Pós-Graduação em Ciência da Computação

“XDTv: um Método Ágil para o Desenvolvimento de Aplicações para TV Digital”

Por

Mario Godoy Neto

Tese de Doutorado

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE, FEVEREIRO/2014

UNIVERSIDADE FEDERAL DE PERNAMBUCO

CENTRO DE INFORMÁTICA

PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

MARIO GODOY NETO

"XDTv: um Método Ágil para o Desenvolvimento de Aplicações para TV Digital"

ESTE TRABALHO FOI APRESENTADO À PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO DO CENTRO DE INFORMÁTICA DA UNIVERSIDADE FEDERAL DE PERNAMBUCO COMO REQUISITO PARCIAL PARA OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIA DA COMPUTAÇÃO.

ORIENTADOR: CARLOS ANDRÉ GUIMARÃES FERRAZ

RECIFE, FEVEREIRO/2014

Catalogação na fonte
Bibliotecária Joana D'Arc L. Salvador, CRB 4-572

Godoy Neto, Mario.
XDTV: um método ágil para o desenvolvimento de aplicações para TV digital / Mario Godoy Neto. – Recife: O Autor, 2014.
227 f.: fig., quadro.

Orientador: Carlos André Guimarães Ferraz.
Tese (Doutorado) - Universidade Federal de Pernambuco. CIN. Ciência da Computação, 2014.
Inclui referências e apêndices.

1.Engenharia de software. 2. Desenvolvimento ágil de software. 3. Televisão digital. I. Ferraz, Carlos André Guimarães (orientador). II. Título.

005.1 (22. ed.) MEI 2014-76

Tese de Doutorado apresentada por **Mario Godoy Neto** à Pós Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título "**XDTv: Um Método Ágil para o Desenvolvimento de Aplicações para TV Digital**" orientada pelo **Prof. Carlos André Guimarães Ferraz** e aprovada pela Banca Examinadora formada pelos professores:

_____

Prof. Fernando da Fonseca de Souza

Centro de Informática / UFPE

_____

Prof. Alexandre Marcos Lins de Vasconcelos

Centro de Informática / UFPE

_____

Profa. Thais Vasconcelos Batista

Departamento de Informática e Matemática Aplicada/UFRN

_____

Profa. Tatiana Aires Tavares

Laboratório de Aplicações de Vídeo Digital / UFPB

_____

Prof. Fernando Antonio Mota Trinta

Departamento de Computação / UFC

Visto e permitida a impressão.

Recife, 27 de fevereiro de 2014.

_____

Profa. Edna Natividade da Silva Barros

Coordenadora da Pós-Graduação em Ciência da Computação do

Centro de Informática da Universidade Federal de Pernambuco.

Todo o conhecimento por mim adquirido através dos métodos formais de ensino não teria validade alguma sem as lições de vida passadas pelos meus amados pais, que certamente, representam os mais valiosos ensinamentos.

Agradecimentos

Agradeço a Deus por estar sempre ao meu lado, possibilitando que eu tenha as condições físicas e psicológicas para iniciar e concluir esta pesquisa ao longo de cinco anos.

Agradeço toda minha família, em especial os meus pais, Mario e Regina, que sempre me apoiaram em todos os aspectos da vida, incentivando-me pelos caminhos que optei trilhar, e servindo constantemente como espelho para um crescimento moral, sem os quais seria impossível concluir mais esta importante etapa da vida. Agradeço, ainda, às minhas irmãs, Maíra e Simone, pelo carinho e companheirismo de sempre, e a minha linda sobrinha Iara. A vocês, acompanha também um pedido de desculpas pela ausência do filho, irmão e tio, agora, sertanejo.

Agradeço aos grandes amigos: Tainara, Andrezza, Fabrício Soares, Ricardo Argenton, Thiago Amaral, pela ajuda, carinho e atenção durante os dias difíceis. Agradeço às turmas de Tópicos Avançados em Engenharia de Software 2011 e 2012 (UNIVASF), e às valiosas contribuições da banca examinadora, destacando a contribuição ímpar do professor Sérgio Soares (UFPE).

Agradeço à CAPES, ao projeto RH-TVD e à Universidade Federal do Vale do São Francisco - UNIVASF, pelo apoio e incentivo a esta pesquisa.

Por fim, mas não menos especial, agradeço profundamente ao meu orientador, Carlos André Guimarães Ferraz, que sempre se mostrou disponível e acessível, auxiliando-me prontamente durante as dificuldades acadêmicas, bem como as extracurriculares, encontradas durante este percurso. Muito obrigado pela confiança em mim depositada e, acima de tudo, pela crença no meu empenho e capacidade. Valeu Carlinhos!

Espero, sinceramente, continuar e aprofundar nossas valiosas parcerias após a conclusão desta pesquisa.

Resumo

Nos últimos tempos, têm surgido cada vez mais dispositivos computacionais com aplicabilidades diversas. *Smartphones* têm propósitos distintos de *tablets*, que têm propósitos diferentes de aparelhos de TV, que se apresentam como plataformas de computação não-convencional. Tais distinções apresentam, em especial, implicações merecedoras de estudos sobre o desenvolvimento de software para os mais diferentes dispositivos. Aplicações de software para TV Digital (TVD), por exemplo, possuem peculiaridades que demandam atenção especial em seu processo de desenvolvimento, tais como: coleta de requisitos referentes ao conteúdo multimídia (dimensão, tempo de exibição, posição e sincronismo) e a necessidade de um rápido desenvolvimento das aplicações, ou seja, o mais próximo possível do tempo de produção do conteúdo audiovisual da TV associado à aplicação, como: telejornais, notícias de última hora, publicidade entre outros. A presente pesquisa pretende apontar um método de desenvolvimento de software mais rápido e adequado às peculiaridades das aplicações de TVD. Para isso, através do levantamento bibliográfico foram estudados os métodos ágeis mais utilizados Scrum, *eXtreme Programming* (XP) e um método híbrido, formado por ambos. Visando avaliar o desempenho de tais métodos, foi realizado o primeiro experimento controlado. Em sequência, a adequação de tais métodos foi novamente avaliada através de um *survey* com profissionais experientes em TVD. Os resultados obtidos foram analisados através da técnica multicritério de apoio à decisão, que apontaram indícios que o método híbrido é o mais adequado, porém, existem pontos de melhoria que podem aprimorar o processo de desenvolvimento. Os resultados do primeiro experimento associados ao *survey* viabilizaram a especificação e customização de um novo método híbrido, proposto pela presente tese, denominado *eXtreme Digital Television* (XDTv) que compartilha da filosofia ágil para auxiliar no tratamento das peculiaridades do ambiente de TVD. Um segundo experimento avaliou o desempenho do método XDTv e o comparou com o método híbrido (Scrum/XP) que foi melhor avaliado no primeiro experimento. Os dados dos experimentos foram coletados sob diferentes perspectivas e as análises permitem concluir que o método XDTv apresentou melhor desempenho, revelando-se mais rápido e mais adequado ao desenvolvimento de aplicações para TV Digital.

Palavras-chaves: engenharia de software, métodos ágeis, TV digital.

