessible rich internet applications. *Norsk Informatikkonferanse (NIK)*.

[Thatcher et al. 2002] Thatcher, J., Bohman, P., Burks, M., Henry, S. L., Regan, B., Swierenga, S., and Urban, M. (2002). *Constructing Accessible Web Sites*. Glasshaus.

[Usability.gov 2016a] Usability.gov (2016a). Planning a usability test. Usability.gov: Improving the User Experience (U.S. Department of Health & Human Services). Url: `https://www.usability.gov/how-to-and-tools/`

methods/planning-usability-testing.html. Acessado em 2 de setembro de 2016.

[Usability.gov 2016b] Usability.gov (2016b). Running a usability test. Usability.gov: Improving the User Experience (U.S. Department of Health & Human Services). Url: https://www.usability.gov/how-to-and-tools/methods/running-usability-tests.html. Acessado em 2 de setembro de 2016.

[van Wamelen and de Kool 2008] van Wamelen, J. and de Kool, D. (2008). Web 2.0: a basis for the second society? In *Proceedings of the 2nd international conference on Theory and practice of electronic governance*, volume 1 of *ICEGOV '08*, pages 349–354, New York, NY, USA. ACM.

[W3C 2005] W3C (2005). Introduction to Web Accessibility. W3C. Url: https://www.w3.org/WAI/intro/accessibility.php. Acessado em 05 jun. 2016. World Wide Web Consortium.

[W3C 2014] W3C (2014). Accessible rich internet applications (wai-aria) 1.0. W3C. Url: https://www.w3.org/TR/wai-aria/. Acessado em 05 jun. 2016. World Wide Web Consortium.

[W3C 2015] W3C (2015). Notes on using aria in html. W3C. Url: https://www.w3.org/TR/aria-in-html/. Acessado em 05 jun. 2016. World Wide Web Consortium.

[Watanabe 2014] Watanabe, W. M. (2014). *Avaliação automática de acessibilidade em RIA*. Tese de Doutorado em Ciências de Computação e Matemática Computacional, Intituto de Ciências Matemáticas e de Computação de São Carlos (ICMC/USP), Universidade de São Paulo, São Carlos - SP.

[Watanabe et al. 2012] Watanabe, W. M., Fortes, R. P. M., and Dias, A. L. (2012). Using Acceptance Tests to Validate Accessibility Requirements in RIA. In *Proceedings of the International Cross-Disciplinary Conference on Web Accessibility*, W4A '12, pages 15:1–15:10, New York, NY, USA. ACM.

Apêndice A: As 10 Heurísticas de Nielsen

Como indicado na Tabela 2.1, a definição do conjunto de heurísticas considerado para a avaliação é de grande importância. O conjunto tradicional das dez heurísticas de Nielsen está disponível online no portal do grupo de consultoria Nielsen & Norman Group. A seguir, apresentamos uma tradução feita a partir das descrições mostradas por Nielsen:

1. **Visibilidade do estado do sistema -** O sistema deveria sempre manter os usuários informados sobre o que está acontecendo, usando de *feedback* em tempo apropriado.
2. **Compatibilidade entre o sistema e o mundo real -** O sistema deveria falar a linguagem do usuário, com palavras, frases e conceitos familiares ao usuário, em vez de termos próprios do sistema. Seguir convenções do mundo real, fazendo a informação aparecer em uma ordem natural e lógica.
3. **Controle e liberdade do usuário -** Os usuários frequentemente escolhem funções por erro e irão precisar de uma saída de emergência bem sinalizada para sair do estado indesejado sem ter que passar por um diálogo extenso. Suportar undo e redo.
4. **Consistência e padrões -** Os usuários não deveriam ter que imaginar se palavras, situações, ou ações diferentes significam a mesma coisa. Seguir plataforma de convenções.
5. **Prevenção de erro -** Ainda melhor do que boas mensagens de erro é um design cuidadoso que previne o acontecimento de um problema em primeiro lugar. Tanto eliminar condições de propensão ao erro quanto checar por tais condições e apresentar uma opção de confirmação ao usuário antes que o mesmo cometa à ação.
6. **Reconhecimento em vez de lembrança -** Minimizar a carga de memória do usuário fazendo objetos, ações, e opções visíveis. O usuário não deveria ter que se lembrar da informação de uma parte do diálogo para outra. Instruções para o uso do sistema deveriam estar visíveis ou facilmente resgatáveis sempre que apropriado.
7. **Flexibilidade e eficiência de uso -** Aceleradores – não vistos por usuários novatos – devem frequentemente acelerar a interação para o usuário experiente de forma que o sistema possa abranger ambos os usuários inexperientes e experientes. Permitir que os usuários adaptassem ações frequentes.
8. **Estética e *design* minimalista -** Diálogos não deveriam conter informação que é irrelevante ou raramente necessária. Toda unidade extra de informação em um diálogo compete com as unidades relevantes de informação e diminui sua visibilidade relativa.
9. **Ajudar os usuários a reconhecer, diagnosticar, e recuperar de erros -** Mensagens de erro deveriam ser elaboradas em linguagem plena (não em códigos), precisamente indicar o problema, e fornecer a solução de maneira construtiva.

10. Ajuda e documentação - Apesar de que o sistema possa ser usado sem documentação, pode ser necessário prover ajuda e documentação. Qualquer informação do tipo deveria ser fácil de buscar, ser focada nas tarefas dos usuários, listar passos concretos para serem executados, e não ser muito extensa.[17]

[17]Retirado e traduzido do Portal Nielsen & Norman Group, acessado em 03 de outubro de 2016 pelo endereço: `https://www.nngroup.com/articles/ten-usability-heuristics/`

Biografia Resumida dos Autores

Renata Pontin M. Fortes é professora do Departamento de Ciências de Computação do ICMC (Instituto de Ciências Matemáticas e de Computação) na Universidade de São Paulo, campus de São Carlos. Possui doutorado na área de Engenharia Web pelo Instituto de Física de São Carlos, da USP, e mestrado e graduação pelo ICMC/USP em São Carlos. Atualmente é professora associada II do ICMC/USP. Tem experiência na área de Ciência da Computação, atuando principalmente nos seguintes temas: engenharia de web, projetos de software livre, acessibilidade na web, interação homem-computador e processos de software. Tem diversas publicações em periódicos, conferências e capítulos de livros na área de Acessibilidade e Engenharia Web.

Humberto Lidio Antonelli é doutorando no programa de Pós-Graduação em Ciências da Computação e Matemática Computacional no ICMC/USP. Possui mestrado em Ciências da Computação pelo ICMC/USP e graduação em Ciência da Computação pela Universidade Federal de Goiás. Pesquisador na área de acessibilidade e usabilidade de interfaces Web para diferentes contextos. Tem experiência na área de Interação Humano-Computador e Linguagens de Programação Web. Atualmente, realiza pesquisas com avaliação de acessibilidade em *Rich Internet Applications* (RIA). Atua principalmente nos seguintes temas: acessibilidade, gerenciamento de decisões, Interface Humano-Computador e Engenharia Web.

André de Lima Salgado é mestrando em Ciências da Computação e Matemática Computacional no ICMC/USP. Possui Bacharelado em Sistemas de Informação pela Universidade Federal de Lavras (UFLA). Realiza pesquisas na área de Interação Humano-Computador (IHC), com foco em inspeção de Usabilidade com Avaliação Heurística conduzidas por avaliadores com pouca experiência na área, também chamados novatos. Adicionalmente, o autor também realiza pesquisas relacionadas a Experiência de Usuário, Acessibilidade, Arquitetura da Informação, Brinquedos Inteligentes e *Design* Centrado no Usuário. O autor é membro dos grupos de pesquisa ALCANCE (UFLA) e INTERMÍDIA (ICMC/USP).

Chapter 3

Crowdsourcing & Multimedia: Enhancing Multimedia Activities with the Power of Crowds

Ricardo M. C. Segundo[1], Marcello N. de Amorim[1] e Celso A. S. Santos[1]
[1]Universidade Federal do Espírito Santo (UFES)

Abstract

This short course aims to present the concept of crowdsourcing and empower participants to implement this model of production in various activities linked with Multimedia and Web Systems, such as annotation, generation, summarization, synchronization, recommendation, retrieval, presentation and evaluation of the content quality. The idea behind crowdsourcing is to take advantage of the processing power of a multitude of employees to accomplish tasks that are "difficult for a computer", but which are apparently "simple to human intelligence". Describing the contents of an image or a video as inappropriate is an example of such difficult tasks, because the description of the problem by means of algorithms and automated techniques applied to parameters of this content is very complex and inaccurate. Another complex task is the subjective assessment of the quality of video encoders, the results depend on the user's perception and not only on parameters such as signal-to-noise ratio, resolution or frame rate. The crowdsourcing model tends to provide reliable results for these kinds of problems related to Multimedia and Web Systems. The additional content support for this course brings the fundamental concepts of crowdsourcing, a discussion of suitable scenarios for their use within the multimedia and examples of practical use of the concept in real-world examples.

3.1. Introduction

The use of crowdsourcing facilitates large or complex tasks, which would be: (i) non-viable to be resolved automatically by a computer, because of their characteristics; and (ii) non-viable to be resolved by one or few people due to the effort required to perform them. A typical example in the area of multimedia is the annotation of large multimedia databases (containing images, videos, audios) to identify which emotions each of these content tends to provoke. The task in question can be performed by any human collaborator, while any automatic system would have enormous difficulties in executing it.

Additionally, the huge amount of images makes it very costly to be held by a single person, or even a small group. In order to achieve the expected result the processing power of the crowd can be applied. This short course aims to empower participants to shape and build applications based on crowdsourcing, considering important issues such as the identification of activities that require human intelligence, the definition of the microtasks, as well as some points concerning implementation and integration of applications, in addition to essential issues, such as the characteristics of the crowd, employee retention and verification of the reliability of contributions.

Videos, for example, have a high-cost processing that encourages optimization of techniques and the use of alternative ones. Similar challenges for other media format (images) had success with crowdsourcing approaches. One such approach is the ESP Game, by [Von Ahn *et al.* 2008], that helps to determine the contents of images, providing meaningful labels based on contributions gathered from the crowd. Also, the investigation of crowdsourcing by the Multimedia community (http://www.crowdmm.org/) encourages a deeper investigation of the use of crowdsourcing with multimedia. Crowdsourcing is an approach to gather different kinds of collaborations such as services, information, knowledge and different kinds of data content from large communities of volunteers.

This material main purpose is to introduce readers to what is crowdsourcing, what kind of multimedia research can be done using it and how rapidly develop an application. To this purpose, the rest of this text presents: in Section 3.2 some concepts about crowdsourcing systems; Section 3.3 presents some challenges to multimedia research; Section 3.3 lists some researches involving crowdsourcing in diverse areas of multimedia, from text to videos; Section 3.4 guides readers through the development of a crowdsourcing image annotation application; and Section 3.5 concludes the text.

3.2. The Crowd

Crowdsourcing is a term coined by [Howe 2006] to describe systems that are characterized by the contribution of people to forge from small tasks a bigger solution. But before we talk about crowdsourcing, lets discuss two important concepts: Crowd Wisdom and Human Computation.

3.2.1. Crowd Wisdom

Collective intelligence is all intelligence that arises from the different knowledge shared on the same subject, assuming that no one has full knowledge, however we all know and have knowledge of something. [Levy 1994] says that collective intelligence is a distributed intelligence that is everywhere, incessantly valued, coordinated in real time, resulting in effective mobilization of competences, which seeks recognition and enrichment of people.

An example used by [Levy 1994] demonstrates a Crowd Wisdom situation: a student wants to know the weight of his backpack. Before measuring it in a scale, he asks his friends how much they think it weights. In the end, the mean of the opinions is almost the same as the real one measured in the balance. This story tries to show that with each person experiences together and probed, he can solve problems, as how to find out the weight of the backpack.

Nowadays the Internet has been used as a way to make more agile this kind of collective intelligence and the concept has gotten new forms. There are three ways to generate collective intelligence: conscious, unconscious and full intelligence.

On unconscious the user is working without even knowing it, simply by the act of navigating a web page or using an application that registers information. Examples of unconscious collective intelligence are: filling forms, click on pictures and links. ”Each click with the mouse or the keyboard is a decision to be registered and used by a particular system that organizes and allows others to take advantage of the trail left by those who came before. In this way, a user can know what is the best-selling book in a bookstore, or in the case of a blog, which published articles was more read or reviewed, creating a relevant criterion, which speeds up the visitor's decision”[Cavalcanti & Nepomuceno 2007].

In conscious intelligence, users involved must strive to achieve a result and those involved can develop something for a cause. Examples of this are the intelligence development of free software and user support provided through forums where people are willing to solve given problems.

The full intelligence is one that involves both the unconscious and conscious intelligence in a single environment.

3.2.2. Human Computation

Human computation [Law & Ahn 2011] is a term introduced by Luis von Ahn in his thesis and it refers to identify which part of a problem can be automated and which part require human intelligence to solve the problem. In this approach humans take the processing of a HIT (Human Intelligent Task), that usually presents better performance and efficiency, in comparison with a computer, because these tasks are associated with activities hard for computers but easy for humans. [Law & Ahn 2011] defines Human Computation as:

> "... human computation is simply computation that is carried out by humans. Likewise, human computation systems can be defined as intelligent systems that organize humans to carry out the process of computation whether it be performing the basic operations (or units of computation), taking charge of the control process itself (e.g., decide what operations to execute next or when to halt the program), or even synthesizing the program itself (e.g., by creating new operations and specifying how they are ordered). The meaning of ”basic” varies, depending on the particular context and application.For example, the basic unit of computation in the calculation of a mathematical expression can be simple operations (such as additions, subtractions, multiplications and divisions) or composite operations that consist of several simple operations. On the other hand, for a crowd-driven image labeling system, a user who generated a tag that describes the given image can also be considered to have performed a ”basic” unit of computation." [Law & Ahn 2011]

Human Computation is extremely suitable to Crowdsourcing once it provides the base to identify, in the problem to be solved, the HITs that can be modelled as micro-tasks. Micro-tasks are simple and small, and demands an ordinary human to solve it.

Not all Crowdsourcing projects are based in micro-tasks, although it is the most common and usually associated to success cases. Fragmenting the problem into micro-tasks allows spreading the processing among a crowd of contributors, maximising the use of the processing power of this crowd.

Other characteristic that make micro-tasks suitable to Crowdsourcing is the independence between different micro-tasks. The independence of the job execution is very important in a scenario where you have potentially an enormous amount of users. Actually in these terms, Crowdsourcing is a particular case of Collaborative Approach where it is not desirable that users interact to others in order to avoid biasing their judgement, and a job can't depend on other results. Unlike the conventional Collaborative Approaches based in the 3C Model (Collaboration, Communication and Coordination), Crowdsourcing asks the collaborators for independent results that are merged in a future phase.

3.2.3. Crowdsourcing

Crowdsourcing is a term that was first described in 2006 by Jeff Howe and Mark Robinson, at the time, both were editors of a magazine called Wired Magazine, that defined the term in the article: The Rise Of Crowdsourcing [Howe 2006].

Crowdsourcing uses the knowledge of a group of people who will be collaborators of information regardless of their creed, or geographic location. Crowdsourcing can be said as a type of outsourcing, where collective knowledge and volunteers are used to create content, develop new technologies, solve everyday problems and disposal services.

A more elaborated definition is found in [Estellés-Arolas & González-Ladrón-de Guevara 2012]:

> "Crowdsourcing is a type of participative online activity in which an individual, an institution, a non-profit organization, or company proposes to a group of individuals of varying knowledge, heterogeneity, and number, via a flexible open call, the voluntary undertaking of a task. The undertaking of the task, of variable complexity and modularity, and in which the crowd should participate bringing their work, money, knowledge and/or experience, always entails mutual benefit. The user will receive the satisfaction of a given type of need, be it economic, social recognition, self-esteem, or the development of individual skills, while the crowd-sourcer will obtain and utilize to their advantage what the user has brought to the venture, whose form will depend on the type of activity undertaken."

The commitment to the task, complexity and modularity, and the crowd participation by bringing their work, money, knowledge and/or experience, always implies mutual benefit to workers and crowdsourcers. The user will receive the satisfaction of a particular type of befit, whether economic, social recognition, self-esteem, or the development of individual skills, while the crowdsourcer will obtain and use the advantage over what the user has brought to the venture, which will depend on the type of activity developed.

Crowdsourcing makes use of collective intelligence, the experiences and knowledge that are acquired through the crowd connected via the internet for the resolution of

problems, which may be responsible for the creation of new content and even the development of new technologies.

The Crowdsourcing can be considered a method of solving problems that in general has four fundamental challenges:

- How to recruit and retains users?
- What contribution can these users provide?
- How to join and use the contributions made by the crowd to solve the target problem?
- How to evaluate the user contributions and the users themselves?

Here we won't go into details on how to solve these questions, this may vary widely depending on how you will use the crowd within your problem. [Pedersen *et al.* 2013] defines a conceptual model that describes most of these dependences. The model is composed of: Problem, People (Problem Owner, Individual, and Crowd), Governance, Process, Technology and Outcome. Crowdsourcing systems can be classified according to its dimensions. These dimensions [Doan *et al.* 2011] are:

1. Nature of collaboration: can be implicit or explicit and specifies if the users know that their actions are generating contributions that are being used by the system, even if the main purpose of the work is unknown to the user.
2. Type of target problem: refers to the main problem approached in the paper, not the crowdsourcing aspect.
3. How to recruit and retain users: the crowd of users may be gathered by an open call recruitment, through a crowdsourcing platform, or by a direct selection made by the projects conductors. Also, collaborations may be extract from other systems.
4. What can users do: describes the classes of actions that can be performed by the users, such as annotation, evaluation and content generation.
5. How to combine their inputs: describes the methods used in order to merge individual contributions in a final output.
6. How to evaluate them: details which techniques are applied in order to verify reliability and quality of the contributions, such as gold patterns and control questions. Additionally, define if malicious users should be blocked or bad contributions lead to punishment.
7. Degree of manual effort: describes if the individual tasks are trivial or require a significant cognitive or manual effort.
8. Role of human users: establishes if users may collaborate providing self-generated content, executing individual tasks in a divide-and-conquer approach, acting as a system component over the target artifact, or as a perspective provider producing different perspectives of the problem that often can be combined in a better solution.

9. Standalone versus piggyback architectures: Some crowdsourcing applications count on standalone specific systems headed to collect collaborations. When collaborations are extracted from existing systems that weren't built specifically to collect the contributions the application is using a piggybacking approach.

3.3. Multimedia Powered By the Crowd

Crowdsourcing can be used in multiple activities linked with Multimedia and Web Systems, such as annotation, generation, summarization, synchronization, recommendation, retrieval, presentation and evaluation of the contents quality. In this section we describe examples of this use in some multimedia activities involving text, audio, image and video objects.

3.3.1. CrowdText

Analysing text to detect emotions, digitalizing, summarizing, and have characteristics extracted are examples of text usage in multimedia. We present next some examples that may show readers how extensive the possibilities could be.

In medical context we can use the crowd to extract medical terms in patient-authored texts [MacLean & Heer 2013]. This work is based on the increasingly number of health-oriented applications, where users describe their conditions and receive feedback. They extract from forms medical terms that can be used to train classifiers. Other example in medical scope is the use of gamification techniques and crowdsourcing for training Natural Language Processing Tools on medical context [Dumitrache *et al.* 2013].

Analysing texts to extract characteristics is other activity that the crowd can do. In [Borromeo & Toyama 2015], the crowd is used to analyse sentiments in the text. It compared four different techniques: two crowdsouerced ones (paid and volunteered), a manual and a automatic. Text simplification is other aspect that can be analysed by the crowd. [Lasecki *et al.* 2015] use the crowd to find the best simplified text, which is reduced complexity text that retains the same meaning of the original.

Maybe, one of the most common use of corwdsourcing regarding text is its digitalization. [Von Ahn *et al.* 2008] presented the reCAPTCHA system, where he implicitly ask humans to perform the digitalization of texts. This is implicitly, because the humans think they are simply entering a CAPTCHA to have access to a link, but in fact, they are transcribing a piece of text that a computer could not digitalize.

Translation is another common activity. [Schlippe *et al.* 2013] used the crowd to normalize texts to be used in their statistical machine translation method. [Corney *et al.* 2010] uses humans to directly translate non-Roman texts to be catalogued in libraries. As a final example, we have Duolingo ([von Ahn 2013]), an application that helps people learn other languages, while also helping to translate texts from one language to other.

3.3.2. CrowdImage

In this subsection we present some examples of using the power of the crowd for processing images, such as quality evaluations and annotations.

We start with some examples of quality assessment. [Ghadiyaram & Bovik 2016]

created both a image dataset ("LIVE In the Wild Image Quality Challenge Database") that represents the real world scenario (not the ones generated by algorithms), and a crowdsourcing platform to perform a quality assessment on their dataset, with more than 1162 images, resulting in more than 350,000 scores on this dataset. In their experiment, users had to say what kind of distortions the images presented. [Ribeiro *et al.* 2011] also presented a quality evaluation on images. They use a crowdMOS subjective measure, where users score from 1 (Bad) to 5 (Excellent) depending on the quality of the images. [Saad *et al.* 2016] focus are not in presenting a form for assessing images using the crowd, their focus is presenting challenges and pitfall in using the crowd for that. Examples of these points are tying the crowd responses to the stimulus response, using full-screen mode with real images, the length of the study and others.

Medical problems can also be solved with aid of the crowd. [Foncubierta Rodríguez & Müller 2012] used the crowd to create a ground truth to evaluate computer-based tools. They in first instance use the crowd to annotate medical images within multiple medical categories. The initial results are used as input for an automatic tool that annotates the rest of the dataset. The crowd is then used in a second instance to validate the results. Not directly related to medical tools, but also in the context of helping people, we find the UCap application ([Hoonlor *et al.* 2015]), that aims in helping blind people in their daily activities. The task to the crowd is to identify consumer products and inform the nutrition values to blind people when shopping.

Clustering images is other common computational problem that requires high performance, being a good candidate to be solved by the crowd. [Guo *et al.* 2015] used a hybrid solution to the clustering problem. Its proposal is to use the crowd to validate the output from a similarity algorithm, in this way the time and cost of the entire process is optimized.

3.3.3. CrowdAudio

Audio is the first media that presents a characteristic that not images nor text have: audio is a continuous media object. With the continuous characteristic, we need more time from the users to receive their contribution, and probably some more effort. In the sequence, we present some examples of using the crowd with audio.

First we have an example of using the crowd to generate media. In [Rizvi *et al.* 2015], the owner ask for content within an explicit criteria, such as noise pollution, where the crowd shall record noise pollution from a city. From the contributions, the owner can measure noise level in a city. This activity is done by the crowd, that receives the request on their devices, such as smartphones and tablets, and uses this device to collect and send the requested content.

As in image and text contributions, annotation is broadly used for audio context. The AMG1608 dataset ([Chen *et al.* 2015b]) used crowd to annotate 1608 30-second music clips using 665 subjects. Their goal is to provide a base to test and train automated recognition approaches. Similar to the AMG1608 dataset, the YourSpeech dataset ([Freitas *et al.* 2010]) aims to provide training for Automatic Speech Recognition, but they use the crowd to both collect the audio and transcript it.

Transcription methods using crowdsourcing are broadly approached by researches. [Lee & Glass 2011] use the crowd to transcribe short audio clips of about five to six seconds. These audios had originally a duration of an hour or more, but an audio of this size would require too much effort of one crowd member, therefore they automatically partitioned the audio and made each crowd task with 5 audio clips. [Saito 2015] also uses small chunks of audio to be transcribed by the crowd. But their goal was not just transcribe the audio files, their goal was create caption data to hearing-impaired people.

3.3.4. CrowdVideo

Video combines characteristics of the objects described previously: frames are composed images, the video has the continuous characteristic also present in audio, and subtitles, for example, are composed of text units. That is why we present a deeper analysis of crowdsourced video works, describing not only key papers, but also describing some characteristics of the crowd.

3.3.4.1. Crowd Characteristics in Video Domain

We analysed multiple papers that use Crowdsourcing with videos, for multiple purposes, from annotation to video evaluation. Now we describe some works in relation to their crowdsourcing dimensions.

Nature of Collaboration The nature of collaboration reveals if the crowd actively contributed with the problem or if their contributions were implicitly collected. In explicit systems users can evaluate, share, network, build artifacts, and execute tasks [Doan *et al.* 2011].

A systems can let users build artifacts: [Wilk *et al.* 2015] created a composed live video presentation of user generated video. The workers selected the most suitable view of a recorded scene to switch, in other words, they chose the best video with the best audio.

Implicitly standalone systems provides a service such that when using it users implicitly collaborate. Games and video players are used to annotate and evaluate videos. Some presented papers that uses games as an approach to extract information from videos. In [Pinto & Viana 2013] the goal is to annotate the videos in form of games, where the crowd annotate a video (or a frame from it) and gains some punctuation from it.

Using video players, the systems can identify regions of interest in a video by analyzing where users zoomed a video [Carlier *et al.* 2011]; video highlights using the user navigation times inside a video; reconstruct 3D models using videos published by users; improve QoE using prefetching for video streaming [Pang *et al.* 2011]; and [Zegarra Rodriguez *et al.* 2015] used the crowd attention to a video to evaluate the quality of transmission.

Some systems presented both explicit and implicit collaborations. In [Liao & Hsu 2013] users that trace a route using GPS to their destination can be summoned to take photos of specific places that require a little detour from the original route. The

contributions are explicit, but rely on other applications to select the members of the crowd.

Recruitment of users Recruitment of user (worker) is an essential part of crowdsourcing: no users means no collaborations. Most studies pays their users using a Crowdsourcing recruitment platform (Amazon Mechanical Turk, Microworkers, CrowdFlow). In these systems the payment done to the crowd varies depending on the task, and sometimes depending on the user profile. In [Baveye *et al.* 2015] workers were paid US$0.05 for answering five comparisons from a pair of video excerpts, selecting the one that conveys "the most positive emotion" or "the calmest emotion". [Vondrick *et al.* 2013] paid $0.05 per object annotated, where a complete video clip annotation could cost between $1 and $2. In [Park *et al.* 2014] a total of 19 workers participated in experiments, who worked for an effective hourly wage between $4 and $6 as compensation. In general, however, the median value of a task used by most was between: 0.2 and 0.3 dollars.

Regarding recruitment of users we found papers that use from 9 to more than half a million contributions. The median of members is 288.7, disregarding the paper by [Huberman *et al.* 2009], that had 579471 members. This high value occurred because they used a piggybacking approach on youtube platform, allowing them to achieve such a number in their analyses.

Retaining Plans Crowdsourcing systems should consider how to further encourage and retain users after they are recruited.Gamification is a common technique used and brings to users an enjoyable experience that leads them to return and collaborate again. Among multiple best practices detailed in [Hoßfeld *et al.* 2014], they support the use of gaming factors in crowdsourcing systems. They claim that 80% of the users returns to a game, compared to only 23% for a regular task and unreliable ratings in their task annotation game are reduced to 2.3% instead of 13.5%, compared to a non-gaming task. Enjoyable experience with gamification has an intersection with another retaining strategy: competitions, such as top rated users.

Tasks The tasks executed by the crowd may have direct influence in collaborations resulted from the crowd. If a crowdsourcer delegates a task that is cognitively hard, he may receive less collaborations from users, while easy tasks stimulates users to contribute more. In [Keimel *et al.* 2012] from 99 users, only 7 accomplished all proposed tasks. In principle they expect users to access all videos from videoset. But probably the members of the crowd thought that the effort was not worth it, doing only part of the experiment.

As highlighted before, gaming is a common feature found in our primary studies. In [Di Salvo *et al.* 2013] users play a game where they mark fishes in the videos to gain score. For the TAG4VD game [Pinto & Viana 2013], players watch a video, and annotate a frame, resulting in points.

Another common task to the crowd is to make them watch videos and evaluate their quality: in [Rainer & Timmerer 2014] users watched and evaluated videos with a 0 - 100 quality scale using a single stimulus approach (one video at a time); in opposition, we

found that in [Figuerola Salas *et al.* 2013] users would watch two simultaneous videos (Double Stimulus Impairment Scale) and evaluated them in a 0-10 scale; instead of scales, users may watch videos and answer question on their perception about purpose, content and emotions [Kim *et al.* 2013]; a set of videos can be used instead to achieve a goal: in [Hossfeld *et al.* 2014] after watching the same video in 3 different resolutions each user filled a QoE test form.

Users may also receive tasks asking them to submit videos. These videos may contain extra information besides the video itself (like gps position, tags and others) to help on further processing [Chen *et al.* 2015a; Ferracani *et al.* 2015] or may be focused in collecting content from a specific event [Venkatagiri *et al.* 2015; Wilk *et al.* 2015] where it will provide a multiple view environment to spectators. On the other hand, the spectators may receive tasks to choose the best cameras and audio inputs [Wilk *et al.* 2015] to compose a video stream, or implicitly [Leftheriotis *et al.* 2012] users watching videos have their action monitored to collect navigation data.

For annotation tasks: [Aran *et al.* 2014] used users to produce crowdsourced annotations on the vlogs: the first type included several aspects of vlogs, such as resolution, amount of motion, and framing. The second one was the personality impressions about the vloggers; in [Vondrick *et al.* 2013; Burmania *et al.* n.d.] users identified objects inside key frames and annotated what they were; [Park *et al.* 2014] also made users watch the video, but they identified previously determined annotations on them just like what happened in [Baveye *et al.* 2015] where annotations were based on users relating video frames with emotions.

Management of users and contributions Crowdsourcing systems often must manage malicious users (users that try to receive the benefits by any means, without a real contribution). To do so, we can use a combination of techniques that block, detect, and detain those cases [Doan *et al.* 2011].

Gold standard (also found in papers as ground truth or control questions) consists in the fact that if a worker answers too many questions incorrectly, it suggests he is an unethical worker [Le *et al.* 2010]. Among the tasks sent to the crowd, questions with known answers are presented to users. Gold standard answers are hidden from the workers and used in postprocessing to estimate the true worker error-rate. With videos, however, we can use their temporal aspect to identify these users. We can verify how long a user watched a video before answering questions about it. We can even monitor if users are focused on the video (listening if user moved to other pages).

[Anegekuh *et al.* 2014] presents in their paper intituled "A Screening Methodology for Crowdsourcing Video QoE Evaluation" most of these techniques. They present an algorithm that takes into account some of the issues that have been identified to impact the quality of crowdsourcing results ((i)Accurate answers to questions; (ii) Determine if an evaluator has unique identification; (iii) Determine if an evaluator actually performs the task (i.e. watch the video before scoring); (iv) Accurate scores to hidden reference video; (v) verify if the evaluator not syndicate member).

3.4. Developing a Crowdsourced Multimedia Application

Until now, only fundamental concepts of crowdsourcing and examples of its use were addressed in the text. In this section we will setting out the steps involved with the development of crowdsourcing applications. There are multiple segments where crowdsourcing could be used, but we focus in only one to guide you through the development and use it as base to develop it in your own context. The context chosen in this guide is the annotation one. We will build an application to classify a human image dataset into: old and young people. The characteristics of the application are:

Case: Classification of a Image Dataset according to the age group of people in its pictures.

Human Computation: Humans can easily estimate the age group of people but this a very hard task for computers.

Job: The crowd classifies a person from an image into an age Group: baby, child, teen, adult or senior. In each job, each member makes 5 contributions (classifies 5 images);

Dataset: The images used for this case will be retrieved from Facebook user profiles.

Application: The job will be presented to users on a webpage. For each job 5 images will be displayed and should be annotated one by one. These images are retrieved from Facebook at the beginning of the job. In order to annotate each image the user must press the button associated to the age group of the person in the picture. The options are Baby, Child, Teenager, Adult, Senior, and Not a Person for the cases where there is not a human in the picture.

Infrastructure: All services used are certificated by SSL. An easy, free, and practical solution is used: we used Dropbox to host HTML, JS ans CSS files, and Google Spreadsheets to persist the contributions.

Thinking about the nine dimensions we could characterize the application as follows:

1. Nature of collaboration: explicit.
2. Type of target problem: annotate images.
3. How to recruit and retain users: voluntary service of "friends".
4. What can users do: choose from six option in which category an images fits.
5. How to combine their inputs: we going to use majority voting to determine an image category.
6. How to evaluate them: if a user misses a golden standard, we disregard his contribution.

7. Degree of manual effort: see and click.

8. Role of human users: executing individual tasks in a divide-and-conquer approach.

9. Standalone versus piggyback architectures: standalone.

Figure 3.1 shows the crowdworkers' interface. You can see the image to be annotated, the options (buttons) and our question to the crowd.

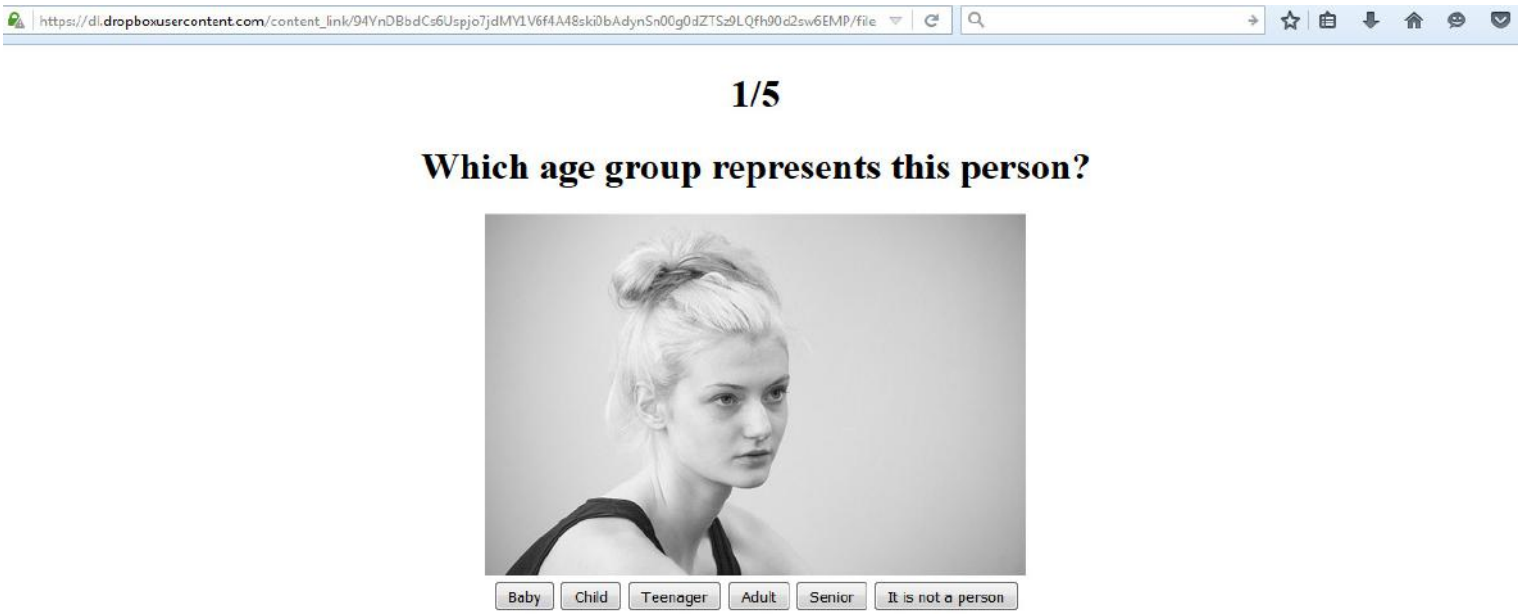


Figure 3.1. Application screen shot

3.4.1. Architecture

For this course, we developed an Architecture (Figure 3.2) that will allow users to implement a simple crowdsourcing APP task using free services and in a reduced time.

Our client (the crowd) needs a modern browser compatible with HTML5 specifications. Through the browser, the user accesses (1) the URL for the Web Application that is hosted in DropBox. The Application is composed only of HTML + JS + CSS specifications and is detailed in section 3.4.2. The DropBox returns (2) the code to the browser that constructs the application and enables the user contribution. After the user evaluates the 5 images, the contribution is sent through HTTPRequest through a get requisition.

The Google Spreadsheet (`https://support.google.com/drive/answer/37603?hl=en`) service allows to include new records as spreadsheets lines (over two million lines) from GET or POST HTTP requests, therefore we can use this service to store contributions from the crowd. In order to proceed it a HTTP request must be composed, including as parameters the columns' IDs and the values to store. We choose the method GET for this request for demonstration reasons because the parameters are explicit in the URL.

This request follows the pattern described latter, including the spreadsheet's ID, columns' IDs and values. To submit the HTTP request using JS you can use a XML-HttpRequest, although, as we are using Dropbox to host HTML and JS files we are in a Cross Domain scenario, and is not possible to proceed with the correct configuration for HTTP Access control (CORS).

The chosen way to skip this restriction was configure the request's mode as NO-CORS, and in this case is more practical to use the `fetch()` method from Fetch API instead of the `XMLHttpRequest.send()` method. The Fetch API is a modern replacement for `XMLHttpRequest` and works well, but as it is a new resource some old browsers do not support it.

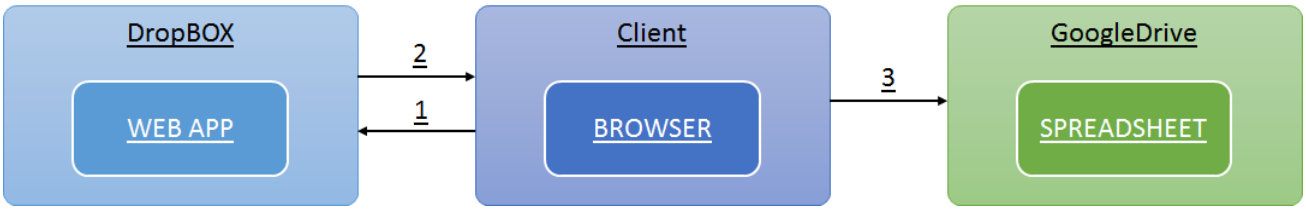


Figure 3.2. APP Architecture

This kind of architecture is identified as a serverless architecture. This refers to applications that significantly depend on third-party services and puts much of the application behavior and logic on the front end. Such architectures remove the need for the traditional server system sitting behind an application.

The main weakness of our crowdsourcing application is related to the users(crowdworkers). We don't have control on recruiting them, nor there is any procedure to give to the user any benefits. This shall be solved aggregating other services such as a gamified one that would rank the users.

3.4.2. Coding

Based on our architecture and requisites, some development prerequisites for the first part of the code are necessary:

- A DropBox Account (`http://dropbox.com/`)
- A GoogleDrive Account (`http://drive.google.com/`)
- A JavaScript/HTML Editor (any)

3.4.2.1. Configuring GoogleDrive

We are going to use Google Spreadsheets as our database. All contributions from users are stored in a spreadsheet for further analysis. In GoogleDrive you will: create a form with the fields you wish to recover from users and create a spreadsheet for this form.

In our tutorial application we want to recover the following informations for each image:

- Duration - we save how long the user took to annotate the image;
- Classification - we save the classification option selected by the user;
- IMG URI - the URI that identifies the annotated image;
- User Id - there is no way to guarantee 100% accuracy for anonymous contributions in our architecture, but we can have clues of which users made more than one contribution using fingerprinting techniques;
- Session Id - an ID, to identify all contributions made in sequence (each user identifies 5 images in sequence);

The first step is to create a new Google form. Title it as you wish, it will not interfere in the application. After, add the fields as short answers, in our case we used: time _ duration, , image_ url ,image_ classification, user_id and session_id. Now go to the responses Tab, and click on Create Spreadsheet. This ends for now our conventional use of Google.

Now we have an informal use, where we need to identify each field and the document to be accessed by our application. To get these information we will use a pre filled link. This is a tricky part, so let's do it step by step:

1. First, in your form, click on the "Get pre-filed link" option;
2. Answer the questions (what you answer does not matter) then click submit;
3. Copy the generated URL;
4. Identify the form Reference and the fields:
   (a) The reference to the form is the value starting with the **https://** until right before the **/viewform**;
   (b) the fields start with the **entry** keyword followed by a dot and a number. The order of the fields are the same of the ones in the form;

As an example, the following lines present the original URL, the reference and the fields.

```
Pre-Filled URL:
        https://docs.google.com/forms/d/e/1FAIpQLSfoexmSLhvScIx4OEcZxXHx6VikDJjN5QApXMw6ZNqgEgaYVg/viewform?
              entry.274168468&entry.1599775086&entry.1175968285&entry.405326048&entry.1284275155&entry
              .595916278

Reference:
        https://docs.google.com/forms/d/e/1FAIpQLSfoexmSLhvScIx4OEcZxXHx6VikDJjN5QApXMw6ZNqgEgaYVg

Fields:
        entry.274168468
        entry.1599775086
        entry.1175968285
        entry.405326048
        entry.1284275155
        entry.595916278
```

3.4.2.2. Configuring DropBox

Dropbox will be used to host our web application. So to achieve this, create a folder to host all files needed. In our case, we called it "mypage". Also create an **index.html** to test and put some content inside. From this file, copy its dropboxlink, and concatenate it with the parameter: **& raw=1**. The new link now can be accessed as a web page. Paste it in the browser, and see your Web Page.

You also need the script link concatenated with **& raw=1**, to use it to import the script.

Figure 3.3 shows an example:

```
Original link:
https://www.dropbox.com/s/2t48p3pu3qbn9pu/index.html?dl=0

Modified link:
https://www.dropbox.com/s/2t48p3pu3qbn9pu/index.html?dl=0&raw=1
```

Figure 3.3. A typical figure

3.4.2.3. Codification

Now let's put our hands on some code. In the folder created in DropBox, you will need to have: an **index.html** and a **script.js**, so if you don't have it yet, create it.

The first step now, is to create the structure of our interface in the HTML file. We are going to need an image tag to define the image to be annotated, buttons for the options and a text area to inform how many images the user has done. The image will be set in the script, and we will get them from a remote service.

The following code contains this structure. We comment each important part of it, so you won't get lost. Note that the link to the script can't be a relative one, using dropbox we need the external link, described in section 3.4.2.2

```
<!-- Application Structure -->
<!DOCTYPE html>
<html>
<head>
        <title>MyFirst Crowd</title>
        <meta charset="utf-8">
        <!-- Turn our app more beautiful and "responsive", using Pure CSS modules -->
        <link rel="stylesheet" href="http://yui.yahooapis.com/pure/0.6.0/pure-min.css">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <!-- Include FingerPrinting Script -->
        <script src="https://cdn.jsdelivr.net/fingerprintjs2/1.4.1/fingerprint2.min.js"></script>
        <!-- Include our script (this must be the dropbox link to the script, concatened to &raw=1) -->
        <script src="https://www.dropbox.com/s/0sifut74vfiz44n/script.js?dl=0&raw=1"></script>

</head>
<body>
        <div id="app" style="text-align: center">
                <!-- Output for the messages -->
                <h1 id="out">1/5</h1>
                <!-- Question to the user -->
                <h1 id="question">Which age group represents this person?</h1>
                <!-- Image area, with a default one -->
                <img height="300"  id="image"/>
                <!-- Hides the buttons, as there is no images at beggining -->
                <div id="btn_Bar" style="display:none">
                        <button id="btn_1" value="baby" class="button-xlarge pure-button">Baby</button>
                        <button id="btn_2" value="child" class="button-xlarge pure-button">Child</button>
                        <button id="btn_3" value="teenager" class="button-xlarge pure-button">Teenager</button>
                        <button id="btn_4" value="adult" class="button-xlarge pure-button">Adult</button>
                        <button id="btn_5" value="senior" class="button-xlarge pure-button">Senior</button>
```

```
                    <button id="btn_6" value="not_person" class="button-xlarge pure-button">It is not a person</
                        button>
                </div>
        </div>
</body>
</html>
```

Next, we need to develop the logic in the script. The script will: get the images from a dataset, get the user contribution, the user id, include a golden standard image and send it to our database. Now we can go to the code. As the code is a little more complex than the HTML, we going to show and explain what each part of the code does.

First, let's see the variables of our script. There is no logic here, and the comments explain well what they are for.

```
var images = [];                    //Array to store the images URL;
var contrs = [];                    //Array to store the contributions: {time_start, time_stop,image_url, image_class,
     user_id, contr_id}
var goldenImgsIds =     //Array with our golden images
        ["http://goo.gl/ZXa6MJ",
        "http://goo.gl/VRAKdt"];
var id;                             //Stores the user fingerprinting;
var counter = 0;                    //Sets in which step the user is;
var contrNumber = 5;    //HowMany contribution we want;
var btnBar;                         //Reference to the ButtonBar;
var img;                            //Reference to the image container;
var out;                            //Reference to the output field;
var time = 0;                       //Stores how long the took the contribution;
var stamp = new Date().getTime(); //A stamp to this contribution;
```

When the HTML is built (window.onload), our application begins. We randomly load the images from the remote server, we include a golden standard image among them to help on evaluating the contribution, get references to the HTML components, add listeners to the button(the user action), get a timestamp to identify the user and start collecting the contributions.

```
//Function called when the HTML is done;
window.onload = function(){
        //Connects to the image service, and retrieves images;
        images = loadRemoteImages();
        //Inserts a Golden Standard
        images = goldenStandardIn(images);
        //Gets the references to the html elments;
        img = document.getElementById("image");
        btnBar = document.getElementById("btn_Bar");
        out = document.getElementById("out");
        //Gets the user fingerprinting and attributes it to the id variable;
        //This is an asynchronous function;
        new Fingerprint2().get(function(result, components){
                this.id = result;
        });
        //Starts the application listeners;
        btnBar.addEventListener("click",buttonClick);
        //Starts the first contribution;
        nextContribution();
}
```

A Golden standard is useful to help on judge the user contribution. A Golden Standard image is an image that we already know, and is included to help judge the user contribution. If the user mistake the golden standard, there is a chance that his other contributions are also wrong. In our APP we include among the images, one of those that we already know (goldenImgsIds) into a random position. In this case, we have only two in one example to keep it short.

```
//Randomly inserts a golden image in one of the images position;
function goldenStandardIn(images){
        //The position to put the golden image;
        var r1 = Math.floor(Math.random() * contrNumber);
        //One of the listed golden images;
        var r2 = Math.floor(Math.random() * (goldenImgsIds.length));
        //replaces the position;
        images[r1] = goldenImgsIds[r2];
        return images;
}
```

The contributions are started when we call nextContribution. This function writes in the screen how many contributions the user has done (2/5, for example), loads the image to be annotated on the screen, and when the image is loaded it starts to count how long the user will take to activate the buttons.

```
//Loads the next contribution from the user;
function nextContribution(){
        out.innerHTML = (counter+1)+ '/' + contrNumber;
        //Sets the image to be evaluated and shows the buttons;
        img.src = images[counter];
        //Ao carregar a imagem;
        img.addEventListener("load",function(){
                btnBar.style.display = 'block';
                //Starts counting time;
                time = new Date().getTime();
        });
}
```

When the image is loaded, the user can click on the option he thinks is right. When the button is activated, it starts some logic: the timer is finished and we store the duration of the contribution, we hide the buttons and the image and create a Contribution Object to store all information about that contribution (duration, image-url, user-choice, user-id, contr-id). If the user made enough contributions (in this case 5 times), we send it to the server, otherwise we get the next contribution.

```
//Handles the Click of the buttons;
function buttonClick(event){
        //Stops counting time, and saves its duration;
        time = (new Date().getTime()) - time;
        //Hides the image and blocks the buttons, so users don't click twice;
        img.src = '';
        btnBar.style.display = 'none';
        //Creates an contribution:
        var c = {
                time: time,
                image_url:      images[counter],
                image_class:event.target.value,
                user_id:        id,
                contr_id:       id + stamp
        }
        //Adds the contribution to the contribution array;
        contrs.push(c);
        //Increments the counter;
        counter++;
        //If all the contributions are done, sends the contribution to the database;
        if(counter >= contrNumber){
                sendContributions(contrs);
        //Else, gets the next contribution;
        }else{
                nextContribution();
        }
}
```

When all contributions are done, they are sent one by one to the spreadsheet (storeContribution, that comes next) and we inform it to the user. In the end we hide the figure, the button and invite the user to contribute again.

```
//Send all contributions, one by one;
function sendContributions(cs){
        //Hide all interface components;
        btnBar.style.display = 'none';
        img.style.display = 'none';
        document.getElementById('question').style.display = 'none';
        //Informs that all informations are being sent;
        out.innerHTML = 'Sending␣Answers.'
        for(var i = 0; i < cs.length; i++){
                storeContribution(cs[i]);
        }
        //Thanks the contribution.
        out.innerHTML = 'Answers␣Sent,␣Thank␣YOU␣!!␣:)';
        //Creates a link to reload the page, and contribute again;
        var a = document.createElement('a');
        var linkText = document.createTextNode("my␣title␣text");
        a.innerHTML = "Click␣here␣to␣Contribute␣again";
        a.href = "./index.html";
        document.getElementById('app').appendChild(a);
}
```

This is the function that communicates with Google Spreadsheet.

```
//Sends the contribution to the Database
function storeContribution(c){
        //Your Spreadsheet URL;
        var url = "https://docs.google.com/forms/d/e/1FAIpQLSfoexmSLhvScIx4OEcZxXHx6VikDJjN5QApXMw6ZNqgEgaYVg";
        //The form we send our infromation;
        var action = 'formResponse';
        //The field's id extraceted earlier:
        var fields = ['entry.274168468','entry.1175968285','entry.405326048','entry.1284275155','entry.595916278'];
        //We construct the URL to the database;
        var str = url+'/'+action+'?'     +
                fields[0]+'='+c.time     + '&' +
                fields[1]+'='+c.image_url        + '&' +
                fields[2]+'='+c.image_class      + '&' +
                fields[3]+'='+c.user_id          + '&' +
                fields[4]+'='+c.contr_id;
        //We send the form;
        fetch(str,{mode: 'no-cors'});
}
```

Finally, this is the function to retrieve images from the image dataset, five for each request.

```
//Loads the images from the picture server;
function loadRemoteImages(){
        //Local array to store the images URL;
    var imgs = [];
    //Gets five images from facebook randomly (from a specific range)
    for(var i=0; i < 5; i++) {
                rnd = Math.floor(Math.random() * 1000)%100 + 100000;
        url = "http://graph.facebook.com/v2.5/" + rnd + "/picture?height=300&height=300";
                imgs[i] = url;
    }
    //Retruns the array;
    return imgs;
}
```

3.4.3. Data Analysis

All workers' contribution are stored in a spreadsheet. Hence, we can use multiple approaches to analyse it: use metrics, manual analysis, MOS, its up to you. But one approach that the spreadsheet allows us to do is create a Google App Script that allows us to program an script to analyse data.

Our analysis follows these steps:

1. We have two spreadsheets indeed, one with the contributions and a second one with the Golden Standards;

2. The first step in our algorithm is to order by the contribution id, followed by the creation of an object for each session of an user (the five contributions in a row that he does);

3. Verify if in that session, the user scored the right value to the golden standard. If he missed, all contributions are discarded;

4. For each image, we count how many opinions for each possibility it had;

5. With these values we calculate the mode for each image, finding the opinion on that image.

The results for fifty images from Facebook profiles during 5 minutes with 4 crowd members resulted in 365 contributions. The list of the answers can be found in Appendix A.

3.5. Final Remarks

In this tutorial, we guided you through an overview of what is crowdsourcing and how to use it in diverse multimedia scenarios: text digitalization, audio transcription, video quality assessment and audio translation. We also showed, step by step, how to develop a web application to provide a simple platform to perform some crowdsourcing work, in our case, we performed the annotation of a image dataset about the age of the people in the images. This application lacks control of the crowd and maybe some security (anyone can submit to our form once the url is public), however it provides a free architecture that can be implemented by anyone, it also provides scalability and fast development (in four hours you were able to perform it). Next we showed how to aggregate value to our applications putting it in CrowdFlower Crowdsourcing platform.

3.6. Acknowledgments

This work is funded by the Brazilian Research Funding Agencies CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) and FAPES (Fundação de Amparo à Pesquisa e Inovação do Espírito Santo).

References

Anegekuh, Louis, Sun, Lingfen, & Ifeachor, Emmanuel. 2014. A screening methodology for crowdsourcing video QoE evaluation. *Pages 1152–1157 of: Global Communications Conference (GLOBECOM), 2014 IEEE*. IEEE.

Aran, Oya, Biel, Joan-Isaac, & Gatica-Perez, Daniel. 2014. Broadcasting oneself: Visual discovery of vlogging styles. *Multimedia, IEEE Transactions on*, **16**(1), 201–215.

Baveye, Yoann, Dellandrea, Emmanuel, Chamaret, Christel, & Chen, Liming. 2015. LIRIS-ACCEDE: A video database for affective content analysis. *Affective Computing, IEEE Transactions on*, **6**(1), 43–55.

Borromeo, Ria Mae, & Toyama, Motomichi. 2015. Automatic vs. Crowdsourced Sentiment Analysis. *Pages 90–95 of: Proceedings of the 19th International Database Engineering & Applications Symposium*. ACM.

Burmania, Alec, Parthasarathy, Srinivas, & Busso, Carlos. Increasing the Reliability of Crowdsourcing Evaluations Using Online Quality Assessment.

Carlier, Axel, Ravindra, Guntur, Charvillat, Vincent, & Ooi, Wei Tsang. 2011. Combining content-based analysis and crowdsourcing to improve user interaction with zoomable video. *Pages 43–52 of: Proceedings of the 19th ACM international conference on Multimedia*. ACM.

Cavalcanti, Marcos, & Nepomuceno, Carlos. 2007. *O conhecimento em rede: como implantar projetos de inteligência coletiva*. Elsevier.

Chen, Si, Li, Muyuan, Ren, Kui, & Qiao, Chunming. 2015a. Crowd map: Accurate reconstruction of indoor floor plans from crowdsourced sensor-rich videos. *Pages 1–10 of: Distributed Computing Systems (ICDCS), 2015 IEEE 35th International Conference on*. IEEE.

Chen, Yu-An, Yang, Yi-Hsuan, Wang, Ju-Chiang, & Chen, Homer. 2015b. The AMG1608 dataset for music emotion recognition. *Pages 693–697 of: 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE.

Corney, Jonathan, Lynn, Andrew, Torres, Carmen, Di Maio, Paola, Regli, William, Forbes, Graeme, & Tobin, Lynne. 2010. Towards crowdsourcing translation tasks in library cataloguing, a pilot study. *Pages 572–577 of: 4th IEEE International Conference on Digital Ecosystems and Technologies*. IEEE.

Di Salvo, R, Giordano, D, & Kavasidis, I. 2013. A crowdsourcing approach to support video annotation. *Page 8 of: Proceedings of the International Workshop on Video and Image Ground Truth in Computer Vision Applications*. ACM.

Doan, Anhai, Ramakrishnan, Raghu, & Halevy, Alon Y. 2011. Crowdsourcing systems on the world-wide web. *Communications of the ACM*, **54**(4), 86–96.

Dumitrache, Anca, Aroyo, Lora, Welty, Chris, Sips, Robert-Jan, & Levas, Anthony. 2013. Dr. Detective: combining gamication techniques and crowdsourcing to create a gold standard in medical text. *Pages 16–31 of: Proceedings of the 1st International Conference on Crowdsourcing the Semantic Web-Volume 1030*. CEUR-WS. org.

Estellés-Arolas, Enrique, & González-Ladrón-de Guevara, Fernando. 2012. Towards an integrated crowdsourcing definition. *Journal of Information science*, **38**(2), 189–200.

Ferracani, Andrea, Pezzatini, Daniele, Bertini, Marco, Meucci, Saverio, & Del Bimbo, Alberto. 2015. A System for Video Recommendation using Visual Saliency, Crowdsourced and Automatic Annotations. *Pages 757–758 of: Proceedings of the 23rd Annual ACM Conference on Multimedia Conference*. ACM.

Figuerola Salas, Óscar, Adzic, Velibor, Shah, Akash, & Kalva, Hari. 2013. Assessing internet video quality using crowdsourcing. *Pages 23–28 of: Proceedings of the 2nd ACM international workshop on Crowdsourcing for Multimedia*. ACM.

Foncubierta Rodríguez, Antonio, & Müller, Henning. 2012. Ground truth generation in medical imaging: a crowdsourcing-based iterative approach. *Pages 9–14 of: Proceedings of the ACM multimedia 2012 workshop on Crowdsourcing for multimedia*. ACM.

Freitas, Joao, Calado, António, Braga, Daniela, Silva, Pedro, & Dias, M. 2010. Crowdsourcing platform for large-scale speech data collection. *Proc. FALA*.

Ghadiyaram, Deepti, & Bovik, Alan C. 2016. Massive online crowdsourced study of subjective and objective picture quality. *IEEE Transactions on Image Processing*, **25**(1), 372–387.

Guo, Xintong, Gao, Hong, & Wang, Hongzhi. 2015. Image Clustering Based on the Human Intelligence. *Pages 366–373 of: Intelligent Systems and Knowledge Engineering (ISKE), 2015 10th International Conference on*. IEEE.

Hoonlor, Apirak, Ayudhya, Srisupa Palakvangsa Na, Harnmetta, Sukritta, Kitpanon, Suttichai, & Khlaprasit, Krisanat. 2015. UCap: A crowdsourcing application for the visually impaired and blind persons on Android smartphone. *Pages 1–6 of: 2015 International Computer Science and Engineering Conference (ICSEC)*. IEEE.

Hossfeld, Tobias, Seufert, Michael, Sieber, Christian, & Zinner, Thomas. 2014. Assessing effect sizes of influence factors towards a QoE model for HTTP adaptive streaming. *Pages 111–116 of: Quality of Multimedia Experience (QoMEX), 2014 Sixth International Workshop on*. IEEE.

Hoßfeld, Tobias, Keimel, Christian, Hirth, Matthias, Gardlo, Bruno, Habigt, Julian, Diepold, Klaus, & Tran-Gia, Phuoc. 2014. Best practices for QoE crowdtesting: QoE assessment with crowdsourcing. *Multimedia, IEEE Transactions on*, **16**(2), 541–558.

Howe, Jeff. 2006. The rise of crowdsourcing. *Wired magazine*, **14**(6), 1–4.

Huberman, Bernardo A, Romero, Daniel M, & Wu, Fang. 2009. Crowdsourcing, attention and productivity. *Journal of Information Science*.

Keimel, Christian, Habigt, Julian, & Diepold, Klaus. 2012. Challenges in crowd-based video quality assessment. *Pages 13–18 of: Quality of Multimedia Experience (QoMEX), 2012 Fourth International Workshop on*. IEEE.

Kim, Samuel, Georgiou, Panayiotis G, & Narayanan, Shrikanth. 2013. Annotation and classification of Political advertisements. *Pages 1092–1096 of: INTERSPEECH*.

Lasecki, Walter S, Rello, Luz, & Bigham, Jeffrey P. 2015. Measuring text simplification with the crowd. *Page 4 of: Proceedings of the 12th Web for All Conference*. ACM.

Law, Edith, & Ahn, Luis von. 2011. Human computation. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, **5**(3), 1–121.

Le, John, Edmonds, Andy, Hester, Vaughn, & Biewald, Lukas. 2010. Ensuring quality in crowdsourced search relevance evaluation: The effects of training question distribution. *Pages 21–26 of: SIGIR 2010 workshop on crowdsourcing for search evaluation*.

Lee, Chia-ying, & Glass, James R. 2011. A Transcription Task for Crowdsourcing with Automatic Quality Control. *Pages 3041–3044 of: Interspeech*, vol. 11. Citeseer.

Leftheriotis, Ioannis, Gkonela, Chrysoula, & Chorianopoulos, Konstantinos. 2012. Efficient video indexing on the web: A system that crowdsources user interactions with a video player. *Pages 123–131 of: User Centric Media*. Springer.

Levy, Pierre. 1994. *Les technologies de lÍntelligence–Lávenir de la pensée à lÃćâĆňâĎćÃćâĆňËIJere informatique*.

Liao, Chen-Chih, & Hsu, Cheng-Hsin. 2013. A detour planning algorithm in crowdsourcing systems for multimedia content gathering. *Pages 55–60 of: Proceedings of the 5th Workshop on Mobile Video*. ACM.

MacLean, Diana Lynn, & Heer, Jeffrey. 2013. Identifying medical terms in patient-authored text: a crowdsourcing-based approach. *Journal of the American Medical Informatics Association*, **20**(6), 1120–1127.

Pang, Derek, Halawa, Sherif, Cheung, Ngai-Man, & Girod, Bernd. 2011. Mobile interactive region-of-interest video streaming with crowd-driven prefetching. *Pages 7–12 of: Proceedings of the 2011 international ACM workshop on Interactive multimedia on mobile and portable devices*. ACM.

Park, Sunghyun, Shoemark, Philippa, & Morency, Louis-Philippe. 2014. Toward crowdsourcing micro-level behavior annotations: the challenges of interface, training, and generalization. *Pages 37–46 of: Proceedings of the 19th international conference on Intelligent User Interfaces*. ACM.

Pedersen, Jay, Kocsis, David, Tripathi, Abhishek, Tarrell, Alvin, Weerakoon, Aruna, Tahmasbi, Nargess, Xiong, Jie, Deng, Wei, Oh, Onook, & de Vreede, Gert-Jan. 2013. Conceptual foundations of crowdsourcing: A review of IS research. *Pages 579–588 of: System Sciences (HICSS), 2013 46th Hawaii International Conference on*. IEEE.

Pinto, José Pedro, & Viana, Paula. 2013. TAG4VD: a game for collaborative video annotation. *Pages 25–28 of: Proceedings of the 2013 ACM international workshop on Immersive media experiences*. ACM.

Rainer, Benjamin, & Timmerer, Christian. 2014. A quality of experience model for adaptive media playout. *Pages 177–182 of: Quality of Multimedia Experience (QoMEX), 2014 Sixth International Workshop on*. IEEE.

Ribeiro, Flávio, Florencio, Dinei, & Nascimento, Vítor. 2011. Crowdsourcing subjective image quality evaluation. *Pages 3097–3100 of: 2011 18th IEEE International Conference on Image Processing*. IEEE.

Rizvi, AS M, Ahmed, Shamir, Bashir, Minhajul, & Uddin, Md Yusuf Sarwar. 2015. MediaServ: Resource optimization in subscription based media crowdsourcing. *Pages 1–5 of: Networking Systems and Security (NSysS), 2015 International Conference on*. IEEE.

Saad, Michele A, McKnight, Patrick, Quartuccio, Jacob, Nicholas, David, Jaladi, Ramesh, & Corriveau, Philip. 2016. Consumer-photo quality assessment: Challenges and pitfalls in crowdsourcing. *Pages 1–6 of: 2016 Eighth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE.

Saito, Takashi. 2015. A framework of human-based speech transcription with a speech chunking front-end. *Pages 125–128 of: 2015 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*. IEEE.

Schlippe, Tim, Zhu, Chenfei, Lemcke, Daniel, & Schultz, Tanja. 2013. Statistical machine translation based text normalization with crowdsourcing. *Pages 8406–8410 of: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE.

Venkatagiri, Seshadri Padmanabha, Chan, Mun Choon, Ooi, Wei Tsang, & Chiam, Jia Han. 2015. On Demand Retrieval of Crowdsourced Mobile Video. *Sensors Journal, IEEE*, **15**(5), 2632–2642.

von Ahn, Luis. 2013. Duolingo: learn a language for free while helping to translate the web. *Pages 1–2 of: Proceedings of the 2013 international conference on Intelligent user interfaces*. ACM.

Von Ahn, Luis, Maurer, Benjamin, McMillen, Colin, Abraham, David, & Blum, Manuel. 2008. recaptcha: Human-based character recognition via web security measures. *Science*, **321**(5895), 1465–1468.

Vondrick, Carl, Patterson, Donald, & Ramanan, Deva. 2013. Efficiently scaling up crowdsourced video annotation. *International Journal of Computer Vision*, **101**(1), 184–204.

Wilk, Stefan, Kopf, Stephan, & Effelsberg, Wolfgang. 2015. Video composition by the crowd: a system to compose user-generated videos in near real-time. *Pages 13–24 of: Proceedings of the 6th ACM Multimedia Systems Conference*. ACM.

Zegarra Rodriguez, Demostenes, Lopes Rosa, Renata, & Bressan, Graca. 2015. No-reference video quality metric for streaming service using DASH standard. *Pages 106–107 of: Consumer Electronics (ICCE), 2015 IEEE International Conference on*. IEEE.

