UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

SYNCSMARTV: FRAMEWORK PARA SINCRONIZAÇÃO DE APLICAÇÕES SMART TV COM PROGRAMAS TELEVISIVOS

CEDRICK BAMBA NSIMBA

ORIENTADOR: PROF. DR. CESAR AUGUSTO CAMILLO TEIXEIRA

São Carlos - SP
Fevereiro/2015

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

SYNCSMARTV: FRAMEWORK PARA SINCRONIZAÇÃO DE APLICAÇÕES SMART TV COM PROGRAMAS TELEVISIVOS

CEDRICK BAMBA NSIMBA

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação, área de concentração: Sistemas Distribuídos e Redes
Orientador: Dr. Cesar Augusto Camillo Teixeira

São Carlos - SP
Fevereiro/2015

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

N961sf

Nsimba, Cedrick Bamba.
SyncSmartv : framework para sincronização de aplicações *smart* TV com programas televisivos / Cedrick Bamba Nsimba. -- São Carlos : UFSCar, 2015.
125 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2015.

1. Sistemas distribuídos. 2. *Smart* TV. 3. Televisão digital interativa. 4. Sincronização. 5. Software de aplicação. I. Título.

CDD: 005.43 (20ª)

UNIVERSIDADE FEDERAL DE SÀO CARLOS

Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

Folha de Aprovaçàо

Assinaturas dos membros da comissão examinadora que avaliou e aprovou a Defesa de Dissertação de Mestrado do candidato Cédrick Bamba Nsimba, realizada em 09/02/2015:

Prof. Dr. Cesar Augusto Camillo Teixeira
UFSCar

Prof. Dr. Luis Carlos Trevelin
UFSCar

Prof. Dr. Celso Alberto Saibel Santos
UFES

AGRADECIMENTOS

Aos meus pais, irmãos, e a toda minha família que, com muito carinho e apoio, não mediram esforços para que eu chegasse até esta etapa de minha vida.

Ao professor e orientador Cesar Teixeira pela paciência na orientação e incentivo que tornaram possível a conclusão desta dissertação.

Aos amigos e colegas do LINCE, pelo incentivo e pelo apoio constantes.

À minha avó Ndona Teresa *in memoriam* pelo provimento de recursos para o meu deslocamento ao Brasil para realizar meus estudos.

"La connaissance s'acquiert par l'expérience, tout le reste n'est que de l'information"

Albert Einstein

RESUMO

Aplicações Smart TV normalmente são desvinculadas do conteúdo da programação sintonizada no aparelho de TV. Algumas emissoras oferecem aplicativos específicos para programas mas esses aplicativos são, em geral, apenas levemente acoplados ou sincronizados ao programa. Além disso, os aplicativos são de total domínio das emissoras, que detêm o conhecimento sobre o conteúdo transmitido. A partir do princípio de que informações de sincronismo possam advir de serviços de terceiros, independentes de emissoras, obtidos por processamento local de conteúdo ou oferecidas diretamente pelo telespectador, oportunidades se abrem para o desenvolvimento de aplicativos bastante interessantes, com potencial de promover maior interatividade na TV, novas possibilidades de interação quando se considera o uso de dispositivos móveis e computadores, satisfação aos telespectadores e novos modelos de negócio. Neste trabalho de mestrado apresenta-se e avalia-se o *framework* SyncSmartv. O *framework* oferece diversas APIs que favorecem o desenvolvimento de aplicações para Smart TV sincronizadas com o programa sintonizado. Pretende-se fornecer ao desenvolvedor facilidades para construção de aplicações nesse domínio, de forma transparente, sem a necessidade de se preocupar com detalhes de implementação de nível baixo.

Palavras-chave: televisão conectada, interatividade, Smart TV, integração de aplicações, programas televisivos, monitoramento de canais da TV.

ABSTRACT

Smart TV applications are typically disconnected from the content of the tuned programming on the TV set. Some broadcasters offer specific applications for programs, however, these applications are generally just loosely coupled or synchronized to the program. Furthermore, applications are fully under domain of broadcasters, that have the information about the transmitted content. Based on the fact that synchronized information may arise from third-parties, obtained by local content processing or offered directly by the viewers, new opportunities will be opened up for developing a bunch of new interesting applications aiming to promote the user interaction level in TV. Moreover, it will have new user interaction possibilities while using mobile devices and computers, furthermore a new business model will be emerged. In this master thesis, the SyncSmartv framework is presented and evaluated. The aforementioned framework offers several APIs that facilitate the development of Smart TV applications synchronized with TV program contents. The aim is to provide some facilities for developers to implement applications in this area in a clear approach without being concerned about low-level implementation details.

Keywords: connected television, interactivity, Smart TV, application integration, television programs, TV channels monitoring.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API - Application Programming Interface

CHP – Connected Home Platform

DNS-SD – Domain Name System – Service Discovery

EPG - Electronic Programming Guide

GUI – Grafic User Interface

HD - Hight Definition

HDMI - Hight Definition Multimedia Interface

ID - IDentification

IP - Internet Protocol

LPS – Linha de Produtos de Software

MPEG - Moving Picture Experts Group

MPEG-TS – MPEG -Transport Stream

MVC – Model View Control

NTSC - National Television System Committee

PAL - Phase Alternation Line

PC - Personal Computer

PLC - Power Line Comunication

QoS - Quality of Service

SDK - Software Development Kit

SECAM - Séquentielle Couleur Mémoire

SSDP – Simple Service Discovery Protocol

TCP – Transmission Control Protocol

TV - Televisão

TVDi - Televisão Digital Interativa

UML – Unified Modeling Language

UPnP – Universal Plug and Play

Zeroconf – Zero configuration

SUMÁRIO

Capítulo 1

INTRODUÇÃO

A televisão, tanto como aparelho físico quanto como meio de recepção e modelo de negócios está em processo de fortes mudanças devido à evolução de tecnologias. Os grandes momentos na linha evolutiva da televisão foram o advento da televisão colorida em 1954 e o surgimento dos transistores nos anos 60. Após essa época, a televisão passou por poucas mudanças. Atualmente, entretanto, com o crescimento da internet, o advento das redes sociais, da televisão digital e o paradigma de aplicações de dispositivos móveis, abrem oportunidade para que se explorem novos modelos de televisão muito mais interativa e social.

As plataformas de TVs Conectadas, também denominadas *Smart TVs* ou TVs Híbridas, com capacidade de processamento de dados, de proporcionar interatividade por meio de aplicações, de conectividade de diversos dispositivos, podem não só estender certos aspectos da funcionalidade dos aparelhos de TV convencional, como também abrir novas e interessantes possibilidades de interação, principalmente quando se considera o uso de celulares, computadores e *tablets* para esse fim.

Seria natural que o valor agregado à TV Digital, resultando no que se chama Smart TV, estivesse sendo utilizado mais intensamente para maximizar a interatividade de telespectadores por meio de aplicações interessantes, criando assim uma TV de interação social remota/presencial em grande escala. Infelizmente, diversos fatores característicos da Smart TV, relacionados talvez a modelo de negócios,

no qual emissoras e anunciantes continuam a ser os principais donos do conteúdo veiculado por aplicações, têm impedido a exploração efetiva da interatividade na TV.

Neste trabalho, entende-se por aplicações interessantes para usuários da TV aquelas que, além de oferecer acesso a uma ampla variedade de serviços de vídeos online, são capazes de saber o que está sendo assistido na TV no momento. Com essas informações, pode-se oferecer funcionalidades avançadas incluindo interfaces de usuário atraentes para interação social remota promovendo uma TV cada vez mais social por exemplo, *two-way communications* com dispositivos como *smartphones e tablets* permitindo a esses dispositivos enviarem vídeos para TV e saberem do estágio do decorrer da programação televisiva.

Infelizmente, tanto no Brasil como em outras partes do mundo, a interatividade não tem sido promovida em larga escala pelas emissoras de TV, no contexto da Smart TV. Um dos motivos é a não disponibilização das informações de sincronismo do conteúdo na TV. A partir do momento onde essas informações serão oriundas de outras fontes, a não ser emissoras de TV, novas oportunidades se abrirão para melhorar a experiência do telespectador, promover maior interatividade e criar novos interessantes modelos de negócio. Esse trabalho apresenta o *framework* SyncSmartv com propósito de facilitar a construção de aplicações interessantes para Smart TV.

As aplicações Smart TV disponíveis até o momento não estão integradas com os programas televisivos. A falta de mecanismos que facilitariam sua comunicação com o conteúdo audiovisual da TV, para saber dos eventos ocorrendo na programação televisiva, por exemplo, pode ser um fator que contribua com essa falta de integração. O emprego adequado de técnicas de sincronização pode resolver esse problema, e seria interessante que soluções encontradas pudessem ser disponibilizadas para facilitar o processo de construção de aplicações Smart TV sincronizadas com o programa da TV. Uma das formas de se promover essa disponibilização é através de reuso de software. Reuso de Software é uma prática na qual se utiliza artefatos de software existentes para construção de novos produtos de software. Os artefatos reutilizáveis vão desde trechos de código e bibliotecas até componentes de software, serviços Web, arquiteturas de projetos, padrões e frameworks. Além de diminuir os custos e o tempo

de desenvolvimento de software, uma vez bem aplicado, o reuso pode proporcionar grande ganho de qualidade e produtividade.

Alguns conceitos importantes relacionados ao histórico da televisão são introduzidos em 1.1. Na seção 1.2 são apresentados os elementos motivadores que levaram ao desenvolvimento deste projeto de mestrado. Na seção 1.3 são apresentados os objetivos deste trabalho. Por fim, a seção 1.4 apresenta a organização desta dissertação.

1.1 Contexto

Os sistemas de TV evoluíram, passando de analógico para digital. No sistema digital, o sinal é transmitido em representação binária. Com essa representação, o sinal é facilmente interpretado e recuperado no destino, mesmo que tenha sido exposto às interferências e ruídos do meio de transmissão. Além disso, o sistema digital permite que sejam adicionados programas computacionais no fluxo normal de transmissão, possibilitando assim a criação de programas interativos, em que o usuário pode utilizar o controle remoto ou outro dispositivo (celular, *tablet*, PC...) para interagir com o programa. Evolução natural da interatividade da TV associada ao modelo de Redes Sociais tem conduzido atualmente ao que se denomina TV Social (BACHMAYER, LUGMAYER e KOTSIS, 2009). A TV Social utiliza-se das possibilidades de interação para promover, por exemplo, comunicação/colaboração entre telespectadores assistindo um mesmo programa.

Tanto no sistema analógico como no digital, os meios de transmissão utilizados são radiofrequência, cabo e satélite. O modelo tradicional de entrega de conteúdo das emissoras é o mesmo nos dois sistemas. É amplamente conhecido por sua denominação na língua inglesa: *broadcasting*. O mesmo conteúdo é distribuído a todos os telespectadores que sintonizam um mesmo canal.

Em 1999 surgiram as Smart TVs (TVs Conectadas ou TVs Híbridas). O termo *Smart TV* é utilizado para descrever a integração da plataforma da televisão com a internet com o objetivo de estender o conjunto de funcionalidades básicas da televisão digital comum. Outras categorias de TV, além da Smart TV, são a IPTV e a WebTV. A IPTV é definida como a conectividade da plataforma da TV com a internet, usando o protocolo IP para o transporte de seus conteúdos numa infraestrutura dedicada e paralela à internet comum (rede pública). Segundo Jeong, Hong e Lee (2011) apesar de muitas pesquisas científicas conduzidas para facilitar o acesso e, compartilhamento de conteúdo no ambiente IPTV, os usuários não vêm tendo boa experiência devido à falta de conteúdo variado, à falta de aplicações não pagas *(free Apps)* e à falta de interfaces convenientes para facilitar o acesso ao conteúdo de mídia. A WebTV é uma outra forma de TV que oferece acesso ao conteúdo da TV pela Internet. Diferentemente da IPTV e da TV convencional em que a qualidade de serviço é garantida por usarem a rede de distribuição com reserva de recursos, na WebTV a qualidade de serviço não é garantida pois ela fica sujeita ao indeterminismo da Internet (rede pública).

Segundo Park e Kim (2011), Smart TV é geralmente definida como um meio que oferece transmissão de TV comum, com acesso à internet, aplicativos e serviços inteligentes por meio de uma plataforma operacional e um processador presente no próprio televisor ou em um *set-top box* externo.

Se formos comparar a Smart TV com a televisão convencional, diremos que ela é uma televisão convencional que recebe o sinal da programação pelos meios comuns (radiofrequência, cabo e satélite) e partilha a sua tela entre a apresentação do conteúdo televisivo e a interação com aplicativos accessíveis pela Internet. A Internet pode também ser usada pela Smart TV como mais um meio de recepção de conteúdo.

A tela grande de Smart TVs comparada a dispositivos móveis como *tablets* e celulares, atrai a atenção do usuário para desfrutar a visualização de conteúdos multimídia. Aplicações como de ginástica, em que o usuário precisa acompanhar detalhes dos exercícios, aplicações de coreografia de dança, aplicações de *streaming* de vídeo, jogos são alguns tipos de aplicações apropriadas para Smart TV.

1.2 Motivação

O advento da TV Digital e das TVs Conectadas vem inovando a forma como as pessoas assistem televisão e mudando o paradigma passivo de o usuário somente consumir conteúdo. No ambiente da TV Digital Interativa o usuário consegue ter interação em níveis diferentes com o conteúdo da TV. As emissoras enviam junto com seus conteúdos, aplicações interativas capazes de serem executadas no processador do próprio televisor.

Entretanto, diferentemente do cenário de interatividade na TVDI, o conteúdo audiovisual dos programas na TV Conectada, na maioria das vezes, não dialoga com as próprias aplicações encontradas na TV e demais recursos da plataforma. A única interatividade básica oferecida é a possibilidade de se alternar o que se exibe na tela: ou o programa da TV ou aplicações. Tal modelo de conectar a TV à *internet*, oferecendo um conteúdo audiovisual sem mecanismos pré-definidos de diálogo com demais recursos da mesma plataforma caracteriza a diferença entre Smart TV e TVDI.

Apesar da limitação das plataformas de TVs Conectadas citada acima, elas trazem novos conceitos importantes na TV, como: a convergência midiática; extensão de serviços com acesso à internet; a conectividade com diversos dispositivos, que, não só altera certos aspectos da funcionalidade dos aparelhos de TV, mas também abre novas possibilidades de interação ao se utilizarem celulares, computadores e *tablets* para recepção e o compartilhamento de conteúdo. Além disso, a adoção do conceito de lojas virtuais, proveniente do mundo dos *smartphones,* no qual fabricantes e terceiros disponibilizam aplicações para *download,* caracteriza também o modelo Smart TV.

Um estudo sobre o mercado de venda de Smart TVs, realizado pela *Strategy Analytics e BI Intelligence* (STRATEGY, 2014), avaliou a venda de Smart TVs em relação aos outros tipos de TVs/dispositivos de TVs (Google's Chromecast, Apple TV, etc.) encontrados no mercado e fez uma previsão para os próximos três anos. Segundo a pesquisa cujos resultados estão mostrados na figura 1, a venda de Smart TVs está crescendo para dominar o mercado geral de TVs tendo 33% de vendas globais de TVs em 2013 e fechando 2014 com a taxa de 44%. De acordo com a pesquisa, em 2017, Smart

TVs vão representar 73% de vendas totais de TVs no mundo inteiro passando por 54% e 63% em 2015 e 2016 respectivamente.

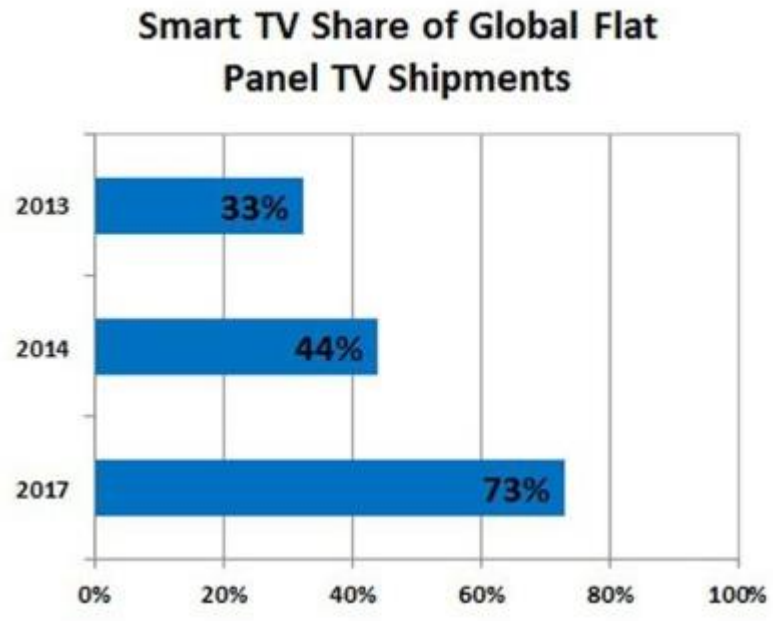


Figura 1: As taxas de vendas de Smart TVs, por ano

[Fonte: Strategy, 2014]

Normalmente isso sugeriria que o paradigma de Smart TVs estaria muito difundido. Entretanto, pesquisas vêm sendo realizadas para explicar por que o conceito Smart TV ainda não é tão difundido quanto o conceito *Smartphone*, por exemplo. Os resultados de alguns desses trabalhos são apresentados e discutidos nos próximos parágrafos.

Em sua pesquisa sobre consumidor de produtos eletrônicos conduzida em cinco países europeus e norte americano em maio de 2013, Bachelet (2013) descobriu que apenas menos da metade de 6115 pessoas entrevistadas, donas de Smart TV, conectaram suas TVs à internet. A Figura 2 exibe, por faixa etária, a comparação entre o número de pessoas proprietárias de Smart TV e número de pessoas com Smart TV conectada à internet.

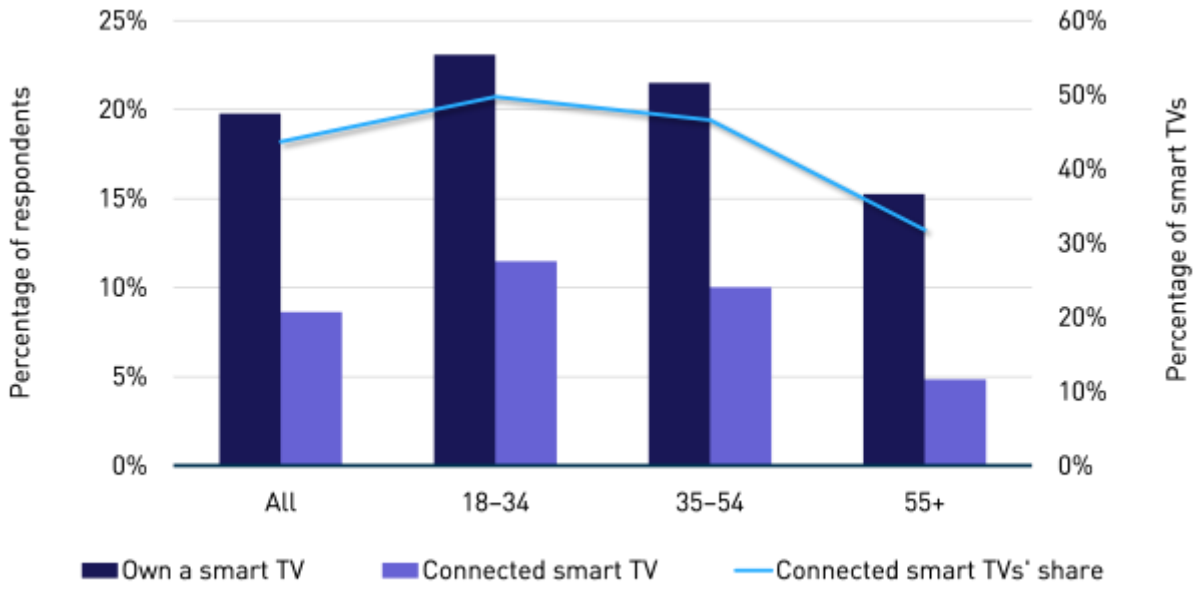


Figura 2: As taxas de propriedade e de conexão de Smart TVs, por faixa etária

[Fonte: Analysys Mason, 2013]

Segundo Bachelet (2013), dois elementos são responsáveis pelo fato da Smart TV não ter ainda se tornado popular e sido amplamente aceita pelos consumidores assim como são os *Smartphones*:

- Falta de conteúdo e aplicações atraentes: apesar da maioria de Smart TVs oferecerem uma ampla gama de conteúdos e aplicações, a maioria desses conteúdos e aplicações é irrelevante e de pouco interesse do usuário.
- Interfaces de Usuário pobres: Faltam interfaces ricas que podem integrar as aplicações da TV com o próprio conteúdo audiovisual dos programas da TV.

Schofield (2012) afirma que:

> If TVs are going to be truly smart they must do more than offer a wide variety of online video services. Instead they must add advanced functionality including voice control, motion control, advanced advertising, attractive user interfaces and two-way communications with other smart devices – so-called 'second screens' – allowing these devices both to send video to the TV and know what is being watched. Manufacturers should focus less on adding more content and more on improving how users can interact with that content.

Isso sugere que o potencial de interatividade das TVs Conectadas ainda não foi devidamente explorado. Novos mecanismos de sincronização e de interação com o ambiente da TV para enriquecer a experiência do usuário podem contribuir.

Além dos trabalhos citados acima (BACHELET, 2013) e (SCHOFIELD, 2012), há um trabalho no qual o autor fez estudos para determinar quais fatores são determinantes na decisão do consumidor em utilizar a nova geração de TVs inteligentes, Smart TVs (SUNGJOON, 2012). Como resultado da pesquisa, conclui-se que a percepção de utilidade e de facilidade de uso, em inlgês, "*Perceived Usefulness*" e "*Perceived ease of use*" respectivamente são dois fatores que influenciam positivamente na decisão do consumidor em começar a usar a Smart TV. Mais especificamente, "*Perceived Usefulness*" é definida como o grau em que uma pessoa entende que uma dada tecnologia vai trazer melhorias no desempenho do seu trabalho ou na sua vida cotidiana. E "*Perceived ease of use*" é definida como o grau em que uma pessoa acredita que o uso de uma dada tecnologia seria fácil.

A constatação de que a interatividade criativa e inovadora não vem sendo promovida e nem explorada de maneira adequada nas plataformas de TVs Conectadas é uma das principais razões para a sua baixa adoção. Outro motivo para a baixa adesão aos recursos interativos deve-se à falta de facilidades que permitam a integração de aplicações Smart TV com o próprio conteúdo audiovisual dos programas das emissoras. A constatação desses aspectos motivou a realização do trabalho de mestrado aqui proposto.

1.3 Objetivos

Levando em conta as considerações apresentadas na seção da motivação, esta pesquisa de mestrado tem como objetivo geral propor e implementar um *framework* que possa contribuir com o desenvolvimento de aplicações para Smart TV que sejam integradas e sincronizadas com o programa exibido na TV. Toma-se como premissa que aplicações para Smart TV integradas ao programa exibido possam promover maior interatividade e melhor qualidade da experiência em assistir e interagir com os programas.

Como objetivos específicos da pesquisa apresentam-se:

- Incorporar ao *framework* dois mecanismos de sincronização entre aplicações e programa de TV (sincronização do próprio usuário e sincronização por serviços de terceiros) apresentados adiante no capítulo sobre mecanismos de sincronização;
- Desenvolver um conjunto de aplicações *Mobile* e Smart TV que orientem a concepção do *framework*;
- Fazer um comparativo do desenvolvimento com e sem o uso do *framework* proposto.
- Avaliar a aceitação pelos usuários das aplicações desenvolvidas

1.4 Organização da Dissertação

No capítulo 2 discutem-se conceitos e técnicas de reuso de software. Em seguida, são apresentados aspectos clássicos da Smart TV (definição, arquitetura, aplicações, entre outros). No capítulo 3 são apresentados conceitos e mecanismos referentes à sincronização de aplicações da TV com a programação da TV. É apresentado também o modelo de notificação construído para a promoção do

sincronismo tratado no contexto deste trabalho de mestrado. No capítulo 4 apresenta-se a proposta do trabalho, destacando os problemas envolvidos, a metodologia adotada e o *framework* proposto. No capítulo 5 é apresentado o método experimental utilizado para a validação do trabalho. No capítulo 6 comparam-se trabalhos relacionados a este. Por fim, o capítulo 7 apresenta a conclusão geral sobre o trabalho, alinhando os objetivos iniciais com os resultados encontrados.

Capítulo 2

REFERENCIAL TEÓRICO E TECNOLÓGICO

Este capítulo, inicialmente, discute conceitos e técnicas de reuso de software. Em seguida, são apresentados conceitos, tecnologias e ferramentas envolvidos no desenvolvimento de aplicações para as plataformas Smart TV. O conteúdo apresentado neste capítulo visa auxiliar o entendimento da proposta do trabalho de mestrado e descrever o tipo de ferramental utilizado na realização de provas de conceitos para validação do trabalho.

2.1 Reuso de Software

O reuso é uma atividade inerente ao processo de solucionar problemas cotidianos dos seres humanos. Na medida em que encontramos soluções, que acreditamos que sejam adequadas para os nossos problemas, estas serão utilizadas na resolução de problemas similares. Nossa capacidade de abstração garante a adaptação necessária dessas soluções ao novo contexto de problemas. Essa mesma ideia do reuso existe na Engenharia de Software, entretanto amparada de uma sistemática ampla e formal para sua realização.

Reuso de Software é uma prática na qual se utiliza artefatos de software existentes para construção de novos produtos de software (PRESSMAN, 2006). Os artefatos reutilizáveis vão desde trechos de código e bibliotecas até componentes de software, serviços *Web*, arquiteturas de projetos, padrões e *frameworks*. Além de diminuir os custos e o tempo de desenvolvimento de software, uma vez bem aplicado, o reuso pode proporcionar um grande ganho de qualidade e produtividade.

Existem duas formas de explorar o conceito de reuso no desenvolvimento de um software, as quais são: "desenvolvimento para reuso" e "desenvolvimento com reuso". O desenvolvimento para reuso visa prover recursos para reutilização. Por exemplo, as API's (*Application Programming Interface*) e *Plugins* são componentes de software construídos e disponibilizados para serem reutilizados por diferentes aplicações. No desenvolvimento com reuso, os recursos disponíveis são reutilizados para criar novos produtos de software. Por exemplo, a utilização de API's para acesso a serviços remotos em uma aplicação. Existem diferentes técnicas de reuso de software, algumas delas são apresentadas detalhadamente na seção 2.2.

Vale salientar que no contexto deste trabalho será utilizada a técnica de "desenvolvimento para reuso".

2.1.1 Técnicas de reuso de software

O reuso de software pode ocorrer por meio de Componentes, *Frameworks* e Linha de Produtos de Software (LPS).

Componente de software é uma das formas mais importantes e utilizadas para reuso de produtos de software. Componentes são construídos quebrando grandes blocos em pequenas unidades interoperáveis, com o objetivo de serem reutilizados em diferentes aplicações. Em (CLEMENS, 1998) o autor define um componente de software como uma unidade contendo um conjunto de funcionalidades, accessível por meio de uma interface bem definida e que pode ser combinada com outros componentes. Por exemplo, componentes de software dos diferentes domínios, como componentes UML (*Unified Modeling Language*) podem ser usados para dar apoio à modelagem no processo de desenvolvimento de um software através de diagramas

padrões de casos de uso, de atividades, de classes, entre outros; e componentes GUI (*Graphical User Interface*) podem ser reutilizados para construção de uma interface gráfica de uma aplicação.