Abstract

Lately, more and more computational devices with different applications have emerged. Smartphones have distinct purposes than tablets, which have different purposes than TV sets, that present themselves as non-conventional computational platforms. Such distinctions have, specially, implications deserving studies about software development for different devices. Software applications for Digital TV (DTV), for instance, have peculiarities demanding special attention on their development process, such as: collecting requirements concerning the multimedia content (dimension, time of exhibition, position and synchronism) and the need of a fast development of applications, in other words, as close as possible to the production time of audiovisual content for TV associated with the application, such as: newscasts, breaking news, publicity and others. This research intends to indicate the software development method that is faster and adequate for the peculiarities of DTV applications. For this, a bibliographic survey was used to collect the mostly used Agile methods, Scrum and eXtreme Programming (XP) and a hybrid method, composed by both. Aiming to evaluate the performance of those methods, the first controlled experiment was performed. In sequence, the suitability of those methods was evaluated again by a survey with experienced professionals in DTV. The results gathered were analyzed using a multi-criteria technique of decision support, which showed evidence that the hybrid method is the most adequate one. However, there are improving points that may enhance the development process. The results from the first experiment associated with the survey enabled the specification of another hybrid method, proposed by this thesis, called eXtreme Digital Television (XDTv) that shares the Agile philosophy to help treating peculiarities of the DTV environment. A second experiment evaluated the performance of XDTv method and compared it with the hybrid method (Scrum/XP) which was evaluated in the first experiment. Data from experiments were collected under different perspectives and the analysis allowed to conclude that the XDTv method had better performance, showing to be faster and more adequate for the development of Digital TV applications.

Keywords: software engineering, agile methods, digital TV.

Lista de Figuras

Lista de Gráficos

Lista de Quadros

Abreviaturas

IDE	Integrated Development Environment ou Ambiente de Desenvolvimento Integrado
ISDB-T	Integrated Services Digital Broadcasting-Terrestrial ou Serviços Integrados de Radiodifusão Digital - Terrestre.
ISDB-Tb	Integrated Services Digital Broadcasting-Terrestrial Brazil ou Serviços Integrados de Radiodifusão Digital - Terrestre Brasil.
ITU	International Telecommunications Union ou União Internacional de Telecomunicações
MFSM	Modelo de Fluxo e Sincronismo de Mídia
NCL	Nested Context Language ou Linguagem de Contexto Aninhada
PATv	Protótipo para Aplicações de Televisão
SBTVD	Sistema Brasileiro de TV Digital
TVD	Televisão Digital
UHF	Ultra High Frequency
XDTv	eXtreme Digital Television
XP	eXtreme Programming

SUMÁRIO

CAPÍTULO 1.

1. INTRODUÇÃO

"O homem torna-se o que ele trabalha em si."
Divaldo Franco

Desde a primeira transmissão televisiva no Brasil, em 18 de setembro de 1950, viabilizada por Assis Chateaubriand, a interação dos telespectadores com o conteúdo audiovisual da TV estava limitada a cartas ou ligações telefônicas. No entanto, tal característica está em processo de transformação com a chegada da TV Digital interativa (TVDi[1]), pois, além do áudio e vídeo de alta definição, é oferecido também uma nova plataforma computacional capaz de executar aplicações.

As aplicações para TVD apresentam peculiaridades que demandam atenção especial em seu processo de desenvolvimento, como: velocidade, agilidade em atender modificações, coleta de requisitos do conteúdo multimídia e a forte demanda por manipulação e sincronização destes conteúdos. Existem diferentes definições que pretendem conceituar a característica multimídia. A presente tese segue o conceito definido em (CHAPMAN; CHAPMAN, 2004) onde se considera que um conteúdo multimídia é produzido através da combinação de pelo menos um tipo de mídia dinâmica, como: vídeo, áudio ou animação, com pelo menos um tipo de mídia estática, como: texto, gráficos ou imagens.

[1] A partir desse momento, a sigla TVD será adotada para indicar TV Digital interativa (TVDi)

Visando auxiliar no trato de tais peculiaridades, o presente trabalho especificou o método híbrido, customizado para o ambiente de TVD, denominado *eXtreme Digital Television* (XDTv) que compartilha da filosofia ágil, cujo objetivo é oferecer um método rápido e adequado ao contexto das aplicações de TVD.

Os três principais padrões de TVD são: o Norte-Americano, denominado *Advanced Television System Committee* (ATSC) (ATSC, 2006; SHELLHAMMER, 2008; SONG; PARK, 2006); o Europeu, denominado *Digital Video Broadcasting* (DVB), ou Transmissão de Vídeo Digital (ETSI, 2005); e o Nipo-Brasileiro, denominado *Integrated Services Digital Broadcasting-Terrestrial Brazil* (ISDB-Tb), ou, Serviços Integrados de Radiodifusão Digital - Terrestre Brasil (ABNT NBR, 2007; PAIVA, 2010; Soares, 2010).

O fácil acesso às ferramentas *open source* utilizadas para montar um laboratório de desenvolvimento de aplicações para TVD, necessário para realizar os experimentos necessários, associadas à crescente demanda por aplicações e técnicas que auxiliam esse mercado (FERNANDES; TAVARES; PÔRTO, 2011; GODOY NETO; FERRAZ, 2012; LULA; GUIMARÃES; TAVARES, 2011; QUINTERO et al., 2013;), foram os principais fatores que motivaram a presente tese a adotar o ISDB-Tb como ambiente de TVD utilizado para o desenvolvimento e execução dos experimentos realizados. No entanto, as contribuições desta tese não se restringem ao sistema ISDB-Tb, pois podem ser aplicados para os demais padrões (ATSC ou DVB).

No Sistema Brasileiro de Televisão Digital (SBTVD) é adotado o padrão ISDB-Tb, no qual, podem existir diversos aparelhos receptores do sinal digital, como: TV tradicional (domiciliar), TV móvel (metrô, taxi, ônibus, entre outros), dispositivos *móbile* (celulares, *smart phones*, entre outros), computadores pessoais, entre outros dispositivos. Cada aparelho receptor possui uma plataforma computacional que se assemelha aos computadores tradicionais, com dispositivos de entrada e saída de dados, memória, processador, barramentos e um sistema operacional responsável por seu gerenciamento. A escolha do sistema operacional fica a critério do fabricante do aparelho receptor. Dessa forma, é possível que cada marca e modelo de televisor adote

um sistema operacional distinto, o que dificultaria o trabalho dos desenvolvedores de aplicações se não fosse incorporado ao sistema de TVD um *middleware*, como o Ginga[2], adotado pelo SBTVD (BARBOSA; SOARES, 2008; SOARES et al., 2010), o Multimedia Home Platform (MHP)[3] adotado pelo padrão DVB (SANZ, 2012), ou o DTV Application Software Environmen (DASE) adotado pelo padrão ATSC (SHELLHAMMER, 2008).

Nesse contexto, o *middleware* funciona como uma camada intermediária, viabilizando uma interface padrão de comunicação entre a aplicação e os aparelhos receptores (SANZ, 2012). Dessa forma, os desenvolvedores de aplicações para TVD não precisam se preocupar com os distintos sistemas operacionais adotados pelos fabricantes, pois a aplicação é executada sobre o *middleware*, independente do sistema operacional adotado.

O *middleware* Ginga disponível nos televisores atuais é capaz de executar aplicações interativas desenvolvidas através da linguagem declarativa *Nested Context Language* (NCL)[4] e da linguagem de script Lua[5].

Outra vantagem do *middleware* Ginga refere-se ao seu reconhecimento como um padrão pela *International Telecommunication Union* (ITU), que pode ser utilizado também para serviços de IPTV, ou seja, transmissão de sinal de TV através do Internet Protocolo (IP), ou ainda através da TV conectada à Internet. A recomendação ITU-T H.761 fornece a especificação do NCL e do Ginga-NCL para a interoperabilidade entre os *frameworks* de aplicação multimídia de IPTV (ITU, 2011), aumentando assim sua aplicabilidade e comunicação com outras plataformas computacionais. Existe, ainda, uma tendência em adicionar a linguagem Java através do Ginga-J, porém, este padrão ainda não está certificado. Por esse motivo, o desenvolvimento de aplicações através do Ginga-J não faz parte do escopo da presente tese.