A. Answers from APP test

Timestamp	time	image_url	image_classificati	user_id	contr_id
8/30/2016 15:22:46	2458	http://goo.gl/QsuqUq	adult	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581327259
8/30/2016 15:22:46	7399	http://graph.facebook.com/v2.5/100043/picture?height=300	teenager	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581327259
8/30/2016 15:22:46	4218	http://graph.facebook.com/v2.5/100036/picture?height=300	teenager	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581327259
8/30/2016 15:22:46	13545	http://graph.facebook.com/v2.5/100045/picture?height=300	not_person	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581327259
8/30/2016 15:22:46	5269	http://graph.facebook.com/v2.5/100007/picture?height=300	adult	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581327259
8/30/2016 15:23:11	1691	http://goo.gl/VRAKdt	senior	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581369492
8/30/2016 15:23:11	8077	http://graph.facebook.com/v2.5/100019/picture?height=300	adult	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581369492
8/30/2016 15:23:12	2217	http://graph.facebook.com/v2.5/100017/picture?height=300	adult	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581369492
8/30/2016 15:23:12	4459	http://graph.facebook.com/v2.5/100010/picture?height=300	not_person	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581369492
8/30/2016 15:23:12	1909	http://graph.facebook.com/v2.5/100043/picture?height=300	teenager	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581369492
8/30/2016 15:24:08	1739	http://graph.facebook.com/v2.5/100000/picture?height=300	not_person	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581394784
8/30/2016 15:24:08	8791	http://goo.gl/QsuqUq	adult	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581394784
8/30/2016 15:24:08	19288	http://graph.facebook.com/v2.5/100022/picture?height=300	adult	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581394784
8/30/2016 15:24:08	1794	http://graph.facebook.com/v2.5/100029/picture?height=300	not_person	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581394784
8/30/2016 15:24:08	17694	http://graph.facebook.com/v2.5/100037/picture?height=300	adult	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581394784
8/30/2016 15:24:32	3437	http://graph.facebook.com/v2.5/100025/picture?height=300	teenager	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581453053
8/30/2016 15:24:32	889	http://graph.facebook.com/v2.5/100002/picture?height=300	not_person	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581453053
8/30/2016 15:24:32	6435	http://graph.facebook.com/v2.5/100029/picture?height=300	not_person	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581453053
8/30/2016 15:24:32	1425	http://graph.facebook.com/v2.5/100028/picture?height=300	not_person	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581453053
8/30/2016 15:24:32	1545	http://goo.gl/QsuqUq	adult	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581453053
8/30/2016 15:24:45	1387	http://graph.facebook.com/v2.5/100030/picture?height=300	adult	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581474094
8/30/2016 15:24:45	1497	http://graph.facebook.com/v2.5/100033/picture?height=300	not_person	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581474094
8/30/2016 15:24:45	1071	http://goo.gl/QsuqUq	adult	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581474094
8/30/2016 15:24:45	3161	http://graph.facebook.com/v2.5/100023/picture?height=300	adult	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581474094
8/30/2016 15:24:45	1244	http://graph.facebook.com/v2.5/100019/picture?height=300	adult	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581474094
8/30/2016 15:25:21	1201	http://graph.facebook.com/v2.5/100029/picture?height=300	not_person	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581487957
8/30/2016 15:25:21	6070	http://graph.facebook.com/v2.5/100014/picture?height=300	adult	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581487957
8/30/2016 15:25:21	1886	http://graph.facebook.com/v2.5/100009/picture?height=300	adult	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581487957
8/30/2016 15:25:21	3638	http://goo.gl/QsuqUq	adult	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581487957
8/30/2016 15:25:21	16416	http://graph.facebook.com/v2.5/100049/picture?height=300	not_person	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581487957
8/30/2016 15:25:50	1752	http://goo.gl/VRAKdt	senior	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581527929
8/30/2016 15:25:50	7628	http://graph.facebook.com/v2.5/100000/picture?height=300	not_person	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581527929
8/30/2016 15:25:50	8316	http://graph.facebook.com/v2.5/100018/picture?height=300	not_person	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581527929
8/30/2016 15:25:50	676	http://graph.facebook.com/v2.5/100029/picture?height=300	not_person	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581527929
8/30/2016 15:25:50	913	http://graph.facebook.com/v2.5/100002/picture?height=300	not_person	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581527929
8/30/2016 15:26:09	1218	http://graph.facebook.com/v2.5/100043/picture?height=300	teenager	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581555786
8/30/2016 15:26:09	2336	http://graph.facebook.com/v2.5/100015/picture?height=300	adult	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581555786
8/30/2016 15:26:09	2152	http://goo.gl/QsuqUq	adult	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581555786
8/30/2016 15:26:09	1272	http://graph.facebook.com/v2.5/100000/picture?height=300	not_person	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581555786
8/30/2016 15:26:09	2594	http://graph.facebook.com/v2.5/100032/picture?height=300	adult	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581555786
8/30/2016 15:26:28	3725	http://graph.facebook.com/v2.5/100005/picture?height=300	adult	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581572613
8/30/2016 15:26:28	1484	http://graph.facebook.com/v2.5/100048/picture?height=300	not_person	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581572613
8/30/2016 15:26:28	1255	http://graph.facebook.com/v2.5/100046/picture?height=300	not_person	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581572613
8/30/2016 15:26:28	1242	http://goo.gl/VRAKdt	senior	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581572613
8/30/2016 15:26:28	5304	http://graph.facebook.com/v2.5/100015/picture?height=300	adult	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581572613
8/30/2016 15:26:52	6656	http://graph.facebook.com/v2.5/100038/picture?height=300	not_person	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581592074
8/30/2016 15:26:52	1185	http://graph.facebook.com/v2.5/100002/picture?height=300	not_person	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581592074
8/30/2016 15:26:52	5840	http://graph.facebook.com/v2.5/100031/picture?height=300	not_person	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581592074
8/30/2016 15:26:52	1316	http://goo.gl/VRAKdt	senior	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581592074
8/30/2016 15:26:52	816	http://graph.facebook.com/v2.5/100034/picture?height=300	not_person	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581592074
8/30/2016 15:27:17	1724	http://graph.facebook.com/v2.5/100014/picture?height=300	adult	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581614220
8/30/2016 15:27:17	1591	http://graph.facebook.com/v2.5/100038/picture?height=300	not_person	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581614220
8/30/2016 15:27:17	1418	http://goo.gl/VRAKdt	senior	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581614220
8/30/2016 15:27:17	8861	http://graph.facebook.com/v2.5/100018/picture?height=300	not_person	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581614220
8/30/2016 15:27:17	7363	http://graph.facebook.com/v2.5/100032/picture?height=300	adult	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581614220
8/30/2016 15:28:12	1727	http://graph.facebook.com/v2.5/100005/picture?height=300	adult	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581641657
8/30/2016 15:28:12	1341	http://goo.gl/VRAKdt	senior	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581641657
8/30/2016 15:28:12	1211	http://graph.facebook.com/v2.5/100028/picture?height=300	not_person	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581641657
8/30/2016 15:28:12	40585	http://graph.facebook.com/v2.5/100046/picture?height=300	not_person	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581641657
8/30/2016 15:28:12	2887	http://graph.facebook.com/v2.5/100036/picture?height=300	teenager	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581641657
8/30/2016 15:28:39	2849	http://graph.facebook.com/v2.5/100047/picture?height=300	adult	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581694996
8/30/2016 15:28:39	1886	http://graph.facebook.com/v2.5/100004/picture?height=300	not_person	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581694996
8/30/2016 15:28:39	8849	http://graph.facebook.com/v2.5/100034/picture?height=300	not_person	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581694996
8/30/2016 15:28:39	6483	http://graph.facebook.com/v2.5/100014/picture?height=300	adult	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581694996
8/30/2016 15:28:39	1690	http://goo.gl/VRAKdt	senior	066b471d3fc1c2fa4c216b2d7e099242	066b471d3fc1c2fa4c216b2d7e0992421472581694996
8/30/2016 15:23:43	4814	http://graph.facebook.com/v2.5/100043/picture?height=300	adult	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581347639
8/30/2016 15:23:43	4602	http://graph.facebook.com/v2.5/100007/picture?height=300	adult	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581347639
8/30/2016 15:23:43	12646	http://graph.facebook.com/v2.5/100039/picture?height=300	adult	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581347639
8/30/2016 15:23:43	2956	http://graph.facebook.com/v2.5/100041/picture?height=300	adult	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581347639
8/30/2016 15:23:43	3843	http://goo.gl/VRAKdt	senior	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581347639
8/30/2016 15:24:08	2350	http://graph.facebook.com/v2.5/100011/picture?height=300	not_person	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581386853
8/30/2016 15:24:08	1124	http://graph.facebook.com/v2.5/100028/picture?height=300	not_person	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581386853
8/30/2016 15:24:09	3104	http://graph.facebook.com/v2.5/100031/picture?height=300	not_person	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581386853
8/30/2016 15:24:09	2507	http://goo.gl/QsuqUq	adult	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581386853
8/30/2016 15:24:09	7733	http://graph.facebook.com/v2.5/100042/picture?height=300	not_person	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581386853
8/30/2016 15:24:23	3298	http://graph.facebook.com/v2.5/100005/picture?height=300	not_person	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581411678
8/30/2016 15:24:23	1390	http://graph.facebook.com/v2.5/100000/picture?height=300	not_person	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581411678

Timestamp	time	image_url	image_classificati	user_id	contr_id
8/30/2016 15:24:23	924	http://goo.gl/QsuqUq	adult	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581411678
8/30/2016 15:24:23	643	http://graph.facebook.com/v2.5/100008/picture?height=300	not_person	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581411678
8/30/2016 15:24:23	1951	http://graph.facebook.com/v2.5/100041/picture?height=300	adult	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581411678
8/30/2016 15:24:43	2953	http://graph.facebook.com/v2.5/100019/picture?height=300	adult	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581425386
8/30/2016 15:24:43	1735	http://graph.facebook.com/v2.5/100002/picture?height=300	not_person	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581425386
8/30/2016 15:24:43	1956	http://graph.facebook.com/v2.5/100017/picture?height=300	adult	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581425386
8/30/2016 15:24:43	2193	http://graph.facebook.com/v2.5/100030/picture?height=300	adult	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581425386
8/30/2016 15:24:43	4432	http://goo.gl/QsuqUq	adult	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581425386
8/30/2016 15:25:13	2112	http://graph.facebook.com/v2.5/100012/picture?height=300	not_person	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581447355
8/30/2016 15:25:13	4965	http://graph.facebook.com/v2.5/100004/picture?height=300	not_person	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581447355
8/30/2016 15:25:13	3539	http://goo.gl/QsuqUq	adult	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581447355
8/30/2016 15:25:13	9518	http://graph.facebook.com/v2.5/100042/picture?height=300	not_person	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581447355
8/30/2016 15:25:13	1532	http://graph.facebook.com/v2.5/100003/picture?height=300	not_person	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581447355
8/30/2016 15:25:30	535	http://graph.facebook.com/v2.5/100018/picture?height=300	not_person	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581476195
8/30/2016 15:25:29	2165	http://graph.facebook.com/v2.5/100026/picture?height=300	not_person	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581476195
8/30/2016 15:25:29	1864	http://graph.facebook.com/v2.5/100001/picture?height=300	not_person	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581476195
8/30/2016 15:25:29	2388	http://goo.gl/VRAKdt	senior	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581476195
8/30/2016 15:25:29	1861	http://graph.facebook.com/v2.5/100035/picture?height=300	not_person	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581476195
8/30/2016 15:25:42	1045	http://graph.facebook.com/v2.5/100002/picture?height=300	not_person	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581493318
8/30/2016 15:25:42	1503	http://goo.gl/VRAKdt	senior	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581493318
8/30/2016 15:25:42	2048	http://graph.facebook.com/v2.5/100043/picture?height=300	adult	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581493318
8/30/2016 15:25:42	1018	http://graph.facebook.com/v2.5/100004/picture?height=300	not_person	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581493318
8/30/2016 15:25:42	511	http://graph.facebook.com/v2.5/100018/picture?height=300	not_person	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581493318
8/30/2016 15:25:57	709	http://goo.gl/QsuqUq	adult	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581507577
8/30/2016 15:25:57	603	http://graph.facebook.com/v2.5/100043/picture?height=300	adult	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581507577
8/30/2016 15:25:57	1066	http://graph.facebook.com/v2.5/100003/picture?height=300	not_person	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581507577
8/30/2016 15:25:57	1880	http://graph.facebook.com/v2.5/100039/picture?height=300	adult	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581507577
8/30/2016 15:25:58	2961	http://graph.facebook.com/v2.5/100016/picture?height=300	adult	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581507577
8/30/2016 15:26:18	1875	http://graph.facebook.com/v2.5/100044/picture?height=300	adult	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581524609
8/30/2016 15:26:18	2905	http://graph.facebook.com/v2.5/100031/picture?height=300	not_person	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581524609
8/30/2016 15:26:18	1480	http://goo.gl/VRAKdt	senior	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581524609
8/30/2016 15:26:18	1883	http://graph.facebook.com/v2.5/100026/picture?height=300	not_person	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581524609
8/30/2016 15:26:18	2759	http://graph.facebook.com/v2.5/100001/picture?height=300	not_person	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581524609
8/30/2016 15:26:59	4686	http://graph.facebook.com/v2.5/100000/picture?height=300	not_person	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581549657
8/30/2016 15:26:59	3991	http://goo.gl/VRAKdt	senior	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581549657
8/30/2016 15:26:59	1735	http://graph.facebook.com/v2.5/100024/picture?height=300	adult	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581549657
8/30/2016 15:26:59	9343	http://graph.facebook.com/v2.5/100022/picture?height=300	not_person	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581549657
8/30/2016 15:26:59	6212	http://graph.facebook.com/v2.5/100042/picture?height=300	not_person	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581549657
8/30/2016 15:27:19	1334	http://graph.facebook.com/v2.5/100011/picture?height=300	not_person	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581587715
8/30/2016 15:27:19	5268	http://goo.gl/QsuqUq	adult	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581587715
8/30/2016 15:27:19	484	http://graph.facebook.com/v2.5/100004/picture?height=300	not_person	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581587715
8/30/2016 15:27:19	539	http://graph.facebook.com/v2.5/100033/picture?height=300	not_person	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581587715
8/30/2016 15:27:19	1671	http://graph.facebook.com/v2.5/100022/picture?height=300	not_person	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581587715
8/30/2016 15:27:38	3588	http://graph.facebook.com/v2.5/100026/picture?height=300	adult	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581603418
8/30/2016 15:27:38	3343	http://graph.facebook.com/v2.5/100010/picture?height=300	not_person	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581603418
8/30/2016 15:27:38	1029	http://goo.gl/QsuqUq	adult	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581603418
8/30/2016 15:27:38	1561	http://graph.facebook.com/v2.5/100013/picture?height=300	adult	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581603418
8/30/2016 15:27:38	1296	http://graph.facebook.com/v2.5/100015/picture?height=300	adult	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581603418
8/30/2016 15:27:52	2497	http://graph.facebook.com/v2.5/100020/picture?height=300	adult	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581621952
8/30/2016 15:27:52	1043	http://goo.gl/VRAKdt	senior	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581621952
8/30/2016 15:27:52	1554	http://graph.facebook.com/v2.5/100017/picture?height=300	adult	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581621952
8/30/2016 15:27:52	1437	http://graph.facebook.com/v2.5/100019/picture?height=300	adult	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581621952
8/30/2016 15:27:52	1010	http://graph.facebook.com/v2.5/100004/picture?height=300	not_person	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581621952
8/30/2016 15:28:14	650	http://goo.gl/QsuqUq	adult	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581635318
8/30/2016 15:28:14	2500	http://graph.facebook.com/v2.5/100021/picture?height=300	adult	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581635318
8/30/2016 15:28:14	1597	http://graph.facebook.com/v2.5/100026/picture?height=300	adult	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581635318
8/30/2016 15:28:14	3186	http://graph.facebook.com/v2.5/100035/picture?height=300	adult	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581635318
8/30/2016 15:28:14	8486	http://graph.facebook.com/v2.5/100021/picture?height=300	adult	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581635318
8/30/2016 15:28:37	11229	http://graph.facebook.com/v2.5/100036/picture?height=300	adult	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581656912
8/30/2016 15:28:37	819	http://graph.facebook.com/v2.5/100030/picture?height=300	adult	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581656912
8/30/2016 15:28:37	1234	http://graph.facebook.com/v2.5/100022/picture?height=300	not_person	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581656912
8/30/2016 15:28:37	728	http://goo.gl/VRAKdt	senior	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581656912
8/30/2016 15:28:37	3213	http://graph.facebook.com/v2.5/100030/picture?height=300	adult	3a726a4789f6cde1069bc5805ba9b51c	3a726a4789f6cde1069bc5805ba9b51c1472581656912
8/30/2016 15:23:25	13098	http://goo.gl/VRAKdt	senior	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581365357
8/30/2016 15:23:25	2673	http://graph.facebook.com/v2.5/100038/picture?height=300	not_person	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581365357
8/30/2016 15:23:25	3515	http://graph.facebook.com/v2.5/100004/picture?height=300	not_person	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581365357
8/30/2016 15:23:25	2508	http://graph.facebook.com/v2.5/100007/picture?height=300	adult	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581365357
8/30/2016 15:23:25	11089	http://graph.facebook.com/v2.5/100045/picture?height=300	not_person	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581365357
8/30/2016 15:23:39	1396	http://graph.facebook.com/v2.5/100038/picture?height=300	not_person	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581415310
8/30/2016 15:23:39	2869	http://graph.facebook.com/v2.5/100036/picture?height=300	adult	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581415310
8/30/2016 15:23:39	1129	http://graph.facebook.com/v2.5/100029/picture?height=300	not_person	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581415310
8/30/2016 15:23:39	1300	http://goo.gl/VRAKdt	senior	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581415310
8/30/2016 15:23:39	1174	http://graph.facebook.com/v2.5/100049/picture?height=300	not_person	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581415310
8/30/2016 15:24:00	5674	http://graph.facebook.com/v2.5/100035/picture?height=300	adult	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581429337
8/30/2016 15:24:00	1545	http://graph.facebook.com/v2.5/100015/picture?height=300	adult	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581429337
8/30/2016 15:24:00	1220	http://graph.facebook.com/v2.5/100035/picture?height=300	adult	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581429337
8/30/2016 15:24:00	1182	http://goo.gl/VRAKdt	senior	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581429337

Timestamp	time	image_url	image_classificati	user_id	contr_id
8/30/2016 15:24:00	4236	http://graph.facebook.com/v2.5/100019/picture?height=300	adult	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581429337
8/30/2016 15:24:28	1461	http://graph.facebook.com/v2.5/100038/picture?height=300	not_person	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581454633
8/30/2016 15:24:28	1060	http://graph.facebook.com/v2.5/100049/picture?height=300	not_person	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581454633
8/30/2016 15:24:28	2481	http://graph.facebook.com/v2.5/100040/picture?height=300	adult	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581454633
8/30/2016 15:24:28	5231	http://graph.facebook.com/v2.5/100010/picture?height=300	senior	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581454633
8/30/2016 15:24:28	4759	http://goo.gl/QsuqUq	adult	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581454633
8/30/2016 15:24:47	1995	http://goo.gl/QsuqUq	adult	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581478683
8/30/2016 15:24:47	1700	http://graph.facebook.com/v2.5/100041/picture?height=300	adult	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581478683
8/30/2016 15:24:47	3310	http://graph.facebook.com/v2.5/100022/picture?height=300	adult	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581478683
8/30/2016 15:24:47	2074	http://graph.facebook.com/v2.5/100014/picture?height=300	adult	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581478683
8/30/2016 15:24:47	848	http://graph.facebook.com/v2.5/100046/picture?height=300	not_person	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581478683
8/30/2016 15:25:01	1671	http://goo.gl/VRAKdt	senior	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581494980
8/30/2016 15:25:01	3777	http://graph.facebook.com/v2.5/100042/picture?height=300	not_person	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581494980
8/30/2016 15:25:01	1872	http://graph.facebook.com/v2.5/100011/picture?height=300	not_person	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581494980
8/30/2016 15:25:01	1541	http://graph.facebook.com/v2.5/100035/picture?height=300	adult	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581494980
8/30/2016 15:25:01	1356	http://graph.facebook.com/v2.5/100049/picture?height=300	not_person	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581494980
8/30/2016 15:25:20	2928	http://graph.facebook.com/v2.5/100040/picture?height=300	adult	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581512833
8/30/2016 15:25:20	2096	http://graph.facebook.com/v2.5/100017/picture?height=300	adult	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581512833
8/30/2016 15:25:20	1924	http://graph.facebook.com/v2.5/100013/picture?height=300	adult	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581512833
8/30/2016 15:25:20	872	http://goo.gl/QsuqUq	adult	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581512833
8/30/2016 15:25:20	1920	http://graph.facebook.com/v2.5/100029/picture?height=300	not_person	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581512833
8/30/2016 15:25:35	1296	http://graph.facebook.com/v2.5/100014/picture?height=300	adult	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581529591
8/30/2016 15:25:35	1174	http://goo.gl/VRAKdt	senior	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581529591
8/30/2016 15:25:36	1591	http://graph.facebook.com/v2.5/100038/picture?height=300	not_person	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581529591
8/30/2016 15:25:36	1412	http://graph.facebook.com/v2.5/100038/picture?height=300	not_person	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581529591
8/30/2016 15:25:36	4294	http://graph.facebook.com/v2.5/100032/picture?height=300	teenager	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581529591
8/30/2016 15:25:49	1475	http://graph.facebook.com/v2.5/100016/picture?height=300	adult	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581544318
8/30/2016 15:25:49	1787	http://graph.facebook.com/v2.5/100038/picture?height=300	not_person	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581544318
8/30/2016 15:25:49	1402	http://goo.gl/VRAKdt	senior	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581544318
8/30/2016 15:25:49	1372	http://graph.facebook.com/v2.5/100034/picture?height=300	not_person	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581544318
8/30/2016 15:25:49	1487	http://graph.facebook.com/v2.5/100039/picture?height=300	adult	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581544318
8/30/2016 15:26:04	2882	http://graph.facebook.com/v2.5/100040/picture?height=300	adult	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581557896
8/30/2016 15:26:04	728	http://goo.gl/QsuqUq	adult	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581557896
8/30/2016 15:26:04	1820	http://graph.facebook.com/v2.5/100012/picture?height=300	adult	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581557896
8/30/2016 15:26:04	2514	http://graph.facebook.com/v2.5/100045/picture?height=300	not_person	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581557896
8/30/2016 15:26:04	1444	http://graph.facebook.com/v2.5/100027/picture?height=300	adult	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581557896
8/30/2016 15:26:21	1629	http://graph.facebook.com/v2.5/100023/picture?height=300	adult	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581573180
8/30/2016 15:26:21	1355	http://graph.facebook.com/v2.5/100004/picture?height=300	not_person	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581573180
8/30/2016 15:26:21	1558	http://graph.facebook.com/v2.5/100033/picture?height=300	not_person	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581573180
8/30/2016 15:26:21	4131	http://goo.gl/VRAKdt	senior	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581573180
8/30/2016 15:26:21	1429	http://graph.facebook.com/v2.5/100024/picture?height=300	adult	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581573180
8/30/2016 15:26:32	1097	http://goo.gl/VRAKdt	senior	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581590882
8/30/2016 15:26:32	958	http://graph.facebook.com/v2.5/100016/picture?height=300	adult	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581590882
8/30/2016 15:26:32	993	http://graph.facebook.com/v2.5/100039/picture?height=300	adult	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581590882
8/30/2016 15:26:32	1060	http://graph.facebook.com/v2.5/100034/picture?height=300	not_person	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581590882
8/30/2016 15:26:32	1602	http://graph.facebook.com/v2.5/100035/picture?height=300	adult	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581590882
8/30/2016 15:26:43	860	http://goo.gl/QsuqUq	adult	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581601564
8/30/2016 15:26:43	1343	http://graph.facebook.com/v2.5/100017/picture?height=300	adult	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581601564
8/30/2016 15:26:43	1286	http://graph.facebook.com/v2.5/100002/picture?height=300	not_person	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581601564
8/30/2016 15:26:43	1075	http://graph.facebook.com/v2.5/100036/picture?height=300	adult	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581601564
8/30/2016 15:26:43	1319	http://graph.facebook.com/v2.5/100012/picture?height=300	adult	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581601564
8/30/2016 15:26:55	1187	http://graph.facebook.com/v2.5/100046/picture?height=300	not_person	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581612201
8/30/2016 15:26:55	994	http://graph.facebook.com/v2.5/100034/picture?height=300	not_person	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581612201
8/30/2016 15:26:55	722	http://graph.facebook.com/v2.5/100029/picture?height=300	not_person	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581612201
8/30/2016 15:26:55	2057	http://goo.gl/QsuqUq	adult	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581612201
8/30/2016 15:26:55	2047	http://graph.facebook.com/v2.5/100033/picture?height=300	not_person	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581612201
8/30/2016 15:27:06	1338	http://graph.facebook.com/v2.5/100041/picture?height=300	adult	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581623562
8/30/2016 15:27:06	665	http://goo.gl/QsuqUq	adult	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581623562
8/30/2016 15:27:06	991	http://graph.facebook.com/v2.5/100035/picture?height=300	adult	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581623562
8/30/2016 15:27:06	1615	http://graph.facebook.com/v2.5/100019/picture?height=300	adult	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581623562
8/30/2016 15:27:06	1581	http://graph.facebook.com/v2.5/100029/picture?height=300	not_person	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581623562
8/30/2016 15:27:18	1288	http://graph.facebook.com/v2.5/100042/picture?height=300	not_person	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581636799
8/30/2016 15:27:18	1129	http://goo.gl/VRAKdt	senior	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581636799
8/30/2016 15:27:18	831	http://graph.facebook.com/v2.5/100024/picture?height=300	adult	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581636799
8/30/2016 15:27:18	982	http://graph.facebook.com/v2.5/100036/picture?height=300	adult	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581636799
8/30/2016 15:27:18	1034	http://graph.facebook.com/v2.5/100045/picture?height=300	not_person	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581636799
8/30/2016 15:27:27	1599	http://graph.facebook.com/v2.5/100002/picture?height=300	not_person	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581646426
8/30/2016 15:27:27	470	http://graph.facebook.com/v2.5/100003/picture?height=300	not_person	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581646426
8/30/2016 15:27:27	440	http://graph.facebook.com/v2.5/100004/picture?height=300	not_person	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581646426
8/30/2016 15:27:27	1070	http://goo.gl/QsuqUq	adult	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581646426
8/30/2016 15:27:28	1394	http://graph.facebook.com/v2.5/100045/picture?height=300	not_person	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581646426
8/30/2016 15:27:37	1065	http://graph.facebook.com/v2.5/100006/picture?height=300	not_person	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581656083
8/30/2016 15:27:37	988	http://graph.facebook.com/v2.5/100035/picture?height=300	adult	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581656083
8/30/2016 15:27:37	426	http://graph.facebook.com/v2.5/100029/picture?height=300	not_person	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581656083
8/30/2016 15:27:37	1655	http://goo.gl/QsuqUq	adult	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581656083
8/30/2016 15:27:37	1027	http://graph.facebook.com/v2.5/100027/picture?height=300	adult	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581656083
8/30/2016 15:27:50	1777	http://graph.facebook.com/v2.5/100042/picture?height=300	not_person	92cba199f84acdafc2b713302cdc4564	92cba199f84acdafc2b713302cdc45641472581665324

Dr. Celso Alberto Saibel Santos is Dr. in Fondamentalle et Informatique Parallelisme by the Université Paul Sabatier Toulouse III (1999), MSc. in Engineering of Electronic Systems for the POLI-USP (1994) and Electrical Engineer UFES (1991). Has great experience in guidance at all levels of training and coordination of research and innovation projects, working mainly in the areas of multimedia, hypermedia and the Web. He published numerous scientific papers related to these areas in several vehicles. He has worked at University of Salvador (Unifacs), from 2001 to 2009, and at the Dep. of Computer Science of the Federal University of Bahia (UFBA), from 2009 to 2012. Currently, he is adjoint professor at Department of Informatics of the UFES and fellow of Productivity and Technological Development of Innovative Extension from CNPq-Brazil, since 2012.

Marcello Novaes de Amorim has received his MSc. in Informatics (2007) and BSc. in Computer Science (2005), both from Federal University of Espírito Santo (UFES). He is currently Ph.D. candidate in Computer Science also at UFES and has special interest in assistive technologies, crowdsourcing, mobile devices, education and application of new technologies.

Ricardo Mendes Costa Segundo has received his MSc. in Informatics (2011) and BSc. in Computer Science (2009) from Federal University of Paraíba (UFPB). He is currently a PhD candidate in Computer Science at UFES and his research interests are related to Multimedia Systems, Video Synchronization and Mobile Applications. He is also Lecturer at the University of Vila Velha (UVV) since.

Capítulo 4

Ecossistemas de Software no Desenvolvimento de Plataformas para Web, Redes Sociais e Multimídia

Rodrigo Santos[1] e Davi Viana[2]

[1] Universidade Federal do Estado do Rio de Janeiro (UNIRIO) – rps@uniriotec.br

[2] Universidade Federal do Maranhão (UFMA) – davi.viana@ufma.br

Abstract

A software ecosystem (SECO) is a set of actors and artifacts that exchange resources and information based on a common technological platform, in which external players are also included. This context has affected decisions on the management and development of those platforms in several domains, especially regarding the architecture, governance and collaboration models. As such, it is important to integrate mechanisms and tools to support the exchange of information, resources and artifacts, as well as to ensure an effective communication and interaction among organizations, developers and users. This chapter presents how the SECO reality changes the development of web, social networks and multimedia platforms.

Resumo

Um ecossistema de software (ECOS) é um conjunto de atores e artefatos, internos e externos a uma organização ou comunidade, que trocam recursos e informações centrados em uma plataforma tecnológica comum. Este contexto tem afetado decisões de gerenciamento e desenvolvimento de tais plataformas, notadamente sobre modelos de arquitetura, de governança e de colaboração, nos mais variados domínios de aplicação. É necessário integrar mecanismos e ferramentas para apoiar a troca de informações, recursos e artefatos, bem como assegurar a comunicação e interação dos desenvolvedores e usuários. O objetivo deste capítulo é apresentar como os ECOSs afetam o desenvolvimento de plataformas para web, redes sociais e multimídia.

4.1. Introdução

O desenvolvimento de plataformas para web, redes sociais e multimídia avançam cada vez mais em pesquisa teórica e aplicada, apresentando desafios que estão além das questões técnicas. A construção desses sistemas como produtos de software tem evoluído para o desenvolvimento de múltiplos produtos, derivados de uma plataforma baseada em uma arquitetura comum e integrados com outros sistemas por meio de redes de atores e artefatos [Manikas 2016]. Conforme apontado na primeira tese de doutorado do Brasil no assunto [Santos 2016], esse conjunto de elementos forma um ecossistema de software (ECOS) e requer a integração de mecanismos e ferramentas para apoiar a troca de informações, recursos e artefatos, bem como para assegurar a comunicação e interação dos desenvolvedores e usuários [Bosch 2012]. Nesse sentido, as redes sociais que se formam no ECOS, fortemente dependentes de web e multimídia, precisam ser analisadas a fim de auxiliar uma melhor concepção, gerenciamento e evolução de um ECOS [Hanssen e Dybå 2012]. Além disso, decisões de gestão e o monitoramento de tais plataformas dependem notadamente dos modelos de arquitetura, de governança e de colaboração, e do seu uso nos mais variados domínios de aplicação.

Pelas razões apresentadas, este capítulo tem grande relevância para a comunidade de Multimídia e Web por propiciar o contato com um tópico atual e de caráter prático na indústria (ECOS). Esta combinação pode auxiliar no tratamento de desafios gerados pela utilização de ECOS, como lidar com questões econômicas e sociais em conjunto com as questões técnicas, conforme [Boehm 2006] [Bosch 2009] [Seichter et al. 2010] [Santos et al. 2012] [Manikas 2016] [Santos 2016]. Dessa forma, o desenvolvimento de plataformas para web, redes sociais e multimídia é afetado pelos ECOS [Santos e Oliveira 2013], e conhecer este assunto é de grande importância para a comunidade de Web e Multimídia. Ademais, desdobramentos de utilização de ECOS podem contribuir para entender, analisar e resolver tais questões pelo fato de que este conceito vem sendo extensamente utilizado no mercado. Nos casos reais da indústria de software, por exemplo, existem ECOSs de dispositivos móveis, como Android e iOS; ECOS de gestão de conteúdo e comunidades na web, como o Moodle; ECOS de redes sociais, como Facebook; e ECOS de desenvolvimento colaborativo, como o Portal do Software Público Brasileiro (SPB).

O objetivo deste capítulo é apresentar como os ECOSs afetam o desenvolvimento de plataformas para web, redes sociais e multimídia. Para isso, os principais conceitos e estratégias de ECOS serão discutidos, seguidos pela discussão de aspectos que influenciam a análise do desenvolvimento de plataformas para web, redes sociais e multimídia. Alguns desafios enfrentados pela academia e pela indústria de software serão apontados nas considerações finais visando explorar alguns pontos importantes de se considerar em tais plataformas, como por exemplo a questão de transparência [Santos et al. 2016]. Para isso, este capítulo está organizado da seguinte forma: a Seção 4.2 apresenta a trajetória, conceitos principais e algumas estratégias para ECOS; na Seção 4.3, são discutidos como a Aprendizagem Organizacional e Gestão do Conhecimento podem influenciar o desenvolvimento de plataformas de ECOS para web, redes sociais e multimídia; por fim, a Seção 4.4 conclui o capítulo com algumas considerações finais pautadas em desafios para ECOS centrados nas plataformas supracitadas, bem como possíveis desdobramentos para trabalhos futuros.

4.2. Conceitos e Estratégias de Ecossistemas de Software

Nesta seção, o objetivo é apresentar alguns conceitos básicos de ECOS, isto é, a sua definição, elementos-chave e atores envolvidos no processo de desenvolvimento, a fim de compreender a interação entre atores e artefatos [Seichter et al. 2010]. A meta é mostrar a importância de cuidar das redes formadas em torno do desenvolvimento de plataformas e seus produtos e serviços em um ECOS, bem com a complexidade de lidar com isso devido a diversos fatores, entre eles o ciclo de vida social das redes criadas durante o desenvolvimento de plataformas para web, redes sociais e multimídia.

4.2.1. Dos Sistemas Monolíticos aos Ecossistemas

Software pode ser definido como um conjunto de instruções que proveem características, funções e desempenho desejados, quando tais instruções são executadas corretamente, além de estruturas de dados que permitem aos programas manipularem informações e o conhecimento que descreve a operação e uso dos programas em formato impresso e virtual [Messerschmitt e Szyperski 2003] [IEEE, 2004]. Entretanto, o software é um elemento de um sistema lógico, mais do que físico, o que o diferencia significativamente do hardware. Conforme Pressman (2010), software é desenvolvido ou "engenheirado", envolve pessoas e requer uma gestão de projetos efetiva para tratar os custos relacionados a isso. Além disso, o software não se desgasta como o hardware, mas se deteriora, sendo que, em sua maioria, continua sendo construído de maneira customizada e baseado em componentes e bibliotecas existentes.

Historicamente, esses apontamentos remontam a conferência do comitê de ciência da Organização do Tratado do Atlântico Norte em 1968 na qual, de acordo com Brooks (1995), apenas o artigo apresentado por Doug McIlroy tinha relação direta com a engenharia. A contribuição de McIlroy se referia à ideia de decompor sistemas de software em unidades reutilizáveis, chamadas "componentes", e ao uso de biblioteca de programas utilitários para efetuar tarefas como classificação de itens ou ainda rotinas matemáticas. Essa noção de reutilização é antiga e originária da busca por soluções consistentes para problemas, que pudessem ser aplicadas a novos problemas e cuja repetitividade as tornaria aceitas, generalizadas e padronizadas, como acontece na Matemática e na Física [Werner 1992]. Apesar de trazer vários benefícios como o aumento da produtividade e a redução do esforço de desenvolvimento, os maiores obstáculos são de natureza gerencial, organizacional, econômica e técnica, ou seja, reutilizar demanda um ciclo contínuo de aprendizagem.

Nos últimos anos, com a globalização do desenvolvimento de software e os novos modelos de negócio e de colaboração, as plataformas para redes sociais, web e multimídia começaram a sofrer com as pressões de abertura para agregar novos envolvidos, externos à organização [Santos et al. 2014b]. Isso é chamado de inovação aberta [Chesbrough 2003] e tem motivado a transição da visão tradicional da engenharia de software para os ecossistemas, a fim de investigar as questões sociais e econômicas intrínsecas ao ciclo de vida das plataformas supracitadas. A partir desse discurso, pode-se definir uma trajetória do desenvolvimento de software monolítico para as plataformas dos ecossistemas, ilustrada pelas "4 gerações" apresentadas na Figura 4.1. Existem duas direções: (a) *Da ES tradicional para os ECOSs*, que considera a evolução histórica e o amadurecimento da engenharia de software nas últimas três décadas, em busca de

maximizar os benefícios prometidos ao se reutilizar software; e (b) *Dos ECOSs para a ES sustentável e diversificada*, que considera as diferentes dimensões que precisam ser levadas em consideração no desenvolvimento de plataformas para redes sociais, web e multimídia, visando a sua sustentabilidade[1] e diversidade[2].

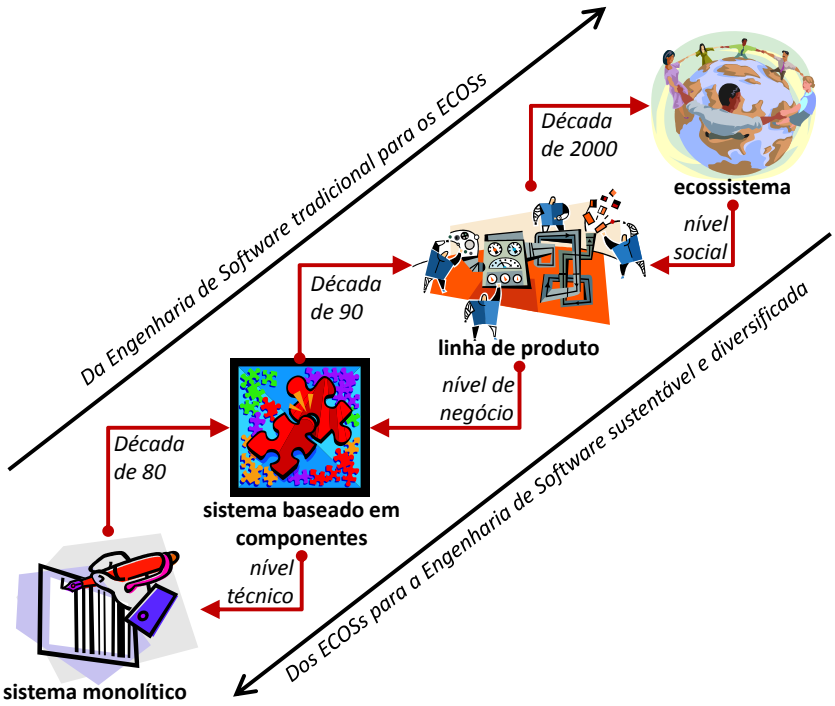


Figura 4.1. Evolução dos sistemas monolíticos para os ecossistemas. Adaptado de Santos (2016)

Em um primeiro momento, a principal reação contra a abordagem "codifica-e-remenda" da década de 1960 foi estabelecer processos em que a codificação era mais bem organizada, baseada na programação estruturada, que era precedida pela modelagem e engenharia de requisitos. Entre as décadas de 1970 e 1980, princípios de modularidade foram explorados e fortalecidos, e.g., coesão e acoplamento, além de técnicas de encapsulamento e de tipos abstratos de dados, o que contribuiu para a construção dos ***sistemas monolíticos***: sistemas formados por conjuntos de rotinas que têm permissão para interagir livremente entre si. Melhorias foram buscadas em direção à produtividade e à escalabilidade. Por exemplo, organizações que despendiam 60% do seu esforço em testes descobriram que 70% destas atividades representavam retrabalho, podendo ser evitado ao estruturar as fases anteriores do ciclo de vida [Boehm 2006].

Por sua vez, os avanços da computação para melhorar a produtividade da década de 1980, tais como sistemas especialistas, linguagens de alto nível, orientação a objetos, estações de trabalho poderosas e programação visual, propiciaram a reutilização de software. Sistemas operacionais mais poderosos, sistemas de gerenciamento de banco de dados, *frameworks* para interface gráfica, *middleware* distribuído e automação de

[1]Sustentabilidade é a capacidade do ECOS sobreviver às diferentes perdas [Dhungana et al. 2010].

[2]Diversidade é a capacidade de prover diversas oportunidades no ECOS [Dhungana et al. 2010].

escritório permitiram superar a reutilização simples, do tipo "copia-e-cola" [Boehm 2006]. As linguagens de programação orientadas a objeto impulsionaram a prática de modularização. Segundo Biffl et al. (2006), no final dos anos 1980, abordagens de arquitetura de software viabilizaram a reutilização efetiva de componentes de aplicação através de *frameworks* e de linguagens de quarta geração específicas de domínio. Os engenheiros de software passaram a observar a experiência de outros setores da indústria na utilização de ***sistemas baseados em componentes***.

Na década de 1990, com a expansão da Internet e da web, um mercado competitivo de software emergiu, de modo que o software passou a ser tratado como elemento de diferenciação no mundo dos negócios. Por exemplo, o caso da Hewlett Packard (HP) foi relatado por Poulin (1996) e mostrou que investimentos em arquitetura de software e componentes reutilizáveis aumentou o tempo de desenvolvimento dos três primeiros produtos (1986-1987), mas reduziu o tempo, ano a ano, até 1992, quando o estudo de caso foi finalizado. No final da alguns elementos ganharam destaque na indústria: software crítico, gerência de dependências, serviços web, manutenção de sistemas legados, componentes de prateleira (COTS), desenvolmento *open source* e usabilidade. Para as organizações, estava claro o esforço para atingir os benefícios de reutilização, o que levou ao conceito de ***linha de produto de software*** no final dos anos 1990, inspirado pelo sucesso do conceito de linha de produção em organizações como a Toshiba [Boehm 2006]. Nesta abordagem, a organização organiza uma base de ativos reutilizáveis, isto é, artefatos de software, utilizada para desenvolver e compor sistemas.

As organizações que arcaram com os custos de adoção de uma linha de produtos alcançaram sucesso com a reutilização [Bosch e Bosch-Sijtsema 2010]. No entanto, McGregor (2010) afirma que a inovação passou a forçar a abertura das fronteiras da organização, de modo que modelos de negócio, processos, produtos, serviços, recursos, informações e artefatos sejam alvos de uma comunidade em torno de uma plataforma ou tecnologia de software central da organização, podendo gerar vantagem competitiva [Goldman e Gabriel 2005]. Isso significa que a plataforma pode estar disponível para atores externos à organização e, uma vez que ela decide abrir a sua fronteira, pode ocorrer a transição de linhas de produto para ***ecossistemas*** [Bosch 2009]. Para obter sucesso, as organizações precisam descobrir maneiras de explorar os desenvolvedores externos potenciais para torná-los parte de uma comunidade, pela combinação de licenças e recompensas bem definidas, o que requer compreender e adaptar modelos de negócio, além de modelar e analisar as redes socio-técnicas [Campbell e Ahmed 2010].

Bosch (2009) afirma que duas razões motivam uma organização a utilizar a abordagem de ECOS: (1) ela pode perceber que a quantidade de funcionalidades que ela precisa desenvolver para satisfazer as necessidades dos clientes e usuários é muito maior do que ela pode construir, considerando o investimento em P&D e o tempo disponível para um retorno adequado; e (2) a tendência à customização em massa direciona a demanda por investimento em P&D para aplicações de software de sucesso, de modo que estender o produto (e a plataforma) com o apoio de atores externos à organização parece ser um mecanismo efetivo. Entretanto, vale a pena destacar que a abordagem de ECOS ainda está em maturação na academia e na indústria: o primeiro registro do uso deste termo na área de engenharia de software remonta o ano de 2003 [Barbosa et al. 2013]. Apesar disso, as motivações continuam vigentes [Messerschmitt e Szyperski 2003] [Jansen et al. 2013]: o software é ubíquo, faz o ambiente interativo, é importante,

é sobre pessoas, pode ser melhor, faz parte de uma indústria em constante mudança, é uma criação social, é sofisticado e complexo, e pode ser controlado.

4.2.2. Conceitos Básicos de ECOS

Na última década, o estudo de ECOS tem se tornado um campo de pesquisa ativo e definido pelo aparecimento de novos modelos de arquitetura de software, de colaboração de desenvolvedores e de negócio abertos [Hanssen e Dyba 2012]. Isso contribui para a criação de comunidades interconectadas em redes complexas de atores e de organizações, interessadas em uma tecnologia de software central (ou plataforma), que oferece oportunidades para geração de valor por diversos atores [Hanssen 2012]. Hanssen e Dyba (2012) afirmam que a abordagem de ECOS representa um salto radical em como a engenharia de software está sendo feita, de maneira que o desenvolvimento está se tornando um processo aberto em um complexo ambiente distribuído. Barbosa et al. (2013) sumarizaram a maior parte dos termos e acrônimos utilizados para se referir a ECOS (Tabela 4.1), uma vez que diferentes grupos de pesquisa e da indústria têm investigado o tema de modo independente.

Tabela 4.1. Termos e acrônimos utilizados para ECOS na literatura. Fonte: [Barbosa et al. 2013]

TERMINOLOGIA **(em ordem decrescente de importância)**
Software Ecosystems
Digital (Business) Ecosystem* / *D(B)E
SECO (Software Ecosystem)
Mobile Learning Ecosystems (MLES)* / *Mobile Ecosystem
FOSS Ecosystem* / *Open Ecosystem

Alguns exemplos de ecossistemas são o ECOS MySQL/PHP, o ECOS Eclipse, o ECOS Microsoft e o ECOS iPhone. Estes exemplos podem ser usados para estabelecer as características típicas de um ecossistema. Por exemplo, um ECOS *pode estar contido* em outro ECOS, como o ECOS Microsoft CRM (*Customer Relationship Management*), contido no ECOS Microsoft; pode-se dizer que ECOS iPhone com sua AppStore é um *ECOS fechado e controlado*, ao passo que o ECOS MySQL/PHP e o ECOS Eclipse são *ECOS abertos*, desde que as organizações tenham acesso ao código fonte e às bases de conhecimento, visando maior flexibilidade no desenvolvimento. Considerando as pesquisas básicas feitas por Santos e Werner (2010; 2011), Barbosa e Alves (2011), Hanssen e Dyba (2012) e Manikas e Hansen (2013), observou-se a existência de diversas definições para ECOS na literatura, embora a de Jansen et al. (2009) e a de Bosch (2009) sejam as mais citadas pelos estudos publicados desde 2003:

> *Um ECOS consiste em um conjunto de atores funcionando como uma unidade, que interage com um mercado distribuído entre software e serviços, juntamente com as relações entre estas entidades. Estas relações são frequentemente apoiadas por uma plataforma tecnológica ou por um mercado comum e realizadas pela troca de informação, recursos e artefatos.* [Jansen et al. 2009]

Um ECOS consiste em um conjunto de soluções de software que possibilitam, apoiam e automatizam as atividades e transações realizadas por atores em um ecossistema social ou de negócios, e por organizações que provêm estas soluções. O foco está nos aspectos organizacionais do ECOS e nas relações entre as organizações enquanto entidades de software e serviços. [Bosch 2009]

Apesar das numerosas definições existentes e de pelo menos 40 publicações não citar nenhuma delas conforme resultados de uma ampla revisão da literatura no assunto [Manikas e Hansen 2013], dois conceitos são ditos fundamentais em ECOS: (1) um interesse comum em uma *tecnologia de software central*, isto é, sistemas, serviços ou plataforma de software; e (ii) uma *rede de organizações*, isto é, algum tipo de relacionamento, seja simbiótico, ou de evolução conjunta, comercial ou técnica. Nesse contexto, os diferentes relacionamentos que a organização pode ter com tecnologia de software central, os três papéis principais são apontados na Tabela 4.2. Outros papéis compreendem organizações de padronização, revendedores e operadores de sistemas [Jansen et al. 2009]. Cada um dos atores pode se beneficiar de suas participações no ECOS, e o espectro de relacionamentos simbióticos depende de seus papéis e atividades.

De acordo com Manikas e Hansen (2013), pelo menos cinco relacionamentos podem ser identificados em ECOS: dois atores podem (1) ter benefícios mútuos (*mutualismo*); (2) estar em competição direta (*competição/antagonismo*); (3) não ser afetados (*neutralismo*); ou (4) um não ser afetado enquanto o outro é beneficiado (*amensalismo*) ou (5) prejudicado (*parasitismo*) pelo relacionamento. Em resumo, existem três elementos chave em um ECOS [Santos e Werner 2010]: (i) *software*, como plataforma tecnológica comum, tecnologia de software central, soluções de software, plataforma de software ou linha de produto; (ii) *transações*, como um senso que inclui modelos de lucro ou recompensa, mas também possíveis benefícios além dos financeiros, e.g., algo que o ator obteria ao se envolver em comunidades *open source*; e (iii) *relacionamentos*, como conexões entre atores com base nos elementos (i) e (ii).

Tabela 4.2. Principais papéis em um ECOS.
Adaptado de Hanssen (2012)

PAPEL	DESCRIÇÃO
Organização Chave ou Dono da Plataforma (*Keystone*)	Uma organização, ou um pequeno grupo, que de alguma forma conduz o desenvolvimento da tecnologia de software central.
Usuários Finais ou Clientes (*End-users*)	Papel chave para a tecnologia de software central, pois representa quem precisa dela para realizar seu negócio, seja de qual tipo for.
Organizações Externas ou Desenvolvedores Externos (*Third-parties*)	Utilizam a tecnologia de software central como base para produzir soluções ou serviços relacionados.

Por fim, Manikas e Hansen (2013) listam 43 ecossistemas que têm sido explorados, dos quais 30 são discutidos cada um em uma publicação, 12 estão presentes em mais de uma publicação e um não foi nomeado, embora seja mencionado em três publicações. O ECOS Eclipse/Eclipse Foundation é o mais estudado, com sete publicações, seguido pelos ECOSs: GNOME e Open Design Alliance (quatro); Software Público Brasileiro e Linux/Linux Kernel (três); Android, GX Software, Evince, FOSS, FreeBSD, iPhone/iPad App Store e SAP (dois); Apache Web Server,

Artop, Brasero, CAS Software AG, CSoft, CubicEyes, Debian, Google Chrome, Google, Gurux, Firefox, HIS GmbH, HISinOne, Mac App Store, Microsoft, Nokia Siemens Networks, Nautilus, Pharo, Ruby, S. Chand Edutech, SOOPS BV, Squeak, Symbian, TFN 200, UniImprove, Unity, US Department of Defense, WattDepot, WinMob e World of Worcraft (um). Somente dois ECOSs são fechados (GX Software e SAP), o que pode refletir o desafio de conseguir dados desse tipo de ECOS, ao contrário dos ECOS abertos, que possuem repositórios públicos disponíveis, conforme discutido em estudos como [Santana e Werner 2013].

4.3. Análise de Aspectos que Influenciam o Desenvolvimento de Plataformas para Web, Redes Sociais e Multimídia

Analisar os diversos aspectos que influenciam o nascimento, desenvolvimento, amadurecimento e eventual "morte" de plataformas para web, redes sociais e multimídia mais complexas aparece como uma preocupação em ECOS [Santos e Werner 2010]. Mais especificamente, na medida em que as organizações "abrem" as suas estruturas de negócios para outras contribuírem, mais ecossistemas começam a se formar, o que requer uma análise mais cuidadosa dos diversos aspectos que influenciam o desenvolvimento dessas plataformas. Nesse sentido, um desafio está no gerenciamento da *diversidade de organizações e relações de um ECOS* criado em torno de programadores, fornecedores, parceiros e clientes/usuários [Lima et al. 2014]. Isso pode dificultar a identificação e compreensão dos papéis que cada ator possui no ECOS, bem como os impactos de suas ações e decisões no desempenho do ECOS como um todo. Além disso, é necessário *analisar e tratar o conhecimento sobre a plataforma e os produtos e serviços nela desenvolvidos*. Por fim, é necessário fazer com que *os diversos atores aprendam os conhecimentos para que melhorem sua produtividade*.

Diversas abordagens foram propostas com o objetivo de lidar com esses aspectos de influência durante o ciclo de vida de tais plataformas em ECOS. Essas abordagens podem auxiliar as organizações no tratamento do conhecimento organizacional. Primeiramente, Campbell e Ahmed (2010) afirmam que o conceito de ECOS tem raiz nas teorias do desenvolvimento de plataformas similares (web e multimídia) e das redes sociais e pode ser um caminho para a transição, evolução e inovação na engenharia de software. Para explorar isso, os autores apresentam uma visão de ECOS em três dimensões, a fim de entender as funções assumidas por cada um de seus pilares:

- *arquitetura*: envolve a plataforma em que o ECOS está inserido, assim como questões da arquitetura de software, linha de produto e processos de software;
- *negócio*: envolve o conhecimento sobre o mercado, decisões que os atores devem tomar sobre modelos de negócio, definição do portfólio de produtos do ECOS, e estratégias de licenças e de vendas;
- *social*: define a forma como a rede de atores irá se relacionar dentro do ECOS para atingir seus objetivos e fomentar o crescimento do ECOS por meio de uma proposição de valor onde todos possam obter ganhos.

Em outra perspectiva, Jansen et al. (2009) discutiu que o ciclo comercial de um ECOS pode ser analisado em quatro fases, uma vez que a dimensão "negócio" envolve uma transação entre atores: (1) o estabelecimento de um relacionamento de mercado

com uma organização chave e focada; (2) o surgimento de uma rede preliminar; (3) a diminuição do poder da organização chave e o estímulo das comunidades; e (4) a manutenção de uma comunidade de criação do ECOS, onde não existem organizações chave e o poder é distribuído. Nesse contexto, as dimensões "arquitetura" e "negócio" de Campbell e Ahmed (2010) são normalmente alvos de controle pela organização chave, mas a dimensão "social" é dinâmica. Ou seja, esta dimensão é a "pedra angular" para o sucesso do ECOS ao afetar as fases do seu ciclo de vida, i.e., nascimento, desenvolvimento, amadurecimento e morte/transformação. Avançando nesta discussão, Santos et al. (2013) apresentam uma generalização das três dimensões para tratar transações não comerciais:

- *técnica*: trata o entendimento da abertura das plataformas e do envolvimento de atores externos. É dirigida por três fatores: (a) *engenharia da plataforma*: definição, modelagem e construção do sistema; (b) *arquitetura*: estrutura dos componentes do sistema, relacionamentos e princípios e diretrizes para sua evolução; e (c) *gerência de requisitos*: funcionalidades específicas e comuns;
- *transacional*: trata o entendimento da colaboração ou competitividade, das expectativas dos envolvidos e do impacto na produtividade do seu ambiente. É dirigida por três fatores: (a) *visão do ambiente*: realizar planos e estar aberto a ações situadas; (b) *inovação*: usar a criatividade e atuar conforme um segmento; e (c) *planejamento estratégico*: seguir objetivos e ousar oportunidades;
- *social*: trata o entendimento da abertura da organização como um todo, das comunidades criadas ao redor da plataforma e do desenvolvimento colaborativo. É dirigida por três fatores: (a) *utilidade*: compensações e recompensas esperadas e percebidas pelos membros da rede, financeiras ou não; (b) *promoção*: reconhecimento implícito ou explícito de capacidades e habilidades do colaborador; e (c) *ganho de conhecimento*: armazenamento e manutenção de experiências e oportunidades para os interessados se envolverem com novas tecnologias e ferramentas no ambiente colaborativo.

Caso a transação não seja comercial, o efeito das plataformas na formação da comunidade com base em sites de redes sociais on-line (e.g., Facebook) pode ser analisado em quatro fases, por meio do ciclo social do ECOS (Figura 4.2), estendido por Santos et al. (2014a) a partir de [Russ 2007] e [Jansen et al. 2009]:

- *iniciação*: criação de uma página para a plataforma em um site de rede social, a nível organizacional, visando estabelecer um relacionamento de mercado com os demais atores. Isso agrega valor ao relacionamento com fornecedores, clientes, distribuidores e organizações externas e amplifica a capacidade de *marketing*, suporte, pesquisa de mercado e de tecnologia com seus parceiros. Distribuidores podem criar suas páginas e agregar valor à sua rede de relacionamento, incluindo relacionamento direto com a organização chave. O fator mais importante para atrair usuários da rede social já existente é o valor do conteúdo dessa página;
- *propagação*: o contágio social é iniciado pela adesão de novos atores e artefatos, originando o ECOS. Surge uma rede preliminar de atores com interesses em comum (perfil), ou páginas de artefatos produzidos por fornecedores ou organizações externas. São criados conteúdos e comentários e são formados

grupos ou comunidades, com a consequente diminuição do poder da organização chave. Cresce o estímulo da participação, atração de novos membros e formação de comunidades, quando a massa crítica é alcançada;

- *amplificação*: estabelecimento de uma estrutura auto organizável e manutenção de uma comunidade engajada e calcada na rede de atores e de artefatos de um ECOS. Nenhuma organização é dominante pelo fato do poder se tornar distribuído, com a vantagem de estar no site de rede social e utilizar funcionalidades e recursos de comunicação, colaboração, recomendação e de *marketing*, que ajudam na divulgação e interação;
- *terminação*: normalmente, um serviço de rede social on-line termina devido à saturação ou substituição por um novo serviço, ou ainda porque surgem mercados e tendências que provocam a "evaporação" dos integrantes desta rede e, logo, da comunidade. Pode ser decorrente de um término ou quebra de sustentabilidade e/ou diversidade de um ECOS.

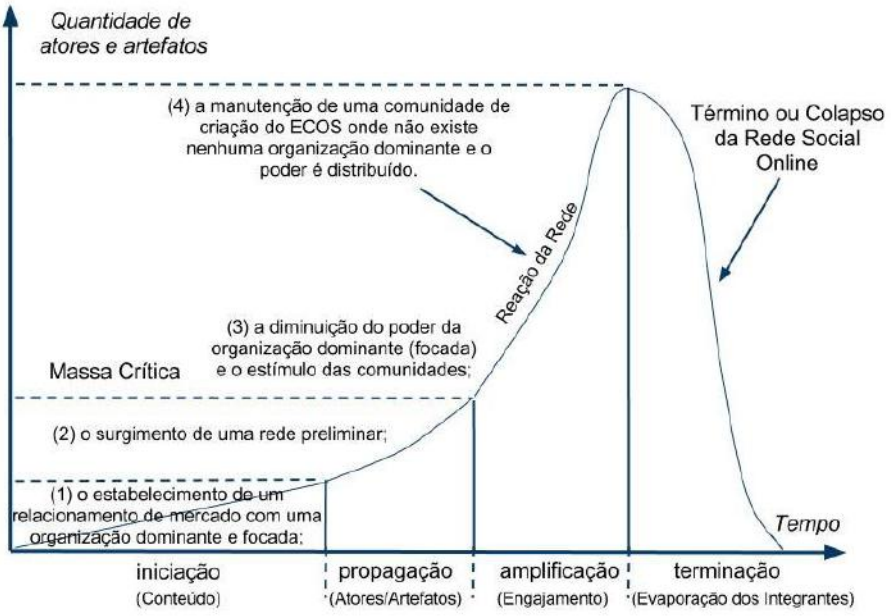


Figura 4.2. Ciclo de vida social de um ECOS. Adaptado de Santos et al. (2014a)

As redes sociais em um ECOS agem de forma semelhante à propaganda "boca a boca", onde os consumidores disseminam a sua percepção sobre produtos e serviços, com um poder multiplicador muito maior [Seichter et al. 2010]. O principal atrativo está em atingir a 800 milhões de pessoas dentro de uma única plataforma [Santos et al. 2013]. Fica claro que as organizações que não acompanham a rapidez da inovação de seus mercados viabilizada pelas redes em um ECOS podem não conseguir atravessar as barreiras cada vez mais altas [Chesbrough 2003]. Neste contexto, a abertura da plataforma pode estimular a inovação e representa um mecanismo estratégico para a criação da novidade (produto ou serviço), pois atende a dois requisitos: (i) a novidade gera ganho social, por permitir que mais bens e serviços sejam entregues à sociedade; (2) gera retorno para o inovador (financeiro e outros), por ser consumida por ela.

Tal ponto de vista motivou Santos et al. (2013) a expandirem o ciclo de vida social para classificar as organizações sob o ponto de vista da adoção de um modelo de inovação em ECOS, quando um novo ator pode ser incorporado à fase "propagação": o **intermediário** (*technology broker*). Os intermediários têm o papel de: (i) facilitar o encontro entre oferta e demanda; (ii) agregar os atores certos a fim de fomentar a inovação; e (iii) apoiar e coordenar o trabalho colaborativo entre os diferentes níveis [Schilling 2008]. Esse papel pode ser analisado a partir da Figura 4.3 e da Figura 4.4, em que a primeira ilustra o processo da inovação fechada em um sistema tradicional (cada organização concebe suas inovações internamente) e a segunda ilustra o processo de inovação aberta (existem intermediários para agir como mediadores ou avaliadores dos diferentes ecossistemas e gerar relatórios ou realizar ações).

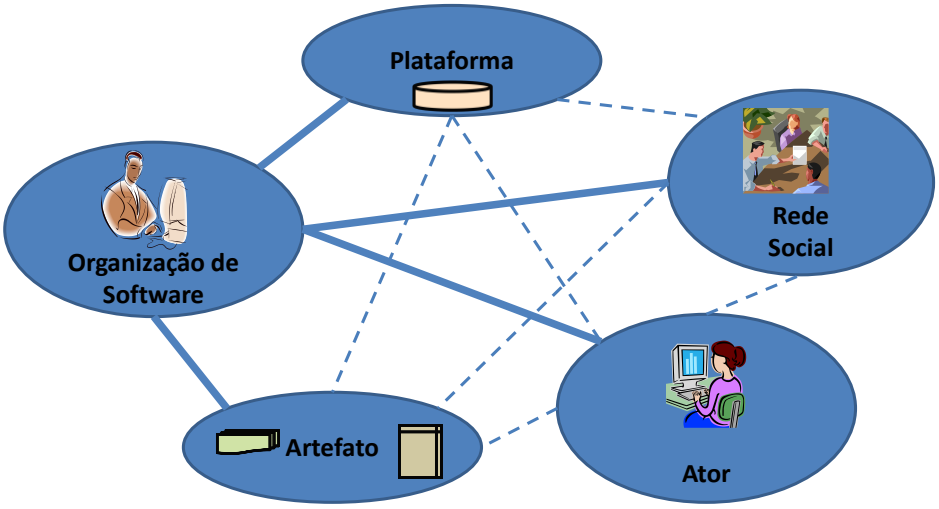


Figura 4.3. Fontes de inovação fechada em um sistema tradicional, onde as linhas pontilhadas são relações que não estão estabelecidas claramente, dado que a organização de software centraliza as relações. Adaptado de Santos et al. (2014a)

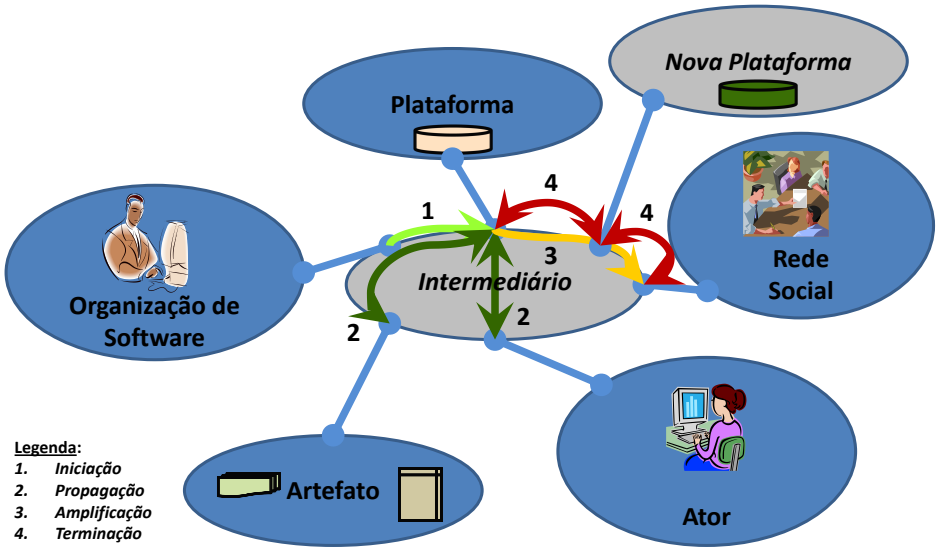


Figura 4.4. Fontes de inovação aberta em um ECOS, incorporando aspectos do ciclo de vida social. Adaptado de Santos et al. (2014a)

Por fim, Manikas e Hansen (2013) classificam os ECOS quanto a sua missão em *proprietários* e/ou *abertos*. No ECOS proprietário estrito, o código fonte e os demais artefatos produzidos são protegidos por serem produtos que geram retorno, e os novos atores provavelmente devem se certificar para participar do ecossistema. No ECOS aberto tradicional, os atores não necessariamente participam do ecossistema para obter rendimentos financeiros por suas atividades, sendo mais fácil colaborar, pois tipicamente não requer qualquer atestado de novos atores. Os autores ainda observam que ecossistemas abertos tratam mais as questões de natureza técnica e social, ao passo que aqueles proprietários incluem questões estratégicas e de negócio. Uma das razões é que ecossistemas abertos permitem o uso de técnicas de mineração e de processamento sobre código, *logs* de *commits* etc., embora careça de um modelo de negócio para participação de atores. Por outro lado, ECOSs proprietários conseguem prontamente informação sobre as estratégias do ecossistema e seu posicionamento no mercado, mas o acesso ao código fonte e *commits* de desenvolvedores é mais difícil.

Entre outros aspectos que influenciam o desenvolvimento de plataformas para redes sociais, web e multimídia, parte das abordagens está voltada para o uso direto de práticas de engenharia de software, e outra trabalha com a adaptação dessas práticas para o contexto de ECOS, ambas com foco na plataforma. Questões de arquitetura de software são de suma importância, devendo-se tratar gerenciamento, regras de negócio e restrições, integração e existência de múltiplas funcionalidades [Cataldo e Herbsleb 2010]. Segurança e confiabilidade consistem em instrumentos para equilibrar modularidade e flexibilidade na arquitetura da plataforma a fim de assegurar interoperabilidade e manutenção de interfaces [Bosch 2010]. Inconsistências causadas por evoluções devem ser verificadas e validadas nas dependências de componentes, o que ressalta a importância de mecanismos de coordenação que afetam abordagens centradas em processo [Bosch e Bosch-Sijtsema 2010]. A constante evolução demanda processos adaptáveis, para que o desenvolvimento seja centrado em integração, implantação independente e grupos de liberações de versões da plataforma.

A análise e projeto arquitetural passam a incluir princípios como identificar metas de negócio e descrever requisitos significativos, táticas e avaliação arquitetural. O processo de elicitação de requisitos se torna um desafio relevante em ECOS, pois os novos atores se juntam aos tradicionalmente envolvidos no desenvolvimento, alguns deles numerosos e distantes do gerenciamento central do ECOS [Fricker 2009]. Cadeias de valor e cadeias de resultados passam a apoiar a engenharia de requisitos [Yu e Deng 2011], além da modelagem, análise e visualização de redes sociais, técnicas e socio-técnicas para identificar redes de influência e de interoperabilidade [Santos et al. 2013]. Bosch (2009) acrescenta mais duas questões: uso de processos ágeis e a composição de produtos e serviços em ecossistemas. Nesse contexto, a gestão do conhecimento e a aprendizagem no ECOS merecem ser analisados e são discutidos nas subseções a seguir.

4.3.1. Teoria de Criação do Conhecimento e Processo de Aprendizagem

Considerando o desenvolvimento de plataformas para redes sociais, web e multimídia, a gestão do conhecimento e a aprendizagem dos atores são cruciais, sobretudo em ECOS. Nesse sentido, o conhecimento sempre se origina nas pessoas sendo criado através da interação entre o Conhecimento Tácito e Explícito [Nonaka e Takeuchi 1995]. Essa interação deu origem ao modelo SECI (do inglês, *Socialization, Externalization,*

Combination e Internalization). Cada processo deste modelo representa uma conversão de conhecimento. No processo de socialização, o Conhecimento Tácito é compartilhado diretamente com outra pessoa. Esse conhecimento socializado pode não se tornar Explícito. No processo de externalização, o Conhecimento Tácito é convertido em Conhecimento Explícito. Neste caso, a organização possui a possibilidade de compartilhar o conhecimento com toda a organização. Já na etapa de combinação ocorre a combinação de componentes isolados do Conhecimento Explícito para a geração de um novo Conhecimento Explícito. A Figura 4.5 apresenta componentes do modelo SECI e o processo de aprendizagem criado através do modelo.

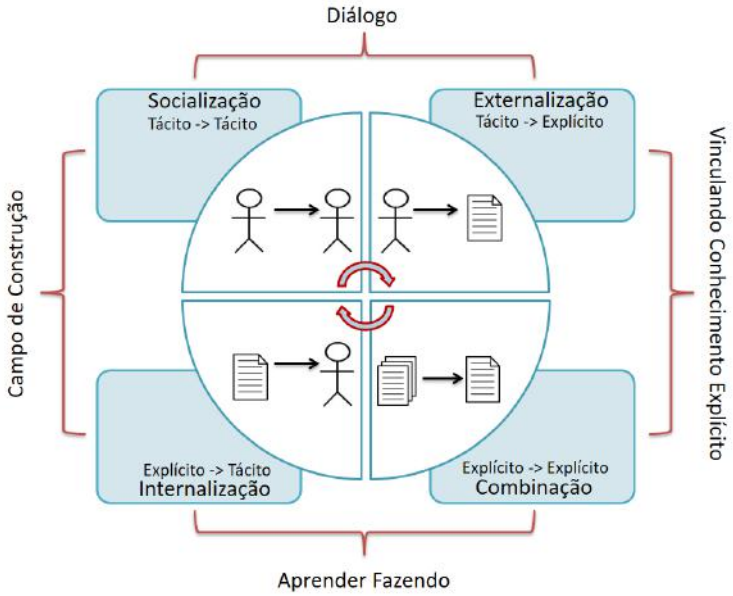


Figura 4.5. Modelo SECI e Processo de Aprendizagem. Adaptado de Nonaka e Takeuchi (1995)

Um processo de aprendizagem pode ser gerado a partir da realização dos quatro processos do modelo SECI. Ao executar os quatro processos, tem-se a espiral do conhecimento [Schneider 2009]. Inicialmente, é realizada uma socialização onde ocorre a troca de Conhecimento Tácito através de diálogo. Em seguida, esse conhecimento é externalizado por meio da escrita, além de ser combinado com outros conhecimentos através de vínculos entre os componentes do Conhecimento Explícito. O conhecimento externalizado é aplicado em alguma atividade de trabalho específica, neste momento ocorre o "aprender fazendo", pois a pessoa irá utilizar o Conhecimento Explícito de forma prática. Por fim, verifica-se que há um campo de construção do conhecimento, onde a base de Conhecimentos Tácitos é enriquecida. Neste momento, o processo de aprendizagem se inicia novamente [Nonaka e Takeuchi 1995]. Para que as organizações utilizem o conhecimento de forma eficaz, é necessário gerenciar tanto o Conhecimento Tácito quanto o Explícito [Hansen et al. 2000] [Choi e Lee 2003]. A utilização desses dois conhecimentos é importante para acumular o conhecimento corporativo e explorar novos potenciais [Choi e Lee 2003].

As plataformas para redes sociais, web e multimídia, bem como os produtos e serviços construídos a partir delas podem fazer uso deste processo, uma vez que é necessário transmitir o conhecimento gerado durante o desenvolvimento e fazer com

que novos colaboradores passem a cooperar com o ECOS criado. Por exemplo, Lopes et al. (2009) apresentam uma proposta de plataforma baseadas em *Service Oriented Architecture* (SOA), que permite que as funcionalidades sejam disponibilizadas e utilizadas como um conjunto de serviços descritos através de suas interfaces. O conceito de externalização do conhecimento pode auxiliar na disponibilização de conhecimentos sobre como utilizar a plataforma. De modo semelhante, o conceito de internalização pode apoiar na aprendizagem de utilização da plataforma.

4.3.2. Aprendizagem pelo Circuito Simples e Circuito Duplo de Argyris e Schön

Nesta abordagem, a aprendizagem é vista como um processo cíclico. Para Argyris e Schön (1978), a aprendizagem ocorre através da mudança de comportamento ao serem verificados os resultados de estratégias de ação e realização de correções. Essas estratégias de ação são atividades regidas por variáveis definidas a partir de objetivos e restrições. Quando as ações ou atividades são executadas, determinadas consequências ou resultados são identificados [Schneider 2009]. Existem duas formas de verificar a ocorrência da aprendizagem, são elas: circuito simples e circuito duplo. No circuito simples (do inglês, *single-loop learning*), a aprendizagem ocorre através da comparação entre as consequências das estratégias de ação executadas com as consequências desejáveis de uma estratégia de ação definida. Se houver um desvio entre as consequências planejadas e as reais, busca-se uma nova estratégia de ação dentro do conjunto de variáveis [Argyris e Schön 1978]. É importante frisar que esse circuito promove aprendizagem incremental [Souza-Silva e Davel 2007] e que pode ocorrer na mente das pessoas, ou através de uma simulação de duas alternativas [Schneider 2009].

Schneider (2009) apresenta um exemplo da aprendizagem de circuito simples aplicado ao desenvolvimento de sistemas. Quando um programador está trabalhando com uma linguagem que ele não está tão familiarizado e ocorre um erro de compilação (consequências reais), o programador irá alterar seu código fonte (estratégia de ação) para alcançar o seu resultado satisfatório (consequências desejáveis). Sendo que o resultado satisfatório é definido a partir do objetivo de obter um software com dado conjunto de requisitos (variáveis). Desta forma, entende-se que o programador aprendeu a não ocasionar mais o erro de compilação. Além disso, as consequências devem fornecer, sempre que possível, um *feedback*. Este *feedback* é um importante componente para que a aprendizagem ocorra de acordo com essa teoria, pois é através dele que é possível realizar melhorias no comportamento pessoal, como acontece em ECOS. A Figura 4.6 apresenta o processo da aprendizagem em circuito simples.

Figura 4.6. Aprendizagem de Circuito Simples (*single-loop learning*). Adaptado de Schneider (2009)

Em algum momento, pode ser necessária a mudança de alguma variável, objetivo e/ou restrições devido a questionamentos surgidos a partir do *feedback*. Quando isso ocorre, tem-se a possibilidade do ciclo duplo de aprendizagem (do inglês, *double-loop learning*). O processo é semelhante ao circuito simples, além das mudanças nas estratégias de ações (circuito simples), também é necessário verificar alterações nas variáveis (circuito duplo), conforme discute Argyris e Schön (1978). Neste caso, o conhecimento é adquirido neste nível mais alto, isto é, durante o circuito duplo. Levando em consideração o desenvolvimento de software, tem-se que o programador implementa um módulo. Se houver um comportamento diferente do esperado (consequências desejáveis), o programador pode alterar o módulo (circuito simples) ou pode começar a questionar as consequências desejáveis de modo que se analisem as possíveis alterações nos requisitos para satisfazer as consequências desejáveis (circuito duplo). Neste momento do circuito duplo, pode ocorrer à aprendizagem a partir do resultado atingido com a implementação do módulo [Schneider 2009]. A Figura 4.7 representa graficamente a ocorrência do ciclo duplo.

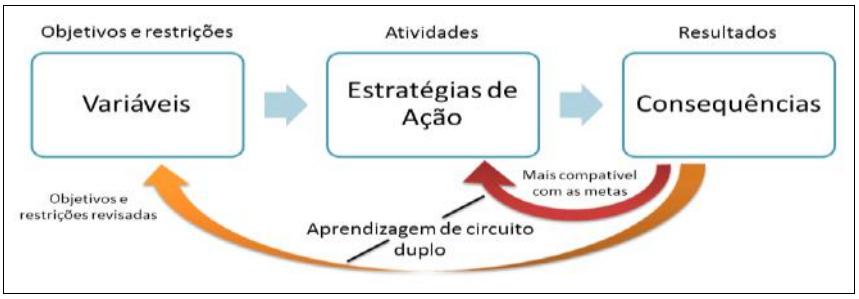


Figura 4.7. Aprendizagem de Circuito Duplo (*double-loop learning*). Adaptado de Schneider (2009)

4.3.3. Comunidades de Prática

Comunidades de prática são grupos de pessoas que compartilham os mesmos interesses e mesmos problemas em um tópico. Essas comunidades possuem como propósito principal a criação de conhecimento, possuindo conexões fracas, autogerenciamento e informalidades [Voss e Schafer 2003]. Segundo Miranda (2004), existem diversas definições para comunidades de prática. Contudo, todas apresentam que as comunidades de prática são estruturas sociais que têm facilidade em lidar com o conhecimento.

As pessoas que participam de uma comunidade de prática não trabalham necessariamente juntas todos os dias, mas elas se conhecem e se encontram devido ao valor encontrado nas suas interações [Wenger et al. 2002]. Essas interações incluem troca de informações, insights e conselhos, além de auxiliar na resolução de problemas, discutir situações e explorar ideias. As comunidades podem criar conhecimento explícito por meio de ferramentas, padrões, manuais e outros documentos, ou podem apenas desenvolver um entendimento tácito sobre o que está se compartilhando [Wenger et al. 2002]. Isso também acontece no contexto de ECOS.

A comunidade de prática é constituída de duas grandes partes: (a) *participação na comunidade*, interagindo e envolvendo-se nas iniciativas sociais; e (b) *concretização*, em que ocorre o processo de criar artefatos a partir dos resultados pela comunidade de prática, como documentos e através da interação face a face [Mestad et al. 2007].

Existem alguns indicadores que descrevem quando uma comunidade de prática foi criada [Mestad et al. 2007]. Esses indicadores são definidos por Wenger et al. (2002) e são características que podem ser necessárias para o sucesso de implementação deste tipo de comunidade. A Tabela 4.3 apresenta alguns exemplos desses indicadores.

Tabela 4.3. Indicadores de Wenger que descrevem a existência de uma comunidade de prática. Adaptado de Wenger et al. (2002)

1. Relacionamentos mutualmente sustentados.
2. Formas comuns de engajar-se para fazer as coisas juntos.
3. Fluxo rápido de informações e propagação de inovação.
4. Ausência de considerações introdutórias, as conversações e interações nas comunidades devem ser apenas a continuação de um processo em andamento.
5. Definição rápida de um problema a ser discutido.
6. Saber o que os outros sabem, o que eles podem fazer, e como eles podem contribuir para uma iniciativa.
7. Capacidade de avaliar a adequação das ações e produtos.
8. Ferramentas específicas, representações e outros artefatos.
9. Conhecimento local, histórias compartilhadas.
10. Jargões e atalhos para comunicação.
11. Certos estilos reconhecidos por todos os membros.
12. Discurso compartilhado refletindo certa perspectiva.

A aprendizagem nas comunidades de prática é estimulada através dos relacionamentos estabelecidos, como em ECOS. No contexto do desenvolvimento de plataformas para redes sociais, web e multimídia, as comunidades de prática são utilizadas para apoiar o acesso a especialistas e recursos de informação. Além disso, podem ser aplicadas para contribuir com o compartilhamento de conhecimento sobre tecnologias e, assim, contribuir para a aprendizagem e melhorar a produtividade durante o desenvolvimento de plataformas web e redes sociais. Esta teoria de comunidades de prática também serve como base para o desenvolvimento de sistemas de *e-discourses* [Voss e Schafer 2003]. *E-discourse* é um processo de comunicação argumentativa orientada a objetivos que é apoiado por uma ferramenta. Através dessa comunicação, as pessoas podem contribuir com ideias e perspectivas para soluções de problemas.

4.3.4. Teoria das Redes de Aprendizagem

A forma com que a aprendizagem é organizada no desenvolvimento de software é descrita pela teoria das redes de aprendizagem [van der Krogt 1998]. As pessoas aprendem em todas as organizações, seja uma empresa com estrutura hierárquica ou caótica, como acontece no contexto de ECOS. A rede de aprendizagem consiste em várias atividades de aprendizagem organizadas pelos membros da empresa, possuindo três grandes componentes [Škerlavaj e Dimovski 2006]:

- **Processo de Aprendizagem:** desenvolvimento de políticas e programas de aprendizagem e forma de execução dos programas;
- **Estruturas de aprendizagem:** estrutura do conteúdo para aprendizagem, estrutura organizacional e o clima da empresa para aprendizagem;
- **Atores:** colaboradores em todos os níveis.

Os atores da aprendizagem interagem uns com os outros para organizar as atividades do processo de aprendizagem. Com o passar do tempo, certos padrões estabelecidos inicialmente são desenvolvidos. Esses padrões formam as estruturas de aprendizagem [Poell et al. 2000]. A forma da rede de aprendizagem depende da dinâmica dos atores envolvidos e das características de trabalho da organização [van der Krogt 1998] [Poell et al. 2000]. Existem quatro tipos teóricos de redes de aprendizagem: liberal, vertical, horizontal e externa. No tipo *liberal*, normalmente a estrutura da aprendizagem é orientada a indivíduos, ou seja, cada colaborador pode ter sua forma de aprender. No tipo *vertical*, os sistemas de aprendizagem são auxiliados por planos de política e programas de aprendizagem elaborados. Por outro lado, no tipo *horizontal*, a aprendizagem é orientada a organização como um todo. Por fim, no tipo *externa*, a aprendizagem é estruturada de acordo com os profissionais externos (consultores). Para analisar os tipos de rede de aprendizagem pode ser utilizada a análise de redes sociais [Wasserman e Faust 1994]. A Tabela 4.4 apresenta um maior detalhamento dos quatro tipos teóricos de redes de aprendizagem.

Tabela 4.4. Quatro tipos teóricos de redes de aprendizagem. Adaptados de Škerlavaj e Dimovski (2006) e van der Krogt (1998)

	Tipos de Redes de Aprendizagem			
	Liberal	Vertical	Horizontal	Externa
Processo de Aprendizagem	Atividades simples	Linearmente planejado	Orientados a organização	Coordenado externamente
Desenvolvimento de políticas de aprendizagem	Implícito	Planejado	Aprendizagem	Inspiradora
Desenvolvimento de programas de aprendizagem	Coletar	Projetar	Desenvolver	Inovar
Execução dos programas de aprendizagem	Auto direcionada	Guiada	Aconselhada	Consultiva
Estrutura de Aprendizagem				
Estrutura do conteúdo para aprendizagem	Não estruturado (orientado a indivíduos)	Estruturado (orientado a atividades ou funções)	Aberta ou temática (orientada a organização ou problema)	Sistemática (orientadas a profissionais)
Estrutura Organizacional	Fracamente acoplada (contratos)	Centralizada (Formalizada)	Horizontal (igualitária)	Direcionada externamente (profissional)
Clima Organizacional para a aprendizagem	Liberal	Regulativa (regras e leis)	Integrativa (Aprendizagem e trabalho)	Inspiradora
Atores	Individuais	Oficiais, especialistas	Grupos	Atores externos

4.3.5. Ciclo de Aprendizagem de Kolb

Por fim, a teoria da aprendizagem experimental é descrita por Kolb (1984) como sendo o processo por onde o conhecimento é aprendido através da transformação da experiência, algo importante no contexto de ECOS. Neste ciclo, a "experiência concreta" dos indivíduos é base para a "observação reflexiva" que suporta reflexões na mente desses indivíduos. A observação reflexiva é necessária ao processo de criação de conclusões mais genéricas ou abstratas, gerando assim a "conceituação abstrata". Essas conclusões ou abstrações são aplicadas em mais situações, gerando assim a

"experimentação ativa". Durante a experimentação ativa, novas implicações podem ser testadas, criando novas experiências.

Contudo, Kolb descreve que nem sempre o indivíduo consegue passar por todas as fases do ciclo. Neste método, o ciclo de Kolb busca dar um sentido às experiências dos indivíduos, grupos e organizações por meio da experiência, reflexão, conclusão e experimentação. De forma paralela, os componentes do ciclo de Kolb auxiliaram no desenvolvimento dos quatro estilos de aprendizagem: divergência, assimilação, convergência e acomodação. Cada estilo representa uma combinação de dois componentes do ciclo de aprendizagem. A Figura 4.8 apresenta o ciclo de aprendizagem de Kolb juntamente com os estilos de aprendizagem. Esses estilos são encontrados nas comunidades de ECOS.