Outra forma de reuso bastante utilizada é através de *frameworks*. Segundo Gamma (1995) um Framework pode ser definido como:

> A set of cooperating classes that makes up a reusable design for a specific class of software. A framework provides architectural guidance by partitioning the design into abstract classes and defining their responsibilities and collaborations. A developer customizes the framework to a particular application by subclassing and composing instances of framework classes.

Isso significa que *framework* é uma arquitetura pré-definida pronta para ser usada, consistindo em blocos reusáveis semi-completos para a construção de novas aplicações customizadas. Pressman (2006) se refere ao *framework*, termo geralmente denominado *arcabouço* na língua portuguesa, como um esqueleto de código reutilizável que pode ser completado com as características específicas de uma aplicação. Um arcabouço não é um padrão arquitetural, mas um esqueleto com uma coleção de "pontos conectáveis" (também chamados de *ganchos* e *encaixes*) que lhe possibilitam adaptar-se a um domínio específico de problemas (APPLETON, 1998). Os pontos de conexão possibilitam a um projetista integrar classes ou funcionalidades específicas de um determinado domínio de problemas no esqueleto para construir novas aplicações.

Os conceitos de *Hot-Spots* e *Frozen-Spots* são fundamentais no desenvolvimento de um *framework*. *Hot-Spots* representam as partes do *framework* de aplicação que são específicas de sistemas individuais. Eles são projetados para serem genéricos e podem ser adaptados às necessidades de uma aplicação. De outro lado *Frozen-Spots* definem a arquitetura geral de um sistema de software, seus componentes básicos e os relacionamentos entre eles (PRESSMAN, 2006). Eles permanecem fixos ao instanciar o *framework* para a construção de uma nova aplicação.

Grails é um exemplo de *framework* bastante usado para criação de aplicações Web, e é baseado no modelo MVC (*Model View and Control*). MVC é um modelo que consiste em dividir a implementação de uma aplicação em três camadas: *Model*,

camada que provê acesso a banco de dados; *View* provê a interface gráfica do usuário e *Control* contém toda lógica do negócio da aplicação. *Hibernate* é um exemplo de *framework* também bastante usado e provê recursos para realizar o mapeamento de objetos Java e objetos relacionais. Através de relações de herança, associação existentes entre classes, ou através de arquivos XML ou através de anotações Java, *Hibernate* consegue gerar estruturas de tabelas de banco de dados relacionais que reflete as relações/anotações Java encontradas no modelo orientado a objeto da aplicação.

A linha de produtos de software ou família de produtos de software é outra forma de reuso bastante usada pelas indústrias e consiste em derivar novos produtos de software de um conjunto de produtos de software que compartilham uma arquitetura de domínio específico em comum.

Segundo Clements, Northrop (2001), linha de produtos de software pode ser definida como:

> A software product line (SPL) is a set of software intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way.

Nessa definição *core assets* significa um conjunto de *assets* ou artefatos disponíveis para serem reusados no desenvolvimento de novos produtos de software. *Os cores assets* formam o núcleo comum da família e são reutilizados cada vez que uma nova aplicação é desejada.

2.1.2 Considerações

A LPS é a técnica de reuso usada para o domínio mais restrito no qual há perspectiva de construção de um número grande de produtos com similaridades bem definidas. Ela admite diferentes tipos de *assets,* de análise (modelos de casos de uso), de projeto (modelos UML, padrões de projeto, ou arquiteturas) e de implementação (componentes ou *frameworks*). Isso a difere de componentes e *frameworks*.

Componente de software é um conjunto de elementos de software que podem ser chamados para executar uma determinada tarefa. No cenário de execução de um

componente de software, é a aplicação cliente que chama o componente para executar uma tarefa e controla a estrutura do programa e o fluxo de execução. Isso difere do cenário de execução do *framework* que é baseado no princípio bastante conhecido em inglês como *The Hollywood principle: "Don't call us, we'll call you." (*GAMMA, 1995). Isto é, é o *framework* que chama a aplicação, lhe oferece uma estrutura base, controla essa estrutura e o fluxo de execução do sistema. Como esse trabalho de mestrado pretende fornecer facilidades para os desenvolvedores de aplicações Smart TV integradas com a programação da TV, faz-se necessário optar por uma técnica de reuso que pode oferecer uma base estrutural, um esqueleto de código pronto para ser usado para criação de novas aplicações com menos esforço. Nesse caso, o reuso por meio de *framework* é a opção mais adequada.

2.2 Smart Tv

Smart TV é uma plataforma que integra recursos da Internet e da Web em TVs equipadas com dispositivo computarizado (interno ou externo a essas TVs). A ideia principal é compartilhar a tela da TV com aplicações já instaladas na TV ou obtidas em lojas virtuais de aplicativos e conteúdo.

2.2.1 Aplicações Smart TV

Aplicações Smart TV são programas computacionais que funcionam em TVs Digitais Conectadas. Com tais aplicações, usuários podem acessar conteúdo da Web via suas televisões e demais serviços acessíveis pela Internet. Uma aplicação Smart TV pode acessar recursos específicos da TV como, alterar o volume do programa da TV ou reproduzir um vídeo que não faça parte do programa da TV. Funções oferecidas pela API do sistema de arquivos da Smart TV permitem aos usuários utilizarem um sistema de armazenamento de arquivos. Os usuários podem baixar aplicações tanto da loja virtual do fabricante e instalá-las em seus televisores, bem como criar suas

próprias aplicações Smart TV com SDK (*Software Development Kit*) fornecido pelo fabricante.

Existem três tipos de aplicações Smart TV, aplicações *full-screen, single-wide* e *ticker* (SAMSUNG, 2014). As aplicações *fullscreen* são aquelas que ocupam a tela inteira do televisor, não compartilham o controle remoto com a TV enquanto estão em execução e não são associadas à programação televisiva. Como exemplo de aplicações Smart TV *fullscreen,* temos aplicações de exibição de filmes, jogos, exercícios físicos, entre outras.

Aplicações *single-wide* têm as características das aplicações *full-screen,* entretanto difere delas por serem exibidas numa parte da tela da TV, ou seja, elas dividem a tela do televisor com o programa da TV. Como exemplo desse tipo de aplicações, temos o TWITTER.

Aplicações *ticker* compartilham a tela de exibição com a imagem da TV e também o controle remoto. Como as aplicações *fullscreen* e *single-wide*, as *ticker* não são associadas à programação televisiva, portanto permanecem na tela da TV enquanto o usuário faz outras coisas com a TV, como mudar de canal, alterar o volume da TV, entre outras. A aplicação de relógio disponível numa parte da tela enquanto o programa da TV está ocorrendo, é um exemplo de aplicação *ticker*.

2.2.1.1 Arquitetura de Aplicações Smart TV

Na figura 3, é apresentada uma arquitetura típica de aplicações Smart TV.

O *Application Manager* (Gerenciador de Aplicações) controla a instalação e remoção das aplicações na TV, e executa um conjunto de tarefas para elas.

Browser Layer é a camada de navegadores na qual, as aplicações Smart TV são executadas. Vale ressaltar que essa camada de navegadores pode ser um sistema operacional específico dependendo do fabricante.

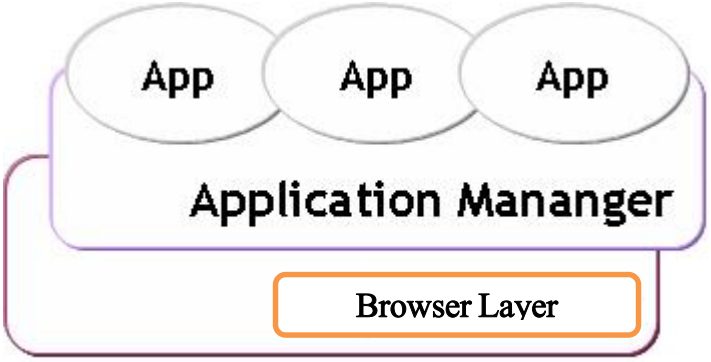


Figura 3: Arquitetura Típica de Aplicações Smart TV
[Fonte: SamsungDForum, 2014]

2.2.1.2 Estrutura de Arquivos de Aplicações Smart TV

Na figura 4, é apresentada uma estrutura básica típica de arquivos de aplicações Smart TV. Nota-se que uma aplicação Smart TV consiste em arquivos HTML, CSS e JavaScript. A página HTML mostra a estrutura principal da aplicação, o arquivo CSS apresenta o estilo das páginas (das telas no caso) e os arquivos JavaScript controlam o comportamento de aplicações.

Quando se deseja executar uma aplicação na TV, deve-se criar um arquivo de configuração contendo informações sobre os recursos da TV que essa aplicação fará uso e também a versão da própria aplicação.

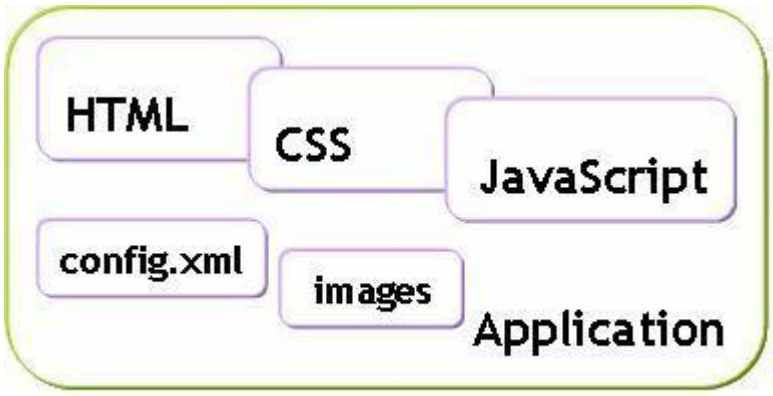


Figura 4: Estrutura de arquivos Típica de Aplicações Smart TV
[Fonte: SamsungDForum, 2014]

2.2.2 Software Development Kit

O uso do SDK permite o desenvolvimento de aplicações e teste das suas funções básicas em um emulador (interno ou externo ao SDK). Entretanto o emulador não é uma simulação exata da TV, uma vez que ele possui uma plataforma de hardware diferente da plataforma da TV. A única maneira de se ter certeza de como a aplicação desenvolvida se comportará na TV é executá-la no aparelho da TV.

A plataforma do PC, na qual o emulador é executado, é diferente da plataforma da TV nos seguintes pontos:

- Possui pouca memória: aplicações podem sofrer o fenômeno de "*out of memory*";
- O tempo de resposta de teclas do controle remoto é diferente;
- Nem todas as teclas do controle remoto são alocadas para aplicação;
- O *playback* do Vídeo e áudio pode ter um comportamento diferente devido ao tipo diferente de hardware;

Os recursos integrados ao SDK permitem ao desenvolvedor fazer o *upload* das suas aplicações do seu servidor local para a TV a fim de testes.

Como pode ser observado na figura 5, que ilustra uma interface inicial típica de SDK, a interface está dividida em cinco principais partes. A primeira parte, BARRA DE MENU ao topo, inclui funcionalidades como EMULATOR, DEBUG, TOOLS, etc. A segunda parte, a da esquerda, apresenta a estrutura de arquivos da aplicação. A terceira parte, a do meio, refere-se ao editor de arquivos HTML, CSS e JAVASCRIPT. Na quarta, logo abaixo da terceira parte, encontra-se o CONSOLE. A última parte é a da direita, onde se encontram as propriedades dos elementos HTML, CSS e JavaScript.

Este capítulo apresentou, primeiramente, conceitos fundamentais e técnicas relacionados ao reuso de software. O conhecimento adquirido nesta seção permitiu escolher definir a forma e a técnica de reuso a ser adotada neste trabalho, os quais foram o desenvolvimento para reuso e reuso por *framework* respectivamente. Em seguida, foi apresentada a plataforma Smart TV (plataforma principal de execução das aplicações deste trabalho).

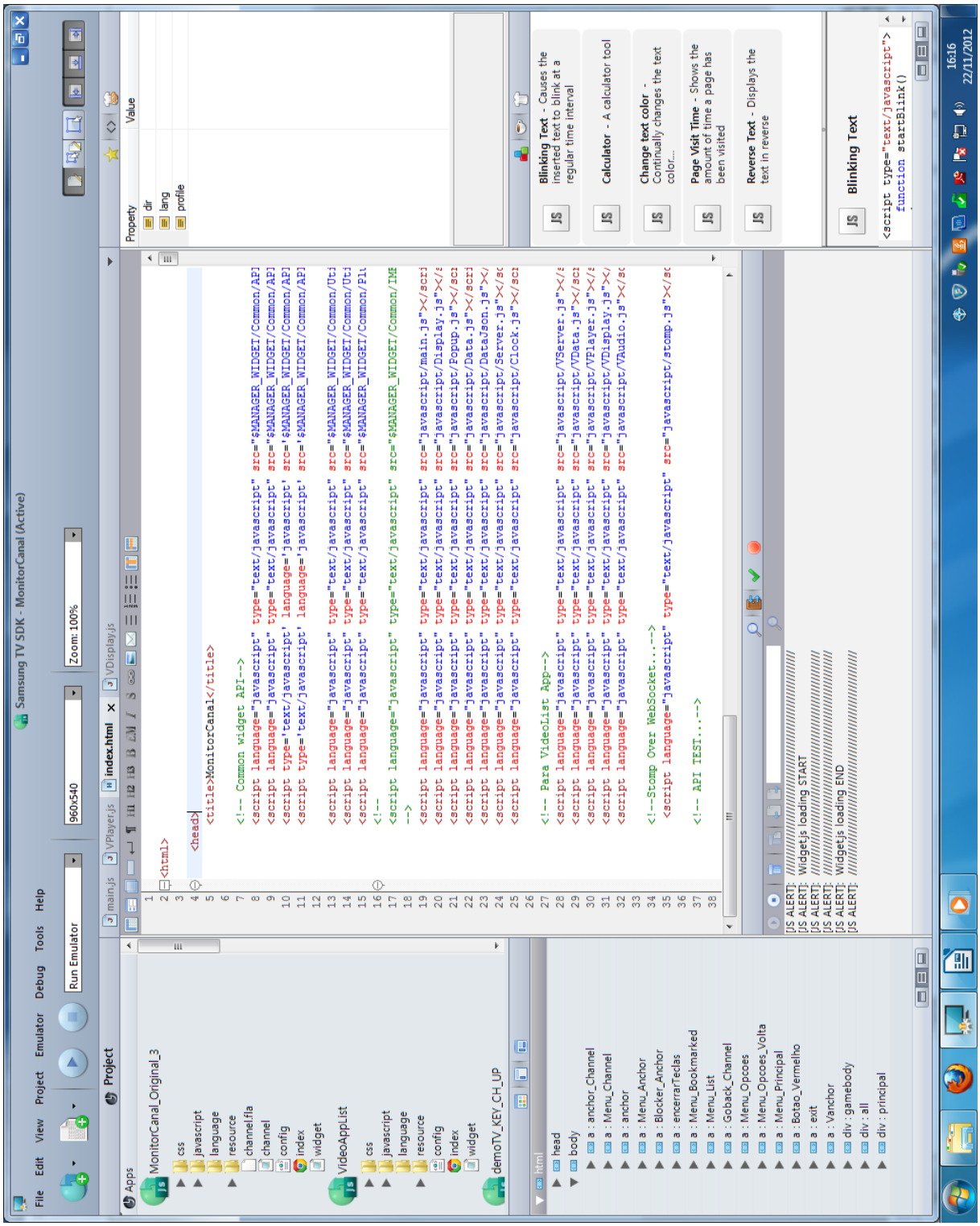


Figura 5: Interface Principal do SDK Típico para Smart TV

[Fonte: SamsungDForum, 2014]

Capítulo 3

SINCRONIZAÇÃO E NOTIFICAÇÃO

Para se alcançar o objetivo de oferecer aplicações integradas à programação televisiva torna-se essencial o conhecimento do que está sendo apresentado ao telespectador a cada momento. Essa informação pode então ser notificada a interessados e utilizada para promover a sincronização entre a programação e a aplicação.

A sincronização no domínio da TV é um assunto amplo. Como apresentado em (TEIXEIRA et al., 2015), diferentes aspectos devem ser considerados. A notificação, por sua vez, realizada através de serviços, deve atender requisitos para viabilizar a sincronização.

As seções deste capítulo são dedicadas a esses dois assuntos, sincronização e notificação. Discutem-se alguns aspectos da sincronização que aplicações Smart TV devem se preocupar para promover a integração com a programação televisiva, e como são geradas as demandas a serviços de notificação.

3.1 Aspectos da Sincronização

Em (TEIXEIRA et al., 2015), os autores estudam a sincronização no contexto de aplicações multimídia e consideram que sincronização consiste em garantir que ações sejam realizadas obedecendo-se referências temporais definidas por um relógio ou estabelecidas pela ocorrência de eventos, admitindo-se tolerâncias que variam em função do tipo de aplicação. No caso de programação televisiva, o caráter do evento a ser considerado como referência é um dos aspectos importantes da sincronização abordados neste trabalho.

Outros três aspectos relevantes para a integração de aplicações com a programação televisiva são: Fonte - refere-se a quem proporciona a informação sobre a ocorrência do evento; Acoplamento - relaciona-se à tolerância admitida pela aplicação em desvios ao momento de referência; Exposição - categoriza duas situações possíveis: informações de sincronismo explícitas e informações de sincronismo implícitas.

3.1.1 Sobre o caráter do evento

De maneira geral o caráter do evento pode ser classificado em sintático ou semântico.

Classificam-se como sintáticos os eventos relacionados a características físicas da programação. Exemplos não exaustivos dessa classe podem ser: início ou término de um programa, início ou término de um comercial, ocorrência de um padrão específico de quadro, ocorrência de um padrão específico no áudio, ocorrência de um período de silêncio no áudio, ocorrência de frame com certa predominância de cor, derivada significativa no volume do áudio, tempo decorrido desde o início do programa ou de um conjunto de comerciais, entre outros.

Classificam-se como semânticos os eventos em que o entendimento do conteúdo da programação televisiva é necessário. Exemplos não exaustivos dessa classe podem ser: nome do programa que se iniciou ou terminou, nome do programa

em exibição - solicitado a qualquer momento, produto que está sendo anunciado no comercial, identificador de um comercial, preço do produto anunciado, nome da atriz em foco na cena, marca da camisa vestida pelo ator, classificação etária do programa, entre outros.

Exemplos, como ocorrência de uma vinheta, poderiam ser classificados tanto em uma como outra classe, dependendo se forem encarados como uma sequência específica de frames ou como um trecho de conteúdo.

Eventos sintáticos são mais apropriados para serem detectados automaticamente ou com pouca ajuda humana. Eventos semânticos em geral requerem intervenção humana mais significativa, pelo menos em alguma fase de anotação, e/ou processamento mais intenso.

3.1.2 Sobre a fonte

A emissora, que é quem melhor conhece o que e quando um determinado conteúdo está sendo transmitido, é a fonte mais natural e segura para difundir sinais de sincronismo em relação a sua programação. Entretanto, por questões relativas a modelos de negócio, a emissora pode não se interessar em propiciar esse tipo de informação. Nesse caso, a sincronização realizada por terceiros, que podem ser anunciantes, agências publicitárias ou mesmo entidades não envolvidas na cadeia tradicional do negócio, pode ser uma alternativa importante.

A sincronização por terceiros pode ser por monitoramento automático ou semiautomático, em que um sistema computacional realiza o monitoramento de canais da TV e gera sinais de sincronismo de interesse. Pode ainda ser por monitoramento humano, em que a geração de sinais de sincronização é provida por uma pessoa, com essa finalidade exclusiva, ou que esteja promovendo a interação de telespectadores em ambiente de TV Social.

MELO (2014) propõe uma solução de sincronização baseada em análise supervisionada em que uma entidade é responsável por monitorar canais de TV e gerar sinais de sincronização para aplicações interessadas. O serviço proposto oferece avisos às aplicações registradas em função de eventos que possam estar ocorrendo na

programação dos canais monitorados. Uma API do serviço foi criada e disponibilizada para gerar esses sinais de sincronização de maneira otimizada. Os sinais de sincronização gerados são relacionados a diversos eventos dos programas da TV, como por exemplo, início e término de blocos comerciais, início e término de programas da TV, entre outros.

POMPONET (2014) propõe um sistema de TV Social no qual um agente humano, através da integração de tecnologias web e de dispositivos móveis, incentiva telespectadores a interagirem consigo e com outros assistindo um mesmo programa. Para fazer isso o agente necessita acompanhar atentamente o programa em questão. Assim, ele tem condições também de oferecer diretamente ou indiretamente a aplicações interessadas diferentes sinais de sincronismo, que informam eventos ocorrendo no programa.

A sincronização pode ainda ser explicitamente provida pelo próprio telespectador. Ao assistir televisão o usuário pode interagir com uma aplicação, através do controle remoto, e informá-la sobre eventos ocorrendo no programa. Em (TEIXEIRA et al., 2011), por exemplo, propõe-se uma aplicação para futebol em que o telespectador pode, sincronizado ao acionamento de uma tecla do controle remoto, promover a exibição de um curto trecho do hino de seu time (o que ele provavelmente fará apenas quando ocorrer um gol de seu time).

Pode-se considerar também que a fonte provedora de sinais de sincronismo seja inexistente. Isso acontece no caso da sincronização implícita, quando a própria aplicação se encarrega de processar o conteúdo do que está sendo visto pelo usuário no momento em busca de alguma informação que possa servir para sincronização. Essa é uma opção que passa a ser viável à medida em que os processadores de televisores e demais receptores, como dispositivos móveis, ganham mais capacidade de processamento.

3.1.3 Sobre o grau de acoplamento

O acoplamento da sincronização relaciona-se à tolerância admitida pela aplicação em desvios ao momento de referência. Em geral, o que se denomina

sincronização fortemente acoplada refere-se a situações em que existe pouca tolerância na diferença entre o momento em que ocorre o evento e o momento que se considera como referência para a sincronização. O oposto, quando essa tolerância é grande, denomina-se sincronização levemente acoplada. Não se pretende definir com precisão o que é pouca ou grande tolerância. Exemplos podem estabelecer extremos e servirem de referência. Evidentemente situações intermediárias poderão ser consideradas como pertencentes tanto a uma categoria como a outra.

Um exemplo extremo de sincronização fortemente acoplada é a sincronização labial para dublagens. Na outra extremidade de sincronização levemente acoplado pode-se citar como exemplo a sincronização por guias de programação eletrônico (*EPG*) que informam horários de programas. Em um jogo de futebol, por exemplo, em que já se sabe que terá duração aproximada de 105 minutos (45 do primeiro tempo, 45 do segundo tempo e 15 do intervalo), aplicações podem, tomando-se por base o EPG, oferecer conteúdo complementar ao telespectador levemente acoplado com a partida.

Um mesmo evento pode ser associado tanto a uma sincronização levemente acoplada como fortemente acoplada, dependendo da aplicação. O evento "início de programa" por exemplo, pode compor uma sincronização levemente acoplada para aplicações que vão apresentar conteúdo complementar genérico, que pode ser visualizado ao longo do programa. Entretanto, para uma aplicação que pretende alertar o telespectador para que mude de canal pois programa de seu interesse está começando, o evento "início de programa" compõe sincronização fortemente acoplada.

3.1.4 Sobre a exposição

Os sinais de sincronismo podem ser explicitamente produzidos ou estar implícitos e ocultos no conteúdo. No caso da TV Digital Interativa (TVDI), as emissoras podem enviar, junto com o conteúdo (shows, filmes, novelas, comerciais, entre outros), também dados de sincronismo. Essa sincronização é feita através de marcas temporais no fluxo de transporte da TV Digital usando MPEG-TS (MPEG Transport Stream), por exemplo, para enviar *timestamps* que permitem a apresentação de conteúdo

complementar sincronizado com o programa da TV. Dessa forma, um *middleware,* no ambiente do telespectador, processa esses dados, extrai as informações de sincronismo e as entrega à aplicação TVDI para que as ações programadas possam ser realizadas.

Alternativamente, no caso de aplicações Smart TV, sinais explícitos de sincronização podem ser enviados à aplicação através da Internet. Emissoras e desenvolvedores de aplicações podem juntos determinar quais informações de sincronismo são interessantes para cada tipo de programa da TV e para cada tipo de aplicação. As informações obtidas relativas a cada tipo de programa televisivo e a cada tipo de aplicação podem ser disponibilizadas na nuvem por meio de um serviço *web*, por exemplo, para que aplicações possam se registrar para receber notificações. Durante a transmissão de um programa da TV no qual aplicações se registraram para receber notificações, a emissora envia sinais de sincronização diretamente para as aplicações ou pode enviá-los para um determinado servidor de notificações e esse último, por sua vez, vai propagá-los para as aplicações.

Como outro exemplo de sincronização explícita, emissoras podem enviar informações de sincronização dentro do vídeo no formato de QR-Codes (Quick Response Codes) (COSTA et al., 2013). As âncoras de sincronização são visíveis tanto para o usuário como para as aplicações. O objetivo em (COSTA et al., 2013) é usar o QR-Codes para sincronizar programa da TV com aplicação *mobile*. Ao aparecer um QR-Code na tela, COSTA et al. (2013) sugerem que o usuário tire uma foto do QR-Code com a câmera do dispositivo móvel e os processe usando um aplicativo com esse propósito para extrair informações de sincronização contidas nele. As informações de sincronismo extraídas são passadas para aplicação *mobile*. Por exemplo, se a informação extraída for uma *url,* o navegador disponível no dispositivo pode ser invocado para o acesso. Para aplicação Smart TV seria necessário técnicas de processamento de imagem para extrair o conteúdo do QR-Code. Para isso, a aplicação teria que monitorar o fluxo de vídeo dos programas da TV e capturar a tela a cada intervalo de tempo definido. Cada imagem da tela capturada seria submetida ao processamento para identificar frames que contem QR-Codes, recuperar os QR-Codes e tratá-los para extrair as informações de sincronismo contidas neles.

Uma forma de sincronização explícita semelhante a QR-Codes é através de uso da esteganografia digital (CARVALHO, 2008). A diferença é que, enquanto as âncoras de QR-Codes são visíveis para o usuário e para aplicações, as da esteganografia são visíveis somente para as aplicações, ou seja, a esteganografia procura ocultar informações dentro de uma mídia como texto, imagem, áudio e vídeo. Para as aplicações identificarem as informações de sincronização, emprega-se processo semelhante ao usado no QR-Code. Entretanto, em vez de se utilizar o processamento de imagem, neste caso, utiliza-se o processamento de vídeo, pois a informação é ocultada em vários *frames*. Zhao, Koch, Luo (1998) classifica a esteganografia em 4 grandes áreas de aplicação: *Copyright Protection, Authentication, Secret and Invisible Communication, and Hidden Annotation.* Esteganografia em mídias digitais para sincronização, como sugerida nesse trabalho, se encaixa em uma nova área: *"Synchronization"*.

Se informações de sincronismo não forem explicitamente providenciadas ainda assim é possível promover algum tipo de sincronismo entre aplicações e a programação televisiva. A detecção automática de eventos sintáticos, como apresentados em 3.1.1, através da detecção de características físicas decorrentes de outros propósitos que não o sincronismo, pode dar dicas úteis para sincronização. Um exemplo de característica física é a inserção de quadros totalmente pretos ou totalmente brancos entre diferentes comerciais, que a legislação de alguns países exige.