Independente da linguagem a ser utilizada, faz-se necessário o estudo de processos e métodos de engenharia de software adequados às peculiaridades das aplicações para TVD. Segundo Sommerville (2011), o fator

[2] www.ginga.org.br
[3] www.mhp.org
[4] www.ncl.org.br
[5] www.lua.org

mais significante para essa escolha do método de desenvolvimento é o tipo da aplicação que será desenvolvida, cabendo ao engenheiro de software a escolha do método mais adequado.

Diferentes autores adotam conceitos distintos para “método de desenvolvimento” e “processo de software”. Para tal, o presente trabalho segue a nomenclatura de Sommerville (2011), onde um processo de software é uma abordagem sistemática e organizada utilizada para auxiliar o desenvolvimento de software, composta por quatro atividades básicas: especificação, desenvolvimento, validação e evolução. Como exemplos, podem ser destacados os processos baseados em planos ou em agilidade.

O “método de desenvolvimento”, chamado por alguns autores de “metodologia” de desenvolvimento, representa uma abstração do processo de software, não pretende ser uma descrição definitiva, pois pode ser adaptado a outras realidades (SOMMERVILLE, 2011). Cada método representa uma perspectiva particular de um processo. Estes podem ser ampliados ou adaptados para criar processos mais específicos, conforme é apresentado no Capítulo 2.

A literatura apresenta vários métodos de desenvolvimento, onde cada método concentra seus esforços em auxiliar diferentes demandas do processo de desenvolvimento, normalmente são voltados para um determinado tipo de software, como: aplicações personalizadas, vendidos em larga escala (software de prateleira), embarcados, *stand-alone*, acesso remoto, entre outros. Tais aplicações podem ser executadas sobre diferentes plataformas computacionais.

Por esse motivo, é necessário cautela ao escolher o método que será adotado para o desenvolvimento, visto que diferentes tipos de software possuem demandas específicas, sendo imprescindível verificar a capacidade do método em atender às necessidades do projeto que será desenvolvido.

Adotar um método não apropriado pode, entre outros problemas: a) dificultar a comunicação com o cliente; b) prejudicar a coleta de requisitos e inviabilizar as alterações dos mesmos; e c) atrasar a entrega do software gerando transtornos irreparáveis que podem acarretar em cancelamento do

projeto. Neste contexto, a **Figura 1.1** apresentada alguns resultados do relatório Standich Group (CHAOS, 2013), onde são contabilizados os projetos de software finalizados nas seguintes condições: a) Sucesso (*Succeeded*) – dentro do prazo e orçamento previsto; b) Fracassado (*Failed*) – cancelados antes de serem finalizados; c) Modificados (*Challenged*) – finalizados com atraso, acima do orçamento.

Figura 1.1 – Projetos de sucesso, cancelado e modificados

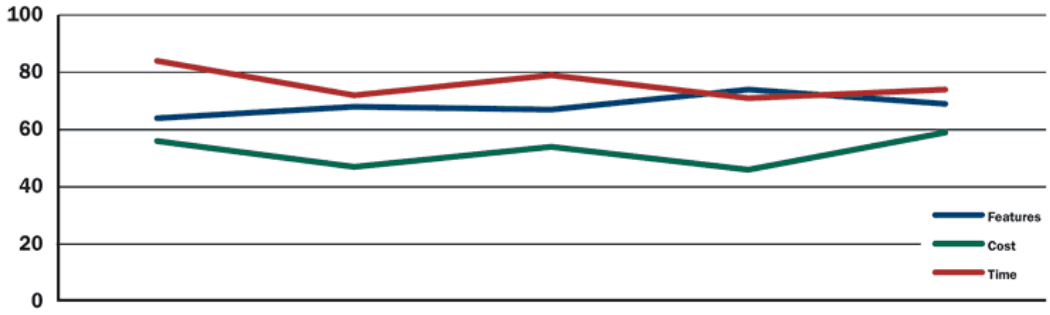
	2004	2006	2008	2010	2012
Successful	29%	35%	32%	37%	39%
Failed	18%	19%	24%	21%	18%
Challenged	53%	46%	44%	42%	43%

Fonte: CHAOS, 2013.

Ao observar a **Figura 1.1** é possível notar que, apesar dos crescentes casos de êxito entre os anos 2004 e 2012, apenas 39% de todos os projetos de software finalizados cumpriram o prazo, o orçamento, e apresentavam os recursos e funcionalidades solicitadas (*successful*). Porém, o restante, 61% dos projetos foram desenvolvidos diferente do planejado, sendo que, 43% chegaram ao final acima do orçamento, e / ou com menos recursos e funcionalidades necessárias (*challenged*). Os 18% restantes fracassaram, ou seja, foram cancelados antes da entrega (*failed*).

A **Figura 1.2** apresenta o tempo (*time*) e o custo (*cost*) adicional, bem como o percentual de funcionalidades (*features*) entregues, que tornaram os projetos (*challenged*) diferentes do solicitado/esperado.

Figura 1.2. – Tempo e custo ultrapassados e funcionalidades entregues



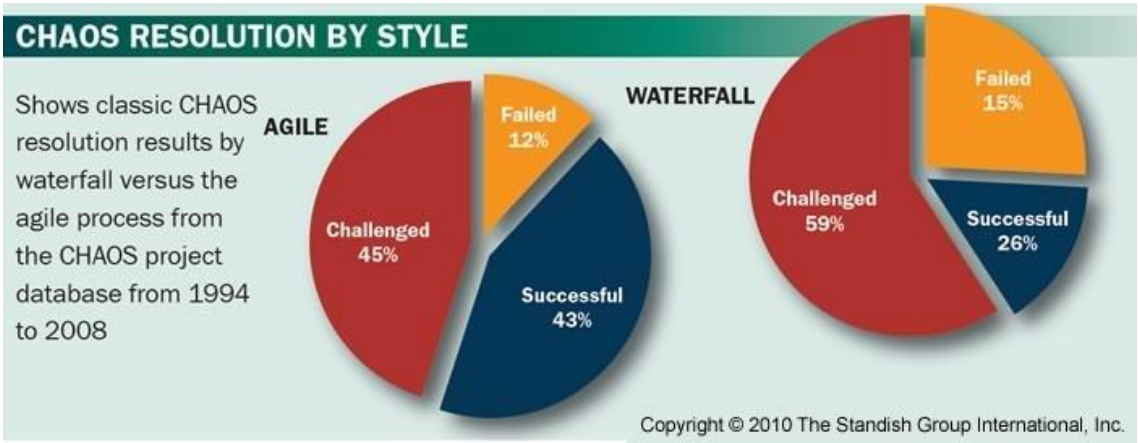
	2004	2006	2008	2010	2012
TIME	84%	72%	79%	71%	74%
COST	56%	47%	54%	46%	59%
FEATURES	64%	68%	67%	74%	69%

Fonte: CHAOS, 2013.

A **Figura 1.2** apresenta que, mesmo reduzindo as funcionalidades (*features*) solicitadas, a extrapolação do tempo (*time*) é um problema constante, relacionado ao aumento do custo (*cost*) do projeto.

A **Figura 1.3** apresenta uma comparação entre os projetos de 1994 até 2008 que adotaram métodos ágeis (*agile*) e aqueles que adotaram o método Cascata (*Waterfall*).