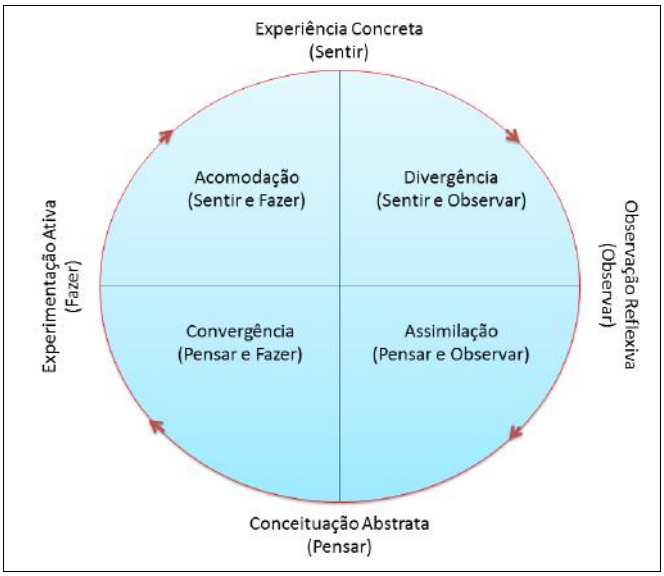


Figura 4.8. Ciclo de aprendizagem e estilos de aprendizagem. Adaptado de Kolb et al. (2001)

No estilo de aprendizagem *divergência*, as pessoas são hábeis em observar as coisas de diferentes perspectivas. Elas preferem observar a fazer, com o propósito de obter informações e usar a imaginação para resolver os problemas. Neste estilo, as pessoas preferem trabalhar em grupos e são emotivas. No estilo *assimilação*, existe uma abordagem concisa e lógica. Ideias e conceitos são mais relevantes que as pessoas. Estas pessoas requerem uma explicação boa e clara ao invés de prática. Pessoas que possuem este estilo preferem ler e explorar modelos teóricos. Já no estilo *convergência*, as pessoas podem resolver problemas e usarão sua aprendizagem para encontrar soluções para as questões práticas. As pessoas que possuem este estilo são mais abertas a experimentar novas ideias, simular e realizar atividades práticas. Por fim, no estilo *acomodação*, as pessoas preferem inicialmente praticar e tirar suas conclusões com base na experiência obtida. Também existe a preferência por trabalhar em equipes para completar atividades [Kolb et al. 2001]. Uma pessoa não possui somente um estilo de aprendizagem, pode ter tendências claras para um ou mais estilos de aprendizagem.

4.4. Considerações Finais

Dado o papel relevante do conceito "ecossistema" na engenharia de software do século XXI, este capítulo apresentou como ECOSs afetam o desenvolvimento de plataformas para web, redes sociais e multimídia. Neste contexto, é importante ressaltar que a inovação é constante em um ECOS e que diferentes tecnologias podem se unir para fortalecer as redes de produção de software que se formam em torno de uma plataforma e dos seus envolvidos. A integração de tecnologias e serviços através da web e mobile já é uma realidade, mas novas possibilidades vêm sendo exploradas. Por exemplo, o gerenciamento de informações evolui rumo à convergência digital para integrar diferentes mídias em um único ambiente, com a construção de novas funcionalidades como contribuições de desenvolvedores externos. Além disso, a colaboração dentro das redes de produção de software em ECOS pode ser ainda mais estimulada pela adoção de software social (e.g., Wikis, blogs etc.), que permite aos atores interagir e compartilhar conhecimento em uma dimensão social, realçando o potencial humano de criação, em vez de privilegiar a transmissão um para muitos. Nesse contexto, diversos aspectos de influência precisam ser analisados ao longo do desenvolvimento das plataformas supracitadas, conforme investigado e discutido na Seção 4.3.

Alguns desafios e fatores de sucesso foram apontados por Bosch (2009) e Jansen et al. (2009), resumidos na Tabela 4.5: (a) caracterizar e modelar ECOS, considerando diferentes perspectivas, e.g., no Brasil, a organização e evolução destes ecossistemas podem depender diretamente da atuação do Estado, ou seja, de questões políticas; (b) estabelecer os relacionamentos entre as redes de produção de software; (c) gerenciar a qualidade em ECOS; (d) planejar portfólios e linhas de produtos em ECOS; (e) prover métodos, técnicas e ferramentas para desenvolver arquiteturas de sistemas que atendam à extensibilidade, à portabilidade e à variabilidade; e (f) tratar implicações gerais em engenharia de software, incluindo mecanismos de comunicação, agilidade na concepção e desenvolvimento de soluções e composição de produtos e serviços.

Tabela 4.5. Desafios para ECOS.
Adaptado de Bosch (2009) e Jansen et al. (2009)

NÍVEIS	DESAFIOS	DESCRIÇÃO
Organizacional	#1: *Portfólio e plano da linha de produção*	Gerenciar configurações e reutilização para funcionalidades da plataforma.
	#2: *Gestão do conhecimento*	Aplicar mineração para extrair e visualizar *feedback* dos envolvidos.
	#3: *Arquitetura estensível, portável e variável*	Explorar interfaces, requisitos e comunicação sobre plataformas.
	#4: *Integração de sistemas na organização*	Desenvolver *frameworks* para gerência do produto de software e seus ciclos.
Rede de Produção	#1: *Estabelecimento das relações*	Identificar papéis e relacionamentos.
	#2: *Tempo de liberações*	Gerenciar versões e seus impactos.
	#3: *Gestão de qualidade*	Gerar diretrizes para certificação.
Ecossistema	#1: *Caractetização e modelagem de ECOS*	Identificar tamanho, vizinhança, padrões e papéis no ECOS.
	#2: *Políticas e estratégias entre ECOSs*	Determinar instrumentos para permitir orquestração em cada ECOS.
	#3: *Estratégia para vencer e lucrar na rede*	Integrar modelos de negócio ao ECOS.

Por fim, pode-se destacar dois pontos de vista, como mostra a Figura 4.9 [Barbosa et al. 2013]. Do ponto de vista do *gerenciamento*, tem-se: (i) como lidar com

ECOS do tipo *open source*, i.e., plataformas abertas e as redes constituídas; (ii) como implementar governança de ECOS, i.e., gerir estratégias de sobrevivência e agregação da comunidade em uma plataforma diante de competidores e colaboradores; e (iii) como realizar a análise de ECOS, i.e., identificar, coletar, relacionar e extrair informação a partir de redes constituídas em uma plataforma. Por outro lado, do ponto de vista da *engenharia*, tem-se: (i) como abrir a plataforma, i.e., permitir que atores externos à organização chave contribuam para a evolução da plataforma (*transparência*); (ii) como manter a qualidade da plataforma, i.e., verificar se a plataforma continua atendendo às necessidades da comunidade, bem como dos atores e relacionamentos que a compõem, considerando suas funcionalidades e componentes; e (iii) como modelar a arquitetura da plataforma, i.e., planejar o processo de construção e evolução da plataforma considerando questões técnicas, modelos transacionais e redes sociais.

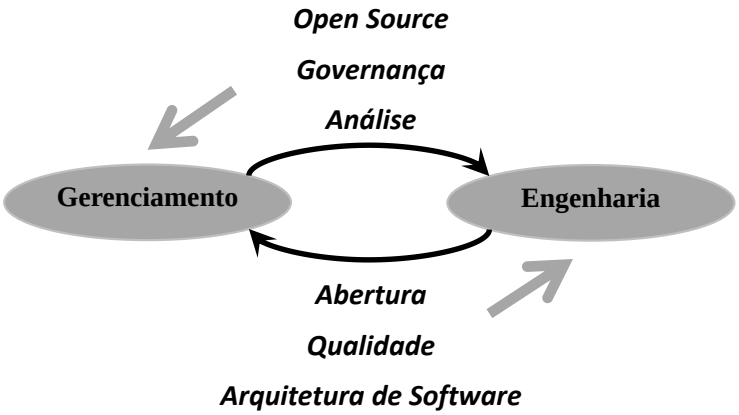


Figura 4.9. Desafios em ECOS.
Adaptado de Barbosa et al. (2013)

Agradecimentos

O primeiro autor gostaria de agradecer ao CNPq (Proc. No. PDJ 150539/20016-9) pelo apoio financeiro para realização deste trabalho.

Referências

Argyris, C., Schön, D.A. (1978) "Organizational Learning: A Theory of Action Perspective". Jossey-Bass.

Barbosa, O.A.L.P., Alves, C.F. (2011) "A Systematic Mapping Study on Software Ecosystems". In: Proceedings of the 3rd International Workshop on Software Ecosystems (IWSECO) – 2nd International Conference on Software Business (ICSOB), Brussels, Belgium, pp. 15-26, June.

Barbosa, O.A.L.P., Santos, R.P., Alves, C.F., Werner, C.M.L., Jansen, S. (2013) "A Systematic Mapping Study on Software Ecosystems from a Three-Dimensional Perspective". In: Jansen, S., Brinkkemper, S., Cusumano, M.A. (Org.), Software Ecosystems: Analyzing and Managing Business Networks in the Software Industry, pp. 59-81, Edward Elgar Publishing.

Biffl, S., Aurum, A., Boehm, B., Erdogmus, H., Grünbacher, P. (2006) "Value-Based Software Engineering". Springer-Verlag.

Boehm, B. (2006) "A View of 20th and 21st Century Software Engineering". In: Proceedings of the 28th International Conference on Software Engineering (ICSE), Shanghai, China, pp. 12-29, May.

Bosch, J. (2009) "From Software Product Lines to Software Ecosystem". In: Proceedings of the 13th International Software Product Line Conference (SPLC), San Francisco, USA, pp. 1-10, August.

Bosch, J. (2010) "Architecture Challenges for Software Ecosystems". In: Proceedings of the 4th European Conference on Software Architecture (ECSA) – 2nd International Workshop on Software Ecosystems (IWSECO), Copenhagen, Denmark, pp. 93-95, August.

Bosch, J. (2012) "Software Ecosystems: Implications for Strategy, Business Model and Architecture". In: Proceedings of the 34th International Conference on Software Engineering (ICSE), Tutorials, Zurich, Switzerland, June.

Bosch, J., Bosch-Sijtsema, P. (2010) "From Integration to Composition: On the Impact of Software Product Lines, Global Development and Ecosystems". The Journal of Systems and Software 83(1):67-76.

Brooks, F.P. (1995) "The Mythical Man-Month: Essays on Software Engineering". Addison-Wesley Professional, 2nd Ed.

Campbell, P.R.J., Ahmed, F. (2010) "A Three-dimensional View of Software Ecosystems". In: Proceedings of the 4th European Conference on Software Architecture (ECSA) – 2nd International Workshop on Software Ecosystems (IWSECO), Copenhagen, Denmark, pp. 81-84, August.

Cataldo, M., Herbsleb, J.D. (2010) "Architecting in Software Ecosystems: Interface Translucence as an Enabler for Scalable Collaboration". In: Proceedings of the 4th European Conference on Software Architecture (ECSA) – 2nd International Workshop on Software Ecosystems (IWSECO), Copenhagen, Denmark, pp. 65-72, August.

Chesbrough, H.W. (2003) "Open Innovation: The New Imperative for Creating and Profiting from Technology". Harvard Business School Publishing Corporation.

Choi, B., Lee, H. (2003) "An Empirical Investigation of KM Styles and their Effect on Corporate Performance". Information & Management 40(5):403-417.

Dhungana, D., Groher, I., Schludermann, E., Biffl, S. (2010) "Software Ecosystems vs. Natural Ecosystems: Learning from the Ingenious Mind of Nature". In: Proceedings of the 4th European Conference on Software Architecture (ECSA) – 2nd International Workshop on Software Ecosystems (IWSECO), Copenhagen, Denmark, pp. 96-102, August.

Fricker, S. (2009) "Specifications and Analysis of Requirements Negotiation Strategy in Software Ecosystems". In: Proceedings of the First International Workshop on Software Ecosystems (IWSECO) – 11th International Conference on Software Reuse (ICSR), Falls Church, USA, pp. 19-33, September.

Goldman, R., Gabriel, R.P. (2005) "Innovation Happens Elsewhere". Morgan Kaufmann.

Hansen, M., Nohria, N., Tierney, T. (2000) "What's your Strategy for Managing Knowledge". The Knowledge Management Yearbook, v. 2001, pp. 55-69.

Hanssen, G.K. (2012) "A Longitudinal Case Study of an Emerging Software Ecosystem: Implications for Practice and Theory". The Journal of Systems and Software 85(7):1455-1466.

Hanssen, G.K., Dybå, T. (2012) "Theoretical Foundations of Software Ecosystems". In: Proceedings of the 4th International Workshop on Software Ecosystems (IWSECO) – 3rd International Conference on Software Business (ICSOB), Cambridge, USA, pp. 6-17, June.

IEEE (2004) "Std 1517 - IEEE Standard for Information Technology - Software Life Cycle Processes - Reuse Processes". Institute of Electrical and Electronics Engineers.

Jansen, S., Cusumano, M., Brinkkemper, S. (2013) "Software Ecosystems: Analyzing and Managing Business Networks in the Software Industry". Edward Elgar Publishing.

Jansen, S., Finkelstein, A., Brinkkemper, S. (2009) "A Sense of Community: A Research Agenda for Software Ecosystems". In: Proceedings of the 31st International Conference on Software Engineering (ICSE), New and Emerging Research Track, Vancouver, Canada, pp. 187-190, May.

Kolb, D.A. (1984) "Experiential Learning: Experience as the Source of Learning and Development". Prentice Hall.

Kolb, D.A., Boyatzis, R.E., Mainemelis, C. (2001) "Experiential Learning Theory: Previous Research and New Directions". Perspectives on Thinking, Learning, and Cognitive Styles 1(2001):227-247.

Lima, T.M.P., Barbosa, G.S., Santos, R.P., Werner, C.M.L. (2014) "Uma Abordagem Socio-técnica para Apoiar Ecossistemas de Software". iSys: Revista Brasileira de Sistemas de Informação 7(3):19-37.

Lopes, F., Delicato, F., Batista, T., Pires, P. (2009) "Uma Plataforma Baseada em Serviços Web para Integração de Middleware de Contexto". In: Proceedings of the XV Brazilian Symposium on Multimedia and the Web (WEBMEDIA), Fortaleza, Brasil, Article No. 13, October.

Manikas, K. (2016) "Revisiting Software Ecosystems Research: A Longitudinal Literature Study". The Journal of Systems and Software 117(2016):84-103.

Manikas, K., Hansen, K.M. (2013) "Software Ecosystems – A Systematic Literature Review". The Journal of Systems and Software 86(5):1294-1306.

McGregor, J.D. (2010) "A Method for Analyzing Software Product Line Ecosystems", In: Proceedings of the 4th European Conference on Software Architecture (ECSA) – 2nd International Workshop on Software Ecosystems (IWSECO), Copenhagen, Denmark, pp. 73-80, August.

Messerschmitt, D.G., Szyperski, C. (2003) "Software Ecosystem: Understanding an Indispensable Technology and Industry". The MIT Press, 1st Ed.

Mestad, A., Myrdal, R., Dingsoyr, T., Myrdal, R. (2007) "Building a Learning Organization: Three Phases of Communities of Practice in a Software Consulting Company". In: Proceedings of the 40th Annual Hawaii International Conference on System Sciences (HICSS), Big Island, USA, p. 189a, January.

Miranda, R. (2004) "Um Ambiente de Apoio a Comunidades de Prática no Contexto da Estação TABA". Dissertação de Mestrado, COPPE/UFRJ, Rio de Janeiro, Brasil.

Nonaka, I., Takeuchi, H. (1995) "The Knowledge-Creating Company". Oxford University Press, 17th Ed.

Poell, R.F., Chivers, G.E., van der Krogt, F.J., Wildemeersch, D. (2000) "Learning-Network Theory: Organizing the Dynamic Relationships between Learning and Work". Management Learning 31(1):25-49.

Poulin, J.S. (1996) "Measuring Software Reuse: Principles, Practices, and Economic Models". Addison Wesley Professional.

Pressman, R.S. (2010) "Software Engineering – A Practitioner's Approach". McFraw-Hill, 7th Ed.

Russ, C. (2007) "Online Crowds – Extraordinary Mass Behavior on the Internet". In: Proceedings of the International Conference on New Media Technology, Graz, Austria, pp. 65-76, September.

Santana, F., Werner, C. (2013) "Towards the Analysis of Software Projects Dependencies: An Exploratory Visual Study of Software Ecosystems". In: Proceedings of the 5th International Workshop on Software Ecosystems (IWSECO) – 4th International Conference on Software Business (ICSOB), Potsdam, Germany, pp. 9-16, June.

Santos, R.P. (2016) "Managing and Monitoring Software Ecosystem do Support Demand and Solution Analysis". PhD Thesis, COPPE/UFRJ, Rio de Janeiro, Brasil.

Santos, R.P., Cappelli, C., Maciel, C., Leite, J.C.S.P. (2016) "Transparência em Ecossistemas de Software". In: Anais do VII Congresso Brasileiro de Software: Teoria e Prática (CBSoft) – X Workshop em Desenvolvimento Distribuído de Software, Ecossistemas de Software e Sistemas-de-Sistemas (WDES), Maringá, Brasil, pp. , 75-79, Setembro.

Santos, R.P., Esteves, M.G.P., Freitas, G., Souza, J.M. (2014a) "Using Social Networks to Support Software Ecosystems Comprehension and Evolution". Social Networking 3(2):108-118.

Santos, R.P., Oliveira, J. (2013) "Análise e Aplicações de Redes Sociais em Ecossistemas de Software". In: Anais do IX Simpósio Brasileiro de Sistemas de Informação (SBSI), Minicursos, João Pessoa, Brasil, v. 2, pp. 19-24, Maio.

Santos, R.P., Valença, G.A., Viana, D., Estácio, B.J.S., Fontão, A.L., Marczak, S., Werner, C.M.L., Alves, C.F., Conte, T.U., Prikladnicki, R. (2014b) "Qualidade em Ecossistemas de Software: Desafios e Oportunidades de Pesquisa". In: Anais do V

Congresso Brasileiro de Software: Teoria e Prática (CBSoft) – VIII Workshop em Desenvolvimento Distribuído de Software, Ecossistemas de Software e Sistemas-de-Sistemas (WDES), Maceió, Brasil, v. 2, pp. 41-44, Setembro.

Santos, R.P., Werner, C.M.L. (2010) "Revisiting the Concept of Components in Software Engineering from a Software Ecosystem Perspective". In: Proceedings of the 4th European Conference on Software Architecture Workshops (ECSA) – 2nd International Workshop on Software Ecosystems (IWSECO), Copenhagen, Denmark, pp. 135-142, August.

Santos, R.P., Werner, C.M.L. (2011) "A Proposal for Software Ecosystems Engineering". In: Proceedings of the 3rd International Workshop on Software Ecosystems (IWSECO) – 2nd International Conference on Software Business (ICSOB), Brussels, Belgium, pp. 40-51, June.

Santos, R.P., Werner, C.M.L., Alves, C.F., Pinto, M.J., Cukierman, H.L., Oliveira, F.M., Egler, T.T.C. (2013) "Ecossistemas de Software: Um Novo Espaço para a Construção de Redes e Territórios envolvendo Governo, Sociedade e a Web". In: Werner, C.M.L., Oliveira, F.J.G., Ribeiro, P.T. (Org.), Políticas Públicas: Interações e Urbanidades, pp. 337-366, Letra Capital.

Santos, R.P., Werner, C.M.L., Barbosa, O.A.L.P., Alves, C.F. (2012) "Software Ecosystems: Trends and Impacts on Software Engineering". In: Proceeding of the XXVI Brazilian Symposium on Software Engineering (SBES), Natal, Brazil, pp. 206-210, September.

Schilling, M.A. (2008) "Strategic Management of Technological Innovation". McGraw Hill, 2nd Ed.

Schneider, K. (2009) "Experience and Knowledge Management in Software Engineering". Springer.

Seichter, D., Dhungana, D., Pleuss, A., Hauptmann, B. (2010) "Knowledge Management in Software Ecosystems: Software Artifacts as First-class Citizens". In: Proceedings of the 4th European Conference on Software Architecture (ECSA) – 2nd International Workshop on Software Ecosystems (IWSECO), Copenhagen, Denmark, pp. 119-126, August.

Škerlavaj, M., Dimovski, V. (2006) "Social Network Approach to Organizational Learning". Journal of Applied Business Research 22(2):89-98.

Souza-Silva, J., Davel, E. (2007) "Da Ação à Colaboração Reflexiva em Comunidades de Prática". Revista de Administração de Empresas 47(2007):1-13.

van der Krogt, F.J. (1998) "Learning Network Theory: The Tension Between Learning Systems and Work Systems in Organizations". Human Resource Development Quarterly 9(2):157-177.

Voss, A., Schafer, A. (2003) "Discourse Knowledge Management in Communities of Practice". In: Proceedings of the 14th International Workshop on Database and Expert Systems Applications (DEXA), Prague, Czech Republic, pp. 782-786, September.

Wasserman, S., Faust, K. (1994) “Social Network Analysis: Methods and Applications”. Cambridge University Press.

Wenger, E., Mcdermott, R., Snyder, W.M. (2002) “Cultivating Communities of Practice”. Harvard Business Review Press.

Werner, C.M.L. (1992) “Reutilização de Software no Desenvolvimento de Software Científico”. Tese de Doutorado, COPPE/UFRJ, Rio de Janeiro, Brasil.

Yu, E., Deng, S. (2011) “Understanding Software Ecosystems: A Strategic Modeling Approach”. In: Proceedings of the 3rd International Workshop on Software Ecosystems (IWSECO) – 2rd International Conference on Software Business (ICSOB), Brussels, Belgium, pp. 65-76, June.

Biografia Resumida dos Autores

Rodrigo Santos é doutor e mestre em Engenharia de Sistemas e Computação pela COPPE – Universidade Federal do Rio de Janeiro (UFRJ) no tema Ecossistemas de Software, com período sanduíche na University College London (UCL). É professor do Departamento de Informática Aplicada da Universidade Federal do Estado do Rio de Janeiro (UNIRIO) e pesquisador na COPPE/UFRJ nos temas Ecossistemas de Software, Engenharia de Requisitos e Aquisição. Atua como professor em disciplinas de graduação e de pós-graduação em Computação desde 2007 e em pesquisas e desenvolvimento em Processo de Software e Gestão e Desenvolvimento de Ecossistemas no governo pela Fundação Coppetec. É avaliador de cursos superiores de Computação pelo Ministério da Educação (MEC). Organizou o CBSoft/WDES 2015 e está organizando o SBQS/WASHES 2016. Ministrou palestras, minicursos e tutoriais em eventos como CLEI (2016), IHC (2016), SBQS (2016, 2015, 2009), CBSOFT (2016, 2013, 2012), SBSI (2013, 2011, 2010), CIbSE (2012), SBIE (2010), ICTAC (2010) e ERIs (Campos/RJ - 2015, 2010, 2008; Itacoatiara/AM - 2013; Porto Velho/RO - 2012; Itajubá/MG - 2012; Manaus/AM - 2010, 2010; Teresina/PI - 2008; Lavras/MG - 2008, 2007). CV Lattes: http://lattes.cnpq.br/8613736894676086.

Davi Viana é doutor e mestre em Informática pelo Programa de Pós-Graduação em Informática da Universidade Federal do Amazonas (UFAM). Graduado em Ciência da Computação pela UFAM. Possui curso Técnico em Informática pela Fundação Nokia de Ensino. Atualmente é professor do Curso de Engenharia da Computação da Universidade Federal do Maranhão (UFMA). É ainda membro colaborador no Programa de Pós-Graduação em Ciência da Computação (PPGCC) da UFMA e colaborador no Laboratório de Sistemas Distribuídos Inteligentes (LSDi) da UFMA. Ministrou cursos sobre Melhoria de Processos de Software. Além disso, foi membro do comitê de organização do IHC 2013 e co-organizador geral do SBQS 2015. Está organizando o SBQS/WASHES 2016. Tem interesse de pesquisa nas áreas de qualidade de software, melhoria processo de software (MPS), implementação de programas de MPS com ênfase na adoção de modelos de maturidade, e engenharia de software experimental (ESE). CV Lattes: http://lattes.cnpq.br/9297257833779277.

Capítulo 5

ESPIM: um sistema para coleta de dados de usuários e intervenção programada a distância usando o método ESM e dispositivos móveis

Isabela Zaine[1], Kamila R. H. Rodrigues[1], Bruna C. R. da Cunha[1], Yuri N. Z. G. Magagnatto[1], Alex F. Orlando[1], Caio C. Viel[1], Olibário J. Machado Neto[1], André Carlomagno Rocha[1], Maria da Graça C. Pimentel[1]

[1]Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo (USP)

Abstract

The present course aims to introduce the Experience Sampling and Programmed Intervention Method (ESPIM), which combines selected procedures from the field of Psychology, such as Individualized Programmed Teaching and Experience Sampling Method (ESM), and the advantages promoted by Ubiquitous Computing. Our goal is to enhance ubiquitous data collection and interventions in naturalistic settings. The method is supported by a computational system - ESPIM - developed for the Web and mobile platforms. It allows the design of interventions based on both, explicit and pervasive data collection. The system supports multimedia data exchange between the stakeholders involved in planning data collection and/or intervention. We expect ESPIM to enable researchers and professionals from various fields, as health and education, to create and execute experiments or interventions carried out remotely. During the course attendants will learn about the fundamentals of the ESM and relevant results deriving from this method, data collection and interventions in naturalistic settings. Also, they will be presented to the latest version of the system ESPIM, its functionalities and computational development details. Attendants will have the opportunity to interact with the system, evaluating it in terms of usability and user experience, contributing to the improvement of the system.

Resumo

O presente curso tem o objetivo de apresentar o Método de Amostragem de Experiências e Intervenção Programada (ESPIM), que combina procedimentos

selecionados do campo da Psicologia, tais como Programação de Ensino Individualizada e Método de Amostragem de Experiências (ESM), e as vantagens proporcionadas pela Computação Ubíqua. O objetivo é ampliar o alcance da coleta de dados e intervenções a distância em ambientes naturais. O método é suportado por um sistema computacional - ESPIM - desenvolvidos para as plataformas Web e *mobile*. Ele permite que se programem intervenções baseadas em dados coletados de maneira explícita ou pervasiva. O sistema suporta a troca de dados entre as diferentes partes interessadas envolvidas no planejamento da coleta de dados e/ou intervenção. Espera-se que o ESPIM permita a pesquisadores e profissionais de diversas áreas de conhecimento, como educação e saúde, criar e executar experimentos ou intervenções conduzidos a distância. No decorrer do curso os participantes irão aprender sobre os fundamentos do ESM, bem como resultados relevantes decorrentes desse método, da coleta de dados e intervenções em ambiente natural. Será apresentada também a versão mais recente do ESPIM, suas funcionalidades e detalhes de desenvolvimento computacional. Os participantes terão a oportunidade de interagir com o sistema e avaliá-lo em termos de usabilidade e experiência de uso, contribuindo assim, para melhorias no mesmo.

5.1. Contextualização

A obtenção de informações acuradas acerca de comportamentos, sentimentos e atividades de pessoas em seu cotidiano e em ambiente natural tem sido um desafio para pesquisadores e profissionais de diversas áreas de conhecimento. O *Experience Sampling Method* - ESM [Csikszentmihaly *et al*. 1977], [Csikszentmihaly and Larson 1987] é um método da psicologia comumente utilizado para a coleta de dados acerca da experiência das pessoas em seu cotidiano e ambiente natural. Atualmente sistemas computacionais têm sido usados por pessoas de diferentes formas em seu cotidiano, de maneira inconsciente e automática, de modo ajudá-las a cumprir tarefas. Neste contexto, a Computação Ubíqua [Weiser 1991] é uma área que estuda o desenvolvimento de modelos que levam em consideração as atividades realizadas por indivíduos e os contextos em que essas atividades ocorrem [Yin *et al*. 2008], [Kapoor and Horvitz 2008]. As soluções computacionais, no contexto do método ESM, que fazem uso da computação ubíqua, permitem e facilitam o acesso a informações sobre comportamentos de interesse, bem como facilitam o planejamento de ações interventivas.

Do ponto de vista de relevância e aplicabilidade social, quanto mais pervasiva[1], contextualizada e individualizada for uma coleta de dados, potencialmente maior será o alcance de ações interventivas em diferentes domínios, uma vez que os dados previamente coletados podem ser usados para o planejamento de intervenções aplicáveis ao cotidiano e necessidades de diferentes indivíduos. Desta maneira, esta equipe procurou modelar o *Experience Sampling and Programmed Intervention Method* - ESPIM, um método que combina: a) técnicas de ESM; b) procedimentos selecionados de programação de ensino individualizada, e c) Computação Ubíqua; para aumentar a efetividade da coleta de dados. O método é apoiado por um sistema multiplataforma e, além de facilitar a coleta de dados a distância, permite o planejamento, autoria e

[1] O dicionário Cambridge associa a palavra pervasivo ao conceito de onipresente ao defini-la como presente ou visível em todas as partes de uma coisa ou lugar.

aplicação de intervenções também a distância por diferentes áreas de domínio, tais como a educação e cuidados de saúde física e mental. A seção seguinte apresenta em maior detalhes fundamentos teóricos importantes para a concepção do ESPIM.

5.2. Referencial Teórico

Esta seção apresenta o método *Experience Sampling Method* (ESM) para a coleta de dados a distância, em tempo real e em ambiente natural, bem como a sua aplicabilidade nas diversas áreas da saúde e da educação. Também serão discutidos conceitos da abordagem da psicologia da Análise do Comportamento para a programação de condições de ensino que podem ser estendidos ao planejamento e à execução de ações interventivas baseadas em dados previamente coletados.

5.2.1. Método ESM e Programação de Ensino

O *Experience Sampling Method* - ESM [Csikszentmihaly *et al*. 1977], [Csikszentmihaly and Larson 1987], [Hektner *et al.* 2007] é um método para coletar informações sobre a experiência das pessoas, no que se refere ao contexto e conteúdo, em ambiente natural. O foco nas experiências das pessoas pode ser descrito como interesse em ter acesso aos eventos ditos conscientes, tais como pensamentos, sentimentos e sensações. Esse método originou-se na abordagem psicológica da Fenomenologia Sistemática [Csikszentmihaly 2000] que combina esforços da fenomenologia pura em enfatizar a experiência como ela é percebida pelas pessoas com ferramentas de investigação de caráter empírico, tais como o uso de tecnologias, desenhos experimentais, análise estatística de dados, dentre outros [Hecktner *et al.* 2007]. O método prevê que participantes recebam lembretes de maneira aleatória ou programada de acordo com intervalos de tempo ou ocorrência de eventos de interesse para que façam um autorrelato sobre o que estão fazendo ou sentindo em um determinado momento.

O método ESM tornou-se mais popular a partir da segunda metade da década de 1970, com a publicação de um dos primeiros trabalhos de ESM por Csikszentmihalyi *et al.* [1977] sobre as atividades diárias e a qualidade das experiências relatadas por adolescentes. Nesse caso específico, aparelhos de *pager* eletrônico sinalizavam aleatoriamente durante o período de uma semana momentos para que os participantes fizessem autorrelatos. Desde então, o ESM tem sido usado para obter informações sobre os mais diversos temas, entre eles: comportamento de adolescentes, depressão, estresse e satisfação no trabalho, qualidade de vida, estado de humor antes e depois de assistir TV, monitoramento de pacientes em serviços de saúde, coleta de dados sobre experiência de aprendizagem em sistemas *online*, entre outros (para uma revisão abrangente, consultar Hecktner *et al*. [2007]).

Além de ser um método eficiente para coleta de dados sobre experiências cotidianas, os dados podem ser valiosos para o planejamento de programas de intervenção levando-se em consideração as particularidades da experiência, comportamento, ambiente físico e social de uma amostra de indivíduos ou de um indivíduo em particular, aumentando assim, o alcance e a efetividade da intervenção proposta.

Para ilustrar o escopo de trabalhos da literatura envolvendo ESM, a Tabela 5.1 descreve exemplos de estudos com temas e populações variadas que utilizam o método ESM com uso de algum tipo de recurso presente em dispositivos móveis.

Tabela 5.1. Exemplos de estudos que utilizam o método de ESM e dispositivos móveis.

Autores	n*	Tema da pesquisa	Recursos de captura
Floridou & Müllensiefen (2015)	40	Imaginário musical	Folheto e Celular para lembretes
Fuller-Tyszkiewicz et al. (2015)	144	Avaliação da auto-estima	Dispositivo móvel Palm com PalmOS
Hartley et al. (2014)	27	Estudo de emoções em pacientes diagnosticados com psicose	Dispositivo móvel Palm com PalmOS
Hofmann et al. (2014)	208	Estudo sobre autocontrole e dieta alimentar	Celular para lembretes e registro de respostas
Kackar et al. (2011)	331	Análise do comportamento de adolescentes durante o trabalho de casa	Celular p/ lembretes e questionários
Kramer et al. (2014)	102	Acompanhamento de tratamento de depressão	Dispositivo móvel psymate[1]
López et al. (2012)	**	Análise de atividades de diretores e chefes de escolas	Celular p/ lembretes e captura de log de atividades
Maes et al. (2015)	139	Imaginário musical	Dispositivo móvel psymate
McShane & Zirkel (2008)	12	Análise do comportamento de pessoas que sofrem de bulimia	Celular p/ lembretes e para responder emails
Mujagic et al. (2015)	26	Pacientes com Síndrome do cólon irritável	Dispositivo móvel psymate
Myllykangas et al. (2002)	78	Estudo da atividade de idosos por meio de desafios	PDAs para sinalização e feedback
Nielsen & Cleal (2011)	58	Análise da transformação de seguidores em lideres	PDAs p/ alertas e questionários
Rieh et al. (2010)	333	Avaliação da evolução de credibilidade das pessoas	Celular p/ lembretes e Interface Web para questionário
Swendeman et al. (2015a)	50	Automonitoramento de atividades de pacientes com HIV	Lembretes via email e celular
Wang et al. (2012)	28	U&G em uso de mídias sociais	Celular para lembretes e envio de SMS
Wang & Tchernev (2012)	32	Comportamento multitarefa e gratificações	Celular p/ lembretes e envio de email

* Número de participantes. ** Diretores e chefes de escolas 6 escolas municipais no Chile

A coleta de informações à respeito das experiências diárias no momento em que ocorrem é interessante pois permite que sejam traçados fluxos de eventos e relações entre eventos internos e externos. Combina, portanto, a validade ecológica das observações naturalísticas sobre comportamentos observáveis, com métodos não-intrusivos de autorrelato sobre experiências privadas. O fato dos autorrelatos serem feitos no momento de ocorrência de eventos de interesse também é interessante, pois diminui o viés da memória acerca dos acontecimentos (como esquecimento e distorções).

Outra abordagem da Psicologia, a Análise do Comportamento, também está comprometida com a qualidade da coleta de dados sobre situações de interesse. De maneira sucinta, a Análise do Comportamento é uma ciência do campo da psicologia que se baseia na filosofia do Behaviorismo Radical [Skinner 1953, 1974], que por sua vez tem o objetivo de estudar o comportamento dos organismos, entre eles, o humano. Parte-se da constatação de que existe ordem e regularidade no comportamento. Essa abordagem trata o homem sob uma perspectiva monista, ou seja, sem separação entre mente e comportamento. Os fenômenos ditos mentais por outras abordagens são entendidos como fenômenos comportamentais e dizem respeito às relações de um organismo com seu ambiente físico e social. Assim, as explicações para o comportamento são sempre buscadas nas interações funcionais entre o organismo e o ambiente. O objeto de estudo da Análise do Comportamento são os eventos que

ocorrem dentro de uma realidade natural, que podem ser acessados instrumental e empiricamente. Em decorrência, considera como objeto de estudo os comportamentos públicos observáveis e os comportamentos privados, tais como sentimentos e pensamentos.

A despeito de diferenças do ponto de vista de embasamento filosófico, as duas abordagens anteriormente apresentadas concordam em relação à importância da acuidade na coleta de dados. As informações coletadas em situações cotidianas podem ser usadas para estabelecimento de linha de base de ocorrência sobre comportamentos de interesse, definição de objetivos terapêuticos ou educacionais para a intervenção, além de usar determinadas variáveis de contexto, temporais ou autorrelatos, como gatilho, para iniciar uma intervenção planejada. Essa intervenção pode ser, por exemplo, engajamento em uma atividade educacional suplementar, administração de medicamentos, realização de exercícios de reabilitação fisioterapêutica, dentre outros. Em todos esses casos, considera-se que quanto mais informações forem obtidas sobre o cotidiano da população de interesse e sobre a percepção sobre suas experiências diárias, mais individualizado e potencialmente eficaz será o programa interventivo.

Tomando como base o exemplo de programas de intervenção educativos, psicólogos e pesquisadores da área de Análise do Comportamento têm proposto diretrizes para a programação de ensino dos mais variados comportamentos voltadas à populações com necessidades educativas especiais heterogêneas, desde comportamentos relativamente simples; como lavar as mãos e alimentar-se sozinho, a repertórios complexos; como comportamentos simbólicos de leitura e escrita. Já na década de 1950 Skinner [1958, 1961, 1972] apresenta uma proposta denominada Instrução Programada (PI – *Programmed Instruction*). Para esse autor, instruir significa instalar, alterar e eliminar comportamentos. Dessa maneira, o planejamento de ensino deve garantir ou maximizar as condições de aprendizagem de um comportamento, bem como de manutenção do que foi aprendido. Para tanto, o autor elabora algumas recomendações básicas. As principais são: 1) a programação do ensino deve ser iniciada pelo estabelecimento do repertório de entrada do aprendiz (o que ele já sabe); 2) definição de objetivos claros de ensino (o que ele deve aprender); 3) especificação detalhada do comportamento alvo do aprendiz (condições em que ele deverá ocorrer, consequências que seguem a emissão do comportamento); 4) situações aversivas e que favoreçam o erro devem ser minimizadas; 5) divisão do conteúdo total em pequenas unidades de ensino, com gradual progressão do nível de dificuldade; 6) apresentação de excelência do domínio do conteúdo atual para a progressão entre diferentes unidades de ensino; 7) fornecimento de *feedback* constante de desempenho; 8) garantia de alta densidade de reforços[2]; e 9) programação de contingências de reforçamento arbitrárias e naturais (passando das arbitrárias para as naturais)[3].

[2] "Reforço" é uma consequência que segue um comportamento que tem o efeito de aumentar a probabilidade de que o mesmo comportamento (ou outro similar) ocorra novamente no futuro em uma situação semelhante. No contexto leigo, ele pode ser entendido como "recompensa".

[3] Consequências arbitrárias são aquelas que são artificialmente estabelecidas, não havendo uma relação direta com as respostas que as produzem. Já as consequências naturais são aquelas decorrentes da própria atividade em questão, mantendo uma relação direta com as respostas que o produzem. Por exemplo, ler um livro porque ele será matéria de prova trata-se de uma consequência arbitrária, enquanto que ler um livro porque a leitura permite conhecer o conteúdo trata-se de uma consequência natural.

Nesse contexto, Skinner propõe máquinas de ensinar, que seriam dispositivos de apresentação de informações organizadas em uma sequência programada, com a principal característica do reforço imediato de respostas corretas.

De maneira semelhante, Keller, ao final da década de 1960, elabora o Sistema Personalizado de Ensino (PSI – *Personalized System of Instruction*), cujas principais características eram: 1) aprendizagem de acordo com o ritmo de cada aprendiz; 2) a exigência de domínio completo de uma unidade anterior antes de passar para a próxima fase; 3) o uso de demonstrações e palestras como meio de motivação; 4) a ênfase na palavra escrita mediando a comunicação entre professor e o aluno; e 5) o uso de monitores, que permitem a repetição de testes, *feedback* imediato ao aprendiz e tutoria acessível [Keller 1968]. Apesar de existir um foco da área em contextos educativos, a programação de ensino individualizada apresenta diretrizes importantes que podem ser aplicadas a diferentes cenários interventivos, uma vez que quase sempre intervenções envolvem a aprendizagem de algum tipo de repertório novo.

A princípio tanto as pesquisas envolvendo o ESM quanto as diferentes aplicações da área de programação de ensino individualizada eram desenvolvidas manualmente, com materiais impressos. No caso do ESM, comumente se utilizavam formulários ou diários em papel para realização de registros e dispositivos de *pagers* ou mensagens de texto (SMS) em aparelhos celulares para sinalizar o momento em que os autorrelatos deveriam ser feitos [Riddle and Arnold 2007]. Tal prática ainda é comum em estudos recentes, como no trabalho de Udachina *et al*. [2014].

Atualmente, dispositivos móveis como *smartphones* e *tablets* são amplamente populares, acessíveis à população e contam com uma série de recursos como câmeras de alta qualidade, microfone, ferramentas de gravação de voz e editores de texto, GPS para localização e sensores como acelerômetro; giroscópio; detecção de luz ambiente; detecção de proximidade, entre outros. Esse tipo de tecnologia permite grande flexibilização do método de amostragem de experiências [Baxter *et al*. 2015], podendo os usuários serem imediatamente sinalizados para realizar relatos de experiência de diversas formas (como por meio de texto, vídeo, fotografia) e envio de dados imediatamente ao pesquisador. Além disso, o uso de dispositivos móveis possibilita que dados sejam coletados sem a presença direta do experimentador, permitindo a coleta de informações por mais tempo sem a necessidade de deslocamento por parte do experimentador e dos participantes. Outra vantagem no uso de dispositivos móveis é a possibilidade de interferir minimamente nas atividades cotidianas dos participantes e de não provocar, ou pelo menos minimizar, a sensação de estarem sendo observados. Por fim, a adoção dos dispositivos móveis neste contexto aplicado tem potencial para aumentar a probabilidade de fornecimento de informações, bem como a adesão à determinadas ações de intervenções. Tais dispositivos também podem servir como meio para a realização total ou parcial da coleta de dados ou intervenções a distância. Por exemplo, indivíduos acompanhados por profissionais de saúde ou educação podem ser alertados durante o dia para administrar medicamentos, fazer exercícios de reabilitação fisioterapêutica, de estimulação cognitiva ou atividades escolares complementares. O próprio dispositivo pode servir como meio para o registro da atividade (por exemplo, texto, vídeo) e a interação desses usuários com o sistema pode ser imediatamente enviada aos profissionais responsáveis pelas intervenções para que esses façam as análises cabíveis. Novas ações do procedimento interventivo — baseadas nas informações enviadas pelos participantes — podem ser planejadas e implementadas rapidamente e de maneira individualizada.

Para exemplificar, a situação ilustrada a seguir em *storyboards* pela Figura 5.1 descreve um contexto de interação em que informações estão sendo coletadas em tempo real utilizando *smartphones*. O cenário apresentado é o de acompanhamento da rotina e de realização de atividades cotidianas por um indivíduo idoso. A interação ocorre em um ambiente familiar entre o idoso e seu neto. Nas cenas, o idoso recebe uma notificação em seu *smartphone* que desencadeia uma sequência de ações, entre elas: tirar uma *selfie* com o neto, divulgar a foto e responder à questões relacionados ao contexto em que a foto foi tirada.

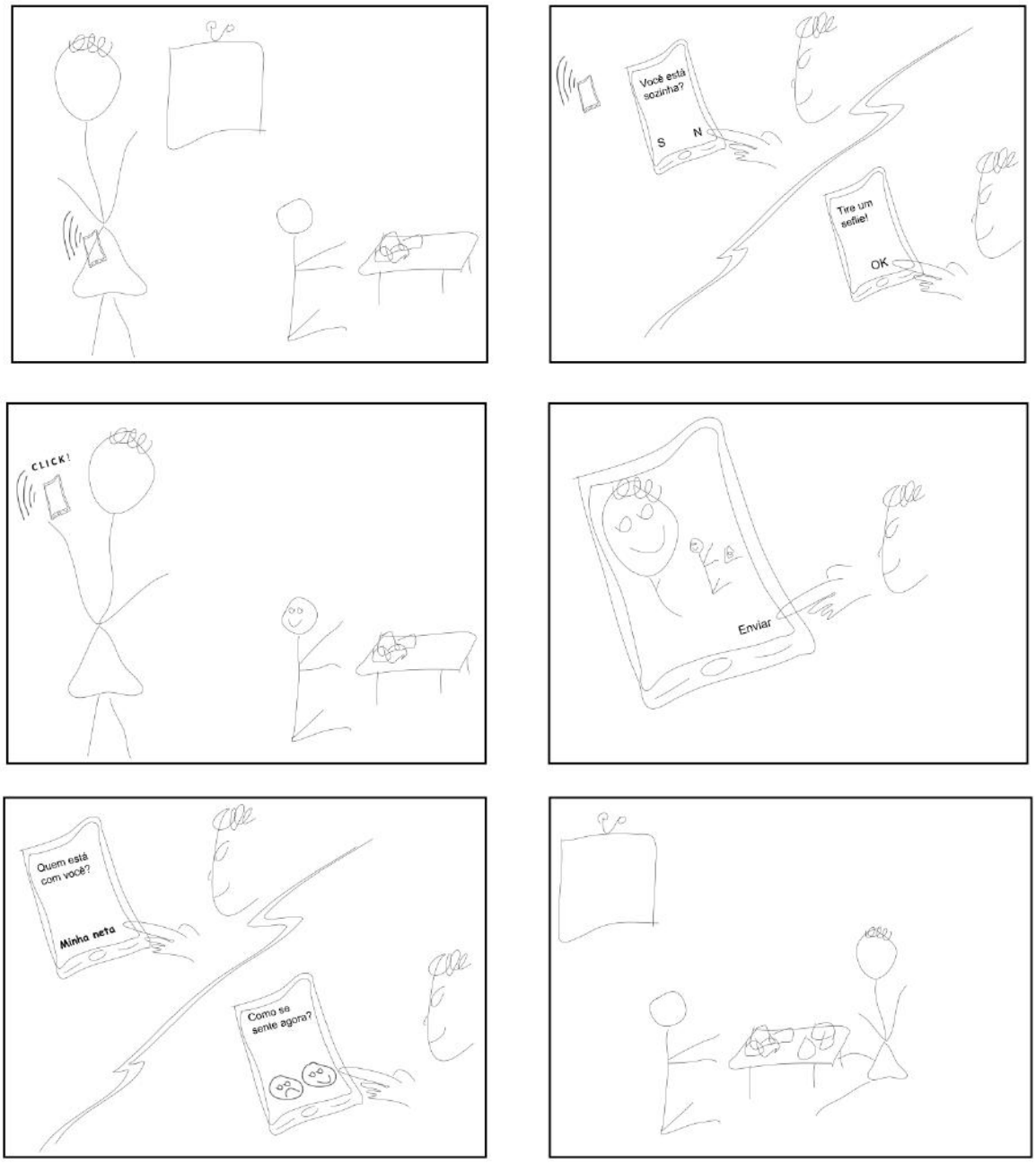


Figura 5.1. *Storyboards* que descrevem um cenário de coleta de dados em tempo real utilizando *smartphones*.

Outras vantagens do uso de Tecnologias da Informação e Comunicação (TICS) aplicáveis a dispositivos móveis para a realização de estudos envolvendo o método de ESM incluem: 1) controle preciso de tempo; 2) acompanhamento da participação efetiva dos usuários; 3) obtenção de dados auxiliares de avaliação de comportamentos e; 4) redução de erro humano. No primeiro caso, o experimentador pode programar o envio de alertas em intervalos de tempo fixo, variável ou restrito e especificar um intervalo de tempo para que o participante responda ao alerta. No segundo caso, pode-se ter um rastreio de participação efetiva e de alertas que foram "perdidos" ou não atendidos, dessa forma, o pesquisador pode acompanhar a motivação dos participantes em atenderem aos alertas, e modificar a forma ou frequência dos mesmos quando necessário. Com relação à obtenção de dados auxiliares, pode-se programar para que o sistema registre informações complementares ao conteúdo, como a latência de resposta entre diferentes questões. Finalmente, os dados coletados podem ser facilmente registrados em formatos compatíveis com softwares estatísticos automaticamente importados para análise, reduzindo a ocorrência de erros humanos [Barrett and Barrett 2001].

No caso das pesquisas em programação de ensino, os primeiros programas eram impressos em folhas de papel sequencialmente organizadas em pastas do tipo fichário e o registro de desempenho feito à mão, pelo preenchimento de protocolos de registro [De Rose *et al*. 1989]. Da mesma forma que o advento dos dispositivos móveis aumentou o alcance das pesquisas de ESM, a popularização dos computadores pessoais e avanço das TICs permitiu maior controle de variáveis, rapidez e flexibilidade de programas de ensino individualizados, de modo que os computadores passaram a ser utilizados para autoria e aplicação de programas de ensino cada vez mais complexos, bem como para o registro do desempenho dos aprendizes [De Souza *et al*. 2009]. Também a programação de ensino individualizada tem se beneficiado dos dispositivos móveis, sendo comum a criação de aplicativos simples para o ensino de variados repertórios comportamentais, como relações de igualdade ou desigualdade entre estímulos, reconhecimento de expressões faciais e emoções, formação de conceitos, dentre outros.

A presença cada vez mais intrínseca de diferentes soluções de software, desenvolvidas para a computação cada vez mais presente no cotidiano das pessoas, podendo ser elas um sistema, um aplicativo ou um serviço, pode e deve ser aproveitada com objetivos de pesquisa e/ou intervenções clínicas.

5.2.2. Computação Ubíqua

A adoção de sistemas computacionais no cotidiano das pessoas é objeto de estudo da área de Computação Ubíqua, que busca criar soluções para viabilizar o conceito da ubiquidade. Weiser [1991], um dos principais autores e fundadores da área, aponta que as tecnologias mais profundas são aquelas que tendem a "desaparecer", a ponto de serem indistinguíveis do ambiente. Para definir esse conceito, Weiser utilizou a palavra *ubiquitous* ou ubíquo. Segundo o dicionário Cambridge, a palavra ubíquo significa encontrado ou existente em toda a parte. Logo, ser ubíquo significa ser onipresente. Além da onipresença, Weiser também associou o "desaparecimento" dos computadores ao significado da palavra ubíquo. Dessa maneira, para ser ubíquo, segundo a definição de Weiser, os sistemas devem ser onipresentes e proporcionarem transparência de uso aos usuários.

Atualmente a ubiquidade tem estado cada vez mais enraizada no cotidiano das pessoas, de modo que essas usam naturalmente sistemas computacionais em seu dia-a-dia para ajudarem-nas a cumprir tarefas cotidianas de maneira inconsciente e automática.

Ainda considerando este contexto, são encontrados na literatura trabalhos que fazem referência à Computação Pervasiva. A palavra pervasivo, por si só, não define a ubiquidade descrita por Weiser, entretanto, Satyanarayanan [2001] define a computação pervasiva como outra forma de se referir à computação ubíqua. Araújo [2003] por sua vez, aponta que a computação ubíqua e a computação pervasiva são definidas pelo grau de "embarcamento"[4] e de mobilidade de cada uma. De acordo com Araújo, para ser ubíqua, a solução de software deve ter um alto nível de embarcamento e de mobilidade, enquanto que o conceito de pervasivo envolve apenas um alto nível de embarcamento. Essa definição é apoiada por Lyytinen e Yoo [2002], que se referem à computação ubíqua como a possibilidade de qualquer dispositivo configurar os serviços do ambiente em que está, mesmo enquanto se move com o usuário.

Um dos principais objetivos da computação ubíqua é o desenvolvimento de modelos que levam em conta as atividades realizadas por indivíduos e os contextos em que ocorrem [Yin *et al*. 2008], [Kapoor and Horvitz 2008], o que torna soluções de computação ubíqua grandes facilitadoras não apenas ao acesso de informações sobre comportamentos de interesse, mas também de ações interventivas.

Uma vez que o acesso à informações atualmente é cada vez mais facilitado pela computação ubíqua, este grupo de pesquisadores viu a oportunidade de unir o acesso a dados de interesse, por meio de técnicas baseadas em procedimentos selecionados do método de ESM, para implementar planos de intervenções programadas — de modo que tanto a coleta de informações quanto a intervenção planejada possam fazer uso de dispositivos móveis, vestíveis ou integrados a ambientes inteligentes, por exemplo. Acreditamos que quanto mais contextualizadas e individualizadas sejam as ações interventivas, essas serão mais relevantes, adequadas e aplicáveis ao cotidiano e necessidades de diferentes indivíduos, aumentando assim, as chances de sucesso das intervenções propostas.

5.3. O método ESPIM

Como anteriormente mencionado, o ESPIM - *Experience Sampling and Programmed Intervention Method* - combina técnicas de ESM, procedimentos selecionados de programação de ensino individualizada e a computação ubíqua, com o objetivo de aumentar a efetividade e otimizar a coleta de dados, bem como o planejamento, autoria e aplicação de intervenções a distância programadas por diferentes áreas de domínio, em especial saúde e educação.

Atualmente o método ESPIM tem o suporte de um sistema para a plataforma Web, que trata da interface com o especialista, e de um aplicativo para a plataforma *mobile*, que permite a coleta de dados e intervenções a distância. Este sistema é uma ampliação a um sistema anterior de uma plataforma intitulada *SmartESM*, destinada em

[4] O termo "embarcamento" é utilizado por Araújo [2003] para se referir ao grau de inteligência dos computadores, embutidos em um ambiente pervasivo, para detectar, explorar e construir dinamicamente modelos de seus ambientes.

especial, ao envio de alertas para responder a questionários, contando com uma aplicação Web de interface gráfica que permite visualizar, adicionar, editar e gerenciar elementos relevantes (e.g. participantes, pesquisadores, experimentos, questionários, lembretes, alertas etc.), e com um aplicativo móvel desenvolvido na plataforma Android, para ser usado em *smartphones* [Pimentel *et al.* 2016].

O sistema ESPIM está sendo desenvolvido utilizando uma abordagem participativa e de coprodução [Schuler and Namioka 1993], em que profissionais de diferentes áreas, entre elas computação, saúde e educação, colaboram na elicitação de requisitos e também na avaliação do sistema.

A adoção da abordagem participativa corrobora com princípios do Design Centrado no Usuário (DCU) [Lowdermilk 2013], cujo foco está nas necessidades, desejos e limitações dos usuários durante todo o projeto, a cada tomada de decisão, desde a concepção até o lançamento do produto. O envolvimento com usuários alvo permite identificar a necessidade real desses. O DCU possui quatro etapas básicas:

1. *Identificar requisitos*: etapa de clarificar problemas, levantar necessidades e entender os pontos de conflitos dos usuários através de pesquisas, observações e entrevistas;
2. *Criar soluções alternativas*: etapa de explorar ideias e sugestões, momento em que são levantadas hipóteses de soluções para as necessidades levantadas;
3. *Construir protótipos testáveis*: etapa em que são criados modelos testáveis do que pode vir a ser o produto final;
4. *Avaliar com usuários*: etapa em que os protótipos são levados para testes com usuários e *feedbacks* sobre o que funciona e o que pode melhorar são coletados.

Este minicurso apresenta para a comunidade do Simpósio Brasileiro de Sistemas Multimídia e Web (*WebMedia*) o método ESM e o ESPIM. O conteúdo será abordado de maneira teórica e também prática. Na etapa teórica serão apresentados o método e a sua aplicabilidade nas diversas áreas de conhecimento. A coleta de dados de maneira ubíqua sobre as experiências dos indivíduos – sendo realizada em tempo real e em ambiente natural – bem como o planejamento e a implementação de intervenções baseada em dados previamente coletados, também serão abordados nessa etapa.

Na sequência, a infraestrutura tecnológica utilizada no desenvolvimento do ESPIM e os seus componentes serão detalhados. Os componentes do ESPIM incluem: um aplicativo móvel a ser utilizado por usuários que estão sendo acompanhados; um sistema Web para o planejamento das coletas e intervenções, a ser utilizado por pesquisadores e profissionais e, um *Web Service* responsável pelo armazenamento das informações. Esses componentes são descritos na Seção 5.4.

Ainda na etapa teórica serão apresentadas as funcionalidades do ESPIM e atividades que cada perfil de usuário pode realizar, entre as atividades: a criação de programas personalizados para coleta de dados e cadastro de participantes.

Na etapa prática do curso cada participante poderá interagir com o ESPIM criando experimentos/intervenções, observando como a coleta de dados pode ser feita e como os participantes são acompanhados. Espera-se que *feedbacks* sobre o sistema possam ser coletados nessa etapa, principalmente no que diz respeito a aspectos relacionados à usabilidade e à experiência de uso.

5.3.1. Exemplo de cenário de uso do ESPIM

Esta seção se dedica à apresentação em detalhes de um possível cenário para o uso do ESPIM, tanto para a coleta de dados e de experiências dos usuários, como para servir de gatilho para o início de atividades de intervenção programadas por especialistas de área. O cenário retomará alguns aspectos apresentados anteriormente pela Figura 1, em um contexto que envolve idosos.

Cenário Exemplo - Acompanhamento e avaliação de desempenhos cognitivos de idosos

O senhor Josias, 83 anos, foi diagnosticado clinicamente com Doença de Alzheimer, apresentando indícios de perdas cognitivas, tais como confusão mental, prejuízos em adquirir e relembrar novas informações e dificuldade em reconhecer pessoas em seu convívio diário. Dessa maneira, o geriatra responsável pelos cuidados do senhor Josias elaborou uma série de atividades para que o mesmo se engajasse, utilizando seu *smartphone*, durante o período de uma semana. As tarefas tinham os objetivos de: 1) Coletar dados acerca da rotina de Josias; 2) Coletar informações acerca dos sentimentos (experiências) de Josias ao longo da semana e, 3) Aumentar a probabilidade de que Josias realizasse atividades de estimulação cognitiva. O fluxo de atividades tanto de coleta de dados quanto de intervenções programadas pelo geriatra é apresentado pela Figura 5.2.

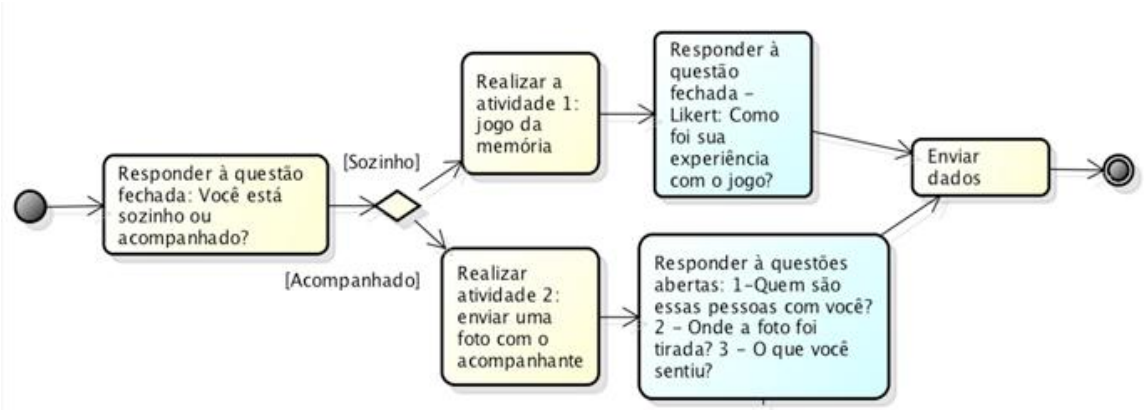


Figura 5.2. Fluxograma de encadeamento das atividades planejadas para o caso de acompanhamento de idosos.

O geriatra considerou apropriado que fossem feitas interações diárias com o sistema, 3 vezes ao dia, uma no período da manhã (das 8h às 12h), uma no período da tarde (das 12:30h às 17:30h) e outra no período da noite (das 18h às 21:30h). Assim, o sistema irá mandar três alertas diários para o início da interação, por um período de 7 dias consecutivos. Observa-se que o gatilho inicial é o recebimento de um alerta no *smartphone* para que o senhor Josias responda à seguinte pergunta de texto: “Você está sozinho ou acompanhado?”. Essa pergunta será respondida por meio da seleção de uma das possibilidades de resposta em uma questão do tipo múltipla escolha com uma opção de escolha, em que as opções são: ( ) sozinho; ( ) acompanhado. O geriatra tem planos distintos de acordo com respostas diferentes, assim, a continuidade de interação com o sistema poderá tomar dois caminhos: se a resposta à questão anterior for “sozinho”, será enviado imediatamente uma solicitação para que o senhor Josias inicie uma atividade de *Jogo de memória*. Uma instrução é exibida na tela do *smartphone*:

"Vamos jogar o jogo da memória!", seguido de um botão "Início". No momento em que o idoso seleciona o botão, o jogo (atrelado previamente pelo geriatra a essa intervenção) é iniciado. O sistema mantém o registro automático de informações à respeito da interação do usuário com o jogo. Entre os registros coletados estão: a quantidade de erros/acertos e o tempo gasto na realização da atividade.

Assim que a atividade do *Jogo da Memória* for encerrada, uma mensagem de *feedback* é enviada ao senhor Josias, por exemplo, "Muito bem! Você terminou o jogo!". Em seguida, o sistema imediatamente inicia uma nova interação, solicitando que o senhor Josias relate o grau de satisfação ao jogar o jogo. O sistema apresenta a seguinte pergunta: "Como foi sua experiência com o jogo?". O senhor Josias irá selecionar sua resposta, em uma escala analógica do tipo Likert com cinco opções: 0 - *detestei*; 1 - *não gostei*; 2 - *neutro*; 3: *gostei*; e 4: *adorei*. A resposta a essa pergunta encerra o episódio interativo e os dados coletados são enviados ao servidor Web do ESPIM. Uma nova interação com o sistema irá ocorrer novamente (ou não) de acordo com os critérios pré-estabelecidos pelo especialista de área, neste caso, o geriatra.

Agora consideremos novamente a pergunta inicial que foi o gatilho para esta interação. Caso a resposta do senhor Josias seja "acompanhado", o geriatra programou a interação de outras diferentes atividades. Ele deverá tirar uma foto com o acompanhante e enviá-la. O sistema irá solicitar que Josias faça isso por meio da instrução de voz: "Legal, vamos tirar uma foto com seu acompanhante". O aplicativo aciona então a câmera do dispositivo, a foto é tirada e enviada ao servidor. Ao final do envio da foto, o aplicativo apresentará uma mensagem de *feedback*, por exemplo: "OK! Foto enviada com sucesso!". O fim dessa atividade marca o encadeamento para a atividade seguinte em que o Josias irá responder a três perguntas abertas, por meio da gravação de áudio, por exemplo: a) "Quem são essas pessoas com você?"; b) Onde a foto foi tirada?; "O que você sentiu?". Responder à essas três questões encerra o episódio interativo e os dados coletados são enviados ao servidor Web do ESPIM para análise do especialista.

A seção a seguir descreve como o sistema ESPIM foi desenvolvido considerando questões relacionadas aos seus componentes, bem como as linguagens de programação e *frameworks* utilizados no desenvolvimento desses componentes.

5.4. Desenvolvimento do sistema ESPIM

Conforme exposto, o sistema ESPIM é multiplataforma (Web e *mobile*) e tem sido desenvolvido com o apoio de uma equipe multidisciplinar (profissionais da computação, saúde e educação). O sistema deve apoiar pesquisadores que utilizam o ESM e servir de laboratório para pesquisas associadas ao método.

As subseções a seguir detalham a arquitetura do sistema ESPIM e os seus componentes.

5.4.1. Infraestrutura Tecnológica

A Figura 5.3 ilustra a arquitetura de alto nível proposta para a implementação do sistema ESPIM. As definições de um programa – definido por um especialista – e os dados de resultados, coletados via intervenção com os usuários acompanhados, serão armazenados em um servidor Web. Para armazenar e acessar esses dados uma arquitetura REST é utilizada, de forma que os dados são obtidos e enviados via HTTP.

O cliente, neste caso um aplicativo móvel ou uma página da Web, não tem acesso direto à modelagem ou aos dados do programa.

Para facilitar a implementação de clientes que precisam se comunicar com o serviço Web, foi desenvolvida uma biblioteca em Java que simplifica as chamadas REST para o *Web Service*. A biblioteca de monitoramento simplifica todas as operações e converte os documentos JSON em objetos Java. O uso da biblioteca não é obrigatório, de modo que é possível desenvolver clientes Web e móveis utilizando diferentes linguagens de programação, independente da linguagem utilizada no *Web Service* e na biblioteca.

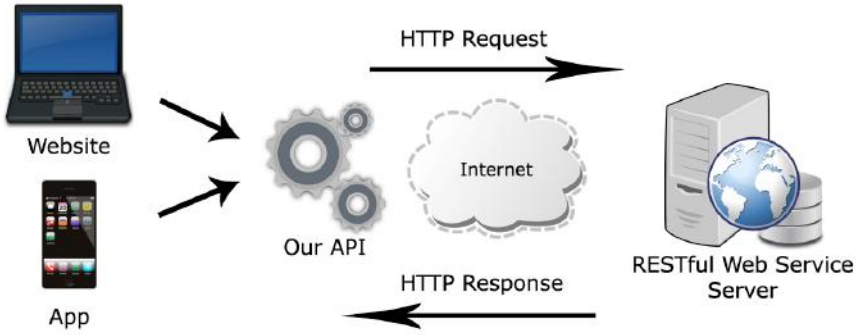


Figura 5.3. Arquitetura de alto nível proposta para o sistema ESPIM.

Os componentes básicos do sistema ESPIM foram desenvolvidos com base na arquitetura da Figura 5.3. Os componentes são:

1) Um *Web Service RESTful*, desenvolvido em Java com o *framework Spring Boot*, responsável pelo armazenamento das informações;

2) Um sistema Web, desenvolvido com as linguagens HTML5 e JavaScript para o planejamento das coletas e intervenções, a ser utilizado por usuários especialistas e;

3) Um aplicativo móvel desenvolvido em Android a ser utilizado pelos usuários que estão sendo acompanhados por especialistas.

Esses componentes são descritos em detalhes nas Seções 5.4.1.1, 5.4.1.2 e 5.4.1.3, respectivamente.

Além dos três componentes apontados acima, foram desenvolvidos componentes para acessar e controlar sensores de dispositivos móveis que podem ser utilizados na construção de intervenções que fazem uso de sensores para coleta de dados específicos, tais como a localização via GPS, sensor de luminosidades, etc. Tais componentes foram organizados em uma arquitetura estrutural que será descrita na Seção 5.4.2.

5.4.1.1. *Web Service RESTful*

O *Web Service RESTful* contém a modelagem do método ESPIM e é responsável por armazenar e prover dados para o sistema Web e para o aplicativo móvel. A Figura 5.4 apresenta a modelagem de eventos para o método ESPIM.

A modelagem proposta prevê a definição de diferentes tipos de eventos generalizáveis para diferentes cenários no domínio da saúde e educação. Os principais elementos da modelagem são:

- *Programa*: um programa é um conjunto de eventos de monitoramento que podem estar associados a um ou mais participantes monitorados. Eventos podem ser classificados como ativos, que são aqueles que necessitam de uma interação explícita com o dispositivo móvel, ou passivos, em que participantes têm informações coletadas em plano de fundo e não precisam interagir diretamente;
- *Pessoa*: pode ser um participante do programa, ou seja, o indivíduo monitorado, ou um observador/cuidador, responsável pelo participante e seu respectivo programa;
- *Evento*: um evento pode ser do tipo ativo ou passivo. Um evento passivo é aquele que coleta informações de sensores de *smartphones* em intervalos contínuos de tempo. Enquanto um evento ativo possui um conjunto de intervenções. Uma intervenção se caracteriza pela necessidade de interação explícita com o usuário, ou seja, ocorre uma interrupção que deve ser atendida pelo participante. Um evento ativo pode coletar dados de sensores durante a interação do participante com suas intervenções;
- *Gatilho de Evento*: um gatilho de evento pode ter diferentes tipos e condições: manual, temporal e contextual. Um gatilho manual dispara um evento apenas quando requisitado pelo participante. O gatilho temporal dispara eventos agendados em horários e dias específicos. Já o gatilho contextual considera informações de sensores e de contexto para disparar eventos;
- *Sensor*: o objeto sensor define quais informações de sensores serão coletadas e em que condições. Utilizando os sensores do dispositivo, podem ser coletadas informações sobre atividades (via acelerômetro e giroscópio), luz ambiente (via câmera) e áudio ambiente (via microfone), por exemplo;
- *Intervenção*: um evento ativo é composto por uma ou mais intervenções. Uma intervenção representa uma ocorrência em que é necessário interagir com o dispositivo, ou seja, é uma situação em que o usuário é interrompido pelo dispositivo para interagir com lembretes, questionamentos ou requisições de tarefa. Uma intervenção pode apresentar na tela diferentes estímulos: texto, imagem, áudio ou vídeo. O modelo atualmente prevê quatro tipos de intervenção que determinam a forma de interação com o participante: vazia, questão, mídia e tarefa. Uma intervenção vazia se caracteriza por não necessitar da entrada de dados por parte do usuário, sendo adequada para representar lembretes, mensagens ou estímulos. Uma intervenção em forma de questão pode ser de diferentes tipos, como aberta, múltipla escolha com uma única seleção, múltipla escolha com múltipla seleção, escala Likert e diferencial semântico. Uma intervenção de mídia requer que o participante capture uma imagem, um áudio ou um vídeo. Já uma tarefa trata da execução de tarefas externas ao modelo, permitindo que outros aplicativos sejam iniciados pela aplicação móvel, como um jogo. Diferentes tipos de intervenção podem ser combinadas em um único evento. Outro elemento importante em uma intervenção é uma condição. Se a intervenção for uma questão de múltipla escolha, o especialista pode definir um novo caminho dependendo da resposta selecionada pelo participantes. Isto implica na criação de fluxos não sequenciais de execução;
- *Resultado*: o objeto resultado é responsável pela persistência dos dados coletados e seus metadados associados. Um resultado é sempre associado a um participante e à sua intervenção correspondente.

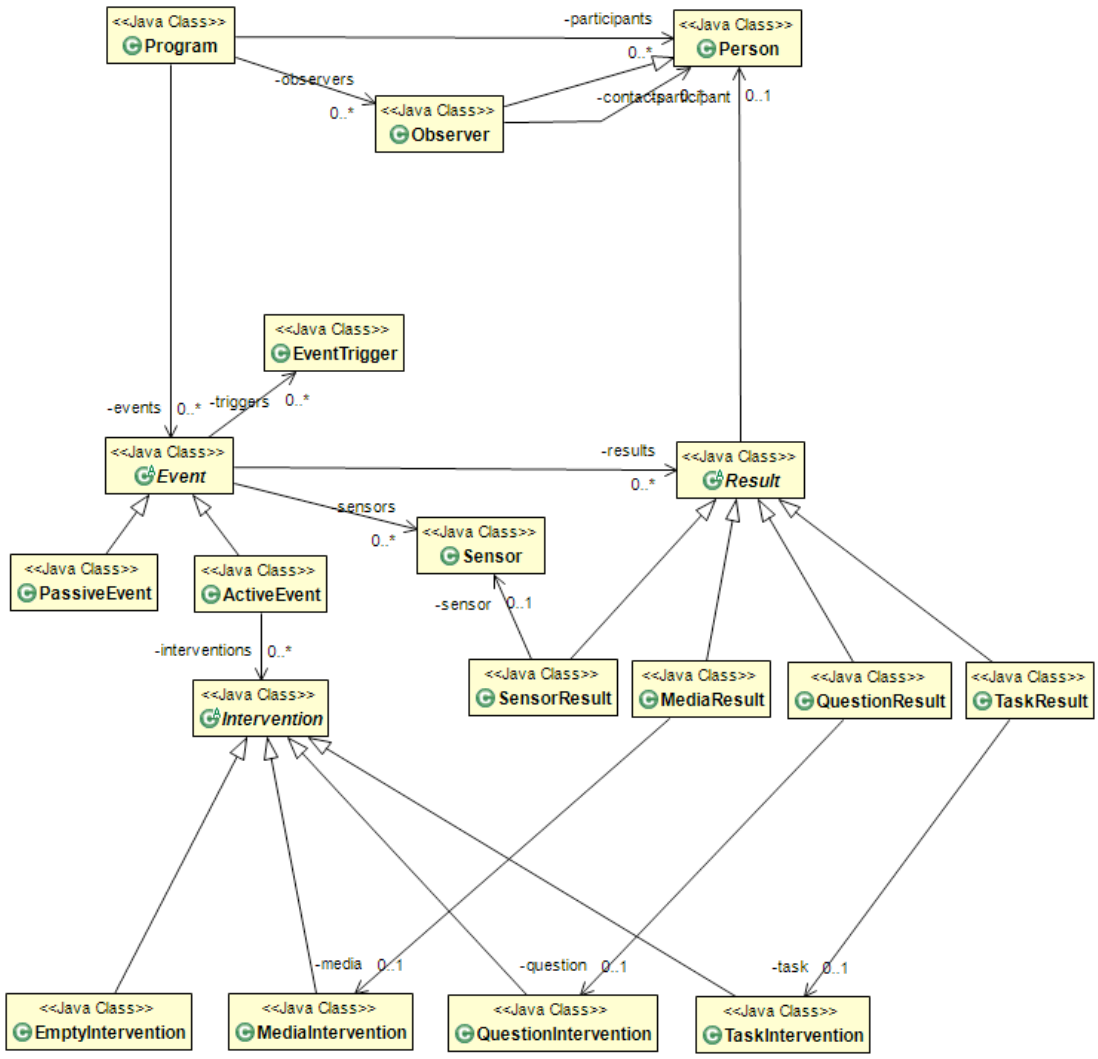


Figura 5.4. Modelagem de um programa no método ESPIM.

5.4.1.2. Sistema Web para Especialistas

A interface Web permite a criação de programas personalizados, cadastro de participantes e visualização dos resultados enviados ao servidor. A interface para criação de programas é baseado no modelo já discutido e apresentado na Figura 5.4.

Um usuário especialista cadastrado no sistema pode criar um programa, composto por eventos que compreendem um conjunto de intervenções. O programa pode ser associado a um ou mais participantes. Atualmente o interface Web possibilita a criação de intervenções com questões do tipo aberta, múltipla escolha com uma opção ou múltiplas escolha com mais de uma opção, mensagens e tarefas (em que um aplicativo externo é iniciado).

A interface foi construída utilizando o *framework* MCV Grails. O *framework* foi escolhido para facilitar a integração do *Web Service* com a biblioteca Java desenvolvida, pois o *framework* também suporta a linguagem Java. As páginas HTML foram construídas utilizando o biblioteca JavaScript jQuery e o *framework* CSS Bootstrap. A interface se comunica com o *Web Service* por meio de requisições HTTP que trocam

informações no formato JSON, de modo que o *Web Service* é o responsável pela persistência dos dados.

O *login* do especialista na interface é feito por meio de seu conta no Google. O cadastro de participantes é feito por meio de um formulário e também é associado a seu endereço de e-mail da conta Google. Esse formato foi utilizado para facilitar a integração e o *login* com sistemas Android. Dessa maneira, o usuário não precisa fornecer *login* e senha para utilizar a aplicação Android. Além disso, o mecanismo de autenticação é de responsabilidade da API do Google. O cadastro de programas, por sua vez, é feito por um formulário que oferece a funcionalidade de associar participantes àquele programa específico. Os eventos são construídos por meio de um formulário dinâmico semelhante a um fluxograma, que permite criar intervenções e conectá-las de maneira flexível, criando assim, fluxos não sequenciais.

A Figura 5.5 ilustra um exemplo de evento que pode ser programado pela interface Web. Nesse exemplo, semelhante àquele ilustrado no Cenário Exemplo com um idoso, uma questão de múltipla escolha (“Você está sozinho ou acompanhado?”) define o fluxo das intervenções. Se o participante estiver sozinho, ele deve jogar um aplicativo de jogo de memória e, no final, é parabenizado por sua atividade. Se estiver acompanhado, o participante deve responder quem o acompanha (questão aberta) e qual o seu humor naquele momento (questão de múltiplas opções) frente à interação realizada.