3.2 Notificação

Para promover o sincronismo tratado no contexto deste trabalho, é preciso estabelecer um protocolo de comunicação entre quem precisa receber sinais de sincronismo (aplicação da TV) e quem fornece esses sinais (serviços de sincronização). Para isso, faz-se necessário criar uma API de notificações, ou seja, uma interface que permite aplicações terem acesso às diferentes funcionalidades de notificações oferecidas, registrar essas notificações no serviço de sincronização e esse último, por

sua vez, vai avisar as aplicações sobre eventos ocorrendo na programação da TV e relativos às notificações previamente registradas por elas.

As notificações tratadas neste trabalho são de vários tipos, como por exemplo, início e término de blocos comerciais, início, pausa e término de programas da TV, cenas de sexo, de violência, de crime, e também algumas notificações geradas a partir dos eventos acionados pelo próprio usuário, como por exemplo, o envio do canal sintonizado no momento para demais usuários previamente inscritos para essa notificação, o registro do monitoramento e compartilhamento do estado (*qual canal sintonizou?*, *qual nível do volume?*, *por exemplo*) de um usuário específico aos demais previamente registrados e a ele vinculados.

A API de notificações descrita no Anexo A classifica-se em API síncrona e assíncrona. A assíncrona é aquela que disponibiliza funcionalidades relativas às notificações oriundas dos eventos sensíveis ao estado enquanto a síncrona permite acesso às notificações relativas aos eventos de ações imediatas. Notificação de início de comercial, por exemplo, no qual a aplicação do usuário inscreve-se para sua recepção e esse aviso pode acontecer em qualquer momento durante a sessão do telespectador assistindo televisão. Esse é um exemplo de uma notificação que pode ser requisitada pela API assíncrona. Durante um bloco comercial, o usuário pode acionar uma determinada tecla do controle remoto e ser notificado imediatamente sobre o *token/ID* do comercial passando no momento para que determinadas ações possam ser tomadas. Esse é um exemplo de notificações acessadas pela API síncrona.

Em relação à fonte das notificações tratadas neste trabalho, pode se dizer que elas podem ser fornecidas por serviço, no qual aplicações se registram e são avisadas por esse serviço. Pode ser também de forma direta (com quem oferece o serviço), neste caso, o *framework* aqui proposto implementa o protocolo do provedor desse serviço, consequentemente, facilita aplicações se registrarem e receberem notificações desejadas de forma transparente.

Quando for indireta, ou seja, quando há necessidade de processamento para obtenção do sinal, a sincronização fortemente acoplada deve ser realizada no cliente para evitar possível variância de atraso na rede. Em uma sincronização fortemente

acoplada a identificação inequívoca dos pontos de sincronismo no conteúdo da TV pode ser feita usando técnicas de processamento de áudio e vídeo. No exemplo da sincronização labial citada acima, ASR (Automatic Speech Recognition) (GAO, ZHAO, YAN, 2010) pode ser usado no áudio do programa da TV para recuperar cada fala e essas informações podem ser utilizadas para sincronização e apresentação de conteúdo complementar (áudio de dublagens, por exemplo).

Capítulo 4

SYNCSMARTV

A proposta deste trabalho é a concepção e desenvolvimento de um *framework* de referência que facilite a construção de aplicações Smart TV integradas e sincronizadas com o conteúdo audiovisual apresentado na TV. Pretende-se criar mecanismos que permitam ao desenvolvedor construir aplicações nesse domínio, de forma transparente, sem a necessidade de se preocupar com detalhes de implementação de nível baixo, como por exemplo, como usar API's do módulo responsável pelas funções básicas de canais da Smart TV para implementar a funcionalidade de mudança de um canal para outro. Busca-se permitir que esse desenvolvedor se preocupe mais com a lógica do negócio das suas aplicações e consequentemente possa construir aplicações integradas com menor custo e esforço.

Neste trabalho, entende-se por aplicações integradas, aquelas sincronizadas com a programação televisiva. Para construir tais aplicações, alguns problemas listados na seção 4.1 devem ser contornados.

4.1 Domínio

Para construir aplicações Smart TV integradas, os problemas listados a seguir devem ser contornados. O *framework* proposto neste trabalho visa cobrir todos esses

problemas envolvidos, encapsulando módulos reutilizáveis que implementam soluções para esses problemas. A seguir, a lista dos problemas envolvidos:

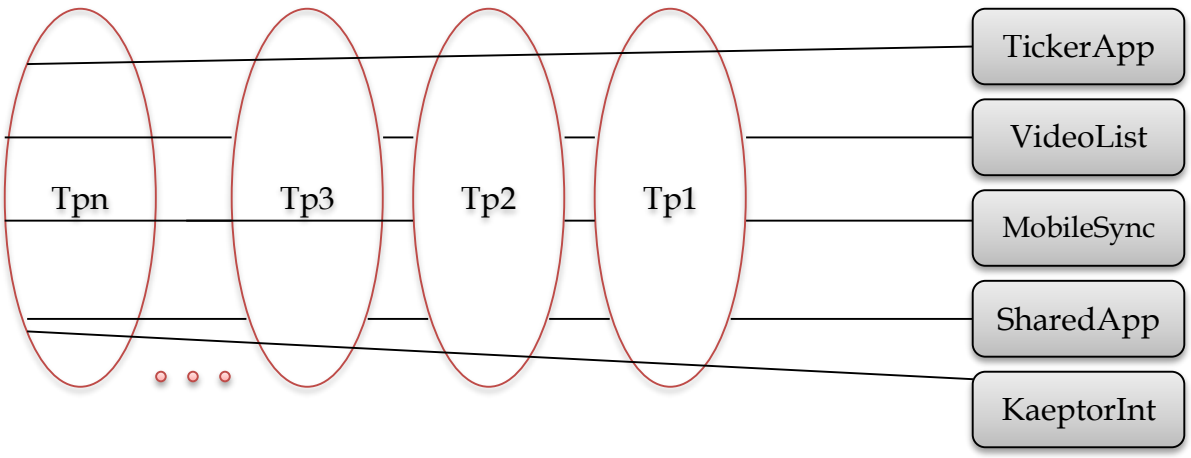
1. Como sincronizar o conteúdo da TV com aplicações Smart TV;
2. Como dispositivos móveis (*Smartphones*, *Tablets*, e outros) que se encontram no ambiente da TV podem interagir com ela;
3. Como dispositivos móveis conectados no ambiente da TV podem saber da existência de um serviço disponível na TV para o uso;
4. Como aplicações Smart TV podem atuar sobre controles da TV, como mudar de canal, controlar volume, entre outras;
5. Como aplicações *ticker* devem compartilhar o controle remoto com a TV;
6. Como uma aplicação Smart TV pode identificar o canal sendo assistido pelo usuário para registrá-lo no serviço de monitoramento de canais da TV.

4.2 Metodologia

A pesquisa iniciou-se com a definição de uma plataforma Smart TV de referência para os estudos, os testes e os desenvolvimentos. Embora a escolha tenha caído sobre produto de um fabricante específico, foi importante reunir, por meio desse, as características representativas da maioria das Smart TVs, pelo menos aquelas com dispositivos computacionais embutidos.

Para a construção e validação do *framework* proposto neste trabalho foi desenvolvido um conjunto ferramental que deu apoio a esse processo. Os artefatos desenvolvidos e/ou analisados durante esse processo serviram para fatorar módulos comuns e reutilizáveis existentes entre eles, e também para avaliar a viabilidade de implementação de cada um dos módulos dentro do domínio da pesquisa de mestrado. Os módulos reutilizáveis fatorados fazem parte dos componentes do *framework*.

Na figura 6 são apresentados os principais artefatos que levaram a definição dos componentes reutilizáveis contidos no *framework* apresentado.



Legenda:

Tp1: TVNewsApp

Tp2: DescriptionApp

Tp3: BackApp

Tpn: TVMonitor

Figura 6: Artefatos consideradostruídos para apoiar a construção do *framework*

Na extremidade à direta da figura encontram-se algumas aplicações (representadas em retângulos cinzas) construídas e que apresentam características típicas de uma ou mais aplicações/componentes situadas à esquerda (representadas em forma de elipse). Essas aplicações são elementares, sem interesse para usuários, construídas com finalidade de explorar funcionalidades das Smart TV. Na esquerda, encontram-se aplicações mais complexas, de interesse de usuários, algumas desenvolvidas durante este trabalho, outras tomadas como referência. Uma aplicação da extremidade direita apresenta funcionalidades similares com um componente de aplicações da esquerda quando houver uma linha que origina-se no retângulo correspondente à aplicação da direita e que passa por cima da elipse correspondente à aplicação da esquerda.

A seguir, uma breve descrição dessas aplicações.

- **TickerApp:** é uma aplicação Smart TV que consegue permanecer na tela, isto é, que continua executando em *front-end* enquanto o telespectador estiver mudando de canais. A aplicação exibe uma mensagem simples "App executando..." na tela enquanto o telespectador navega pela TV.

Embora essa aplicação pareça simples, contém um grau de complexidade considerável no que tange ao gerenciamento de uso de controle remoto entre a aplicação e a TV.

- **VideoList:** uma aplicação que consegue receber notificações, geradas pelo próprio telespectador, de que iniciou-se um comercial no canal corrente e com base nessa informação iniciar o *playback* de uma lista de vídeos. VideoList lê o título, a descrição e a informação do URL de cada vídeo disponível em um servidor de mídia remoto/local e apresenta-os para o usuário permitindo que ele toque, pare, pause, avance e volte no vídeo.
- **MobileSync:** é uma aplicação que implementa o protocolo *http* para promover a comunicação *two-ways* e compartilhamento de conteúdo de mídia, entre a TV e o celular, de forma sincronizada com a programação da TV. Com ela, o telespectador munido de um dispositivo móvel (*smartphone, tablet*) com suporte para *Bluetooth*, *Wifi*, por exemplo, é avisado sobre um serviço disponível da TV e pode conectar seu dispositivo à TV.
- **SharedApp:** permite ao usuário usufruir da tela da Smart TV utilizando-a como "*second screen*" para exibir a galeria de fotos e tocar uma lista de vídeos encontradas no seu dispositivo móvel ou em servidor de mídias remoto/local.
- **KaeptorInt:** é uma aplicação que implementa um ou mais protocolos de um determinado provedor de serviço de sincronização facilitando aplicações Smart TV receberem notificações sobre a programação televisiva de forma transparente.
- **BackApp:** é uma aplicação Smart TV que oferece uma interface gráfica para que o usuário possa registrar um canal da TV para o monitoramento. Dessa forma, ao estar em um determinado canal e receber uma notificação relativa ao canal sendo monitorado, a aplicação

muda automaticamente de canal para aquele sendo monitorado, por exemplo.

- **DescriptionApp:** permite ao usuário ter acesso a detalhes do produto em foco em um comercial ou em um bloco comercial, por exemplo.
- **TVNewsApp:** é uma aplicação que consegue informar o telespectador sobre o estágio em que se encontra o decorrer do jornal de um canal de TV. Com isso, o telespectador consegue acompanhar a programação do jornal de um canal da TV sem a necessidade de sempre sintonizar aquele canal.
- **TVMonitor:** é uma aplicação integrada completa (contendo um conjunto considerável de características apresentadas por demais artefatos descritos acima), capaz de receber notificações sobre eventos que estejam ocorrendo na programação da TV e de permitir aos usuários interagir com os recursos da própria plataforma da TV e demais conteúdos complementares, por exemplo, de forma sincronizada. Esses usuários, ao assistirem televisão, podem utilizar o controle remoto para fazer algumas configurações da aplicação. O telespectador, munido de um dispositivo móvel (*smartphone, tablet*) com suporte para *Bluetooth*, *Wifi*, por exemplo, ao ser avisado sobre um serviço disponível da TV, pode conectar seu dispositivo à TV e usufruir desse serviço para fazer o *playback* de seus vídeos localizados no celular ou em um servidor de mídia remoto ou também visualizar a galeria de fotos do seu celular na tela da TV.

Além das aplicações descritas acima, também foi implementada **AdminSync**. Essa aplicação emula sinais de sincronização gerados por terceiros. Isso facilita o desenvolvimento e os testes do SyncSmartv. Essa aplicação foi implementada na forma de aplicações *Android*.

Para conduzir o processo de desenvolvimento do *framework* foi adotada a metodologia proposta por BOSCH et al. (1999) descrita nos próximos parágrafos. BOSCH et al. (1999) afirmam que o desenvolvimento de um *framework* é diferente do

desenvolvimento de uma aplicação comum. Isso se deve à necessidade que o projeto de *framework* apresenta de cobrir todas as características pertinentes a um determinado domínio. Tomando como base as dificuldades encontradas durante o desenvolvimento de *frameworks* BOSCH et al. (1999) criaram seis atividades para auxiliar o desenvolvimento de um *framework*:

1. Análise de domínio: o objetivo desta atividade é descrever o domínio no qual o *framework* será desenvolvido, levantar seus requisitos e identificar os conceitos envolvidos. O resultado dessa atividade é um modelo de análise de domínio. Normalmente, a maioria dos desenvolvedores utiliza as aplicações já desenvolvidas no domínio e/ou os documentos de requisitos ou modelos de casos de uso para esse propósito;

2. Projeto arquitetural: recebe como entrada o modelo de análise de domínio da atividade anterior e cria-se um projeto arquitetural do *framework* a partir dele;

3. Projeto do *framework*: nessa atividade montam-se os diagramas de classes do *framework*;

4. Implementação: as classes abstratas e concretas são implementadas utilizando a linguagem de programação escolhida;

5. Teste do *framework*: o objetivo dessa atividade é avaliar o *framework* com finalidade de verificar se possui as funcionalidades planejadas e avaliar também a sua usabilidade. Dependendo do resultado do teste, pode haver ou não a necessidade de remodelar o *framework*;

6. Documentação: essa atividade é uma das mais importantes em que se descreve como usar o *framework*, ou seja, deve-se elaborar o manual do usuário e um documento do projeto do *framework*.

Vale ressaltar que na atividade "Teste do framework" foram realizados dois tipos de avaliação: 1. verificação das facilidades oferecidas pelo framework para desenvolvedores; 2.

verificação da usabilidade e aceitação pelos usuários de aplicações desenvolvidas com o framework.

Na próxima seção é apresentado o desenvolvimento passo a passo do framework proposto neste trabalho utilizando a metodologia de BOSCH et al. (1999).

4.3 Desenvolvimento

Esta seção apresenta a realização das atividades da metodologia de desenvolvimento adotada para construção deste *framework*.

A tarefa mais difícil quando se trata de implementar uma abordagem para reuso é, de fato, a descoberta do que realmente deve ser reutilizável entre os requisitos de domínio. Para o desenvolvimento do *framework* deste trabalho, as seguintes atividades (descritas na seção 3.2) foram realizadas: 1. Análise de domínio, 2. Projeto arquitetural, 3. Projeto do *framework,* 4. Implementação, 5. Teste do *framework* e 6. Documentação.

Antes de entrar em detalhes sobre o processo de desenvolvimento apresentado acima, vamos dar uma visão geral de algumas funcionalidades comuns identificadas que podem ser implementadas e reusadas em diferentes aplicações que sejam sincronizadas com o programa televisivo.

4.3.1 Visão Geral

Os requisitos de domínio deste trabalho foram descobertos de maneira incremental, desenvolvendo um conjunto de aplicações Smart TV integradas com os programas da TV e analisando e identificando as similaridades existentes entre elas.

Ao realizar esse procedimento, observou-se em aplicações construídas que alguns requisitos funcionais e não funcionais são comuns a elas:

1. Meios para receber informações de sincronização;
2. Meios para comunicar-se com outros dispositivos que se encontram no ambiente como *Smartphones*, *Tablets*, entre outros;

3. Em aplicações com vários usuários, é preciso ter informações de cada um deles para controlar a interação. Essa funcionalidade oferece meios para obter informações sobre dispositivos móveis conectados no ambiente da TV com finalidade de controlar a interação de usuários;
4. Funcionalidades que permitam à aplicação atuar sobre controles da TV, como mudança de canal, controle de volume, entre outras;
5. Meios para que aplicações *ticker* compartilhem o controle remoto com a TV;
6. Recursos para identificação do canal sendo assistido pelo usuário no momento do seu registro no serviço de monitoramento de canais da TV.

Como requisitos não funcionais que podem dar apoio a esses requisitos funcionais, temos:

1. Boas Interfaces para viabilizar o recebimento de sinais de sincronização, sem a necessidade de interferência do telespectador para não perturbar a sua concentração ao assistir televisão;
2. Uma arquitetura flexível e fácil para estabelecer conexão com os serviços citados acima e demais artefatos de software do sistema.
3. Bom *layout* da aplicação, isto é boa resolução da tela, bom uso de *grid*, boa organização de componentes gráficos na tela, entre outros.

4.3.2 Análise de Domínio

Essa etapa visa comparar aplicações Smart TV integradas à programação com a finalidade de identificar as similaridades existentes entre elas. As comparações podem ser feitas entre aplicações já construídas ou entre aplicações construídas e especificações (documentos de requisitos ou modelo de casos de uso) de algumas aplicações a serem construídas. Nem toda similaridade encontrada na etapa da comparação precisa ser projetada e implementada. A decisão de construir ou não uma similaridade é uma tarefa difícil e depende muito da experiência do projetista, analista

ou desenvolvedor no domínio. Na Figura 7 é apresentado um diagrama de tarefas simplificado que descreve o procedimento utilizado para a descoberta dos requisitos do *framewok* deste trabalho. Essa figura baseia-se em fluxograma de GASPAR (2010), tendo sido acrescentada a fase "Identificar Artefatos".

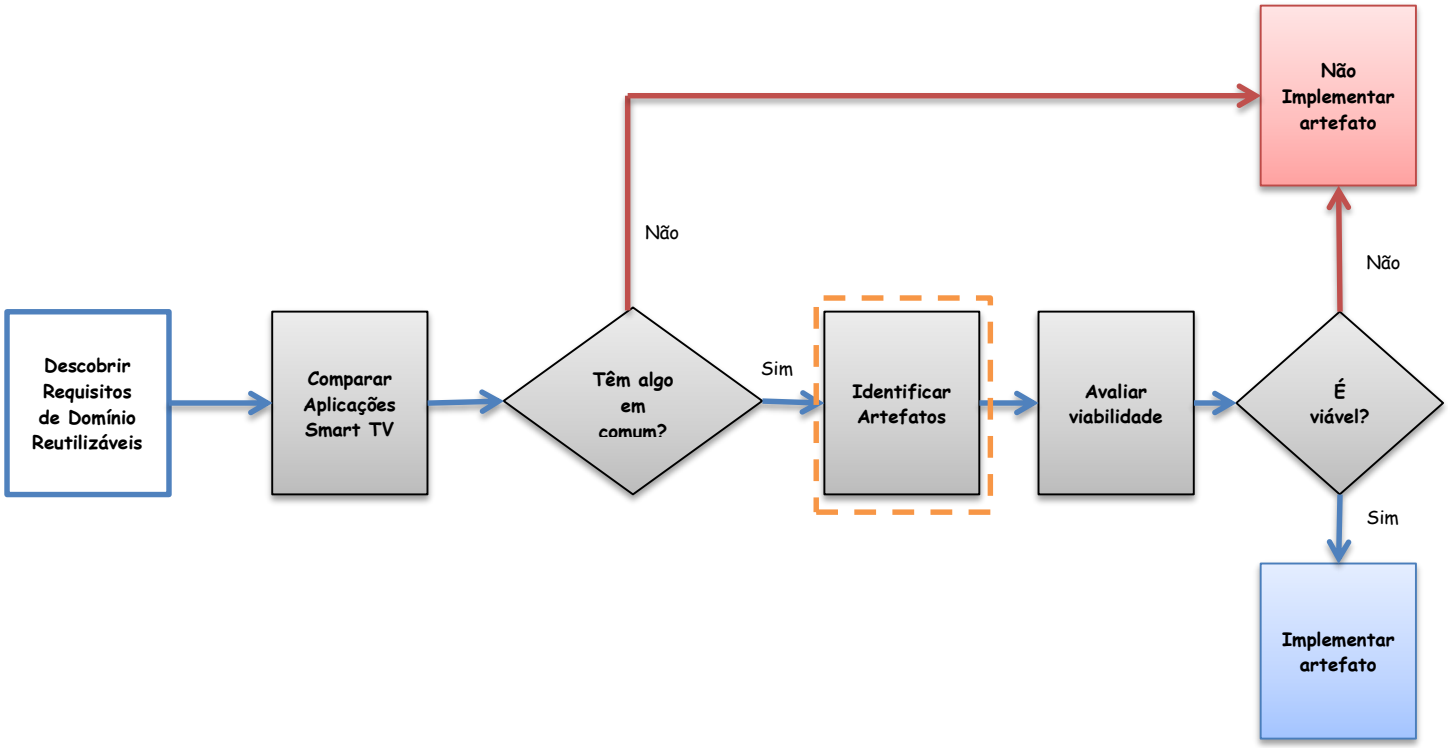


Figura 7: Diagrama de Tarefas para Descoberta de Requisitos
[Baseado no fluxograma do GASPAR (2010)]

Para conduzir essa atividade de análise de domínio foi desenvolvido um conjunto ferramental, descrito na seção 4.2, que deu apoio a esse processo.

Depois da construção das aplicações citadas na seção 4.2, o diagrama de tarefas da figura 7 foi seguido para a descoberta de requisitos de domínio e seus artefatos de software.

A seguir, ilustra-se apenas a análise de domínio do primeiro problema (como sincronizar o conteúdo de TV com aplicações Smart TV), isto é, exemplificou-se apenas como alguns artefatos adequados para resolver este problema foram identificados.

Vale ressaltar que, esse mesmo processo foi aplicado para realizar o restante da análise de domínio do SyncSmartv.

Primeiramente as aplicações são comparadas e ao se identificar uma funcionalidade em comum, é feito um projeto simplificado (pode ser uma descrição textual) de artefatos necessários para implementar a funcionalidade. Em seguida, uma avaliação de viabilidade é feita sobre os artefatos encontrados, analisando, por exemplo, o esforço estimado na construção e o nível de reuso, ou seja, quantas aplicações vão se beneficiar do artefato e depois decide-se se é viável ou não sua implementação.

Depois da identificação do primeiro requisito funcional (1.Meios para receber informações de sincronização) entre aplicações, citado acima, foram esboçados dois artefatos: 1. Um serviço de *login* e estabelecimento de conexão com o serviço de monitoramento de canais da TV e 2. Um serviço de registro e comunicação com o canal registrado. Na Figura 8, os casos de uso em branco são funcionalidades do artefato 1 (serviço de *login* e estabelecimento de conexão com o serviço de monitoramento de canais da TV) e os em cinza faz parte do artefato 2 (serviço de registro e comunicação com o canal registrado).

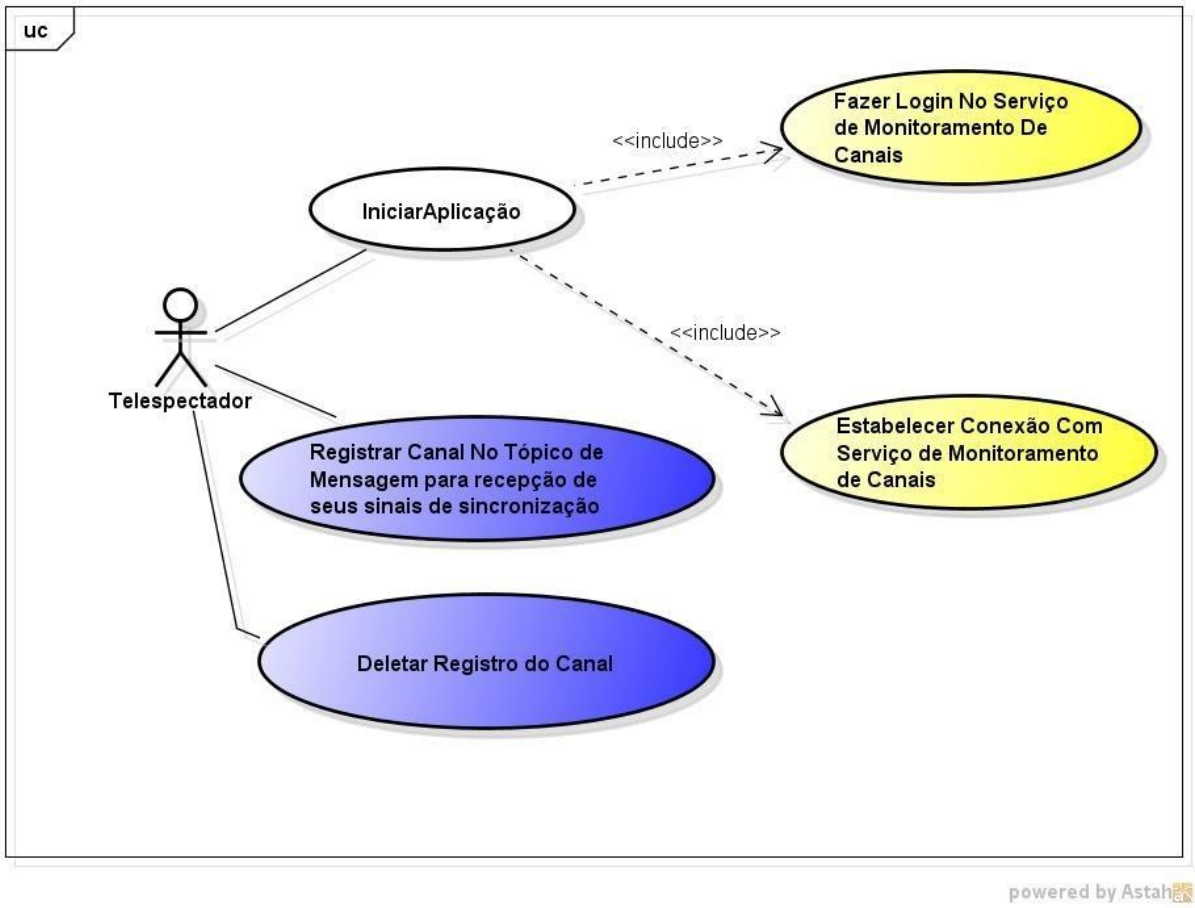


Figura 8: Parte do Diagrama de Casos de Uso da Aplicação TVMonitor

Na Figura 9 é apresentado um trecho de código *Javascript* para fazer o *login* da aplicação Smart TV no *backend* do serviço de monitoramento de canal e outro para estabelecimento da conexão entre o *frontend* (aplicação Smart TV) e o *backend* (serviço de monitoramento). Durante o *login,* a aplicação é autenticada por meio de duas chaves, uma de sessão e outra da credencial do dispositivo (TV no caso). Depois de ser autenticada no servidor, a aplicação precisa estabelecer uma conexão segura com o servidor para receber sinais de sincronização. Essa comunicação é feita por meio de *WebSocket API* que faz um papel de protocolo TCP na *web*. Para facilitar a troca de mensagens entre aplicações Smart TV e o servidor do *broker* de mensagens do Apache ActiveMQ, foi utilizado o protocolo de mensagens STOMP. Por fim, depois do estabelecimento da conexão, o *broker* está pronto para receber as notificações dos canais do serviço de monitoramento e propagá-las para as aplicações clientes registradas.

```
Main.Login = function(username, password){

    kaeptor.auth.user.device = "teste";
    kaeptor.auth.login({
        username:   username,
        password:   password,
        success:    function(){

                this.sucesso_Login = 1;
                alert("Logged!");
                Main.ConnectStomp();

        },
        error:       function(){

                alert("Not logged!");

        }
    })
}
```

```
Main.ConnectStomp = function(){

        var destination;
        var login;
        var passcode;
        login = 'guest';
        passcode = 'guest';

        var url = 'ws://200.18.98.24:61614/stomp';
        client = Stomp.client(url);

        client.debug = function(str){
            alert(str + "\n");
        };
        var onconnect = function(frame) {
            client.debug("connected to Stomp");
            boolReady = true;

        };

        client.connect(login, passcode, onconnect);

        return false;
}
```