Figura 1.3 – Comparação entre os métodos ágeis e cascata



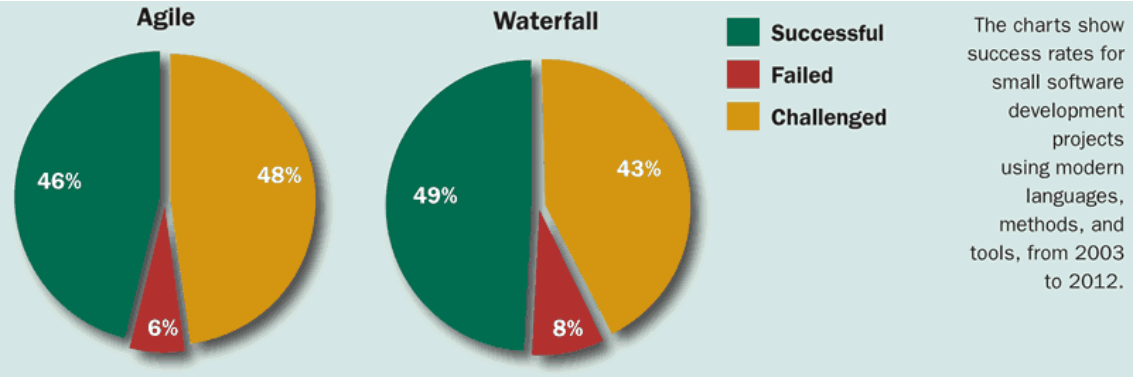
Fonte: CHAOS, 2010.

A **Figura 1.3** aponta que, no período de 1994 até 2008, os projetos que adotaram métodos ágeis obtiveram melhores resultados nos três aspectos

observados, com menor quantidade de fracassos, menor quantidade de modificações em seu planejamento inicial, e, o mais relevante, uma quantidade significativamente maior de projetos que obtiveram sucesso.

O último relatório disponibilizado pelo Standich Group considerou apenas os projetos entre 2003 e 2012. Nesse período os resultados sobre os métodos ágeis e Cascata foram mais homogêneos, apesar do método Cascata apresentar 3% a mais de sucesso, apresentou também 2% a mais de projetos fracassados, conforme apresenta a **Figura 1.4**.

Figura 1.4 – Comparação entre os métodos ágeis e Cascata



Fonte: CHAOS, 2013.

A Figura 1.4 aponta que os resultados obtidos em 2010 e 2012 diminuíram a vantagem dos métodos ágeis, que se mantinha elevada desde a primeira pesquisa realizada em 1994. No entanto, os projetos Cascata fracassaram (*failed*) 33,33% a mais que os métodos ágeis, apresentando dessa forma, leve vantagem para os métodos ágeis.

A pesquisa do Standish Group revela ainda que a presença do cliente/usuário é um fator relevante, implicando diretamente no sucesso do projeto. Quando esta relação é falha, há alto índice de fracasso. A pesquisa aponta que um dos principais motivos que implicam na falta de participação está relacionado à quantidade de tempo que o cliente precisa se distanciar da sua rotina de trabalho, gerando dificuldades em agendar reuniões que respeitem seus compromissos.

Os clientes das aplicações de TVD, são aqueles que conhecem as demandas dessas aplicações, normalmente, são profissionais da área de TV,

como: produtores, roteiristas, diretores, equipe de publicidade e propaganda, entre outros.

Segundo CHAOS (2010), o time de desenvolvimento necessita de *feedback*, revisão de requisitos, pesquisa básica, prototipagem e outros recursos para construção do consenso sobre o projeto. Isso indica que o planejamento e a escolha do método de desenvolvimento adequado são primordiais, possibilitando evitar ou minimizar o fracasso do projeto de software.

O estouro do prazo de entrega e do orçamento são os motivos mais comuns de cancelamento de projetos. Existem algumas atividades que influenciam diretamente nesse problema, como por exemplo, as falhas na coleta de requisitos. Minimizar a diferença entre os requisitos solicitados pelo cliente e aqueles implementados pelos desenvolvedores, é fundamental para o sucesso do produto final. Caso ocorram divergências, é preciso que estas sejam localizadas e corrigidas o mais rápido possível, diminuindo o tempo e o custo do software.

Segundo Pressman (2011), uma alteração com custo de “1x” realizada na atividade de coleta e especificação de requisitos terá custo de “1,5x” até “6x” se realizada na fase de implementação, e de “60x” até “100x” se realizada na fase de manutenção do software. Mudanças nos requisitos podem tornar os artefatos planejados, projetados, especificados e executados, obsoletos ou desatualizados (ABRANTES; TRAVASSOS, 2012). Dessa forma, quanto mais rápido ocorrer a detecção de falhas durante o processo de desenvolvimento, menor será o custo e o tempo dedicado para sua correção (SCHACH, 2009).

A forma como cada método define as atividades do processo de desenvolvimento é fundamental para minimizar o impacto gerado pelas solicitações de alterações de implementação. Alguns métodos são mais rígidos, dirigidos a planos, não preveem modificações em seus requisitos, outros mais flexíveis, permitindo alterar os requisitos sem, necessariamente, acarretar no atraso da entrega e na elevação do custo do produto.

A demanda por flexibilidade levou a comunidade científica a elaborar diversos métodos com esse propósito. Em 2001, renomados membros da

comunidade científica formalizaram o Manifesto Ágil (BECK et al., 2001), estabelecendo princípios norteadores fundamentais para definir as características dos métodos considerados ágeis. Entre os métodos ágeis existentes na literatura, os métodos Scrum e XP são os mais adotados para o desenvolvimento de software (VERSIONONE, 2009, 2010, 2011, 2013).

Empresas que operam em um ambiente que sofre alterações inesperadas passam por dificuldades para a obtenção de forma completa e estável dos requisitos e suas prioridades no início do projeto, o que inviabiliza um modelo dirigido a planos para tais empresas (SOMMERVILLE, 2011).

A dissertação apresentada por Asfora (2009) ressalta que a priorização dos requisitos de software é uma atividade crítica no processo de desenvolvimento, pois uma decisão equivocada pode afetar a qualidade e a aceitação do produto final pelo cliente. O desenvolvimento ágil contribui para a atividade de priorização de requisitos, porém, segundo o autor, as orientações oferecidas pelos métodos Scrum e XP são básicas (ASFORA, 2009).

1.1. Justificativa

A demanda por velocidade e agilidade no desenvolvimento de aplicações para TVD torna-se cada vez mais clara na literatura. Trabalhos científicos já destacam a necessidade de utilizar métodos ágeis para produção do conteúdo audiovisual (BELLOTTI et al., 2011; FERNANDES; TAVARES; PÔRTO, 2011; LULA; GUIMARÃES; TAVARES, 2011; MARQUES NETO, 2011; MATTOS; SILVA; MUCHALUAT-SAADE, 2013; ROBERTO; FER; BOTELHO, 2008; SAITO et al., 2011; VEIGA, 2006),

O desenvolvimento de aplicações interativas associadas ao conteúdo audiovisual também já possui resultados na literatura especializada (BELLOTTI et al., 2011; FERNANDES; TAVARES; PÔRTO, 2011; FERREIRA; KULESZA; SOUZA FILHO, 2012; LULA; GUIMARÃES; TAVARES, 2011), conforme publicado em trabalhos anteriores (GODOY NETO; FERRAZ, 2011, 2012, 2013a, 1013b). No entanto, a literatura não apresenta métricas suficientes para realizar uma análise comparativa sobre métodos de desenvolvimento adequado ao contexto de aplicações para TVD.

Os trabalhos relacionados não apresentam a especificação de um método customizado e estruturado para auxiliar na especificação de requisitos multimídia e na definição de um modelo de processo de desenvolvimento de aplicações. Tais carências configuram um dos principais problemas referente ao desenvolvimento de aplicações para TVD. A falta de um método adequado implica no aumento do tempo necessário para o desenvolvimento de aplicações multimídias (MARQUES NETO, 2011). No entanto, Marques Neto (2011) avalia que, devido a abordagem menos formal dos métodos ágeis, estes são fortes candidatos ao desenvolvimento de aplicações multimídia.