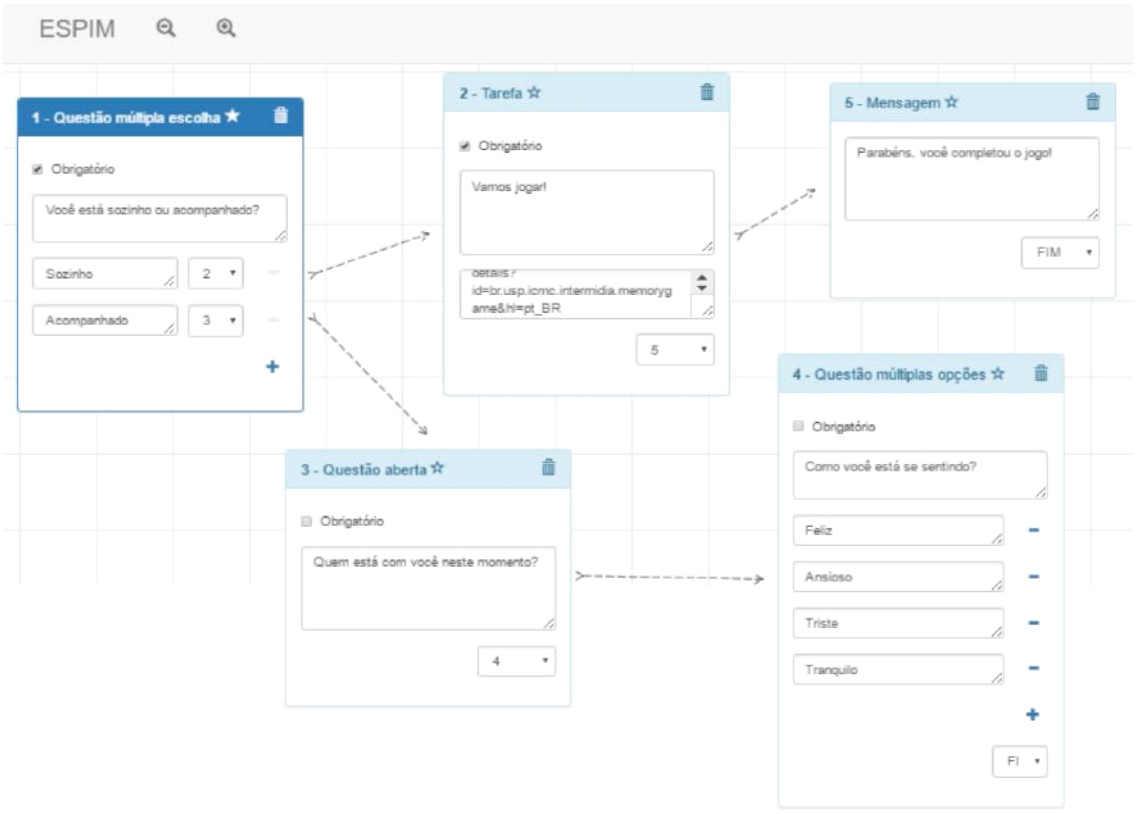


Figura 5.5. Exemplo da interface Web para criação de um conjunto de intervenções com condição.

Após finalizar o bloco de eventos, a interface verifica erros e inconsistências no grafo de intervenções e notifica o usuário ao identificar problemas. Se o evento estiver consistente, no próximo passo o especialista deve definir um gatilho temporal,

delimitando o horário e dias da semana que o evento será disparado no dispositivo do participante, conforme ilustrado na Figura 5.6. Essas informações são codificadas em uma expressão *CRON*[5].

Figura 5.6. Definindo o gatilho temporal.

Ao finalizar o programa, os dados associados a ele são codificados em JSON e enviados ao *Web Service* para persistência e posterior recuperação por um sistema de acompanhamento (por exemplo, a aplicação móvel descrita na seção a seguir).

É também pela interface Web que o especialista pode acessar os dados coletados pelas intervenções. Nesse caso, a interface acessa o *Web Service* que retorna um documento de texto no formato *Comma-Separated Value* (CVS). Esse documento pode ser aberto em qualquer processador de texto e é compatível com softwares de planilha eletrônica, como o *Microsoft Excel*, contendo os dados associados àquele programa. Esse arquivo fica disponível para *download* pelo especialista.

5.4.1.3. Aplicativo Móvel para Acompanhamento

Os programas ESPIM são entidades totalmente personalizáveis. Ou seja, um profissional ou pesquisador pode definir um programa de acordo com sua área, metodologia e pessoas envolvidas. Isto implica na necessidade de desenvolvimento de aplicações que leiam as definições de um programa ESPIM e se comportem de acordo com os eventos, gatilhos e intervenções definidos.

Neste contexto, foi desenvolvido um aplicativo para o sistema Android que recupera os programas estipulados pelo especialista, para o usuário do dispositivo via REST. O aplicativo então gera as telas de intervenções dinamicamente por meio da leitura de arquivos JSON, os quais definem a estrutura de um programa. Além de gerar as telas, o aplicativo também cria dinamicamente notificações com base nos horários definidos nos gatilhos do arquivo JSON. Para a comunicação com o *Web Service*, o aplicativo desenvolvido utiliza a biblioteca em Java que simplifica as chamadas REST.

As Figuras 5.7 e 5.8 ilustram as telas correspondentes ao programa definido pela Figura 5.6. Tais figuras mostram o fluxo de acordo com a resposta para a primeira questão respondida pelo participante. Intervenções obrigatórias são acompanhadas de um asterisco (*). Se o participante tentar seguir para a próxima intervenção sem responder ou realizar a tarefa estipulada, o aplicativo retorna uma mensagem de alerta. Todas as telas apresentam o botão “Próximo”, para avançar a questão, e o botão

[5] https://docs.oracle.com/cd/E12058_01/doc/doc.1014/e12030/cron_expressions.htm

"Voltar", para destacar a possibilidade de voltar à questão anterior. Há um cuidado especial com o fluxo das telas pois, em casos de fluxos condicionais, a ação de voltar deve considerar as escolhas anteriores do participante. Na primeira tela não é apresentado o botão "Voltar". Na última tela o botão "Terminar" é exibido e, ao clicar no botão, uma mensagem de confirmação é exibida. Para os casos em que uma tarefa deve ser realizada, o participante deve clicar no botão "Iniciar" e, na sequência, é disparado o aplicativo correspondente àquela tarefa.

Já para uma intervenção do tipo *mensagem*, o botão "Continuar" apresenta o mesmo comportamento dos botões "Próximo" e "Terminar". Após responder às intervenções, o aplicativo envia as respostas das questões em formato JSON para o *Web Service*.

Para prover uma usabilidade adequada a um público mais amplo, a interface do aplicativo seguiu uma série de diretrizes para usuários da terceira idade e foi submetida a uma avaliação heurística com a colaboração de quatro alunos de pós-graduação.

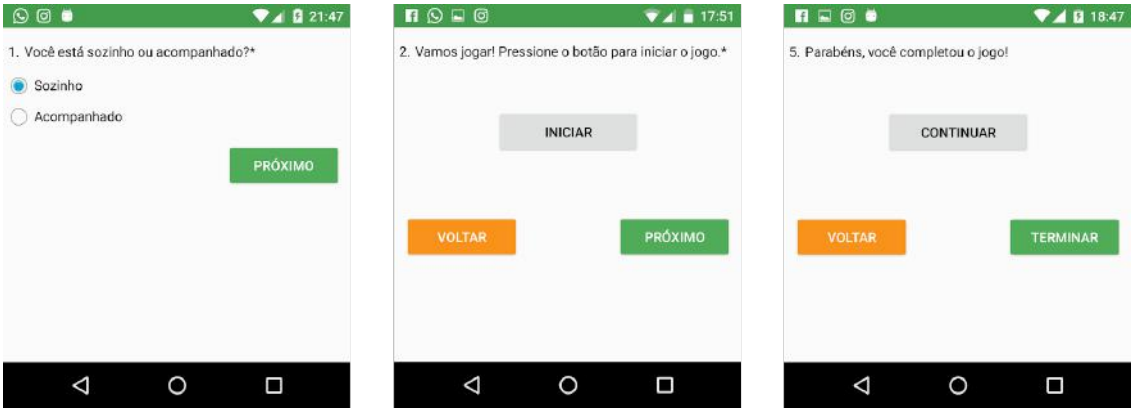


Figura 5.7. Fluxo de intervenções 1 - Se o participante estiver sozinho: realizar a tarefa de jogar.

Figura 5.8. Fluxo de intervenções 2 - Se o participante estiver acompanhado: perguntar quem o acompanha e qual o seu humor.

5.4.2. Componentes para acesso a Sensores

Os aplicativos em Android que utilizam os sensores dos dispositivos móveis para acessarem funcionalidades diversas podem ser implementados utilizando um modelo arquitetural que disponibiliza as principais funcionalidades de cada sensor desses

dispositivos e permite que o desenvolvedor escolha quais sensores utilizar, de forma simplificada. Essa arquitetura foi elaborada para que futuras aplicações, que utilizem um número maior de sensores e outros hardwares (como câmera e microfone), possam ser implementadas de forma mais simples e em tempo reduzido. O modelo segue o padrão de projetos MVC (*Model View Controller*), que separa a definição dos dados a serem armazenados (*Model*) da camada de visualização (*View*) e da entrada de dados (*Controller*). A arquitetura completa está ilustrada na Figura 5.9.

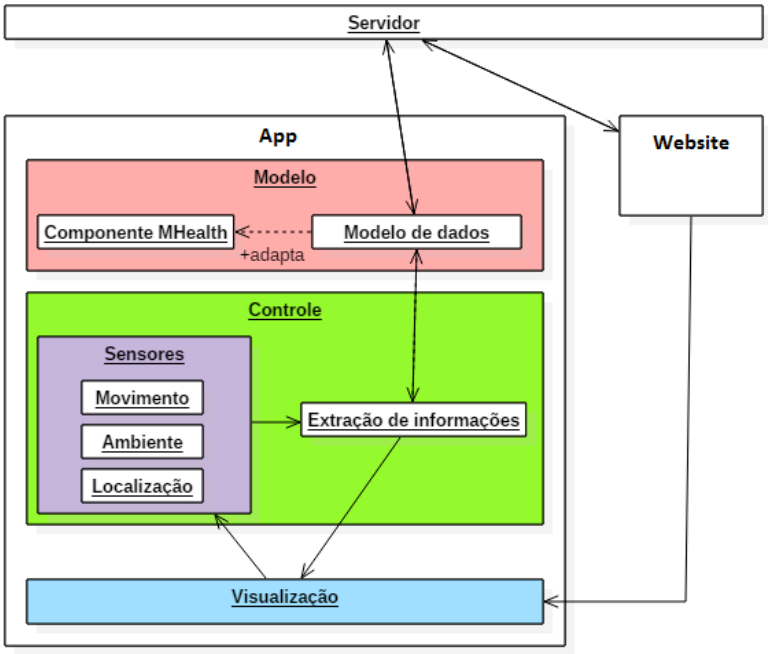


Figura 5.9. Arquitetura para desenvolvimento de aplicativos de dispositivos móveis.

A interface do aplicativo Android é representada pelo componente "Visualização". A partir da interface, a aplicação tem acesso à camada de controle. O desenvolvedor também pode contar com arquivos XML que representam fragmentos de interfaces para a câmera, para o microfone e para os sensores, os quais podem acessar as funcionalidades desses dispositivos por meio da camada de controle.

O componente "Sensores" contém classes de fachada com as funcionalidades mais importantes de cada sensor já implementadas, cabendo ao desenvolvedor escolher quais sensores serão utilizados em sua aplicação, bem como as funcionalidades que ele julgar necessárias.

Apesar dos dados dos sensores disponibilizados no módulo “Sensores” serem suficientes para a implementação de muitas aplicações, o desenvolvedor pode realizar operações sobre os dados obtidos. Esse pós-processamento de informações está ilustrado pelo componente “Extração de informações”, o qual é implementado inteiramente pelo desenvolvedor, com base em suas necessidades.

Após o processamento das informações pela aplicação, os dados podem ser armazenados na camada “Dados”. Nessa etapa, o desenvolvedor tem a opção de armazenar os dados em JSON no próprio dispositivo, para posterior envio ao servidor. Essa opção é necessária em caso de limitações momentâneas de conexão com a Internet.

A camada "Dados" está associada à esquemas pré-definidos no formato JSON implementados em Java, que definem a estrutura correta de parâmetros médicos específicos para aplicações para a saúde, por exemplo. Tais esquemas são recomendados pela plataforma *openMHealth*[6], que disponibiliza um repositório contendo recomendações de como armazenar dados médicos referentes a diversos parâmetros, tais como altura, alimentação, batimentos cardíacos, dentre outros. Os esquemas são, portanto, uma compilação de atributos utilizados para descrever os parâmetros de interesse dos profissionais da área médica. Contudo, o uso dos esquemas não é obrigatório para o desenvolvimento de aplicações usando a arquitetura apresentada.

A seção a seguir detalha as funcionalidades do ESPIM Web em sua atual versão.

5.5. Utilizando o ESPIM para criar programas para coleta de dados a distância

As figuras abaixo ilustram telas da interface Web do ESPIM. Por meio delas é possível realizar o cadastro no sistema, criar novos participantes, editar informações de participantes existentes, programar uma coleta de dados, criar eventos de coleta, selecionar usuários para eventos específicos, definir disparos, dentre outras funcionalidades.

A Figura 5.10 exibe a tela em que o especialista realiza o seu cadastro no sistema informando nome, e-mail, senha e o seu papel. O papel define a função do especialista na coleta de dados que ele realizará por meio do ESPIM, por exemplo: terapeuta, professor, geriatra etc.

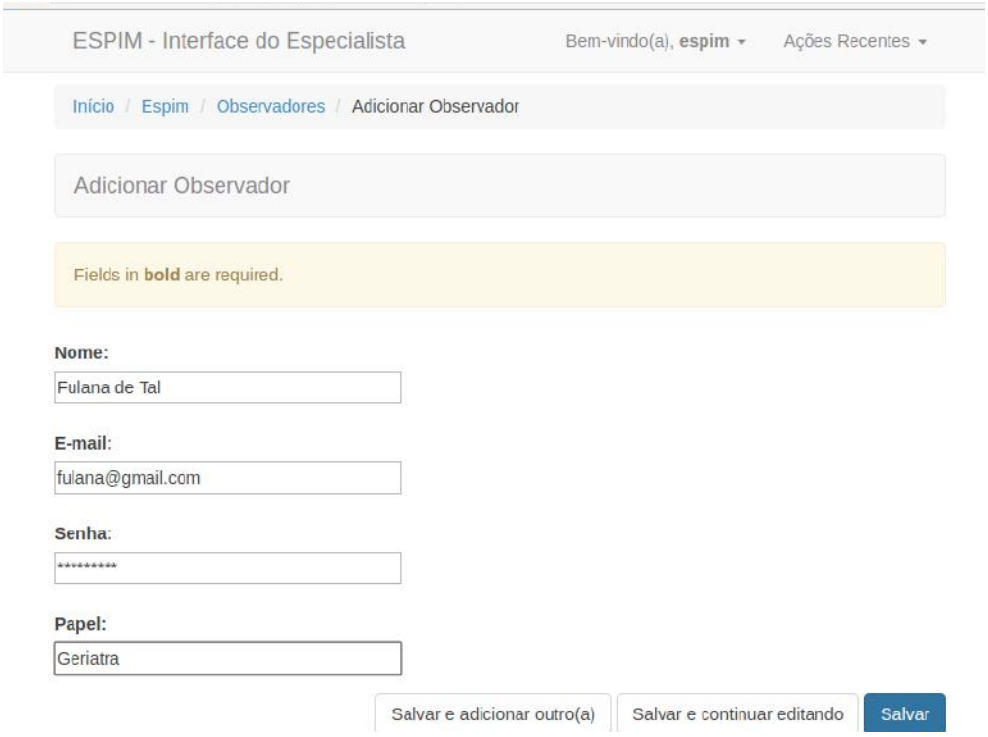


Figura 5.10. Tela de cadastro do especialista no ESPIM.

[6] Documentação disponível em http://www.openmhealth.org/documentation/\#/schema-docs/schema-library.

Após realizar o seu cadastro no ESPIM, o especialista pode adicionar novos participantes, bem como novos observadores, eventos e programas, conforme ilustrado na Figura 5.11.

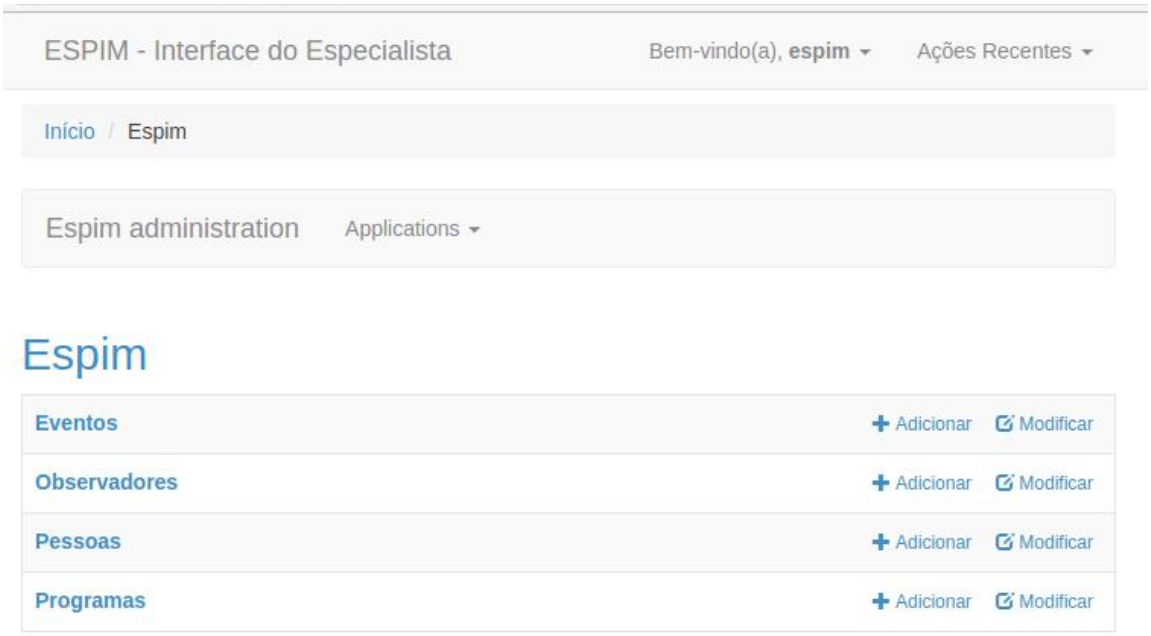


Figura 5.11. Tela com possibilidades de cadastros.

O especialista pode cadastrar novos participantes e editar informações de participantes já adicionados, conforme ilustrado na Figura 5.12.

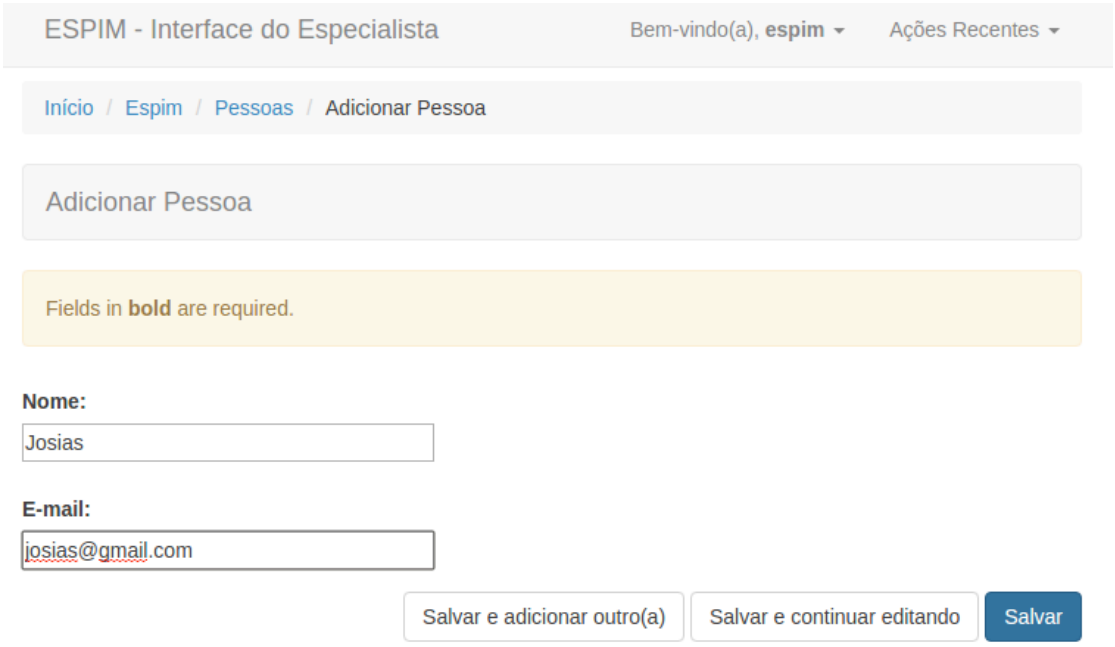


Figura 5.12. Tela de cadastro de novos participantes.

O especialista também pode cadastrar um programa interventivo. Ao realizar esse cadastro é preciso definir um título, uma breve descrição, associar participantes e observadores para aquela intervenção (Figura 5.13).

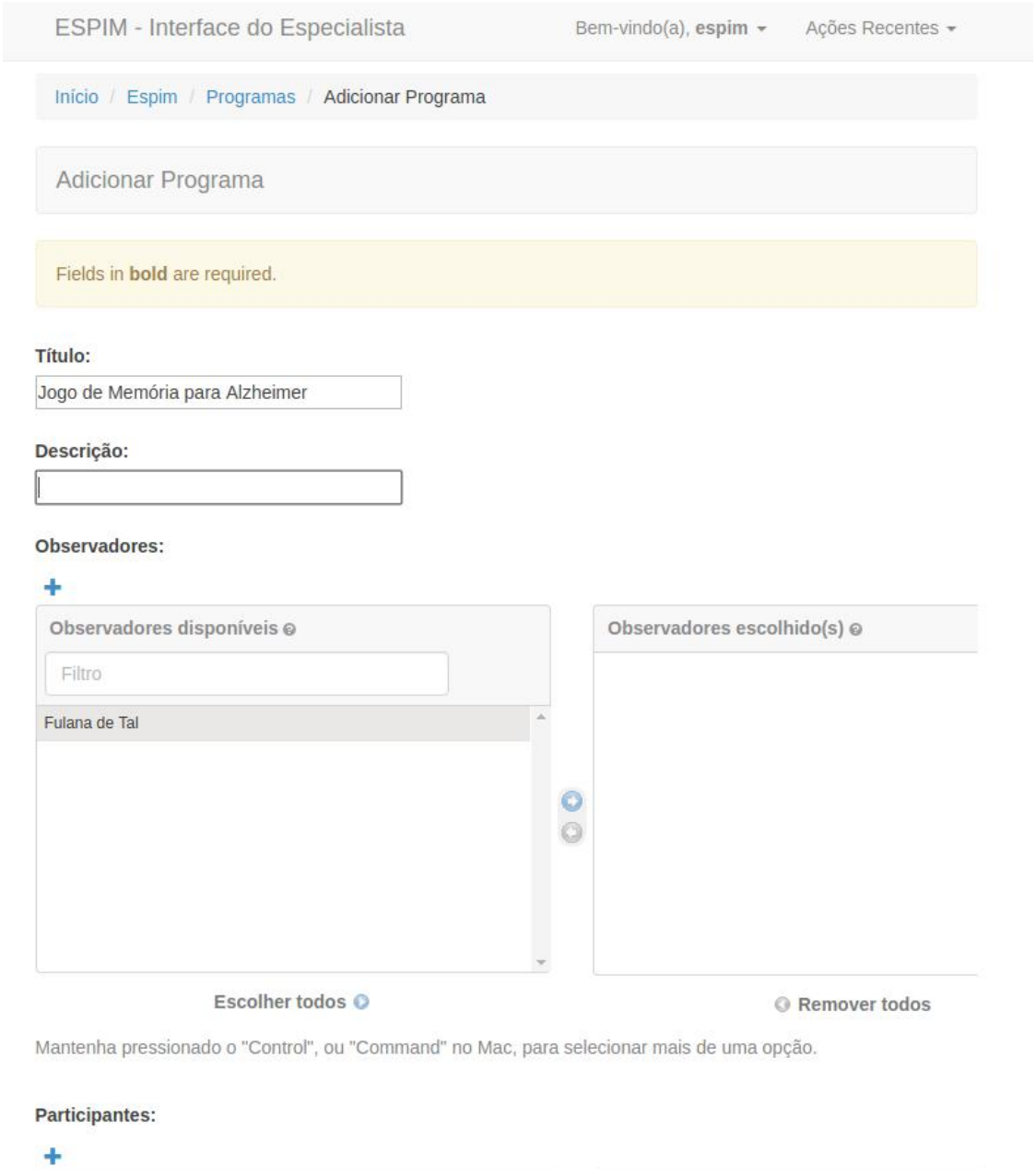


Figura 5.13. Tela de cadastro de programas interventivos.

As Figuras 5.14a e 5.14b ilustram a tela em que o especialista seleciona os participantes que farão parte da coleta de dados.

Prevendo a existência de diversos participantes, a interface do ESPIM permite que os especialistas sejam selecionados pelo nome usando também o serviço de auto completar.

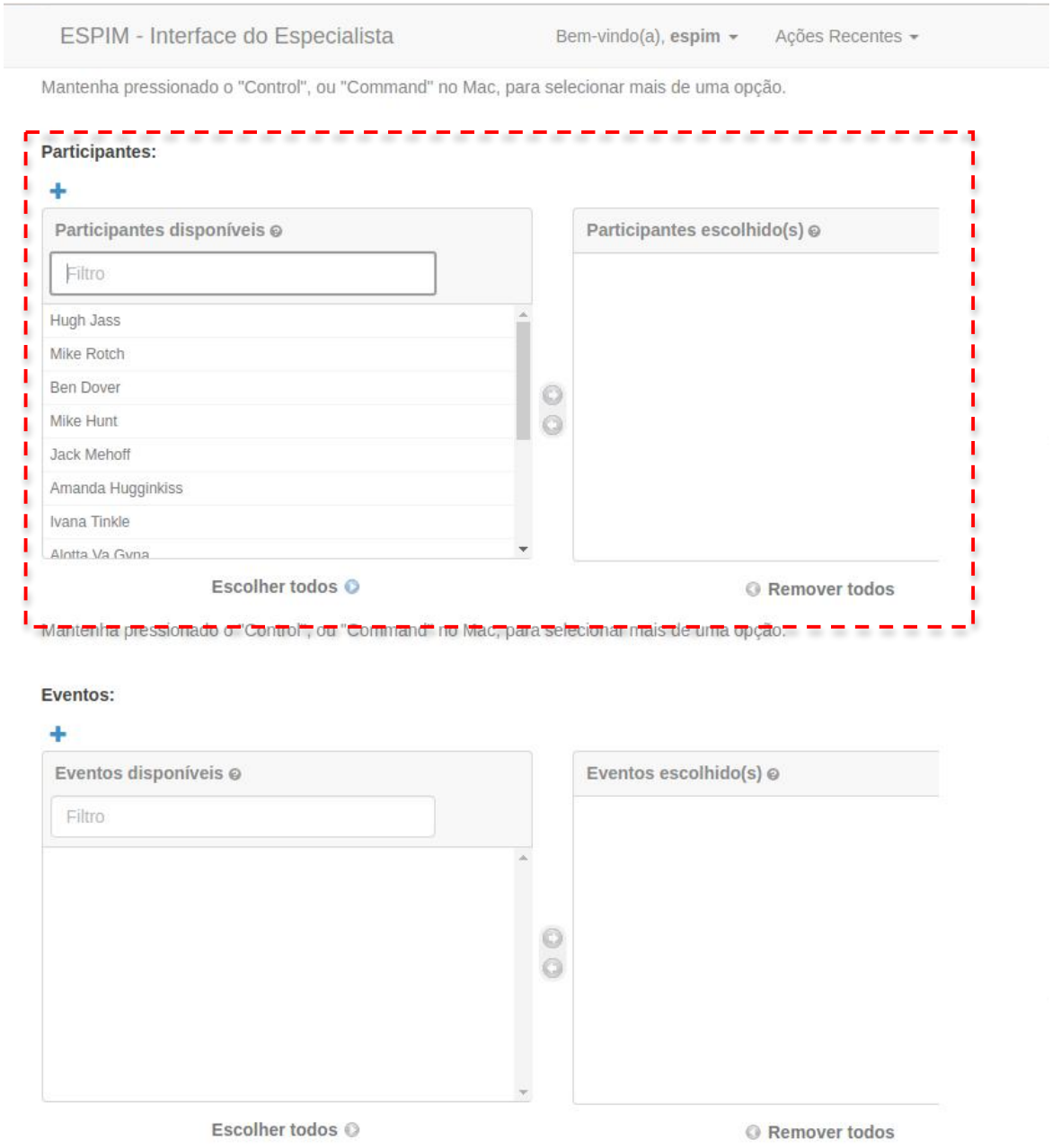


Figura 5.14 a. Tela de seleção de participantes para uma intervenção.

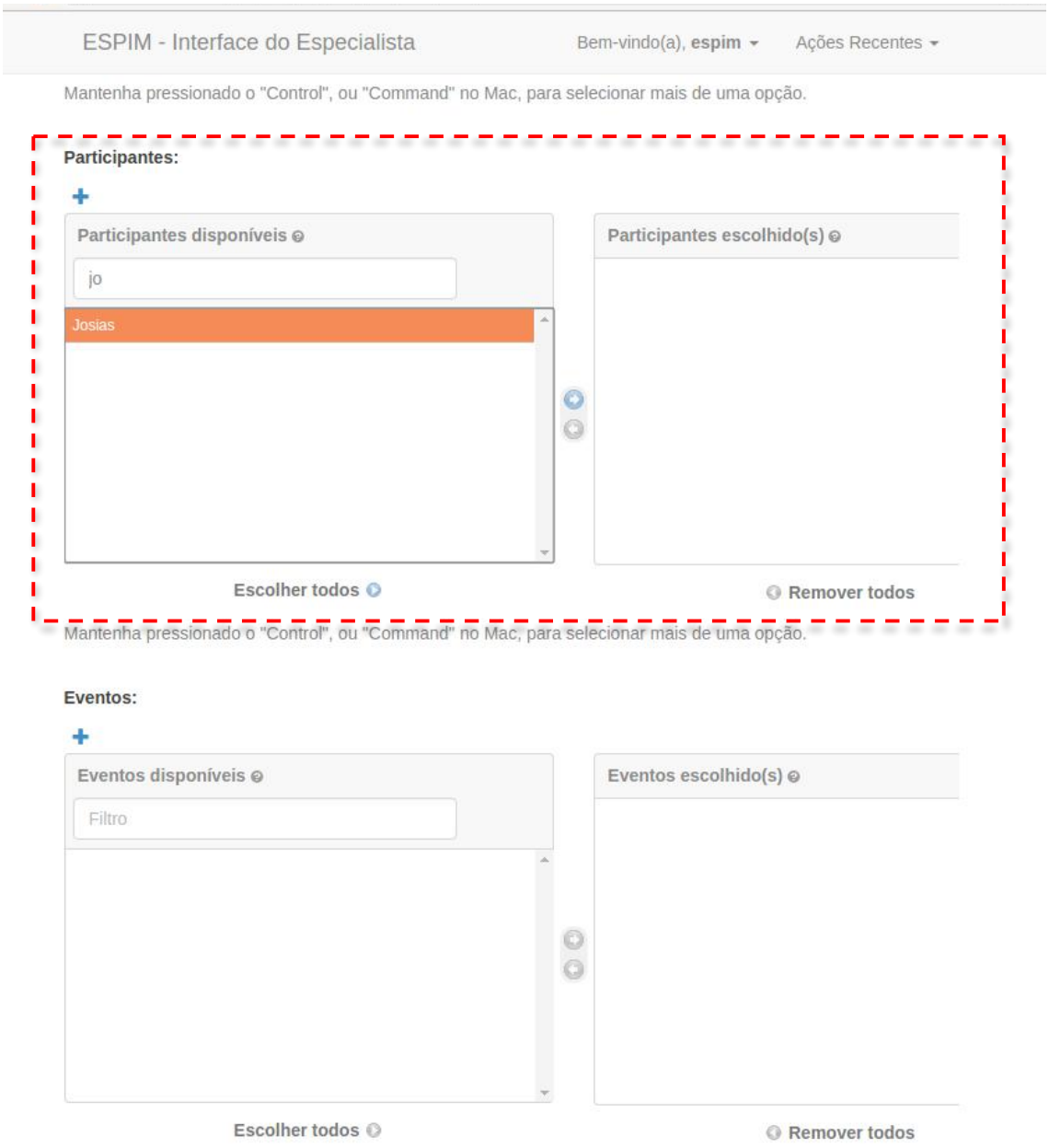


Figura 5.14 b. Tela de seleção de participantes para uma intervenção.

Ao planejar uma coleta de dados o especialista também pode programar eventos (tela ilustrada na Figura 5.15) e associá-los a uma coleta de dados.

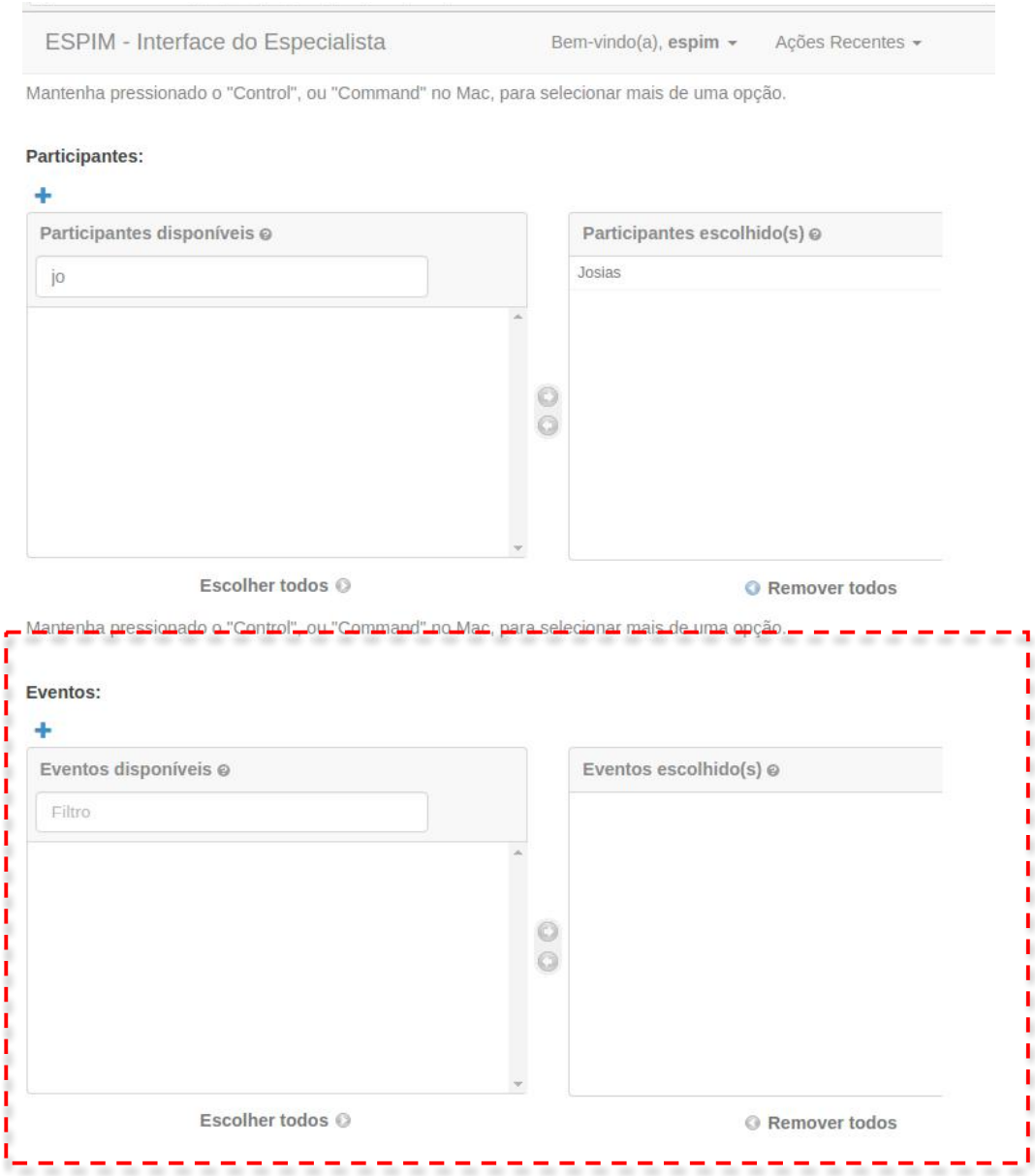


Figura 5.15. Telas de criação de eventos.

Entre as possibilidades de eventos o especialista pode programar questões de múltipla escolha com uma opção, questões com múltiplas opções de escolha, questões abertas, mensagens ou tarefas, conforme ilustrado no trecho em destaque na Figura 5.16.

Figura 5.16. Possibilidades de eventos disponíveis ao especialista.

A Figura 5.17 ilustra a criação de uma questão de múltipla escolha (*radio button*).

Figura 5.17. Criando eventos com questões de múltipla escolha.

As Figuras 5.18a e 5.18b ilustram um exemplo de evento do tipo múltipla escolha e a criação de um evento do tipo Tarefa associado ao programa interventivo em elaboração.

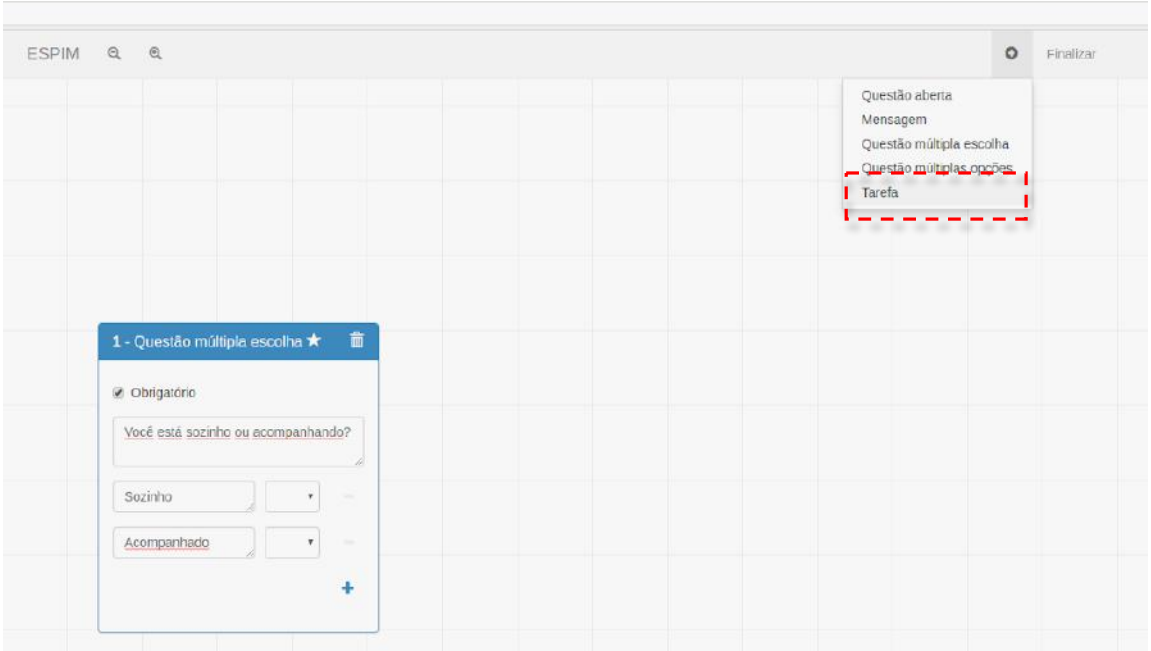


Figura 5.18a. Criando eventos do tipo Tarefa.

Figura 5.18b. Criando eventos do tipo Tarefa.

A caixa com destaque em azul mais escuro na Figura 5.18b representa o primeiro evento do programa interventivo e o ícone em estrela permite alteração desse primeiro evento ao ser clicado.

A Figura 5.19 ilustra a criação da tarefa iniciada na Figura 5.18 e a criação de um novo evento, também do tipo Tarefa, associado ao programa em elaboração. Caso a tarefa seja uma aplicação externa ao ESPIM, um campo específico é disponibilizado para carregar a URL onde a aplicação/aplicativo em questão está disponibilizada.

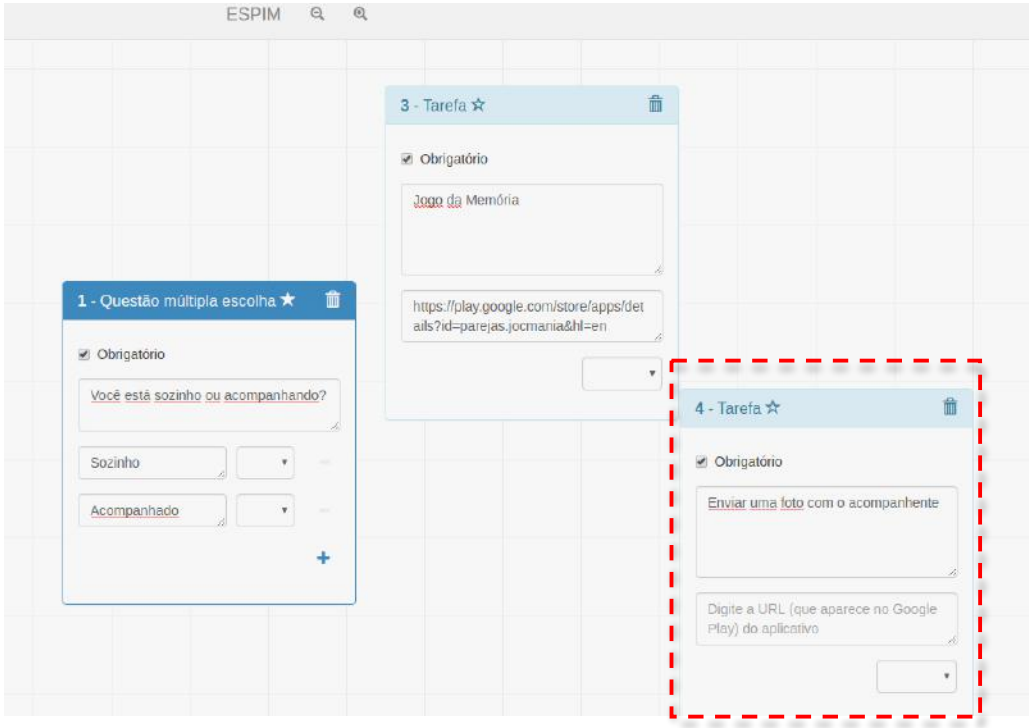


Figura 5.19. Instanciação de eventos do tipo Tarefa.

Após a criação dos eventos, ao escolher uma das opções numéricas na caixa de seleção o especialista faz uma ligação entre o evento e o seu sucessor, conforme o fluxo de tarefas desejado pelo mesmo para o participante. Veja destaque na Figura 5.20.

A seta tracejada indica o caminho apontado. Também por meio da caixa de seleção o especialista pode informar que o evento que ele está manipulando é o evento final.

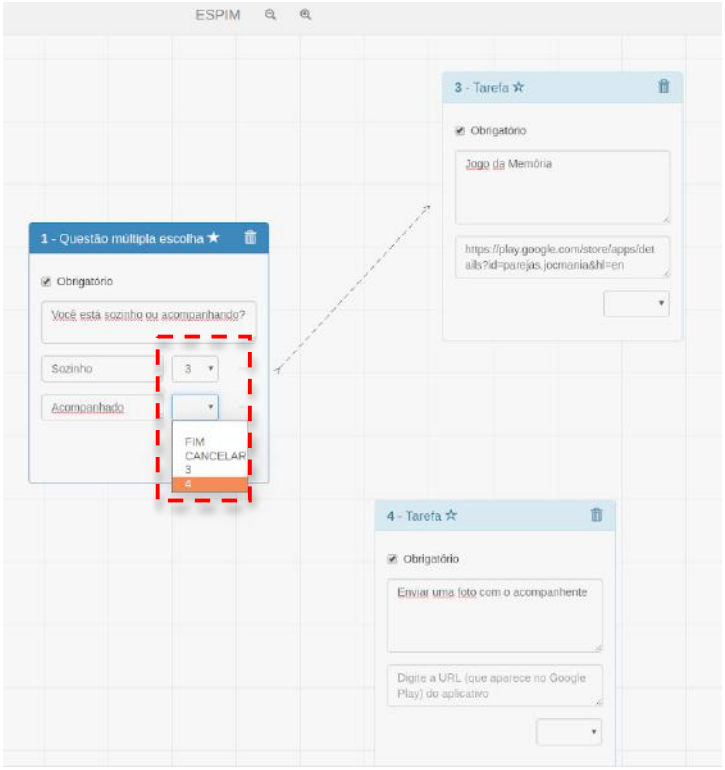


Figura 5.20. Fazendo a ligação entre eventos.

O especialista também pode prever e programar uma questão de múltipla escolha tipo Likert, conforme ilustrado na Figura 5.21.

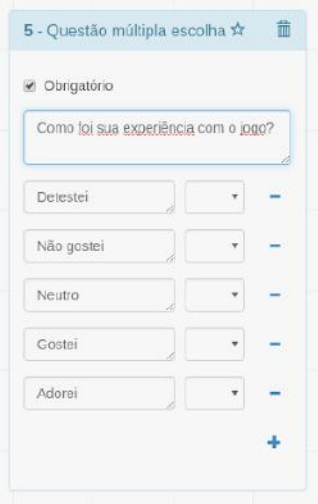


Figura 5.21. Criando eventos do tipo Questão de múltipla escolha.

Eventos de múltiplas opções (*checkbox*) também podem ser criados, conforme ilustrado na Figura 5.22.

Figura 5.22. Criando eventos com Questões de múltiplas opções.

A Figura 5.23 ilustra a criação de eventos do tipo Questões abertas, em que o participante deve digitar um texto como resposta e, em versões futuras, gravar um áudio.

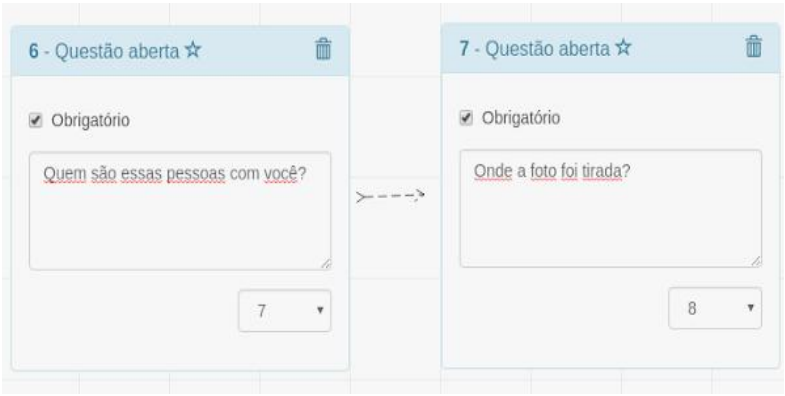


Figura 5.23. Criando eventos com Questões abertas.

Ao finalizar o programa interativo e o fluxo de eventos a ser realizado pelo participante, o especialista pode criar um evento do tipo Mensagem com a marcação de fim desse fluxo, acompanhada de uma mensagem a esse participante, conforme ilustrado na Figura 5.24.

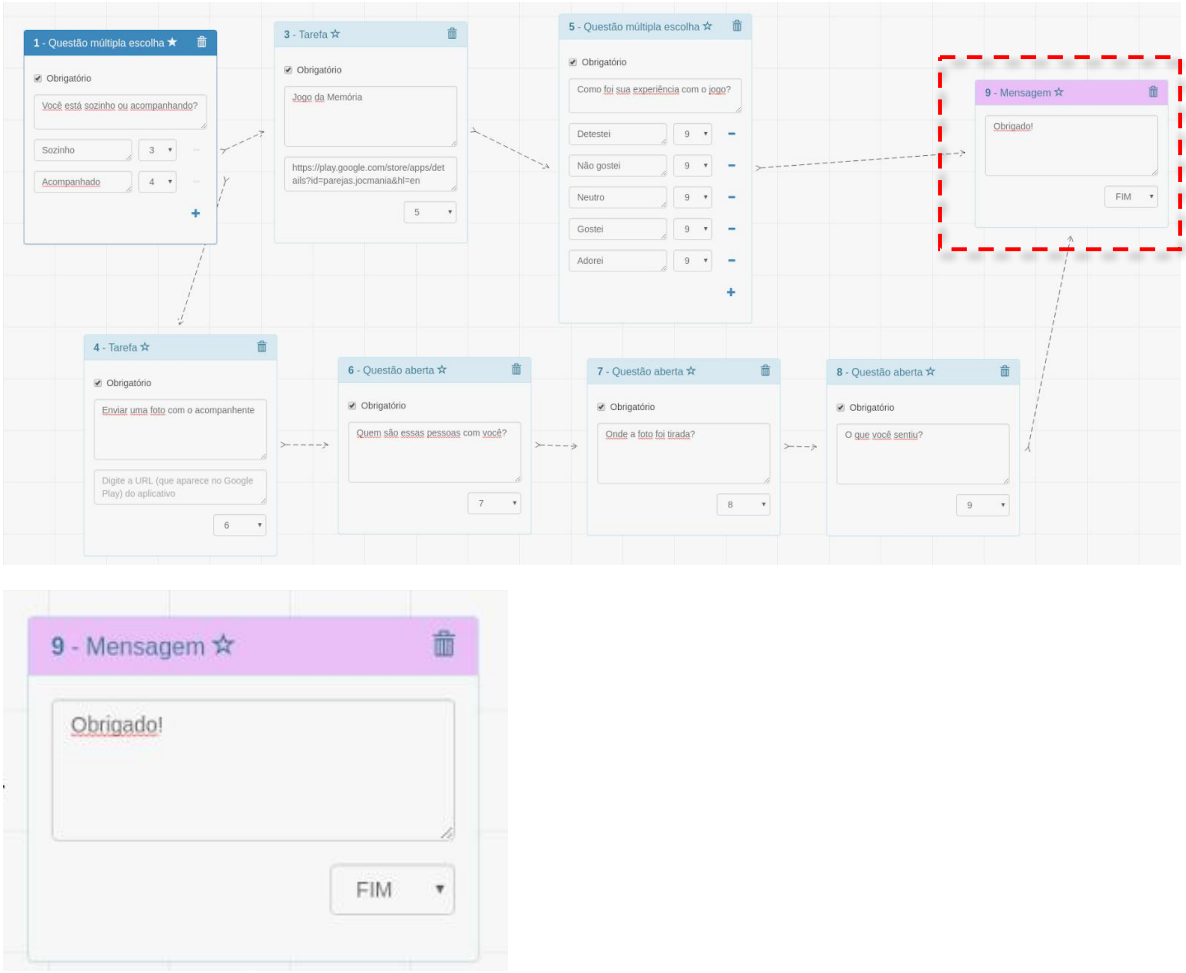


Figura 5.24. Finalizando o fluxo de eventos para um determinado programa com uma mensagem.

A versão atual do sistema ESPIM foi concebida com o apoio de profissionais de diferentes áreas de atuação, entre elas, computação e saúde. Além das funcionalidades implementadas de protótipos anteriores, como o *SmartESM*, outros requisitos foram coletados junto aos profissionais de área e, após ponderações, deverão fazer parte das próximas versões do sistema.

Outros trabalhos existentes na literatura também serviram de inspiração para a concepção de novas funcionalidades do ESPIM.

A Tabela 5.2 exibe uma comparação entre o ESPIM e funcionalidades já implementadas por outros sistema semelhantes.

Tabela 5.2. Comparação entre o sistema ESPIM e outros sistemas para coleta de dados.

	ESPIM	MyExperience (FROELICH, et al., 2007)	PsychLog (GAGGIOLI et al., 2013)	AndWellness (HICKS et al., 2010)	StudentLife (WANG et al., 2014)	Ohmage (TANGMU-NARUNKIT et al., 2015)	PACO (BAXTER et al.., 2015)
Autorrelato via texto (aberto/fechado)	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Autorrelato via multimídia	Previsto: foto, vídeo, áudio.	Não	Não	Foto	Não	Foto	Foto
Coleta de dados via sensores	Previsto	Sim	Sim	Sim	Sim	Sim	Sim
Integração com outros dispositivos	Previsto	Não	Sim	Não	Não	Sim	Não
Recursos de autoria para pesquisador/ especialista	Web	XML	TXT	GUI	Não	GUI	Web
Recursos de visualização de informações para profissional/ paciente	Previsto	Não	Básicos	Avançados	Não	Avançados	Básicos
Recursos de sistemas de recomendação	Não	Não	Sim	Sim	Não	Não	Não
Recursos de bate-papo ao vivo	Previsto	Não	Não	Não	Não	Não	Não

As versões do ESPIM serão submetidas à avaliações com usuários alvo para que seja observado aspectos relacionados à experiência de uso e à usabilidade da sua interface. Para esta etapa especialistas da computação, da saúde e da educação especial, bem como pacientes e cuidadores serão convidados a interagirem com o sistema e programas interventivos criados, e fornecerem *feedback* sobre o sistema.

5.6. Agradecimentos

Agradecemos o apoio financeiro das agências de fomento à pesquisa - FAPESP, CNPq, CAPES - que têm possibilitado o desenvolvimento deste projeto.

Referências Bibliográficas

Araújo, R. B. (2003). Computação ubíqua: Princípios, tecnologias e desafios. In: XXI Simpósio Brasileiro de Redes de Computadores. 2003. p. 11-13.

Barrett, L. F., & Barrett, D. J. (2001). An introduction to computerized experience sampling in psychology. Social Science Computer Review, 19(2): 175–185.

Baxter, K. K., Avrekh, A., & Evans, B. (2015). Using experience sampling methodology to collect deep data about your users. In Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems, CHI EA'15, pages 2489–2490, New York, NY, USA. ACM.

Csikszentmihaly, M. (2000). Beyond boredom and anxiety (2nd ed.). San Francisco: Jossey-Bass.

Csikszentmihaly, M., Larson, R., & Prescott, S. (1977). The ecology of adolescent activity and experience. Journal of Youth and Adolescence, 6(3): 281–294.

Csikszentmihaly, M., & Larson, R. (1987). Validity and reliability of the experience-sampling method. The Journal of Nervous and Mental Disease, 175(9), 526–536.

De Rose, J., Souza, D., Rossito, A., & De Rose, T. (1989). Equivalência de estímulos e generalização na aquisição de leitura após história de fracasso escolar. Psicologia: Teoria e Pesquisa, 5(2): 325–346.

De Souza, D. G., De Rose, J. C., Faleiros, T. C., Bortoloti, R., Hanna, E. S., & Mcilvane, W. J. (2009). Teaching generative reading via recombination of minimal textual units: A legacy of verbal behavior to children in Brazil. International Journal of Psychology and Psychological Therapy, 9(1): 19.

Floridou, G. A. & Mullensiefen, D. (2015). Environmental and mental conditions predicting the experience of involuntary musical imagery: An experience sampling method study. Consciousness and Cognition, 33(0): 472 – 486.

Froehlich, J., Chen, M. Y., Consolvo, S., Harrison, B., & Landay, J. A. (2007). Myexperience: A system for in situ tracing and capturing of user feedback on mobile phones. In Proceedings of the 5th International Conference on Mobile Systems, Applications and Services, MobiSys'07, pages 57–70, New York, NY, USA.

Fuller-Tyszkiewicz, M., Mccabe, M., Skouteris, H., Richardson, B., Nihill, K., Watson, B., & Solomon, D. (2015). Does body satisfaction influence self-esteem in adolescents' daily lives? An experience sampling study. Journal of adolescence, 45: 11–19.

Gaggioli, A., Pioggia, G., Tartarisco, G., Baldus, G., Corda, D., Cipresso, P., & Riva, G. (2013). A mobile data collection platform for mental health research. Personal and Ubiquitous Computing, 17(2): 241–251.

Hartley, S., Haddock, G., Vasconcelos, Sa, D., Emsley, R., & Barrawclough, C. (2014). An experience sampling study of worry and rumination in psychosis. Psychological medicine, 44(08): 1605–1614.

Hektner, J. M., Schmidt, J. A., & Csikszentmihaly, M. (2007). Experience sampling method: Measuring the quality of everyday life. Sage.

Hicks, J., Ramanathan, N., Kim, D., Monibi, M., Selsky, J., Hansen, M., & Estrin, D. (2010). Andwellness: an open mobile system for activity and experience sampling. In Wireless Health 2010, pages 34–43. ACM.

Hofmann, W., Adriaanse, M., Vohs, K. D., & Baumeister, R. F. (2014). Dieting and the self-control of eating in everyday environments: An experience sampling study. British journal of health psychology, 19(3): 523–539.

Kackar, H. Z., Shumow, L., Schmidt, J. A., & Grzetich, J. (2011). Age and gender differences in adolescents' homework experiences. Journal of Applied Developmental Psychology, 32(2): 70 – 77.

Kapoor, A. & Horvitz, E. (2008). Experience sampling for building predictive user models: A comparative study. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI'08, pp. 657–666, New York, NY, USA. ACM.

Keller, F. S. (1968). Good-bye, teacher... Journal of Applied Behavior Analysis, 1(1): 79.

Kramer, I., Simons, C. J., Hartmann, J. A., Menne-Lothmann, C., Viechtbauer, W., Peeters, F., Schruers, K., Bemmel, A. L., Myin-Germeys, I., Delespaul, P., *et al.* (2014). A therapeutic application of the experience sampling method in the treatment of depression: a randomized controlled trial. World Psychiatry, 13(1): 68–77.

Lowdermilk, T. (2013). Design Centrado no Usuário. Editora: Novatec Editora, 184p.

Lyytinen, K; Yoo. & Y. (2002). Issues and Challenges in Ubiquitous Computing. Communications Of The ACM, p.63-65, 2002.

Lopez, V., Ahumada, L., Galdames, S., & Madrid, R. (2012). School principals at their lonely work: ´ Recording workday practices through {ESM} logs. Computers & Education, 58(1): 413 – 422.

Mcshane, J. M. & Zirkel, S. (2008). Dissociation in the binge–purge cycle of bulimia nervosa. Journal of Trauma & Dissociation, 9(4): 463–479.

Maes, I. H., Delespaul, P. A., Peters, M. L., White, M. P., Van Horn, Y., Schruers, K., Anteunis, L., & Joore, M. (2015). Measuring health-related quality of life by experiences: The experience sampling method. Value in Health, 18(1): 44–51.

Mujagic, Z., Leue, C., Vork, L., Lousberg, R., Jonkers, D., Keszthelyi, D., Hesselink, M., Schagen, T., Os, J., Masclee, A., *et al.* (2015). The experience sampling method-a new digital tool for momentary symptom assessment in IBS: an exploratory study. Neurogastroenterology & Motility, 27(9): 1295–1302.

Myllykangas, S. A., Gosselink, C. A., Foose, A. K., & Gaede, D. B. (2002). Meaningful activity in older adults: Being in flow. World Leisure Journal, 44(3): 24–34.

Nielsen, K. & Cleal, B. (2011). Under which conditions do middle managers exhibit transformational leadership behaviors? — An experience sampling method study on the predictors of transformational leadership behaviors. The Leadership Quarterly, 22(2): 344 – 352.

Pimentel, M. G., Rocha, A. C., Cunha, B. C. R., Orlando, A. F., Machado Neto, O., Viel, C., Antunes, E., & Zaine, I. (2016). Apoio ao envelhecimento no lugar por meio de amostragem de experiências e de intervenção programada. Revista da Faculdade de Medicina de Ribeirão Preto e do Hospital das Clínicas da Universidade de São Paulo, 49 (Supl 2), 11.

Riddle, M. M. D. & Arnold, D. M. V. (2007). The day experience method: A resource kit.

Rieh, S. Y., Kim, Y. M., Yang, J. Y., & St. Jean, B. (2010). A diary study of credibility assessment in everyday life information activities on the web: Preliminary findings. Proceedings of the American Society for Information Science and Technology, 47(1): 1–10.

Satyanarayanan, M. (2001). Pervasive Computing: Vision and Challenges. Personal Communication, IEEE, v. 8, n.4, p.10-17, 2001.

Schuler, D., & Namioka, A. (1993). Participatory Design: Principles and Practices. Hillsdale, NJ: Lawrence Erlbaum Associates.

Skinner, B. F. (1974). About behaviorism. New York: Appleton-Century-Crofts.

Skinner, B. F. (1953). Science and human behavior. New York: Simon and Schuster.

Skinner, B. F. (1958). Reinforcement today. American Psychologist, 13(3): 94.

Skinner, B. F. (1961). Teaching machines. Scientific American, 205(11): 90–102.

Skinner, B. F. (1972). Humanism and behaviorism. The Humanist, 32: 18–20.

Swendeman, D., Comulada, W., Ramanathan, N., Lazar, M., & Estrin, D. (2015). Reliability and validity of daily self-monitoring by smartphone application for health-related quality-of-life, antiretroviral adherence, substance use, and sexual behaviors among people living with HIV. AIDS and Behavior, 19(2): 330–340.

Tangmunarunkit, H., Hsieh, C., Jenkins, J., Ketcham, C., Selsky, J., & Alquaddoomi, F. (2015). Ohmage: A general and extensible end-to-end participatory sensing platform. ACM Transactions on Intelligent Systems and Technology (TIST), 6(3), 38.

Udachina, A., Varese, F., Myin-Germeys, I., & Bentall, R. P. (2014). The role of experiential avoidance in paranoid delusions: An experience sampling study. British Journal of Clinical Psychology, 53(4): 422– 432.

Wang, R., Chen, F., Chen, Z., Li, T., Harari, G., Tignor, S., Zhou, X., Ben-Zeev, D., & Campbell, A. T. (2014). Studentlife: Assessing mental health, academic performance and behavioral trends of college students using smartphones. In Proceedings of the 2014. ACM International Joint Conference on Pervasive and Ubiquitous Computing, UbiComp'14, pages 3–14, New York, NY, USA.

Wang, Z. & Tchernev, J. M. (2012). The "myth" of media multitasking: Reciprocal dynamics of media multitasking, personal needs, and gratifications. Journal of Communication, 62(3): 493–513.

Wang, Z., Tchernev, J. M., & Solloway, T. (2012). A dynamic longitudinal examination of social media use, needs, and gratifications among college students. Computers in Human Behavior, 28(5): 1829 – 1839.

Weiser, M. (1991). The computer for the 21st century. Scientific american, 265(3), 94–104.

Yin, J., Yang, Q., & Pan, J. J. (2008). Sensor-based abnormal human-activity detection. Knowledge and Data Engineering, IEEE Transactions on, 20(8): 1082–1090.

Biografia Resumida dos Autores

Isabela Zaine - Psicóloga e Doutora em Psicologia pela UFSCar, Especialista em Psicoterapia Analítico-Comportamental, Docente de Ensino Superior do curso de Psicologia. Atua em análise do comportamento aplicada e experimental, programação de ensino, processos básicos de aprendizagem e tecnologia de ensino. Atualmente atua como pós-doutoranda na Universidade de São Paulo - Instituto de Ciências Matemáticas e de Computação, realizando pesquisas relacionadas a tecnologias interativas aplicadas a contextos educacionais e de saúde.

Kamila R. H. Rodrigues - Professora de Ensino Superior, Doutora em Ciência da Computação. Tem interesse nas áreas de Sistemas Multimídia e Interação Humano-Computador. Atua em pesquisas com o foco em acessibilidade e respostas emocionais de usuários, bem como em suas experiências durante a interação com sistemas multimídia interativos. Atualmente é pós-doutoranda na Universidade de São Paulo - Instituto de Ciências Matemáticas e de Computação, realizando pesquisas com documentos interativos para coleta especializada de experiências cotidianas de indivíduos.

Bruna C. R. da Cunha - Doutoranda em Ciências da Computação e Matemática Computacional pela Universidade de São Paulo. É Bacharel em Ciências de Computação pela Universidade de São Paulo (2009) e Mestre em Ciências da Computação e Matemática Computacional pela Universidade de São Paulo (2014). Realiza pesquisas nas áreas de Interação Usuário-Computador, User Experience, Tecnologia Assistiva e Computação Ubíqua.

Yuri Nehase Zuliani Goulart Magagnatto - Mestrando em Ciências da Computação e Matemática Computacional ICMC/USP. Tecnólogo em Sistemas para Internet na Faculdade de Tecnologia de Jaú (FATEC JAHU). Desenvolve pesquisas sobre utilização de aplicativos móveis na área da saúde, mais precisamente para a área de terapia ocupacional. Trabalhou como desenvolvedor web em empresas de e-commerce. Possui formação técnica em informática.

Alex F. Orlando - Graduado em Ciência da Computação e Mestre em Engenharia de Software pelo PPG-CC, ambos pela UFSCar. Atualmente é doutorando no PPG-CCMC do ICMC-USP, gerente executivo do Projeto SACI@Ipê, gerente do projeto GEIC e empresário na bluedotsoft. Tem experiência na área de Ciência da Computação, com ênfase em Software Básico, atuando principalmente nos seguintes temas: tecnologia, tendências, mercado, informação e multimídia.

Caio C. Viel - Aluno de Doutorado no programa de Computação e Matemátia Computação na Universidade de São Paulo - Instituto de Ciências Matemáticas e de Computação. Possui graduação em Engenharia da Computação e mestrado em Computação pela Universidade Federal de São Carlos. Realiza pesquisas nas áreas de computação ubíqua e em sistemas multimídia interativos com foco em aplicações de Captura e Acesso.

André Luiz Carlomagno Rocha - Mestre em Ciência da Computação pela Universidade Federal da Bahia - UFBA (2014), atuando na área de Sistemas Web. Bacharel em Ciência da Computação pela Universidade Salvador - UNIFACS (2010), com ênfase em Sistemas de Computação e Redes de Computadores. Está envolvido também com estudos na área de coleta de dados usando métodos de pesquisa para capturar, por meio de estratégias usando dispositivos móveis e vestíveis, a experiência do usuário em tempo-real, bem como, o contexto em que ele se encontra.

Maria da Graça C. Pimentel - Professora Titular da Universidade de São Paulo, Doutora em Ciências da Computação pela University of Kent at Canterbury (1994), e Livre-docente pela Universidade de São Paulo (2001). Atua nas áreas de Engenharia de Documentos, Computação Ubíqua, Web Semântica, Interação Usuário-computador e CSCW, com pesquisas aplicadas à Tecnologias Assistivas, TV Digital e à Educação. Coordena projetos de pesquisa financiados por CNPq, CAPES e FAPESP, e coordenou projetos financiados pela FAPESP, CNPq, CAPES, FINEP e MCT/CITIC.

Capítulo 6

Introdução Prática à Internet das Coisas: Prática utilizando Arduíno e Node.js

Cintia Carvalho Oliveira, Daniele Carvalho Oliveira, João Carlos Gonçalves, Júlio Toshio Kuniwake

Abstract

The Internet of Things brings the idea of "all connected", from television sets that choose the schedule according to the viewer's preferences to cars that select the faster path to the desired point by its driver. With the expansion of Web access not only computing devices such as laptops, personal computers and smartphones but also equipment such refrigerators, television, among others, the possibilities of communication and automation expand and cities, homes and smart environments became one reality. This work aims to explore the potential of this technology by presenting the necessary steps to develop a simple application that uses the Arduino platform and framework Node.js.

Resumo

A Internet das Coisas traz a ideia de "tudo conectado", desde aparelhos de televisão que escolhem a programação de acordo com a preferência do espectador a carros que selecionam o trajeto mais rápido até o ponto desejado por seu condutor. Com a expansão do acesso à Web não só de dispositivos computacionais como notebooks, computadores pessoais e smartphones mas também de equipamentos como geladeiras, televisão, entre outros, as possibilidades de comunicação e automatização se ampliam e cidades, casas e ambientes inteligentes passam a ser uma realidade. Este trabalho tem como objetivo explorar as potencialidades desta tecnologia apresentando os passos necessários para o desenvolvimento de uma aplicação simples que utiliza a plataforma Arduino e o framework Node.js.

6.1. Introdução

Internet das Coisas (IoT – *Internet of Things*) é uma estrutura de rede global e dinâmica que possibilita a autoconfiguração, baseando-se em protocolos padronizados de

comunicação, onde seus componentes físicos e virtuais possuem identidades, assim usando interfaces inteligentes ligadas a Internet [Atzori et al. 2010].

Na Internet das Coisas seus componentes são capazes de interagir entre si, deste modo trocando informações em diferentes ambientes além de reagir automaticamente aos eventos do mundo real, deste modo não havendo exatamente uma intervenção direta do ser humano em sua infraestrutura. A IoT é uma tecnologia que tem como objetivo conectar diferentes softwares, dispositivos eletrônicos, máquinas industriais, "coisas" no termo geral, tudo fazendo uso de métodos dinâmicos, além do uso de sensores, nanotecnologia, *wireless*, componentes conectados via Internet.

Internet of Things exige uma forma de segurança feita sob medida, pois ao contrário de computadores tradicionais, seus elementos são constituídos de pouca capacidade de energia, processamento e memória [Atzori et al. 2010], assim realizando seu trabalho de forma colaborativa. Podemos visualizar a IoT como uma tecnologia "maciça", uma tecnologia que engloba diferentes conceitos para se tornar único, a principalmente três componentes que são combinados para poder haver uma aplicação IoT, que são:

Dispositivos: Os dispositivos são itens que podem ter diferentes tamanhos, grandes a minúsculos, porém não é o tamanho do dispositivo que faz a diferença e sim como esses dispositivos são equipados para favorecer a comunicação entre seus diferentes componentes, tais dispositivos podem ser equipados com sensores, chips, antenas, dentre outros.

Redes de Comunicação: A comunicação via Internet que conhecemos pode ser usada para a IoT, no entanto as redes atuais possuem alcance limitado, obrigando que certas aplicações façam uso de redes móveis, tais como 3G e 4G. Porém essas redes móveis têm como foco aparelhos como *laptops*, *smartphones* e *tablets*. Em um ambiente na qual a IoT é amplamente implementada, haverá chips, sensores, dispositivos conectados por toda área, obrigando assim uma otimização das redes móveis atuais. É onde entra a 5G, um recurso que melhorará claramente a Internet móvel atual, evitando gargalos na rede e proporcionando maior velocidade do mesmo.

Sistema de Controle: Não adianta ter dispositivos que se conectam uns com os outros se não houver um controle adequado de seus dados, tais informações precisam ser tratadas, sejam em serviço nas nuvens ou grandes servidores, um sistema de controle é de grande importância quando se trata de IoT pois é através dele que será possível utilizar de forma adequada tudo o que a Internet das Coisas tem a oferecer.

6.1.1. Visão e Motivação

A IoT é um contexto que tem como objetivo desenvolver um caminho entre acontecimento do mundo real e as representações no ambiente digital, com o intuito de integrar ambos os paradigmas, obtendo assim benefícios ao seu uso. Atualmente existe muitos meios para realizar a captura dessas "coisas", desde códigos simples até os mais sofisticados e complexas tecnologias [Atzori et al. 2010].

Redes de sensores são atualmente muito estudados quando se trata de IoT, pois são classificadas como *Objetos Inteligentes,* ou seja, efetuam comunicações e operações computacionais entre si. Sensores é uma área muito estudada fora de ambientes acadêmicos, havendo aplicações em diversas áreas, essas redes de sensores são

agrupadas por um conjunto de nós, onde tais nós possuem funcionalidades diversas como medir temperaturas, ruídos, humidade, luminosidade, entre outros. Possuem como propósito coletar informações importantes e encaminhar para um ou mais nós específicos, que chamamos de *estações-base*, que funciona como ponte entre a rede de sensores com o ambiente externo, em casos específicos alguns nós possuem poder de processamento, assim sendo possível efetuar operações em cima de informações coletadas na rede.

6.1.2. Aplicações Existentes

Com o grande crescimento da Internet aliado a um eficiente processo tecnológico, é possível aplicar a IoT em diversas áreas, sejam em indústrias, serviços, produtos, entre outros. Uma área de aplicação que mais favorece é a melhoria na qualidade de vida das pessoas, usando a IoT para desenvolver edifícios e casas inteligentes [Vermesan e Friess, 2013].

No segmento de casas inteligentes devemos ressaltar um conjunto de controle ou gestão de eficiência em consumo de energia, onde podemos controlar aqueles aparelhos que estão consumindo energia em excesso, efetuando uma gestão dos aparelhos, sejam em ar-condicionado, termostatos ou segurança. Um exemplo seria um termostato que tem a capacidade de regular a temperatura de um ambiente de acordo com a presença de alguma pessoa no local, ajustando a temperatura de acordo com o critério definido pelo usuário [Vermesan e Friess, 2013].

Já na área de saúde podemos destacar aplicações que monitoram o estado de pacientes dentro de um hospital, emitindo algum sinal para os médicos quando algum paciente tem queda de pressão, ritmo cardíaco alterado, algumas anomalias no estado de saúde do paciente.

Em áreas agrícolas a IoT possibilita um monitoramento de suas lavouras sejam com o monitoramento do solo, umidade no ambiente, condições climáticas, detectando áreas que necessita de irrigação, detecção de níveis de agrotóxicos usados na lavoura, deste modo melhorando a qualidade e produção daquele produto, além de gerar menor custo ao produtor [Vermesan e Friess, 2013].

Em ambientes externos a IoT também pode contribuir, efetuando a prevenção de certos problemas, como monitoramento de combustão de gases, visualizar pontos com maior risco de incêndios, controlar o nível de emissão de poluição no ar, efetuar a prevenção de deslizamento de terra através do nível de humidade no solo, antecipar a detecção de terremotos através de sensores, [Vermesan e Friess, 2013].

A Internet das Coisas possibilita uma coleção de vantagens em sua utilização podendo ser usada em pequenas aplicações ou até mesmo ser aplicadas no conceito de cidades inteligentes.

6.1.3. Arquitetura da Internet das Coisas

A IoT possui um conceito de arquitetura composto por cinco Camadas: negócio, aplicação, processamento, transporte e percepção. O autor Miao et al. (2010) destaca

que a Internet das Coisas se difere da Internet, pois na IoT sua estrutura necessita ser gerenciada e operada assim, pensando nisso, a arquitetura da IoT foi desenvolvida com base no TMN *Telecommunications Management Network*.

Serão listadas e explicadas as respectivas camadas da arquitetura da Internet das coisas.

- **Camada de Negócio**: está camada atua como um gerente da IoT pois é responsável por efetuar questões administrativas com privacidade de usuários.
- **Camada de Aplicação:** está camada fica a cargo de desenvolver aplicações baseadas em informações analisadas em camadas anteriores
- **Camada de Processamento:** está camada fica a cargo do armazenamento, análise e processamento de dados encaminhados através da camada de transporte.
- **Camada de Transporte:** está camada fica a cargo de transmitir dados processados "informações", recebidas por meios de conexões como, *Wi-Fi*, infravermelho, 3G, rádio frequências, entre outros. Protocolos de endereçamento também são encontrados nesta camada.
- **Camada de Percepção:** está camada fica a cargo de efetuar a coleta de dados, tais informações são adquiridas através de atuadores, sensores, etc. Também é nessa camada que informações vão ser convertidas em sinais para ser transmitida na rede.