Agora o objeto ActiveMQ está pronto para registrar um canal e encaminhar as suas notificações para a aplicação

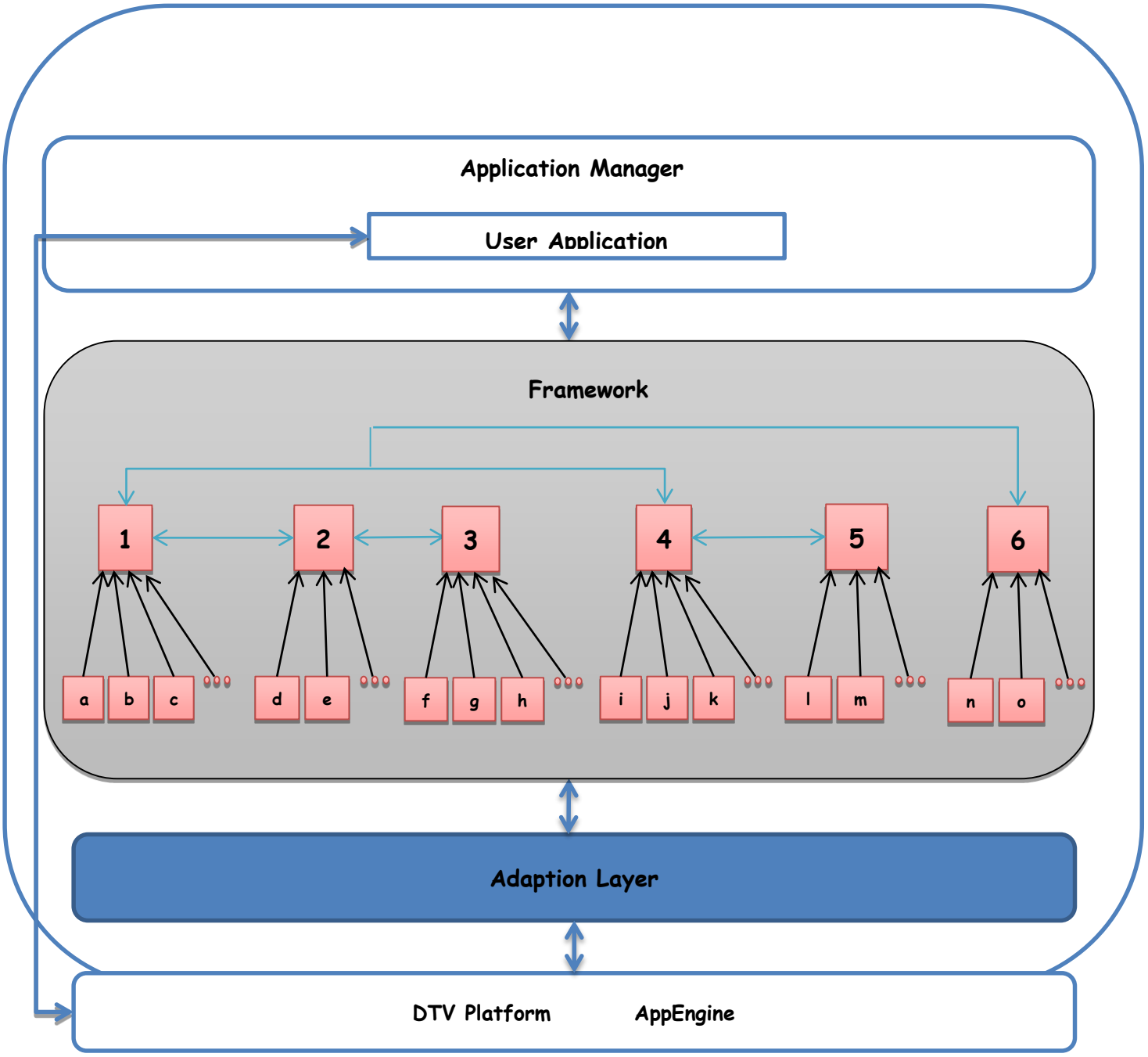
Figura 9: Trecho de Código do TVMonitor

Após essa descrição, fica evidenciada a necessidade de se ter uma arquitetura bem definida para auxiliar os acessos, já que toda aplicação Smart TV que interaja com programas televisivos necessitará de acesso aos serviços de comunicação e sessão. Na seção 4.3.3 é apresentada a arquitetura do *framework* proposto com foco em detalhar os seus componentes de reuso.

4.3.3 Projeto Arquitetural

Levando-se em consideração os requisitos funcionais levantados, foi proposto um *framework* que provê funcionalidades para que aplicações, que interajam com o conteúdo audiovisual dos programas da TV, possam ser construídas. O foco dessa arquitetura é fornecer um arcabouço para que desenvolvedores de aplicações Smart TV possam construir aplicações sincronizadas com a programação televisiva com facilidades. As tarefas complexas, que se referem à construção de aplicações desse domínio, foram detectadas e encapsuladas na forma de módulos reutilizáveis. Dessa forma, o desenvolvedor pode trabalhar em camadas de alto nível, preocupando-se mais com a lógica do negócio do que com detalhes de implementação de baixo nível.

Na Figura 10 é apresentada uma arquitetura para a construção de aplicações Smart TV em geral e principalmente aquelas integradas a programação televisiva. Em uma vista rápida essa arquitetura é baseada na arquitetura AppFramework (Samsung, 2014), onde as camadas "Framework" e "Adaption Layer" foram adicionados. O SyncSmartv integra um conjunto de ferramentas e componentes que permitem um engenheiro de software projetar, desenvolver e implantar rapidamente uma nova aplicação Smart TV integrada. A principal funcionalidade SyncSmartv é oferecida usando sua API. Seguindo a especificação dessa API, um desenvolvedor pode criar diversas aplicações, tais como aplicações de monitoramento que detectam cena do crime ou sexo e executam ações pré-definidas pelo usuário correspondentes a esse tipo de notificação.


Application Manager
User Application
Framework
1
2
3
4
5
6
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
Adaption Layer
DTV Platform
AppEngine

Legenda:

1: Sync Event Manager

a: MPEG-TS b: Audio Matching c: Serviço de Terçeiros

2: Communication Manager

d: Bluetooth e: Wifi

3: User Device Discovery

f: UPnP g: Url h: Zeroconf

4: TV Control Manager

i: TV Channel Change j: Mute k: Volume Control

5: Input Manager

l: Register Keys m: Unregister Keys

6: TV Channel ID

n: Image Processing o: Audio Processing

Figura 10: Arquitetura do Framework

A figura consiste em 4 camadas, a primeira chamada de *Application Manager* é onde as aplicações são construídas, usando linguagens como JAVASCRIPT, HTML, CSS entre outras, e são gerenciadas. Normalmente a aplicação construída é enviada diretamente para *AppEngine* da camada *DTV Platform* para sua execução. Caso o desenvolvedor queira construir uma aplicação integrada e sincronizada com o programa da TV com facilidades, ele pode aproveitar dos recursos disponibilizados na camada do *framework*. Em seguida, a aplicação construída através do *framework* utiliza-se da camada *Adaption Layer* (chamada de camada de conectores ou camada de porta) para a implementação de artefatos genéricos especificados nos módulos do *framework* de acordo com as especificidades de cada plataforma Smart TV sendo usada para execução da aplicação. Por fim, a aplicação é executada na camada *DTV Platform*.

A Figura 10 fornece uma visão geral da arquitetura proposta que atende aos requisitos funcionais elencados na seção 4.3.1.

4.3.3.1 Módulos

A arquitetura do *framework* proposto tem seus módulos listados a seguir na ordem em que aparecem na figura. Esta subseção traz uma descrição de alto nível desses módulos reutilizáveis. Questões técnicas mais aprofundadas de cada módulo serão descritas na especificação do *framework*.

- **Sync Event Manager**

O módulo *Sync Event Manager* provê uma interface para que aplicações Smart TV possam receber notificações sobre eventos ocorrendo na programação televisiva. Os sinais de sincronismo recebidos podem ser tanto relacionados ao estado do programa da TV, como ao início, término e identificação de comercial, entre outros. O registro da aplicação ao serviço de sincronização e o seu cancelamento do serviço são feitos pelo módulo *Sync Event Manager* que orquestra também o tratamento dos sinais de sincronismo recebidos de forma transparente. O *Sync Event Manager* pode conter diferentes artefatos que possibilitam a geração de sinais de sincronismo como MPEG-TS, Audio Matching e Serviço de Terceiros, explicados na primeira seção do capítulo 3, entre outros.

O protocolo de comunicação entre aplicações Smart TV e serviços de sincronização de terceiro implementado nesse módulo permite acesso às diversas funcionalidades de notificações. Entre elas, encontram-se aquelas relacionadas ao ambiente da TV Social, como por exemplo, notificações sobre configurações (*qual canal sintonizou?, qual volume?, etc.*) atuais de um grupo de usuários ao assistir televisão. Com base nessas informações, grupos específicos de usuários podem interagir sobre programas da TV de suas preferências. Isso promoveria a TV Social.

- **Communication Manager**

Ao criar-se aplicações para o ambiente da TV Digital Interativa, onde existem facilidades tecnológicas e meios para TVs interagirem com outras mídias, é necessário

levar em consideração a disponibilidade de dispositivos móveis inteligentes presentes no ambiente da TV. Esse fato cria uma experiência em que os usuários podem interagir com a TV através desses dispositivos. Esse módulo provê recursos necessários para que se estabeleça uma comunicação *two-way* entre aplicações da Smart TV e dispositivos móveis (*Smartphones, tablets*, entre outros) encontrados no mesmo ambiente. Para que seja possível essa comunicação, artefatos como *Bluetooth, Wifi*, entre outros, podem ser usados.

- **User Device Discovery**

Esse módulo oferece mecanismos necessários para que dispositivos conectados ao ambiente da TV possam descobrir a existência de um serviço da TV disponível para uso, fazer registro e depois usar essas informações para estabelecer a comunicação. A descoberta de serviços da TV por dispositivos móveis pode acontecer de diferentes formas: o serviço da TV pode enviar uma mensagem *broadcast* para todos dispositivos conectados no ambiente sobre sua disponibilidade, ou fazer isso de forma explícita, onde é disponibilizada uma *url* para viabilizar o uso do serviço. A *url* disponibilizada pode ser acessada pelo *browser* do dispositivo móvel, por exemplo. Para implementar essa solução de descoberta por *url* foi implementado um serviço web in *Grails* no qual aplicações Smart TV integradas registram seus *url's* (IP/porta) de requisição/de acesso para o armazenamento no servidor. Dessa forma, um dispositivo móvel com aplicação **MobileSync** instalada, ao executar aplicação faz uma requisição para esse serviço web solicitando uma lista de serviços (*url's* de requisição) de TV disponível no seu ambiente. Com o conhecimento desses *url's*, o dispositivo móvel consegue se conectar à TV.

A descoberta pode ser feita pelo *UPnP discovery protocol* conhecido como SSDP (*Simple Service Discovery Protocol*) (UPNP TEAM, 2014). UPnP (*Universal Plug and Play*) é um conjunto de protocolos de rede para conexão *peer-to-peer* entre dispositivos como computadores, TVs, impressoras, dispositivos móveis, entre outros e permite a descoberta de serviços disponíveis para que esses dispositivos possam compartilhar

dados, se comunicarem dentro de uma mesma rede, como por exemplo, a rede doméstica. SSDP é um protocolo de descoberta de serviços dentro de uma rede UPnP.

Pode-se também usar o Zeroconf (*Zero Configuration Networking*) (http://www.google.com/patents/US20020003780) para a descoberta de serviços. Zeroconf é um conjunto de técnicas que criam de forma automática uma rede IP sem necessidade de configuração ou servidores. Isso permite que usuários conectem seus dispositivos e aguardem que a conexão seja estabelecida. Zeroconf possui um protocolo de descoberta de serviços chamado de DNS-SD (*Domain Name System-Service Discovery*) que é um protocolo construído com base no DNS (Sistema de gerenciamento de nomes de domínio que visa transformar nomes de domínio em endereços IP).

- **TV Control Manager**

Ao ocorrer eventos como recepção de sinais de sincronismo, sensores do ambiente, por exemplo, aplicações Smart TV podem precisar tomar algumas decisões como mudar de canal, diminuir ou aumentar o volume da TV, entre outras. Para que isso seja possível, é preciso que aplicações acessem funções nativas da plataforma da TV para executar essas tarefas. Considerando que o usuário, ao assistir um canal, realiza registro pela aplicação para que ele seja levado para um determinado canal assim que começar um comercial, é desejável que ao receber o sinal de sincronismo sobre o início do intervalo comercial, a aplicação execute essa tarefa de forma transparente para o usuário. Esse módulo provê funcionalidades para que aplicações Smart TV atuem sobre controles da TV.

- **Input Manager**

Aplicações *ticker* necessitam de mecanismos para gerenciar o compartilhamento do uso de teclas do controle remoto com a TV. Em outras palavras, é desejável que o *Input Manager* seja capaz de multiplexar funções diferentes para uma mesma tecla, dependendo do contexto de utilização (TV ou Aplicação *ticker*) que o usuário deseja no momento.

É natural que quando uma aplicação *ticker* seja escondida (colocada em *background*), todas as teclas do controle remoto previamente registradas para o uso na aplicação percam seus registros automaticamente, ou seja, não são mais reconhecidas pela aplicação *ticker*. Devido a essa natureza, é desejável que o *Input Manager* ofereça funcionalidades de registro de teclas assim que a aplicação *ticker* for inicializada e quando ela for exibida de novo depois de ter sido colocada em *background*.

- **TV Channel ID**

O monitoramento de canais da TV requer o reconhecimento, de forma transparente, do canal sendo assistido pelo usuário no momento. Considere, por exemplo, que o usuário, ao assistir um determinado canal, queira registrar esse canal no serviço de monitoramento, para receber notificações relacionadas a ele enquanto estiver em outros canais. Para isso, é preciso obter o id (*identification*) do canal da TV e, por meio da interface oferecida por *Sync Event Manager,* registrá-lo no serviço de monitoramento.

A detecção do id (*identification*) do canal sendo assistido pode ser feita de diversas maneiras: diretamente na Smart TV, através das API's do módulo TVChannel (somente para canais abertos com sinal de TV Digital), usando o protocolo HDMI, no caso de TV por assinatura ou por serviço externo de terceiros que monitora canais de TV.

A detecção do canal da TV pelo serviço de terceiros pode ser feita de duas formas: a aplicação Smart TV captura a imagem da tela em intervalo de tempo pré-definido, submetendo-as para o serviço de terceiros para efetuar o processamento digital com finalidade de identificar sequência de frames compatível com amostra dos canais nos servidores. Quando a compatibilidade é detectada, o canal é identificado e uma resposta contendo o id do canal é enviada de volta para a aplicação Smart TV. O mesmo processo pode acontecer com o áudio, onde cada fala é capturada e enviada para o serviço para ser processada.

4.3.3.2 Relacionamentos entre Módulos

Os relacionamentos entre os módulos da figura 10 indicam as dependências existentes entre eles, isto é, um módulo pode depender de outro para sua execução e vice-versa. A seguir, são apresentados alguns exemplos dessas dependências.

- *Sync Event Manager* e *TV Channel ID*

  Depois de *TV Channel ID* obter o id (*identification*) do canal da TV, é precioso registrá-lo no serviço de monitoramento. Esse registro é feito por meio da interface oferecida por *Sync Event Manager*.
- *TV Control Manager* e *Input Manager*

  Depois de *Input Manager* registrar as teclas de controle remoto a serem utilizadas na aplicação, aquelas teclas escolhidas para oferecer tarefas como mudança de canal, alteração de volume, entre outros precisam ser registradas no *TV Control Manager* para poder usufruir das funcionalidades disponíveis nele e executar tais tarefas.
- *Sync Event Manager* e *TV Control Manager*

  Algumas funcionalidades como mudança de canal ou *mute* podem ser relacionadas a sinais de sincronismo. Ou seja, o usuário pode configurar pela aplicação que ao receber sinal de sincronismo associado ao início de comercial que ele seja levado para um determinado canal. Para isso, é preciso que os dois módulos *Sync Event Manager* e *TV Control Manager* se comuniquem.
- *User Device Discovery* e *Communication Manager*

  A comunicação entre Smart TV e dispositivos móveis inteligentes encontrados no ambiente da TV é estabelecida somente depois de *User Device Discovery* ter descoberta a presença de cada dispositivo

no ambiente e informações sobre serviço da TV disponível para o uso.

- *Sync Event Manager* e *Communication Manager*

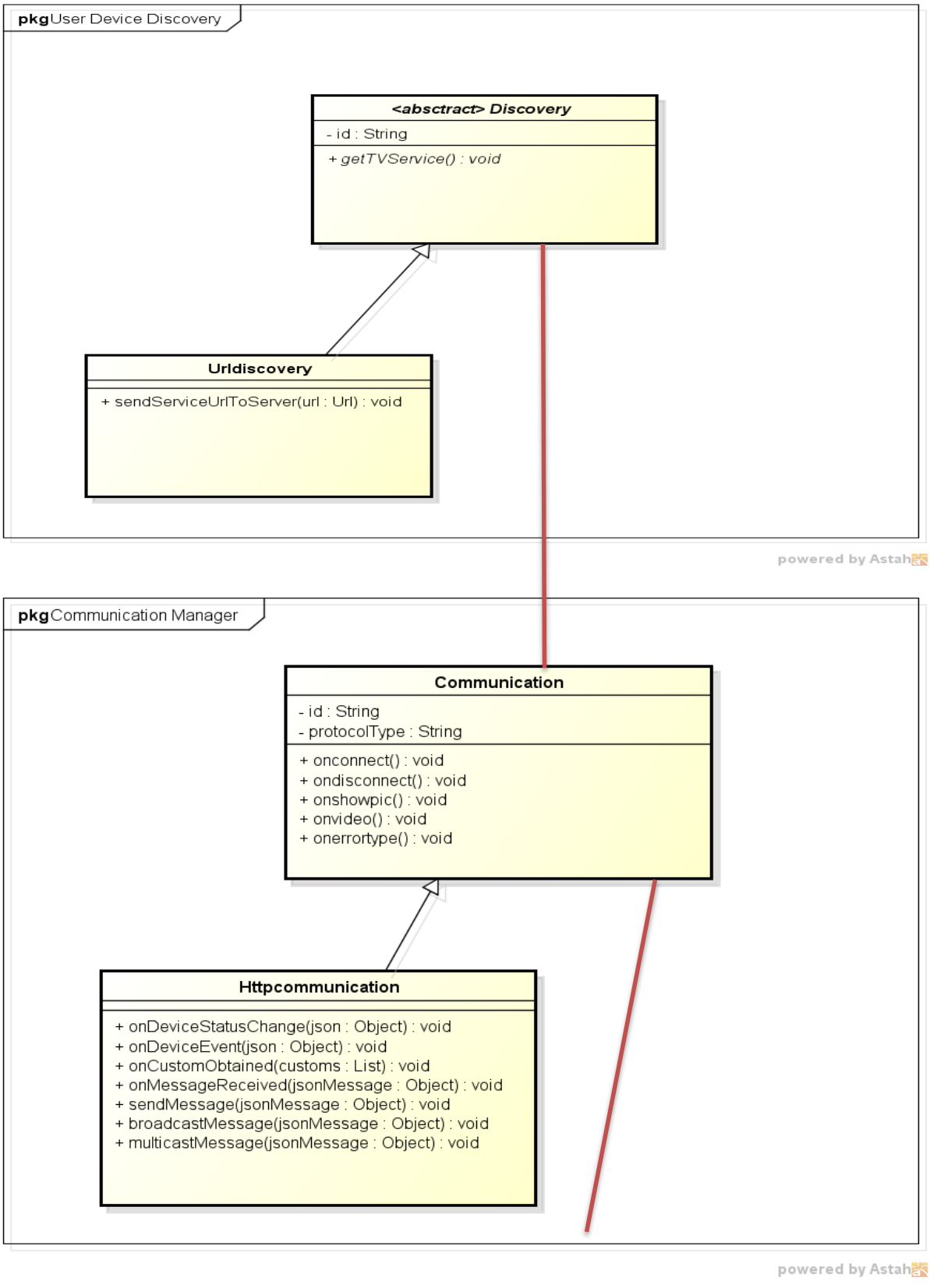
  Existem casos onde o dispositivo móvel pode usar a tela da Smart TV como "segunda tela". Em outras palavras, ao chegar um sinal de sincronismo relacionado ao início de comercial do *Sync Event Manager*, esse sinal é propagado para o dispositivo móvel, por meio do *Communication Manager*, avisando que a tela da Smart TV está liberada para uso. Dessa forma, o usuário pode fazer na Smart TV o *playback* de vídeos/músicas encontrados no seu celular ou em um servidor na nuvem, por exemplo.

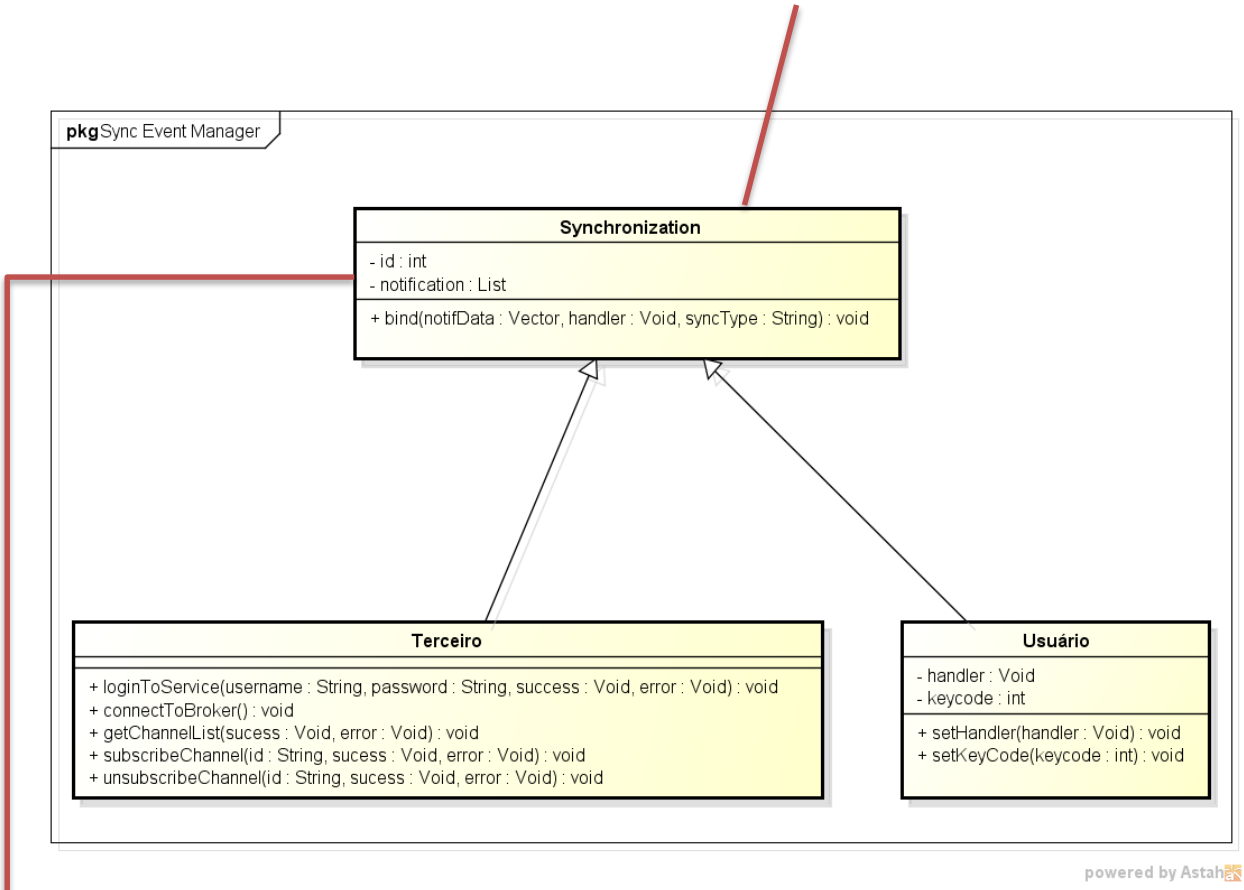
4.3.4 Projeto do *framework*

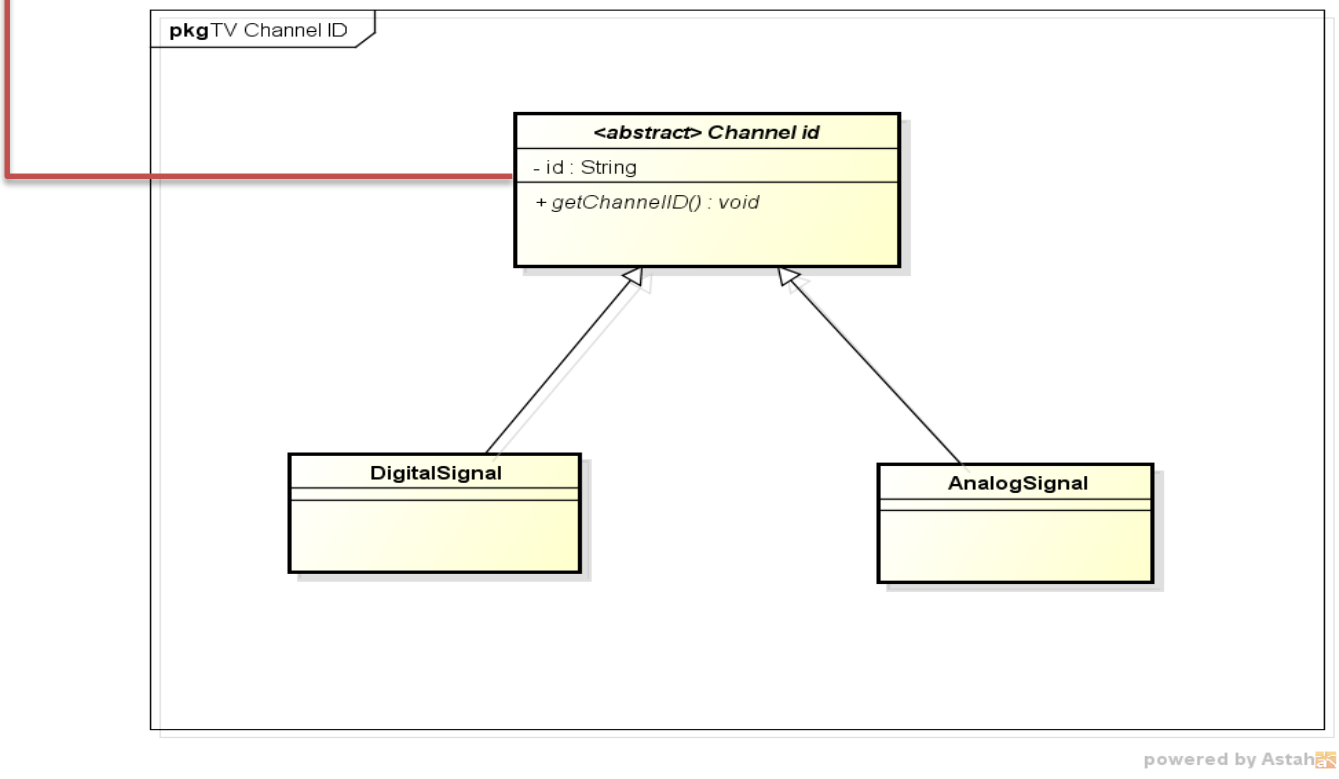
O objetivo desta atividade foi construir o diagrama de classes do *framework* SyncSmartv. O diagrama foi construído de modo a prevenir a inserção, no futuro, de novas classes correspondentes a novas funcionalidades, novos serviços de sincronização, novas formas de descoberta de serviços da TV, novas plataformas de TV, etc.