Apesar de concordar que os métodos ágeis sejam fortes candidatos ao desenvolvimento de aplicações para TVD, a presente tese não considera que os métodos ágeis disponíveis na literatura sejam perfeitamente adaptados ao desenvolvimento de aplicações para TVD, pois, segundo Sommerville (2011), a escolha do método precisa levar em consideração o projeto e o tipo de aplicação que se pretende desenvolver.

Não existem métodos de desenvolvimento universais, diferentes tipos de software exigem abordagens diferentes (SOMMERVILLE, 2011). Desta forma, não se pode afirmar que o método “A” é melhor que o método “B”, pois cada método pode ser adequado a diferentes tipos de projetos de software (SOMMERVILLE, 2011). O tipo de aplicação a ser desenvolvida determina qual método será mais adequado para seu desenvolvimento. Utilizar um método não adequado pode diminuir a qualidade, aumentar o tempo do desenvolvimento e elevar o custo e o preço final do software.

No entanto, ao analisar a filosofia ágil a presente pesquisa considera que esta satisfaz algumas das peculiaridades inerentes às aplicações de TVD, como a agilidade e flexibilidade quanto a alterações de requisitos, a entrega rápida de software funcionando, a comunicação e a autonomia para a tomada de decisão. Dessa forma, é possível que os métodos ágeis possam contribuir para seu processo de desenvolvimento. Diante disso, é fundamental realizar experimentos que mensurem as variáveis “tempo” e “adequação” dos principais métodos ágeis utilizados pela indústria de TVD.

Os princípios ágeis sugerem que os usuários possam utilizar módulos funcionais da aplicação antes de sua implementação completa, no entanto, esta prática é uma das diferenças entre as aplicações tradicionais e as aplicações de TVD, o custo elevado da transmissão das aplicações de TVD, e o acesso imediato dos usuários finais (leigos), tornam inviável a transmissão de apenas parte dos requisitos funcionais.

Atualmente, o poder computacional da infraestrutura de hardware é limitado se comparado às demais plataformas (QUINTERO et al., 2013). A escolha das aplicações que serão transmitidas fica a critério das emissoras, que as distribui via rádio difusão *Ultra High Frequency* (UHF) de forma unilateral, em broadcast, para todos os televisores e demais aparelhos receptores. A maior parte da banda disponível para cada emissora é reservada para transmissão de áudio e vídeo, e apenas uma pequena parte da banda é reservada para a transmissão de dados, acarretando em limitações que influenciam no desenvolvimento de tais aplicações (QUINTERO et al., 2013).

O público alvo de tais aplicações está diretamente relacionado à transmissão *broadcast*, pois todos os aparelhos sintonizados no mesmo canal receberão as mesmas aplicações, deixando a critério dos vários tipos de usuários aceitarem ou não sua execução. Uma das características históricas da TV e de seus telespectadores é o foco no entretenimento e disseminação de informações (BELLOTTI et al., 2011). Diante disso, as aplicações de TVD não ocupam o papel principal no ambiente de TV. São coadjuvantes, com o objetivo de agregar valor ao conteúdo audiovisual transmitido.

O curto ciclo de vida da aplicação está relacionado à necessidade de transmissão e sincronismo com o conteúdo audiovisual produzido pela TV, que pode durar poucos minutos e ser veiculado durante poucos dias. Essa é uma das características que diferenciam aplicações de TVD das demais plataformas, nas quais, normalmente, é desejável que seu ciclo de vida seja o mais longo possível, passando por diversas etapas de evoluções e aprimoramentos. Tal característica implica no fato dos métodos de desenvolvimento apresentarem forte preocupação com a evolução e

manutenção do software, fato este que não reflete a realidade de aplicações para TVD (GODOY NETO; FERRAZ, 2012).

Características como o curto ciclo de vida, os limitados recursos de hardware e de interação, o contexto dos usuários e o papel coadjuvante das aplicações de TVD, implicam que grande parte destas aplicações seja simples, ou seja, apresente poucos requisitos funcionais. A pequena quantidade de requisitos funcionais reduz a prioridade da realização solicitações de mudança (*change request*).

A demanda por desenvolvimento em um curto espaço de tempo é característica marcante da TVD, pois, algumas aplicações devem acompanhar a velocidade com que o conteúdo áudio visual é produzido pela emissora como: telejornais, notícias de última hora, campanhas publicitárias, entre outros (FERNANDES; TAVARES; PÔRTO, 2011; GODOY NETO; FERRAZ, 2011, 2012; LULA; GUIMARÃES; TAVARES, 2011; PEQUENO et al., 2010; QUINTERO et al., 2013; VEIGA, 2006; VEIGA; TAVARES, 2006, 2007).

O desenvolvimento de conteúdos interativos de TVD envolve diferentes perfis profissionais, formados em diferentes áreas de conhecimento, desde diretor de arte, diretor de áudio, roteiristas, produtor, designer, atores, diferentes artistas, jornalistas e também engenheiros de software. Para isso, a comunicação entre os membros do time deve ser facilitada, de modo a ser clara e objetiva (BELLOTTI et al., 2011; FERNANDES; TAVARES; PÔRTO, 2011; GODOY NETO; FERRAZ, 2012; LULA; GUIMARÃES; TAVARES, 2011; QUINTERO et al., 2013; VEIGA, 2006; VEIGA; TAVARES, 2006, 2007).

Aplicações de TVD possuem como forte característica a manipulação de conteúdos multimídias (imagem, vídeo, áudio e texto) (FERREIRA; KULESZA; SOUZA FILHO, 2012; QUINTERO et al., 2013). Esse fluxo de mídia determina o conteúdo e o ritmo da interação do usuário, chamado por alguns autores de ambiente dirigido a mídia ou *media-driven environment* (BELLOTTI et al., 2011). Desse modo, as características de cada elemento de mídia (altura, largura, início e duração da exibição, posição na pilha de mídias, ordem de exibição, entre outros.) são importantes requisitos de tais aplicações. Para isso, a linguagem NCL viabiliza o sincronismo de diferentes mídias, que são

gerenciadas através dos recursos oferecidos pela linguagem, como: regiões, descritores, conectores, portas, links, contexto, mídias, entre outros.

A presente tese considera que um método adequado ao desenvolvimento de aplicações para TVD deve concentrar-se no auxilio das atividades de planejamento e na execução das atividades de Especificação de Requisitos, Implementação e Testes, auxiliando no trato de suas peculiaridades por meio de artefatos customizados e um ciclo de vida ágil. Isso porque, quando comparada às demais plataformas computacionais, as aplicações de TVD possuem peculiaridades, como: infraestrutura de hardware, transmissão de conteúdo, público alvo, usabilidade, curto ciclo de vida, necessidade de desenvolvimento em curto espaço de tempo e equipes de desenvolvimento compostas por profissionais com diferentes formações (BELLOTTI et al., 2011; SAITO et al., 2011; SANZ, 2012; QUINTERO et al., 2013; VAVILOV; MELIKHOVA; LOGUNOV, 2009).