6.1.4. Computação nas Nuvens e a IoT

Computação em nuvem (*Cloud Computing*) idealizou um modelo de computação distribuída onde se tem recursos computacionais (hardware, plataformas de desenvolvimento, armazenamento e comunicação), que são disponibilizados e virtualizados como serviços reconfiguráveis. Muitas vezes são recursos pagos para poderem ser utilizados e geralmente por grandes centros de dados na internet, deste modo oferecendo suporte para os mais diversos serviços de armazenamento e compartilhamento de dados, independentemente do local que se encontra é possível acessar seus dados, bastando estar conectado à Internet. [Armburst et al., 2010]

O conceito em nuvem tem como objetivo ser global e prover serviços variados, que podem ir desde um usuário final que hospeda pequenos documentos de dados pessoais na Internet, até grandes empresas que terceirizam toda a infraestrutura da TI (*Tecnologia da Informação*) para outras empresas.

Computação em Nuvem é uma tecnologia que disponibiliza recursos computacionais virtualizados através da Internet, utilizando computadores e servidores que compartilham informações. Tal estrutura é capaz de integrar grandes redes a diversos tipos de sensores, a *Cloud Computing* apresenta algumas soluções para problemas denominados de *Sensor-Cloud* (*nuvem de sensores virtuais*), onde essa integração visa solucionar problemas de distribuição e armazenamento de dados adquiridos através de sensores, desta forma permitindo processar informações de várias Redes de Sensores Sem Fio (RSSF), melhorando assim o compartilhamento de informações em grande escala. [Ansari et al., 2013]

Uma *Sensor-Cloud* motiva a necessidade de uma estrutura de RSSF, com o intuito de melhorar as operações e manutenções do sistema, permitindo fluxos de comunicação direta no meio de dispositivos heterogêneos.

Redes de Sensores Sem Fio é visualizada como conceitos e elementos da IoT, que é baseada na junção digital com o meio externo, desta forma tais tipos de estrutura é vista com uma concepção promissora de cidades inteligentes, onde a computação em nuvem disponibiliza soluções para gestão da *Internet das Coisas*, sendo então uma alavanca para implementação e o bom funcionamento do futuro da IoT.

6.1.5. Exemplos

Nesta sessão são apresentados dois exemplos de aparelhos que fazem uso da IoT para benefício de seus usuários. O primeiro é o Nike FuelBand SE (Figura 6.1) uma pulseira conectada a Internet e através dela é possível registrar atividades físicas do usuário, como: distância percorrida, velocidade alcançada, contador de passos, verificador de calorias gastas, etc.

Figura 6.1. Pulseira Nike conectada a Internet

Outro exemplo é o Nest Learning Thermostat (Figura 6.2), um produto baseado na Internet das Coisas. Este produto é basicamente um termostato inteligente cuja função é aprender quais níveis de temperatura deve estar em determinada hora do dia, ou seja, o termostato vai saber que temperatura deve estar durante a noite, pela manhã, ou de tarde, o produto é controlado através de um smartphone, onde o usuário consegue controlar o termostato através da internet.

Figura 6.2. Termostato Inteligente conectado a Internet

6.2. Arduino

A plataforma Arduino foi criada em 2005 por Massimo Banzi et. al. (2005) com propósito de se tornar uma plataforma livre e de baixo custo para ser utilizada por artistas, designers, hobbistas, para fins educacionais e qualquer pessoa interessada em

criar objetos ou ambientes interativos [Arduino, 2016a] [Perez, 2013]. O Arduino é uma plataforma de prototipagem eletrônica *open-source* que se baseia em hardware e software flexíveis e fáceis de usar. [Arduino, 2016b]

Este tópico apresentará uma visão geral sobre a plataforma Arduino, seus componentes e os primeiros passos para iniciar o desenvolvimento utilizando esta plataforma.

6.2.1. Introdução

Arduino pode ser utilizado para coisas tão simples como piscar um Led, ou algo bem mais elaborado, como interagir com um smartphone entre outros [Boxall, 2013]. É possível ainda utilizar de sensores para sentir o ambiente que o cerca, de forma a interagir com o ambiente, controlar luzes, motores ou outros atuadores. Além disso, é possível criar projetos autônomos utilizando Arduino, ou projetos que se comunicam com computadores para realizar tarefas mais complexas, utilizando softwares específicos. [Arduino, 2016b]

O Arduino é formado por dois componentes, um de hardware e o outro de software. O hardware é uma placa de prototipagem onde são construídos os projetos. A placa do Arduino pode ser construída de forma caseira, adquirindo-se os componentes e montando manualmente, ou a placa pode ser adquirida já montada. O projeto para montagem, que pode ser baixado gratuitamente, está disponível sob licença *open-source*, de forma que é possível adaptá-lo para cada propósito [Arduino, 2016b]. A Figura 6.3 mostra o Arduino utilizado neste texto, o Arduino Leonardo.

Figure 6.3. Arduino Leonardo

O software é uma IDE, a ser instalada no computador para a programação do controlador. A IDE é conhecida como sketch e é onde toda a programação é feita, ou seja, como o controlador se comportará. O upload da programação para o controlador é feito através de comunicação serial. O micro controlador do Arduino é programado com a linguagem de programação Arduino, criada a partir da linguagem Wiring. Já o ambiente de programação é baseado no ambiente Processing. [Souza, 2013]

6.2.2. Instalação

A programação do Arduino é feita através de ambiente de desenvolvimento integrado (IDE) próprio, que pode ser baixado do link: https://www.arduino.cc/en/Main/Software. Neste link é possível fazer o download do arquivo instalador ou do pacote, recomendamos o instalador, pois já permite a instalação de todos os componentes necessários, incluindo os drivers. [Arduino, 2016c]

Ao efetuar o *download* do arquivo, a instalação é simples, basta seguir os passos a seguir:

1. Execute o arquivo baixado e aceite a Licença de Uso (Figura 6.4);
2. Nas Opções de Instalação marque todas as opções (Figura 6.5);
3. Escolha o local de instalação e clique em Instalar (Figuras 6.6 e 6.7);
4. Após a instalação, serão exibidas duas mensagens de segurança do Windows solicitado a instalação de dois drivers, permita que sejam instalados (Figuras 6.8 e 6.9).

Figura 6.4. Licença de Uso

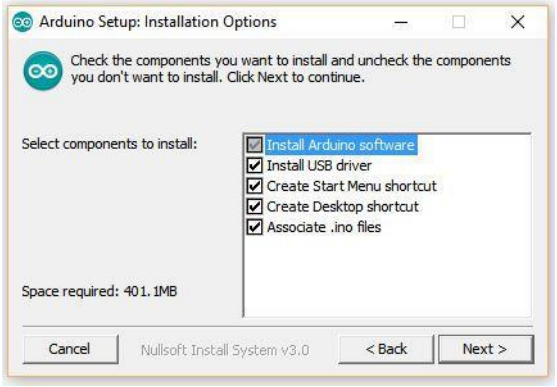


Figura 6.5. Opções de Instalação

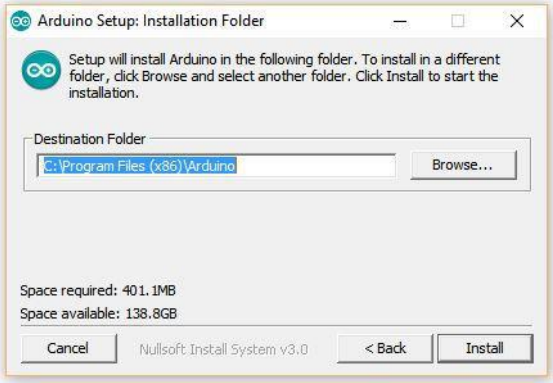


Figura 6.6. Local de Instalação

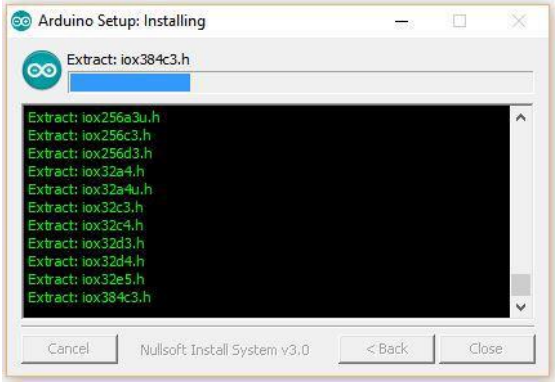


Figura 6.7. Instalação

Figura 6.8. Instalação de Driver

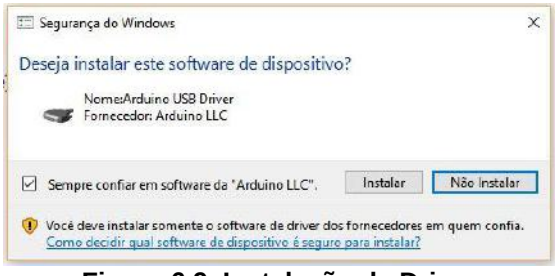


Figura 6.9. Instalação de Driver

Assim que a instalação for concluída, você deve conectar a placa do Arduino no computador e aguardar o reconhecimento e instalação do driver.

6.2.3. Programação no Arduino

A programação do Arduino é feita através do ambiente de desenvolvimento Arduino (Figura 6.10). O código mostrado na Figura 2.9 representa o mínimo para que um programa possa ser compilado [Robot@escola, 2016]:

- O "void setup()" é utilizado para a inicialização das variáveis, modos de pinas, onde é definida a velocidade de transmissão serial, dentre outras funções.
- O "void loop()" é onde se cria um ciclo de repetição para as instruções que devem ser executadas repetidamente.

Além dos dois elementos principais previamente destacados, o código do Arduino é constituído de mais 3 elementos:

- As bibliotecas, incluídas no início do programa, são constituídas de funções pré-feitas pela comunidade de programadores ou pelos criadores da plataforma.
- As variáveis são declaradas de forma global, após a importação das bibliotecas.
- As funções, criadas pelo programador, seguem às declarações de variáveis.

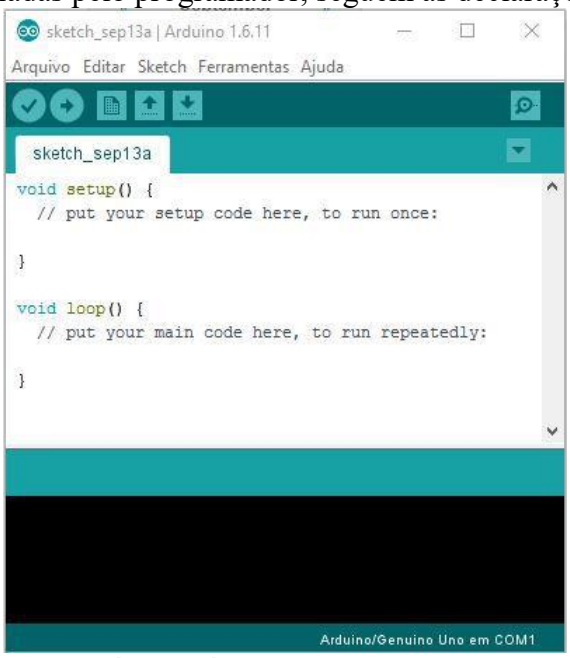


Figura 6.10. Código básico de programação do Arduino

Para carregar e iniciar a programação no Arduino, basta clicar no botão upload, no Arduino IDE. O software do Arduino comanda automaticamente um reset na placa, lançando um *bootloader* – que é responsável por receber, armazenar, e iniciar o novo sketch. [Aloi, 2013]

Como exemplo, o código mostrado na Figura 6.11, controla o Led do pino D13 da placa do Arduino.

Figura 6.11. Código para controle do LED do pino D13 (Laboratório de Garagem, 2014)

6.2.4. Instalação e uso do módulo Ethernet Shield

O módulo Ethernet Shield é utilizado para conectar o Arduino a uma rede Ethernet, permitindo criar projetos que envolvam o envio e o recebimento de informações através da rede ou da internet. A Instalação do módulo é feita de forma simples, basta conectar o módulo Ethernet Shield à plataforma Arduino, como mostra a Figura 6.12 [Arduino e cia, 2013].

Figura 6.12. Encaixe do Módulo Ethernet [ARDUINO E CIA, 2013]

Após encaixar o Ethernet Shield à placa do Arduino, deve-se conectar o cabo de rede. Com o shield ethernet devidamente encaixado na placa do Arduino, basta ligar o cabo de rede. Na parte superior, temos os leds de status, que mostram o funcionamento do módulo e o status de conexão à rede, como mostra a Figura 6.13. [Arduino e cia, 2013]

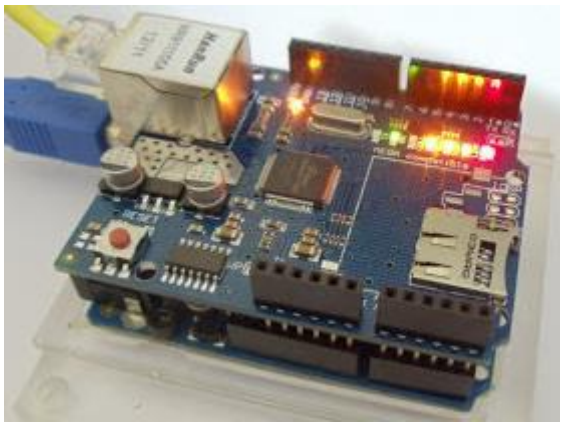


Figura 6.13. Módulo Ethernet conectado [Arduino e cia, 2013]

Como o dispositivo tem uso simplificado, não é necessário se preocupar com esquemas de sinalização de redes ethernet, uma vez que todo o controle e conexão é feito através de funções da biblioteca. Uma vez conectado, pode-se ler ou escrever dados através da conexão, de forma semelhante à uma conexão serial convencional. [Cândido, 2013]

Para testar a placa, será apresentado um projeto simples, que configura o endereço IP para valores apresentados na Figura 6.14.

Endereço IP	192.168.0.100
Gateway	192.168.0.1
Máscara	255.255.255.0

Figura 6.14. Endereços a serem configurados no módulo Ethernet [Arduino e cia, 2013]

O código utilizado para teste do Módulo Ethernet é mostrado na Figura 6.15. Como exemplo, está apenas sendo configurado os endereços de IP, Gateway e Máscara do módulo, para os valores mostrados na Figura 6.14. É importante salientar que os endereços IPs são separados por vírgula, ao invés do ponto, como é normal. O código da Figura 6.15 deve ser carregado para o Arduino.

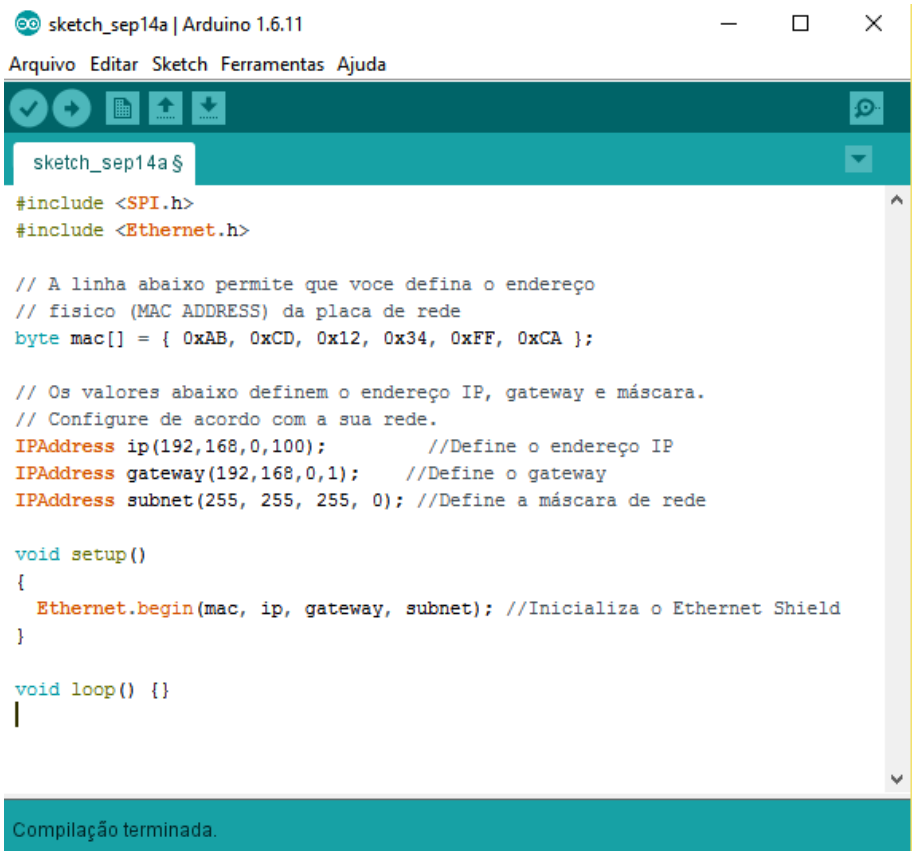


Figura 6.15. Programa Exemplo de comunicação com o Módulo Ethernet [Arduino e cia, 2013]

Para testar se o Módulo foi configurado corretamente, utilize o comando ping. Abra o *prompt* de comando do Windows, e digite o comando: ping 192.168.0.100 (o endereço que utilizamos para configurar o módulo). Caso o módulo tenha sido configurado com sucesso, o resultado será como o mostrado na Figura 6.16.

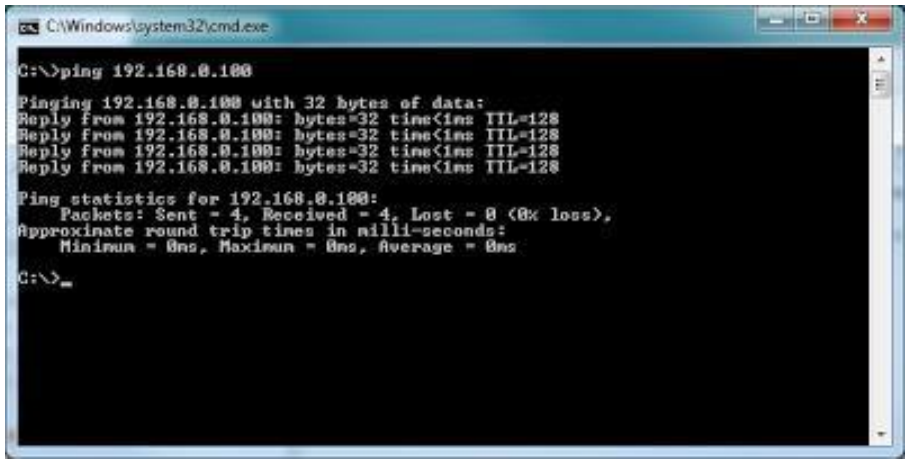


Figura 6.16. Resultado do comando ping [Arduino e cia, 2013]

6.2.5. Sensores e Atuadores

De forma a permitir maior ação do Arduino, podem ser acoplados a ele sensores e atuadores. Os sensores são conversores de grandezas físicas em sinais elétricos, já os atuadores, convertem energia elétrica, hidráulica ou pneumática em energia mecânica, permitindo gerar movimento. [Automação e robótica, 2012]

Os sensores podem ser classificados como:

- Sensores externos: que fazem observação do ambiente exterior ao Arduino. Pode-se citar sensores de contato, de proximidade, de força, de distância, de laser, de ultrassom, de infravermelhos e sensores químicos. [Ribeiro, 2004]
- Sensores internos: fornecem informação sobre os parâmetros internos do projeto de Arduino, como a velocidade ou sentido de rotação de um motor, ou o ângulo de uma junta. Potenciômetros, codificadores e os sensores inerciais (incluindo acelerômetros, giroscópios, inclinômetros e bússolas), são exemplos de sensores internos. [Automação e robótica, 2012]

Os sensores também podem ser classificados de acordo com a forma que a energia é envolvida no processo de sensoriamento. [Automação e robótica, 2012]

- Sensores ativos: como sensores laser, ultrassom e de contato, realizam o sensoriamento através da emissão de energia para o ambiente ou por modificarem o ambiente.
- Sensores passivos: como os sensores óticos, recebem energia do ambiente [Ribeiro, 2004].

Ainda existem os sensores de distância, como o laser ou o ultrassom, sensores de posicionamento absoluto, por exemplo sistemas de GPS, sensores ambientais (de temperatura, de umidade, etc.) e inerciais (que indicam a aceleração ou a velocidade) [Ribeiro, 2004].

Os Atuadores realizam a conversão da energia elétrica, hidráulica ou pneumática em energia mecânica. É possível classificar os atuadores de acordo com o tipo de energia que utiliza [Ribeiro, 2004]:

- Atuadores Hidráulicos: utilizam um fluido à pressão para movimentar o braço. Possuem baixa precisão, mas têm grande potência e velocidade, por isso são utilizados em robô que operam grandes cargas.
- Atuadores Pneumáticos: utilizam um gás à pressão para movimentar o braço. Possuem baixa precisão, e são utilizados em tarefas do tipo pega-e-coloca em robôs de pequeno porte.
- Atuadores Eletromagnéticos: motores elétricos (de passo, servos, Corrente Continua ou Corrente Alternada) ou músculos artificiais, usados em robôs de pequeno e médio porte.

6.3. Node.js

A Internet das Coisas necessita, entre outros recursos, de uma comunicação cliente servidor em tempo real no sentido de tornar eficaz a troca de informações entre os diversos dispositivos conectados. Existem diversas tecnologias que podem cumprir com este papel, todavia algumas possuem melhor desempenho e são mais adequadas para uma situação especifica. Também podemos escolher a tecnologia com base em sua simplicidade e compatibilidade com os recursos disponíveis atualmente [Pereira, 2014].

A maior parte dos sistemas web desenvolvidos atualmente contam com uma arquitetura bloqueante (*Blocking-Thread*), isto é, enquanto um processo é executado os demais ficam em uma fila aguardando para serem executados [Pereira, 2014]. Logo percebemos que este tipo de arquitetura pode trazer lentidão ao sistema e exigir diversos recursos computacionais para manter a fila. Tecnologias que apresentam tal arquitetura não são recomendadas, uma vez que um dos objetivos da Internet das Coisas é uma comunicação rápida entre sensores e elementos de processamento.

O Node.js nasceu em 2009, apresentando como proposta ser uma arquitetura não bloqueante (*non-blocking thread*). O Node.js aproveita ao máximo os recursos disponíveis e garante uma boa performance em sistemas que trabalham com grande carga de processamento. É uma plataforma escalável de baixo nível, onde o programador trabalha com protocolos de rede e internet ou usa bibliotecas especificas para uma determinada tarefa [Pereira, 2014].

A linguagem de programação utilizada no Node.js é o Javascript, pois sua construção foi baseada na *engine* Javascript V8 [Node.js, 2016].

6.3.1. Instalação e Configuração

O Node.js está disponível e pode ser baixado gratuitamente no endereço <https://nodejs.org/en/download/>, seu processo de instalação é simples e pode ser feito em diversas plataformas, conforme mostra a Figura 6.17. A configuração aqui apresentada foi feita em um ambiente com sistema operacional Windows, mas é semelhante nas demais plataformas.

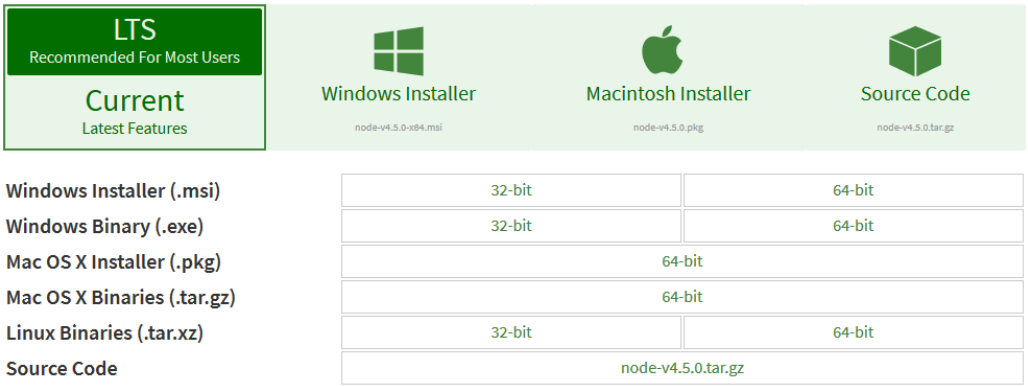


Figura 6.17. Tela de download do Node.js [Node.js, 2016]

Após efetuar o *download* da versão desejada, é necessário clicar no executável e dar início ao processo de instalação, exceto quando se tratar do sistema operacional

Linux. No processo de instalação é necessário apenas concordar com os termos e aguardar até que seja finalizado.

Após a conclusão da instalação podemos verificar se o Node.js está funcionando perfeitamente, para isto é necessário apenas abrir o *prompt* e executar o seguinte comando: `node -v`, conforme mostra a Figura 6.18, onde vemos que foi instalada a versão 4.5.0 do Node.js.

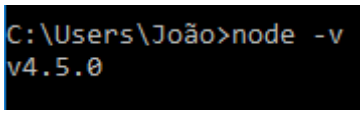


Figura 6.18. `Prompt` **de comando mostrando a versão do Node.js instalada.**

Para ter certeza que o Node.js está funcionando perfeitamente ainda podemos testar nosso primeiro programa que mostra uma mensagem *Hello Word*. Para criar e executar o programa basta digitar o comando: `node`, que nos permite executar Javascript no terminal e fornecer o seguinte comando: `console.log("Hello Word");` depois da execução do programa aparecerá a mensagem, conforme mostra a imagem 6.19.

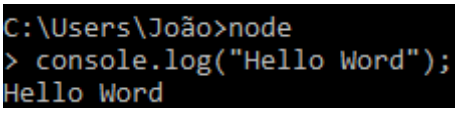


Figura 6.19. Testando o primeiro programa no prompt de comando.

O Node.js conta com um gerenciador de pacotes, ele se chama NPM (*Node Package Manager*) [Pereira, 2014]. Este gerenciador de pacotes possui o mesmo objetivo dos demais gerenciadores que estão presentes em outras tecnologias de desenvolvimento, isto é, gerenciar pacotes que são necessários para o desenvolvimento e execução de uma determinada aplicação. A seguir são listados e descritos alguns comandos do NPM.

- `npm install nomeDoMódulo`: instala um determinado módulo no projeto.
- `npm install nomeDoMódulo –save`: instala um determinado módulo e atualiza o package.json na lista de dependências da aplicação.
- `npm list`: lista todos os módulos do projeto.
- `npm –v`: mostra a versão atualmente instalada do npm.
- `npm remove nomeDoMódulo`. Remove um módulo do projeto.

Existem diversos outros comandos, a documentação completa do NPM pode ser obtida em <https://docs.npmjs.com/>.

Existe uma infinidade de módulos disponíveis em <https://www.npmjs.com/>, cada um para uma determinada tarefa. É interessante que o programador mantenha um arquivo com o nome e a versão de cada módulo que o sistema necessitar, tanto para ele, quanto para outro desenvolvedor que futuramente estiver realizando alguma manutenção na aplicação. No Node.js usa-se o arquivo `package.json`.

O arquivo `package.json` é formado principalmente por dois atributos o nome e o versão dos pacotes que serão instalados com a execução do seguinte comando: `npm install`. Um exemplo de um arquivo `package.json` é apresentado a seguir:

```
{
   “name”: “primeiro_projeto”,
   “version”: “0.0.1”,
   “description”: “um_projeto_qualquer”,
   “author”: “senhor_exemplo”,
   “private”: true,
   “dependencies”: {
      “módulo 1”: “0.1.2”,
      “módulo 2”: “1.1.2”
}
}
```

As dependências que foram descritas, módulo 1 e módulo 2, serão instalados com a execução do comando: `npm install`, conforme mencionado anteriormente. A criação deste arquivo pode ser feita manualmente ou com a ajuda de um assistente que pode ser acessado pelo prompt através da execução do seguinte comando: `npm init`, em seguida serão solicitadas algumas informações e ao fim do processo um arquivo package.json será gerado, conforme mostra a imagem 6.20.

```
C:\Users\João>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg> --save` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
name: (João) PrimeiroProjeto
Sorry, name can no longer contain capital letters.
name: (João) primeiro_projeto
version: (1.0.0)
description: um projeto qualquer
entry point: (index.js)
test command:
git repository:
keywords: projeto qualquer meu
author: senhor_exemplo
license: (ISC)
About to write to C:\Users\João\package.json:

{
  "name": "primeiro_projeto",
  "version": "1.0.0",
  "description": "um projeto qualquer",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [
    "projeto",
    "qualquer",
    "meu"
  ],
  "author": "senhor_exemplo",
  "license": "ISC"
}
```

Figura 6.20. Criação de um arquivo package.json através do assistente

6.3.2. Criando a primeira aplicação com Node.js

Agora que já aprendemos um pouco sobre Node.js, podemos criar nossa primeira aplicação Web.

Primeiramente vamos criar um arquivo chamado: *server.js*, este será o nosso servidor. Após a criação do arquivo, inicialmente vamos declarar o módulo http que é nativo do Node.js, através da seguinte instrução:

```
http = require('http');
```

Após a declaração do módulo http, vamos criar uma função que é responsável por levantar o servidor e colocar seu call-back em funcionamento. O callback é representado pela função: function(request, response) e é executado toda vez que a aplicação recebe uma requisição.

```
var servidor=http.createServer(function(request, response)){
... });
```

A função declarada é chamada de *Event Loop*, ou seja, a cada requisição ele será acionada e irá enviar uma resposta ao usuário. Vamos enviar uma mensagem para o

usuário conforme o endereço acessado, ou seja, vamos criar rotas para o usuário acessar através da url. As rotas e o envio de uma mensagem em uma página da web pode ser feitas da seguinte forma:

```
var servidor = http.createServer(function(request, response){
  response.writeHead(200, {"Content-Type": "text/html"});
  if(request.url == "/")
    response.write("<h1>Página principal do meu site</h1>");
  else if(request.url == "/contato")
    response.write("<h1>e-mail: exemplo@exemplo.com !!</h1>");
  else
    response.write("<h1>Página não encontrada!!</h1>");
    response.end();
});
```

O método `http.writeHead` constrói um novo cliente HTTP e código 200 refere-se é um código de sucesso da requisição HTTP, existem diversos outros códigos de status que podem ser informados neste método. Foram criadas diversas condições e dentro de cada uma é exibida uma mensagem ao usuário. O método request.url retorna o endereço que o usuário acessou.

A aplicação está quase terminada, falta apenas a criação do método `listen` que traz a porta que o servidor está sendo executado. Este método pode ser criado através das seguintes instruções:

```
servidor.listen(3000, function() {
     console.log("O servidor está em execução"); });
```

O servidor será executado na porta 3000, conforme mostra o código. Agora a aplicação está terminada e para ser executada são necessários os seguintes passos:

1. Navegar até o diretório da aplicação;
2. Executar o comando: `node nome_servidor.js;`
3. Abrir o navegador e acessar o endereço: http://localhost:3000/.

Após a execução destes passos é possível navegar entre as diversas rotas criadas no servidor.

A simplicidade e praticidade do Node.js tornam esta tecnologia ideal para o desenvolvimento de aplicações da Internet das Coisas.

6.4. Desenvolvimento de aplicação prática

Nesta sessão será apresentado um sistema de controle de leds via página Web com Arduino. A aplicação consiste no controle de leds montados em um circuito conectado a uma placa arduíno e controlado por uma página web hospedada em um servidor criado com o framework Node.js

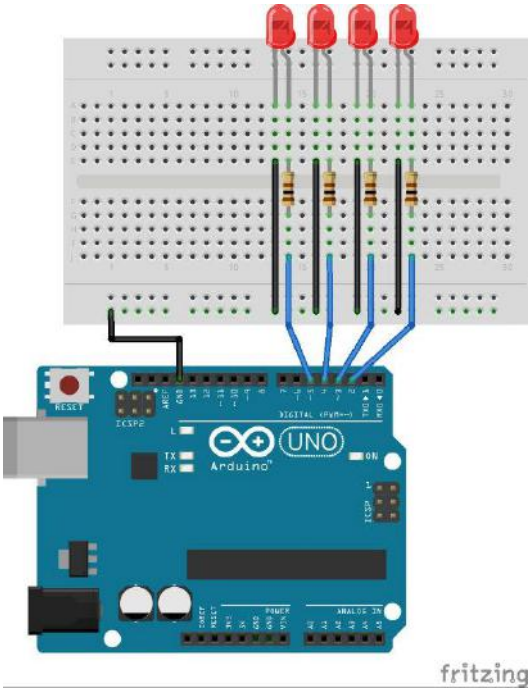
6.4.1. Montagem do Circuito

Para a montagem do circuito será necessário a utilização dos seguintes materiais:

- Uma placa arduíno
- Uma protoboard (placa onde é montado o circuito)

- 4 leds
- 9 fios
- 4 resistores de 300k

O circuito montado é apresentado na Figura 6.21.



6.21. Circuito montado [PedroHS, 2015]

6.4.2. Implantação do Código no Arduíno

Para o funcionamento do sistema é necessário que seja importado para o arduíno um firmware que permite sua comunicação com um computador servidor, no caso o localhost, ou seja, a máquina que possui o servidor Node.js instalado e ficará ligada ao arduíno através de um cabo USB.

O protocolo a ser utilizado será o Firmata que é uma biblioteca de comunicação do hardware com o computador servidor. Ele permitirá que o hardware seja controlado pelo software do servidor. Então para a aplicação é necessário a instalação do Firmata no Arduino como um firmware e assim ele estabelecerá a comunicação entre hardware e software no computador servidor para que este controle o Arduino [Arduino, 2016d] [Firmata, 2016].

Para a instalação é necessário abrir o ambiente Arduino IDE (Figura 6.22)

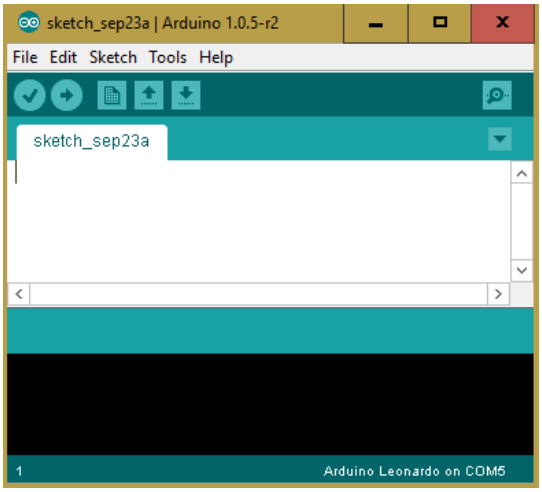


Figura 6.22. Tela inicial do Arduino IDE

Depois vá em Examples > Firmata > StandardFirmata (Figura 6.23_

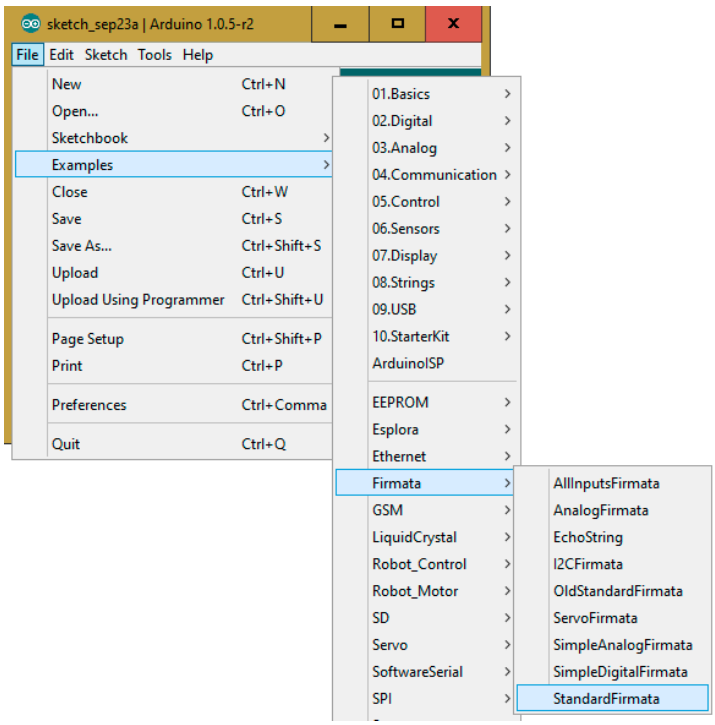


Figura 6.23. Selecionando o protocolo StandardFirmata

Ao selecionar o programa StandardFirmata ele será carregado na área de edição de código do Arduino IDE (Figura 6.23).

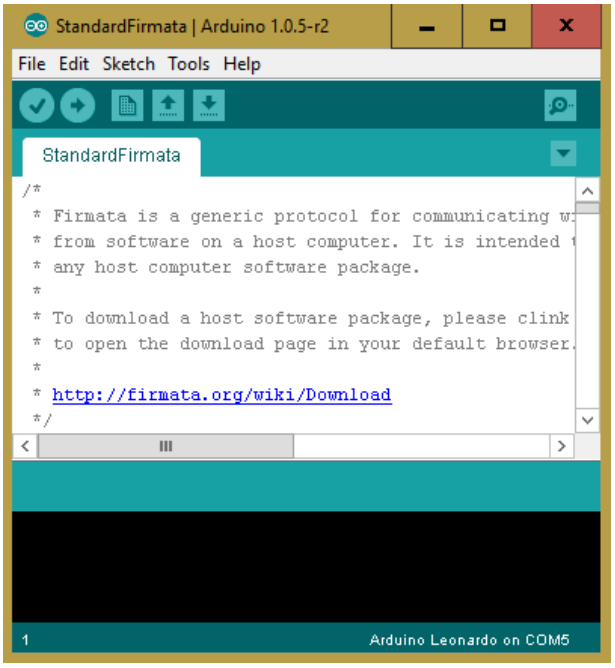


Figura 6.24. Código do StandardFirmata no ambiente do Arduino IDE

Antes do código ser enviado para o arduino é necessário executar a verificação do código com o ícone  . Esta operação evitará que o código com erros seja transferido para a placa arduino o que evita que este venha a danificá-la. Para que o código seja carregado no arduíno o botão. Depois de verificar o código é necessário realizar o upload para a placa arduíno através do ícone  .

Após a transferência do protocolo para o arduino será exibido no rodape do Arduino IDE uma mensagem de confirmação que o software foi transferido corretamente.

A partir da transferência a comunicação entre o computador e o arduino poderá ser feita permitindo que uma página web através do servidor Node.js controle os leds que estão montados no circuito.

6.4.1. Criação da Aplicação Javascript

A aplicação será construída em Javascript utilizando o servidor Node.js e o framework de código aberto Johnny Five que permite controlar o harware utilizando Javascript. Para instalá-lo é necessário instalar o Node.js e NPM (como descrito na sessão 6.2). Em seguida criar a pasta onde será colocado o projeto e via prompt ela deverá ser selecionada. Para instalação do framework Johnny Five na pasta do projeto o seguinte comando deverá ser executado no prompt de comando: `npm install johnny-five` (Figura 6.25).

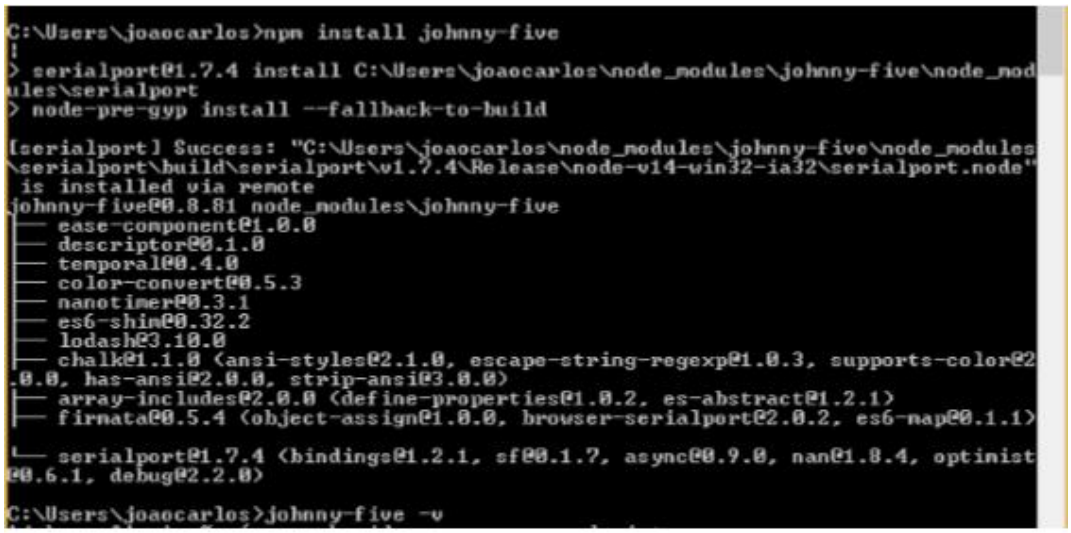


Figura 6.25. Instalação do johnny-five no terminal do Windows

Após a instalação do framework Johnny Five é necessário criar o servidor que atenderá as requisições feitas pelo navegador. Este servidor será chamado **app.js** e deverá ser gravado na pasta do projeto. Na Figura 6.26 é apresentado as definições das bibliotecas que serão utilizadas e variáveis e na Tabela 6.1 cada linha do código é comentada.

```
1 var http = require('http').createServer(servidor);
2 var fs = require('fs');
3 var five = require('johnny-five');
4 var arduino = new five.Board();
5 var led0, led1, led2, led3;
```

Figura 6.26. Importação das bibliotecas e definição das variáveis

Tabela 6.1. Comentários em relação ao código da Figura 6.26

Linha	Comentário
1	O módulo com o protocolo HTTP é importado (o módulo http é nativo do Node.js) e a função createServer é responsável por deixar o servidor passível de receber requisições. O método recebe como parâmetro o nome da função que será responsável por iniciar o servidor.
2	O módulo fs (File System) permite a leitura de arquivos externos.
3	O módulo johnny-five permite controlar a placa arduino através do Javascript
4	Criação do objeto arduino (representação em software da placa arduino) através do método five.Board()
5	Variáveis que irão armazenar a representação em software dos Leds ligados à placa arduíno.

Na linha 4 da Figura 6.26 foi criado o objeto arduino que é a representação em software da placa arduino através do método da biblioteca johnny-five `Board()`. Este método inicia a comunicação com a placa, conectando o servidor com ela e somente após a comunicação ser estabelecida é possível identificar os componentes da placa e executar a ligação destes com seus correspondentes em software. Na Figura 6.27 é apresentado o código da função que é executada quando a comunicação entre servidor e na Tabela 6.2 cada linha de código da Figura 6.27 é comentada.

```
 7 arduino.on('ready', function(){
 8     console.log("Pronto");
 9     led0 = new five.Led(2);
10     led1 = new five.Led(3);
11     led2 = new five.Led(4);
12     led3 = new five.Led(5);
13 });
```

Figura 6.27. Conexão com a placa arduino

Tabela 6.2. Comentários em relação ao código da Figura 6.27

Linha	Comentário
7	Quando a placa arduíno entra no estado de 'ready', ou seja, quando a comunicação entre hardware e servidor é estabelecida a função definida entre as linhas 8 e 12 é chamada
8	Uma mensagem é exibida no terminal de comando
9 a 12	Cada led é presentado em software através do método do framework johnny-five `Led()` que recebe como parâmetro a porta digital da placa arduíno na qual cada Led está conectado.

É necessário criar uma função que representa o servidor que irá atender as requisições do usuário. Na Figura 6.28 é apresentado o código do método servidor e na Tabela 6.3 cada linha de código é comentada.

```
15 function servidor(req, res){
16     var url = req.url;
17     if(url == '/'){
18         res.writeHead(200);
19         res.end(fs.readFileSync('view/index.html'));
20     } else if(url == '/led0'){
21         res.writeHead(302, {'Location': '/'});
22         res.end();
23         led0.toggle();
24     }else if(url == '/led1'){
25     //...
26     } else{
27         res.writeHead(200);
28         res.end("<h1> Erro 404 </h1>");
29     }
30 };
```

Figura 6.28. Método servidor

Tabela 6.2. Comentários em relação ao código da Figura 6.28

Linha	Comentário
15	O método recebe como parâmetro os objetos de requisição e resposta (req e res respectivamente). O objeto requisição possui as informações que são enviadas ao servidor e o objeto resposta será a mensagem que retornará ao cliente (solicitante da requisição)
16	A url da requisição é lida
17, 20, 24	Verificação de qual url foi solicitada

18	O método res.writeHead constrói um novo cliente HTTP e código 200 refere-se é um código de sucesso da requisição HTTP
19, 22, 28	Finalização da conexão com o cliente, podendo enviar alguma informação. Na linha 19 estamos redirecionando o cliente para a página view/index.html e na linha 28 enviamos uma mensagem em HTML
21	O código 302 indica um redirecionamento para, neste caso, a página raiz, ou seja a index
23	O método `toggle()` muda o estado do Led, se ele estiver aceso ele apagará e vice versa.

A aplicação está quase terminada, falta apenas a criação do método `listen` que traz a porta que o servidor está sendo executado (Figura 6.29). Função já descrita na sessão 6.3.2.

```
32 http.listen(3000, function(){
33     console.log("Servidor on-line!");
34 });
```

Figura 6.29. Método listen

Agora que temos o servidor pronto é necessário a criação da página na qual o cliente terá acesso e poderá mudar o estado dos leds. O arquivo se chamará index.html (Figura 6.30) e ficará dentro da pasta view que será criada na pasta do projeto.

```
 1 <html>
 2   <head>
 3     <title>Aplicação IoT Leds</title>
 4   </head>
 5   <body>
 6     <ul>
 7       <li><a href="/led0">Led 01</a></li>
 8       <li><a href="/led1">Led 02</a></li>
 9       <li><a href="/led2">Led 03</a></li>
10       <li><a href="/led3">Led 04</a></li>
11     </ul>
12   </body>
13 </html>
```

Figura 6.30. Código HTML da página view/index.html

A aplicação agora poderá ser executada através da execução em prompt do comando `node app.js`.

Na Figura 6.31 podemos ver que a placa arduíno foi reconhecida como conectada à porta serial COM3 e a mensagem do status do servidor também foi mostrada. Agora precisamos acessar a view que foi criada, para isso é necessário acessar o endereço <localhost:3000> (Figura 6.32).

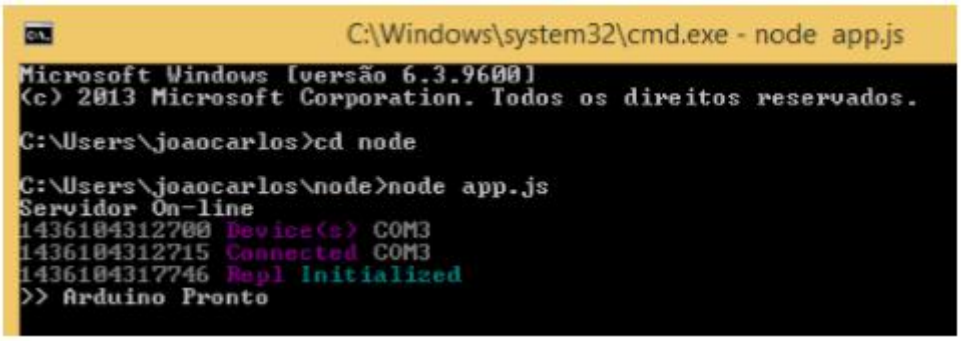


Figura 6.31. Execução da aplicação

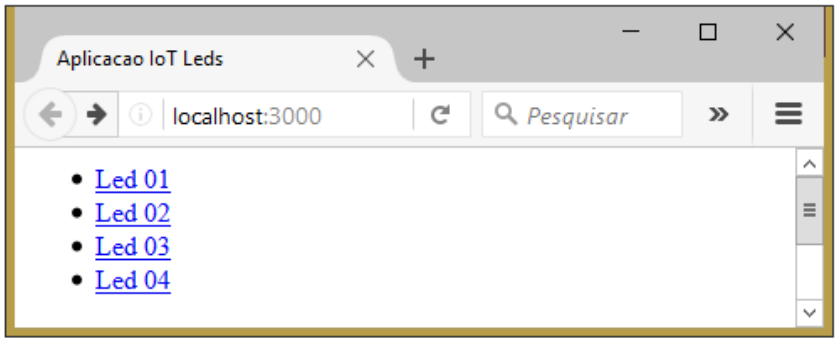


Figura 6.31. Renderização em navegador da página view/index.html

Ao clicar nos links da Figura 6.31 os estados de ligado/desligado dos Leds é alternado.

Referências

Aloi, R. (2013) "Guia para o Arduino Leonardo", Disponível em: http://renatoaloi.blogspot.com.br/2013/12/guia-para-o-arduino-leonardo.html Acesso em: Set. 2016

Ansari, W. S. et al. (2013) "A Survey on Cloud-Sensor: architecture, applications and approaches". In: J. Distrib. Sens. Networks, [S.l.], v.2013, p. 1–36.

Arduino (2016a), "*Credits*", Disponível em: https://www.arduino.cc/en/Main/Credits, Acesso em: Set. 2016

Arduino (2016b), "*Learning*", Disponível em: http://playground.arduino.cc/Portugues/HomePage, Acesso em: Set. 2016

Arduino (2016c), "*Install the Arduino Software (IDE) on Windows PCs*", Disponível em: https://www.arduino.cc/en/Guide/Windows, Acesso em: Set. 2016

Arduino (2016d), "*Firmata Library*", Disponível em: https://www.arduino.cc/en/reference/firmata, Acesso em: Set. 2016

Arduino e Cia, (2013), "Ethernet shield Wiznet W5100 - Parte 1", Disponível em: http://www.arduinoecia.com.br/2013/06/ethernet-shield-wiznet-w5100-parte-1.html. Acesso em: Set. 2016

Armbrust, M., Fox, A., Griffith, R., et al. (2010) “A View of Cloud Computing”. Communications of The ACM, Vol 53, No 4, p. 50-58.

Atzori, L., Iera, A., e Morabito, G. (2010) “The Internet of Things: A survey”. Computer Networks.

Automação e Robótica, (2012). “Sensores, Atuadores e Unidades de Controle”. Disponível em: http://automacaoerobotica.blogspot.com.br/2012/07/sensores-e-atuadores-aplicados-robotica.html, Acesso em: Set. 2016

Boxall, J. (2013). “*Arduino workshop: A Hands-On introduction with 65 projects*”. No Starch Press.

Candido, R., (2013). “Acendendo um LED via Internet com Arduino e o Ethernet Shield”. Disponível em: https://br.renatocandido.org/2013/09/acendendo-um-led-via-internet-com-arduino-e-o-ethernet-shield/. Acesso em: set. 2016

Firmata (2016) “Firmata”. Disponível em: http://firmata.org/wiki/Main_Page. Acesso em: set. 2016

Laboratorio de Garagem, (2014), Tutorial: Como utilizar o Ethernet Shield com Arduino, Disponível em: http://labdegaragem.com/profiles/blogs/tutorial-como-utilizar-o-ethernet-shield-com-arduino, Acesso em: set. 2016

Miao, W., et. al. (2010) “Research on the architecture of Internet of things”. In.: 3rd IEEE International Conference on Advanced Computer Theory and Engineering (ICACTE). Sichuan, China, 21 ago. 2010. p. 484-487.

Pereira, C. R. (2014). “Aplicações web real-time com Node.Js”. Editora Casa do Código.

Node.js (2016) “Node. Js”. Disponível em: <https://nodejs.org/en/>. Acesso em: ago 2016.

PedroHS (2015) “Controlando Leds via página web utilizando Node.js e Arduino”. Disponível em: <https://pedrohs.github.io/controle-de-leds-via-http/>. Acesso em: set. 2016

Perez, A. L. F. et al. (2013) “Uso da Plataforma Arduino para o Ensino e o Aprendizado de Robótica”. Proceedings of the International Conference on Interactive Computer aided Blended Learning (ICBL2013).

Rea, S.; Aslam, M. S.; Pesch, D. (2013) “Serviceware-A service based management approach for WSN cloud infrastructures”. In: IEEE INT. Conf. Pervasive Comput. Commun. Work. Percom Work. 2013, 2013. Anais. . . [S.l.: s.n.], n. March, p.133–138.

Ribeiro, M. I. (2004) “Sensores em Robótica”, Enciclopédia Nova Activa Multimédia, Volume de Tecnologias, pags. 228-229. Portugal.

Robot@escola. (2016) “Tutorial de Programação Arduino – Lição 3”, Disponível em: http://escoladerobotica.ipcb.pt/?page_id=297. Acesso em: set. 2016

Souza, F., (2013) “Arduino - Primeiros Passos”, Disponível em: http://www.embarcados. com.br/arduino- primeiros-passos, Acesso em: Set. 2016.

Vermesan O. e Friess P., (2013) “Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems “

Capítulo 7

MM4DM: o papel de multimídia em processos de tomada de decisão na era da computação cognitiva

Marcio Ferreira Moreno[1], Rafael Brandao[1] e Renato Cerqueira[1]

[1] IBM Research | Brazil – {mmoreno, rmello, rcerq}@br.ibm.com

Abstract

The increasing momentum towards cognitive computing unlocks a diverse set of opportunities and challenges for the multimedia research area. In fact, with a different approach from the one present in the traditional artificial intelligence systems, cognitive computing glimpses a human-machine collaboration, where a more symbiotic interaction is required. The main goal of the Multimedia for Decision-Making (MM4DM) tutorial is to discuss how the multimedia research area enrolls decision-making processes in the era of cognitive computing. In this context, this tutorial discusses topics of multidisciplinary interest, always from a multimedia perspective, aiming to inspire heterogeneous participation of researchers from industry and academia.

Resumo

O crescente uso da computação cognitiva em diferentes setores do mercado traz um conjunto diversificado de oportunidades e desafios para pesquisa científica na área de multimídia. Com uma abordagem diferente da apresentada por tradicionais sistemas de inteligência artificial, a computação cognitiva vislumbra uma colaboração humano-máquina, em que uma interação mais simbiótica se faz necessária. O minicurso de multimídia para tomada de decisão (MM4DM – Multimedia for Decision-Making) possui como objetivo principal discutir o papel da área de multimídia nos processos de tomada de decisão na era da computação cognitiva. Nesse contexto, o minicurso debate tópicos de interesse multidisciplinares, sempre de uma perspectiva multimídia, almejando inspirar uma participação diversificada de pesquisadores da indústria e da academia.

7.1. Introdução

A computação cognitiva possui relevância ímpar demonstrada ao ter saído do status de tendência tecnológica para definir uma nova era na computação. De fato, a era da computação cognitiva [Soffer 2016] traz não apenas novos desafios para pesquisa científica, como também contribuições multidisciplinares capazes de apresentar soluções sob novas perspectivas em diversas áreas, incluindo a de sistemas para tomada de decisão. Os processos de tomada de decisão, por sua vez, compõem uma área de interesse da academia e, principalmente, da indústria. A natureza colaborativa homem-máquina [Licklider 1960] dessas duas áreas, em que uma interação mais simbiótica entre essas duas partes se faz necessária, consiste no principal argumento dos autores deste capítulo, que defendem que o sucesso dessas áreas depende fortemente do engajamento da comunidade multimídia.

Computação cognitiva possui uma abordagem diferente da apresentada por tradicionais sistemas de inteligência artificial. De fato, a computação cognitiva possui como objetivo principal aumentar a compreensão e capacidade humana, não importando a atividade ou processo. Assim, a computação cognitiva vislumbra uma colaboração humano-máquina em que uma interação mais natural entre essas duas entidades se faz necessária.

O processo de tomar uma decisão pode ser visto [Power 2015] como um método cognitivo de fazer escolhas, a partir da definição de metas, identificação e consolidação de informações (evidências), reflexão sobre alternativas, até a escolha de uma delas para tomar ações. Nessa perspectiva, o papel da área de pesquisa em multimídia vai muito além de uma análise isolada de conteúdo multimídia para melhorar, por exemplo, indexação e recuperação de conteúdo. Na verdade, multimídia desempenha um papel chave no processo de qualificação semântica de dados não estruturados [Moreno et al. 2016a] [Moreno et al. 2016b] [Moreno et al. 2016c] [Moreno et al. 2016d] exigida para a tomada de decisão. Esse fato traz diversos desafios relacionados a tópicos de interesses distintos, reunidos em domínios específicos, que serão detalhados na Seção 7.2 deste capítulo.

Alguns conceitos são necessários para discutir esses desafios em multimídia. A seção a seguir apresenta as principais considerações necessárias. Mais especificamente, os seguintes conceitos serão apresentados: Processos de tomada de decisão. O que é um sistema de suporte a tomada de decisão. O que é um sistema cognitivo. Sistemas cognitivos e computação cognitiva. A questão da simbiose homem-máquina. A importância de multimídia em computação cognitiva e em sistemas de suporte a tomada de decisão.

7.1.1. Conceitos Básicos

Tomada de decisão consiste em um processo cognitivo geralmente estudado sob perspectivas diferentes e multidisciplinares como, por exemplo, Ciência da Computação, Estatística, Economia, Administração, Psicologia, Ciências Sociais e Filosofia. Esse processo pode ser definido como uma atividade para resolução de problemas, em que o resultado alcançado é uma solução considerada satisfatória por um indivíduo ou grupo de pessoas. Os usuários nesse processo são aqueles que tomam decisões, ou tomadores de decisão neste capítulo. Durante o processo, eles podem recorrer a conhecimento explícito e tácito, consumindo e produzindo uma variedade de conteúdo e artefatos. Por mais simples e direta que essa descrição possa parecer, ela engloba um amplo espaço

agrupando inúmeras atividades. De fato, essas atividades vão desde escolhas cotidianas, aparentemente simples, como escolher uma refeição ou comprar roupas, até cenários críticos, tais como diagnósticos médicos ou decisões estratégicas de negócios. No prefácio de seu livro sobre tomada de decisão multicritério, Triantaphyllou discute brevemente o tópico tomada de decisão. Em suas palavras, esse é "provavelmente o desafio eviterno mais intelectual das ciências e engenharias". Na verdade, como ele mesmo afirma, até mesmo os nossos antepassados mais distantes muitas vezes recorreram a entidades, divindades ou aos sábios de sua época, a fim de obter suporte a tomada de decisões.

Nas últimas décadas, diversos mecanismos, metodologias e teorias como a programação linear e dinâmica, teste de hipóteses, tomada de decisão multicritério, teoria dos jogos, entre outros, foram propostos para apoiar a tomada de decisão, individual e em grupo. Nessa área de pesquisa tão ampla, diferentes abordagens computacionais aproveitaram o poder de computação, cada vez mais disponível, para fazer progressos substanciais em sua busca por melhores sistemáticas metodologias para suporte a tomada de decisão.

Mais recentemente, os avanços na área de inteligência artificial e áreas de aplicação (por exemplo, *deep learning*, *reinforcement learning*, computação visual, *NLP* etc.) vêm trazendo os sistemas de suporte a tomada de decisão para uma nova fase, alcançando melhores resultados em cenários complexos da nossa realidade. Na visão dos autores deste capítulo, a tomada de decisão deve explorar o poder cognitivo aumentado, resultante da simbiose homem-máquina, em uma perspectiva já estabelecida na computação cognitiva.

Na prática, no entanto, as disciplinas e as teorias mencionadas, dependem em última instância da interpretação humana. É necessário que alguém analise e correlacione os conceitos e significados extraídos de dados heterogêneos em diferentes formatos e modalidades, com o objetivo de formalizar e modelar os problemas de decisão e possíveis alternativas. Em outras palavras, os tomadores de decisão comumente devem interpretar conhecimento manifestado em diferentes tipos de mídia.

Do ponto de vista da multimídia, a tomada de decisão é também um tema de pesquisa com desafios expressivos e multidisciplinares. Em termos gerais, aspectos que podem ser de particular interesse para a comunidade científica referem-se à análise multimodal de dados e fusão multimodal, sensoriamento e aquisição de dados, modelos de representação, compreensão e sincronização de conteúdo, recuperação e técnicas de apresentação personalizadas, significação social em dados multimídia, conhecimento subjetivo e raciocínio contextual sobre dados enriquecidos. Explorar e compreender esses aspectos de um ponto-de-vista multimídia implica em alavancar a extração semântica e uma significação mais rica do conteúdo, o qual se torna cada vez mais acessível com as redes sociais, e cada vez mais usado como evidências em diferentes decisões estratégicas.

Computação cognitiva também envolve diversas áreas de pesquisa. Seu objetivo é aumentar a compreensão e capacidade humana, independente da atividade ou processo. De fato, um aspecto comum promovido por diferentes autores na literatura é que ela precisa de uma relação simbiótica, em que os sistemas de hardware e software respondem e aprendem com os padrões de comportamento e interação humana.

Outra característica intrínseca à computação cognitiva é que ela foi desenhada para que a interação com o usuário seja uma experiência mais natural. Sistemas programáveis tradicionais possuem processos pré-determinados e determinísticos, que são adequados

para o processamento de dados estruturados. No entanto, esses sistemas possuem limitações no processamento de dados qualitativos e imprevisíveis. O fato é que aplicações muitas vezes têm de lidar com ambiguidade e incerteza. Em contraste com o desenvolvimento de software tradicional, onde as regras são explicitamente programadas previamente para modelar como processo de interação com um artefato de software deve ser, programas cognitivos são projetados utilizando uma abordagem probabilística, com o objetivo de adaptar e fazer com que informações não estruturadas adquiridas durante a interação tenham sentido.

Atualmente, diferentes sistemas, tais como IBM Watson[1], Haven OnDemand[2], Microsoft Cognitive Services[3] e TensorFlow[4], oferecem diferentes serviços "inteligentes" para a extração semântica e compreensão do conteúdo utilizando técnicas de Machine Learning, NLP (Natural Language Processing), Computação Visual etc. Reconhecimento de fala, análise de sentimento, detecção facial e percepção de conceitos são alguns dos recursos oferecidos por esses sistemas cognitivos computacionais. Em geral, esses sistemas executam em nuvem e oferecem seus serviços como PaaS (Platform as a Service) ou SaaS (Software as a Service). Detalhes sobre diferentes modelos de serviços em nuvem são discutidos na Seção 7.3. Para utilizar esses serviços em nuvem, desenvolvedores geralmente têm de lidar com diferentes lógicas, ou seja, abstrações, linguagens e APIs que abordam especificidades de processamento de dados. Quando o objetivo desses desenvolvedores é explorar a relação semântica entre as diferentes modalidades de mídia, cabe a eles especificar e escrever código com os detalhes sobre a extração, mapeando e empregando conhecimentos e conceitos inferidos a partir de dados multimídia. Tais tarefas podem ainda envolver funcionalidades como decodificação ou transcodificação de conteúdo, a fim de processar partes específicas de dados em um formato particular. Em última análise, os desenvolvedores podem ter que lidar com aspectos de processamento multimídia intrínsecos que estão fora do escopo de suas aplicações.

Na visão dos autores deste capítulo, sistemas cognitivos computacionais ainda carecem de uma abstração holística que permita a descrição de entidades tais como conteúdo multimídia, o conhecimento abstrato, recursos cognitivos, usuários, dispositivos e a relação entre eles em uma única lógica e notação. Essa abstração deve ser ampla e flexível para ajudar especificação e uso de recursos cognitivos independentemente do domínio da aplicação.

7.1.2. Organização

O minicurso de multimídia para tomada de decisão (MM4DM – *Multimedia for Decision-Making*) possui como objetivo principal discutir o papel da área de multimídia na integração cognitiva homem-máquina em processos de tomada de decisão. O minicurso debate tópicos de interesse multidisciplinares, sempre a partir de uma perspectiva multimídia, almejando inspirar a participação diversificada de pesquisadores da indústria e da academia. Os principais desafios para a área multimídia nesse contexto são discutidos na Seção 7.2.

[1] https://www.ibm.com/watson
[2] https://www.havenondemand.com
[3] https://www.microsoft.com/cognitive-services
[4] https://www.tensorflow.org

O minicurso foi desenhado para um público com conhecimento básico sobre os fundamentos de multimídia e de programação web (habilidades básicas de JavaScript, node.js, HTML5). Esse conhecimento prévio é necessário uma vez que o curso explora como estudo de caso o desenvolvimento de um protótipo que utiliza serviços web, mais especificamente, serviços cognitivos IBM Watson em multimídia. A ideia do protótipo, discutido na Seção 7.3, é ilustrar o uso de sistemas cognitivos na aplicação de técnicas para extrair conhecimento a partir de conteúdo multimídia, não só para alimentar bases de conhecimento, mas para entender os comandos do usuário, os estilos relevantes de apresentação e conceitos de tomada de decisão.

Assim, o restante deste capítulo está organizado da seguinte maneira:

7.2) **MM4DM**. Uso de multimídia em computação cognitiva e em processos de tomada de decisão. Estado da arte e desafios atuais de pesquisa. Serviços cognitivos. Integração com os desafios multimídia para suporte à tomada de decisão.
7.3) **Estudo de caso**. Conceitos básicos de computação em nuvem. IaaS, PaaS e SaaS. A plataforma Bluemix e seu catálogo de serviços. Acessando APIs e serviços cognitivos IBM Watson para extração de conhecimento em conteúdo multimídia (AlchemyAPI). Modelo conceitual hiperconhecimento.
7.4) **Considerações Finais**. Destaques do minicurso. Agenda de pesquisa.

7.2. MM4DM

Durante nossas pesquisas nas áreas de computação cognitiva e suporte a tomada de decisão [Moreno et al. 2016d], aliada à nossa experiência científica na área de multimídia, nós identificamos e classificamos os desafios multimídia no suporte a tomada de decisão em quatro categorias, como segue. Desafios relacionados com a extração de conhecimento a partir de conteúdo multimídia, consumo do conhecimento utilizando conteúdo multimídia, captura da intenção do tomador de decisão por meio de processamento de conteúdo multimídia, e modelagem de processos de tomada de decisão relacionando conceitos e conteúdo multimídia. A Figura 1 ilustra a classificação dessas quatro categorias de desafios.

Extração de conhecimento, e associação de conceitos a partir de conteúdo multimídia, consiste em um desafio que pode ser abordado por meio de diferentes estratégias.

Por um lado, no processamento de conteúdo multimídia tradicional, são utilizados padrões de metadados para descrever conhecimento e associá-lo ao conteúdo. No entanto, essa abordagem é comumente baseada em uma tarefa manual de anotação para identificação e descrição semântica dos dados. Além disso, relacionar os aspectos e abstrações de alto nível do conhecimento não é muitas vezes o ideal. De fato, existe uma lacuna semântica entre os padrões de metadados, que visam descrever recursos de conteúdo de baixo nível (por exemplo, configurações de codificação, formatos, visuais e de áudio), e ontologias que se concentram em descrever o conhecimento abstrato. O primeiro, em geral, carece de recursos e formalismos para descrever construções abstratas e para *reasoning*. O segundo, geralmente, não fornece mecanismos ricos para lidar com aspectos tradicionais multimídia, tais como sincronismo de mídia [Moreno et al. 2016e] [Moreno and Soares 2011], abstrações [Moreno et al. 2016e] [Moreno et al. 2016f] de segmentos de conteúdo, relacionamentos etc.

Por outro lado, áreas de aplicação AI, particularmente NLP, CV e *deep learning*, estão impulsionando a percepção das máquinas, com o objetivo de extrair e inferir conhecimento através de processamento multimídia. Técnicas de processamento de linguagem natural fizeram progressos significativos, especialmente em conjunto com reconhecimento de fala. Diversas pesquisas em NLP estão concentrando esforços para a concepção de sistemas de diálogo, capazes de interagir naturalmente com as pessoas. A percepção visual das máquinas por meio de técnicas de CV, muitas vezes se concentra em rotular automaticamente o conteúdo de imagem e vídeo. Com o aumento da disponibilidade de computação em larga escala e avanços nos algoritmos de redes neurais, metodologias CV têm impulsionado benchmarks de classificação visuais para níveis comparáveis aos de humanos. *Deep learning* pode ser descrita como um conjunto de técnicas de aprendizagem que emprega treinamento de redes neurais convolucionais. Tais técnicas estão se beneficiando diretamente as aplicações nas áreas de NLP e CV, processando especificamente áudio para reconhecimento de voz e criação de rótulos em vídeo para reconhecimento de objetos e tarefas.

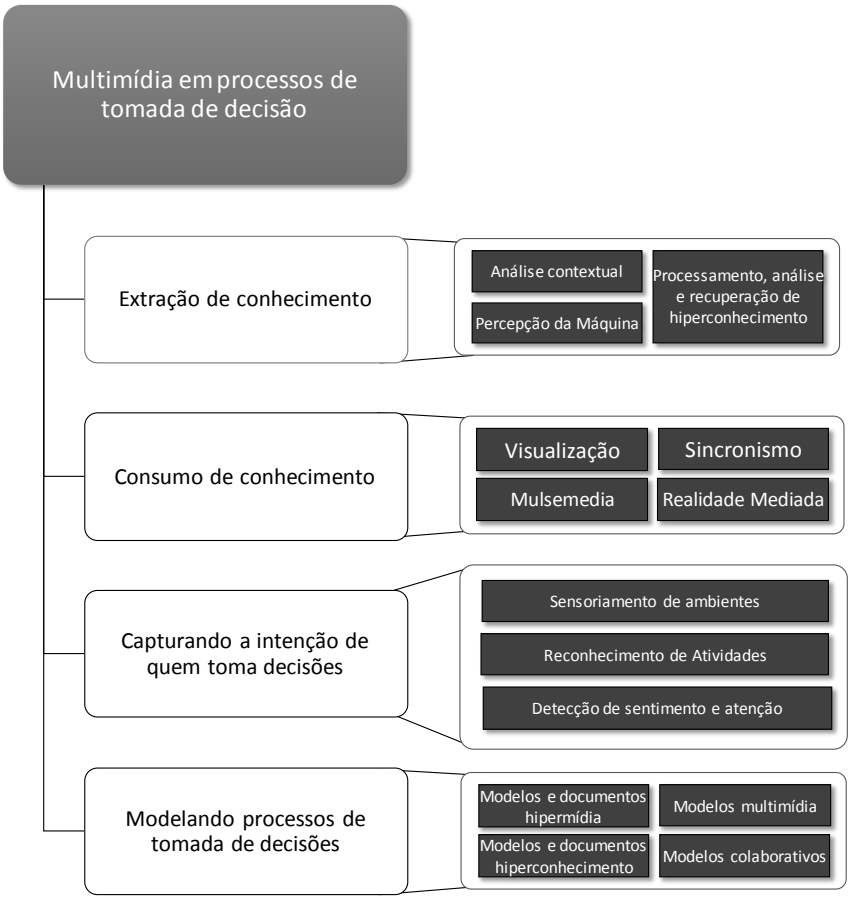


Figura 7.2.1. Desafios da área de Multimídia em Processos de Tomada de Decisão

Na extração de conhecimento, existem ainda outros pontos interessantes além da compreensão de fala e conteúdo audiovisual. É necessário compreender o significado do ambiente e do contexto e formar novas relações entre dados multimodais para descobrir e recuperar conteúdos que apresentem conceitos relacionados. Por exemplo, os conceitos

podem ter significados divergentes quando analisados sobre diferentes perspectivas e contextos [Soares et al. 2009] [Soares et al. 2010] [Soares et al. 2015]. Ao lidar com cenários cognitivamente complexos, percepção e conhecimento contextual desempenham um papel fundamental.

Conforme já mencionado no início desta seção, ***consumo de conhecimento*** consiste em outro desafio multimídia no contexto de tomada de decisões. Soluções nesse contexto devem abordar os aspectos de visualização, bem como adaptação de conteúdo, sincronismo de conteúdo multimodal e percepção por meio de sensores multimodais, sempre levando em consideração os objetivos e preferências dos tomadores de decisão. Outro tema relevante no consumo de conhecimento é a realidade mediada para os tomadores de decisão. Por exemplo, a realidade aumentada ou realidade virtual poderia ser aplicada para melhorar a experiência de consumo de conhecimento durante processos de tomada de decisão.

O terceiro desafio multimídia consiste na ***captura de intenção dos tomadores de decisão***. Para apoiar a dinâmica de um processo de tomada de decisão, é vital inferir a intenção dos tomadores de decisão. Além de identificar conceitos e conhecimentos, é necessário seguir as trilhas interpretativas desses usuários. Pesquisas sobre *reinforcement learning*, na área de inteligência artificial, podem auxiliar na estruturação da intenção dos tomadores de decisão. Ao contrário de outras técnicas de *machine learning*, que se concentram em reconhecimento de padrões, *reinforcement learning* possui como foco o suporte a processos de tomada de decisão. Juntamente com técnicas de *deep learning*, essa metodologia atingiu recentemente um sucesso significativo em aplicações do mundo real. Muitos trabalhos nessa área possuem como foco o desenvolvimento de arcabouços para suporte a tomada de decisão sequencial orientada à prática.

Além de aprender a partir de protocolo verbal dos tomadores de decisão, bem como de suas expressões visuais, é possível conseguir informações relevantes por meio de mineração de dados extraídos a partir da instrumentação do ambiente. Mais especificamente, essa instrumentação pode ser feita com rastreadores de atenção como, por exemplo, dispositivos que desempenham *eye* e *head tracking*. Esses dados podem ser valiosos para a identificação das atividades em curso, bem como para detectar a intenção e o sentimento das pessoas presentes no ambiente. Essa abordagem está alinhada com os aspectos de detecção inteligente, normalmente explorados no campo de Internet das Coisas (IoT – *Internet of Things*). Além disso, é possível explorar a instrumentação de objetos epistémicos. Ou seja, objetos usados para auxiliar a criação de conhecimento em um processo de decisão. Esses objetos podem ser físicos e tangíveis como, por exemplo, uma ferramenta, artefato de auxílio visual, ou um dispositivo computacional. Caso contrário, esses objetos podem ser virtuais e intangíveis, como uma interface interativa ou outro artefato de software. Rastrear a manipulação de objetos pode ser útil para inferir a intenção das pessoas em determinadas atividades. Por exemplo, se o objeto é um dispositivo computacional, ele poderia potencialmente ser usado para registrar os dados sobre a sua própria manipulação.

Finalmente, especificamos como o quarto desafio multimídia ***a modelagem de processos de tomada de decisão*** envolvendo assistentes cognitivos e conteúdo multimídia. Modelos abstratos que possuem esse objetivo devem definir entidades e regras que reflitam as abstrações adequadas ao contexto de tomada de decisão. Além disso, esses modelos devem considerar os desafios multimídia discutidos nesta seção, lidando ainda com outras

questões tradicionais abordadas em modelos conceituais hipermídia e multimídia, tais como a descrição de conteúdo e sua apresentação, reuso de conceitos e conteúdo, sincronismo espaço-temporal etc. Nesse contexto, existe também outro tema que merece destaque, o da modelagem da colaboração entre pessoas e sistemas cognitivos computacionais em processos de tomada de decisão. Entre os principais problemas dessa modelagem estão a especificação de dados de entrada agregados originados de um grupo de tomadores de decisão e de saída originados por assistentes cognitivos, especificação do "fluxo" de conhecimento desde sua criação até o seu armazenamento e consumo, bem como a modelagem de regras para estruturação de dados e conhecimento.