O ponto de partida desta atividade foi uma análise detalhada nos códigos-fonte das aplicações descritas na seção de 4.3.2 e seus diagramas de casos de uso.

Vale ressaltar que após essa análise detalhada nos códigos-fontes e documentos das aplicações do conjunto ferramental de apoio à construção do SyncSmartv, chegou-se às seguintes classes e seus relacionamentos descritos detalhadamente na seção 4.3.3.1.


pkgUser Device Discovery
<absctract> Discovery
- id : String
+ getTVService() : void
UrlDiscovery
+ sendServiceUrlToServer(url : Url) : void
powered by Astah
pkgCommunication Manager
Communication
- id : String
- protocolType : String
+ onconnect() : void
+ ondisconnect() : void
+ onshowpic() : void
+ onvideo() : void
+ onerrortype() : void
Httpcommunication
+ onDeviceStatusChange(json : Object) : void
+ onDeviceEvent(json : Object) : void
+ onCustomObtained(customs : List) : void
+ onMessageReceived(jsonMessage : Object) : void
+ sendMessage(jsonMessage : Object) : void
+ broadcastMessage(jsonMessage : Object) : void
+ multicastMessage(jsonMessage : Object) : void
powered by Astah


pkgSync Event Manager
Synchronization
- id : int
- notification : List
+ bind(notifData : Vector, handler : Void, syncType : String) : void
Terceiro
+ loginToService(username : String, password : String, success : Void, error : Void) : void
+ connectToBroker() : void
+ getChannelList(sucess : Void, error : Void) : void
+ subscribeChannel(id : String, sucess : Void, error : Void) : void
+ unsubscribeChannel(id : String, sucess : Void, error : Void) : void
Usuário
- handler : Void
- keycode : int
+ setHandler(handler : Void) : void
+ setKeyCode(keycode : int) : void
powered by Astah


pkgTV Channel ID
<abstract> Channel id
- id : String
+ getChannelID() : void
DigitalSignal
AnalogSignal
powered by Astah

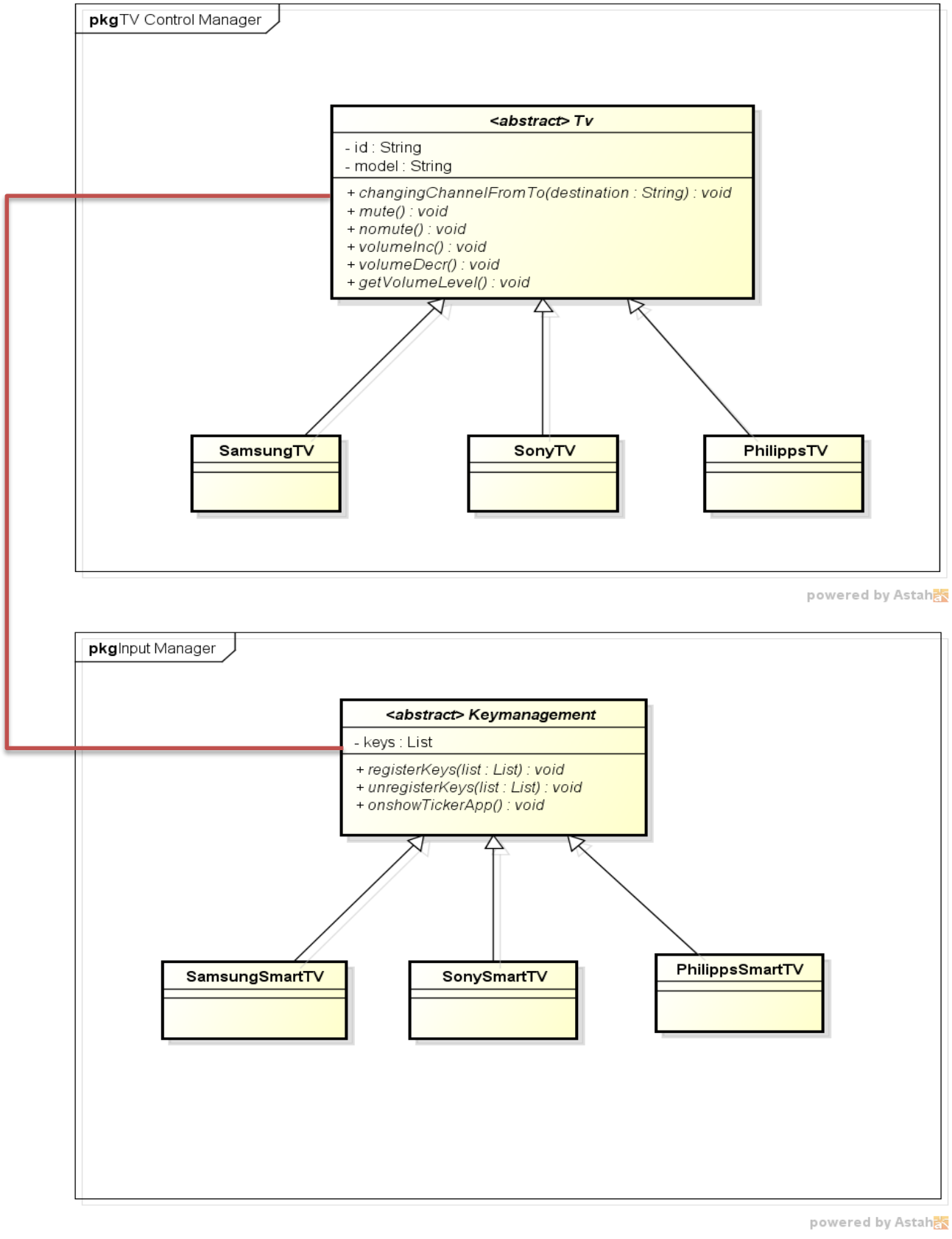
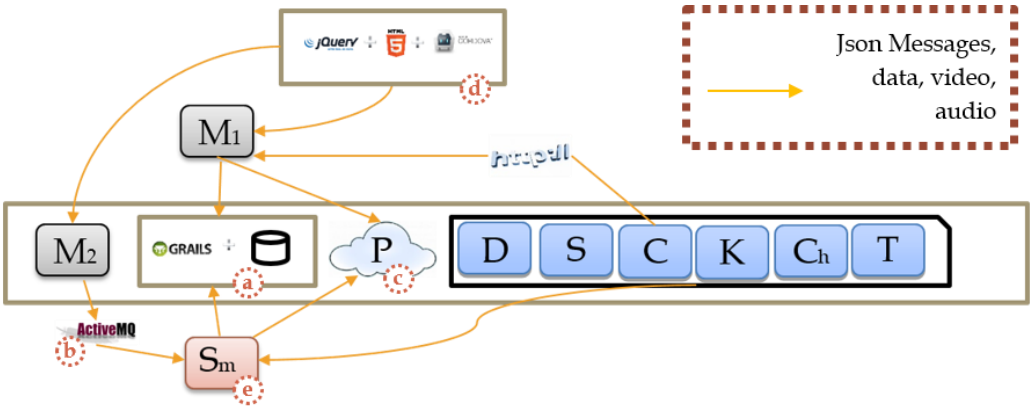


Figura 11: Diagrama de Classes do *Framework*

4.3.5 Implementação do *framework*

Com base no diagrama de classes apresentado na seção anterior iniciou-se a implementação do *framework* e de demais artefatos necessários para o seu funcionamento. Durante esse processo foram desenvolvidos os componentes de software presentes na figura 11 de forma a atender os requisitos funcionais e não funcionais listados na seção 4.3.1.



M1: First Mobile Application (MobileSync)
M2: Second Mobile Application (AdminSync)
Sm: Smart TV Application (TVMonitor)
P: PHP Server
D: Discovery
S: Synchronization
C: Communication
K: Keymanagement
Ch: Channel identification
T: TV

Legenda:

M1: Mobile Application 1	**S:** Synchronization
M2: Mobile Application 2	**C:** Communication
Sm: Smart TV Application	**K:** Keymanagement
P: PHP Server	**Ch:** Channel identification
D: Discovery	**T:** TV

Figura 12: Arquitetura geral da implementação do SyncSmartv e de módulos externos necessários para viabilizar o seu funcionamento

A figura 12 apresenta uma ilustração da arquitetura geral da implementação do *framework* aqui proposto e de componentes externos necessários para o seu funcionamento. As principais classes implementadas do sistema são "D" (Discovery), "S" (Synchronization), "C" (Communication), "K" (Keymanagement), "Ch" (Channel id) e "T" (TV).

A classe "Discovery" permite a descoberta, pelo dispositivo móvel presente no ambiente, do serviço da TV disponível no momento. Os usuários munidos de seus celulares, por exemplo, poderão se conectar com a TV.

A classe "Synchronization" permite aplicações da TV receber notificações sobre eventos ocorrendo na TV. O serviço de notificações que implementa essa classe está descrito na seção de Notificação e embutido no *framework* proposto. Além disso, sua API está descrita no Anexo A. A classe "Synchronization" foi implementada de modo flexível com finalidade de adicionar novos serviços de sincronização, se necessário, sem implicar ajustes ou correções em outros módulos. Para o escopo deste trabalho, dois serviços de sincronização foram implementados e embutidos no *framework*.

A classe "Communication" permite a comunicação bidirecional entre a TV e os dispositivos móveis conectados a ela, por meio de troca de mensagens *(broadcast message, multicast message and unicast message)*. As mensagens trocadas, na maioria dos casos, são relacionadas aos estados correntes dos dispositivos, como por exemplo, *dispositivo X conectou/desconectou-se da TV, a TV está liberada para ser utilizada como segunda tela, etc.* Vale ressaltar que para possibilitar essa comunicação foi implementado um protocolo de comunicação descrito no Anexo A sobre descrição das API's do *framework*.

A classe "Keymanagement" permite a aplicações registrar, manipular e cancelar o registro das teclas do controle remoto utilizadas durante todo processo da sua execução. Como a maioria das aplicações Smart TV integradas e sincronizadas com a programação televisiva são do tipo *ticker,* faz-se necessário a existência desse módulo para orquestrar o uso das teclas do controle remoto entre a aplicação da TV e a própria TV.

A classe "Channel id" permite a aplicação saber sobre a *string* de identificação de um canal específico. A classe "Channel id" oferece essa funcionalidade tanto por

meio do processamento local como remoto. No processamento local, caso do sinal transmitido for digital, acessa-se o módulo de canais da TV para recuperar essa informação.

Por fim, a classe "Tv" oferece funcionalidades para que aplicações possam executar tarefas como mudança automática de um canal para outro, alterar volumes da TV, etc.

Back-end

O *back-end* é necessário para auxiliar nas tarefas de armazenamento de informações, de comunicação entre diferentes aplicações distribuídas. O *back-end* deste trabalho consistiu em serviços web para descoberta de serviços da TV e gerenciamento de usuários da TV social, de um *broker* de mensagens e de um servidor de mídias.

Para a implementação do *back-end* do *framework* aqui proposto, foram utilizadas as seguintes tecnologias:

- Grails: o desenvolvimento com esse *framework* permitiu criar com facilidades serviços especializados como o de descoberta de serviços da TV ("Discovery"), por exemplo.
- ActiveMQ: é um *framework* da Apache que implementa um *broker* de mensagens utilizando o protocolo STOMP. Esse *broker* de mensagens foi utilizado para auxiliar a troca de mensagens entre aplicações Smart TV e serviços de monitoramento de canais da TV.
- PhpServer ("P"): é um servidor de mídia php implementado para auxiliar a troca de conteúdo de imagens entre dispositivos móveis e a TV.

Front-end

O *front-end* é a base para as aplicações executadas nos dispositivos móveis dos telespectadores para interação com a TV, da aplicação móvel para simular a geração de sinais de sincronização do serviço de terceiros e da aplicação Smart TV.

Os módulos utilizados no *front-end* são:

- Apache Cordova: é um *framework* que permite a criação de aplicativos em HTML 5, Javascript e CSS que funcionem em diversos modelos e sistemas operacionais dos dispositivos móveis encontrados hoje em dia. Com isso, as aplicações foram construídas independentemente das plataformas e depois foram geradas (*build*) versões específicas de plataformas.
- jQuery Mobile: um *framework* para o desenvolvimento de aplicativos mobiles e web responsivos, ou seja, cujas interfaces se auto adaptam de acordo com cada dispositivo de *display*.
- Samsung Smart TV Platform: uma plataforma para o desenvolvimento de aplicações Smart TV usando HTML 5, Javascript e CSS.

A atividade de teste do *framework* que visa averiguar se todas funcionalidades planejadas na seção 4.3 foram corretamente implementadas e avaliar a sua usabilidade, está descrita no próximo capítulo.

4.3.6 Documentação do *framework*

Como em qualquer *framework* a documentação é um item primordial para viabilizar sua utilização, neste trabalho, a forma usada para sua elaboração foi o *JSDoc*, que se encontra disponibilizado no Anexo A e pode ser consultado no endereço eletrônico http://lince.dc.ufscar.br/~cedrick/teste/jsdoc_toolkit-2.4.0/jsdoc-toolkit/out/jsdoc/index.html.

Capítulo 5

AVALIAÇÃO DO FRAMEWORK

Sempre que surge a necessidade de se obter melhorias na produtividade e qualidade em uma empresa ou instituição, pensa-se, entre outras medidas, na adoção de processos de reuso para construção dos produtos de software. Para isso é necessário avaliar a evolução desses processos e identificar as estratégias de reuso mais eficientes. A avaliação é um processo altamente crítico para a melhoria da qualidade do processo de produção do software (ARES et al., 1998). É através dela que se pode averiguar a efetividade de novas ferramentas, métodos, processos e estratégias de desenvolvimento de software. Se nenhuma avaliação for realizada sobre as novas tecnologias adotadas, corre-se o risco de tornar o processo de desenvolvimento de software ineficiente e gerar impactos negativos no software desenvolvido. Modelos e métricas para avaliação de reuso servem a esse propósito (FRAKES; TERRY, 2006).

Para validar o framework proposto foi utilizado o método experimental proposto por Wohlin et al. (2000). Foram avaliados a qualidade e o reuso realizado na construção de aplicações Smart TV integradas com os programas televisivos e construídas por meio do framework proposto. Este capítulo apresenta como foi feita a validação do framework proposto e construído durante este trabalho de mestrado.

Para avaliar o framework foi feito um estudo comparativo entre os processos de desenvolvimento de duas versões de uma mesma aplicação, uma construída com a abordagem de reuso usando o framework proposto e outra construída sem a abordagem.

A aplicação **TVMonitor**, a mais completa desenvolvida pelo autor desta dissertação, possui uma versão não componentizada. Dessa forma, pode-se comparar essa versão com outra construída usando-se o *framework* proposto. Depois de avaliar o impacto do processo do reuso na eficiência das equipes de desenvolvedores, utilizando a metodologia experimental de Wohlin et al.(2000), uma nova avaliação com usuários finais foi feita, com finalidade de avaliar dois parâmetros: a percepção sobre utilidade ("*Perceived Usefulness"*) e a percepção sobre facilidade de uso ("*Perceived Ease of Use"*).

Na próxima seção apresenta-se uma introdução sobre os princípios experimentais da Engenharia de Software que serviram de base para avaliar o *framework*.

5.1 Experimentação do Framework Proposto

Em Engenharia de Software os experimentos são realizados em laboratórios, sob condições controladas. A sua aplicação na avaliação de atividades dos seres humanos envolvidas no processo de criação de software garante uma avaliação sistemática, disciplinada e controlada. O objetivo da experimentação é manipular um conjunto de variáveis associadas a um determinado objeto em estudo enquanto mantêm-se outras variáveis intactas. O efeito dessa manipulação, observado nos resultados do experimento, é medido, e, com base nisso, análises são feitas para validar ou refutar as hipóteses formuladas (WOHLIN et al., 2000).

5.1.1 Princípios da Experimentação

Wohlin et al. (2000) afirmam que os experimentos são apropriados para validar teorias, confirmar o conhecimento convencional, explorar relacionamentos, avaliar a predição de modelos ou validar as medidas associadas ao desenvolvimento de

software. Como vantagens da experimentação apresentam-se o nível de controle obtido com relação aos participantes do experimento, objetos, dentre outros e a possibilidade de desempenhar análises estatísticas para teste de hipóteses.

Normalmente um experimento é formulado através das hipóteses. Essas últimas formalizam as ideias de prováveis relacionamentos de causa e efeito que se deseja investigar em um determinado objeto em estudo.

Cria-se o experimento para testar a teoria ou as hipóteses formuladas através da observação dos resultados obtidos. Ao final de um experimento que foi bem planejado, deve-se ser capaz de tirar conclusões a respeito do relacionamento de causa e efeito para o qual as hipóteses foram formuladas. Os objetivos do experimento podem estar relacionados com *melhoria* ("é possível melhorar o fenômeno?"), *avaliação* ("quão bom é isto?"), *caracterização* ("o que está acontecendo?"), *previsão* ("é possível levantar estimativas?") ou *controle* ("é possível manipular o fenômeno?") do objeto estudado (WOHLIN et al., 2000).

Uma hipótese é uma teoria ou suposição que pode explicar um determinado comportamento de interesse de uma pesquisa. Podemos dizer que um experimento tem como objetivo colher dados, em um ambiente controlado, para validar ou refutar uma hipótese.

Existem dois tipos de hipóteses em um experimento, hipótese principal chamada de hipótese nula (representada por H_0) e as hipóteses alternativas (representadas por Ha, H_1, H_2, etc). A hipótese nula é a principal de um experimento e é baseada na ideia de que não existe um relacionamento estatisticamente significante entre a causa e efeito. Geralmente quando se conduz um experimento o objetivo principal é rejeitar a hipótese nula em favor de uma ou mais hipóteses alternativas. A decisão de rejeição deve depender da averiguação dos resultados obtidos durante o experimento.

Existem dois tipos de variáveis em um experimento, dependentes e independentes. As variáveis dependentes (ou variáveis de resposta) são aquelas sob análise. As variáveis independentes (ou variáveis de entrada) são aquelas manipuladas e controladas durante o experimento. Por exemplo, num estudo para

avaliar o efeito do emprego de um novo método de desenvolvimento na produtividade dos desenvolvedores de software de uma empresa, a variável dependente seria a produtividade. Já as variáveis independentes poderiam ser, por exemplo, a experiência da equipe, o próprio método de desenvolvimento, as ferramentas de suporte, entre outras.

Fator é o nome atribuído a toda variável independente manipulada durante um experimento. Enquanto os fatores (variáveis independentes) apresentam a causa que afeta o resultado do processo de experimentação, as variáveis dependentes apresentam o efeito que é causado pelos fatores do experimento. O valor atribuído a um fator chama-se "tratamento" e o valor de uma variável dependente se chama "resultado". No caso do exemplo de estudo anterior, o fator seria o método de desenvolvimento, pois pretende-se estudar o efeito do emprego de um novo método de desenvolvimento na produtividade. Esse fator poderia assumir dois tratamentos: o método baseado no *framework* e método baseado na LPS.

Durante a execução do experimento os tratamentos são aplicados ao conjunto de objetos e participantes. Os participantes fazem parte de amostra de pessoas tiradas da população de interesse para conduzir o experimento. Um objeto pode ser, por exemplo, uma aplicação que deverá ser construída com diferentes métodos de desenvolvimento. No exemplo citado anteriormente, os programas a serem desenvolvidos com o uso do *framework* e LPS são os objetos, e os participantes são os integrantes da equipe, ou seja, os desenvolvedores.

5.1.2 Fases de um Experimento

O processo de execução de um experimento presume a realização de diferentes atividades. A literatura apresenta quatro fases principais que sempre estão presentes num processo de experimentação em Engenharia de Software:

1. **Definição:** é a fase onde o experimento é expresso em termos de problemas e objetivos.
2. **Planejamento:** é a fase de elaboração do experimento. No planejamento realiza-se a definição do contexto, a formulação das hipóteses, a seleção

das variáveis, a seleção dos participantes, entre outros elementos do experimento.

3. **Execução:** fase subdividida em três: preparação, execução e validação dos dados coletados. Na preparação, os participantes do experimento são preparados e o material necessário que compõe a instrumentação é elaborado. Os participantes são informados a respeito da intenção do experimento e devem dar o seu consentimento em participar com comprometimento nas atividades do experimento. Na execução, o experimento deve ser conduzido e a coleta de dados deve ser realizada. Finalmente, na etapa de validação, deve-se garantir que os dados coletados estão corretos e fornecem uma visão válida dentro do escopo do experimento.

4. **Análise e Interpretação dos Resultados:** é a fase onde é realizada a verificação das hipóteses formuladas na fase do planejamento. Para isso, uma interpretação correta deve ser feita sobre os resultados obtidos utilizando métodos estatísticos apropriados para testar as hipóteses. Finalmente hipóteses podem ser refutadas ou validadas e conclusões do experimento devem ser tiradas.

5.1.3 Desenvolvimento da experimentação do *framework*

A experimentação do *framework* neste trabalho seguiu as fases experimentais, conforme proposto por Wohlin et al. (2000). O experimento consistiu de um estudo comparativo entre os processos de desenvolvimento de duas versões da aplicação TVMonitor, uma construída com a abordagem de reuso usando o *framework* proposto e outra construída sem a abordagem.

A seguir, são apresentadas as fases de definição, planejamento, execução e análise dos resultados do experimento do *framework*.

5.1.3.1 Definição do Experimento

O objetivo do estudo é:

- **Analisar** o uso do framework proposto na construção de aplicações Smart TV sincronizadas com a programação televisiva;
- **Com o propósito** de avaliação;
- **Com respeito à** eficiência em termos de tempo despendido e produtividade;
- **De ponto de vista** de desenvolvedores de software;
- **No contexto de** estudantes de graduação em Ciência e Engenharia da computação.

O número de participantes e objetos envolvidos em um estudo experimental pode caracterizar o contexto de um experimento. Neste experimento, consideram-se doze (12) desenvolvedores e um objeto (aplicação *TVMonitor*).

5.1.3.2 Planejamento do Experimento

O planejamento do experimento envolve as seguintes etapas:

a. **Seleção do contexto:** o experimento ocorreu em ambiente universitário, sendo realizado com estudantes de graduação e de pós-graduação do Departamento de Computação da Universidade Federal de São Carlos (UFSCar).
b. **Formulação das Hipóteses:** para este experimento três hipóteses foram elaboradas com relação ao efeito do método de desenvolvimento no resultado final.

   Para a formulação das três hipóteses, foram consideradas as seguintes métricas:

   τ - Tempo total gasto pela equipe para o desenvolvimento da aplicação Smart TV sincronizada com o programa da TV;

   Þ - Produtividade da equipe em termos de linhas de código produzidas (LOC) por unidade de tempo (Þ= LOC/τ);

μ_τ – Tempo médio despendido pelas equipes para o desenvolvimento da aplicação Smart TV sincronizada com o programa da TV;

μ_P – Produtividade média das equipes no desenvolvimento da aplicação Smart TV sincronizada com o programa da TV;

ε – Eficiência das equipes no desenvolvimento da aplicação Smart TV sincronizada com o programa da TV.

Temos uma hipótese nula e suas duas alternativas correspondentes:

Hipótese nula (H_0): Não há diferença entre equipes utilizando o *framework* proposto e equipes que não o utilizem para a construção da aplicação TVMonitor, com respeito à eficiência(ε) da equipe.

H_0: $\varepsilon_{framework} = \varepsilon_{semframework} \Rightarrow \mu_{\tau framework} = \mu_{\tau semframework}$ e $\mu_{Pframework} = \mu_{Psemframework}$

Hipótese alternativa (H_1): Equipes utilizando o *framework* proposto para a construção da aplicação TVMonitor são, em geral, mais eficientes do que equipes desenvolvendo sem o uso do *framework*.

H_1: $\varepsilon_{framework} > \varepsilon_{semframework} \Rightarrow \mu_{\tau framework} < \mu_{\tau semframework}$ e $\mu_{Pframework} > \mu_{Psemframework}$

Hipótese alternativa (H_2): Equipes utilizando a abordagem sem *o framework* para a construção da aplicação TVMonitor são, em geral, mais eficientes do que equipes desenvolvendo com o uso do *framework*.

H_2: $\varepsilon_{framework} < \varepsilon_{semframework} \Rightarrow \mu_{\tau framework} > \mu_{\tau semframework}$ e $\mu_{Pframework} < \mu_{Psemframework}$

c. **Seleção das variáveis:** a eficiência foi a variável dependente selecionada para este experimento. As independentes são: o método de desenvolvimento, a aplicação desenvolvida, o ambiente de desenvolvimento e as tecnologias de desenvolvimento. Dentre as quatro variáveis independentes, três foram mantidas constantes durante o estudo:

   - ✓ Aplicação = TVMonitor;
   - ✓ Ambiente de desenvolvimento = IDE Eclipse;
   - ✓ Tecnologias de desenvolvimento = HTML, CSS, JAVASCRIPT e JQUERY.

d. **Seleção dos participantes:** a seleção dos participantes do experimento foi feita selecionando os alunos de graduação e da pós-graduação do Departamento de Computação da UFSCar disponíveis e apresentando um conhecimento básico sobre as tecnologias de desenvolvimento escolhidas para a execução do experimento.

e. **Projeto do Experimento:** essa etapa descreve como os testes experimentais foram organizados e executados dividindo os participantes em grupos homogêneos, de modo que cada grupo possuísse em média o mesmo nível de experiência. Para isso, um formulário de perfil de participantes, que se encontra no Apêndice A, foi elaborado e utilizado para capturar a experiência de cada participante.

f. **Tipo do projeto:** O tipo do projeto desse experimento foi de um fator (método de desenvolvimento) com dois tratamentos (abordagem com uso do *framework* proposto e abordagem sem). Neste tipo de projeto o mesmo objeto (aplicação TVMonitor) a ser manipulado é utilizado nos dois tratamentos e os participantes foram escalados em cada tratamento de modo a formar duplas mesclando pessoas com mais e com menos expertise em cada uma das fases de desenvolvimento (projeto, implementação, teste).

g. **Instrumentação:** nesta etapa planejou-se os materiais necessários para apoiar os participantes no experimento, a apresentação dos objetos a serem manipulados e dos instrumentos para coleta de dados e medição.