A pluralidade dos conteúdos da TV permite a transmissão de aplicações que podem ser executadas de modo *stand-alone* ou baseadas em transações, apresentando as mais diferentes finalidades, como: comércio (*t-commerce*), jogos (*t-games*), cidadania (*t-gov*), educação (*t-learning*), entre outras. Estas aplicações possuem características distintas das aplicações voltadas para as demais plataformas. Abaixo é apresentada uma síntese de suas principais peculiaridades:

- **ação coadjuvante –** a aplicação não é o foco dos usuários. Estes, normalmente, buscam por entretenimento, e não solicitaram a aplicação, a qual foi sugerida pela emissora. O foco da TV é produzir e exibir conteúdo audiovisual, sendo que a aplicação apenas agrega valor a esse conteúdo;
- **contexto do usuário –** o ambiente do usuário é informal. Ele não está trabalhando, normalmente está em busca de entretenimento. A relação entre usuários e a aplicação é extremamente diferenciada, pois envolver usuários de todas as idades, diferentes níveis de escolaridade, muitas vezes leigos em informática, além do meio de interação se limitar ao controle remoto. A quantidade

de usuários por aparelho receptor também é distinta, em uma TV é comum existirem mais de um usuário (por exemplo, uma família) assistindo simultaneamente ao conteúdo, diferente das aplicações para computadores pessoais (PC) ou *smart phones*, onde, normalmente, esta relação é de um usuário para cada aparelho;

- ***broadcast* –** a aplicação não é solicitada pelos usuários, e sim, enviada para todos os usuários sintonizados naquela emissora (*broadcast*). Dessa forma, deve ser atrativa o suficiente para conquistar o interesse dos usuários, que podem aceitá-la ou recusá-la. A transmissão em *broadcast* tende a envolver milhares de usuários executando simultaneamente a mesma aplicação. Dessa forma, é fundamental realizar previamente todos os testes necessários para não gerar erros ou falhas nos aparelhos receptores em larga escala;
- **impacto instantâneo –** por envolver grande parte da população. É possível que milhões de usuários recebam simultaneamente em toda a cidade, região ou até mesmo em todo o país a aplicação enviada, oferecendo um forte meio de interação com a população em larga escala;
- **requisitos multimídias** – arquivos de texto, scripts de código lua, imagens, áudio e vídeo são objetos de mídia necessários para a desenvolvimento das aplicações interativas de TVD, ou seja, devem receber atenção especial na etapa de coleta e especificação de requisitos de tais aplicações;
- **rapidez no desenvolvimento –** a expertise dos profissionais de TV no desenvolvimento do conteúdo audiovisual reforça a demanda pela velocidade no desenvolvimento da aplicação de software, pois o conteúdo televisivo muitas vezes necessita ser transmitido em poucas horas. A entrega rápida da versão final possibilita que a aplicação seja transmitida juntamente com o conteúdo audiovisual, agregando valor ao conteúdo transmitido;

- **agilidade para alteração de requisitos** – está relacionada à necessidade de se adaptar às mudanças de requisitos da aplicação sem prejuízo ao prazo de entrega ou ao custo do projeto, possibilitando que tais requisitos sejam alterados durante o processo de desenvolvimento; e
- **Equipes multidisciplinares** – o desenvolvimento da aplicação deve estar associado ao desenvolvimento do conteúdo audiovisual dos profissionais de TV, os quais também podem atuar como clientes durante o processo de desenvolvimento da aplicação.

Diante das peculiaridades supracitadas é possível afirmar que aplicações para TVD são diferentes de aplicações para outras plataformas (BELLOTTI et al., 2011; FERNANDES; TAVARES; PÔRTO, 2011; PEQUENO et al., 2010; SAITO et al., 2011; SANZ, 2012). Segundo Vavilov, Melikhova e Logunov (2009), estas surgem como um novo tipo de aplicação, com características específicas. O que gera a demanda por um método customizado para atender suas peculiaridades.

Conforme apresentado no Capítulo 2, os métodos Scrum e XP contemplam, respectivamente, atividades de planejamento e engenharia de software. Ambas as características são necessárias ao desenvolvimento de aplicações para TVD (MARQUES NETO, 2011; SILVA, 2009; VEIGA, 2006; VEIGA; TAVARES, 2006, 2007). No entanto, é interessante customizar tais métodos para contemplar as peculiaridades dessas aplicações, acarretando em adaptações voltadas para aumentar a velocidade do desenvolvimento e a adequação do ciclo de vida do processo de software.

Dessa forma, é necessário analisar as boas práticas em engenharia de software em busca da estruturação de métodos e procedimentos adequados às demandas inerentes ao desenvolvimento de aplicações para TVD.

1.2. OBJETIVOS DA TESE

A presente tese apresenta como **objetivo geral** a realização do levantamento dos métodos ágeis mais utilizados na literatura científica a fim de avaliá-los, analisando as técnicas utilizadas e possíveis pontos de melhoria, e,

por fim, definir um método capaz de auxiliar no trato das peculiaridades inerentes ao desenvolvimento de aplicações para TVD.

Os **objetivos específicos** da presente tese são:

1. Mensurar a adequação dos métodos ágeis mais utilizados para o desenvolvimento de aplicações para TVD;
2. Propor um método de desenvolvimento que atenda as peculiaridades das aplicações para TVD, com foco no desenvolvimento rápido, bem como nas atividades de Coleta e Especificação de requisitos, Planejamento e Implementação; e
3. Avaliar a adequação e a velocidade do desenvolvimento do método proposto (XDTv) através de um experimento controlado e um *survey* com profissionais experientes em TVD.

O método proposto, apresentado no Capítulo 4, tem como principais objetivos auxiliar o processo de desenvolvimento de aplicações para TVD no trato de suas principais peculiaridades, conforme apresentado a seguir:

- Auxiliar as atividades de planejamento, coleta e especificação de requisitos, e implementação de modo adequado às peculiaridades das aplicações de TVD;
- Oferecer artefatos customizados para a coleta de requisitos multimídia e na compreensão do fluxo das mídias, visando auxiliar na implementação da aplicação;
- Oferecer um modelo de processo que permita realizar alterações nos requisitos, sem prejuízo ao prazo de entrega estipulado ou no custo final do projeto; e
- Contribuir para o desenvolvimento rápido das aplicações de TVD.

1.3. PROCEDIMENTOS METODOLÓGICOS

Baseado nos conceitos da metodologia científica (MARCONI; LAKATOS, 2003), o presente trabalho adotou técnicas de pesquisa bibliográfica, exploratória, qualitativa, quantitativa e experimental. Foi feito o levantamento da literatura científica em busca dos métodos ágeis mais utilizados no processo de desenvolvimento de aplicações para TVD, com o

propósito de analisar as técnicas eficientes no tratamento das peculiaridades inerentes ao contexto dessas aplicações, visando por fim, estruturar tais técnicas através de um método ágil capaz de auxiliar nesse processo de desenvolvimento.

As questões de pesquisa, variáveis dependentes e independentes e as hipóteses apresentadas pela presente pesquisa foram avaliadas e testadas através de dois experimentos controlados. Após o fim de cada experimento, foi realizado um *survey* com desenvolvedores da indústria experientes em TVD.

Ambos os experimentos foram realizados através de times de desenvolvedores formados por alunos matriculados na disciplina Tópicos Avançados em Engenharia de Software, oferecida no 10º período pelo curso de graduação em Engenharia da Computação da Universidade Federal do Vale do São Francisco (UNIVASF). Obrigatoriamente, os participantes haviam cursado a disciplina Engenharia de Software I, a maioria também já havia cursado ou estava cursando paralelamente a disciplina Engenharia de Software II, cada uma com carga horária de 60 horas, nas quais as metodologias Scrum e XP fazem parte da ementa. Ainda assim, foram dedicadas 4 horas para revisão e nivelamento a respeito de tais metodologias.

A adoção de procedimentos metodológicos qualitativos e quantitativos, a utilização de alunos para realização dos experimentos controlados e a realização de *survey* com profissionais experientes são práticas recorrentes para avaliação de times de desenvolvimento de software (OLIVEIRA, 2007; SAMPAIO, 2004).

Os experimentos realizados no presente trabalho foram baseados na proposta de experimentação definida em (TRAVASSOS, 2002), onde é descrito um relatório técnico sobre os procedimentos necessários para realização de experimentos em engenharia de software, estabelecendo as seguintes etapas: Definição, Planejamento, Instrumentação e Execução do experimento.