Em nossa opinião, o quarto desafio merece destaque entre todos especificados. De fato, existe a necessidade de um modelo conceitual que permita aos autores especificar ou expressar como multimídia e assistente cognitivos devem apoiar a tomada de decisões. Mais especificamente, nossa proposta é que modelos conceituais hipermídia considerem aspectos de conhecimento, utilizando conceitos de engenharia de documentos e computação cognitiva para compor o que chamamos de hiperconhecimento. Para atingir essa meta, nossas pesquisas recentes [Moreno et al. 2016a] [Moreno et al. 2016b] [Moreno et al. 2016c] mostram que modelos hipermídia devem ser incrementados com os seguintes recursos:

R1) **Especificação do conhecimento como um elemento de primeira classe**. A descrição do conhecimento deve ser realizada seguindo o design do modelo conceitual, permitir *reasoning*, e considerar o suporte à interoperabilidade, incluindo, por exemplo, o uso de RDF/RDFS, OWL, ou outra solução existente, como nós hipermídia;

R2) **Especificação de relações entre conhecimento e interfaces de nós hipermídia**, as quais podem representar: todo o conteúdo de um nó; intervalo de tempo de um conteúdo ou documento; coordenadas espaciais do conteúdo; segmentos; etc.;

R3) **Eventos de apresentação por meio de conhecimento**. Descrição de âncoras de apresentação por meio da especificação do conhecimento, permitindo que autores de documentos especifique, por exemplo, que um conjunto de nós deve ser apresentado sempre que conceitos específicos ocorrerem, de acordo com a proveniência da ocorrência dos conceitos (por exemplo, qual nó hipermídia);

R4) **Eventos de interatividade por meio de conhecimento**. Descrição de eventos de interatividade usando não apenas interfaces hipermídia tradicionais, mas também especificação do conhecimento. Por exemplo, permitindo que autores especifiquem que um conjunto de nós deve ser apresentado quando um usuário interagir com conceitos específicos, de acordo com a ocorrência dos conceitos em um nó hipermídia;

R5) **Suporte a reuso das especificações de conhecimento e conteúdo**. Permitir que autores de documentos reutilizem nós de conteúdo e nós que estejam relacionados com determinadas descrições de conhecimento, bem como importação e reuso de bases de conhecimento;

R6) **Suporte à inferência de conceitos a partir de conteúdo e bases de conhecimentos**. Permitir que autores de documentos especifiquem o comportamento de agentes cognitivos.

Note que hiperconhecimento abrange todos os desafios discutidos nesta seção. Mais ainda, ele envolve questões semelhantes à modelagem da ontologia, tais como estruturação do conhecimento e *reasoning*. A ideia é preencher a lacuna semântica estabelecida entre conteúdo multimídia e a descrição do seu significado, um desafio explorado em diversos trabalhos. De fato, existem na literatura dois grupos de soluções que tentam preencher a lacuna semântica. O primeiro grupo é composto por padrões e modelos de metadados (por exemplo: MPEG-7, Dublin Core, e PBCore) [Moreno et al. 2016a] e tem como objetivo permitir que entidades de interesse do usuário sejam encontradas por meio de buscas. Essas soluções definem como especificar descritores ou campos pré-definidos para descrever aspectos de baixo nível dos conteúdos de mídia (por exemplo: título, autor, descrição, etc.). Em contraste, o segundo grupo reúne soluções que visam descrever conceitos de alto nível e relações semânticas mais ricas (especificação de ontologias), com vocabulários de termos formalmente definidos, usualmente empregados por uma comunidade de usuários e máquinas em um domínio específico. RDF (*Resource Description Framework*), juntamente com a sua extensão semântica RDFS (RDF *Schema*) e OWL (*Web Ontology Language*) estão entre os mais utilizados para modelar relacionamentos entre dados e representar conhecimento estruturado.

Os dois grupos, entretanto, não oferecem recursos ou poder de expressividade para descrever conhecimento ou relações semânticas ricas entre conhecimento e conteúdo multimídia. No primeiro grupo, os metadados permitem a associação de descritores com o conteúdo em si, mas não há mecanismos para descrever conceitos de alto nível e relações semânticas mais ricas entre esses conceitos e diferentes conteúdos. No segundo grupo, frameworks e linguagens existentes para descrever ontologias são incapazes de oferecer uma integração total com conteúdo e especificações de documentos multimídia.

Em especial, as soluções mencionadas não oferecem recursos para lidar com os seguintes aspectos na autoria de documentos hipermídia. O primeiro aspecto consiste na especificação das relações entre a descrição do conhecimento e conteúdos multimídia na abordagem tradicional hipermídia. Ou seja, um autor do documento deveria ser capaz de especificar as relações entre descrição de conhecimento e as interfaces (âncoras e propriedades) dos nós hipermídia. O segundo aspecto é a especificação de eventos de interatividade por meio do conhecimento e não apenas com o conteúdo. Um terceiro aspecto ausente nas soluções é a especificação das relações para descrever extração e injeção de conhecimento, a partir de, e para os nós hipermídia descritos em um documento multimídia. Por último, mas um aspecto não menos importante, está a especificação de reuso de partes da definição de documentos multimídia por meio descrição de conhecimento.

Outra desvantagem das soluções mencionadas está também relacionada com o suporte a autoria de documentos. De fato, essas soluções obrigam o uso de diferentes tecnologias para especificar a semântica de apresentação do documento. A abordagem utilizada exige que autores de documentos saibam conceitos específicos sobre metadados ou linguagens de ontologia e frameworks. Mais ainda, é necessário que os autores dividam a especificação dos documentos (em, ao menos, especificação de relações entre conteúdos, especificação de documentos e de semântica), criando a possibilidade de existir inconsistências de conteúdo e semântica de apresentação, principalmente, em caso de futuras edições no documento.

Para criar o nosso modelo hiperconhecimento, fizemos a integração dos seis recursos em uma nova versão (3.1) do modelo NCM (*Nested Context Model*). O foco da versão anterior (3.0) está na representação de estruturas hipermídia. Mais especificamente, NCM 3.0 concentra esforços na especificação de relações de sincronismo espaço-temporal.

Entre os modelos conceituais hipermídia existentes, escolhemos estender NCM não só por sua importância histórica em hipermídia[5], mas, principalmente, para tirar proveito de sua capacidade de modelagem e apresentação multimídia. Assim, as extensões promovem um suporte integrado para descrever aspectos de conhecimento e de conteúdo, com uma distinção clara entre conceitos abstratos e instâncias de mídia, inerentes do modelo. NCM também tem prosperado por meio de uma de suas implementações. NCL (*Nested Context Language*) é uma aplicação XML projetada de acordo com NCM 3.0. NCL é parte das recomendações UIT-T para serviços IPTV, IBB (*Integrated Broadcast-Broadband*) e TV digital, bem como padrão ISDB-T (*International Standard for Digital Broadcasting - Terrestrial*).

As extensões NCM, que serão discutidas na Seção 7.3, contribuem não só para motivar avanços em modelos conceituais existentes e nas especificações da linguagem NCL, mas também como um modelo conceitual para a engenharia de documentos de hiperconhecimento. De fato, a escolha de explorar um modelo conceptual para especificar a nossa proposta baseia-se nos argumentos de trabalhos como o de Glushko e McGrath [Glushko and McGrath 2005], que defendem a necessidade de contribuições por meio de modelos conceituais como engenharia de documentos. Ou seja, em uma granularidade que sejam implementáveis.

7.3. Estudo de Caso

7.3.1. Conceitos Básicos da Computação em Nuvem

O constante aumento do acesso à web e disponibilidade de largura de banda permitiu a transformação do poder de computação em mercadoria [Brandao et al. 2016]. As empresas estão adotando gradualmente o paradigma da computação em nuvem, com o objetivo de diminuir os seus custos através da manutenção e processamento de dados fora de suas instalações. Comumente, os serviços de computação em nuvem se encaixam em três modelos diferentes que podem ser vistos como uma pilha. Eles são citados como: Infraestrutura como Serviço (*Infrastucture as a Service* – IaaS), a camada mais básica que oferece captação de recursos de hardware como serviço; Plataforma como Serviço (Platform as a Service – PaaS), que abstrai todas as necessidades de provisionamento de recursos, configuração e tempo de execução; e no nível mais alto, Software como Serviço (Software as a Service – SaaS), em que os usuários têm de lidar apenas com software de aplicação, bases de dados e outros serviços, deixando todos os outros aspectos a serem mantidos pelo provedor do serviço.

A aceitação de soluções de computação em nuvem no mercado tem crescido significativamente, não importa o nível de abstração do modelo de serviço. Soluções de IaaS, como os da SoftLayer[6] ou AWS[7], permitem que diferentes entidades, pequenas a

[5] NCM foi o primeiro modelo conceitual hipermídia a resolver a questão de contextos aninhados indicada no trabalho histórico de Halasz [Halasz 1998].
[6] www.softlayer.com
[7] www.aws.amazon.com

grandes empresas, terceirizem seus centros de operações. Ou seja, essas soluções permitem que as empresas evitem custos de aquisição de hardware e detalhes tradicionais de manutenção. Empresas que usam esse modelo de serviço podem assim concentrar sua mão de obra em ramos mais específicos e mais de acordo com seus negócios. No entanto, nessas soluções, os usuários ainda têm de lidar com detalhes de rede e provisionamento de serviço. Em soluções PaaS, como *IBM Bluemix*[8] e *Amazon Elastic Beanstalk*[9], o objetivo é ocultar esses detalhes de seus usuários, criando uma camada de abstração que permite que esses usuários se concentrem no desenvolvimento de aplicações ao invés de gerenciar detalhes de infraestrutura. Finalmente, soluções SaaS, como salesforce[10] e servicenow[11]) é um modelo em que a compra e utilização de software não estão relacionados com a aquisição de licenças, ao invés disso, os usuários pagam pelo uso de software sob demanda. O modelo SaaS tem sido incorporado na estratégia de quase todas as principais empresas de software.

Durante o minicurso, vamos desenvolver uma aplicação, conforme discutido na Seção 7.3, utilizando os recursos do modelo PaaS, que consiste em uma camada intermediária situada entre a camada de IaaS (com um nível de abstração mais baixo) e SaaS (nível de abstração mais elevado). A Figura 7.2.2 ilustra a disposição dessas camadas, indicando qual recurso que cada uma delas abstrai para seus usuários. Mais especificamente, infraestrutura de hardware no caso de IaaS, plataforma de desenvolvimento e operações em PaaSE e, finalmente, aplicações em camada SaaS.

Figura 7.2.2. Três principais modelos de serviços em nuvem

Dependendo do cenário desejado e de restrição políticas, existem diferentes tipos de estratégias para uso de PaaS. Em geral, elas são definidas como pública, privada, híbrida ou de comunidades. A estratégia pública é geralmente oferecida aos usuários na Web, num modelo pay-as-you-go. É também possível manter um PaaS utilizando clusters privados, com acesso restrito ao domínio de uma organização. Assim, é possível garantir um controle de acesso mais fino, em casos em que a segurança é considerada uma preocupação primordial. Como alternativa, um modelo híbrido é também comumente oferecido. Ao manter os dados confidenciais no local com acesso restrito, juntamente com uma infraestrutura mantida publicamente. Uma quarta estratégia é a utilização de PaaS em uma nuvem para comunidade. Ou seja, a infraestrutura é oferecida para uma comunidade específica, como usuários de uma ou mais organizações. Em tal estratégia, o

[8] www.ibm.com/Bluemix
[9] www.aws.amazon.com/elasticbeanstalk
[10] www.salesforce.com
[11] www.servicenow.com

PaaS pode ser gerido por uma ou mais organizações da comunidade, ou até mesmo por terceiros.

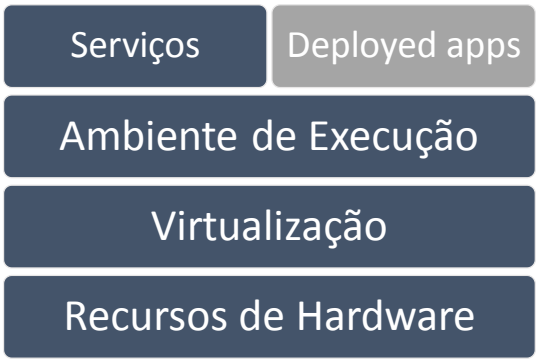


Figura 7.2.3. Recursos comumente oferecidos por abstrações em ambientes PaaS

Geralmente, o provedor de serviços PaaS está encarregado de definir os detalhes de como a infraestrutura opera, como seu sistema operacional, linguagens de programação, serviços e questões de gestão geral de desenvolvimento. Ferramentas adicionais e outros ambientes colaborativos podem ser fornecidas para melhorar a experiência dos usuários. A Figura 7.2.3 resume os recursos comumente abstraídas por PaaS, variando de hardware (por meio de virtualização e provisionamento de recursos), juntamente com ambientes de execução que permitam a execução de serviços e aplicações imantadas (*Deployed apps*).

7.3.2. IBM Bluemix

IBM Bluemix consiste em um PaaS com diversas funcionalidades e serviços, incluindo DevOps e um excelente catalogo de serviços. Esses recursos oferecidos pelo IBM Bluemix podem ser explorados por meio de uma interface web, no endereço bluemix.net.

A Figura 7.3.4 apresenta a tela inicial do IBM Bluemix, onde é possível encontrar diversas informações, bem como se autenticar ou criar um novo usuário.

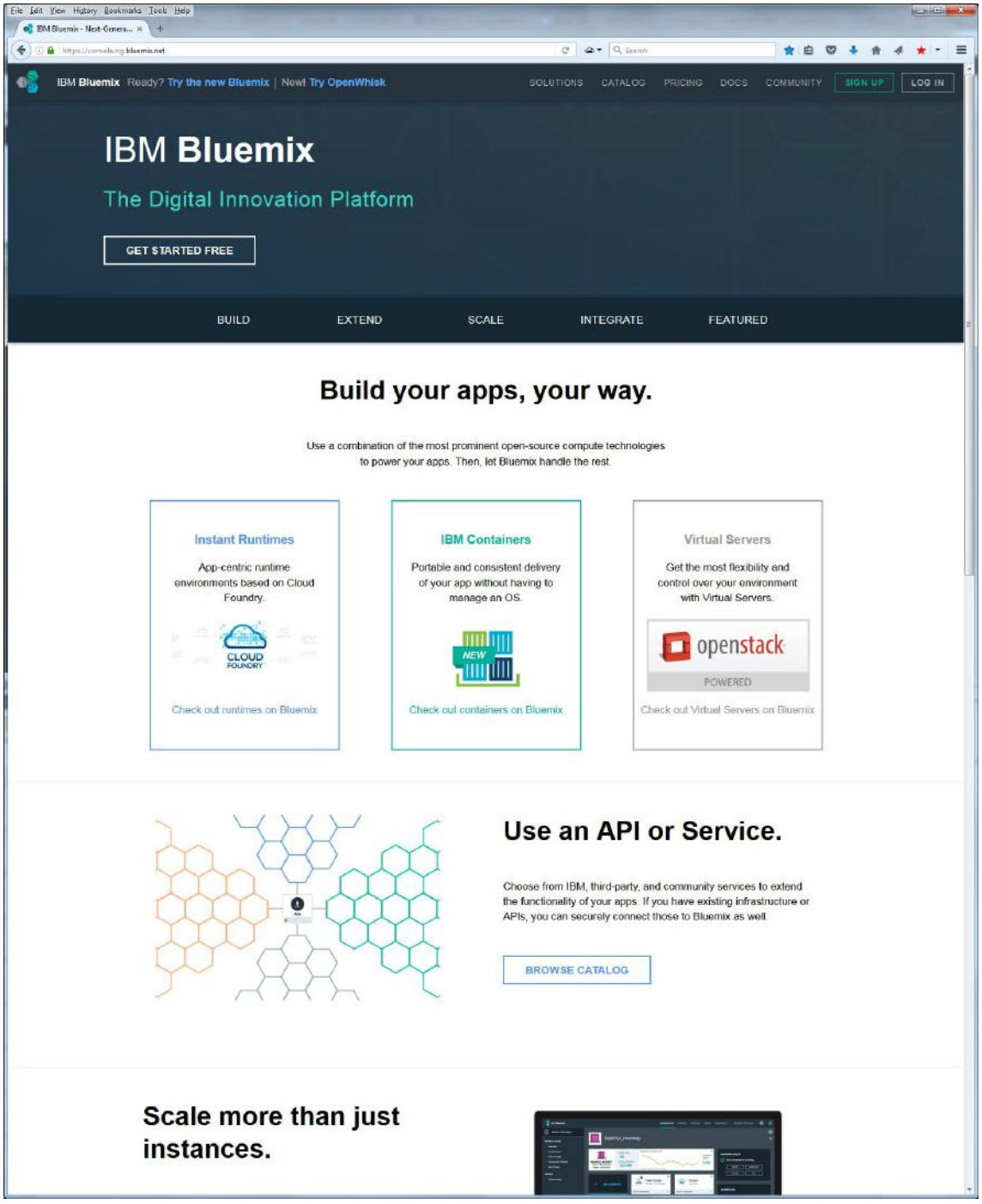


Figura 7.3.4. Pagina Inicial IBM Bluemix

A plataforma oferece um plano gratuito para que usuários testem suas funcionalidades. Para isso, basta que o usuário se registre entrando com algumas informações, como ilustrado na Figura 7.3.6. Após registrado, o usuário deve se autenticar para ter acesso ao sistema, como ilustrado na Figura 7.3.5.

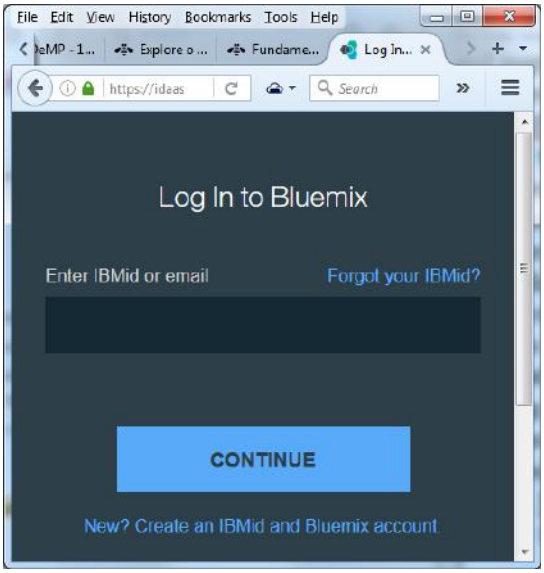


Figura 7.3.5. Login no IBM Bluemix

Caso desejado, o endereço direto para registro de novos usuários na plataforma é https://console.ng.bluemix.net/registration/

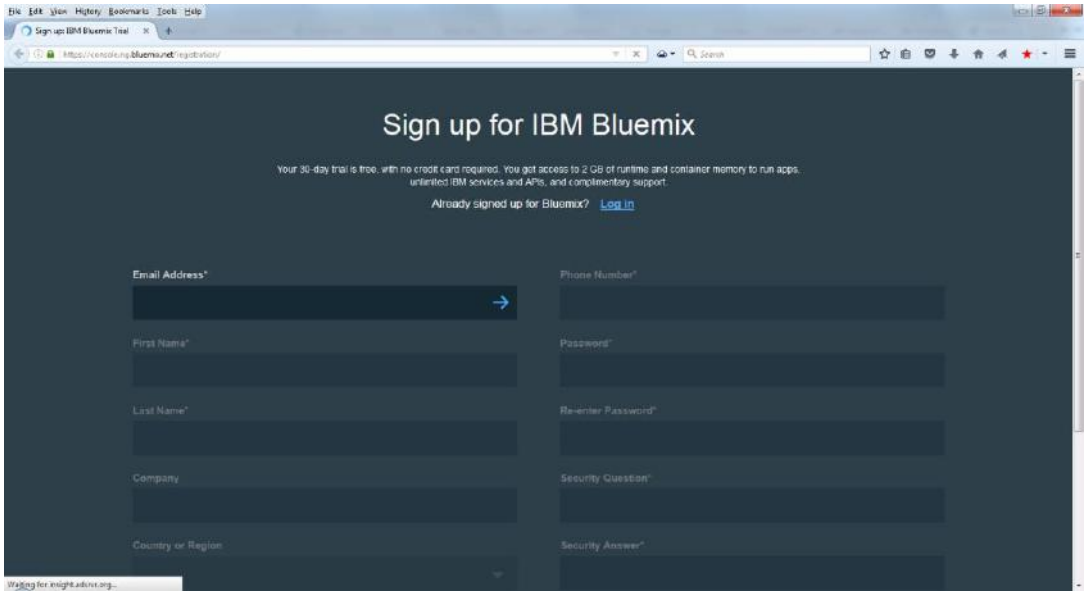


Figura 7.3.6. Criando uma Conta no IBM Bluemix

Uma vez autenticado, usuários passam a ter acesso a todas as funcionalidades do Bluemix. Incluindo *Dashboard*, em que é possível criar uma aplicação e, ainda, o *Catalog*, que apresenta a lista de serviços do Bluemix. O controle de quantos dias o usuário possui para experimentar o Bluemix é também apresentado, como ilustrado na Figura 7.3.7. A figura ilustra ainda, na parte superior da tela, os links para que o usuário acesse *Dashboard* e o *Catalog*.

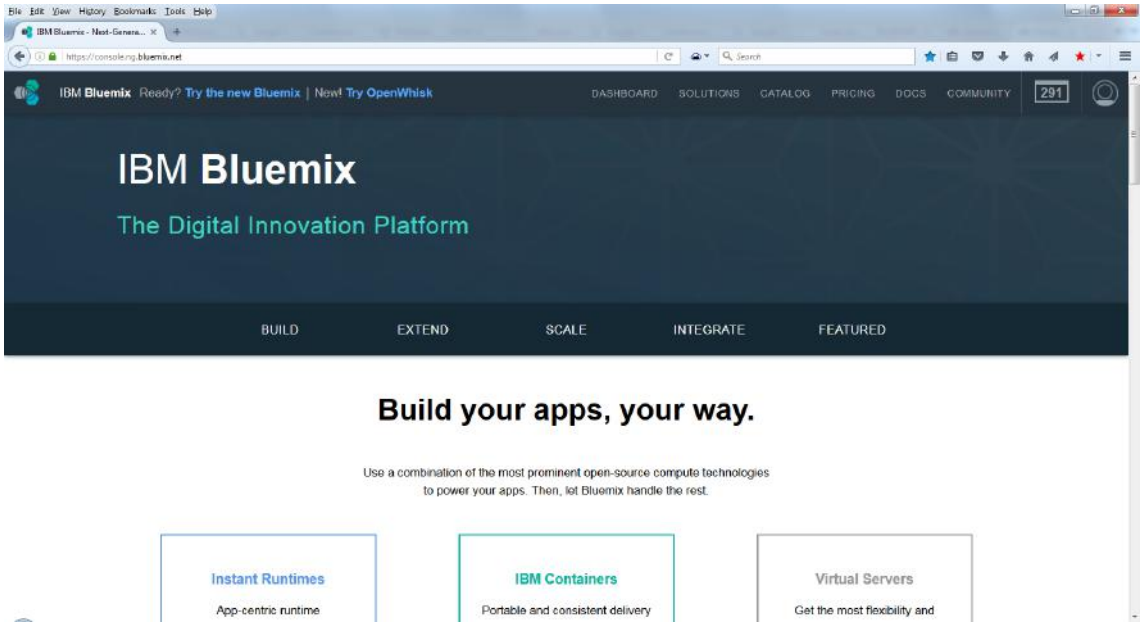


Figura 7.3.7. Usuário autenticado e ambiente oferecido

Ao clicar em *Dashboard* o usuário tem acesso ao controle das principais funções do Bluemix, como ilustrado na Figura 7.3.8. O *Dashboard* oferece abstrações para a criação de aplicações, serviços, containers e outros recursos para o usuário.

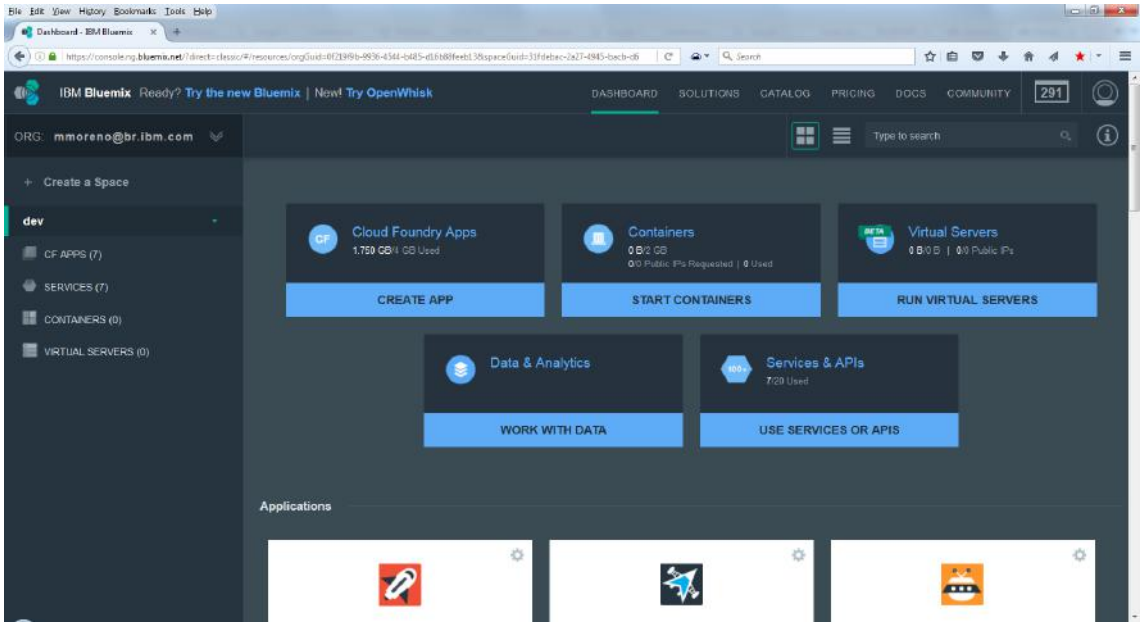


Figura 7.3.8. Painel de Controle

Ao clicar no link de *Catalog*, são apresentados diversos serviços em que usuários podem facilmente fazer uma seleção para acoplar em suas aplicações ou serviços. Cada serviço do catalogo possui um modelo de negócios específico. A maioria possui um modelo em que é possível experimentar gratuitamente o serviço permitindo, por exemplo, um determinado número de chamadas às API do serviço por dia. O endereço direto do catálogo de serviços do Bluemix é https://console.ng.bluemix.net/catalog/. Ao acoplar um serviço à sua aplicação, o Bluemix gerencia e apresenta suas credenciais para que o usuário faça acesso e autenticação ao serviço. Documentação e exemplos sobre uso das

APIs de cada serviço também são apresentadas e indicadas os endereços onde estão disponíveis.

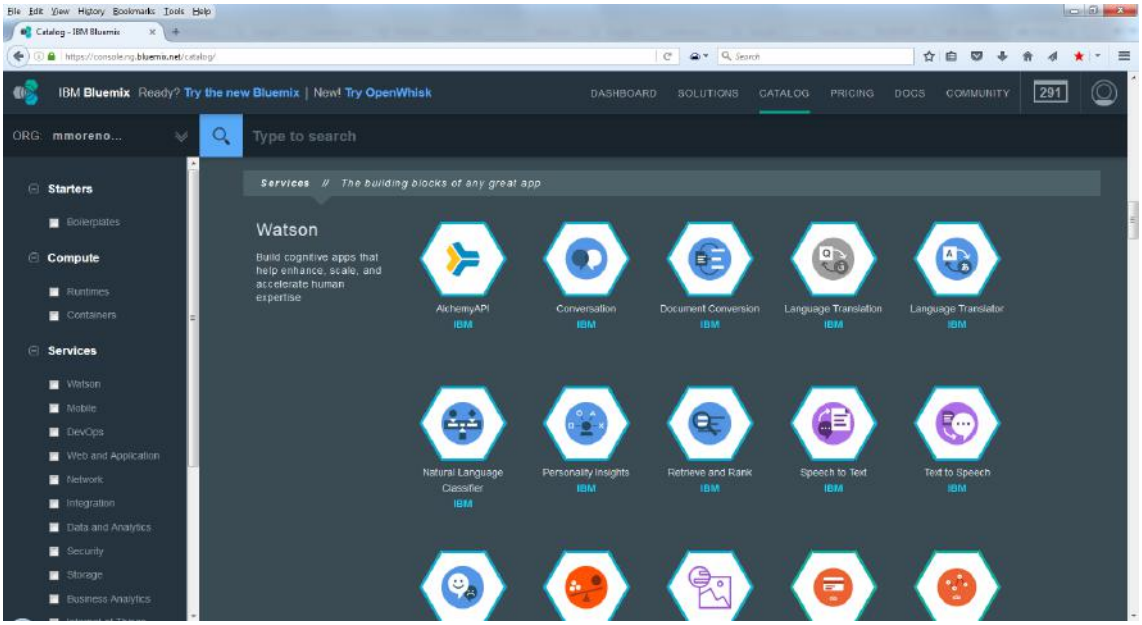


Figura 7.3.9. Catálogo de Serviços

Retomando o painel de funcionalidades (Dashboard) do Bluemix, ao clicar em criar uma aplicação (na Figura 7.3.8, CREATE APP), a tela ilustrada na Figura 7.3.10 é apresentada, permitindo que o usuário especifique o tipo de aplicação a ser criada, voltada para Web ou para dispositivos móveis.

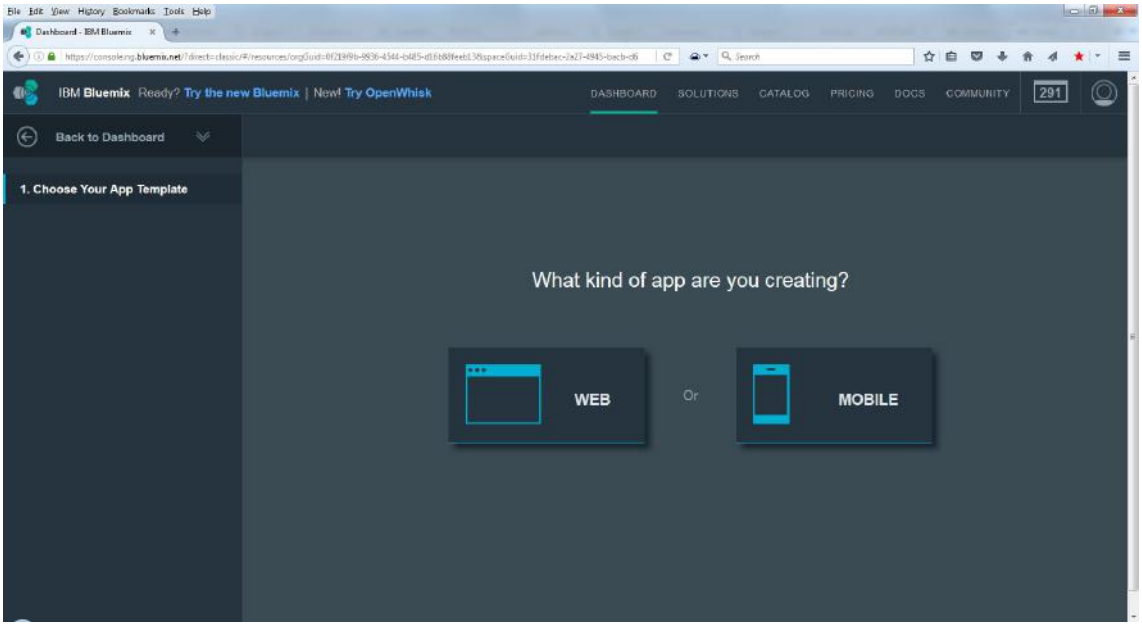


Figura 7.3.10. Criando uma Aplicação

Escolhendo uma aplicação Web, como será feito na aplicação do estudo de caso do minicurso, uma tela para escolha entre diversas linguagens de programação é apresentada, conforme ilustrado na Figura 7.3.11.

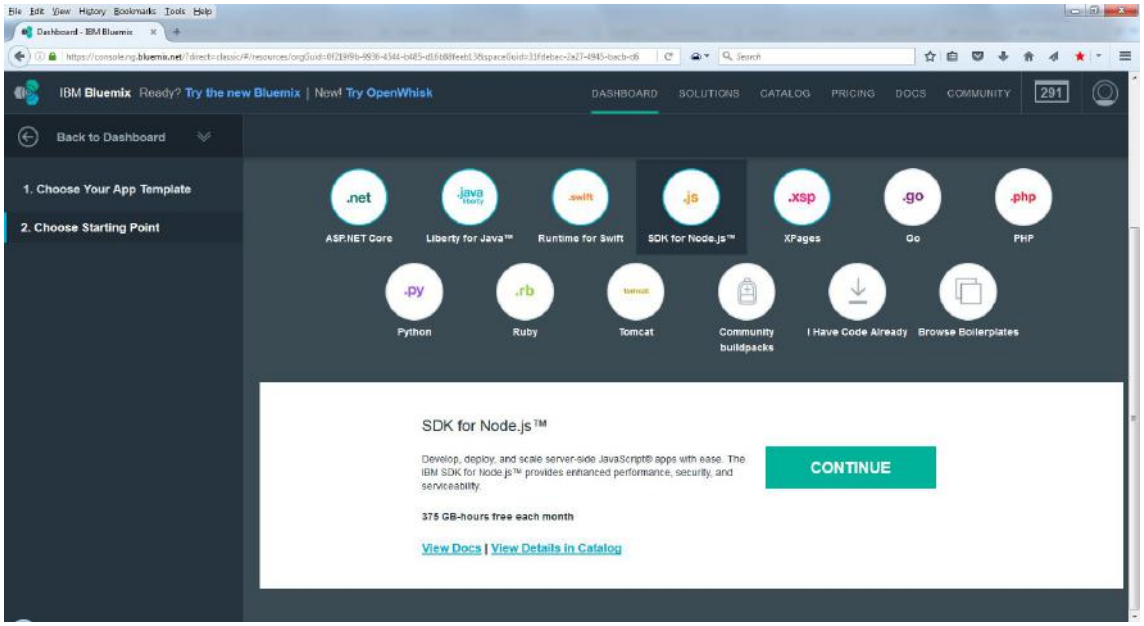


Figura 7.3.11. Escolhendo uma linguagem de programação

No minicurso, a aplicação de estudo de caso será especificada utilizando node.js. Após escolher a linguagem de programação, é necessário definir um nome para a aplicação. Para isso basta entrar com o nome no campo apresentado na Figura 7.3.12. Esse nome será utilizado na URL da aplicação implantada na PaaS. A aplicação criada pelos autores deste capítulo, por exemplo, é denominada mm4dm. Logo, sua URL é http://mm4dm.mybluemix.net

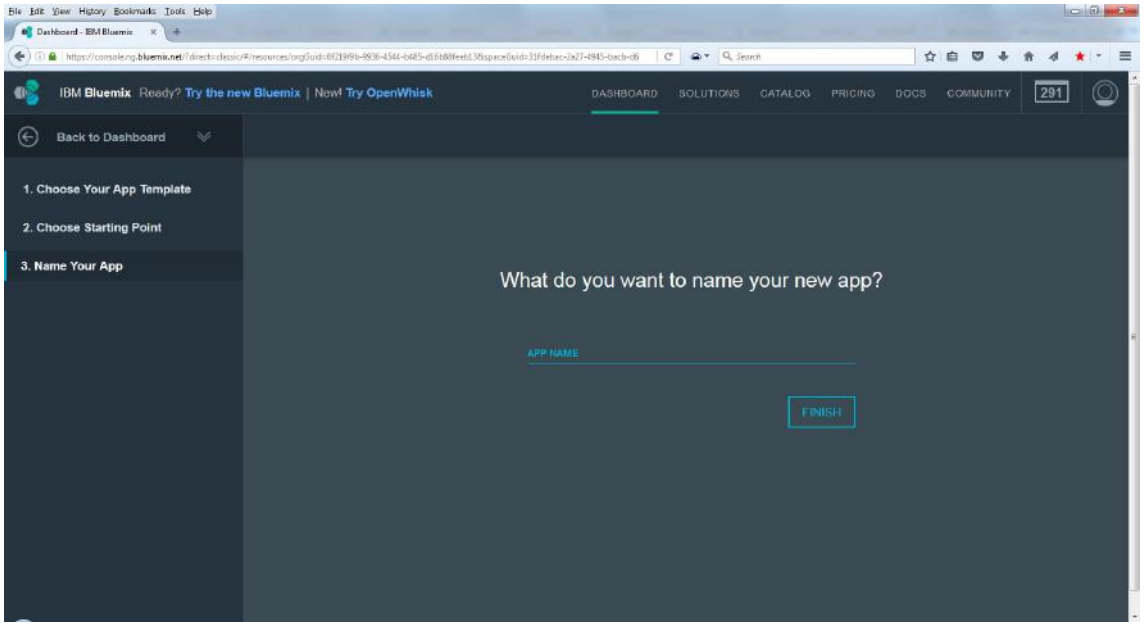


Figura 7.3.12. Batizando sua aplicação

De fato, essa URL estará disponível logo depois da faze de preparação de recursos, iniciada após a atribuição de um nome para a aplicação. A tela de preparação é apresentada na Figura 7.3.13.

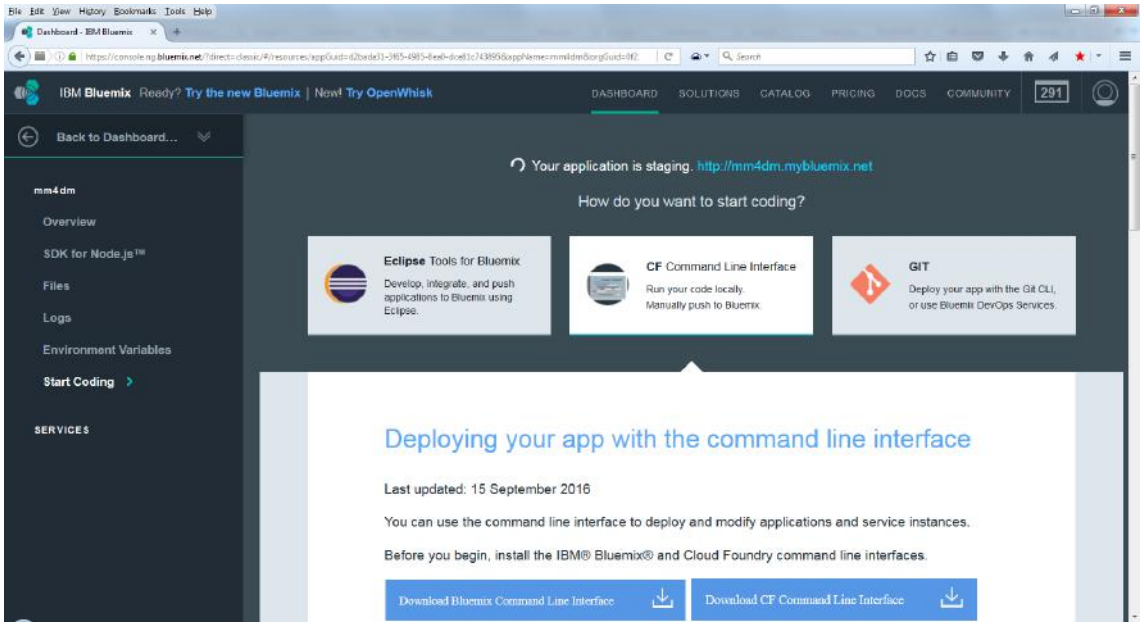


Figura 7.3.13. Criação de recursos para implantação da aplicação

Uma vez que todos os recursos da aplicação e seu ambiente foram preparados, a aplicação já foi implantada no PaaS e já está disponível para uso. Por exemplo, ao criar a aplicação do minicurso, podemos acessá-la imediatamente utilizando um navegador Web e entrando com o endereço da aplicação mencionado. Como não modificamos ou desenvolvemos nada na aplicação ainda, uma aplicação default estará disponível, como apresentado na Figura 7.3.14.

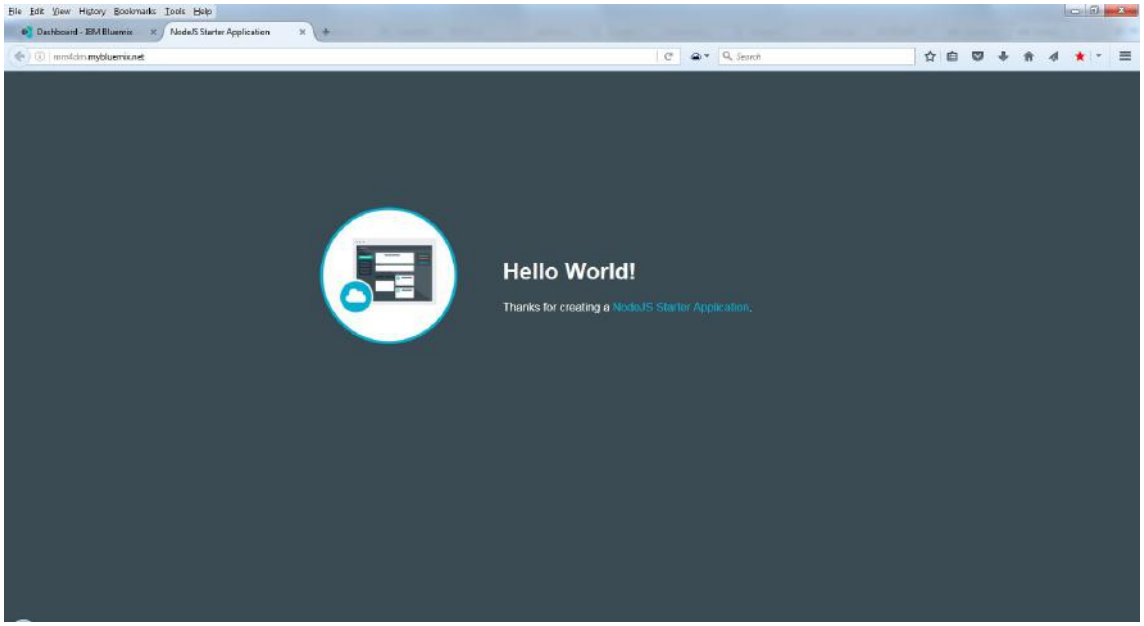


Figura 7.3.14. Aplicação default

Retornando ao ambiente do Bluemix, note que foi criado um painel de controle para a aplicação, ilustrado na Figura 7.3.15, e onde é possível especificar os recursos que devem estar disponíveis para a aplicação. Mais ainda, é possível criar um repositório para controle de versão para a aplicação.

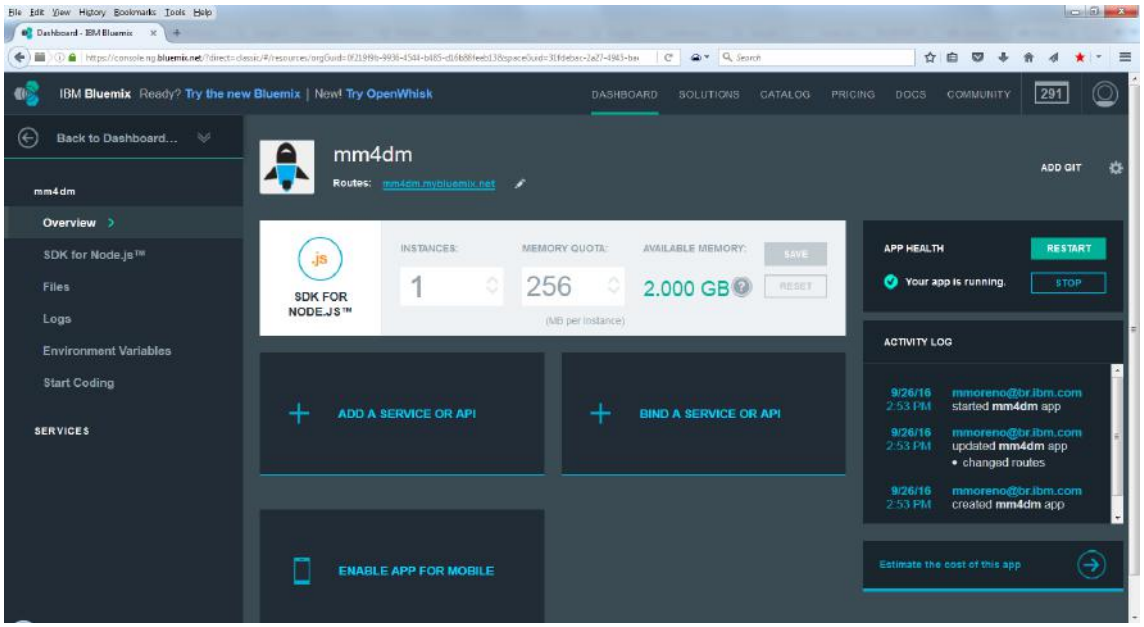


Figura 7.3.15. Painel de controle para aplicação

Para criar um repositório para a aplicação, basta clicar no link ADD GIT (na Figura 7.3.15, canto superior direito). O sistema então responde, como ilustrado na Figura 7.3.16, marcando como default a opção de criar o repositório com o código fonte da aplicação inicial com *hello world*.

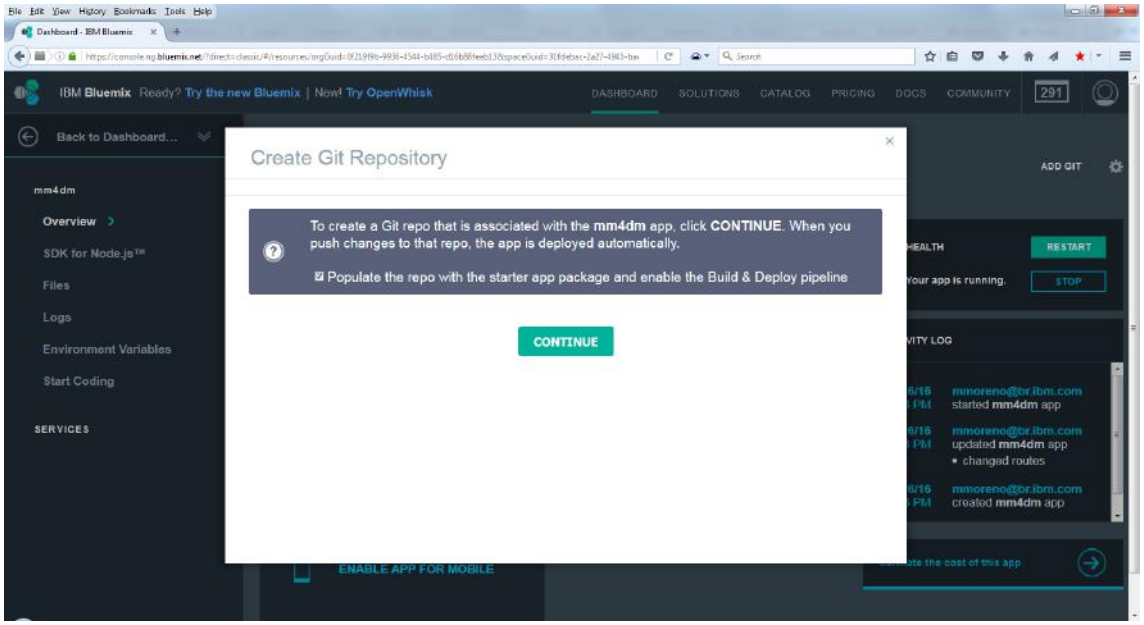


Figura 7.3.16. Criando um repositório git

No nosso estudo de caso, vamos deixar essa opção marcada, o que acarretará na criação do repositório git com uma aplicação inicial a qual tem uma estrutura e organização que irá facilitar o desenvolvimento de nossa aplicação. A Figura 7.3.17 apresenta a tela que o Bluemix exibe quando todo o processo foi executado corretamente.

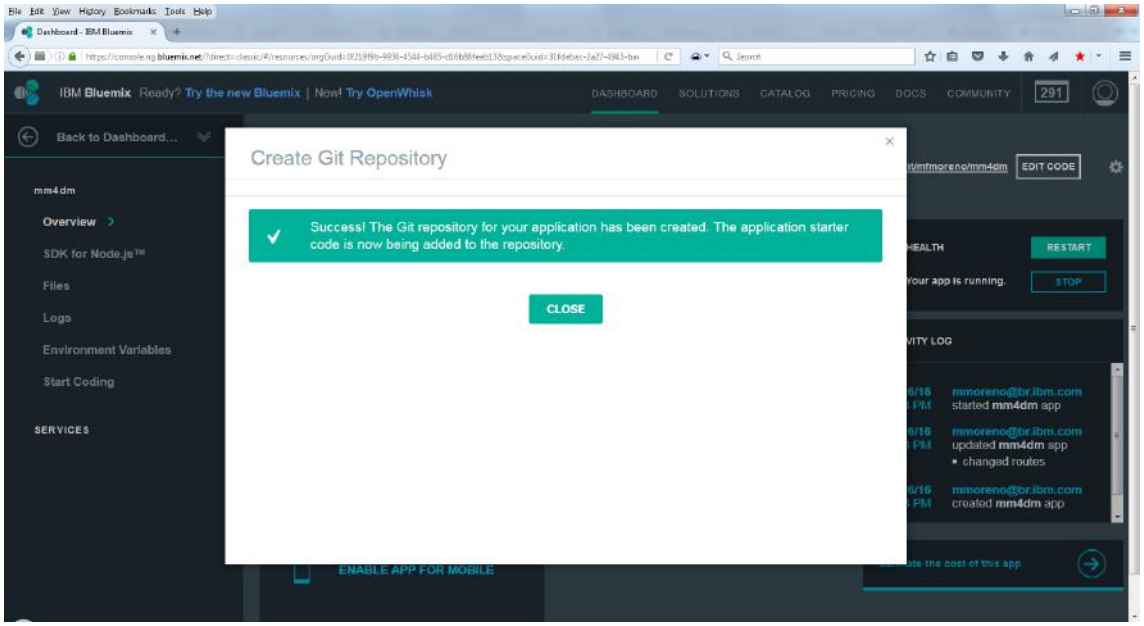


Figura 7.3.17. Git criado com sucesso

A partir desse momento, o ambiente de desenvolvimento do Bluemix, integrado com JazzHub, é oferecido ao usuário. Nesse ambiente, o usuário pode parar a aplicação, executar, checar os logs com o status da aplicação e utilizar diversos outros recursos, como apresentado na Figura 7.3.18. No minicurso, vamos explorar várias dessas funcionalidades do IBM Bluemix DevOps Services.

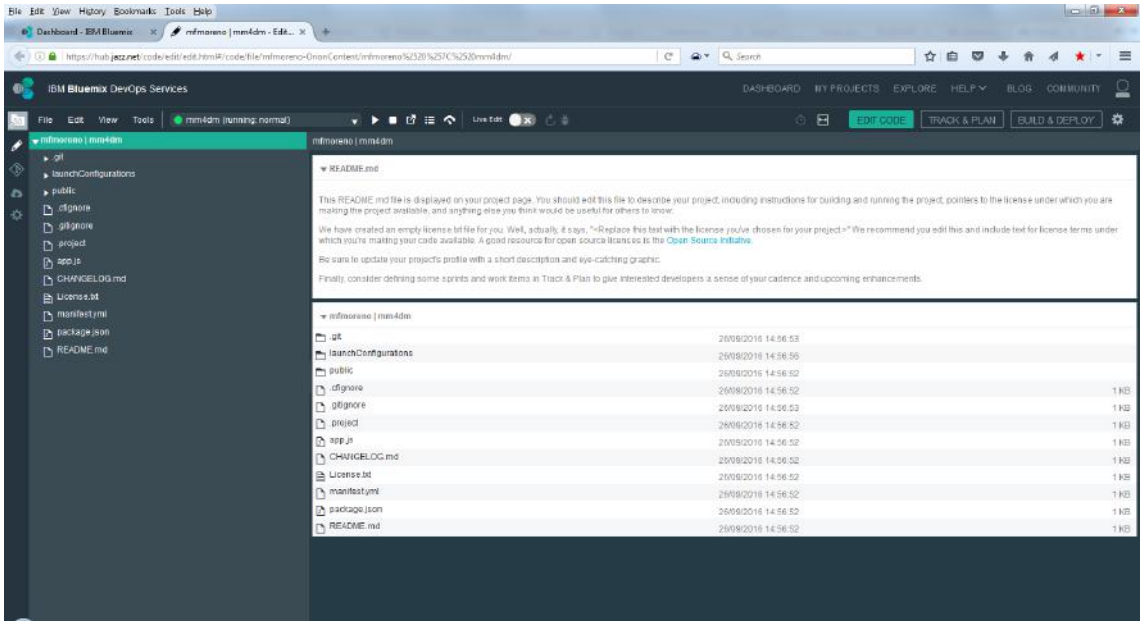


Figura 7.3.18. Ambiente de desenvolvimento e operações

Ainda na Figura 7.3.18, no canto superior direito da tela, o usuário pode clicar no link Build & Deploy. Nesse caso, o usuário pode definir qual é o pipeline de construção e implantação da aplicação. Por default, esse pipeline está integrado com o repositório git. O pipeline é executado sempre que o usuário executar um git commit e git push na branch máster do repositório.

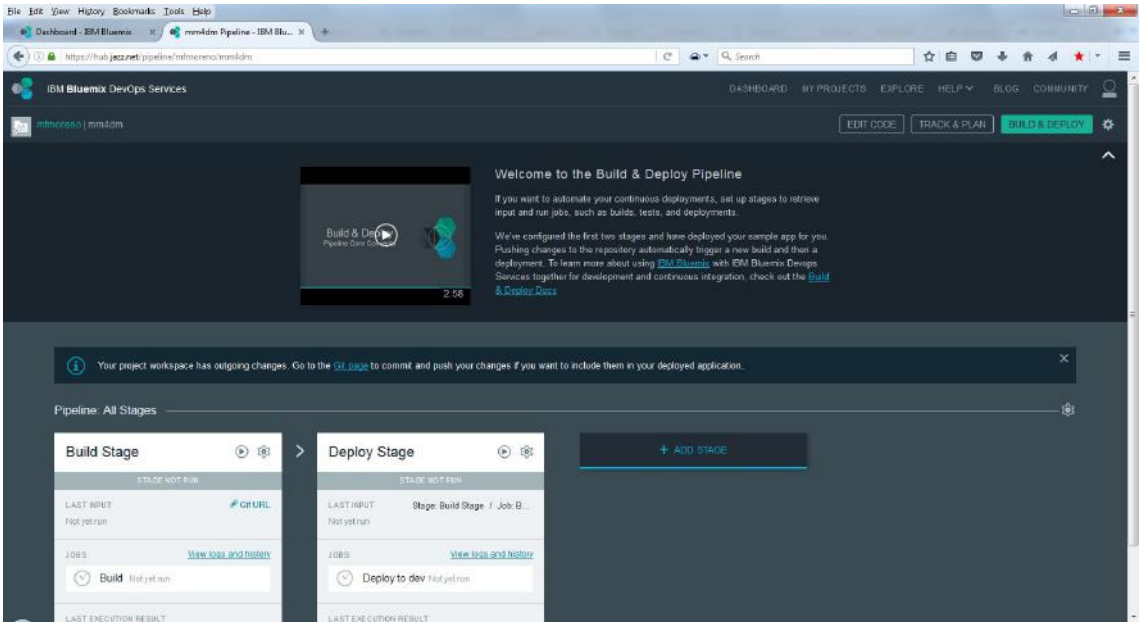


Figura 7.3.19. Pipeline de implantação da aplicação

Retornando ao ambiente de desenvolvimento, note, clicando no arquivo app.js que o código padrão da aplicação inicial está disponível para edições, o que iremos fazer durante o minicurso. A Figura 7.3.20 apresenta a tela correspondente ao momento em que app.js está sendo marcado para edição.

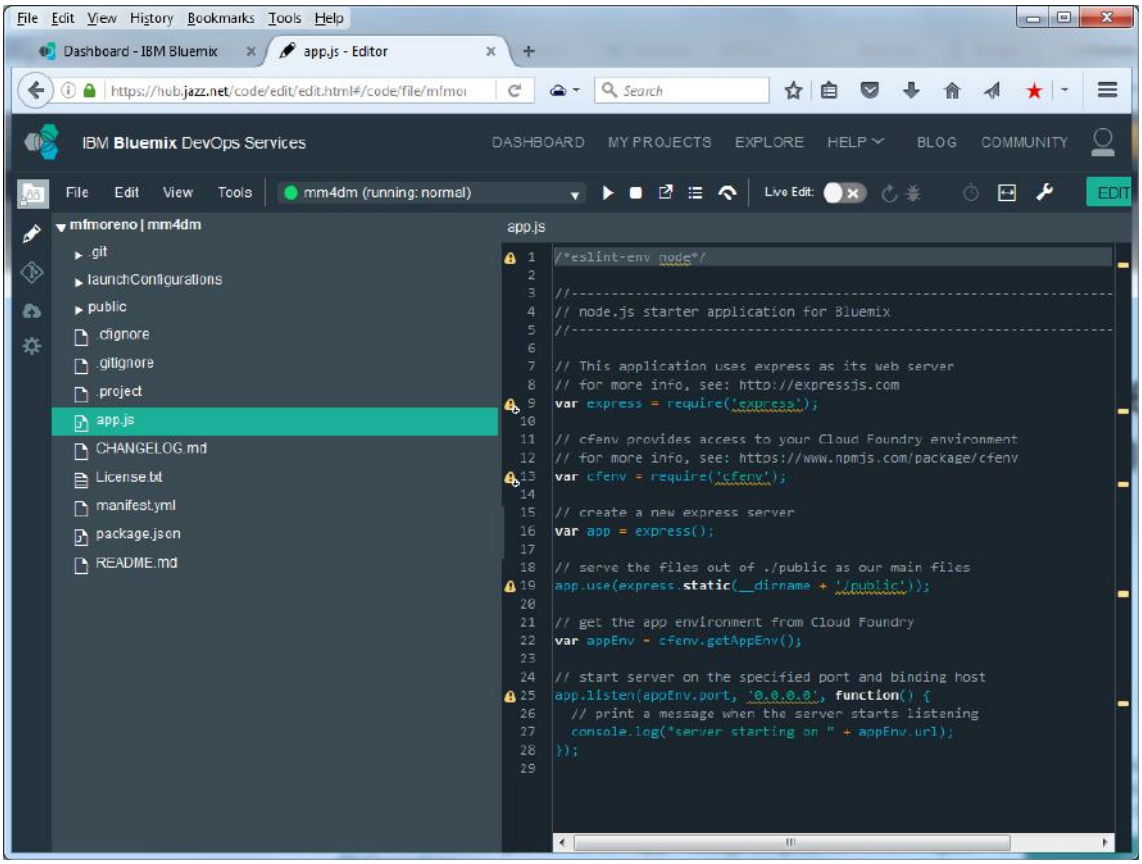


Figura 7.3.20. Desenvolvendo utilizando o ambiente do Bluemix

Outro arquivo que iremos utilizar durante o minicurso é o package.json, que permite ao usuário definir quais as dependências necessárias para a perfeita execução da aplicação. Além disso, é possível definir a versão do node necessária, quais os passos para iniciar a aplicação, bem como metadados da aplicação como, por exemplo, nome, versão, etc. A Figura 7.3.21 apresenta a tela do BlueMix DevOps Services no momento em que o arquivo package.json está marcado para edição.

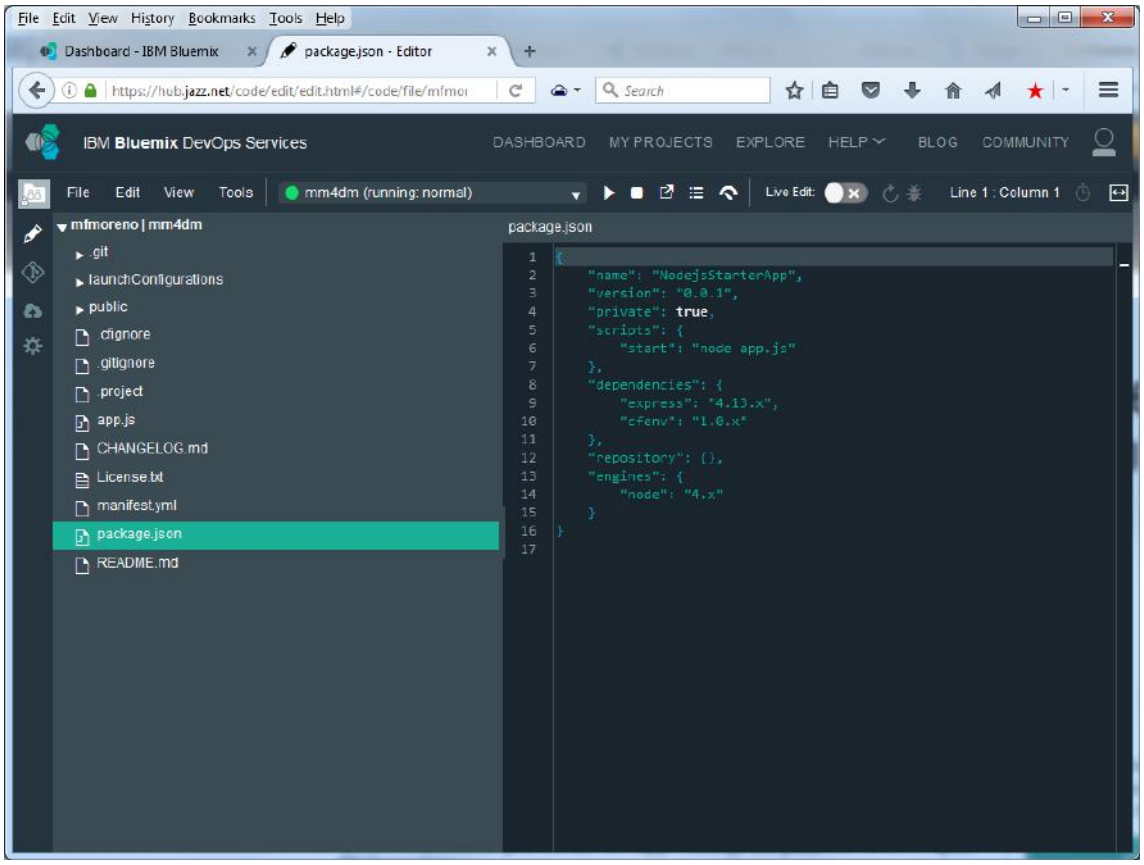


Figura 7.3.21. Definindo dependências e metadados da aplicação

Agora que sabemos como editar e acessar os arquivos da nossa aplicação, podemos retornar ao catálogo de serviços para selecionar um para utilizarmos na nossa aplicação. No nosso estudo de caso, um dos serviços utilizados será o de acesso às APIs do AlchemyAPI.

No catálogo de serviços, ao clicar no serviço do Watson AlchemyAPI, a tela apresentada pela Figura 7.3.22 é exibida. Note, na figura, que além de apresentar um resumo explicativo sobre os recursos que o serviço oferece, detalhes e opções sobre o modelo de negócios do serviço são apresentados para que o usuário selecione uma opção. Ainda na Figura 7.3.22, é possível observar que no instante de redação deste capítulo, o serviço AlchemyAPI oferece no plano gratuito o acesso a eventos de APIs por dia. Após selecionado o plano, é possível então acoplar o serviço à aplicação. Isso é realizado por meio de um clique no botão CREATE.

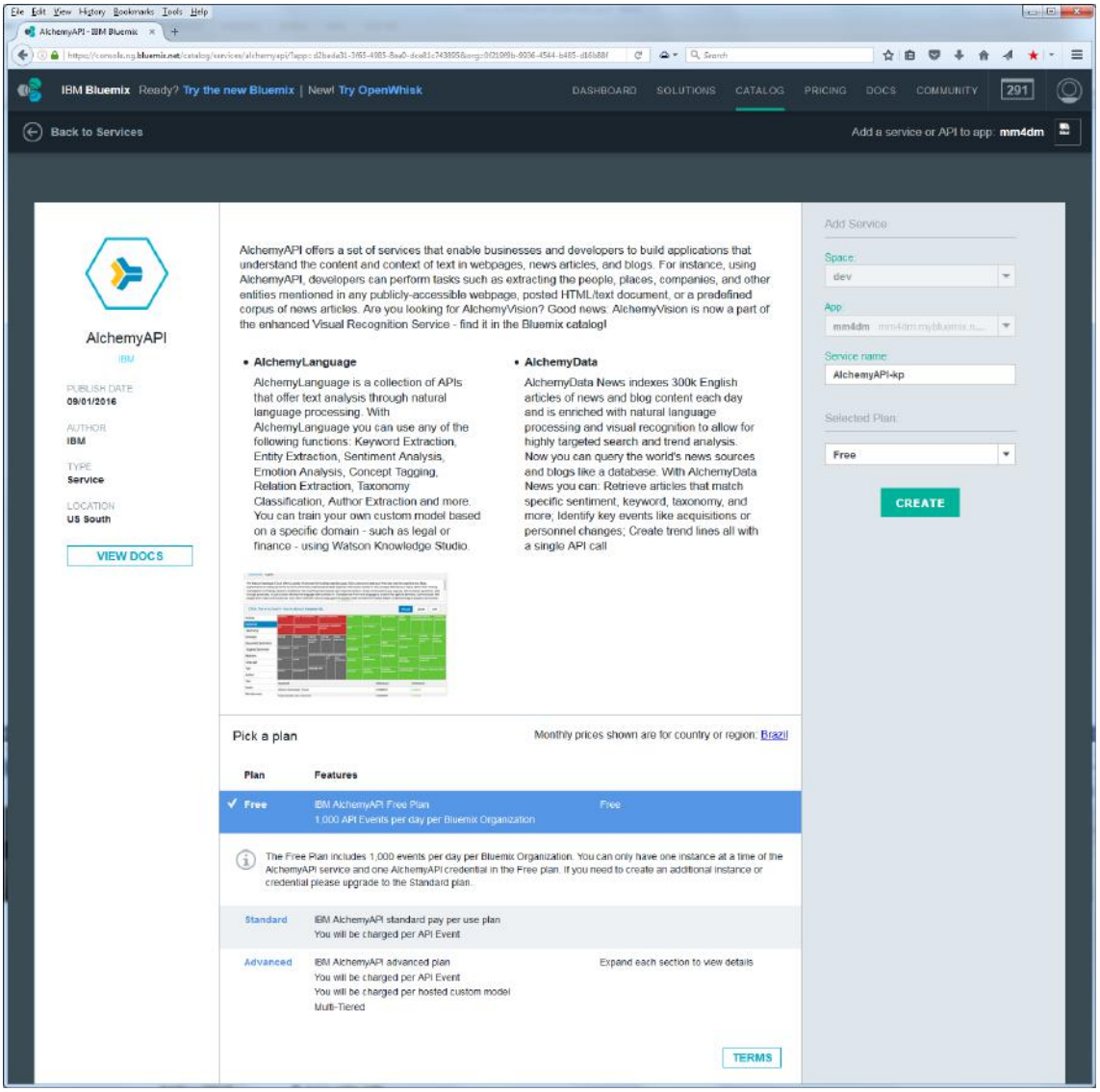


Figura 7.3.22. Serviços Watson por meio do AlchemyAPI

Nesse caso, todas as informações necessárias para autenticação e uso do serviço pela aplicação são criadas e mantidas pelo Bluemix. A Figura 7.3.23 apresenta uma aplicação em que diversos serviços estão acoplados. Ao clicar em um desses serviços, as informações com credenciais do usuário para acesso ao serviço são exibidas.

Essas são as informações básicas que iremos explorar no minicurso. Detalhes sobre a aplicação e todos os códigos fontes e outros serviços que iremos utilizar no nosso caso de estudo podem ser encontrados em mm4dm.mybluemix.net

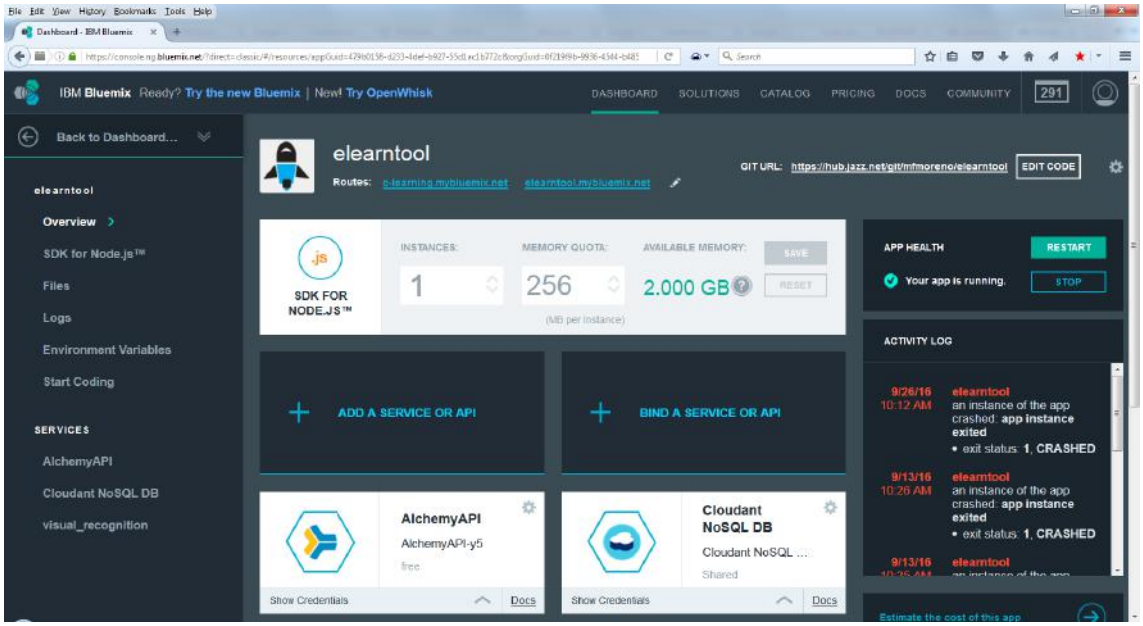


Figura 7.3.23. Aplicação integrada com AlchemyAPI

7.3.3. NCM 3.1: O primeiro modelo conceitual hiperconhecimento

Para discutirmos a nossa solução de modelo conceitual hiperconhecimento, esta seção primeiro introduz os conceitos básicos do NCM 3.0 para, depois, discutir nossas extensões, que visam enriquecer o modelo com suporte a aspectos de conhecimento. Essas extensões são parte da nova versão que propomos em nossos recentes trabalhos [Moreno et al. 2016a] [Moreno et al. 2016b] [Moreno et al. 2016c]. Mais especificamente, NCM 3.1.

NCM 3.0

O objetivo principal é permitir a descrição de atividades e práticas envolvendo conteúdo multimídia, agentes cognitivos e suporte a tomada de decisão. Para isso, foram incorporados no modelo os seis recursos discutidos na Seção 7.2.

O modelo NCM é baseado nos conceitos tradicionais hipermídia de nós e elos (Link). O primeiro representa fragmentos de informação, enquanto o último tem o objetivo de definir os relacionemos entre interfaces (âncoras e propriedades) dos nós. Existem duas classes básicas de nós: nó de conteúdo e nó de composição. Um nó de conteúdo representa os objetos de mídia usuais (imagem, texto, vídeo, áudio, etc.), enquanto o nó de composição é um nó NCM cujo conteúdo é um conjunto de nós (formado por nós de composição ou conteúdo). Por sua vez, um nó de contexto é um nó de composição NCM que pode conter ainda um conjunto de links e outros atributos. Nós de contexto são úteis, por exemplo, para definir uma estrutura lógica para documentos hipermídia.

A Figura 7.3.24 apresenta um diagrama com as principais classes que compõem um *Link*, que tem basicamente dois atributos: um *Connector* e um conjunto de *Binds* (*bindSet*). A entidade *Connector* define a semântica de uma relação por meio de uma cola (classe *Glue*), independente dos componentes que serão incluídos nessa relação, e um conjunto de pontos de acesso, chamados de papéis (*Roles*). Voltando ao conjunto de *Binds* do *Link*,

cada *Bind* associa uma extremidade do elo (interfaces dos nós) a um *Role* no conector referenciado pelo *Link*.

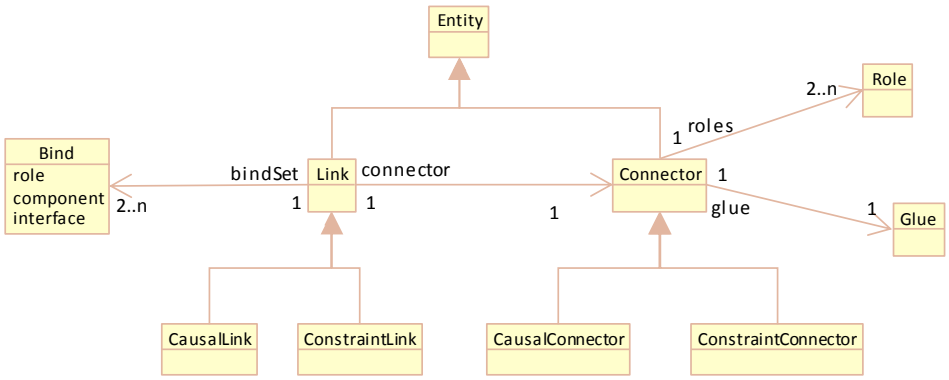


Figura 7.3.24. Diagrama de Classes com foco nas entidades Link e Connector

Teoricamente, conectores podem representar qualquer tipo de relação. No entanto, a versão 3.0 do NCM concentra esforços na especificação de relações de sincronismo espaço-temporal por meio de dois tipos de conectores, causal e de restrição. Um conector causal é denominado *CausalConnector* e possui uma cola causal, capaz de sustentar papéis de condição, de avaliação e de ação; enquanto conector de restrição, denominado *ConstraintConnector*, possui uma cola de restrição capaz de sustentar apenas papéis de avaliação. Links que referenciam conectores causais são chamados *CausalLinks*, enquanto *ConstraintLinks* referenciam conectores de restrição. A Figura 7.3.25 ilustra um exemplo em que esses dois tipos de conectores são utilizados.

No exemplo, um nó de contexto possui uma interface (porta "p1") para uma âncora do nó de conteúdo "Filme", indicando que a apresentação do nó de contexto deve ser iniciada com a apresentação do nó de conteúdo "Filme". Note, na Figura 7.3.25, que a âncora do "Filme" é identificada por λ. Todos os nós em NCM são criados com essa âncora, que representa todo o conteúdo do nó (e.g. intervalo de tempo definido pelo início e fim natural do conteúdo).