5.1.3.3 Execução do Experimento

A execução do experimento passou por duas principais etapas:

(1) **Preparação:** nesta etapa foram efetivamente elaborados os materiais necessários para apoiar o experimento, a saber: os objetos a serem

manipulados, os documentos de guia, os instrumentos para coleta de dados e medição.

- **Objetos:** o módulo móvel da aplicação *TVMonitor,* responsável por realizar a descoberta do serviço da TV, conectar e comunicar-se com a TV para o compartilhamento de conteúdo de mídia de forma sincronizada com a programação da TV, foi planejado e implementado. Também foi implementado outro módulo móvel que trata de disponibilizar uma interface para simular a geração de sinais de sincronização providos por terceiros. Por fim, a aplicação parcial Smart TV contendo todos componentes gráficos necessários foi implementada para ser distribuída juntos com seus arquivos de imagens, CSS e JAVASCRIPT referentes à biblioteca Jquery utilizada ou à API's do *framework* proposto dependendo de qual grupo os participantes estão inseridos.
- **Diretrizes:** os seguintes documentos foram elaborados para troca de informações entre participantes e o experimentador durante o experimento: *(1) Formulário de caracterização dos participantes*, auxiliando na obtenção de informações referentes à experiência do participante sobre as tecnologias e assuntos envolvidos no experimento; *(2) Formulário de consentimento*, auxiliando na obtenção da aceitação e da ciência do participante dos objetivos do experimento e das condições de participação; *(3) Descrição da tarefa*, contendo instruções para auxiliar na execução propriamente dita do experimento; e *(4) Descrição da aplicação e material de apoio*, contendo o detalhamento da aplicação a ser implementada e um diagrama de casos de uso e sua descrição para apoiar a implementação das funcionalidades requeridas.
- **Instrumentos para coleta de dados:** o *Formulário de coleta de dados* foi elaborado para auxiliar os grupos no preenchimento de todas as

informações requeridas durante o desenvolvimento da aplicação – teste:

(2) Execução: os testes experimentais foram executados conforme o projeto experimental adotado no escopo deste trabalho. A tabela 1 traz todos dados coletados pelos grupos durante a execução do experimento.

Tabela 1: Dados Coletados

	Grupo	HInicioProj	HFimProj	HInicioImpl	HFimImpl	HInicioTeste	HFimTeste	LCG Auto	LCG Manual	LOC Total	Tempo Total(τ)	Produtividade(p)
Com Framework	G1	11:20	11:46	12:40	15:21	15:27	16:18	3270	52	2532	3:58	838
	G2	11:56	12:13	12:47	15:19	15:25	16:04	2826	61	2541	3:28	833
	G3	14:30	14:47	15:00	16:10	16:20	16:30	2480	56	2536	1:37	1569
	Média	00:20		2:08		00:33					3:01	1080
Sem Framework	G4	14:20	14:51	15:23	18:05	18:10	18:25	1882	361	2243	4:08	543
	G5	8:30	9:10	9:20	15:17	15:25	16:12	1882	427	2309	7:24	312
	G6	8:30	9:02	9:10	14:48	15:00	15:33	1882	392	2274	6:43	339
	Média	00:34		4:59		00:32					6:05	398

HInicioProj: Hora de Início do Projeto;
HFimProj: Hora de Fim do Projeto;
HInicioImpl: Hora de Início da Implementação;
HFimImpl: Hora de Fim da Implementação;
HInicioTeste: Hora de Início de Testes;
HFimTeste: Hora de Fim de Testes;
LCGAuto: Linhas de Código Geradas Automaticamente;
LCGManual: Linhas de Código Geradas Manualmente.

A organização dos dados mostrados na tabela 1 está feita conforme as duas abordagens de desenvolvimento aplicadas, desenvolvimento com e sem o uso do *framework*. A parte superior da tabela lista os dados dos grupos que utilizaram o *framework* proposto para a implementação da aplicação Smart TV *TVMonitor* e na parte inferior da tabela, são listados os dados dos grupos que implementaram a mesma aplicação sem o uso do *framework* proposto. As colunas referentes à *"LOC Total"*, *"Tempo Total"* e *"Produtividade"*, representam, respectivamente, o total de linhas de códigos implementados pelo grupo, o tempo total despendido e a produtividade de cada grupo no desenvolvimento da aplicação Smart *TVMonitor*. Essa produtividade é calculada em termos de linhas de código produzidas por unidade de tempo (por hora no contexto deste experimento).

As linhas referentes à "Média" mostram os tempos médios gastos pelos grupos na execução de cada atividade do desenvolvimento (projeto, implementação e testes). Também, são listadas as médias do tempo total gasto (μ_τ) e da produtividade total (μ_P).

Vale ressaltar que, como μ_P é a média aritmética das produtividades de cada grupo, seu valor pode diferir da produtividade média calculada em termos da média de linhas de código total e da média do tempo total gasto.

5.1.3.4 Análise e Interpretação dos Resultados

Ao fazer uma análise inicial sobre os dados coletados na tabela 1, foi possível inferir algumas conclusões interessantes. A figura 13, mostra o resumo dessas inferências iniciais. Vale notar que a distribuição de esforços dos grupos (com e sem o uso do *framework*) mantiveram-se quase iguais nas fases de *projeto e testes* do desenvolvimento da aplicação Smart TV TVMonitor. No entanto, observa-se uma grande discrepância de esforços dos grupos de desenvolvimento com uso do *framework* na fase da implementação. Enquanto os grupos de abordagem sem o uso do *framework* gastaram, em média, 4 horas e 59 minutos, os grupos de desenvolvimento com o uso do *framework* gastaram, 2 horas e 08 minutos (uma redução de 57, 2%).

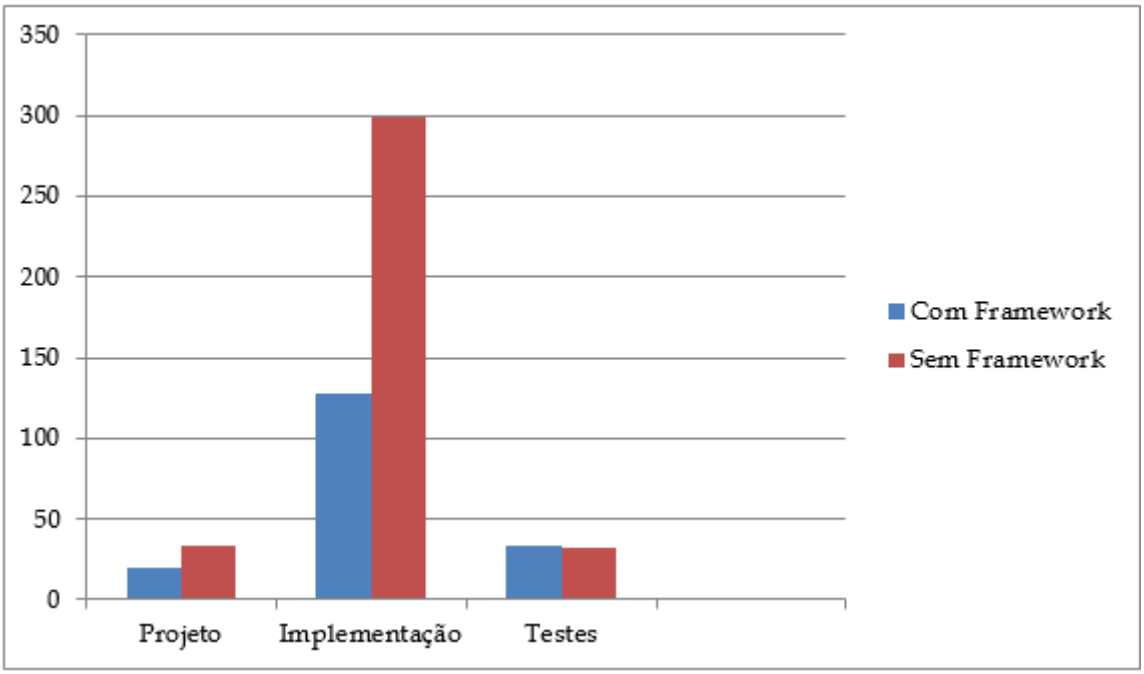


Figura 133: Tempo Médio Gasto (em minutos)

Teste das Hipóteses: o objetivo deste teste é verificar se uma variável difere entre duas amostras independentes, e com algum grau de significância, rejeitar a hipótese nula (H_0) em favor de alguma das hipóteses alternativas (H_1 ou H_2) com base

no conjunto de dados obtido. Para comprovar de forma estatística a redução de esforço, ou seja, o aumento da eficiência de grupos com uso de *framework* conforme mostrada na figura 13, aplicou-se o método chamado de *t-test ou t de student* (MONTGOMERY, 2000). O *t-test* é empregado para verificar a existência de uma diferença entre duas amostras independentes baseando-se nas médias aritméticas dos itens de dados contidos nas amostras envolvidas. As amostras utilizadas neste experimento contêm itens de dados referentes aos tempos totais gastos e produtividade total dos grupos tanto de desenvolvimento com quanto de desenvolvimento sem o *framework*.

Como o *t-test* procura pela diferença entre duas amostras independentes, baseando-se nas médias aritméticas de seus itens de dados considerando a dispersão ou variabilidade desses dados, a sua fórmula é dada pela equação (1). Essa equação é proporcional à diferença das médias de duas amostras e inversamente proporcional à variabilidade ou dispersão total das duas amostras. Essa dispersão é obtida através das variâncias (Sx^2 *e* Sy^2) de cada amostra e dos números de itens de dados contidos em cada amostra (*n* e *m*), conforme mostra a equação (2).

$$t_0 = \frac{\bar{x}-\bar{y}}{Sp\sqrt{\frac{1}{n}+\frac{1}{m}}} \qquad (1)$$

$$Sp = \sqrt{\frac{(n-1)Sx^2+(m-1)Sy^2}{n+m-2}} \quad (2)$$

Após o cálculo de t_0, deve-se checar o valor de t padrão na tabela padrão de distribuição estatística *t de student* para averiguar se é suficientemente grande para poder inferir sobre a hipótese nula baseando-se em algum grau de significância. Portanto, deve-se estabelecer esse grau de significância que representa o "*ponto de corte*" ou o "*nível de risco*" que pode ser assumido em rejeitar a hipótese nula quando há existência de uma diferença entre amostras. Por exemplo, tomando α= 0,02, assume-se um risco de 2% em se encontrar uma diferença entre as médias das amostras mesmo que essa diferença fosse um *falso positivo*. Nesse caso, pode-se dizer que o teste possui 98% de nível de confiança. Também, é preciso calcular o grau de liberdade (gl = n+m-

2), que é usado para determinar a dispersão dos dados das amostras usada no cálculo do Sp e é igual à soma do número de itens de dados nas duas amostras menos 2.

Depois de ter calculado t_o, α e gl, pode-se checar o valor do t padrão na tabela padrão de distribuição estatística *t de student* para averiguar se t_0 é suficientemente significativo. Neste momento, pode-se fazer a seguinte comparação e tomar decisões:

Se $|t_0| > t$ padrão $= t_{\alpha/2, gl} \rightarrow$ **REJEITAR H_0**,

Caso contrário, → **H_0 NÃO É REJEITADA** e nenhuma conclusão é tirada do experimento.

Conforme a figura 14, o valor de t limita inferiormente o valor de t_0, ou seja, o valor calculado do t_0 deve pertencer a]–t , +t[que consiste no intervalo de confiança do teste.

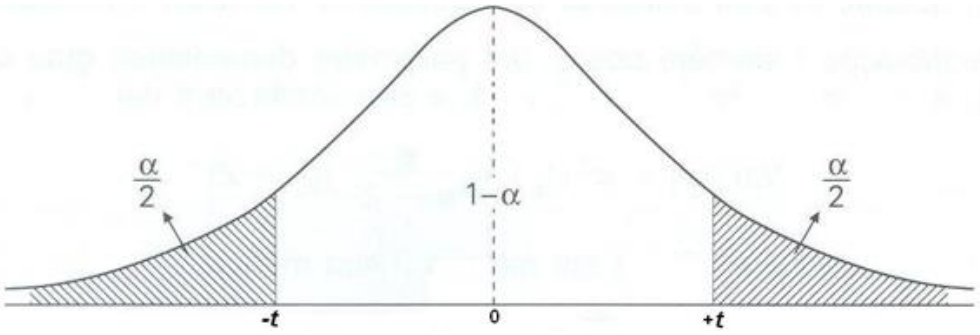


Figura 144: Distribuição t-test ***[Fonte:online Statistic, 2014]***

α : Nível de significância (risco de rejeitar H_0 em caso de um *falso negativo*) **-> quanto menor, maior o nível de confiança do teste.**

Como a variável dependente do experimento (eficiência das equipes) possui dois tratamentos, tempo total (τ) e produtividade (Þ), a aplicação *do t-test* foi realizada em duas etapas como segue abaixo:

(1) Primeira etapa do *t-test* (Tempo Total)

Tabela 2: Dados de análise do t-test dos grupos (Com/Sem Framework-Tempo Total)

Com Framework	X	X - Media	(X - Media)²
G1	3,97	0,95	0,9025
G2	3,47	0,45	0,2025
G3	1,62	-1,4	1,96
Soma	9,06	0	3,065
Media	3,02		

(X - Media)²/n-1
1,5325

Sem Framework	Y	Y - Media	(Y - Media)²
G4	4,13	-1,9533333	3,815511111
G5	7,4	1,31666667	1,733611111
G6	6,72	0,63666667	0,405344444
Soma	18,25	8,8818E-16	5,954466667
Media	6,08333		

(Y - Media)²/n-1
2,97723

$\alpha = 0,2$

$t_{\alpha/2,\ gl} = t_{0,1000,\ 4} = 2,1318$

$Sp = 1,501620791012165$

$t_0 = -2,498498775014366$, temos que $|t_0| > t_{0,1000,\ 4}$ **REJEITAR a hipótese nula H_0 com 20% de significância.**

(2) Segunda etapa do *t-test* (Produtividade Total)

Tabela 3: Dados de análise do t-test dos grupos (Com/Sem Framework-Produtividade Total)

Com Framework	X	X - Media	(X - Media)2
G1	838	-242	58564
G2	833	-247	61009
G3	1569	489	239121
Soma	3240	0	358694
Media	1080		

(X - Media)²/n-1
179347

Com Framework	Y	Y - Media	(Y - Media)2
G4	543	145	21025
G5	312	-86	7396
G6	339	-59	3481
Soma	1194	0	31902
Media	398		

(Y- Media)²/n-1
15951

$\alpha = 0,02$

$t_{\alpha/2,\, gl} = t_{0,01,\, 4} = 4,6041$

$Sp = 130,1691207621838$

$t_0 = 5,475964737075994$, temos que $|t_0| > t_{0,01,\, 4}$ **REJEITAR a hipótese nula H_0 com 2% de significância.**

(3) Conclusões do Experimento

A partir do momento em que a hipótese nula (H_0) é rejeitada, torna-se possível tirar conclusões acerca do impacto das variáveis independentes sobre a variável dependente de um experimento.

De maneira decorrente, a rejeição observada da hipótese nula (H_0) neste experimento leva a afirmar que os grupos que desenvolveram com o *framework* e aqueles que desenvolveram sem ele apresentaram diferenças de eficiência com uma significância estatística. Dessa forma, é provável que essa diferença de eficiência esteja

relacionada aos métodos de desenvolvimento usados e não a uma coincidência por erros de amostragem.

De acordo com a tabela 1 (*Dados Coletados*), pode ser observado que as equipes do *framework* tiveram, em média, menor tempo total do que aquela de abordagem sem o uso do *framework*, ou seja, $\mu_{\tau framework} < \mu_{\tau semframework}$. De outro lado, as equipes do *framework* tiveram maior produtividade em relação às demais equipes, isto é, $\mu_{Þframework} > \mu_{Þsemframework}$. Com isso, fica evidente a possibilidade de validar a hipótese alternativa H_1 em detrimento da H_2, isto é, pode-se afirmar que as equipes que usaram o *framework* são, em geral, mais eficientes do que aquelas que desenvolveram sem o uso do *framework*, e não o contrário. Assim, fica evidente observar que esse resultado está de acordo com as expectativas iniciais do *framework* proposto neste trabalho de mestrado, pois acreditou-se que os artefatos acolhidos na fase de análise de domínio e acoplados ao *framework*, facilitariam a construção de aplicações deste domínio e maximizariam o custo-benefício.

Devido ao fato do experimento ter sido realizado em um ambiente *in-vitro,* onde houve limitação ao escopo de desenvolvedores de aplicações no ambiente universitário, sugere-se, para fim de generalização dos resultados aqui apresentados, que se desenvolva uma nova experimentação, neste caso em ambientes *in-vivo* com mais equipes e fazendo comparação com outras abordagens de desenvolvimento como, por exemplo, LPS (Linha de Produtos de Software), desenvolvimento orientado a componentes, etc.

Capítulo 6

TRABALHOS CORRELATOS

Muitos autores relatam sobre projetos de construção de *frameworks* para apoiar o desenvolvimento de aplicações de TV. No entanto, poucos deles se concentram em reutilização em aplicações Smart TV, especificamente aquelas sincronizadas com a programação da televisão. Dessa forma, este capítulo relaciona alguns dos trabalhos, de relevância, encontrados na literatura fazendo uma descrição em geral sobre o que eles abordam, comparando com funcionalidades do SyncSmartv e evidenciando quais foram as contribuições aproveitadas.

GROUP SHARE-TV (2012) propõe um framework chamado de *Share- TV* utilizado para o desenvolvimento de aplicações convergentes centradas na TV para as plataformas *GoogleTV* e *Ginga-J*. O *Share- TV* permite o desenvolvimento de aplicações de TVs que incluem uma aplicação *mobile* genérica. Quando algum dispositivo *mobile* deseja interagir com a aplicação da TV, não é necessário acessar a loja virtual de aplicativos da plataforma para fazer o download do aplicativo, basta digitar o IP da TV disponível para iniciar o download. Uma vez que a aplicação genérica do *Share-TV* é instalada no dispositivo *mobile* não é mais preciso repetir o processo nas próximas vezes que esse mesmo dispositivo desejar interagir com demais aplicações da TV desenvolvidas usando *Share- TV*. Sendo assim, somente será preciso executar a aplicação genérica que vai iniciar a comunicação com a aplicação convergente da TV, registrando-se e recebendo objetos compartilhados. À medida que os objetos são recebidos, o dispositivo os apresenta na tela permitindo a interação sobre eles. Quando

comparado com o *framework* proposto neste trabalho de mestrado, *Share- TV* provê também serviço de comunicação e faz reuso da aplicação *mobile* genérica, porém esse *framework* se limita as aplicações nas plataformas *GoogleTV* e *Ginga-J*. Outra diferença é a integração de aplicações da TV com a programação televisiva: *Share- TV* é voltado para convergência com aplicações da TV enquanto o trabalho que se propõe para o mestrado aborda aplicações da TV sincronizadas com o conteúdo dos programas da TV.

PAPADOPOULOS et al. (2009) propõem um *framework (CHP: Connected Home Platform)* para design, desenvolvimento e *deployment* de serviços inteligentes voltados para casas inteligentes (*Smart Home*). CHP provê uma arquitetura de rede que permite a integração de conectividade de *devices* de diferentes tipos de tecnologias de controle de casa, como câmeras, equipamento de automação. CHP consiste em diferentes componentes de softwares executando em diferentes elementos de rede como clientes GUI (*Grafic User Interface*) para telefones celulares e PC, *Portal Server* para *login* e autenticação remotos do usuário e um software de *gateway* para acesso a dispositivos de casa com objetivo de controlar a casa remotamente. Além disso, CHP proporciona benefícios para usuários gerando alertas e notificações por email e SMS sobre as condições das suas casas. O trabalho proposto por PAPADOPOULOS et al. (2009) apresenta um *framework* que facilita o desenvolvimento de serviços de controles de casa inteligentes. Apesar do reuso oferecido, da melhoria da experiência do usuário e da presença da TV no ambiente da *Smart Home,* CHP não tem no seu escopo sincronização com a programação da TV.

SAMSUNG SMART TV (2014) propõe um *framework* chamado de AppsFramework. AppsFramework encapsula módulos reutilizáveis para gerenciamento de cenas, *playback* de vídeo/músicas, entre outros. Isso oferece facilidades para o desenvolvedor de aplicações Smart TV evitando realizar sequencias de chamadas complicadas para o sistema operacional para gerenciar cenas (*focus, showing and hiding events*) de uma aplicação, por exemplo. Como pode ser notado no capítulo 4, a arquitetura do SyncSmartv foi baseada na arquitetura do AppsFramework.

FREITAS e TEIXEIRA (2010) propõem uma arquitetura para o suporte ao desenvolvimento de aplicações ubíquas em redes domésticas centradas em TV Digital. A arquitetura proposta é composta por interface de comunicação com dispositivos domésticos, uma camada de protocolo de descoberta automática de serviços, entre outros. Apesar de arquitetura ser projetada para ser implementada nos *middlewares* de TVDI, alguns de seus artefatos reutilizáveis como os citados acima foram aproveitados e implementados no *framework* construído no trabalho de mestrado aqui proposto.

Capítulo 7

Este trabalho apresentou um *framework* de referência que busca facilitar a construção de aplicações Smart TV integradas à TV, permitindo ao desenvolvedor construir aplicações, de grande utilidade para o usuário, nesse domínio com menor custo e esforço.

SyncSmartv é fornecido aos desenvolvedores na forma de esqueleto de códigos semi-completo que integra módulos funcionais relativos à sincronização, notificação e controle da TV. O conjunto ferramental desenvolvido e apresentado na figura 12 com o propósito de provas de conceito do *framework* pode ser reaproveitado no processo de construção de novas aplicações nesse domínio provendo assim facilidades aos desenvolvedores. Além disso, todos os requisitos funcionais planejados durante a fase de planejamento do *framework* foram implementados e usados durante a instanciação do SyncSmartv para testes. Além de tudo, o experimento conduzido nesse trabalho pude provar, estatisticamente, que o SyncSmartv pode ser considerado como uma ferramenta importante para apoiar desenvolvedores de aplicações Smart TV integradas à programação televisiva.

O experimento conduzido teve o propósito de avaliar a eficiência dos desenvolvedores durante a construção de aplicações usando o *framework* proposto. Com a análise dos resultados, concluiu-se que o *framework* é viável e permitiu aos desenvolvedores terem uma redução de esforços de 57,2 % na construção de aplicações Smart TV integradas. Além disso, com o desenvolvimento do trabalho pôde-se

constatar que aplicações sincronizadas com a programação da tv são realmente bastante interessantes e podem trazer bom retorno para os telespectadores. É provável que a atenção que algumas aplicações integradas possam chamar levem à busca pelo desenvolvimento de muitas outras aplicações desse tipo. Isso acontecendo coloca o *framework* SyncSmartv como importante apoio a desenvolvedores. Além de ser um *framework* para o auxílio ao desenvolvimento de aplicações integradas, SyncSmartv possui APIs que podem também apoiar o desenvolvimento de aplicações de TV Social.

A principal contribuição do trabalho é a arquitetura do *framework* para apoiar o desenvolvimento de aplicações Smart TV integradas.

A segunda contribuição é a implementação do *framework* usando a arquitetura proposta.

A terceira contribuição refere-se a API de notificações para promover o sincronismo tratado no contexto deste trabalho.

A quarta contribuição é a criação de uma interface, baseada no protocolo *http*, para facilitar a comunicação *two-way* e o compartilhamento de conteúdo de mídias entre a TV e dispositivos móveis. A aplicação MobileSync implementa essa interface e o seu código fonte está disponível para que desenvolvedores possam usá-lo para seus próprios fins.

O trabalho em questão abre possibilidades para alguns trabalhos futuros, entre os quais:

- Avaliação do *framework* em ambiente *in vivo*.
- Avaliação com usuários finais para verificar dois parâmetros: *Perceived Usefulness* e *Perceived Use of Use*.
- Acréscimo de mais mecanismos de sincronização ao *framework* através de processamento local de áudio e vídeo dos canais da TV.
- Extensões das API's de notificação para apoiar o desenvolvimento de sistemas de TV Social em ambiente de Smart TV.
- Exploração de mecanismos de ajustes de sincronização para superar diferenças de atrasos entre as diferentes formas de difusão da

programação televisiva, como terrestre analógica, terrestre digital, cabo, satélite e internet.

REFERÊNCIAS

APPLETON, B. **Patterns and Software:** Essential Concepts and Terminology. 1998. Disponível em: <http://www.enteract.com/~bradapp/docs/patterns-intro.html >. Acessado em: mar. 2014.

ARES, J; DIESTE, O; GÁRCIA, R; LÓPEZ M; RODRÍGUEZ, S. Formalising the Software Evolution Process. In: Proceedings of the XVIII International Conference of the Chilean Computer Science Society, 1998. **Proceedings...** IEEE Computer Society, 1998, 15-24.

BACHELET, C. Most smart-TV owners do not connect their TVs to the Internet: manufacturers must respond. **Analysys Mason** 2013. Disponível em: <http://www.analysysmason.com/About-Us/News/Insight/smart-TV-May2013/>. Acessado em: mar. 2014.

BOSCH, J; MOLIN, P; MATTSSON, M; BENGTSSON, P; FAYAD, M. Framework Problems and Experiences. In: FAYAD, M.; JOHNSON, R.; SCHMIDT D. Building Application Frameworks: **Object-Oriented Foundations of FrameworkDesign**.Nova Iorque : John Willey and Sons, p. 55-82, 1999.

CARVALHO, D. F. Video Steganography for Confidential Documents: Integrity, Privacy and Version Control. **SIGDOC'08**, Lisbon, Portugal. September 22-24, 2008.

CLEMENS, S. Component Software. Beyond Object-Oriented Programming. [S.l.]: **Addison-Wesley Reading,** MA, 1998.

CLEMENTS, P; NORTHROP, L. M. **Software product lines:** practices and patterns. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2001, ISBN 0-201-70332-7.

COSTA, R. E. O. et al. Using Video Embedding Markings for Supporting Content Sensitive Interaction in Multiple Contexts. **WebMedia'13**, Salvador, Brazil. November 5–8, 2013.

CROWELL, G. How to use QR Codes in Video Marketing and E-Commerce Video Marketing. **Reelseo,** 18 jan. 2011. Disponível em: < http://www.reelseo.com/video-qr-codes/#top >. Acessado em: mar. 2014.

FRAKES, W; TERRY, C. **Software reuse:** metrics and models. ACM Computing Surveys (CSUR), ACM, v. 28, n. 2, p. 415–435, 1996.

FREITAS, G; TEIXEIRA, C. Uma arquitetura de serviços para aplicações ubíquas em redes domésticas centrada em TV digital. In: XVI Simpósio Brasileiro de Sistemas Multimídia e Web (Webmedia 2010), 2010, Belo Horizonte - MG. **Anais do XVI Simpósio Brasileiro de Sistemas Multimídia e Web (Webmedia 2010)**. Porto Alegre: SBC, 2010.

GAMMA, E. et al. **Design patterns:** elements of reusable object-oriented software. [S.l.]: Addision-Wesley, 1995.

GAO, J; ZHAO, Q; YAN, Y. Automatic synchronization of live speech and its transcripts based on a frame-synchronous likelihood ratio test. **Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '10),** pp. 1622–1625,March 2010.

GASPAR, C. Linha de produtos de software para comunicação síncrona na web. 2010. Dissertação (Mestrado em Ciência da Computação) - Centro de Ciências Exatas e de Tecnologia, Universidade Federal de São Carlos, São Carlos, 2010.