A investigação realizada pelo primeiro experimento, apresentada no Capítulo 3 (Investigação sobre Scrum, XP e Híbrido) foi mais abrangente a fim de analisar e mensurar a adoção dos métodos Scrum, XP e o Híbrido formado

por Scrum junto com XP (Scrum/XP) (MUSHTAQ; QURESHI, 2012) durante o desenvolvimento de aplicações para TVD, visando verticalizar e restringir as variáveis da pesquisa em busca do método mais adequado e dos principais pontos de melhoria. Para atender aos objetivos desse experimento, uma aplicação foi desenvolvida individualmente por três times distintos. Além da variável referente ao tempo de desenvolvimento, foram investigados os possíveis pontos de melhoria visando atender às demais peculiaridades das aplicações para TVD.

Baseado nos resultados obtidos pelo primeiro experimento foi especificado o método *eXtreme Digital Television* (XDTv), apresentado no Capítulo 4, como uma alternativa mais rápida e mais adequada às peculiaridades do contexto de TVD.

Os desenvolvedores que participaram do primeiro experimento não participaram do segundo experimento. A execução do segundo experimento contou com a participação de quatro times de desenvolvimento, onde cada time desenvolveu duas aplicações. Conforme apresentado no Capítulo 5 (Avaliação do método XDTv), o segundo experimento realizou uma investigação mais aprofundada sobre as variáveis “adequação” e “tempo”. Para viabilizar uma investigação mais verticalizada, este experimento restringiu a variedade de métodos, avaliando apenas o método que obteve melhor desempenho no primeiro experimento, ou seja, o método Híbrido (Scrum/XP) e o método XDTv, proposto pela presente tese.

Para minimizar a possibilidade de falhas no experimento, é preciso que os diferentes times disponham de condições similares de desenvolvimento. Isso envolve o controle das seguintes variáveis: conhecimento tácito inerente aos membros dos times, os requisitos da aplicação desenvolvida, prazo de desenvolvimento, ferramentas utilizadas, método de desenvolvimento e o perfil do cliente (TRAVASSOS, 2002).

Viabilizar a igualdade de tais condições não é uma tarefa trivial, principalmente em fábricas reais de desenvolvimento de software, pois acarreta diretamente na elevação do custo com a mão de obra dos

programadores, além da improvável possibilidade de viabilizar equipes distintas desenvolvendo a mesma aplicação.

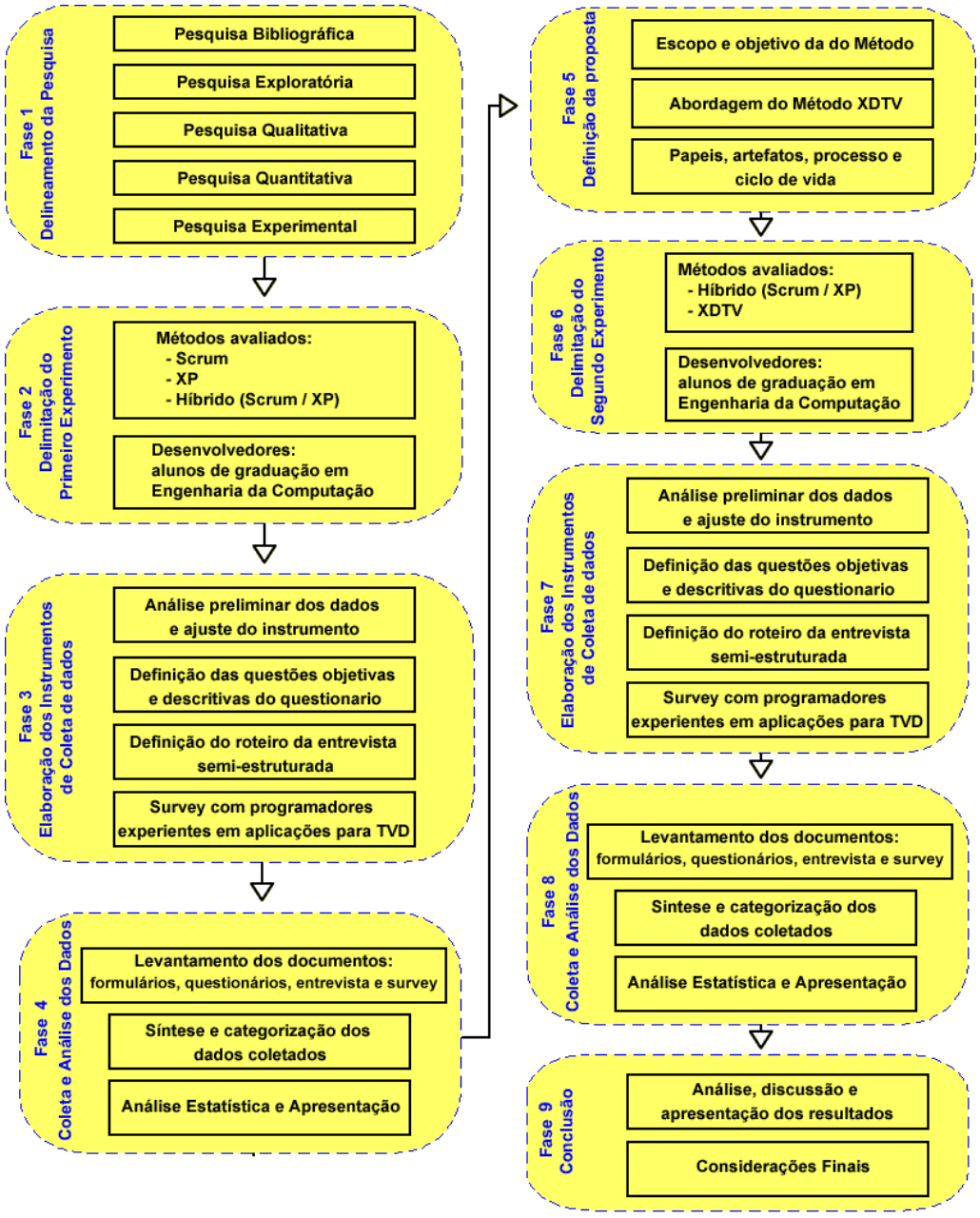
As variáveis de controle dos experimentos foram operacionalizadas através de orientação, supervisão e padronização dos materiais e métodos adotados. As caracterizações dos experimentos, suas questões de pesquisa, hipóteses, variáveis dependentes e independentes são apresentadas nos Capítulos 3 e 5, respectivamente.

Com base nos conceitos oriundos da metodologia científica (MARCONI; LAKATOS, 2003), o presente trabalho pode ser classificado como sendo uma pesquisa:

- **Exploratória**, pois realiza a estruturação e explicitação do problema, o levantamento bibliográfico, a definição de hipóteses, além de realizar um *survey* com profissionais experientes na área alvo e, por fim, elaborar uma análise e síntese para melhor compreensão dos resultados obtidos;
- **Qualitativa e Quantitativa** do ponto de vista do problema. A abordagem Qualitativa é voltada para a análise subjetiva do sujeito, mensura a adequações dos métodos investigados a partir do ponto de vista do time de desenvolvimento. A análise Quantitativa mensura a quantidade de horas trabalhadas, visando apontar o método mais rápido. Os resultados quantitativos e qualitativos são avaliados estatisticamente através da ferramenta PROMETHEEII. A associação de ambas as técnicas viabiliza uma perspectiva mais abrangente para a coleta de resultados, aprimorando sua análise e a conclusão sobre os mesmos; e
- **Bibliográfica** e **Experimental** baseada nos procedimentos de pesquisa. A pesquisa Bibliográfica foi utilizada constantemente como forma de fundamentação teórica através de trabalhos científicos. Os experimentos controlados foram adotados devido à necessidade de avaliação das variáveis investigadas.

A Figura 1.5 representa o desenho metodológico, através de uma síntese das atividades realizadas subdivididas em nove fases, ordenadas de acordo com sua execução.