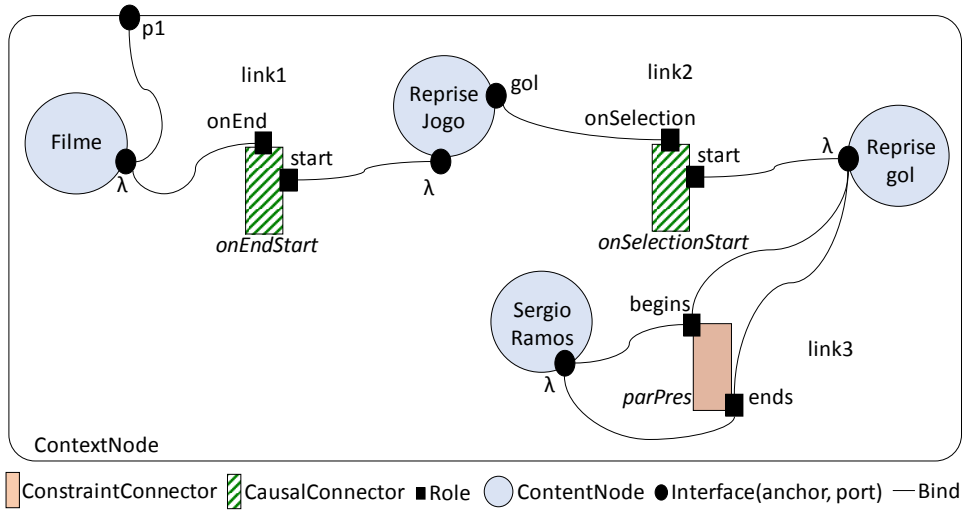


Figura 7.2.25. Exemplo de descrição com diferentes tipos de relações

Por um lado, uma condição deve ser satisfeita numa relação causal para executar um grupo composto por uma ou mais ações. No exemplo da Figura 7.3.25, "link1" faz

referência ao conector causal "*onEndStart*" para especificar que a reprise da semifinal da *Champions League* ("Reprise Jogo") seja apresentada (papel de ação *start*) assim que a apresentação do nó "Filme" termine (papel de condição *onEnd*). De forma similar, "link2" faz referência ao conector "*onSelectionStart*" para determinar que uma reprise do gol (nó de conteúdo "Reprise gol") seja apresentada se o usuário interagir (papel de condição *onSelection*) com a âncora temporal "gol" do nó de conteúdo "Reprise jogo". A âncora temporal "gol" possui especificado os instantes de início e fim do gol que ocorreu durante a semifinal da *Champions League*.

Por outro lado, nas relações de restrição não há causalidade envolvida. No exemplo, o elo "link3" referencia o conector de restrição "parPres" para definir que os dois nós "Reprise gol" e "Sergio Ramos" devem começar (papel de avaliação *begins*) e terminar (papel de avaliação *ends*) a sua apresentação ao mesmo tempo. Ambos os tipos de conectores são impecáveis para especificar relações entre os nós de conteúdo usuais que compõem um documento hipermídia, mas são incapazes de oferecer suporte aos recursos definidos na Seção 7.2 deste capítulo.

NCM 3.1

Para permitir a especificação de relações e relacionamentos capazes de se conectar conhecimento e conteúdo, as extensões do modelo afetam principalmente as entidades *Link* e *Connectors*, bem como as classes *Glue* e *Role*. Uma nova subclasse de nó, denominada nó de conhecimento, foi criada para encapsular informações conhecimento e agregar suporte ao recurso R1 (conhecimento como entidade de primeira classe, ver Seção 7.2), permitindo que o novo nó seja utilizado nas relações NCM. Além das interfaces e atributos usuais dos nós hipermídia, o novo tipo de nó também define o atributo *concept* para especificar o conceito em que a instância do nó está representando. Além disso, um nó de conhecimento pode fazer referência a uma descrição de conceito, que pode ser realizada utilizando conceitos simples como no atributo conceito ou referindo-se ao conteúdo especificado por soluções existentes (por exemplo um URL de um OWL ou URI de RDF/RDFS). Essa é, de fato, uma vantagem que vale a pena destacar. NCM apenas define como os nós são estruturados e relacionados. O modelo não restringe ou prescreve os tipos de conteúdo compatíveis com seus nós. Em outras palavras, NCM 3.1 não só suporta a modelagem do conhecimento, mas também é um modelo capaz de especificar como as soluções existentes podem ser integradas e orquestradas. A Figura 7.3.26 ilustra um exemplo em que o nó de conhecimento "Real Madrid" define não só esse conceito, mas segundo uma estruturação descrita em uma URI RDF por meio de um atributo NCM 3.0 denominado src [Moreno et al. 2016b].

Note que, com essa funcionalidade, NCM 3.1 permite uma interoperabilidade avançada com as soluções existentes. Isso porque RDF [Moreno et al. 2016b] e OWL [Moreno et al. 2016b] apenas descrevem relações binárias, que podem conectar um participante a outro participante ou a um valor. Por exemplo: Sergio Ramos é um jogador de futebol; Sergio Ramos é um defensor. No entanto, existem alguns casos (como o do exemplo acima mencionado), em que o método habitual e adequado para representar conceitos é utilizar relações que podem conectar um participante a mais de um participante ou valor. Essas relações são chamadas relações n-árias [Moreno et al. 2016b], que são, originalmente, suportadas por NCM. O consórcio W3C tem concentrado esforços em extensões para que RDF e OWL suportem a descrição de relações n-árias. Usando essas

relações, o exemplo poderia ser mais conciso e natural, como o seguinte: Sergio Ramos é um jogador de futebol e joga como zagueiro.

O exemplo da Figura 7.3.26 ilustra a utilização das novas estruturas definidas no NCM 3.1. De fato, a Figura 7.3.26 ilustra a utilização das nossas extensões por meio de um exemplo que será discutido ao longo de toda esta seção. A ideia do exemplo é a seguinte: durante a apresentação da partida da final da *Champions League*, cada vez em que um jogador marcar um gol, um usuário pode interagir com a apresentação para assistir reprises do gol e informações sobre o jogador que marcou o gol.

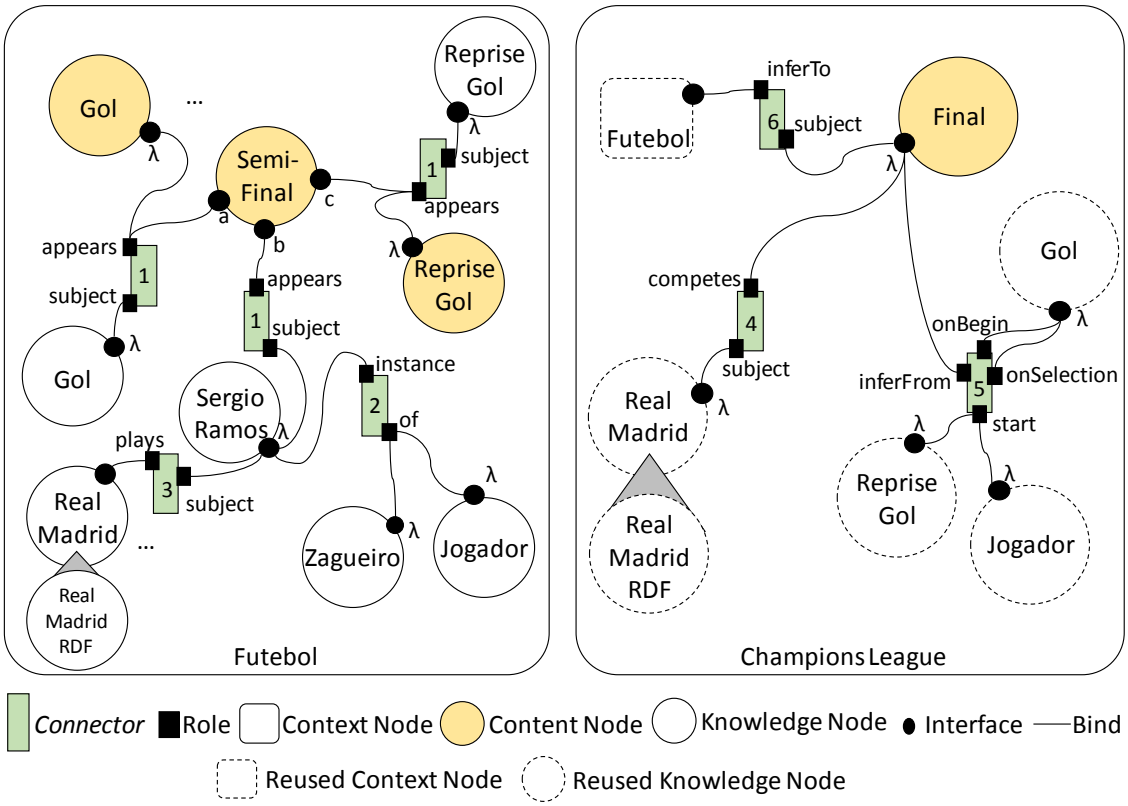


Figura 7.2.26. Exemplo de descrição com diferentes tipos de relações

NCM 3.1 explora o conceito de nós de contexto para aplicar diferentes semânticas a uma coleção de nós. Por exemplo, o conceito "Madrid" no contexto "Futebol" pode significar diferentes times de futebol (e.g. Real Madrid e Atlético de Madrid), mas em outro contexto poderia indicar uma cidade localizada na Espanha. Esse é um aspecto do NCM 3.1 que mostra o suporte ao recurso R5 (reuso). No exemplo da Figura 7.3.26, dois nós de contextos foram especificados. O nó de contexto "Futebol" consiste em uma base de conhecimento NCM, e descreve conceitos relacionados ao contexto “Futebol”. O contexto "Champions League" faz reuso do contexto "Futebol" para estar ciente dos conceitos ali descritos. Assim, esses conceitos podem ser reutilizados em relações com outros conceitos ou mesmo com conteúdo.

Geralmente, a descrição de conhecimentos é feita por meio de dois tipos de relações. O primeiro tipo é o tradicional SPO (sujeito-predicado-objeto) como, por exemplo, no Sujeito "Sergio Ramos", o Predicado "joga" e o objeto "Real Madrid". O segundo tipo pode ser definido como uma relação de hierarquia, como, por exemplo, em "Sérgio Ramos" é um “jogador” “zagueiro”. Para representar os dois tipos de relações, NCM 3.1

introduz dois novos tipos de conectores: 1) Conector de hierarquia, com cola e papéis de hierarquia; e 2) Conector de conhecimento, com cola e papéis de conhecimento. Na mesma direção, as extensões do modelo também definem um link de hierarquia e um link de conhecimento, que correspondem a links que fazem referência a, respectivamente, conectores de hierarquia e de conhecimento. O exemplo da Figura 7.3.26 ilustra o uso desses dois novos tipos de conectores (conectores de conhecimento identificados como "1" e "3", e conector de hierarquia identificado como "2"), que utilizam três novos tipos de papel: papel de hierarquia, papel de sujeito, e papel de objeto.

O papel de hierarquia representa uma hierarquia usando o conceito NCM de *classType*, que, nesse papel, define a função do participante ("pai" ou "filho") na relação. No exemplo da Figura 7.3.26, o conector de hierarquia tem dois papéis para definir que "Sergio Ramos" é um "jogador" e joga como um "zagueiro": o papel "instance" foi definido com o tipo "filho", e o papel "of" foi definido com o tipo "pai ".

Papéis de sujeito e objeto representam, respectivamente, um sujeito e um objeto como nas relações SPO. Voltando ao exemplo da Figura 7.3.26, o participante "Sérgio Ramos" é um sujeito em uma relação por estar ligado a um papel de sujeito, e está relacionado com o participante "Real Madrid", que é um objeto nesse relacionamento por usar um papel de objeto, denominado "plays". Note que os nomes dos papéis de objeto possuem semântica, atuando como predicados.

Com as novas estruturas que oferecem suporte à descrição de conhecimento e a descrição de relações de conhecimento, o suporte ao recurso R1 está completo. O próximo passo foi integrá-las com as interfaces (âncoras e propriedades) dos nós de conteúdo. Esse passo faz parte do suporte ao recurso R2, que foi realizado nas extensões do modelo ao permitir que os nós de conteúdo sejam usados nos papéis de conhecimento e de hierarquia. Na Figura 7.3.26, os conectores "1" e "4" são referenciados por links que exemplificam os relacionamentos das extensões do modelo que permitem representar conceitos por meio de artefatos visuais:

1) O conceito "Gol" aparece (*appears*) em um nó de imagem (subclasse de nó de conteúdo) identificado como "Gol" e em uma âncora espacial “a” do nó de vídeo "Semifinal". Essa âncora define coordenadas (x, y, largura e altura) do conteúdo do nó de vídeo; o mesmo tipo de relação foi especificado com "Sergio Ramos" e "Reprise Gol" e as interfaces de "b" e "c" da "Semifinal";
2) O conceito "Real Madrid" está ligado ao papel compete (*competes*) e tem relação com uma âncora "λ" do nó de vídeo "Final".

Com o suporte para descrever as relações entre conhecimento e conteúdo definido, podemos movemos nosso foco para estender o modelo para suportar as funcionalidades relacionadas aos recursos R3 (eventos de apresentação por meio de conhecimento) e R4 (eventos de interatividade por meio de conhecimento). Para isso, NCM 3.1 estende tanto a cola causal quanto a cola de restrição existentes, com o objetivo de permitir que os novos papéis de conhecimento e hierarquia sejam descritos nesses tipos de relação. No exemplo da Figura 7.3.26, o conector causal "5" é usado em um relacionamento para especificar que cada vez que a apresentação do conceito "Gol" ocorre (papel de condição estendido *onBegin*), o usuário pode interagir com a apresentação do conceito "Gol" (papel de condição estendido *onSelection*) para iniciar (papel de ação estendido start) a apresentação da reprise do gol e a respectiva informação do jogador que marcou o gol.

Note, na Figura 7.3.26, que o conector "5" especifica ainda o papel "*inferFrom*", que é um exemplo do quarto tipo de papel criado no NCM 3.1: o papel de inferência. O papel de inferência completa as funcionalidades necessárias para oferecer suporte aos seis recursos especificados na Seção 7.2. Mais especificamente, ele faz parte do suporte ao recurso R5, e indica qual participante na relação deve ser considerado para inferir os dados de acordo com uma apresentação do conhecimento. O papel de inferência possui um atributo para definir a direção de inferência de ("from") ou para ("to") o participante. O link que faz referência ao conector "5" especifica que a ocorrência da apresentação do conceito "Gol" deve ser inferida a partir (“from”) do nó de vídeo "Final".

De fato, um passo crítico na autoria de documentos multimídia que descrevem relações de sincronismo intermídia, como o modelado no exemplo da Figura 7.3.25 (veja nas discussões sobre NCM 3.0), é determinar os valores temporais de interesse para criar âncoras temporais. Na verdade, esse passo envolve um humano monitorando os instantes em que os eventos de interesse ocorrem no conteúdo, o que pode ser uma tarefa demorada, cansativa e propensa a erro. Além disso, se o conteúdo for editado de uma forma que os instantes de ocorrência dos eventos mudarem, todas as tarefas envolvidas nesse passo devem ser repetidas. O papel de inferência do NCM 3.1 permite que o autor de documentos automatize as tarefas de descoberta de instantes de interesse e criação de âncoras para um sistema computacional cognitivo, capaz de inferir tais eventos a partir de descrições presentes em bases de conhecimento ou no próprio documento.

Outra questão é a repetitividade do propósito dos relacionamentos. No exemplo, para ter o efeito de apresentar a reprise do gol toda vez que o respectivo gol ocorrer no nó de conteúdo "Final", é necessário não apenas criar novas âncoras e extrair informações temporais do fluxo de vídeo "Final", ou mesmo disparar as ocorrências em tempo real (usuais em eventos ao vivo), mas também deve especificar novos links que descrevem os relacionamentos com as âncoras criadas. NCM 3.1 resolve ambas as questões com um documento hiperconhecimento, como discutido nesta seção.

Finalmente, na Figura 7.3.26, o conector "6" consiste em outro exemplo de uso de um papel de inferência, mas dessa vez com a direção inversa (“to”). No exemplo, um link faz referência ao conector "6" para especificar que o exibidor de documentos de hiperconhecimento deve processar o nó de vídeo "Final" para extrair conceitos e inserir esses conceitos na base de conhecimento "Futebol".

7.4. Considerações Finais

Neste capítulo, apresentamos uma visão geral da área de pesquisa de multimídia em processos de tomada de decisão. Discutimos brevemente diferentes perspectivas e soluções computacionais que resolvem o problema de decisão. Além disso, foram identificados e classificados temas de pesquisa relevantes e áreas de aplicação, com potenciais desafios de pesquisa em multimídia para a tomada de decisões. Nós mostramos características gerais da computação cognitiva e algumas soluções que oferecem serviços em nuvem. Em seguida, desenvolvemos uma aplicação no escopo do minicurso, utilizando os recursos do IBM Bluemix. Finalmente, apresentamos nossa contribuição, uma extensão de um modelo abstrato bem estabelecida para lidar com a especificação de programas cognitivos numa perspectiva expressiva e consciente.

A relação mutuamente benéfica entre o julgamento humano e a percepção computacional com raciocínio proposto pela computação cognitiva já vêm alavancando os sistemas de

suporte à tomada de decisão para um novo nível. No entanto, no nosso entendimento, ainda carece de uma abstração unificada que permita que autores modelem uma interação mais natural e dinâmica, com mecanismos adequados para conceber um design holístico de programas cognitivos.

Em nossa visão, a nossa abordagem hiperconhecimento é o passo nessa direção. Nós definimos hiperconhecimento como uma noção que associa ambos os aspectos de conhecimento e multimídia. Ao combinar os recursos de NCM com as extensões sugeridas para a estruturação do conhecimento, chegamos a um modelo flexível que pode ser usado para descrever aspectos do usuário, recursos de software cognitivos e as capacidades do dispositivo por completo, em uma única lógica.

Atualmente, nossa agenda de pesquisa guia os próximos passos em relação ao uso e à evolução do NCM:

a) Uso de NCM 3.1 para criar uma linguagem de especificação hiperconhecimento usando JSON ou uma atualização da NCL;

b) Projeto de conversores para mapear o conhecimento descrito de acordo com outros formatos como RDF / RDFS e OWL para NCM 3.1, enriquecendo os dados existentes com os novos recursos do modelo conceitual hiperconhecimento;

c) Arquitetura e desenvolvimento de um sistema de raciocínio semântico para permitir inferir consequências lógicas de bases de conhecimento, fundamentadas na teoria dos conjuntos;

d) Arquitetura e desenvolvimento de uma solução de banco de dados para suportar o armazenamento de conteúdo e recuperação, semelhante às triplestores RDF, mas explorando NCM 3.1;

e) Especificação e desenvolvimento de uma linguagem de consulta semântica para permitir a análise de informações obtidas implícita e explicitamente a partir de dados contextuais, semelhantes a SPARQL;

f) Explorar a apresentação da informação presente em bases de conhecimento por meio das cadeias temporais em um gráfico temporal, hipermídia (HTG), o que reflete a relação cronológica entre os dados modelados;

g) Por fim, a integração de todos esses mecanismos e criação de novos, para arquitetura e desenvolvimento de uma máquina de execução de hiperconhecimento.

Note que um sucesso razoável nesta agenda de pesquisa tem potencial para contribuir na solução do desafio crítico definido por Kelly [Kelly 2015]: "*If cognitive computing is to fulfill its true promise, the underlying platform must be broad and flexible enough to be applied by any company in any industry. And it must be able to be applied across industries. To do that requires a holistic approach to research and development, with the goal of creating a robust platform with a range of capabilities to support diverse applications from an ecosystem of developers*".

References

[Brandao et al. 2016] Brandao, R.R.M., Moreno, M.F., Ferreira, J.J., Cerqueira, R. Communicability Issues on PaaS Application Development. IHC'16. Oct, 2016, São Paulo, Brazil. In press.

[Glushko and McGrath 2005] Glushko, R., and McGrath, T. Document engineering. Cambridge: Mit Press, 2005.

[Hallaz 1988] Halasz, F. Reflections on NoteCards: seven issues for the next generation of hypermedia systems. Commun. ACM 1988.

[Kelly 2015] J. Kelly, "Computing, cognition and the future of knowing," Whitepaper, IBM Reseach, 2015.

[Licklider 1960] J. C. Licklider, "Man-computer symbiosis," IRE transactions on human factors in electronics, no. 1, pp. 4–11, 1960.

[Mell and Grance 2011] Peter Mell and Tim Grance. 2011. The NIST definition of cloud computing, 2011.

[Moreno and Soares 2011] Moreno, Marcio Ferreira; Soares, L. F. G. Resilient Hypermedia Presentations. In: 2nd International Workshop on Software Aging and Rejuvenation, in conjunction with IEEE 21st International Symposium on Software Reliability Engineering, ISBN 978-1-61284-344-5. DOI 10.1109/WOSAR.2010.5722101. Nov. 1-4, 2010, San Jose CA, USA.

[Moreno et al. 2015] Moreno, M. F., Costa, R. M. R., and Soares, L. F. G. "Interleaved Time Bases in Hypermedia Synchronization." IEEE MultiMedia Magazine, pp. 68-78. 2015.

[Moreno et al. 2016a] Moreno, M.F., Brandao, R.R.M., Cerqueira, R. NCM 3.1: A Conceptual Model for Hyperknowledge Document Engineering. DOI: http://dx.doi.org/10.1145/2960811.2967167. ACM Document Engineering, Vienna, Austria. September 2016.

[Moreno et al. 2016b] Moreno, M.F., Brandao, R.R.M., Cerqueira, R. Extending hypermedia conceptual models to support hyperknowledge specifications. IEEE International Symposium on Multimedia, San Jose, CA, USA, 2016. (in press).

[Moreno et al. 2016c] Moreno, M.F., Brandao, R.R.M., Cerqueira, R. Towards a Conceptual Model for Cognitive-Intensive Practices. IEEE International Symposium on Multimedia, San Jose, CA, USA, 2016. (in press).

[Moreno et al. 2016d] Moreno, M.F., Brandao, R.R.M., Cerqueira, R. Challenges on Multimedia for Decision-Making in the Era of Cognitive Computing. 1st International Workshop on Multimedia Support for Decision-Making Processes, San Jose, CA, USA, 2016. (in press).

[Moreno et al. 2016e] Moreno, M.F., Cerqueira, R., Colcher, S. Synchronization Abstractions and Separation of Concerns as Key Aspects to the Interoperability in IoT. EAI International Conference on Interoperability in IoT. Paris, France, 2016. Full paper. (in press).

[Moreno et al. 2016f] Moreno, M.F. et al. Deepening the separation of concerns in the implementation of multimedia systems. DOI: http://dx.doi.org/10.1145/2851613.2851769. ACM SAC, Pisa, Italy, 2016.

[Power et al. 2015] Power, D. J., Sharda, R. and Burstein, F. 2015. Decision Support Systems. Wiley Encyclopedia of Management.

[Soares et al. 2009] Soares, L.F.G., Costa, R.M.R; Moreno, Marcio Ferreira, Moreno, Marcelo Ferreira. Multiple Exhibition Devices in DTV Systems. In: Proceedings of the ACM International Conference on Multimedia, 2009. Beijing/China. p. 281-289.

[Soares et al. 2010] Soares, L.F.G. et al Ginga-NCL: Declarative Middleware for Multimedia IPTV Services. In: IEEE Communications Magazine. Vol.48, No.6. June 2010. p74-81.

[Soares et al. 2015] Soares, L.F.G., Moreno, M.F., and Vasconcelos, A. Controlling the focus and input events in multimedia applications. DOI: http://dx.doi.org/10.1145/2695664.2695885. ACM SAC, Salamanca, Spain, 2015.

[Soffer et al. 2016] Aya Soffer, David Konopnicki, and Haggai Roitman. 2016. When Watson Went to Work: Leveraging Cognitive Computing in the Real World. In ACM SIGIR conference on Research and Development in Information Retrieval (SIGIR '16). ACM, New York, NY, USA, 455-456. DOI: http://dx.doi.org/10.1145/2911451.2926724.

[Triantaphyllou 2010] E. Triantaphyllou, Multi-criteria decision-making methods: a comparative study. Dordrecht: Kluwer, 2010.

Biografia Resumida dos Autores

Dr. Marcio Ferreira Moreno é membro da equipe de pesquisa sobre soluções de recursos naturais da IBM Research desde outubro de 2015, investigando o papel da área de multimídia em computação cognitiva e processos de tomada de decisão, bem como novas tecnologias (incluindo internet das coisas e computação em nuvem) para o desenvolvimento, integração e operação de sistemas de software. Ao estudar tecnologias multimídia, ele arquitetou e implementou protocolos de rede, aplicações cliente-servidor, aplicações multimídia/hipermídia distribuídas, bem como a implementação de referência do middleware Ginga para sistemas de TV Digital. Além disso, ele tem contribuições nas especificações da NCL (*Nested Context Language*), uma linguagem declarativa para autoria de documentos hipermídia. Ginga e NCL fazem parte de duas normas internacionais, as quais ele também contribuiu: Recomendação ITU-T H.761 para serviços IPTV e os padrões ISDB-T (*International Standard for Digital Broadcasting – Terrestrial*). Dr. Moreno publicou mais de 40 artigos científicos em revistas internacionais e anais de conferências. Fez parte do conselho editorial da Revista Brasileira de Computação como editor convidado, é revisor e parte do comitê de programa de diversas conferências internacionais na área de multimídia, incluindo ACM Multimedia, IEEE ISM, ACM MMSys, e IARIA MMEDIA. Dr. Moreno é ainda organizador do primeiro workshop internacional com o tema *Multimedia Support for Decision-Making Processes* (MuSDeMP'16) e do primeiro workshop internacional intitulado *Synchronism of Things* (SoT'16). Dr. Moreno recebeu o prêmio de melhor tese de doutorado da ACM EuroITV'10, bem como o Prêmio Oscar Niemeyer para projetos científicos e tecnológicos em 2011.

Dr. Rafael Brandão é um pesquisador multidisciplinar na IBM Research. Atualmente, sua pesquisa está focada na modelagem de conhecimento e no uso de tecnologias como Computação Cognitiva e Computação em Nuvem no contexto de aplicações para a indústria de Óleo e Gás. Obteve seu título de doutor em Ciência da Computação pela PUC-Rio, investigando a aquisição e estruturação de dados de análises qualitativas através de uma infraestrutura de software ubíquo, fundamentado em teorias de IHC como, por exemplo, Engenharia Semiótica. Recentemente, foi coautor do livro intitulado "*Software Developers as Users: Semiotic Investigations in Human-Centered Software Development*", em processo de publicação pela editora Springer. Além disso, tem envolvimento nos campos de pesquisa de processos de tomada de decisão e Ciência Cognitiva. Dr. Brandão é também co-organizador do workshop MuSDeMP'16. Trabalhou por cinco anos no Instituto Tecgraf, na PUC-Rio, onde atuou como líder de P&D para desenvolvimento de uma infraestrutura de software para experimentação de sistemas de Captura & Acesso. Do lado da indústria, trabalhou também como consultor multimídia, engenheiro de software e como desenvolvedor para diferentes empresas, onde desenvolveu middleware e outros sistemas distribuídos. Anteriormente, sua agenda de

pesquisa focou em aspectos de autoria multimídia em ambientes de middleware de TV digital fazendo uso de múltiplos dispositivos móveis.

Dr. Renato Cerqueira é gerente sênior da área de Recursos Naturais na IBM Research Brazil, onde investiga novas tecnologias de software para o projeto e construção de sistemas cognitivos, aplicados à caracterização e gestão de recursos naturais. Dr. Cerqueira está particularmente interessado em novas tecnologias e métodos para acelerar o desenvolvimento e evolução de sistemas cognitivos e suas respectivas bases de conhecimento. De 2002 a 2011, foi professor do Departamento de Informática da PUC-Rio. De 1993 a 2011, foi pesquisador do Tecgraf/PUC-Rio, e o líder científico do seu grupo de pesquisa em Engenharia de Sistemas Distribuídos, realizando vários projetos de P&D com parceiros da indústria e da academia. Dr. Cerqueira já publicou mais de 80 artigos, com mais de 2400 citações, e orientou vários alunos de doutorado e mestrado. Ele é formado em Engenharia de Computação pela PUC-Rio, e obteve os títulos de Doutor e Mestre em Ciência da Computação também pela PUC-Rio. Durante 2001, foi Pesquisador Visitante na Universidade de Illinois em Urbana-Champaign, trabalhando com o Prof. Roy Campbell em tecnologias de middleware para Computação Ubíqua. Participou da conferência internacional de middleware (ACM/IFIP/USENIX) como *chair* do comitê técnico de programa, membro do *steering committee*, membro do comitê técnico de programa, *chair* de tutoriais, *chair* de organizadores locais e *chair* do comitê financeiro. Dr. Cerqueira é ainda membro do *steering committee* do *Workshop on Adaptive and Reflective Middleware*, bem como co-organizador do MuSDeMP'16.

Capítulo 8

Programando aplicações multimídia no GStreamer

Guilherme F. Lima, Rodrigo C. M. Santos, Roberto G. de A. Azevedo
Departamento de Informática, PUC-Rio, Rio de Janeiro, Brasil

Abstract

This short course is an introduction to GStreamer, one of the main free/open-source frameworks for multimedia processing. We start presenting GStreamer, its architecture and the dataflow programming model, and then adopt a hands-on approach. Starting with an example, a simple video player, we introduce the main concepts of GStreamer's basic C API and implement them over the initial example incrementally, so that at the end of the course we get a complete video player with support for the usual playback operations (start, stop, pause, seek, fast-forward, and rewind). We also discuss sample filters—processing elements that manipulate audio and video samples—and present some of the filters natively available in GStreamer. Moreover, we show how one can extend the framework by creating a plugin with a custom filter that manipulates video samples. The prerequisite for the short course is a basic knowledge of the C programming language. At the end of the course, we expect that participants acquire a general view of GStreamer, and be able to create simple multimedia applications and explore its more advanced features.

Resumo

Este minicurso é uma introdução ao GStreamer, um dos principais frameworks *de código livre/aberto para processamento de dados multimídia. Começamos apresentando o GStreamer, sua arquitetura e modelo de programação baseado em* dataflow*, e em seguida, adotamos uma abordagem prática. Partindo de um exemplo inicial, um* player *de vídeo, introduzimos cada conceito da API C básica do GStreamer e o implementamos sobre o exemplo, incrementando-o, de forma que ao final do minicurso obtemos um* player *de vídeo completo, com suporte às operações usuais de reprodução de vídeo (*start*,* stop*,* seek*,* fast-forward *e* rewind*). Discutimos também filtros—elementos que manipulam amostras de áudio e vídeo—e apresentamos os diversos filtros disponíveis nativamente no GStreamer. Além disso, mostramos como estender o* framework *criando um* plugin *com um filtro simples que manipula amostras de vídeo. O único pré-requisito para o minicurso é um conhecimento básico da linguagem de programação C. Ao final do minicurso, esperamos que os participantes tenham uma visão geral do GStreamer, e estejam aptos a criar aplicações simples e explorar os recursos mais avançados do* framework*.*

8.1. Introdução

O GStreamer [GStreamer, 2016] é um dos principais *frameworks* de código livre/aberto para processamento de dados multimídia. Além de robusto e flexível, ele suporta diversos formatos de áudio e vídeo, e é amplamente utilizado na indústria e na academia [GStreamer Developers, 2016]. O *framework* em si consiste de um conjunto de bibliotecas C e ferramentas relacionadas. Neste minicurso, apresentamos tanto a parte conceitual quanto a prática do GStreamer.

Na parte conceitual, discutimos o modelo de computação *dataflow* no qual o GStreamer se baseia, e que também é adotado por outros *frameworks* e linguagens multimídia, por exemplo, DirectShow [Chatterjee e Maltz, 1997], Pure Data [Puckette, 2007], CLAM [Amatriain et al., 2008], ChucK [Wang e Cook, 2003], Faust [Orlarey et al., 2009], etc. Nesse modelo, uma aplicação multimídia estrutura-se como um grafo em que os nós são elementos processadores e as arestas representam conexões entre elementos por onde fluem amostras de áudio e vídeo e dados de controle. O modelo de *dataflow* é particularmente interessante para multimídia porque possibilita implementações naturalmente paralelas, modulares e escaláveis.

Na parte prática, apresentamos os principais conceitos da API C [Kernighan e Ritchie, 1988] básica (de baixo nível) do GStreamer 1.8, sua versão estável mais atual, e ilustramos o uso dessa API a partir da construção de um reprodutor (*player*) de vídeo. Apesar de aqui estarmos interessados apenas na reprodução (decodificação e apresentação) de fluxos de mídia, essa mesma API pode ser utilizada para capturar fluxos de áudio e vídeo, codificá-los e transmiti-los na rede. O GStreamer possui uma grande variedade de componentes para tratar cada uma dessas fases de processamento e, portanto, pode ser usado para construir diversos tipos de aplicações multimídia tais como editores de vídeo, transcodificadores, transmissores de fluxos de mídia, *players* de mídia e motores de renderização de linguagens multimídia.

O restante do capítulo está organizado da seguinte forma. Na Seção 8.2 apresentamos uma versão preliminar do exemplo base do minicurso: um *player* de vídeo simples, porém funcional. Na Seção 8.3 discutimos o que está por trás do código aparentemente simples do exemplo anterior e reconstruímos o mesmo exemplo usando a API básica do GStreamer; essa versão reconstruída é o ponto de partida dos incrementos posteriores. Na Seção 8.4 apresentamos alguns do principais filtros de áudio e vídeo disponíveis no GStreamer e discutimos como integrá-los ao exemplo. Na Seção 8.5 adicionamos ao exemplo suporte às operações usuais de controle de reprodução (*start*, *stop*, *pause*, *seek*, *fast-foward* e *rewind*). Na Seção 8.6 apresentamos a arquitetura de *plugins* do GStreamer e mostramos como implementar e integrar ao exemplo um *plugin* simples contendo um elemento que manipula amostras de vídeo. Finalmente, na Seção 8.7 discutimos funcionalidades avançadas do *framework* e listamos algumas referências para estudos posteriores.

Antes de escrever nossa primeira aplicação GStreamer, porém, é preciso entender o modelo de computação *dataflow*, no qual o *framework* se baseia. No restante desta seção apresentamos esse modelo e discutimos como ele é instanciado no GStreamer.

O modelo *dataflow* e sua instanciação no GStreamer

No modelo de computação *dataflow* os dados são processados enquanto "fluem" através de uma rede. Essa rede estrutura-se como um grafo dirigido em que os nós representam elementos de processamento, ou atores, e as arestas representam conexões unidirecionais por onde fluem os dados. Os atores recebem dados através de suas portas de entrada e emitem dados através de suas porta de saída. Um *pipeline* é um *dataflow* em que os dados fluem através das arestas na mesma ordem em que foram produzidos [Kahn e MacQueen, 1977, Lee e Parks, 1995]. O modelo de computação *dataflow*, e em especial o modelo de *pipeline*, é interessante para multimídia porque aproxima a estrutura real do sistema da sua descrição abstrata, idealizada na forma de diagrama de blocos [Yviquel et al., 2015]. Além disso, o modelo de *dataflow* induz implementações flexíveis e eficientes já que ele é naturalmente paralelo, modular e escalável: (1) paralelo porque conceitualmente os atores executam independentemente uns dos outros; (2) modular porque a lógica de processamento está encapsulada nos atores e pode ser reusada em diversos pontos do *dataflow*; e (3) escalável porque a estrutura é naturalmente composicional (um *dataflow* pode ser visto como um ator e vice versa).

A Figura 8.1 apresenta o leiaute de um *pipeline* típico para processamento multimídia. O leiaute nesse caso é uma versão simplificada do *pipeline* de uma aplicação GStreamer que reproduz um vídeo Ogg [Pfeiffer, 2003] . Os nós da figura representam atores e as arestas representam as conexões por onde fluem as amostras de áudio e vídeo e dados de controle. Na terminologia do GStreamer os atores são chamados de "elementos" e as portas de "*pads*". Há dois tipos de *pads*: *sink pads* e *source pads*. As *sink pads* são as portas de entrada através das quais o elemento consome dados, e as *source pads* são as portas de saída através das quais o elemento produz dados. Os elementos são classificados de acordo com o tipo de suas *pads*. Elementos produtores (*sources*) possuem apenas *source pads*. Elementos processadores possuem ambos os tipos, *source pads* e *sink pads*. E, elementos consumidores (*sinks*) possuem apenas *sink pads*. Consequentemente, produtores apenas produzem dados, processadores consomem e produzem dados, e consumidores apenas consomem dados.

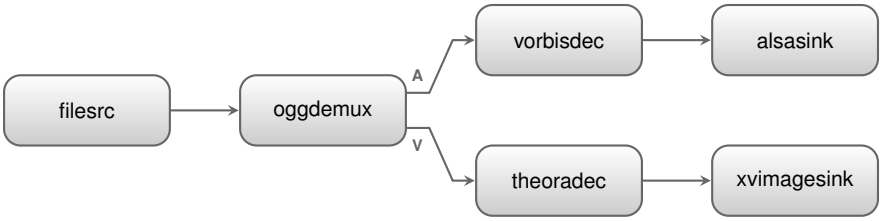


Figura 8.1. Um *pipeline* GStreamer que reproduz um arquivo de vídeo Ogg.

Na Figura 8.1 há um elemento produtor ("filesrc"), três processadores ("oggdemux", "vorbisdec" e "theoradec") e dois consumidores ("alsasink" e "xvimagesink"). O elemento "filesrc" possui uma única *source pad* que está conectada à *sink pad* do elemento subsequente, "oggdemux". Ou seja, os dados produzidos pelo elemento "filesrc" fluem através da sua *source pad* para a *sink pad* do elemento "oggdemux". No diagrama, essa conexão entre as *pads* é representada pela aresta entre os elementos "filesrc" e "oggdemux". Similarmente, as demais arestas denotam conexões entre *source pads* e *sink pads*.

O processo de desenvolvimento de uma aplicação GStreamer com *pipeline* estático (cuja topologia não muda em tempo de execução) é relativamente simples. Basta instanciar os elementos necessários, interconectar suas *pads* e iniciar o *pipeline* resultante. O *pipeline* da Figura 8.1, por exemplo, após iniciado opera da seguinte forma.

1. O elemento "filesrc" lê um arquivo Ogg do disco e escreve o fluxo de bytes resultante na sua *source pad*. Ogg é [Pfeiffer, 2003] um formato para multiplexação de fluxos de áudio, vídeo e texto. Vamos assumir que o arquivo Ogg nesse caso contém apenas dois fluxos multiplexados: um fluxo de áudio (sequência de amostras de áudio) codificado no formato Vorbis [Xiph.Org, 2015]; e um fluxo de vídeo (sequência de quadros de imagem) codificado no formato Theora [Xiph.Org, 2011].

2. O elemento "oggdemux" lê da sua *sink pad* um fluxo de bytes codificado no formato Ogg, demultiplexa-o e escreve os fluxos Vorbis (áudio) e Theora (vídeo) resultantes nas *source pads* correspondentes. Na figura 8.1, a *source pad* que recebe o fluxo de áudio codificado possui o rótulo "A" e a *source pad* que recebe o fluxo de vídeo codificado possui o rótulo "V".

3. O elemento "vorbisdec" lê da sua *sink pad* um fluxo de bytes codificado no formato Vorbis, decodifica-o e escreve o fluxo de áudio PCM resultante na sua *source pad*. Um fluxo de áudio PCM é o que chamamos de "áudio descomprimido" (*raw*), ou seja, é uma sequência de amostras obtidas via *pulse-code modulation* que representa digitalmente o sinal analógico original.

4. O elemento "theoradec" opera de maneira análoga. Ele lê da sua *sink pad* um fluxo de bytes codificado no formato Theora, decodifica-o e escreve o fluxo de vídeo descomprimido (*raw*) resultante na sua *source pad*. Um fluxo de vídeo descomprimido é uma sequência de amostras (quadros) de vídeo em que cada quadro é uma matriz de *pixels* codificados em algum modelo de cor. No modelo de cor RGB (*red-green-blue*), por exemplo, cada *pixel* é codificado como uma sequência de três inteiros que representam tonalidades de vermelho, verde e azul.

5. O elemento "alsasink" lê um fluxo de áudio descomprimido da sua *sink pad* e utiliza a biblioteca ALSA [ALSA, 2016] para reproduzir as amostras do fluxo nos alto-falantes.

6. O elemento "xvimagesink" lê um fluxo de vídeo descomprimido da sua *sink pad* e utiliza a biblioteca X11 [X.Org, 2016] para reproduzir os quadros do fluxo na tela.

A função de um *pipeline*, isto é, o que ele faz ou computa, é o resultado da combinação da função dos seus elementos que, conceitualmente, operam em paralelo. O *pipeline* anterior, portanto, (1) lê um arquivo Ogg, (2) demultiplexa-o, (3–4) decodifica os fluxos de áudio e vídeo resultantes e (5–6) reproduz os fluxos decodificados nos dispositivos de saída correspondentes (alto-falantes e tela). Conceitualmente, tudo isso acontece em paralelo, ou seja, podemos assumir que enquanto os *sinks* "alsasink" e "xvimagesink" estão exibindo amostras, o *source* "filesrc" está lendo bytes do disco, o demultiplexador "oggdemux" está demultiplexando dados Ogg e os decodificadores "vorbisdec" e "theoradec" estão decodificando dados Vorbis e Theora.

A descrição anterior seria suficiente se estivéssemos interessados apenas em exibir as amostras decodificadas o mais rápido possível. Mas esse não é caso. Queremos "tocar" o vídeo original em tempo real, isto é, reproduzi-lo nas mesmas condições em que ele foi gravado (amostrado). Sendo assim, para que o vídeo seja reproduzido corretamente, suas amostras de áudio e vídeo devem ser exibidas na taxa correta. Valores típicos para essas taxas são 44100Hz para amostras de áudio e 30Hz para amostras de vídeo; ou seja, a cada 22,67µs uma nova amostra de áudio deve ser enviada ao dispositivo de saída de áudio, e a cada 33,33ms uma nova amostra de vídeo deve ser enviada ao dispositivo de saída de vídeo.

No GStreamer, em geral, são os elementos *sink* os responsáveis por controlar as taxas de exibição de amostras. Por exemplo, no*pipeline* da Figura 8.1, para manter a taxa de reprodução necessária, os *sinks* "alsasink" e "xvimagesink" armazenam as amostras recebidas numa fila interna e exibem-nas (consomem-nas) apenas no momento adequado. Já os outros elementos operam livres—consomem e produzem dados em taxas arbitrárias. Se durante a execução os elementos que precedem os *sinks* operarem rápido o bastante, as filas dos *sinks* nunca ficarão vazias e todas as amostras serão exibidas no momento correto, mas esse nem sempre é o caso.

Se os elementos que precedem os *sinks* operarem abaixo da taxa de consumo dos *sinks*, pode ser que alguma das filas internas fique vazia e, caso chegue o momento de exibir uma amostra, o *sink* simplesmente não tenha o que exibir, causando interrupções ou saltos na reprodução (amostras "atrasadas" quando chegarem serão descartadas). Há ainda o problema inverso. Se os elementos que precedem os *sink* operarem muito acima da taxa de consumo dos *sinks*, pode ser que a capacidade da fila interna seja excedida e amostras sejam perdidas. Para evitar ambos os problemas, esvaziamento ou estouro das filas, ou mesmo para limitar o uso de CPU, os *sinks* enviam aos elementos que os precedem eventos de QoS (*quality of service*), que indicam o quão atrasada ou adiantada está cada amostra que chega. Os elementos anteriores (produtores e processadores) podem capturar esses eventos e usar a informação de retardo para ajustar a sua taxa de operação.

Dois tipos de dados trafegam através das conexões (arestas) de um *pipeline* GStreamer: segmentos de dados (*buffers*) e eventos (*events*). Os *buffers* carregam segmentos do conteúdo processado entre *pads* (por exemplo, amostras de áudio e vídeo codificadas ou *raw*) e fluem exclusivamente na direção das conexões, ou seja, de *source pads* para *sink pads*. Já os eventos carregam informação de controle; eles também fluem entre conexões, mas podem percorrê-las em ambos os sentidos: fluxo abaixo (*downstream*), de *source pads* para *sink pads*, ou fluxo acima (*upstream*), de *sink pads* para *source pads*. Além de eventos de QoS, elementos podem emitir eventos de EOS (*end-of-stream*) que indicam o fim do fluxo, eventos de *seek* que indicam deslocamentos no fluxo e eventos de *flush* que sinalizam que *caches* internos devem ser descarregados.

Buffers e eventos percorrem as conexões em paralelo. Ou seja, conceitualmente cada conexão entre as *pads* dos elementos pode ser entendida como consistindo de dois canais, um canal unidirecional para *buffers* e outro canal bidirecional para eventos. A Figura 8.2 ilustra a estrutura conceitual de uma conexão entre *pads*. Na figura,"B" é o canal de *buffers* e "E" é o canal de eventos.

Figura 8.2. Estrutura conceitual de uma conexão entre *pads* no GStreamer.

A discussão do modelo *dataflow* e da sua instanciação no GStreamer se encerra aqui. Na maior parte do tempo, programar no GStreamer consiste em montar *pipelines* e controlar seu funcionamento. Até agora discutimos de maneira abstrata os conceitos envolvidos nessa montagem e controle. A partir da próxima seção, veremos como utilizar esses conceitos na prática.

8.2. Olá mundo: Tocando um vídeo

Nosso objetivo nesta seção é usar o GStreamer para tocar um vídeo. Para tal, poderíamos implementar o *pipeline* da Figura 8.1, mas há uma maneira mais simples: basta usar o elemento "playbin". A Listagem 8.1 apresenta um programa C que faz exatamente isso.

```
1  #include <glib.h>
2  #include <gst/gst.h>
3
4  int main (int argc, char *argv[])
5  {
6    GstElement *playbin;
7    char *uri;
8    GstStateChangeReturn ret;
9    GstBus *bus;
10   GstMessage *msg;
11
12   gst_init (&argc, &argv);
13
14   playbin = gst_element_factory_make ("playbin", "hello");
15   g_assert_nonnull (playbin);
16
17   uri = gst_filename_to_uri ("bunny.ogg", NULL);
18   g_assert_nonnull (uri);
19   g_object_set (G_OBJECT (playbin), "uri", uri, NULL);
20   g_free (uri);
21
22   ret = gst_element_set_state (playbin, GST_STATE_PLAYING);
23   g_assert (ret != GST_STATE_CHANGE_FAILURE);
24
25   bus = gst_element_get_bus (playbin);
26   msg = gst_bus_timed_pop_filtered (bus, GST_CLOCK_TIME_NONE,
27                                     GST_MESSAGE_ERROR | GST_MESSAGE_EOS);
28   gst_message_unref (msg);
29   gst_object_unref (bus);
30   gst_element_set_state (playbin, GST_STATE_NULL);
31   gst_object_unref (playbin);
32
33   return 0;
34 }
```

Listagem 8.1. Tocando um vídeo no GStreamer usando o elemento "playbin".

Vejamos o propósito de cada linha da Listagem 8.1. A linha 1 inclui as declarações da GLib [GLib, 2016a], a biblioteca de portabilidade do projeto GNOME [GNOME, 2016], e a linha 2 inclui as declarações do GStreamer—o GStreamer depende do *framework* GObject [GLib, 2016b] da GLib para programação orientada a objetos em C. Na listagem, as chamadas com prefixo "`g_`" ou "`G_`" e referem-se à funções e macros da GLib, e as chamadas com prefixo "`gst_`" ou "`GST_`" e as declarações com prefixo "`Gst`" referem-se à funções, macros e tipos do GStreamer.

A chamada `gst_init`, linha 12, inicializa o GStreamer e trata os argumentos com prefixo "- -gst-" em `argv`.

A próxima chamada, linha 14, cria um elemento "playbin" que é também o *pipeline* da aplicação. No GStreamer, o *pipeline* é, ele próprio, um elemento—que contém outros elementos mas não possui *pads*. Tais elementos contêiner são chamados de "*bins*". A função `gst_element_factory_make` aloca e retorna um novo elemento (`GstElement`) do tipo especificado. O primeiro parâmetro da função é o nome da fábrica do tipo e o segundo parâmetro (opcional) é o nome a ser atribuído ao elemento retornado—esse nome pode ser usado mais tarde para obter uma referência para o elemento a partir do *pipeline*. No exemplo da Listagem 8.1, a chamada cria e retorna um elemento do tipo "playbin" chamado "hello". Um elemento "playbin" nada mais é do que um *pipeline* que se autoconfigura. Dada uma URI, assim que o "playbin" é iniciado ele determina o tipo do conteúdo da URI e constrói um *pipeline* apropriado para reproduzi-la.

A chamada `g_assert_nonnull`, linha 15, é uma asserção que garante que a chamada anterior funcionou corretamente, isto é, retornou um elemento válido (não nulo).

A chamada `gst_filename_to_uri`, linha 17, constrói uma URI a partir do caminho de um arquivo. Nesse caso, "bunny.ogg" é caminho do arquivo de vídeo Ogg que queremos tocar. O vídeo em si é o curta de animação "Big Buck Bunny" [Blender, 2008] produzido pelo Blender Institute e licenciado via Creative Commons.

A chamada `g_object_set`, linha 19, atribui a URI criada anteriormente à propriedade "uri" do elemento "playbin". Note que `g_object_set` é uma função do GObject. Todo elemento (`GstElement`) é também um objeto (`GObject`), e todo objeto possui propriedades cujos valores podem ser obtidos via `g_object_get` e alterados via `g_object_set`.

A chamada `g_free`, linha 20, libera a *string* alocada na linha 16. Nesse ponto, a *string* já não é mais necessária pois foi copiada pela chamada `g_object_set` anterior.

A chamada `gst_element_set_state`, linha 22, inicia o *pipeline* da aplicação, isto é, transiciona o elemento "playbin" do seu estado inicial *null* (`GST_STATE_NULL`) para o estado *playing* (`GST_STATE_PLAYING`). Na Seção 8.3, vamos discutir em detalhes as consequências internas dessa transição. Por enquanto, basta dizer que nesse ponto o elemento "playbin" (1) inspeciona o conteúdo apontado pela URI configurada, (2) instancia e interconecta os elementos necessários para reproduzir esse conteúdo, (3) inicia a sua reprodução numa *thread* separada e (4) devolve o controle à *thread* principal da aplicação.

A chamada `g_assert`, linha 23, é uma asserção que garante que a requisição de transição de estado anterior foi bem sucedida. Após essa linha, podemos assumir que o vídeo está tocando e que aplicação possui pelo menos duas *threads*: a *thread* principal, que está prestes a executar a linha 23, e uma ou mais *threads* secundárias, que operam o *pipeline*. A *thread* principal é chamada de "*thread* da aplicação" e as *threads* secundárias são chamadas de "*streaming threads*". As *streaming threads* se comunicam com a *thread* da aplicação através de mensagens assíncronas postadas num barramento (*bus*) associado ao *pipeline*. Essas mensagens informam a *thread* da aplicação sobre acontecimentos internos do *pipeline*, por exemplo, mudança de estado de elementos, fim do fluxo, erros, *warnings*, etc., e permitem que a aplicação reaja da maneira apropriada.

A chamada `gst_element_get_bus`, linha 25, obtém uma referência para o barramento de mensagens (*bus*) do *pipeline* do "playbin".

A próxima chamada, linha 26, bloqueia a *thread* da aplicação até que uma mensagem de erro ou de EOS (*end-of-stream*, "fim do fluxo") seja postada no barramento. Ou seja, a *thread* da aplicação aguarda nessa chamada até que um erro aconteça ou até que o vídeo seja reproduzido completamente. A função `gst_bus_timed_pop_filtered` recebe o *bus*, o tempo máximo de espera e a máscara dos tipos das mensagens a serem aguardadas, e bloqueia até que o tempo máximo seja atingido ou até que uma mensagem de um dos tipos esperados seja postada no *bus*. A função retorna `NULL` caso o tempo máximo seja atingido ou retorna uma referência para a mensagem recebida. Na Listagem 8.1, estamos aguardando por um tempo ilimitado (`GST_TIME_CLOCK_NONE`) uma mensagem de erro (`GST_MESSAGE_ERROR`) ou uma mensagem de EOS (`GST_MESSAGE_EOS`) que após a chamada é armazenada na variável `msg`.

A chamada `gst_message_unref`, linha 27, libera a mensagem retornada na chamada da linha anterior. Nesse ponto, ou houve um erro ou a reprodução chegou ao fim.

A chamada `gst_object_unref`, linha 28, libera a referência para o *bus* do "playbin", obtida na linha 25. A função `gst_object_unref` é um pseudônimo (*alias*) para a função `g_object_unref`, que libera uma referência para um `GObject`. No GStreamer, a maioria dos tipos que são "objeto" herdam de `GstObject` que, por sua vez, herda de `GObject`.

A chamada `gst_element_set_state`, linha 30, para o *pipeline* da aplicação (caso ele ainda não esteja parado) e libera os recursos utilizados durante o seu processamento, isto é, transiciona o elemento "playbin" de volta para o estado inicial *null*.

Finalmente, a chamada `gst_object_unref`, linha 31, libera o próprio elemento "playbin", alocado na linha 14.

Se tudo correr bem, ao executar o programa "Olá mundo" anterior uma nova janela aparece na tela e o vídeo é reproduzido do inicio ao fim. A Figura 8.3 apresenta um quadro do vídeo em questão. Para rodar o exemplo, porém, primeiro é preciso compilá-lo.

Figura 8.3. Quadro do vídeo *Big Buck Bunny* [Blender, 2008].

Compilando o programa "Olá mundo" no GNU/Linux

Para compilar o programa da Listagem 8.1 precisamos informar ao compilador C o caminho do diretório contendo os cabeçalhos da GLib e do GStreamer. Além disso, precisamos informar ao *linker* o caminho e o nome das bibliotecas dinâmicas correspondentes. No GNU/Linux, a maneira mais fácil de se fazer isso é através da ferramenta *pkg-config*. Por exemplo:

```
$ cc hello.c -o hello `pkg-config --cflags --libs glib-2.0\
        gstreamer-1.0`
```

Esse comando compila e *link*-edita o programa da Listagem 8.1 ("hello.c") e gera um executável chamado "hello". As opções "- -cflags" e "- -libs" do comando *pkg-config* instruem-no a emitir as *flags* de compilação e *link*-edição apropriadas para os módulos listados, no caso, "glib-2.0" e "gstreamer-1.0". Observe que a chamada do comando *pkg-config* aparece entre crases. Ou seja, antes de avaliar o comando mais externo, *cc*, que é a chamada do compilador C, o interpretador de comandos (*shell*) executa o comando *pkg-config* e substitui o texto entre crases pelo resultado dessa execução. Por exemplo, no nosso ambiente, o comando anterior avalia para:

```
$ cc hello.c -o hello\
     -pthread\
     -I/usr/include/gstreamer-1.0\
     -I/usr/lib/gstreamer-1.0/include\
     -I/usr/include/glib-2.0\
     -I/usr/lib/glib-2.0/include\
     -lgstreamer-1.0 -lgobject-2.0 -lglib-2.0
```

Uma forma de tornar o processo de compilação menos trabalhoso é escrever um *makefile* com a descrição da sequência de comandos que constrói o programa. A Listagem 8.2 apresenta o *makefile* (arquivo texto chamado "Makefile") que usamos para construir os exemplos do minicurso. Na listagem, a variável `PROGRAMS`, linha 1, contém o nome dos programas a serem construídos. A variável `MODULES`, linha 2, contém os nomes dos módulos do *pkg-config* necessários. E as variáveis `CFLAGS` e `LDFLAGS`, linhas 3 e 4, contém as *flags* de compilação e *link*-edição correspondentes. O restante do arquivo, linhas 5–8, define os alvos "all" e "clean". (Na linha 7, o primeiro caractere é um TAB.) Uma vez escrito o *makefile*, para construir o programa basta executar o comando "`make`" (ou "`make all`") e para remover os arquivos gerados pelo compilador basta executar "`make clean`". (Veja [Gough, 2005, Mecklenburg, 2005] para mais informações sobre como compilar programas e escrever *makefiles* no GNU/Linux.)

```
1 PROGRAMS= hello
2 MODULES= glib-2.0 gstreamer-1.0
3 CFLAGS= -g -Wall -Wextra `pkg-config --cflags $(MODULES)`
4 LDFLAGS= `pkg-config --libs $(MODULES)`
5 all: $(PROGRAMS)
6 clean:
7         -rm -f $(PROGRAMS)
8 .PHONY: all clean
```

Listagem 8.2. *Makefile* que constrói o programa "Olá mundo".

8.3. Tocando áudio e vídeo a partir de elementos básicos

Na Seção 8.2 utilizamos o elemento "playbin" para tocar um vídeo Ogg no GStreamer. Nesta seção, nosso objetivo é construir a mesma aplicação, mas dessa vez sem usar o elemento "playbin". Usando apenas elementos básicos (*sources*, demultiplexadores, decodificadores e *sinks*) vamos construir um *pipeline* semelhante ao da Figura 8.1, Seção 8.1, que lê um arquivo Ogg do disco, demultiplexa-o, e decodifica e reproduz os fluxos de áudio e vídeo resultantes. Antes disso, porém, vejamos um exemplo mais simples.

Tocando um arquivo de áudio MP3

A Listagem 8.3 apresenta um programa que reproduz um arquivo de áudio MP3 [ISO/IEC, 1993] do disco. Para tal, o programa monta um *pipeline* com três elementos: "filesrc", "mad" e "alsasink". Como vimos, no GStreamer o próprio *pipeline* é um elemento—um elemento sem *pads* que contém outros elementos. Na listagem, a chamada da linha 15 aloca o elemento *pipeline*, que inicialmente está vazio, e as chamadas das linhas 16–18, alocam os elementos "filesrc", "mad" e "alsasink". (As chamadas anteriores, linhas 11–13, inicializam o GStreamer e testam se o programa foi chamado corretamente, isto é, se o caminho para o arquivo MP3 está em `argv[1]`.)

```
1  #include <glib.h>
2  #include <gst/gst.h>
3
4  int main (int argc, char *argv[])
5  {
6    GstElement *pipeline, *src, *dec, *sink;
7    GstStateChangeReturn ret;
8    GstBus *bus;
9    GstMessage *msg;
10
11   gst_init (&argc, &argv);
12   if (argc != 2)
13     return (g_printerr ("usage: %s FILE\n", g_get_prgname ()), 1);
14
15   pipeline = gst_element_factory_make ("pipeline", "mp3");
16   src = gst_element_factory_make ("filesrc", "src");
17   dec = gst_element_factory_make ("mad", "dec");
18   sink = gst_element_factory_make ("alsasink", "sink");
19   g_assert (pipeline && src && dec && sink);
20
21   gst_bin_add_many (GST_BIN (pipeline), src, dec, sink, NULL);
22   gst_element_link_many (src, dec, sink, NULL);
23   g_object_set (G_OBJECT (src), "location", argv[1], NULL);
24
25   ret = gst_element_set_state (pipeline, GST_STATE_PLAYING);
26   g_assert (ret != GST_STATE_CHANGE_FAILURE);
27
28   bus = gst_element_get_bus (pipeline);
29   msg = gst_bus_timed_pop_filtered (bus, GST_CLOCK_TIME_NONE,
30                                     GST_MESSAGE_ERROR | GST_MESSAGE_EOS);
31   gst_message_unref (msg);
32   gst_object_unref (bus);
33   gst_element_set_state (pipeline, GST_STATE_NULL);
34   gst_object_unref (pipeline);
35
36   return 0;
37 }
```

Listagem 8.3. Tocando um arquivo de áudio MP3.

Na Seção 8.1, discutimos a operação dos elementos "filesrc" e "alsasink". O primeiro lê um arquivo do disco e gera um fluxo de bytes correspondente, e o segundo lê um fluxo de amostras de áudio PCM e as reproduz no dispositivo de saída de áudio. Aqui a novidade é o elemento processador, mais precisamente, o decodificador "mad". Ele lê da sua *source pad* um fluxo de bytes no formato MP3, decodifica-o via MAD [Underbit, 2016] (biblioteca para decodificação de áudio MP3) e escreve na sua *sink pad* o fluxo de áudio PCM resultante. Apesar de os três elementos, "filesrc", "alsasink" e "mad", fazerem parte do GStreamer, na prática, eles pertencem a pacotes diferentes, distribuídos separadamente.

A distribuição oficial do GStreamer possui cinco pacotes principais: (1) *gstreamer*, contendo o núcleo do *framework*; (2) *gst-plugins-base*, contendo apenas os elementos básicos; (3) *gst-plugins-good*, contendo elementos cuja implementação é considerada de boa qualidade e cuja licença é LGPL (*Lesser GNU General Public License*); (4) *gst-plugins-ugly*, contendo elementos cuja implementação é também considerada de boa qualidade mas que têm licenças problemáticas; e (5) *gst-plugins-bad*, contendo elementos de qualidade inferior.[1] O único pacote absolutamente necessário é o *gstreamer*. Os demais incrementam-no instalando bibliotecas e *plugins* contendo elementos especializados. O elemento "filesrc", por exemplo, está no *plugin* "coreelements" do pacote *gstreamer*. Já o elemento "alsasink" está no *plugin* "alsa" do pacote *gst-plugins-base*, e o elemento "mad" está no *plugin* "mad" do pacote *gst-plugins-ugly*. Você pode descobrir os elementos e *plugins* disponíveis na sua instalação através do comando "`gst-inspect`"—voltaremos a falar desse comando no final da seção.

De volta à Listagem 8.3, a chamada `g_assert` na linha 19 assegura que as chamadas `gst_element_factory_make` anteriores foram bem sucedidas. Isto é, testa se os elementos "pipeline", "filesrc", "mad" e "alsasink" foram de fato alocados. Um erro comum é tentar instanciar um elemento de um *plugin* que não está instalado. Na listagem, se isso acontecer a chamada `gst_element_factory_make` retornará `NULL` e a asserção falhará, abortando o programa.

Continuando, a chamada da `gst_bin_add_many`, linha 21, adiciona três elementos ao *pipeline*: "filesrc", "mad" e "alsasink". Essa função recebe um *bin* (`GstBin`) e uma sequência de elementos (`GstElement`) terminada por `NULL`, e adiciona os elementos da sequência ao *bin*. Note, que antes da chamada precisamos usar a macro `GST_BIN` para converter a variável `pipeline` para o tipo da sua superclasse `GstBin`.

A chamada `gst_element_link_many`, linha 22, conecta os elementos apontados pelas variáveis `src`, `filter` e `sink` em série. Isto é, conecta a *source pad* do "filesrc" à *sink pad* do "mad", e conecta a *source pad* do "mad" à *sink pad* do "videosink". A função `gst_element_link_many` recebe uma sequência de elementos terminada por `NULL` e conecta-os em série apenas se suas *pads* são compatíveis e se todos possuem o mesmo elemento pai (isto é, se todos foram adicionados ao mesmo *bin*). A Figura 8.4 apresenta a estrutura interna do elemento "pipeline", variável `pipeline`, após a chamada da linha 22.

A chamada `g_object_set`, linha 23, atribui o caminho do arquivo MP3 a ser tocado (primeiro argumento do programa) à propriedade "location" do elemento "filesrc". Essa propriedade indica ao elemento o arquivo que servirá de fonte de bytes.

[1] Essa terminologia é inspirada clássico de faroeste "The Good, the Bad and the Ugly" (1966).

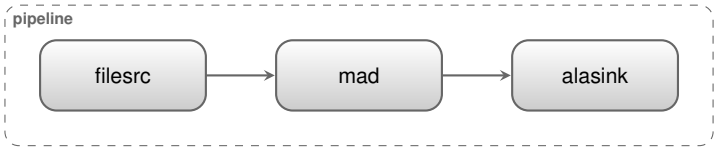


Figura 8.4. *Pipeline* que reproduz um arquivo de áudio MP3.

A chamada `gst_element_set_state`, linha 25, transiciona o "pipeline" do seu estado atual (*null*) para o estado *playing* o que, como vimos na Seção 8.2, faz com que os seus filhos "filesrc", "mad" e "alsasink" comecem a operar. No Gstreamer, todo elemento, incluindo o *pipeline*, possui um estado que pode ser nulo (*null*, identificado pela constante simbólica `GST_STATE_NULL`), pronto (*ready*, `GST_STATE_READY`), pausado (*paused*, `GST_STATE_PAUSED`) ou tocando (*playing*, `GST_STATE_PLAYING`). No estado inicial, *null*, o elemento não possui recursos alocados. No estado *ready*, o elemento aloca recursos globais que não dependem do conteúdo a ser processado. No estado *paused*, o elemento aloca recursos que dependem do conteúdo a ser processado e se prepara para processá-lo. Finalmente, no estado *playing*, o elemento inicia o processamento.

Para chegar do estado *null* (inicial) ao estado *playing* (tocando) todo elemento tem que passar primeiro pelo estados *ready* e *paused*, nessa ordem. De forma análoga, para sair do estado *playing* e voltar ao estado inicial *null*, o elemento tem que passar pelos estados *paused* e *ready*. A Figura 8.5 apresenta a máquina de estados de um elemento GStreamer. Mudanças de estado do *pipeline*, na verdade de *bins* em geral, são propagadas nos elementos filhos, e a direção da propagação é sempre dos *sinks* para os *sources*—dessa forma, dados não são gerados antes que os elementos posteriores no *pipeline* estejam prontos para recebê-los. Por exemplo, na Listagem 8.3, se bem sucedida, a chamada `gst_element_set_state` (linha 25) implica na seguinte sequência transições (sempre dos *sinks* para os *sources*): (1) os elementos filhos "alsasink", "mad" e "filesrc" transicionam de *null* para *ready* e, em seguida, o *pipeline* transiciona para *ready*; (2) os filhos transicionam de *ready* para *paused* e, em seguida, o *pipeline* transiciona para *paused*; e (3) os filhos transicionam de *paused* para *playing* e, finalmente, o *pipeline* transiciona para *playing*. A sequência de transições pode ser realizada sincronamente ou assincronamente. Se a sequência for realizada sincronamente, a *thread* da aplicação bloqueia até a que todas as transições sejam realizadas. Caso contrário, se a sequência for realizada assincronamente, a chamada da linha 25 retorna imediatamente e as transições são realizadas em paralelo (*background*). Em ambos os casos, podemos assumir que nesse ponto a *thread* da aplicação está prestes a executar a asserção da linha 26, que aborta o programa em caso de falha da chamada anterior, e que as *streaming threads* já começaram a operar o *pipeline*, ou seja, os elementos "filesrc", "mad" e "alsasink" já estão produzindo, processando e consumindo dados.

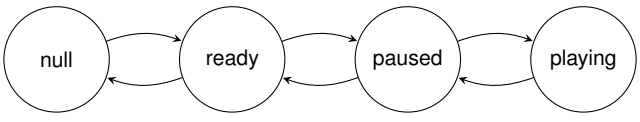


Figura 8.5. Máquina de estados de um elemento (`GstElement`).

A chamada seguinte, linha 28, obtém uma referência para o barramento do *pipeline* e a próxima, linha 29, bloqueia a *thread* da aplicação até que um erro aconteça ou até que o arquivo de áudio seja reproduzido completamente (EOS).

Assim que a chamada `gst_bus_timed_pop_filtered` retorna, a *thread* da aplicação libera as referências do *bus* e da mensagem retornada (linhas 31–32), transiciona o *pipeline* (linha 33) de volta para o estado *null* e libera a sua referência (linha 34). Nesse caso, a chamada `gst_element_set_state` (linha 33) também propaga a transição para os elementos filhos, só que agora as transições são no sentido contrário—de *playing* para *paused*, *ready* e *null*. Finalmente, observe que a chamada `gst_object_unref` (linha 34) quando aplicada a um *bin* libera não apenas uma referência para o *bin* mas também libera uma referência para cada elemento filho, ou seja, nesse ponto os elementos alocados nas linhas 15–28 são liberados.

De volta ao problema inicial

Agora que vimos como alocar, adicionar e interconectar elementos num *pipeline* podemos voltar ao nosso problema inicial: construir um *pipeline* similar ao da Seção 8.1 que usa apenas elementos básicos para tocar um vídeo Ogg. Há uma diferença importante entre esse *pipeline* Ogg (Figura 8.1) e o *pipeline* MP3 (Figura 8.4) que construímos anteriormente. Enquanto no *pipeline* MP3 o número total de *pads* é fixo (conhecido em tempo de compilação) no *pipeline* Ogg esse número é variável. Em particular, no *pipeline* Ogg o número de *sink pads* do elemento "oggdemux" depende do número de subfluxos multiplexados no fluxo Ogg de entrada. Por exemplo, se o fluxo Ogg possuir apenas um subfluxo Vorbis, o elemento "oggdemux" terá uma única *sink pad* que deverá ser conectada à *source pad* do elemento "vorbisdec". No entanto, se o fluxo Ogg possuir dois subfluxos, Vorbis e Theora, o elemento "oggdemux" terá duas *sink pads*, uma para o subfluxo Vorbis que deverá ser conectada ao elemento "vorbisdec", e outra para o subfluxo Theora que deverá ser conectada ao elemento "theoradec". Logo, o número de subfluxos multiplexados no Ogg determina o número de *sink pads* no demultiplexador. *Pads* desse tipo, criadas sob demanda pelo elemento, são chamadas de *sometimes pads*.

No GStreamer, toda *pad* é instanciada de acordo com um *template* que indica a sua direção, capacidade e disponibilidade. A direção (*source* ou *sink*) determina se a *pad* produz ou consome *buffers*. A capacidade, ou *caps*, determina os tipos de *buffers* que podem atravessá-la. E a disponibilidade (*always*, *sometimes* ou *request*) determina o momento em que a *pad* é criada: (1) *always pads* são criadas assim que o elemento é criado; (2) *sometimes pads* são criadas pelo próprio elemento sob condições específicas que normalmente envolvem o conteúdo processado; e (3) *request pads* são criada apenas quando requisitadas pela aplicação.

No *pipeline* MP3 (Figura 8.4) todas as *pads* têm disponibilidade *always*. Isso significa que podemos construir o *pipeline* inteiro estaticamente, conectando todos os seus elementos antes de iniciá-lo. Já no *pipeline* Ogg (Figura 8.1) essa abordagem não é possível. As *sink pads* do elemento "oggdemux" possuem disponibilidade *sometimes* e só serão criadas quando o elemento inspecionar o fluxo de entrada (transicionar para o estado *paused*). Nesse caso, precisamos usar uma abordagem incremental com dois passos. Primeiro, precisamos montar e iniciar um *pipeline* contendo apenas os elementos "filesrc" e

"oggdemux" conectados. Em seguida, precisamos esperar o elemento "oggdemux" notificar a criação de cada *sink pad* para então adicionar e conectar os elementos restantes de acordo com as capacidades da *pad* criada. Esse tipo de notificação assíncrona disparada por elementos é chamada de sinal. Todo sinal possui um nome ("pad-added", nesse caso) e pode ser capturado via um mecanismo de registro de *callbacks*. Ou seja, para capturar um sinal a aplicação registra no elemento uma função *callback* que é chamada sempre que o sinal é disparado.

A Listagem 8.4 apresenta um programa que utiliza a abordagem incremental anterior para montar um *pipeline* que reproduz um arquivo Ogg. A estrutura final desse *pipeline*, apresentada na Figura 8.6, é ligeiramente diferente daquela apresentada na Seção 8.1—a Figura 8.1 é na verdade uma versão simplificada (não funcional) da Figura 8.6. A diferença aqui está nos elementos adicionais "audioconvert", que aparece entre os elementos "vorbisdec" e "alsasink", e "queue", que aparece entre os elementos "oggdemux" e "theoradec".

O elemento "audioconvert" é necessário porque os elementos "vorbisdec" e "alsasink" não podem ser conectados diretamente—as *caps* da *source pad* do "vorbisdec" não são compatíveis com as *caps* do *sink pad* do "alsasink", ou seja, interseção das *caps* dessas *pads* é vazia (podemos entender as *caps* de uma *pad* como o conjunto dos possíveis formatos dos *buffers* que podem atravessá-la). O elemento "audioconvert" converte as amostras produzidas pelo elemento "vorbisdec" para um formato que "alsasink" aceita. Já o elemento "queue" entre os elementos "oggdemux" e "theoradec" é necessário porque esses elementos precisam operar em *streaming threads* separadas. O elemento "queue" é uma fila de *buffers*; quando inserido entre dois elementos, produtor e consumidor, o "queue" força a criação de duas *streaming threads*, uma que opera o produtor e outra que opera o consumidor, e ao mesmo tempo, age como elemento de sincronização dessas *threads*. Isto é, o "queue" bloqueia a *thread* produtora caso a fila fique cheia e bloqueia a *thread* consumidora caso a fila esvazie. Sem o elemento *queue* o *pipeline* da Figura 8.6 não funciona pois não completa a transição do estado *paused* para *playing*.

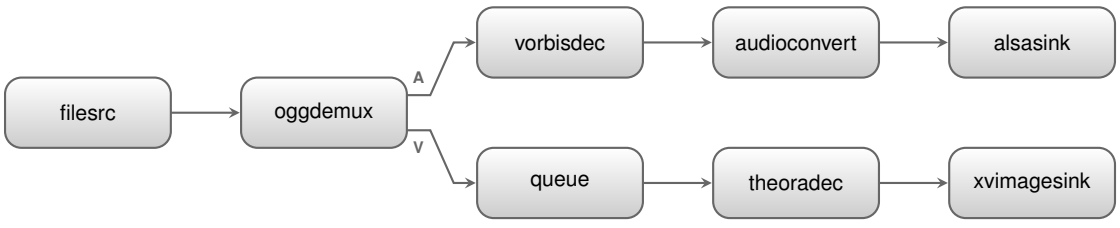


Figura 8.6. Estrutura final do *pipeline* da Listagem 8.4.

De volta à Listagem 8.4, vejamos os trechos relevantes do programa. Na função `main`, as chamadas das linhas 67–69 alocam os elementos "pipeline", "filesrc" e "oggdemux", e as chamadas seguintes, linhas 72–73, adicionam os dois últimos ao *pipeline* e interconecta-os. A chamada da linha 74 atribui o caminho do arquivo Ogg à propriedade "location" do elemento "filesrc". Nesse ponto, portanto, o *pipeline* possui apenas os elementos "filesrc" e "oggdemux", ambos estão conectados e o elemento "filesrc" está configurado com o caminho do arquivo a ser tocado.

A chamada `g_signal_connect`, linha 75, registra a *callback* `pad_added_cb` (linhas 4–54) como tratadora do sinal "pad-added" do elemento "oggdemux", ou seja, a função `pad_added_cb` será chamada sempre que o "oggdemux" criar uma *sometimes pad*.