GROUP SHARE-TV. Share-TV: Um framework para desenvolvimento de aplicativos convergentes centrados na TV para as plataformas GoogleTV e Ginga-J. **Webmedia '12**. São Paulo, 2012.

MELO, E. L. Ad-Avoidance Technology Como Catalisadora da Interatividade na TV. 2014. Tese (Doutorado em Ciência da Computação) - Centro de Ciências Exatas e de Tecnologia, Universidade Federal de São Carlos, São Carlos, 2014.

PAPADOPOULOS, N et al. A Connected Home Platform and Development Framework for Smart Home Control Applications. **Proceedings of the 7th IEEE International Conference on Industrial Informatics (INDIN 2009)**, 2009.

PARK, J; KIM, M. Demand forecasting and strategies for the successfully development of the smart TV in Korea. **Proceeding of International Conference on Advanced Communication Technology**, p. 1475-1478, 2011.

PEITZ, M; VALLETTI, T.M. **Content and advertising in the media:** Pay-tv versus free-to-air. International Journal of Industrial Organization, 2008, v. 26, p. 949-965.

POMPONET, T. Agente Incentivador no Enriquecimento da QoE em TV Social. 2014. Dissertação (Mestrado em Ciência da Computação) - Centro de Ciências Exatas e de Tecnologia, Universidade Federal de São Carlos, São Carlos, 2014.

PRESSMAN, Roger S. Engenharia de software. 6ª ed. Porto Alegre: **Bookman**, 2006.

ROCHET, J.C; TIROLE, J. **Two-Sided Markets:** An Overview. Institut d'Economie Industrielle working paper, 2004, p. 1-44.

RODRIGUES, K. R. H. et al. Interação com conteúdo complementar por meio de múltiplos dispositivos para apoio à apreciação de programas televisivos. **XVII Webmedia.** Florianopolis, Santa Catarina, Brasil, 3-6 de out. 2011.

SAMSUNG SMART TV. AppsFramework. Disponível em: <http://www.samsungdforum.com/Guide/art00017/index.html>. Acessado em: mai. 2014.

SAROSI, G. User Interface Development for SmartTV using Web technology and CEA2014. **Time Warner Cable,** 2011. Disponível em: <http://www.w3.org/2011/09/webtv/papers/webandtvSept2011-twc-v1.0.pdf>. Acessado em: dez. 2013.

SCHOFIELD, J. Smart TVs may be taking off, but they're still not smart enough. **ZDNet,** 2012. Disponível em: <http://www.zdnet.com/smart-tvs-may-be-taking-off-but-theyre-still-not-smart-enough-7000008042/ >. Acessado em: mar. 2014.

SHIN, D; HWANG ,Y; CHOO, H. Smart TV: are they really smart in interacting with people? Understanding the interactivity of Korean Smart TV. **Journal Behaviour & Information Technology**, v. 32, p. 156 - 172, fev. 2013.

STAUFFER, M. Connectivity Solutions For Smart TVs**. IEEE Second International Conference on Consumer Electronics - Berlin (ICCE-Berlin)**, Qualcomm Atheros, San Jose, Calif. USA. 2012.

SUNGJOON, L. A Study on Acceptance and Resistance of Smart TVs. **International Journal of Contents**, Korea, v. 8, n. 3, Sep. 2012.

TEIXEIRA, C. et al. **Discrimination of media moments and media intervals:** sticker-based watch-and-comment annotation. Multimedia Tools and Applications, 2011, p. 1-22.

Teixeira, C.A.C. et al. Mechanisms of synchronization for multimedia applications. 2015.

TEIXEIRA, C.A.C. et al. Taking advantage of contextualized interactions while users watch TV. **Multimedia Tools and Applications,** 2010, p. 587-607.

TOKAN, F. **Media as Multitasking**: An Exploratory Study on Capturing Audiences Media Multitasking and Multiple Media Use Behaviours (Marketing Master's thesis). Department of Marketing Aalto University School of Economics. 2011.

TUOMI, P. A brief history of social iTV entertainment. **Proceedings of the 13th International MindTrek Conference**: Everyday Life in the Ubiquitous Era, September 30-October 02, 2009, Tampere, Finland

UPnP TEAM. UPnP Device Architecture. Disponível em: < http://www.upnp.org/ >. Acessado em: jun. 2014.

WALLIS, C. The multitasking generation. **Time Magazine**, 2006, v. 167, p. 48-55. Disponível em: <http://how-we-learn.blogspot.com.br/2007/03/article-multitasking-generation-time.html>. Acessado em: dez. 2013.

WOHLIN, C; RUNESON, P; HOST, M; OHLSSON, M.C; REGNELL, B; WESSLÉN, A. **Experimentation in Software Engineering:** an introduction. Kluwer Academic Publishers, 2000.

ZHAO, J; KOCH, E; LUO, C. In business today and tomorrow. **Communications of the ACM**, New York, NY, USA ,v. 7, p. 67-72, jul.1998 .

ZHENG H; LUGMAYR A. et al. Predicting TV in the year 2013. **MindTrek '13,** Finland, set. 28-30 2013.

Apêndice A

FORMULÁRIO DE CARACTERIZAÇÃO DO PARTICIPANTE

Este formulário tem por objetivo caracterizar sua experiência acadêmica, pessoal e profissional com relação à Ciência da Computação. Por favor, responda TODAS as questões o mais fielmente possível. Toda informação fornecida é confidencial, sendo que seu nome e quaisquer outros meios de identificação não serão divulgados em nenhuma hipótese.

1) Dados do participante

Registro acadêmico: _____

Idade:_____

2) Formação acadêmica

( ) Graduação

( ) Mestrado

( ) Doutorado

Ano de ingresso: _____. Mês/Ano de Conclusão (ou previsão de conclusão): _____/ _____

3) Experiência profissional

3.1 Analise a opção que melhor reflita o seu grau atual de experiência com as tecnologias listadas a seguir, considerando a escala de 5 pontos.

0 = nenhum

1 = estudei em aula ou livro

2 = pratiquei em projetos em sala de aula

3 = usei em projetos pessoais ou indústria

4 = uso em grande parte dos projetos que realizo

3.1.1) Linguagem de Programação Javascript	0	1	2	3	4
3.1.2) Linguagem de Marcação HTML5	0	1	2	3	4
3.1.3) Linguagem de Marcação XML	0	1	2	3	4
3.1.4) JavaScript Object Notation (JSON)	0	1	2	3	4
3.1.5) Folhas de Estilo (CSS)	0	1	2	3	4
3.1.6) Biblioteca Jquery	0	1	2	3	4
3.1.7) IDE Eclipse	0	1	2	3	4

3.2) Qual das opções a seguir melhor descreve sua experiência anterior com relação ao desenvolvimento de software na prática?

Tenho desenvolvido software por conta própria (sozinho) ( )	Tenho desenvolvido software como membro da equipe durante cursos que realizo ( )	Tenho desenvolvido software como membro de equipe, na indústria, há menos de 2 anos ( )	Tenho desenvolvido software como membro de equipe, na indústria, há 2 anos ou mais ( )

3.2.1) Por favor, detalhe sua resposta indicando o número de meses de experiências relevantes tanto em projetos desenvolvidos durante algum curso, como em projetos desenvolvidos na indústria.

a) Quantos meses de experiência você teve como gerente de projeto de software?

Meses de experiência: _____ (como aluno em um curso) _____(na indústria)

b) Quantos meses de experiência você teve como analista de sistema?

Meses de experiência: _____ (como aluno em um curso) _____(na indústria)

c) Quantos meses de experiência você teve como desenvolvedor (programação)?

Meses de experiência: _____ (como aluno em um curso) _____(na indústria)

3.2.2) Como você distribuiria o tempo gasto em sua experiência com programação?

_____% em esforço individual (sozinho)

_____% em desenvolvimento conjunto com outras pessoas (equipe)

_____% na supervisão de outros programadores (gerência)

100% TOTAL

3.3) Qual sua habilidade em...

a) ...desenvolver softwares para Web?

( ) Especialista ( ) Avançado ( ) Médio ( ) Básico ( ) Nenhum

Meses de experiência: _____ (como aluno em um curso) _____(na indústria)

b) ...desenvolver interfaces gráficas para Web?

( ) Especialista ( ) Avançado ( ) Médio ( ) Básico ( ) Nenhum

Meses de experiência: _____ (como aluno em um curso) _____(na indústria)

c) ...desenvolver aplicações/widgets para TV Digital/Smart TV?

( ) Especialista ( ) Avançado ( ) Médio ( ) Básico ( ) Nenhum

Meses de experiência: _____ (como aluno em um curso) _____(na indústria)

4) Experiência com os conceitos e técnicos empregados no experimento

Esta seção será utilizada para compreender sua familiaridade com os conceitos e técnicas que serão utilizados nas atividades do experimento.

Assinale na tabela a seguir a opção que melhor reflita seu grau atual de experiência com os itens relacionados, considerando a escala de 4 pontos:

0 = Eu não tenho familiaridade com este assunto. Eu nunca fiz isto.

1 = Já li ou estudei sobre isto. Conheço os conceitos e/ou técnicas

2 = Eu utilizo isto algumas vezes em projetos pessoais ou na indústria, mas não sou um (a) especialista.

3 = Eu sou muito familiar com este assunto. Eu me sentiria conformável fazendo isto.

4.1) Desenvolvimento para a plataforma Samsung Smart TV	0	1	2	3
4.2) Aplicações da TV sincronizadas com programas televisivos	0	1	2	3
4.3) Aplicações convergentes.	0	1	2	3

Obrigado por sua colaboração

Apêndice B

FORMULÁRIO DE CONSENTIMENTO

Experimento

Este experimento visa avaliar o uso do framework proposto para apoio à construção de aplicações para Smart TV que sejam integradas e sincronizadas com o programa exibido na TV.

Idade:

Eu declaro ser maior de 18 (dezoito) anos de idade e concordo participar do experimento conduzido por Cédrick Bamba Nsimba da Universidade Federal de São Carlos (UFSCar).

Procedimento:

Este experimento ocorrerá em uma única sessão, que incluíra o desenvolvimento de uma aplicação convergente integrada e sincronizada com o programa exibido na TV. O desenvolvimento dessa aplicação se dará com e sem o uso do framework proposto, conforme determinado pelo experimentador. Entende-se que, uma vez que o experimento tenha sido concluído, os trabalhos desenvolvidos, bem como os dados coletados, serão estudados visando analisar a aplicação dos procedimentos e técnicas propostos.

Confidencialidade:

Eu estou ciente que toda informação coletada neste experimento é confidencial, e meu nome ou quaisquer outros meios de identificação não serão divulgados. Da mesma forma, me comprometo a não comunicar meus resultados aos demais participantes e/ou a grupos enquanto não terminar o experimento, bem como

manter sigilo das técnicas e documentos apresentados que fazem parte do experimento.

Benefícios e liberdade de desistência:

Eu entendo que participo livremente com o único intuito de contribuir para o avanço e desenvolvimento de frameworks de apoio para a construção de aplicações Smart TV integradas e sincronizadas com programas da TV.

Pesquisador responsável:

Cédrick Bamba Nsimba

Programa de Pós Graduação em Ciência da Computação – PPG-CC/DC/UFSCar

Professor responsável:

Prof. Dr. Cesar Augusto Camillo Teixeira

Programa de Pós Graduação em Ciência da Computação – PPG-CC/DC/UFSCar

Ao preencher e assinar este documento dou plena ciência e consentimento com os termos acima expostos.

Nome (em letra de forma) _____ RA:_____

Assinatura:_____

Data: _____/_____/_____

Apêndice C

DESCRIÇÃO DA APLICAÇÃO E MATERIAL DE APOIO-TESTE DO FRAMEWORK SMART TV

Descrição da Aplicação *TVMonitor*

O *TVMonitor* é uma aplicação integrada e sincronizada com a programação televisiva, capaz de receber notificações sobre eventos que estejam ocorrendo na programação da TV e de permitir aos usuários interagir com os recursos da própria plataforma da TV e demais conteúdos complementares, por exemplo, de forma sincronizada. Esses usuários, ao assistirem televisão, podem utilizar o controle remoto para fazer algumas configurações da aplicação.

O telespectador, munido de um dispositivo móvel (*smartphone, tablet*) com suporte para *Bluetooth*, *Wifi*, por exemplo, ao ser avisado sobre um serviço disponível da TV, pode conectar seu dispositivo à TV e usufruir desse serviço para fazer o *playback* de seus vídeos localizados no celular ou em um servidor de mídia remoto ou também visualizar a galeria de fotos do seu celular na tela da TV.

Como a TV, por sua natureza, é um dispositivo de uso familiar, cada usuário precisa cadastrar-se na aplicação e cadastrar também um ou mais dispositivos móveis de sua propriedade.

Sendo de natureza de convergência midiática, o modelo de implementação do *TVMonitor* é constituído por quatro partes: a primeira, executada na Smart TV do usuário, registra suas interações de escolha de opções oferecidas pela aplicação, recebe sinais de sincronização e orquestra o uso dos recursos da TV por ele; a segunda, executada no dispositivo móvel do usuário, realiza a descoberta do serviço da TV, conecta e comunica-se com a TV para o compartilhamento de conteúdo de mídia de forma sincronizada com a programação da TV; a terceira, executada no

servidor, processa os registros de aplicações de usuários e os armazena em uma base de dados, auxilia na descoberta de serviços da TV e na troca de conteúdos de mídia entre dispositivos móveis e TVs; e a quarta, executada em um dispositivo móvel, trata-se de um módulo *android* que disponibiliza uma interface para simular a geração de sinais de sincronização providos por terceiros.

Para atender aos propósitos deste experimento, alguns dos casos de uso específicos para o *TVMonitor* foram identificados e são apresentados no diagrama a seguir. Pede-se aos participantes deste experimento focar-se no desenvolvimento das interfaces e lógicas de negócios que atendam a estes casos de uso, utilizando ou não o *framework* proposto, dependendo do grupo em que cada participante está inserido.

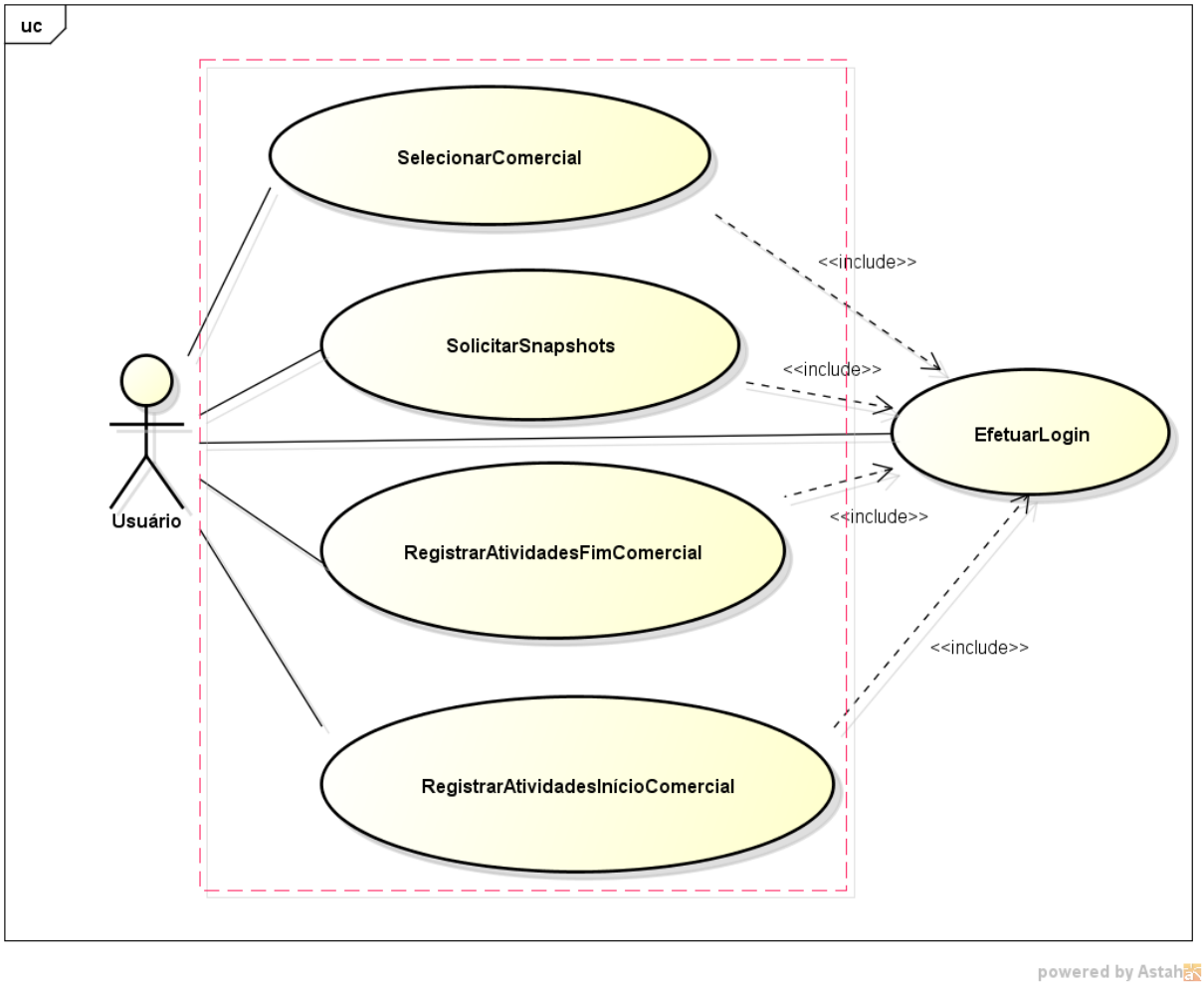


Figura 15: Diagrama de casos de uso do *TVMonitor*

Conforme o diagrama de casos de uso e a descrição do modelo do *TVMonitor* descrito acima, a aplicação a ser desenvolvida constitui a primeira parte do *TVMonitor*. Ela é uma aplicação integrada ao programa da TV permitindo ao telespectador programar ações a serem executadas no início e fim de blocos comerciais e mantém um relógio sincronizado com o servidor possibilitando o telespectador solicitar informações ou dados, como fotos, em torno de um determinado momento do estágio da programação televisiva.

Apoio para implementação das funcionalidades do *TVMonitor*

Ao iniciar a aplicação, a tela contendo uma tecla para o *Menu Principal* é exibida. Ao acionar essa tecla, é apresentado o menu contendo as opções de ***(1)Programar Ações InícioComercial, (2)Programar Ações FimComercial, (3)Selecionar um comercial, (4)Solicitar snapshots***.

Para (1):

- Ao selecionar a ação **"Mudança de canal",** deve-se:
  (a) Listar todos os canais disponíveis para que o telespectador possa escolher para o qual quer ir ao iniciar um comercial no canal corrente.
  (b) Guardar o *ID* do canal escolhido pelo telespectador;
  (c) Registrar a notificação de início de comercial com suas devidas ação e função *callback*;
  (d) Guardar o *ID* do canal corrente;
  (e) Fazer o *subscribe* do canal corrente no serviço de monitoramento.
- Ao selecionar a ação **"Playback Vídeo",** deve-se:
  (a) Guardar o *ID* do canal corrente;
  (b) Registrar a notificação de início de comercial com suas devidas ação e função *callback*;
  (c) Fazer o *subscribe* do canal corrente no serviço de monitoramento.
- Ao selecionar a ação **"Exibir Foto do Mobile",** deve-se:
  (a) Guardar o *ID* do canal corrente;

(b) Registrar a notificação de início de comercial com suas devidas ação e função *callback*;

(c) Fazer o *subscribe* do canal corrente no serviço de monitoramento.

Para (2):

- Ao selecionar a ação **"Voltar ao canal do registro",** deve-se:

(a) Recuperar o *ID* do canal no qual o telespectador fez registro do evento de início de comercial;

(b) Implementar a função *callback* dependendo da ação escolhida para o início de comercial;

(c) Registrar a notificação de fim de comercial com suas devidas ação e função *call-back* implementada o item (b);

Para o *playback* de vídeos e áudios do usuário, o seguinte player oferecido pelo *framework* será chamado:

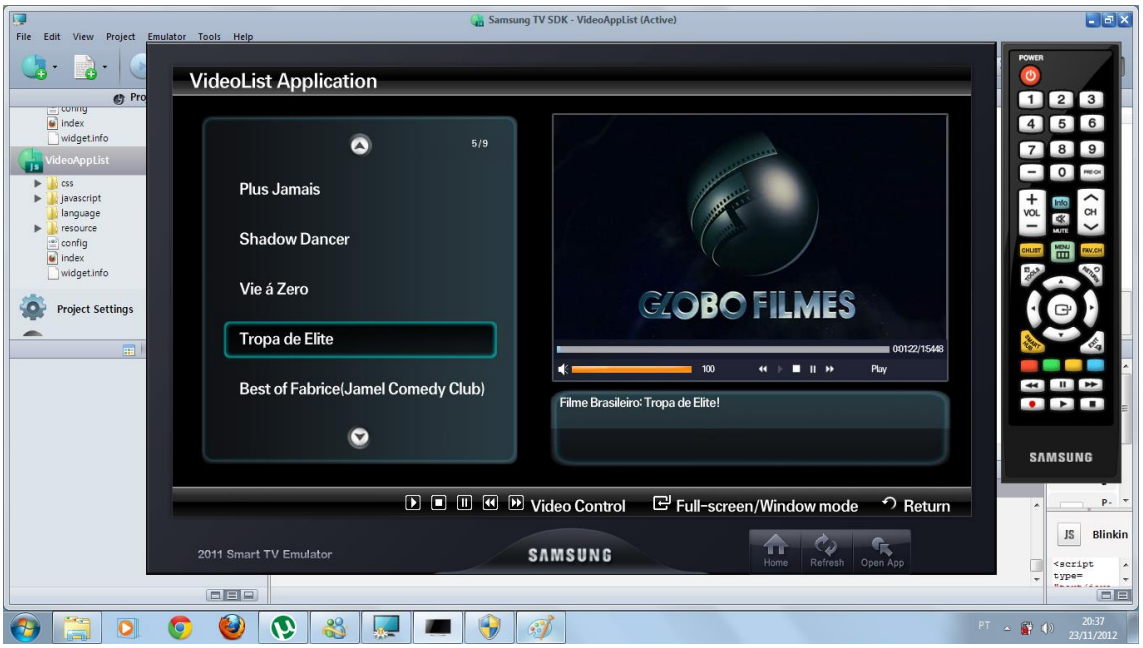


Figura 16: Interface Gráfica do *Player*

Apêndice D

DESCRIÇÃO DA TAREFA –GRUPO DO FRAMEWORK SMART TV

Instruções

Este experimento avaliará o uso do *framework* proposto para apoio à construção de aplicações para Smart TV que sejam integradas e sincronizadas com o programa exibido na TV. A análise dos resultados será baseada nos dados coletados durante a execução das atividades propostas no experimento. Assim, de forma que o experimentador possa melhor avaliar os resultados obtidos, nos formulários entregues ao seu grupo preencha corretamente todos os dados relacionados ao horário de início e término de cada atividade, bem como todos os dados relacionados ao número de linhas de código implementadas.

Da mesma maneira, caso encontre algum problema de ordem técnica durante a execução de determinada atividade, como configuração do IDE, configuração da máquina, etc., anote os horários de identificação e solução do problema juntamente com sua descrição no formulário apropriado.

Pergunte e comente tudo o que julgar necessário.

Contextualização

As aplicações Smart TV disponíveis atualmente nas lojas virtuais dos fabricantes não estão integradas com os programas televisivos. A falta de mecanismos que facilitariam sua comunicação com o conteúdo audiovisual da TV, para saber dos eventos ocorrendo na programação televisiva, por exemplo, pode ser um fator que contribua com essa falta de integração. Como técnicas de sincronização para resolver esse problema é assunto de amplo domínio de desenvolvedores de aplicações para Smart TV, seria interessante que soluções

adequadas pudessem ser disponibilizadas para facilitar o processo de construção de aplicações Smart TV sincronizadas com o programa da TV. Uma das formas de se promover essa disponibilização é através de reuso de software. Reuso de Software é uma prática na qual se utiliza artefatos de software existentes para construção de novos produtos de software. Os artefatos reutilizáveis vão desde trechos de código e bibliotecas até componentes de software, serviços Web, arquiteturas de projetos, padrões e frameworks. Além de diminuir os custos e o tempo de desenvolvimento de software, uma vez bem aplicado, o reuso pode proporcionar grande ganho de qualidade e produtividade.

Tarefa

Você recebeu a descrição de uma aplicação Smart TV denominada *TVMonitor*. Uma vez que o foco do processo avaliado está nas funcionalidades (de comunicação, de gerenciamento de sinais de sincronização, de descoberta de serviços, etc.) oferecidas por ele, a maioria dos componentes gráficos da aplicação a ser desenvolvida já encontram-se implementados. **Sua tarefa é desenvolver as regras de negócio da aplicação *TVMonitor* para Smart TV da Samsung utilizando as facilidades oferecidas pelo framework proposto.**

Utilize o formulário de descrição do *framework* proposto e o material de apoio para auxiliá-lo na execução das atividades. Tendo que os requisitos da aplicação já foram identificados, conforme mostra o diagrama de casos de uso do formulário de descrição da aplicação, inicie o desenvolvimento a partir da atividade de **Projeto**. Para cada atividade realizada anote os dados temporais e de linhas de código no formulário apropriado.

Ao final do experimento, compacte o projeto da aplicação desenvolvida e envie por email para cedrick.bamba@gmail.com, usando como assunto o seguinte padrão: *"GRUPO [Nº de seu grupo] - experimento"*.

Apêndice E

DESCRIÇÃO DA TAREFA –GRUPO DO SEM APOIO DO FRAMEWORK SMART TV

Instruções

Este experimento avaliará o uso do *framework* proposto para apoio à construção de aplicações para Smart TV que sejam integradas e sincronizadas com o programa exibido na TV. A análise dos resultados será baseada nos dados coletados durante a execução das atividades propostas no experimento. Assim, de forma que o experimentador possa melhor avaliar os resultados obtidos, nos formulários entregues ao seu grupo preencha corretamente todos os dados relacionados ao horário de início e término de cada atividade, bem como todos os dados relacionados ao número de linhas de código implementadas.

Da mesma maneira, caso encontre algum problema de ordem técnica durante a execução de determinada atividade, como configuração do IDE, configuração da máquina, etc., anote os horários de identificação e solução do problema juntamente com sua descrição no formulário apropriado.

Pergunte e comente tudo o que julgar necessário.

Contextualização

As aplicações Smart TV disponíveis atualmente nas lojas virtuais dos fabricantes não estão integradas com os programas televisivos. A falta de mecanismos que facilitariam sua comunicação com o conteúdo audiovisual da TV, para saber dos eventos ocorrendo na programação televisiva, por exemplo, pode ser um fator que contribua com essa falta de integração. Como técnicas de sincronização para resolver esse problema é assunto de amplo domínio de

desenvolvedores de aplicações para Smart TV, seria interessante que soluções adequadas pudessem ser disponibilizadas para facilitar o processo de construção de aplicações Smart TV sincronizadas com o programa da TV. Uma das formas de se promover essa disponibilização é através de reuso de software. Reuso de Software é uma prática na qual se utiliza artefatos de software existentes para construção de novos produtos de software. Os artefatos reutilizáveis vão desde trechos de código e bibliotecas até componentes de software, serviços Web, arquiteturas de projetos, padrões e frameworks. Além de diminuir os custos e o tempo de desenvolvimento de software, uma vez bem aplicado, o reuso pode proporcionar grande ganho de qualidade e produtividade.