A chamadas seguintes (linhas 77–85) fazem o mesmo que fizeram nos exemplos anteriores: transicionam o *pipeline* do estado *null* para *playing* (linha 77), bloqueiam a *thread* da aplicação esperando um erro ou o fim da reprodução (linhas 79–80) e, quando o controle retorna à `main`, transicionam o *pipeline* de volta ao estado *null* (linha 84) e liberam os elementos alocados (linha 85). A diferença aqui está no que acontece na transição intermediária do *pipeline* do estado *ready* para o estado *paused*, disparada pela chamada `gst_element_set_state`, linha 77.

Quando o *pipeline* da Listagem 8.4 transiciona para o estado *paused* seus elementos (nesse ponto, apenas "filesrc" e "oggdemux") já estão pausados, ou seja, já carregaram os dados iniciais do fluxo Ogg e estão prontos para começar a operar. Nesse momento, para cada subfluxo multiplexado no fluxo Ogg, o elemento "oggdemux" dispara um sinal "pad-added" que resulta numa chamada à função `pad_added_cb` (linhas 4–54). Por exemplo, se o fluxo Ogg possuir dois subfluxos, um Vorbis e um Theora, a função `pad_added_cb` será chamada uma vez para cada subfluxo. Em ambas as chamadas, o parâmetro `demux` contém o elemento que disparou o sinal ("oggdemux"), `srcpad` contém a *sometimes pad* criada pelo elemento, e `data` contém o dado adicional (*user data*) passado à função `g_signal_connect` (linha 75) no momento do registro da *callback* (no caso, `NULL`).

Na função `pad_added_cb`, a chamada da linha 13 obtém o *pipeline*, a chamada da linha 14 obtém as *caps* (`GstCaps`) associadas à *sometimes pad* criada (parâmetro `srcpad`) e a chamada da linha 15 obtém o tipo (*mime type*) do fluxo que escoará através da *pad*. Nesse ponto, há três possibilidades:

1. Se o tipo é Vorbis ("audio/x-vorbis") e uma *pad* Vorbis ainda não foi encontrada, a *callback* aloca os elementos "vorbisdec", "audioconvert" e "alsasink" (linhas 18–20), adiciona-os ao *pipeline* (linha 22), transiciona os elementos para o mesmo estado do *pipeline*, no caso *paused*, (linhas 23–25), conecta-os em série (linha 26)[2], obtém a *sink pad* do primeiro elemento da série, "vorbisdec", (linha 27), e marca a *flag* `has_vorbis_pad` (linha 28) para indicar que uma *pad* Vorbis foi encontrada.

2. Se o tipo é Theora ("video/x-theora") e uma *pad* Theora ainda não foi encontrada, a *callback* procede de forma análoga, mas agora para os elementos "queue", "theoradec" e "xvimagesink", ou seja, aloca-os, adiciona-os ao *pipeline*, transiciona-os para o mesmo estado do *pipeline*, conecta-os em série, obtém a *source pad* do elemento "queue" e marca a *flag* `has_theora_pad`.

3. Caso contrário, isto é, se o tipo não é Vorbis nem Theora ou se a *pad* em questão não é a primeira Vorbis ou Theora encontrada, a *pad* é simplesmente ignorada (linha 46).

Se a *pad* não for ignorada (casos 1 e 2 anteriores), na linha 48 a variável `sinkpad` conterá a *sink pad* do primeiro elemento da série correspondente à *sometimes pad* criada (parâmetro

[2]Observe que só é possível conectar elementos que estão no mesmo estado. Por isso, antes de conectar os elementos, a chamada `pad_added_cb` primeiro os transiciona para o mesmo estado do *pipeline*.

`srcpad`), ou seja, a *sink pad* do elemento "vorbisdec" se `srcpad` é do tipo Vorbis ou do elemento "queue" se `srcpad` é do tipo Theora. Finalmente, a chamada `gst_pad_link` (linha 48) conecta as *pads* `srcpad` e `sinkpad`, e as chamadas seguintes (linhas 50–53) liberam as referências utilizadas.

```
1  #include <glib.h>
2  #include <gst/gst.h>
3
4  static void pad_added_cb (GstElement *demux, GstPad *srcpad, gpointer data)
5  {
6    GstElement *pipeline;
7    GstCaps *caps;
8    const char *name;
9    GstPad *sinkpad;
10   GstPadLinkReturn ret;
11   static gboolean has_vorbis_pad = FALSE, has_theora_pad = FALSE;
12
13   pipeline = GST_ELEMENT (gst_element_get_parent (demux));
14   caps = gst_pad_query_caps (srcpad, NULL);
15   name = gst_structure_get_name (gst_caps_get_structure (caps, 0));
16   if (g_str_equal (name, "audio/x-vorbis") && !has_vorbis_pad)
17     {
18       GstElement *dec = gst_element_factory_make ("vorbisdec", NULL);
19       GstElement *conv = gst_element_factory_make ("audioconvert", NULL);
20       GstElement *sink = gst_element_factory_make ("alsasink", NULL);
21       g_assert (dec && conv && sink);
22       gst_bin_add_many (GST_BIN (pipeline), dec, conv, sink, NULL);
23       gst_element_sync_state_with_parent (dec);
24       gst_element_sync_state_with_parent (conv);
25       gst_element_sync_state_with_parent (sink);
26       gst_element_link_many (dec, conv, sink, NULL);
27       sinkpad = gst_element_get_static_pad (dec, "sink");
28       has_vorbis_pad = TRUE;
29     }
30   else if (g_str_equal (name, "video/x-theora") && !has_theora_pad)
31     {
32       GstElement *queue = gst_element_factory_make ("queue", NULL);
33       GstElement *dec = gst_element_factory_make ("theoradec", NULL);
34       GstElement *sink = gst_element_factory_make ("xvimagesink", NULL);
35       g_assert (queue && dec && sink);
36       gst_bin_add_many (GST_BIN (pipeline), queue, dec, sink, NULL);
37       gst_element_sync_state_with_parent (queue);
38       gst_element_sync_state_with_parent (dec);
39       gst_element_sync_state_with_parent (sink);
40       gst_element_link_many (queue, dec, sink, NULL);
41       sinkpad = gst_element_get_static_pad (queue, "sink");
42       has_theora_pad = TRUE;
43     }
44   else
45     {
46       goto done;
47     }
48   ret = gst_pad_link (srcpad, sinkpad);
49   g_assert (GST_PAD_LINK_SUCCESSFUL (ret));
50   gst_object_unref (sinkpad);
51 done:
52   gst_caps_unref (caps);
53   gst_object_unref (pipeline);
54 }
55
56 int main (int argc, char *argv[])
57 {
58   GstElement *pipeline, *src, *demux;
59   GstStateChangeReturn ret;
60   GstBus *bus;
61   GstMessage *msg;
62
63   gst_init (&argc, &argv);
```

```
64    if (argc != 2)
65      return (g_printerr ("usage: %s FILE\n", g_get_prgname ()), 1);
66
67    pipeline = gst_element_factory_make ("pipeline", "ogg");
68    src = gst_element_factory_make ("filesrc", "src");
69    demux = gst_element_factory_make ("oggdemux", "demux");
70    g_assert (pipeline && src && demux);
71
72    gst_bin_add_many (GST_BIN (pipeline), src, demux, NULL);
73    gst_element_link_many (src, demux, NULL);
74    g_object_set (G_OBJECT (src), "location", argv[1], NULL);
75    g_signal_connect (demux, "pad-added", G_CALLBACK (pad_added_cb), NULL);
76
77    ret = gst_element_set_state (pipeline, GST_STATE_PLAYING);
78    g_assert (ret != GST_STATE_CHANGE_FAILURE);
79    bus = gst_element_get_bus (pipeline);
80    msg = gst_bus_timed_pop_filtered (bus, GST_CLOCK_TIME_NONE,
81                                      GST_MESSAGE_ERROR | GST_MESSAGE_EOS);
82    gst_message_unref (msg);
83    gst_object_unref (bus);
84    gst_element_set_state (pipeline, GST_STATE_NULL);
85    gst_object_unref (pipeline);
86    return 0;
87  }
```

Listagem 8.4. Tocando um arquivo de vídeo Ogg usando elementos básicos.

Quando iniciado, o programa da Listagem 8.4 demultiplexa o arquivo Ogg fornecido e reproduz os primeiros subfluxos Vorbis e Theora que encontrar. Se não houverem tais subfluxos, o elemento "oggdemux" posta uma mensagem de erro no *bus* (indicando que não há *sink pad* conectada) o que causa o desbloqueio da *thread* da aplicação e, consequentemente, o término do programa.

Um detalhe sutil, mas importante no código da Listagem 8.4 é que a *callback* `pad_added_cb` e a função `main` executam em *threads* diferentes. Consequentemente, caso elas compartilhem dados, para evitar condições de corrida, é necessário sincronizar o acesso a esses dados via *mutexes* e semáforos (tipos `GMutex` e `GCond` da GLib). Na Listagem 8.4 não precisamos fazer isso porque as chamadas `gst_element_*` e `gst_bin_*` que usamos na *callback* para manipular o "pipeline" (único dado compartilhado) já são protegidas internamente por *mutexes*. Além disso, no intervalo em que as *threads* da *callback* e da aplicação estão concorrendo, linhas 77–80, a *thread* da aplicação não modifica o *pipeline*.

Os comandos gst-launch e gst-inspect

Os programas que vimos até agora (Listagens 8.1, 8.3 e 8.4) têm uma característica em comum: a topologia do *pipeline* não muda depois que ele é iniciado (transiciona para o estado *playing*). O comando *gst-launch*, instalado pelo pacote *gstreamer*, permite construir *pipelines* desse tipo diretamente na linha de comando.[3] Por exemplo, os três comandos seguintes constroem e iniciam *pipelines* idênticos aos das Listagens 8.1, 8.3 e 8.4:

```
$ gst-launch playbin uri="file://$PWD/bunny.ogg"
$ gst-launch filesrc location=bunny.mp3 ! mad ! alsasink
$ gst-launch filesrc location=bunny.ogg ! oggdemux name=demux\
               demux. ! vorbisdec ! audioconvert ! alsasink\
               demux. ! queue     ! theoradec    ! xvimagesink
```

[3]Em algumas instalações o nome do comando é *gst-launch-1.0*.

O primeiro comando anterior cria um *pipeline* contendo apenas o elemento "playbin", atribui a URI do vídeo a ser tocado à propriedade "uri" do elemento e inicia o *pipeline*. O segundo comando cria um *pipeline* com os elementos "filesrc", "mad" e "alsasink" conectados em série, atribui o caminho do vídeo à propriedade "location" do "filesrc" e incia o *pipeline*. E o terceiro comando cria um *pipeline* preliminar contento apenas os elementos "filesrc" e "oggdemux" conectados, atribui o caminho do vídeo à propriedade "location" do "filesrc" e inicia o *pipeline*; em seguida, quando o "oggdemux" instancia suas *sometimes pads*, o *gst-launch* adiciona e conecta as componentes conexas restantes, a saber, a série "vorbisdec", "audioconvert" e "alsasink" e a série "queue", "theoradec", e "xvimagesink".

O comando *gst-launch* é particularmente útil para depurar ou testar a viabilidade de *pipelines* antes de implementá-los em C. Por exemplo, usando o *gst-launch* podemos facilmente construir uma versão genérica do *pipeline* da Listagem 8.4, isto é, construir um *pipeline* que demultiplexa e decodifica um fluxo de mídia arbitrário (não apenas Ogg), obtido de uma fonte arbitrária (não apenas do disco), e que reproduz os fluxos resultantes num ambiente arbitrário (não apenas GNU/Linux). O comando, nesse caso, é o seguinte:

```
$ gst-launch uridecodebin uri="file://$PWD/bunny.ogg" name=decbin\
              decbin. ! audioconvert ! autoaudiosink\
              decbin. ! autovideosink
```

Esse *pipeline* possui quatro elementos: "uridecodebin", "autovideosink", "audioconvert" e "autoaudiosink". O elemento "uridecodebin" é um *bin* que demultiplexa e decodifica uma URI, isto é, ele lê o conteúdo de uma URI, demultiplexa-o, decodifica seus subfluxos e escreve os subfluxos decodificados em *sometimes pads* correspondentes. O elemento "audioconvert" é um conversor de formato de áudio PCM. E os elementos "autoaudiosink" e "autovideosink" são *sinks* abstratos de áudio e vídeo que detectam e usam os *sinks* concretos mais adequados para um dado ambiente. Por exemplo, no nosso ambiente GNU/Linux os elementos "autoaudiosink" e "autovideosink" utilizam os *sinks* concretos "alsasink" e "xvimagesink", os mesmos que usamos nos programas das Listagens 8.3 e 8.4, mas em outro ambiente essa escolha pode ser diferente. A Figura 8.7 apresenta a estrutura do *pipeline* criado pelo comando *gst-launch* anterior.

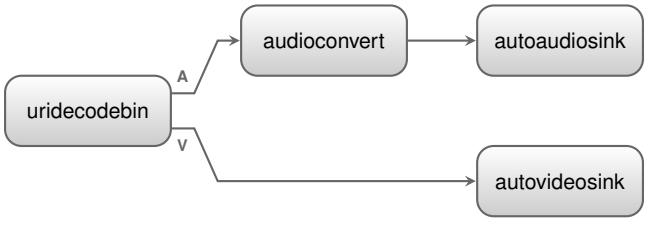


Figura 8.7. *Pipeline* genérico para reprodução de áudio e vídeo.

Além de *pipelines* para reprodução de áudio e vídeo, o comando *gst-launch* pode ser usado para criar *pipelines* que decodificam, transcodificam e transmitem fluxos de mídia na rede. Para mais informações sobre a sintaxe do comando (que é inspirada na sintaxe de *pipelines* da Bourne Shell), opções suportadas e exemplos de uso, veja a página de manual do *gst-launch* ("`man gst-launch`").

Outro programa instalado pelo pacote *gstreamer* é o *gst-inspect*, que permite inspecionar os *plugins* e elementos disponíveis no sistema.[4] Quando executado sem argumentos o *gst-inspect* imprime uma lista com os *plugins* e elementos disponíveis—no nosso ambiente, por exemplo, há 223 *plugins* e 1302 elementos disponíveis. Já quando executado com um nome de *plugin* ou nome de elemento como argumento o *gst-inspect* apresenta as informações do dado *plugin* ou elemento. A Listagem 8.5 apresenta a saída do comando "`gst-inspect equalizer`" que lista as informações do *plugin* "equalizer".

```
1  Plugin Details:
2    Name                     equalizer
3    Description              GStreamer audio equalizers
4    Filename                 /usr/lib/gstreamer-1.0/libgstequalizer.so
5    Version                  1.8.3
6    License                  LGPL
7    Source module            gst-plugins-good
8    Source release date      2016-08-19
9    Binary package           GStreamer Good Plugins (Arch Linux)
10   Origin URL               http://www.archlinux.org/
11
12   equalizer-nbands: N Band Equalizer
13   equalizer-3bands: 3 Band Equalizer
14   equalizer-10bands: 10 Band Equalizer
15
16   3 features:
17   +-- 3 elements
```

Listagem 8.5. Saída do comando "`gst-inspect equalizer`" no Arch Linux 4.7.4.

Na Listagem 8.5, as linhas 2–4 apresentam o nome do *plugin* ("equalizer"), a sua descrição (equalizadores de áudio) e o caminho da biblioteca dinâmica contendo o código do *plugin*. As linhas 5–10, apresentam informações adicionais (versão, licença, pacote, etc.) e as linhas 12–14 listam os elementos includidos no *plugin*, no caso, "equalizer-nbands" (equalizador de *n* bandas), "equalizer-3bands" (equalizador de 3 bandas) e "equalizer-10bands" (equalizador de 10 bandas). Para listar as informações de um desses elementos, "equalizer-3bands" por exemplo, basta rodar o comando "`gst-inspect equalizer-3bands`". A saída nesse caso contém informações como a hierarquia de tipos do elemento, as interfaces que ele implementa, os *templates* de suas *pads*, as *pads* propriamente ditas (no caso, uma *source pad* com disponibilidade *always* do tipo "audio/x-raw", e uma *sink pad* também *always* e "audio/x-raw") e as propriedades do elemento. A Listagem 8.6 apresenta o trecho da saída do comando anterior em que as propriedades "band0", "band1" e "band2" são descritas. Pela descrição, essas propriedades controlam o ganho (dB) que o elemento aplica às faixas de frequência de 100Hz, 1100Hz e 11kHz; as três recebem valores no intervalo real [-24,12] e são inicializadas com zero.

```
1  band0: gain for the frequency band 100 Hz, ranging from -24.0 to +12.0
2         flags: readable, writable, controllable
3         Double. Range: -24 - 12 Default: 0
4  band1: gain for the frequency band 1100 Hz, ranging from -24.0 to +12.0
5         flags: readable, writable, controllable
6         Double. Range:  -24 - 12 Default: 0
7  band2: gain for the frequency band 11 kHz, ranging from -24.0 to +12.0
8         flags: readable, writable, controllable
9         Double. Range:  -24 -12 Default: 0
```

Listagem 8.6. Trecho da seção "Element Properties" da saída do comando "`gst-inspect equalizer-3bands`" no Arch Linux 4.7.4.

[4]Em algumas instalações o nome do comando é *gst-inspect-1.0*.

Os elementos equalizadores instalados pelo *plugin* "equalizer" anterior são processadores de áudio que permitem modificar o ganho dos graves, médios e agudos das amostras de um fluxo de áudio descomprimido. Os três elementos, "equalizer-nbands", "equalizer-3bands" e "equalizer-10bands", possuem a mesma estrutura: todos têm uma *source pad* que consome áudio descomprimido e uma *sink pad* que produz audio descomprimido, e todos eles processam uma amostra por vez, isto é, leem uma amostra da sua *source pad*, modificam-na acordo com os valores de ganho especificados em suas propriedades e escrevem a amostra resultante na sua *sink pad*. Elementos que realizam esse tipo de processamento são chamados de filtros.

8.4. Filtros

Um filtro é um elemento que aplica uma transformação sobre um fluxo de amostras descomprimidas. A transformação é normalmente ortogonal (depende apenas do conteúdo das amostras) e altera o fluxo original afim de produzir efeitos sonoros ou visuais no fluxo resultante. A Tabela 8.1 apresenta alguns filtros de áudio e vídeo disponíveis no GStreamer.

	Plugin	**Elemento**	**Descrição (controle)**
Áudio	audiofx	audiodynamic	Compressão ou expansão
	audiofx	audioecho	Eco
	audiofx	audiopanorama	Panorama estéreo
	equalizer	equalizer-nbands	Equalização (n bandas)
	freeverb	freeverb	Reverberação
	soundtouch	pitch	Tonalidade e tempo
	speed	speed	Velocidade
	volume	volume	Volume
Vídeo	coloreffects	coloreffects	Efeito de colorização
	effectv	dicetv	Efeito de fatiamento
	effectv	edgetv	Efeito de borda
	effectv	revtv	Efeito de relevo
	effectv	shagadelictv	Efeito de espiral caleidoscópica
	videocrop	videocrop	Recorte
	videofilter	videobalance	Brilho, cor e saturação
	videoscale	videoscale	Redimensionamento

Tabela 8.1. Alguns filtros de áudio e vídeo disponíveis no GStreamer.

Um filtro possui exatamente uma *source pad* e uma *sink pad*; ambas as *pads* têm disponibilidade *always* e escoam apenas fluxos descomprimidos ("audio/x-raw" ou "video/x-raw"). Consequentemente, num *pipeline* os elementos filtros normalmente aparecem entre os decodificadores e os *sinks*. Por exemplo, o comando a seguir adiciona o filtro "volume" ao *pipeline* da Figura 8.4, que reproduz um arquivo de áudio MP3.

```
$ gst-launch filesrc location=bunny.mp3\
              ! mad ! volume volume=.5 ! alsasink
```

Nesse caso, o áudio é reproduzido com a metade do volume original—a propriedade "volume" do elemento "volume", inicializada com o valor .5, indica ao elemento que o volume das amostras deve ser reduzido à metade.

Vejamos agora um exemplo mais complexo. O *shell script* da Listagem 8.7 usa o comando *gst-launch* para reproduzir um vídeo com um determinado efeito audiovisual. O *script* recebe dois argumentos, a URI do vídeo a ser reproduzido e o efeito a ser aplicado (número entre 0–4) e usa o comando *gst-launch* para montar o *pipeline* correspondente.

```
1  case "$2" in
2    0) af='volume volume=2';                        vf='coloreffects preset=1';;
3    1) af='pitch pitch=.5';                         vf='dicetv';;
4    2) af='audioecho delay=1000000000 intensity=1'; vf='shagadelictv';;
5    3) af='freeverb room-size=1 level=1';           vf='edgetv';;
6    4) af='equalizer-3bands band0=12 band1=-24';    vf='revtv';;
7  esac
8  if test -z "$1" || test -z "$vf" || test -z "$af" ; then
9    echo >&2 "usage: ${0##*/} URI [0-4]"; exit 1
10 fi
11 gst-launch uridecodebin uri="$1" name=decbin\
12   decbin. ! $af ! audioconvert ! autoaudiosink\
13   decbin. ! videoconvert ! $vf ! videoconvert ! autovideosink
```

Listagem 8.7. *Shell script* que reproduz um vídeo com efeito audiovisual.

A Figura 8.8 apresenta a estrutura final do *pipeline* criado pelo *script* da Listagem 8.7. Na figura, os elementos ⟨*audio filter*⟩ e ⟨*video filter*⟩ variam de acordo com o segundo argumento do *script*, e os elementos "audioconvert" e "videoconvert" garantem que os filtros selecionados gerem (e recebam, no caso do filtro de vídeo) amostras no formato esperado. Há cinco combinações de filtros possíveis:

- Se o argumento é 0, pela linha 2, o *script* utiliza o filtro de áudio "volume" (com "volume" 2) e o filtro de vídeo "coloreffects" (com "preset" 1). Nesse caso, as amostras de áudio são reproduzidas com o dobro do volume original e as amostras de vídeo são colorizadas para simular o efeito "câmera infravermelho" (Figura 8.9b).
- Se o argumento é 1, pela linha 3, o *script* utiliza o filtro de áudio "pitch" (com "pitch" .5) e o filtro de vídeo "dicetv". Nesse caso, o tom (frequência) do áudio diminui para a metade do original e o vídeo ganha um efeito de fatiamento (Figura 8.9c).
- Se o argumento é 2, pela linha 4, o *script* utiliza o filtro de áudio "audioecho" (com "delay" 1s e "intensity" 1) e o filtro de vídeo "shagadelictv". Nesse caso, o áudio ganha um efeito de eco e o vídeo ganha um efeito de espiral caleidoscópica (Figura 8.9d).
- Se o argumento é 3, pela linha 5, *script* utiliza o filtro áudio "freeverb" (com "room-size" 1 e "level" 1) e o filtro de vídeo "edgetv". Nesse caso, o áudio ganha um efeito de reverberação numa sala ampla e o vídeo ganha um efeito de borda (Figura 8.9e).
- Finalmente, se o argumento é 4, pela linha 6, o *script* utiliza o filtro de áudio "equalizer-3bands" (com "band0" 12dB e "band1" -24dB) e o filtro de vídeo "revtv". Nesse caso, o áudio têm os graves acentuados e médios suprimidos, e o video ganha um efeito de relevo (Figura 8.9f).

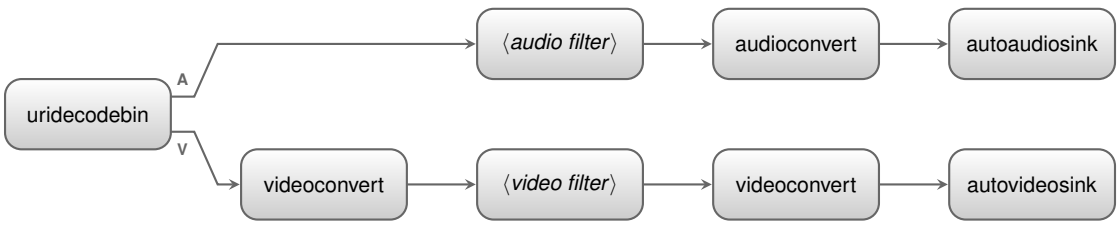


Figura 8.8. Estrutura final do *pipeline* da Listagem 8.7.

(a) Quadro original. (b) Filtro “coloreffects”. (c) Filtro “dicetv”.

(d) Filtro “shagadelictv”. (e) Filtro “edgetv”. (f) Filtro “revtv”.

Figura 8.9. Efeitos de vídeo aplicados pelo *script* da Listagem 8.7.

A Listagem 8.8 apresenta um trecho do programa C equivalente ao *script* da Listagem 8.7. O trecho corresponde ao código da *callback* `pad_added_cb` disparada pelo sinal “pad-added” do elemento “uridecodebin”. O código da função `main` apesar de omitido é trivial. Ele cria um elemento “uridecodebin”, atribui à sua propriedade “uri” o primeiro argumento do programa, registra em seu sinal “pad-added” a *callback* `pad_added_cb` e passa como dado opcional da *callback* o segundo argumento do programa (número entre 0–4 que determina os filtros selecionados). Em seguida, a *thread* da aplicação transiciona o *pipeline* para *playing* e aguarda o fim da reprodução antes de liberar os dados alocados e terminar o programa. A operação da *callback* é similar àquela da Listagem 8.4. A diferença aqui é que os filtros são criados de acordo com o segundo argumento da aplicação (inteiro passado por referência à *callback* via o parâmetro `data`).

```
1  static void pad_added_cb (GstElement *src, GstPad *srcpad, gpointer data)
2  {
3    GstElement *pipeline;
4    GstCaps *caps;
5    const char *name;
6    GstPad *sinkpad;
7    GstPadLinkReturn ret;
8    gint filterno = *((gint *) data);
9    static gboolean has_audio_pad = FALSE, has_video_pad = FALSE;
10
11   pipeline = GST_ELEMENT (gst_element_get_parent (src));
12   caps = gst_pad_query_caps (srcpad, NULL);
13   name = gst_structure_get_name (gst_caps_get_structure (caps, 0));
14   if (g_str_equal (name, "audio/x-raw") && !has_audio_pad)
15     {
16       GstElement *filter = NULL, *conv, *sink;
17       switch (filterno)
18         {
19         case 0:
20           filter = gst_element_factory_make ("volume", NULL);
21           g_object_set (G_OBJECT (filter), "volume", 2.0, NULL);
22           break;
23         case 1:
24           filter = gst_element_factory_make ("pitch", NULL);
25           g_object_set (G_OBJECT (filter), "pitch", .5, NULL);
26           break;
27         case 2:
28           filter = gst_element_factory_make ("audioecho", NULL);
29           g_object_set (G_OBJECT (filter), "delay", GST_SECOND, "intensity", 1., NULL);
30           break;
```

```
31          case 3:
32            filter = gst_element_factory_make ("freeverb", NULL);
33            g_object_set (G_OBJECT (filter), "room-size", 1., "level", 1., NULL);
34            break;
35          case 4:
36            filter = gst_element_factory_make ("equalizer-3bands", NULL);
37            g_object_set (G_OBJECT (filter), "band0", 12., "band1", -24., NULL);
38            break;
39          }
40        conv = gst_element_factory_make ("audioconvert", NULL);
41        sink = gst_element_factory_make ("autoaudiosink", NULL);
42        g_assert (filter && conv && sink);
43        gst_bin_add_many (GST_BIN (pipeline), filter, conv, sink, NULL);
44        gst_element_sync_state_with_parent (filter);
45        gst_element_sync_state_with_parent (conv);
46        gst_element_sync_state_with_parent (sink);
47        gst_element_link_many (filter, conv, sink, NULL);
48        sinkpad = gst_element_get_static_pad (filter, "sink");
49        has_audio_pad = TRUE;
50      }
51    else if (g_str_equal (name, "video/x-raw") && !has_video_pad)
52      {
53        GstElement *pre, *filter = NULL, *pos, *sink;
54        switch (filterno)
55          {
56          case 0:
57            filter = gst_element_factory_make ("coloreffects", NULL);
58            g_object_set (G_OBJECT (filter), "preset", 1, NULL);
59            break;
60          case 1:
61            filter = gst_element_factory_make ("dicetv", NULL);
62            break;
63          case 2:
64            filter = gst_element_factory_make ("shagadelictv", NULL);
65            break;
66          case 3:
67            filter = gst_element_factory_make ("edgetv", NULL);
68            break;
69          case 4:
70            filter = gst_element_factory_make ("revtv", NULL);
71            break;
72          }
73        pre = gst_element_factory_make ("videoconvert", NULL);
74        pos = gst_element_factory_make ("videoconvert", NULL);
75        sink = gst_element_factory_make ("autovideosink", NULL);
76        g_assert (pre && filter && pos && sink);
77        gst_bin_add_many (GST_BIN (pipeline), pre, filter, pos, sink, NULL);
78        gst_element_sync_state_with_parent (pre);
79        gst_element_sync_state_with_parent (filter);
80        gst_element_sync_state_with_parent (pos);
81        gst_element_sync_state_with_parent (sink);
82        gst_element_link_many (pre, filter, pos, sink, NULL);
83        sinkpad = gst_element_get_static_pad (pre, "sink");
84        has_video_pad = TRUE;
85      }
86    else
87      {
88        goto done;
89      }
90    ret = gst_pad_link (srcpad, sinkpad);
91    g_assert (GST_PAD_LINK_SUCCESSFUL (ret));
92    gst_object_unref (sinkpad);
93   done:
94    gst_caps_unref (caps);
95    gst_object_unref (pipeline);
96  }
```

Listagem 8.8. Trecho do programa C equivalente ao *script* da Listagem 8.7.

8.5. Adicionando os controles *play*, *pause*, *stop*, *seek*, *fast-foward* e *rewind*

Os programas que vimos até agora não são interativos. Assim que inciados, eles constroem um *pipeline* que reproduz o conteúdo uma única vez do inicio ao fim. Nesta seção, vamos adicionar os controles *play*, *pause*, *stop*, *seek*, *fast-forward* e *rewind* ao programa "Olá mundo" da Listagem 8.1. Nosso objetivo é fazer com que o programa responda aos seguintes comandos de tecla:

- SPC — pausa ou resume a reprodução (*pause* e *play*)
- → — avança 5s (*seek*)
- ← — retrocede 5s (*seek*)
- F — reproduz 2x mais rápido (*fast-foward*)
- R — reproduz ao contrário 2x mais rápido (*rewind*)
- N — reproduz na velocidade original
- Q — para a reprodução e termina o programa (*stop*)

Escolhemos o programa "Olá mundo" apenas para simplificar as listagens apresentadas. Os mecanismos para captura de teclas e controle do *pipeline* descritos sobre esse exemplo são gerais e se aplicam a qualquer aplicação GStreamer. Para obter teclas vamos capturar as mensagens de eventos de navegação (`GST_NAVIGATION_MESSAGE_EVENT`) postadas pelo *sink* de vídeo no *bus* do *pipeline*. Para implementar as operações de *play*, *pause* e *stop*, vamos transicionar o *pipeline* para os estados correspondentes. E para implementar as operações *seek*, *fast-foward* e *rewind*, vamos postar eventos de *seek* (`GST_EVENT_SEEK`) que requisitam mudanças na posição e na taxa de reprodução do conteúdo.

A Listagem 8.9 apresenta o trecho de código contendo a função `main` do novo programa "Olá mundo" (versão com controles). Até a linha 27, o código da função é idêntico ao código correspondente na Listagem 8.1. A diferença aqui está na forma como capturamos mensagens postadas no *bus*. Até agora, para obter mensagens do *bus*, usamos a função `gst_bus_timed_pop_filtered`, que bloqueia a *thread* da aplicação até que tipos específicos de mensagens sejam postados. Na Listagem 8.9, como a aplicação é orientada a eventos, usamos uma técnica diferente, baseada no *loop* de eventos da GLib.

O *loop* de eventos é criado pela chamada `g_main_loop_new` na linha 28 da Listagem 8.9. O primeiro argumento da chamada (`NULL`) indica que o *loop* irá receber eventos de qualquer fonte registrada via GLib, e o segundo argumento (`FALSE`) indica que o *loop* não deve ser iniciado imediatamente (iniciaremos o *loop* posteriormente).

As chamadas `g_object_set_data` seguintes (linhas 29–30) armazenam no *bus* ponteiros para o *pipeline* e para o *loop*. O *bus* (`GstBus`) também é um objeto (`GObject`) e, como todo objeto, possui uma tabela *hash* em que podem ser armazenados dados de usuário (*user data*). As chamadas das linhas 29–30 armazenam na *hash* do *bus* um ponteiro para o *pipeline*, indexado pela chave "pipeline", e um ponteiro para o*loop*, indexado pelo *loop*. Isso é necessário porque mais tarde, na *callback* do *bus*, não temos acesso direto à esses objetos.

A chamada `gst_bus_add_watch` (linha 31) registra a *callback* `bus_cb` como tratadora (*listener*) de mensagens *bus*. Ou seja, a *callback* será chamada no *loop* de eventos sempre que uma mensagem é postada no *bus*. E a chamada seguinte (linha 32) libera a referência obtida na linha 27.

A chamada `g_main_loop_run` (linha 34) inicia o *loop* de eventos da GLib. A partir desse ponto a aplicação torna-se orientada a eventos, isto é, a *thread* da aplicação passa a ser controlada pela GLib que apenas espera eventos e repassa-os às *callbacks* registradas (`bus_cb`, no caso). O controle só volta à `main` quando o *loop* é terminado explicitamente via uma chamada `g_main_loop_quit`—o que ocorre na *callback* do *bus*. Nesse caso, as chamadas restantes (linhas 36–38) liberam os dados alocados e terminam o programa.

```
1  #include <glib.h>
2  #include <gst/gst.h>
3  #include <gst/video/navigation.h>
4
5  static gboolean bus_cb (GstBus *bus, GstMessage *msg, gpointer data);
6
7  int main (int argc, char *argv[])
8  {
9    GstElement *playbin;
10   char *uri;
11   GstStateChangeReturn ret;
12   GstBus *bus;
13   GMainLoop *loop;
14
15   gst_init (&argc, &argv);
16   playbin = gst_element_factory_make ("playbin", "hello");
17   g_assert_nonnull (playbin);
18
19   uri = gst_filename_to_uri ("bunny.ogg", NULL);
20   g_assert_nonnull (uri);
21   g_object_set (G_OBJECT (playbin), "uri", uri, NULL);
22   g_free (uri);
23
24   ret = gst_element_set_state (playbin, GST_STATE_PLAYING);
25   g_assert (ret != GST_STATE_CHANGE_FAILURE);
26
27   bus = gst_element_get_bus (playbin);
28   loop = g_main_loop_new (NULL, FALSE);
29   g_object_set_data (G_OBJECT (bus), "pipeline", playbin);
30   g_object_set_data (G_OBJECT (bus), "loop", loop);
31   gst_bus_add_watch (bus, bus_cb, NULL);
32   gst_object_unref (bus);
33
34   g_main_loop_run (loop);       /* event loop */
35
36   g_main_loop_unref (loop);
37   gst_element_set_state (playbin, GST_STATE_NULL);
38   gst_object_unref (playbin);
39   return 0;
40 }
```

Listagem 8.9. Função `main` do programa "Olá mundo" com controles.

Vejamos agora a operação da *callback* `bus_cb`, cujo código é apresentado na Listagem 8.10. A cada mensagem postada no *bus*, o *loop* de eventos chama essa *callback* com três argumentos: o *bus*, a mensagem postada e o dado adicional passado à função `gst_bus_add_watch` (linha 31 da Listagem 8.9) no momento do registro da *callback* (no caso, `NULL`).

Na Listagem 8.10, as chamadas das linhas 3 e 4 obtém as referências para o *pipeline* e para o *loop* armazenadas na *hash* do *bus*. E o comando `switch` seguinte (linhas 5–37) trata a mensagem recebida de acordo com o seu tipo. Há três possibilidades:

1. Se a mensagem indica um erro ou o fim do fluxo (EOS), a *callback* termina o *loop* de eventos (linha 40) e remove a si mesmo da lista de tratadores (instrução "`return FALSE`" na linha 41).

2. Se a mensagem é proveniente de um elemento (linha 10), é uma mensagem de evento de navegação (linhas 14–15) e indica um evento de pressionamento de tecla (linhas 18–19), então a *callback* executa a operação associada à tecla pressionada. Ou seja, a *callback* chama a função auxiliar `toggle_pause` se a tecla é "space", chama a função auxiliar `seek` se a tecla é "right" ou "left", chama a função auxiliar `speed` se a tecla é "f", "r" ou "n", e termina o *loop* se a tecla é "q".

3. Caso contrário, a mensagem é ignorada (linha 38).

```
1  static gboolean bus_cb (GstBus *bus, GstMessage *msg, gpointer data)
2  {
3    GstElement *pipeline = g_object_get_data (G_OBJECT (bus), "pipeline");
4    GMainLoop *loop = g_object_get_data (G_OBJECT (bus), "loop");
5    switch (GST_MESSAGE_TYPE (msg))
6      {
7      case GST_MESSAGE_ERROR:
8      case GST_MESSAGE_EOS:
9        goto quit;
10     case GST_MESSAGE_ELEMENT:
11       {
12         GstEvent *evt;
13         const char *key;
14         if (gst_navigation_message_get_type (msg) != GST_NAVIGATION_MESSAGE_EVENT)
15           break;
16         if (!gst_navigation_message_parse_event (msg, &evt))
17           break;
18         if (gst_navigation_event_get_type (evt) != GST_NAVIGATION_EVENT_KEY_PRESS)
19           break;
20         gst_navigation_event_parse_key_event (evt, &key);
21         if (g_ascii_strcasecmp (key, "space") == 0)
22           toggle_pause (pipeline);
23         else if (g_ascii_strcasecmp (key, "right") == 0)
24           seek (pipeline, 5 * GST_SECOND);
25         else if (g_ascii_strcasecmp (key, "left") == 0)
26           seek (pipeline, -5 * GST_SECOND);
27         else if (g_ascii_strcasecmp (key, "f") == 0)
28           speed (pipeline, 2.);
29         else if (g_ascii_strcasecmp (key, "r") == 0)
30           speed (pipeline, -2.);
31         else if (g_ascii_strcasecmp (key, "n") == 0)
32           speed (pipeline, 1.);
33         else if (g_ascii_strcasecmp (key, "q") == 0)
34           goto quit;
35         break;
36       }
37     }
38   return TRUE;
39 quit:
40   g_main_loop_quit (loop);
41   return FALSE;
42 }
```

Listagem 8.10. Função `bus_cb` do programa "Olá mundo" com controles.

A Listagem 8.11 apresenta o código das funções auxiliares `toggle_pause`, `seek` e `resume` chamadas pela *callback* `bus_cb` da Listagem 8.10.

A função `toggle_pause` (linhas 1–12) alterna o estado do *pipeline* entre *paused* e *playing* dependendo do estado atual: se o *pipeline* está pausado ela o inicia, e se o *pipeline* está tocando ela o pausa.

A função `seek` (linhas 14–23) requisita um salto de `offset` nanosegundos (ns) na reprodução do conteúdo a partir do seu ponto atual. A chamada da linha 17 obtém o tempo absoluto do *pipeline* (ns) e armazena-o em `from`. A atribuição seguinte calcula o tempo absoluto (ns) em que o *pipeline* deve estar após o salto e armazena-o em `to`. E a chamada `gst_element_seek_simple` (linhas 19–22) posta um evento de *seek* no *pipeline* que requisita um salto na reprodução para o novo tempo absoluto `to`. A função `gst_element_seek_simple` recebe quatro argumentos: o elemento alvo da operação, o formato do deslocamento, *flags* opcionais e o valor do deslocamento. O formato (segundo argumento) indica o formato em que o descolocamento está especificado—no caso, `GST_FORMAT_TIME` indica um deslocamento no tempo. As *flags* (terceiro argumento) determinam as características da operação: a *flag* `GST_SEEK_FLAG_ACCURATE` indica que a operação deve ter uma precisão razoável, a *flag* `GST_SEEK_FLAG_FLUSH` indica que *caches* internos de elementos subsequentes devem ser descartados e a *flag* `GST_SEEK_FLAG_FLUSH` indica que (se possível) durante a operação apenas quadros chave (*key frames*) devem ser decodificados.

A última função, `speed`, (linhas 25–49) requisita uma mudança na taxa de reprodução do *pipeline* de acordo com o parâmetro *rate*. Esse tipo de mudança de taxa é feita por meio da função `gst_element_seek`, também utilizada para realizar operações de *seek* complexas. Essa função recebe oito argumentos: (1) o elemento alvo; (2) a nova taxa de reprodução; (3) o formato dos deslocamentos; (4) *flags* opcionais; (5) o tipo do início do desolamento; (6) o novo inicio; (7) o tipo do fim do deslocamento; e (8) o novo fim. Como aqui nosso objetivo é apenas alterar a taxa de reprodução o deslocamento é a própria posição atual do *fluxo* (variável `from` preenchida na linha 31). Se a taxa é positiva, o deslocamento é relativo ao início (linhas 34–37), caso contrário ele é relativo ao fim (linhas 41–44). Na prática, valores de taxa superiores a 1 aceleram a reprodução, valores entre 0 e 1 desaceleram a reprodução produzindo um efeito de "câmera lenta", e valores negativos aplicam uma aceleração negativa, o que faz com que o conteúdo seja tocado ao contrário.

```
1  static void toggle_pause (GstElement *pipeline)
2  {
3    switch (GST_STATE (pipeline))
4      {
5      case GST_STATE_PAUSED:
6        gst_element_set_state (pipeline, GST_STATE_PLAYING);
7        break;
8      case GST_STATE_PLAYING:
9        gst_element_set_state (pipeline, GST_STATE_PAUSED);
10       break;
11     }
12 }
13
14 static void seek (GstElement *pipeline, GstClockTimeDiff offset)
15 {
16   GstClockTimeDiff from, to;
17   gst_element_query_position (pipeline, GST_FORMAT_TIME, &from);
```

```
18    to = MAX (from + offset, 0);
19    gst_element_seek_simple (pipeline, GST_FORMAT_TIME,
20                             GST_SEEK_FLAG_ACCURATE
21                             | GST_SEEK_FLAG_FLUSH
22                             | GST_SEEK_FLAG_TRICKMODE, to);
23  }
24
25  static void speed (GstElement *pipeline, double rate)
26  {
27    GstClockTimeDiff from;
28    GstSeekType start, stop;
29    GstClockTimeDiff start_time, stop_time;
30
31    gst_element_query_position (pipeline, GST_FORMAT_TIME, &from);
32    if (rate >= 0.)
33      {
34        start = GST_SEEK_TYPE_SET;
35        start_time = from;
36        stop = GST_SEEK_TYPE_NONE;
37        stop_time = GST_CLOCK_TIME_NONE;
38      }
39    else
40      {
41        start = GST_SEEK_TYPE_NONE;
42        start_time = GST_CLOCK_TIME_NONE;
43        stop = GST_SEEK_TYPE_SET;
44        stop_time = from;
45      }
46    gst_element_seek (pipeline, rate, GST_FORMAT_TIME,
47                      GST_SEEK_FLAG_FLUSH | GST_SEEK_FLAG_TRICKMODE,
48                      start, start_time, stop, stop_time);
49  }
```

Listagem 8.11. Funções `toggle_paused`, `seek` e `speed` do programa "Olá mundo" com controles.

Finalmente, observe que o programa anterior usa as declarações do cabeçalho "gstnavigation.h" (Listagem 8.9, linha 3) instalado pelo pacote *gst-plugins-base*. Ou seja, para compilar o programa é necessário informar ao pré-processador o caminho do diretório contendo esse cabeçalho e ao *linker* os nomes e os caminhos dos diretórios contendo as bibliotecas correspondentes. Para tal basta adicionar o módulo "gstreamer-video-1.0" à variável `MODULES` (linha 2) do *makefile* da Listagem 8.2.

8.6. Plugins

Nesta seção, vamos discutir como novos elementos podem ser criados, encapsulados em *plugins* (bibliotecas dinâmicas), instalados e usados por aplicações GStreamer. Mais especificamente, vamos escrever dois elementos: "myfilter" e "myvideofilter". O elemento "myfilter" é um filtro genérico que lê uma amostra da sua *sink pad*, imprime o seu tamanho na saída padrão e repassa-a à sua *source pad*. Já o elemento "myvideofilter" é um filtro de vídeo que aplica um efeito de ondulação aos quadros que o atravessam.

Elemento "myfilter"

A Listagem 8.12 apresenta o código do elemento "myfilter" e do *plugin* correspondente (também chamado "myfilter"). Na listagem, a estrutura `GstMyFilter` (linhas 4–7) contém os dados de uma instância de um elemento "myfilter", a saber, os dados do tipo pai (`GstElement`) e ponteiros para as *pads* do filtro. A diretiva da linha 9 define uma constante simbólica com o tipo do novo elemento, e as macros seguintes (linhas 10 e 11) declaram e definem o tipo propriamente dito (`GstMyFilter`).

```
#include <glib.h>
#include <gst/gst.h>

typedef struct _GstMyFilter {
  GstElement parent_instance;
  GstPad *sinkpad, *srcpad;
} GstMyFilter;

#define GST_TYPE_MY_FILTER (gst_my_filter_get_type ())
G_DECLARE_FINAL_TYPE (GstMyFilter, gst_my_filter, GST, MY_FILTER, GstElement)
G_DEFINE_TYPE (GstMyFilter, gst_my_filter, GST_TYPE_ELEMENT)

static GstStaticPadTemplate sink_template =
  GST_STATIC_PAD_TEMPLATE ("sink", GST_PAD_SINK,
                           GST_PAD_ALWAYS, GST_STATIC_CAPS ("ANY"));

static GstStaticPadTemplate src_template =
  GST_STATIC_PAD_TEMPLATE ("src", GST_PAD_SRC,
                           GST_PAD_ALWAYS, GST_STATIC_CAPS ("ANY"));

static GstFlowReturn gst_my_filter_chain (GstPad *pad, GstObject *obj, GstBuffer *buf)
{
  GstMyFilter *filter = GST_MY_FILTER (obj);
  g_print ("Processed %" G_GSIZE_FORMAT" bytes\n", gst_buffer_get_size (buf));
  return gst_pad_push (filter->srcpad, buf);
}

static void gst_my_filter_class_init (GstMyFilterClass *klass)
{
  GstElementClass *gstelement_class = GST_ELEMENT_CLASS (klass);
  gst_element_class_set_static_metadata
    (gstelement_class, "An example filter", "Example/myfilter",
     "Example filter", "roberto@telemidia.puc-rio.br");
  gst_element_class_add_pad_template
    (gstelement_class, gst_static_pad_template_get (&src_template));
  gst_element_class_add_pad_template
    (gstelement_class, gst_static_pad_template_get (&sink_template));
}

static void gst_my_filter_init (GstMyFilter *filter)
{
  GstPad *sink = gst_pad_new_from_static_template (&sink_template, "sink");
  GstPad *src = gst_pad_new_from_static_template (&src_template, "src");
  gst_pad_set_chain_function (sink, GST_DEBUG_FUNCPTR (gst_my_filter_chain));
  GST_PAD_SET_PROXY_CAPS (sink);
  gst_element_add_pad (GST_ELEMENT (filter), sink);
  filter->sinkpad = sink;
  GST_PAD_SET_PROXY_CAPS (src);
  gst_element_add_pad (GST_ELEMENT (filter), src);
  filter->srcpad = src;
}

static gboolean my_filter_plugin_init (GstPlugin *plugin)
{
  return gst_element_register (plugin, "myfilter", GST_RANK_NONE, GST_TYPE_MY_FILTER);
}

#define PACKAGE "myfilter"
GST_PLUGIN_DEFINE (
  GST_VERSION_MAJOR, GST_VERSION_MINOR,
  myfilter, "Contains the myfilter element", my_filter_plugin_init,
  "1.0", "LGPL", "TeleMidia/PUC-Rio", "http://www.telemidia.puc-rio.br"
)
```

Listagem 8.12. Código do elemento "myfilter" e *plugin* correspondente.

A macro `GST_DECLARE_FINAL_TYPE` (linha 10) recebe cinco argumentos—o nome do novo tipo (`GstMyFilter`), o prefixo das funções do tipo (`gst_my_filter`), o prefixo das macros do tipo (`GST`), o sufixo das macros do tipo (`MY_FILTER`) e o tipo pai `GstElement`—e usa esses argumentos para emitir as declarações necessárias. Já a macro `GST_DEFINE_TYPE` (linha 11) recebe três argumentos—o nome do novo tipo, o prefixo das funções do tipo e o código (`GType`) do tipo pai—e define a função `gst_my_filter_get_type` que retorna o código `GType` do novo tipo. Ambas as macros fazem parte do *framework* de objetos da GLib.

Continuando, as instruções das linhas 13–18 declaram e inicializam duas variáveis estáticas, `src_template` e `sink_template`, com os *templates* para as *pads* do filtro—no caso, ambas as *pads* terão disponibilidade *always* e *caps* "any" (aceitarão qualquer formato de *buffers*).

A função `gst_my_filter_chain` (linhas 21–25) é a função que processa os *buffers* recebidos pela *sink pad* do filtro e os repassa à *source pad*. A função recebe três argumentos: um ponteiro para a *sink pad*, um ponteiro para o objeto filtro e um ponteiro para o *buffer*; e retorna um código de *status* correspondente. Nesse exemplo, a função apenas imprime o tamanho do *buffer* na saída padrão (linha 24) e repassa-o à *source pad* do elemento via a chamada `gst_pad_push`.

A função `gst_my_filter_class_init` (linhas 28–38) é utilizada pela GLib para inicializar o tipo (classe) `GstMyFilter`. Ele é chamada uma única vez (antes de o primeiro objeto do tipo ser instanciado) e, no caso, registra os metadados do novo elemento "myfilter" (linhas 31–33) e adiciona à classe do elemento os *templates* das suas futuras *pads* (linhas 34–37).

A função `gst_my_filter_init` (linhas 40–51) é utilizada pela GLib para inicializar cada instância (objeto) do tipo `GstMyFilter`. A função, nesse caso, cria as *pads* do novo filtro (linhas 42–43), inicializa-as (linhas 45 e 48), adiciona-as ao elemento (linhas 46–47 e 49–50), e registra a função de encadeamento na *sink pad* (linha 44). A função de encadeamento (`gst_my_filter_chain`, linhas 21–25) é chamada sempre que um *buffer* é recebido pela *sink pad* do elemento.

O restante do código (linhas 53–63) contém o ponto de entrada do *plugin* (função `my_filter_plugin_init`, linhas 53–56) e as declarações associadas. O ponto de entrada, função `my_filter_plugin_init`, é executada assim que o *plugin* (biblioteca dinâmica) "myfilter" é processado pelo GStreamer (o que ocorre durante a chamada `gst_init`) e, nesse caso, simplesmente adiciona os dados do novo *plugin* ao registro do GStreamer. Já a macro `GST_PLUGIN_DEFINE` (linhas 58–63) emite as declarações do *plugin* a partir dos argumentos fornecidos, a saber, versão mínima do GStreamer requirida, nome simbólico do *plugin*, descrição, ponto de entrada, versão, licença, pacote e origem).

Compilando e usando o elemento "myfilter"

Um *plugin* GStreamer é uma biblioteca dinâmica carregada em tempo de execução pelo *framework*. O código da Listagem 8.12 define um *plugin* chamado "myfilter" contendo apenas um elemento, também chamado "myfilter". Para gerar uma biblioteca dinâmica a partir desse código é necessário informar ao compilador os diretórios contendo os

cabeçalhos utilizados (diretivas *include*) e ao *linker* os nomes e diretórios das bibliotecas utilizadas. Além disso, é preciso instruir o *linker* a produzir uma biblioteca dinâmica, ao invés de um executável. No GNU/Linux, isso pode ser feito através do comando:

```
$ cc myfilter.c -o myfilter.so -shared -fPIC\
     `pkg-config --cflags --libs gstreamer-1.0 gstreamer-base-1.0`
```

Esse comando compila e *link*-edita uma biblioteca dinâmica (arquivo "myfilter.so") contendo o *plugin* da Listagem 8.12.

Finalmente, para construir um *pipeline* usando o *plugin* "myfilter" basta instalá-lo num dos diretórios de *plugins* do GStreamer ou indicar ao GStreamer que esse *plugin* deve ser carregado. Por exemplo, assumindo que o arquivo "myfilter.so" gerado pelo comando anterior está no diretório atual, o seguinte comando monta um *pipeline* contendo o elemento "myfilter":

```
$ gst-launch --gst-plugin-path=.\
     videotestsrc ! myfilter ! ximagesink
```

O valor "." para a opção "--gst-plugin-path" informa ao *gst-launch* que, além dos diretórios padrão, *plugins* também devem ser buscados no diretório corrente ("."). Por conta disso, assim que é iniciado o *gst-launch* carrega o *plugin* "myfilter" que adiciona elemento "myfilter" ao registro do GStreamer, tornando-o "visível" à aplicação. Como o elemento "myfilter" consome e produz dados de qualquer tipo (isto é, as *caps* de ambas as suas *pads* são "any") podemos utilizá-lo entre quaisquer elementos. O comando anterior utiliza-o entre os elementos "videotestsrc", que produz *buffers* de teste de vídeo, e "autovideosink", que mostra esses *buffers* na tela. Se tudo der certo, quando o comando é executado os *buffers* de teste são exibidos na tela e o tamanho de cada *buffer* é impresso pelo "myfilter" na saída padrão.

Elemento "myvideofilter"

O próximo exemplo, elemento "myvideofilter", é um filtro de vídeo. A diferença desse elemento em relação ao anterior ("myfilter") é que aqui estamos interessados em filtrar apenas quadros de vídeo. Dessa forma, ao invés de herdar diretamente do tipo `GstElement`, vamos herdar do tipo especializado `GstVideoFilter`, que já trata as tarefas comuns envolvidas no processamento de quadros de vídeo. Além desse tipo, o GStreamer possui tipos especializados para construção de *sources* (`GstBaseSrc`), *sinks* (`GstBaseSink`), filtros em geral (`GstBaseTransform`), filtros de áudio (`GstAudioFilter`), etc.

O elemento "myvideofilter" é um filtro de vídeo que aplica um efeito de ondulação aos amostras que o atravessam. Mais precisamente, o elemento transforma cada pixel (u,v) do quadro recebido num pixel (x,y) no quadro resultante, de acordo com a fórmula:

$$x(u,v) = u + 20 \times \sin(\mathit{factor} \times v)$$
$$y(u,v) = v\,,$$

em que *factor* é o valor da propriedade "factor" (número real) do elemento.

A Listagem 8.12 apresenta o código do elemento "myvideofilter" e do *plugin* correspondente. Na listagem, a estrutura `GstMyVideoFilter` (linhas 8–11) contém os

dados de uma instância de um elemento “myvideofilter”, a saber, os dados do tipo pai `GstVideoFilter` e o valor corrente da propriedade “factor” do elemento. As diretivas das linhas 15–18 declaram e definem o novo tipo (`GstMyVideoFilter`), que agora herda de `GstVideoFilter`, e as instruções seguintes (linhas 20–26) declaram e inicializam variáveis contendo os *templates* das *pads* do elemento—ambas as *pads* esperam *buffers* de vídeo nos formatos RGBx ou RGB.

```
1  #include <math.h>
2  #include <string.h>
3  #include <glib.h>
4  #include <gst/gst.h>
5  #include <gst/video/video.h>
6  #include <gst/video/gstvideofilter.h>
7
8  typedef struct _GstMyVideoFilter {
9    GstVideoFilter parent_instance;
10   gdouble factor;
11 } GstMyVideoFilter;
12
13 enum { PROP_0, PROP_FACTOR };
14
15 #define GST_TYPE_MY_VIDEO_FILTER (gst_my_video_filter_get_type ())
16 G_DECLARE_FINAL_TYPE (GstMyVideoFilter, gst_my_video_filter,
17                       GST, MY_VIDEO_FILTER, GstVideoFilter)
18 G_DEFINE_TYPE (GstMyVideoFilter, gst_my_video_filter, GST_TYPE_VIDEO_FILTER)
19
20 static GstStaticPadTemplate sink_template =
21   GST_STATIC_PAD_TEMPLATE ("sink", GST_PAD_SINK, GST_PAD_ALWAYS,
22                            GST_STATIC_CAPS (GST_VIDEO_CAPS_MAKE ("{BGRx, RGB}")));
23
24 static GstStaticPadTemplate src_template =
25   GST_STATIC_PAD_TEMPLATE ("src", GST_PAD_SRC, GST_PAD_ALWAYS,
26                            GST_STATIC_CAPS (GST_VIDEO_CAPS_MAKE ("{BGRx, RGB}")));
27
28 static void gst_my_video_filter_get_property (GObject *obj, guint id,
29                                               GValue *val, GParamSpec *pspec)
30 {
31   GstMyVideoFilter *filter = GST_MY_VIDEO_FILTER (obj);
32   switch (id)
33     {
34     case PROP_FACTOR:
35       g_value_set_double (val, filter->factor);
36       break;
37     default:
38       G_OBJECT_WARN_INVALID_PROPERTY_ID (obj, id, pspec);
39     }
40 }
41
42 static void gst_my_video_filter_set_property (GObject *obj, guint id,
43                                               const GValue *val, GParamSpec *pspec)
44 {
45   GstMyVideoFilter *filter = GST_MY_VIDEO_FILTER (obj);
46   switch (id)
47     {
48     case PROP_FACTOR:
49       filter->factor = g_value_get_double (val);
50       break;
51     default:
52       G_OBJECT_WARN_INVALID_PROPERTY_ID (obj, id, pspec);
53     }
54 }
55
56 static GstFlowReturn gst_my_video_transform (GstVideoFilter *vf, GstVideoFrame *in,
57                                              GstVideoFrame *out)
58 {
59   GstMyVideoFilter *filter = GST_MY_VIDEO_FILTER (vf);
60   gint i, j, w, h;
```

```
61    gint stride, pixel_stride;
62    guint8 *data, *indata;
63    guint outoffset, inoffset, u;
64
65    indata = GST_VIDEO_FRAME_PLANE_DATA (in, 0);
66    data = GST_VIDEO_FRAME_PLANE_DATA (out, 0);
67    stride = GST_VIDEO_FRAME_PLANE_STRIDE (out, 0);
68    w = GST_VIDEO_FRAME_COMP_WIDTH (out, 0);
69    h = GST_VIDEO_FRAME_COMP_HEIGHT (out, 0);
70    pixel_stride = GST_VIDEO_FRAME_COMP_PSTRIDE (out, 0);
71    for (i = 0; i < h; i++)
72      {
73        for(j = 0; j < w; j++)
74          {
75            outoffset = (i * stride) + (j * pixel_stride);
76            u = i + 20 * sin (filter->factor * j);
77            inoffset = (u * stride) + (j * pixel_stride);
78            if (u < h)
79              memcpy (data + outoffset, indata + inoffset, pixel_stride);
80          }
81      }
82    return GST_FLOW_OK;
83  }
84
85  static void gst_my_video_filter_class_init (GstMyVideoFilterClass *klass)
86  {
87    GObjectClass *gobject_class = G_OBJECT_CLASS (klass);
88    GstElementClass *gstelement_class = GST_ELEMENT_CLASS (klass);
89    GstVideoFilterClass *gstvideofilter_class = GST_VIDEO_FILTER_CLASS (klass);
90
91    gobject_class->set_property = gst_my_video_filter_set_property;
92    gobject_class->get_property = gst_my_video_filter_get_property;
93    g_object_class_install_property
94      (gobject_class, PROP_FACTOR, g_param_spec_double
95       ("factor", "factor", "distortion factor",
96        0.0, 320.0, 2*G_PI/130, G_PARAM_READWRITE));
97
98    gst_element_class_set_static_metadata
99      (gstelement_class, "An example video filter", "Example/myvideofilter",
100      "Example filter", "roberto@telemidia.puc-rio.br");
101   gst_element_class_add_pad_template
102     (gstelement_class, gst_static_pad_template_get (&src_template));
103   gst_element_class_add_pad_template
104     (gstelement_class, gst_static_pad_template_get (&sink_template));
105
106   gstvideofilter_class->transform_frame = GST_DEBUG_FUNCPTR (gst_my_video_transform);
107 }
108
109 static void gst_my_video_filter_init (GstMyVideoFilter *filter)
110 {
111   filter->factor = 2*G_PI/130;
112 }
113
114 static gboolean my_video_filter_plugin_init (GstPlugin *plugin)
115 {
116   return gst_element_register (plugin, "myvideofilter",
117                                GST_RANK_NONE, GST_TYPE_MY_VIDEO_FILTER);
118 }
119
120 #define PACKAGE "myvideofilter"
121 GST_PLUGIN_DEFINE (
122   GST_VERSION_MAJOR, GST_VERSION_MINOR,
123   myvideofilter, "Contains the myvideofilter element", my_video_filter_plugin_init,
124   "1.0", "LGPL", "TeleMidia/PUC-Rio", "http://www.telemidia.puc-rio.br"
125 )
```

Listagem 8.13. Código do elemento “myvideofilter” e *plugin* correspondente.

Na Listagem 8.13, a declaração *enum* da linha 13 cria uma constante simbólica para representar a propriedade "factor" (`PROP_FACTOR`). A lógica de obtenção e atribuição de valores de propriedades está contida nas funções `gst_my_video_filter_get_property` (linhas 28–40) e `gst_my_video_filter_set_property` (linhas 42–54). Nesse caso, o elemento reconhece apenas uma propriedade, "factor", cujo valor é armazenado ou obtido a partir do campo correspondente da estrutura `GstMyVideoFilter`.

A função `gst_my_video_filter_class_init` (linhas 85–107) inicializa o tipo (classe) `GstMyVideoFilter`. Aqui, além de registrar os metadados do elemento (linhas 98–100) e os *templates* das futuras *caps*, a função sobrescreve as funções para manipulação de propriedades (herdadas de `GObject`, linhas 91–92), "instala" na classe os metadados da propriedade "factor" (linhas 93–96) e sobrescreve a função que processa os *buffers* de vídeo recebidos (herdada de `GstVideoFilter`, linha 106). Já a função `gst_my_video_filter_init` (linhas 109–112), nesse caso, apenas inicializa a propriedade "factor" com o seu valor *default*.

A função `gst_my_video_transform` (linhas 56–83) é a função principal do exemplo. Ela manipula os *pixels* do quadro de entrada transformando-os de acordo com a fórmula anterior, e produz um efeito de ondulação no quadro de saída. Observe que diferentemente da função `gst_my_filter_chain` da Listagem 8.12 que recebia um *buffer* (`GstBuffer`), a função `gst_my_video_transform` recebe e devolve (via parâmetros) quadros de vídeo (`GstVideoFrame`).

Finalmente, as linhas 114–125 são análogas às linhas correspondentes da Listagem 8.12. Ou seja, contém o ponto de entrada do *plugin* (linhas 114–118) e as declarações associadas (linhas 120–125).

Compilando e usando o elemento "myvideofilter"

O comando seguinte compila e *link*-edita o *plugin* "myvideofilter" no GNU/Linux:

```
$ cc myvideofilter.c -o myvideofilter.so -shared -fPIC\
     `pkg-config --cflags --libs\
         gstreamer-1.0 gstreamer-base-1.0 gstreamer-video-1.0`
```

Observe que além dos módulos "gstreamer-1.0" e "gstreamer-base-1.0" o *plugin* "myvideofilter" depende do módulo "gstreamer-video-1.0", que contém cabeçalhos e bibliotecas para manipulação de *buffers* de vídeo.

O comando seguinte construí um *pipeline* que aplica o novo elemento ao vídeo *Big Buck Bunny*. O resultado do comando com diferentes valores para a propriedade "factor" é apresentado na Figura 8.10.

```
$ gst-launch --gst-plugin-path=.\
               uridecodebin uri="file://$PWD/bunny.ogg"\
               ! videoconvert\
               ! myvideofilter factor=.10\
               ! videoconvert\
               ! autovideosink
```

(a) *factor* = 0.01 (b) *factor* = 0.05 (c) *factor* = 0.1

Figura 8.10. Efeito de ondulação produzido pelo filtro "myvideofilter".

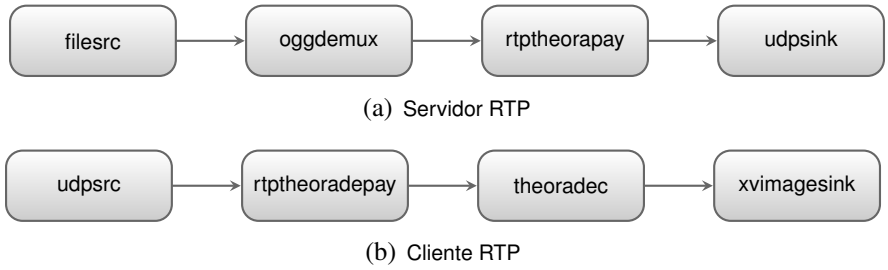
8.7. Conclusão

Neste minicurso apresentamos uma introdução ao *framework* multimídia GStreamer. Discutimos o modelo de computação *dataflow* e, em particular, a forma como esse modelo é instanciado no *framework*. Além disso, através de exemplos incrementais, apresentamos os principais conceitos da API C básica do GStreamer. Mais especificamente, mostramos como aplicações simples para reprodução de conteúdo multimídia podem ser programadas, e discutimos brevemente a criação de novos *plugins* contendo elementos processadores.

Apesar dos exemplos discutidos serem relativamente simples, com o que vimos já é possível que criar aplicações multimídia avançadas. Por exemplo, usando as mesmas APIs podemos usar elementos *mixers* (elementos "audiomixer" e "compositor") para compor diversos fluxos de áudio e vídeo num mesmo *pipeline*. Outra possibilidade é usar o *framework* para transcodificar fluxos de mídia, ou seja, montar *pipelines* que decodificam e re-codificam (em outro formato) um ou mais fluxos de mídia. Nesse caso, o *sink* "filesink" pode ser usado para escrever o fluxo resultante no disco.

Alguns tópicos avançados, porém, foram deliberadamente omitidos. No restante desta seção discutimos brevemente alguns desses tópicos com o objetivo de direcionar os leitores a aprofundamentos futuros.

Um dos temas desafiadores em sistemas multimídia é a transmissão e recepção de dados multimídia em tempo real. Embora não tratados aqui, o GStreamer possui diversos elementos que permitem transmitir e receber fluxos da rede. Por exemplo, o pacote *gst-plugins-good* contém elementos que podem ser usados para criar servidores e clientes RTP, RTCP e RTSP. No Gstreamer, um servidor RTP pode ser implementado como um *pipeline* que fragmenta fluxos codificados, encapsula esses fragmentos em pacotes RTP e transmite-os na rede via UDP, conforme ilustrado no *pipeline* da Figura 8.11a. Na figura, o elemento "rtptheorapay" recebe um fluxo de vídeo Theora e o encapsula em pacotes RTP que são transmitidos na rede via UDP pelo elemento seguinte, "udpsink". De forma análoga, podemos implementar um cliente RTP usando os elementos *udpsrc*, que recebe um fluxo de dados via UDP, e *rtptheorapay*, que desempacota um fluxo RTP e gera um fluxo codificado resultante, como ilustrado no *pipeline* da Figura 8.11b. Além de elementos que encapsulam e desencapsulam formatos específicos de fluxos em pacotes RTP, o GStreamer possui diversos outros elementos que facilitam a implementação desse tipo de aplicação, por exemplo, *rtpmanager*, *rtpbin*, *rtpjitterbuffer*, etc.



(a) Servidor RTP

(b) Cliente RTP

Figura 8.11. Servidor e cliente RTP para vídeos Ogg/Theora.

Outro tema importante que não discutimos é QoS (qualidade de serviço). De fato, mencionamos rapidamente eventos de QoS na Seção 8.1. Além desses eventos, o GStreamer possui outras funcionalidades que visam manter a QoS de aplicações multimídia. Por exemplo, internamente todo *pipeline* mantém um relógio (`GstClock`) que é usado para sincronizar a exibição dos *buffers* que o percorrem. Elementos *sink* usam esse relógio tanto para controlar a apresentação das amostras quanto para descartar amostras atrasadas. Dependendo da aplicação, se necessário, podemos sincronizar o relógio de *pipelines* diferentes (mesmo em máquinas separadas) via objetos `GstNetTimeProvider`, que expõe o tempo de um `GstClock` para a rede, ou `GstNetClientClock`, que sincroniza o relógio de um *pipeline* a um objeto provedor de tempo (`GstNetTimeProvider`), e podemos ainda criar relógios sincronizados com um servidor NTP (`GstNtpClock`) ou PTP (`GstPtpClock`). Dessa forma, é possível usar o GStreamer para implementar aplicações que requerem a sincronização de mídias em múltiplos destinos ou em múltiplas origens—cenários comuns em ambientes híbridos de TV digital [van Deventer et al., 2016].

Há certamente mais temas importantes que não abordamos. Todavia, acreditamos que a partir da base apresentada aqui os leitores estejam aptos a atacar esses temas. No site oficial do GStreamer [GStreamer, 2016] há diversos materiais detalhando o funcionamento interno do *framework*. Além disso, apesar de não ser essencial, àqueles que querem trabalhar diretamente no código do GStreamer, ou mesmo estendê-lo por meio de *plugins*, recomendamos a documentação do *framework* de objetos GObject [GLib, 2016a]. Finalmente, caso C não seja a sua linguagem favorita, existem *bindings* do GStreamer para outras linguagens e ambientes, por exemplo, Lua, Python, Java, C++, Qt, Android, Vala, Ruby, Haskell, etc.

Referências

[ALSA, 2016] ALSA (2016). Advanced Linux Sound Architecture (ALSA) Project Homepage. `http://www.alsa-project.org`. Acessado em 4 de outubro de 2016.

[Amatriain et al., 2008] Amatriain, X., Arumi, P., e Garcia, D. (2008). A framework for efficient and rapid development of cross-platform audio applications. *Multimedia Systems*, 14(1):15–32.

[Blender, 2008] Blender (2008). Big Buck Bunny. `http://peach.blender.org`. Acessado em 4 de outubro de 2016.

[Chatterjee e Maltz, 1997] Chatterjee, A. e Maltz, A. (1997). Microsoft DirectShow: A new media architecture. *SMPTE Motion Imaging Journal*, 106(12):865–871.

[GLib, 2016a] GLib (2016a). GLib reference manual. `https://developer.gnome.org/GLib`. Acessado em 4 de outubro de 2016.

[GLib, 2016b] GLib (2016b). GObject reference manual. `https://developer.gnome.org/GObject`. Acessado em 4 de outubro de 2016.

[GNOME, 2016] GNOME (2016). GNOME. `https://www.gnome.org`. Acessado em 4 de outubro de 2016.

[Gough, 2005] Gough, B. J. (2005). *An Introduction to GCC*. Network Theory Ltd, United Kingdom, rev. edition.

[GStreamer, 2016] GStreamer (2016). GStreamer: Open source multimedia framework. `http://gstreamer.freedesktop.org`. Acessado em 4 de outubro de 2016.

[GStreamer Developers, 2016] GStreamer Developers (2016). GStreamer apps. `https://gstreamer.freedesktop.org/apps/`. Acessado em 4 de outubro de 2016.

[ISO/IEC, 1993] ISO/IEC (1993). *ISO/IEC 11172-3:1993: Information Technology — Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1,5Mbit/s — Part 3: Audio*. International Organization for Standardization, Geneva, Switzerland.

[Kahn e MacQueen, 1977] Kahn, G. e MacQueen, D. B. (1977). Coroutines and networks of parallel processes. In Gilchrist, B., editor, *Proceedings of IFIP Congress 77, Toronto, Canada, 8–12 August, 1977*, pages 993–998, Amsterdam, The Netherlands. North-Holland Publishing Company.

[Kernighan e Ritchie, 1988] Kernighan, B. W. e Ritchie, D. M. (1988). *The C Programming Language*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition.

[Lee e Parks, 1995] Lee, E. A. e Parks, T. M. (1995). Dataflow process networks. *Proceedings of the IEEE*, 83(5):773–801.

[Mecklenburg, 2005] Mecklenburg, R. (2005). *Managing Projects with GNU Make*. O'Reilly, Sebastopol, CA, USA, 3rd edition.

[Orlarey et al., 2009] Orlarey, Y., Fober, D., e Letz, S. (2009). FAUST: An efficient functional approach to DSP programming. In Assayag, G. e Gerzso, A., editors, *New Computational Paradigms for Computer Music*. Éditions Delatour Paris, Sampzon, France.

[Pfeiffer, 2003] Pfeiffer, S. (2003). The Ogg encapsulation format version 0. RFC 3533, RFC Editor.

[Puckette, 2007] Puckette, M. S. (2007). *The Theory and Technique of Electronic Music*. World Scientific Publishing Company, Singapore.

[Underbit, 2016] Underbit (2016). MAD: MPEG audio decoder. `http://www.underbit.com/products/mad/`. Acessado em 4 de outubro de 2016.

[van Deventer et al., 2016] van Deventer, M. O., Stokking, H., Hammond, M., Feuvre, J. L., e Cesar, P. (2016). Standards for multi-stream and multi-device media synchronization. *IEEE Communications Magazine*, 54(3):16–21.

[Wang e Cook, 2003] Wang, G. e Cook, P. R. (2003). ChucK: A concurrent, on-the-fly, audio programming language. In *Proceedings of the 2003 International Computer Music Conference, Singapore, 29 September–4 October, 2003*, pages 219–226, San Francisco, CA, USA. International Computer Music Association.

[Xiph.Org, 2011] Xiph.Org (2011). Theora specification. Xiph.org specification.

[Xiph.Org, 2015] Xiph.Org (2015). Vorbs I specification. Xiph.org specification.

[X.Org, 2016] X.Org (2016). X.Org. `http://www.x.org`. Acessado em 4 de outubro de 2016.

[Yviquel et al., 2015] Yviquel, H., Sanchez, A., Jääskeläinen, P., Takala, J., Raulet, M., e Casseau, E. (2015). Embedded multi-core systems dedicated to dynamic dataflow programs. *Journal of Signal Processing Systems*, 80(1):121–136.

Biografia Resumida dos Autores

Guilherme F. Lima é pesquisador associado do Laboratório TeleMídia da PUC-Rio. Seus interesses de pesquisa incluem linguagens de programação e modelos para sincronismo multimídia. Obteve o Doutorado em Informática pela PUC-Rio em 2015. Também possui Mestrado em Informática (2011) e Bacharelado em Sistemas de Informação (2009), ambos pela PUC-Rio.

Rodrigo C. M. Santos é doutorando em Informática na PUC-Rio. Mestre em Ciência da Computação pela Universidade Federal do Maranhão (2013). Atualmente é pesquisador do laboratório TeleMídia da PUC-Rio e colaborador do laboratório LAWS/UFMA, tendo como principais interesse de pesquisa: sistemas multimídia distribuídos, sincronização de mídia em diferentes dispositivos e linguagens reativas síncronas.

Roberto G. de A. Azevedo é pesquisador associado do Laboratório TeleMídia da PUC-Rio. Possui doutorado (2015) e mestrado (2010) em Informática pela PUC-Rio e é Bacharel em Ciência da Computação pela Universidade Federal do Maranhão (2008). Seus interesses de pesquisa incluem: representação e autoria de cenas multimídia interativas; e representação, codificação, transmissão e renderização de vídeos 3D.

Índice de Autores