Tarefa

Você recebeu a descrição de uma aplicação Smart TV denominada *TVMonitor*. Uma vez que o foco do processo avaliado está nas funcionalidades (de comunicação, de gerenciamento de sinais de sincronização, de descoberta de serviços, etc.) oferecidas por ele, a maioria dos componentes gráficos da aplicação a ser desenvolvida já encontram-se implementados. **Sua tarefa é desenvolver as regras de negócio da aplicação *TVMonitor* para Smart TV da Samsung.**

Utilize o formulário de descrição do serviço de sincronização *kaeptor* e o material de apoio para auxiliá-lo na execução das atividades. Tendo que os requisitos da aplicação já foram identificados, conforme mostra o diagrama de casos de uso do formulário de descrição da aplicação, inicie o desenvolvimento a partir da atividade de **Projeto**. Para cada atividade realizada anote os dados temporais e de linhas de código no formulário apropriado.

Ao final do experimento, compacte o projeto da aplicação desenvolvida e envie por email para cedrick.bamba@gmail.com, usando como assunto o seguinte padrão: *"GRUPO [Nº de seu grupo] - experimento"*.

Apêndice F

FORMULÁRIO DE COLETA DE DADOS - GRUPO DO FRAMEWORK SMART TV

Grupo nº: _____

Anote na tabela abaixo os horários de início e fim de cada atividade realizada.

Atividade	Hora início	Hora fim
Projetar (Modelagem)	_____:_____	_____:_____
Implementar	_____:_____	_____:_____
Testar	_____:_____	_____:_____

Anote as quantidades de linha de código geradas automaticamente (pelo *framework*) e manualmente.

Linhas de Código geradas **automaticamente** pelo *framework* : _____	Inclua na contagem apenas as linhas de código das telas (arquivos html), arquivos javascript e arquivos CSS customizados que não fazem parte de bibliotecas reutilizadas como jQuery, por exemplo.
Linhas de Código implementadas **manualmente** : _____	

Anote os problemas técnicos encontrados durante a execução do experimento, indicando o horário de identificação e resolução do problema, bem como sua breve descrição.

Descrição do Problema	Hora de Identificação	Hora de Resolução
	_____:_____	_____:_____
	_____:_____	_____:_____
	_____:_____	_____:_____
	_____:_____	_____:_____
Descrição do Problema	Hora de Identificação	Hora de Resolução

	_____:_____	_____:_____

Apêndice G

FORMULÁRIO DE COLETA DE DADOS - GRUPO DO SEM APOIO DE FRAMEWORK SMART TV

Grupo nº: _____

Anote na tabela abaixo os horários de início e fim de cada atividade realizada.

Atividade	Hora início	Hora fim
Projetar (Modelagem, decisões de tecnologias...)	_____:_____	_____:_____
Implementar (codificação)	_____:_____	_____:_____
Testar	_____:_____	_____:_____

Anote as quantidades de linha de código geradas automaticamente e manualmente. Considere linhas de código geradas automaticamente todo código que é criado automaticamente pelo IDE (t*emplates* CSS, *templates* das telas, etc) ou por qualquer mecanismo de geração automática de código que possa ter sido empregado pelo grupo.

Linhas de Código geradas **automaticamente** : _____	Inclua na contagem apenas as linhas de código das telas (arquivos html), arquivos javascript e arquivos CSS customizados que não fazem parte de bibliotecas reutilizadas como jQuery, por exemplo.
Linhas de Código implementadas **manualmente** : _____	

Anote os problemas técnicos encontrados durante a execução do experimento, indicando o horário de identificação e resolução do problema, bem como sua breve descrição.

Descrição do Problema	**Hora de Identificação**	**Hora de Resolução**
	_____:_____	_____:_____
	_____:_____	_____:_____
Descrição do Problema	**Hora de Identificação**	**Hora de Resolução**

	_____:_____	_____:_____
	_____:_____	_____:_____
	_____:_____	_____:_____

Anexo A

DOCUMENTAÇÃO DO *FRAMEWORK*

Neste anexo são apresentados o sumário com todas as classes do *framework*, os exemplos de uso das suas API's e, por fim, o conjunto de API's de notificações que implementam a sincronização. Vale notar que para efetivar uma sincronização é preciso fazer um *bind* entre o tipo de notificação, o tipo de sincronização e a função *callback* a ser executada quando o sinal de sincronismo for disparado. Esta documentação pode ser obtida, na integra no site do Lince: (http://lince.dc.ufscar.br/~cedrick/teste/jsdoc_toolkit-2.4.0/jsdoc-toolkit/out/jsdoc/index.html).

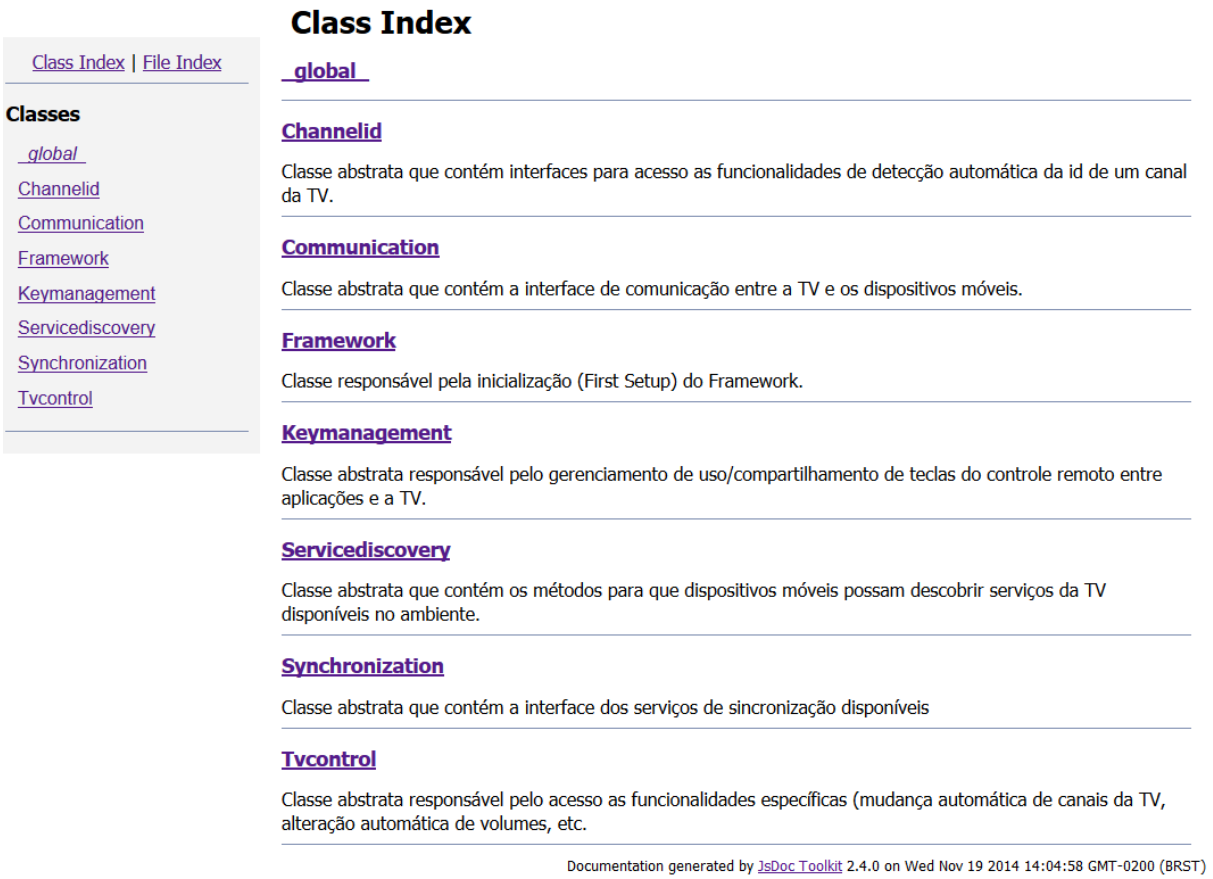


Figura 17: Sumário do SyncSmartv

```
contains a code example, illustrating how the synchronization HANDLER passed to
Synchronization class should be used in the user application.
TESTING SYNCHRONIZATION MANAGER
Synchronization.init(); //First setup
var list =  Synchronization.getHandlerList();//each syncMethod has id (list.id)
and callback function (list.callback)
var sync = new list[0].callback(); //Getting our first syncMethod instance
id="terceiro" and the callback function
sync.login("admin", "admin", function(){alert("LOGGED TO KAEPTOR SERVER");},
function(){alert("NOT LOGGED TO KAEPTOR SERVER");});
Synchronization.bind(notifData1, callback1(eventData), "terceiro");
Synchronization.bind(notifData2, callback2(eventData), "terceiro"); //Knowing
the syncMethod id i can bind some events/notifications(notifData) with
respective callback func.
var channelList = sync.list(success, error); //Retrieve info of avalaible channel
sync.subscribe("4f1476440cf21b20d62b0165", success_subscribe, error_subscribe);
//parameters (id, sucess, error)
sync.unsubscribe("4f1476440cf21b20d62b0165", success_unsubscribe, error_unsubscribe);
```

Figura 18: Syncronization API (Exemplo de Uso)

```
contains a code example, illustrating how the communication HANDLER passed to
Communication class should be used in the user application.
Communication.init(onconnect, ondisconnect, onshowpic, onvideo, onerrortype);//First
 setup callback parameters
var communication = Communication.getServiceCommunication();//RETRIEVE
COMMUNICATION SERVICE
TESTING COMMUNICATION MANAGER
onDeviceStatusChange = function(sParam){
                                        communication.onDeviceStatusChange(sParam);
                                    }; //Receiving device connect and disconnect EVENTS
onDeviceEvent = function(sParam){
                                        communication.onDeviceEvent(sParam);
                                    }; //Managing events occurring at the device instance
onCustomObtained = function(customs){
                                        communication.onCustomObtained(customs);
                                    }; //Used to register a callback function
                                    (onDeviceEvent) of each available device
onMessageReceived = function(message, context){
                                        communication.onMessageReceived(message, context);
                                    }; // Receiving messages from devices after
                                    onDeviceEvent occurs and do something
sendMessage = function(message){
                                        communication.sendMessage(message);
                                    }; // Send message to a given device ID
broadcastMessage = function(message){
                                        communication.broadcastMessage(message);
                                    }; // Send message to all connected devices
multicastMessage = function(message){
                                        communication.multicastMessage(message);
                                    }; // Send message to a given set of connected devices
```

Figura 19: Communication API (Exemplo de Uso)

```
contains a code example, illustrating how the service discovery HANDLER passed to
Servicediscovery class should be used in the user application.
Servicediscovery.init(); //First setup
var discovery = Servicediscovery.getServiceUrl(); // Getting service
discovery instance
discovery.sendUrl(); // Send tv service url to the server
```

Figura 20: Discovery API (Exemplo de Uso)

```
contains a code example, illustrating how the tvcontrol HANDLER passed to
Tvcontrol class should be used in the user application.
Tvcontrol.init(); //First setup
var channelchangeFromTo = Tvcontrol.getChannelChange(); //Getting
 channelchange instance
channelchangeFromTo.changingchannel(); // Channel Change being called
var mute = Tvcontrol.getMuteMode(); // For calling mute mode function
mute.mute();
var nomute = Tvcontrol.getNoMuteMode(); // For nomute mode function
nomute.nomute();
```

Figura 21: TV API (Exemplo de Uso)

```
contains a code example, illustrating how the channelid HANDLER passed to
Channelid class should be used in the user application.
var id = Channelid.init();//GETTING CURRRENT CHANNEL ID...
console.log(id); // id ={
                                ptc: ptc,
                                major: major,
                                minor: minor,
                                sourceID: sourceID,
                                programNumber: programNumber,
                                transportStreamID: transportStreamID
                                }
This id information will be used in channelchanging handler
```

Figura 22: Channelid API (Exemplo de Uso)

```
contains a code example, illustrating how the keymanagement HANDLER passed to
Keymanagement class should be used in the user application.
TESTING KEYMANAGEMENT
Keymanagement.onshow(tvKeyList);//The list of
keys to be used in the application being registered
to application manager
Keymanagement.isRegisteredKey(keycode);//Verifying
if the given key is already registered
Keymanagement.onend();//Unregister ticker application keys
```

Figura 23: Keymanagement API (Exemplo de Uso)

API's DE NOTIFICAÇÃO

(1) VISÃO GERAL

Notificações em relação aos eventos padrões

Para a especificação de cada notificação o seguinte formato será adotado: [**retorno1, retorno2, ..., retornoi**] (**parâmetro1**{Tipo, Valor,...}, **parâmetro2**{Tipo, Valor,...}, ..., **parâmetroi**{Tipo, Valor,...}). Para cancelar o registro de uma determinada notificação, basta realizar a chamada igual ao registro, portanto, com ID/lista de ID de canais da TV vazio(a):

1. **Início/Fim de bloco comercial**

[**codRequisição**{Tipo:"STRING", Valor:"inicio_bloco_comercial/ fim_bloco_comercial"}, **idCanal**{Tipo:"STRING"}]

(**codRequisição**{Tipo:"STRING", Valor:"inicio_bloco_comercial/ fim_bloco_comercial"}, **listaCanaisTV**{Tipo:"ARRAY OF STRING", Valor:"id1, id2, ... idn"}).

2. **Início/Fim de comercial**

[**codRequisição**{Tipo:"STRING", Valor:"inicio_comercial/ fim_comercial"}, **idCanal**{Tipo:"STRING"}, **descrição** {Tipo:"TEXT"}, **horário**{Tipo:"DATE-TIME"}]

(**codRequisição**{Tipo:"STRING", Valor:"inicio_comercial/ fim_comercial"}, **idCanalTV**{Tipo:"STRING"}, **detalhes?** {Tipo:"BOOLEAN", Valor:"true/false"}).

3. **Início/Fim de programa**

[**codRequisição**{Tipo:"STRING", Valor:"inicio_programa/ fim_programa"}, **idCanal**{Tipo:"STRING"}, **descrição** {Tipo:"TEXT"}, **horário**{Tipo:"DATE-TIME"}]

(**codRequisição**{Tipo:"STRING", Valor:"inicio_programa/ fim_programa"}, **listaCanaisTV**{Tipo:"ARRAY OF STRING", Valor:"id1, id2, ... idn"}, **detalhes?** {Tipo:"BOOLEAN", Valor:"true/false"}).

4. **Pausa/Reinício de programa**

[**codRequisição**{Tipo:"STRING", Valor:"pausa_programa/ reinicio_programa"}, **idCanal**{Tipo:"STRING"}, **horário**{Tipo:"DATE-TIME"}]

(**codRequisição**{Tipo:"STRING", Valor:"pausa_programa/reinicio_programa"}, **listaCanaisTV**{Tipo:"ARRAY OF STRING", Valor:"id1, id2, ... idn"}).

Notificações em relação aos eventos de cenas

5. **Início/Fim de cena**

[**codRequisição**{Tipo:"STRING", Valor:"inicio_cena/ fim_cena"}, **idCanal**{Tipo:"STRING"}, **descrição** {Tipo:"TEXT"}, **horário**{Tipo:"DATE-TIME"}, **tipoCena**{Tipo:"STRING", Valor:"sexo/violência/crime..."}]

(**codRequisição**{Tipo:"STRING", Valor:"inicio_cena/ fim_cena"}, **tipoCena**{Tipo:"STRING", Valor:"sexo/violência/crime..."}, **listaCanaisTV**{Tipo:"ARRAY OF STRING", Valor:"id1, id2, ... idn"}, **detalhes?** {Tipo:"BOOLEAN", Valor:"true/false"}, **tempoAntecedencia**{Tipo:"INTEGER(em segundos)"}).

Notificações em relação aos eventos acionados pelo usuário

6. **RegistrarEventosNotifInterno**

[**codRequisição**{Tipo:"STRING", Valor:"registrar_eventos_notif_interno"}, **info**{Tipo:"STRING", Valor:"nível do volume/idCanal do destino/índice de brilho/..."}]

(**codRequisição**{Tipo:"STRING", Valor:"registrar_eventos_notif_interno"}, **tipoEvento**{Tipo:"STRING", Valor:"mudança de canal/volume/..."}).

7. **AutorizarEnvioEstadoPara**

[NULL]

(**codRequisição**{Tipo:"STRING", Valor:"autorizar_envio_estado_para"}, **idApp**{Tipo:"STRING"}, **listaUsuarios**{Tipo:"ARRAY OF STRING", Valor:"user1, user2, ... usern"}, **idServer**{Tipo:"URL"}).

8. **SolicitarMonitoramentoDe**

[**codRequisição**{Tipo:"STRING", Valor:"solicitar_monitoramento_de"}, **resposta**{Tipo:"ARRAY OF STRING", Valor: "aceita, rejeitada, não disponível, ..."}]

(**codRequisição**{Tipo:"STRING", Valor:" solicitar_monitoramento_de"}, **idApp**{Tipo:"STRING"}, **listaUsuarios**{Tipo:"ARRAY OF STRING", Valor:"user1, user2, ... usern"}, **idServer**{Tipo:"URL"}).

9. **EnviarMensagensGenéricas**

[NULL]

(**codRequisição**{Tipo:"STRING", Valor:"enviar_mensagens_genericas"}, **idApp**{Tipo:"STRING"}, **listaUsuarios**{Tipo:"ARRAY OF STRING", Valor:"user1, user2, ... usern"}, **idServer**{Tipo:"URL"}, **mensagem**{Tipo:"TEXT"}).

10. **Gerenciar comercial**

(10.1) Solicitar Fotos (snapshots do momento)

[**codRequisição**{Tipo:"STRING", Valor:"fotos_do_momento"}, **resposta**{Tipo:"ARRAY OF BYTES"}]

/*** Essa chamada deve retornar seis fotos de três em três segundos, por exemplo. Em cada foto retornada vem amarrado o ID, esse último é transparente para o *Kaeptor* realizar as demais tarefas relativas a foto selecionada/requisitada. ***/

(**codRequisição**{Tipo:"STRING", Valor:"fotos_do_momento"}, **idCanal**{Tipo:"STRING"}, **tempoRelogio**{Tipo:"INTEGER"}).

/*** Relógio (necessidade de um relógio comum entre a aplicação e o *backend* para cada canal com finalidade de gerenciar melhor a sincronização, pois pode ocorrer *delay* entre o momento da solicitação de um dado comercial pelo usuário e a posição dele no buffer do backend, por exemplo. Para resolver isso, pode-se estabelecer uma regra que consiste no *backend* sempre enviar dois comerciais (anterior e atual) para o usuário e esse último confirmar a sua escolha entre as duas possibilidades.) ***/

(10.2) Refina Fotos

[**codRequisição**{Tipo:"STRING", Valor:"refina_fotos"}, **resposta**{Tipo:"ARRAY OF BYTES"}]

/*** Essa chamada deve retornar três fotos antes e três depois da foto selecionada. ***/

(**codRequisição**{Tipo:"STRING", Valor:" refina_fotos"}, **idFoto**{Tipo:"STRING"}).

(10.3) Solicitar *token/ID* de um comercial/programa/...

[**codRequisição**{Tipo:"STRING", Valor:"token_comercial/ token_programa/..."},"}, **id**{Tipo:"STRING"}]

(**codRequisição**{Tipo:"STRING", Valor:"token_comercial/token_programa/..."}, **idFoto**{Tipo:"STRING"}).

(2) EXEMPLO DE USO

```
//1.Início/Fim de comercial
function inicio_comercial(handler){
	var notifData = {
		codRequisicao:"Inicio Comercial",
		idCanal:idCanal,
		detalhes:detalhes,
		acao: acao
	};
	Synchronization.bind(notifData, handler, "terceiro");
}
function fim_comercial(handler){
	var notifData = {
		codRequisicao:"Fim Comercial",
		idCanal:idCanal,
		detalhes:detalhes,
		acao: acao
	};
	Synchronization.bind(notifData, handler, "terceiro");
}
//2.Início/Fim de bloco comercial
function inicio_bloco_comercial(){
	var notifData = {
		codRequisicao:"inicio_bloco_comercial",
		listaIdCanais:listaIdCanais,
		acao: acao
	};
	Synchronization.bind(notifData, "terceiro");
}
function fim_bloco_comercial(){
	var notifData = {
		codRequisicao:"fim_bloco_comercial",
		listaIdCanais:listaIdCanais,
		acao: acao
	};
	Synchronization.bind(notifData, "terceiro");
}
//3.Início/Fim de programa
function inicio_programa(){
	var notifData = {
		codRequisicao:"inicio_programa",
		listaIdCanais:listaIdCanais,
		detalhes:detalhes,
		acao: acao
	};
	Synchronization.bind(notifData, "terceiro");
}
function fim_programa(){
	var notifData = {
		codRequisicao:"fim_programa",
		listaIdCanais:listaIdCanais,
		detalhes:detalhes,
		acao: acao
	};
	Synchronization.bind(notifData, "terceiro");

}
```

```
//4.Pausa/Reinício de programa
function pausa_programa(){
	var notifData = {
		codRequisicao:"pausa_programa",
		listaIdCanais:listaIdCanais,
		acao: acao
	};
	Synchronization.bind(notifData, "terceiro");
}
function reinicio_programa(){
	var notifData = {
		codRequisicao:"reinicio_programa",
		listaIdCanais:listaIdCanais,
		acao: acao
	};
	Synchronization.bind(notifData, "terceiro");
}
//5.Início/Fim de cena
function inicio_cena(){
	var notifData = {
		codRequisicao:"inicio_cena",
		tipoCena: tipoCena,
		listaIdCanais:listaIdCanais,
		detalhes: detalhes,
		tempoAntecedencia: tempoAntecedencia,
		acao: acao
	};
	Synchronization.bind(notifData, "terceiro");
}
function fim_cena(){
	var notifData = {
		codRequisicao:"fim_cena",
		tipoCena: tipoCena,
		listaIdCanais:listaIdCanais,
		detalhes: detalhes,
		tempoAntecedencia: tempoAntecedencia,
		acao: acao
	};
	Synchronization.bind(notifData, "terceiro");
}
//6.Registrar eventos de notificação interna
function registrar_eventos_notif_interno(){
	var notifData = {
		codRequisicao:"registrar_eventos_notif_interno",
		tipoEvento: tipoEvento,
		acao: acao
	};
	Synchronization.bind(notifData, "terceiro");
}
//7.Autorizar o envio de estado para...
function autorizar_envio_estado_para(){
	var notifData = {
		codRequisicao:"autorizar_envio_estado_para",
		idApp: idApp,
		listaUsuarios: listaUsuarios,
		idServer: idServer,
		acao: acao
	};
	Synchronization.bind(notifData, "terceiro");
}
```

```
//8.Solicitar monitoramento de...
function solicitar_monitoramento_de(){
	var notifData = {
		codRequisicao:"solicitar_monitoramento_de",
		idApp: idApp,
		listaUsuarios: listaUsuarios,
		idServer: idServer,
		acao: acao
	};
	Synchronization.bind(notifData, "terceiro");
}
//9.Enviar Mensagens Genéricas...
function enviar_mensagens_genericas(){
	var notifData = {
		codRequisicao:"enviar_mensagens_genericas",
		idApp: idApp,
		listaUsuarios: listaUsuarios,
		idServer: idServer,
		mensagem: mensagem,
		acao: acao
	};
	Synchronization.bind(notifData, "terceiro");
}
//10.Solicitar fotos (snapshots do momento)...
function solicitar_fotos_do_momento(){
	var notifData = {
		codRequisicao:"fotos_do_momento",
		idCanal: idCanal,
		tempoRelogio: tempoRelogio,
		acao: acao
	};
	Synchronization.bind(notifData, "terceiro");
}
//11.Refina fotos...
function refinar_fotos(){
	var notifData = {
		codRequisicao:"refina_fotos",
		idFoto: idFoto,
		acao: acao
	};
	Synchronization.bind(notifData, "terceiro");
}
//12.Solicitar token/id de um comercial/programa...
function solicitar_token(){
	var notifData = {
		codRequisicao:"token_evento",
		idFoto: idFoto,
		acao: acao
	};
	Synchronization.bind(notifData, "terceiro");
}
```

Anexo B

TABELA PADRÃO DE DISTRIBUIÇÃO DE PROBABILIDADE ESTATÍSTICA T DE STUDENT

TABELA DA DISTRIBUIÇÃO *t* DE STUDENT - VALORES DE t TAIS QUE P(-t ≤ T ≤ t) = 1 - α ou P(|T| > t) = α (BICAUDAL). Para UNICAUDAL, 2xP(T < -t) = α ou 2xP(T > t) = α

gl \ α	0,500 (50%)	0,250 (25%)	0,200 (20%)	0,100 (10%)	0,050 (5%)	0,025 (2,5%)	0,020 (2,0%)	0,010 (1,0%)	0,005 (0,5%)
1	1,0000	2,4142	3,0777	6,3138	12,7062	25,4517	31,8205	63,6567	127,3213
2	0,8165	1,6036	1,8856	2,9200	4,3027	6,2053	6,9646	9,9248	14,0890
3	0,7649	1,4226	1,6377	2,3534	3,1824	4,1765	4,5407	5,8409	7,4533
4	0,7407	1,3444	1,5332	2,1318	2,7764	3,4954	3,7469	4,6041	5,5976
5	0,7267	1,3009	1,4759	2,0150	2,5706	3,1634	3,3649	4,0321	4,7733
6	0,7176	1,2733	1,4398	1,9432	2,4469	2,9687	3,1427	3,7074	4,3168
7	0,7111	1,2543	1,4149	1,8946	2,3646	2,8412	2,9980	3,4995	4,0293
8	0,7064	1,2403	1,3968	1,8595	2,3060	2,7515	2,8965	3,3554	3,8325
9	0,7027	1,2297	1,3830	1,8331	2,2622	2,6850	2,8214	3,2498	3,6897
10	0,6998	1,2213	1,3722	1,8125	2,2281	2,6338	2,7638	3,1693	3,5814
11	0,6974	1,2145	1,3634	1,7959	2,2010	2,5931	2,7181	3,1058	3,4966
12	0,6955	1,2089	1,3562	1,7823	2,1788	2,5600	2,6810	3,0545	3,4284
13	0,6938	1,2041	1,3502	1,7709	2,1604	2,5326	2,6503	3,0123	3,3725
14	0,6924	1,2001	1,3450	1,7613	2,1448	2,5096	2,6245	2,9768	3,3257
15	0,6912	1,1967	1,3406	1,7531	2,1314	2,4899	2,6025	2,9467	3,2860
16	0,6901	1,1937	1,3368	1,7459	2,1199	2,4729	2,5835	2,9208	3,2520
17	0,6892	1,1910	1,3334	1,7396	2,1098	2,4581	2,5669	2,8982	3,2224
18	0,6884	1,1887	1,3304	1,7341	2,1009	2,4450	2,5524	2,8784	3,1966
19	0,6876	1,1866	1,3277	1,7291	2,0930	2,4334	2,5395	2,8609	3,1737
20	0,6870	1,1848	1,3253	1,7247	2,0860	2,4231	2,5280	2,8453	3,1534

Figura 24: Tabela Padrão de Distribuição t-test *[Fonte:online Statistic, 2014]*