Figura 1.5 – Desenho metodológico para o desenvolvimento da tese



Fonte: o autor.

A pesquisa desenvolvida pela presente tese, a qual foi iniciada na **Fase 1** (Figura 1.5), onde também foi definido o delineamento da pesquisa e o seu

direcionamento através da revisão bibliográfica, apresentada no Capítulo 2, exponde a fundamentação dos conceitos empregados e os métodos mais utilizados para o desenvolvimento de aplicações para TVD. A revisão da literatura apontou os métodos Scrum, XP e o método Híbrido (Scrum/XP) (MUSHTAQ; QURESHI, 2012) como os mais adotados. No entanto, a adoção de tais métodos foi apenas citada em trabalhos relacionados, não havendo detalhamento do experimento, definição das variáveis acarretando na falta das métricas necessárias para a comparação e indicação do método mais rápido ou mais adequado ao desenvolvimento de tais aplicações. O levantamento bibliográfico foi contínuo, realizado paralelamente a todas as nove fases da presente pesquisa.

Em seguida, na **Fase 2**, foi delimitado o primeiro experimento, apresentado no Capítulo 4, responsável por avaliar os métodos ágeis (Scrum, XP e Híbrido) indicados nos trabalhos relacionados em busca de possíveis pontos de melhoria e indícios de qual destes métodos mais adequado ao contexto de TVD.

Na **Fase 3** foram definidos e elaborados os instrumentos para coleta dos dados qualitativos e quantitativos, visando contribuir para o refinamento e precisão dos resultados do primeiro experimento.

A **Fase 4** obteve, sintetizou, organizou, e, por fim, analisou os resultados obtidos através do método Multicritério de apoio a tomada de decisão[6], conforme apresentado na próxima seção, oferecendo indícios relevantes que permitiram a continuidade desta pesquisa.

A **Fase 5** especifica e estrutura o método XDTv, baseando-se tanto no levantamento bibliográfico, como também, nos resultados obtidos através do primeiro experimento. Tal método incorpora os princípios ágeis a um ciclo de vida e artefatos adequados às peculiaridades das aplicações de TVD.

Na **Fase 6** é delimitado o segundo experimento, apresentado no Capítulo 6, responsável por avaliar o método Híbrido (Scrum/XP) e o método XDTv, proposto pela presente tese. O segundo experimento teve como objetivo

[6] O método Multicritério foi adotado através da ferramenta PROMETHEE II

apontar o método mais rápido e o método que melhor atende às atividades de coleta e especificação de requisitos, planejamento e implementação de aplicações para TVD.

Na **Fase 7** foram definidos e elaborados os instrumentos para coleta dos dados qualitativos e quantitativos, visando contribuir para o refinamento e precisão dos resultados do segundo experimento.

A **Fase 8** obteve, sintetizou, organizou, e, por fim, analisou os resultados obtidos através do método multicritério de apoio a tomada de decisão, apresentado na próxima seção. Para isso foi utilizada a ferramenta PROMETHEE II.

Por fim, na **Fase 9,** os resultados obtidos através do segundo experimento foram comparados e avaliados estatisticamente através do método multicritério de apoio a tomada de decisão, implementado pela ferramenta PROMETHEE II que apontou o método de desenvolvimento mais rápido e mais adequado às atividades de coleta de requisitos, planejamento e implementação de aplicações para TVD.

A seguir são apresentados os possíveis **riscos** que, apesar de reduzidos durante o planejamento e execução, podem interferir na validade do experimento:

- é possível que tenha existido compartilhamento parcial de código entre os times, porém, pouco provável. Para reduzir esse risco foram realizados: **a)** solicitação de requisitos não funcionais distintos para cada time; **b)** inspeção dos códigos realizada após a entrega das aplicações (que não apresentaram quaisquer evidências de compartilhamento);
- é possível que exista algum dos participantes com maior ou menor afinidade (*know how*) ou interesse com programação (NCL / Lua), interferindo assim nos resultados coletados. Para reduzir esse risco foram realizadas: **a)** entrevistas individuais sobre interesses pessoais; **b)** análise de currículo; **c)** análise das

disciplinas já cursadas pelos participantes; e **d)** tabela de pontuação e avaliação dos diferentes perfis; e

- o conhecimento exclusivamente acadêmico dos participantes dos experimentos pode ser considerado um risco. Para reduzir esse risco, foram realizados: **a)** um treinamento prévio sobre métodos ágeis e TVD; **b)** a supervisão dos times pelo autor da presente pesquisa, durante a realização de parte dos experimentos; e **c)** dois *surveys* com profissionais experientes em TVD.

1.4. DELINEAMENTO DOS EXPERIMENTOS E ANÁLISE DOS RESULTADOS

No entanto, comparar o desempenho de diferentes times e processos de desenvolvimento de software não é uma tarefa trivial. Segundo Sánchez (2011), variáveis como: área de formação; tempo de experiência; virtudes pessoais; e, diferentes funcionalidades e objetivos do software, podem ser consideradas como fatores de variação, apresentando valores distintos para cada equipe, influenciando diretamente nos resultados obtidos pelo experimento, favorecendo uma das equipes e invalidando os resultados obtidos. Para minimizar tais fatores de variação pode ser utilizado o Quadrado Latino aleatorizado, como forma de organizar os experimentos em linhas e colunas (SÁNCHEZ, 2011).

O princípio da aleatoriedade é fundamental para determinar o arranjo, ou seja, a disposição dos elementos em cada linha e coluna do Quadrado Latino. Para isso pode ser utilizado qualquer procedimento de geração de números aleatórios que garanta que a ordem dos elementos do quadrado seja determinada por fatos eventuais ou pela casualidade, como um algoritmo randômico (SÁNCHEZ, 2011). O autor destaca entre suas conclusões a inviabilidade de contratar desenvolvedores de software para realizar experimentações, e define que o quadrado latino com duas (2) linhas e duas (2) colunas (2x2) é adequado para comparar diferentes métodos de desenvolvimento de software.

O segundo experimento realizado pela presente tese demanda por forte controle sobre os fatores de variação estabelecidos em (SÁNCHEZ, 2011). As aplicações de software desenvolvidas no experimento possuem os mesmos requisitos funcionais, objetivos e cliente em cada etapa do experimento. O conhecimento dos desenvolvedores envolvidos no experimento de avaliação do método XDTv foi analisado através de questionário e entrevista individual. Os resultados foram distribuídos em três grupos, de acordo com o nível de conhecimento levantado. Os times de desenvolvimento foram montados através do sorteio de um membro de cada grupo. Visando formar times de desenvolvimento de software com conhecimentos homogêneos, a presente tese realizou a análise de currículo, questionário e uma entrevista individual, conforme apresentado no Capítulo 5.

Dessa forma, o segundo experimento adotou o Quadrado Latino (2 x 2), onde através do princípio da aleatoriedade, foram distribuídos os times de desenvolvimento, as aplicações e a ordem em que cada método de desenvolvimento foi atribuído para cada time.

Os resultados quantitativos e qualitativos foram analisados através do método multicritério de apoio a decisão, implementado através da ferramenta PROMETHEE II, apresentado nos trabalhos relacionados, Seção 3.10. O autor da presente pesquisa atuou como analista, modelando o problema decisório e montando a matriz de decisão com base em avaliações qualitativas e quantitativas, realizadas pelos participantes e coletadas através dos experimentos controlados.

1.5. ORGANIZAÇÃO DO TRABALHO

Além deste capítulo introdutório, que contextualiza, justifica e apresenta os objetivos da presente tese, os demais capítulos estão organizados da seguinte forma:

- O Capítulo 2 apresenta o referencial teórico sobre o tema abordado no presente trabalho, detalhando as características dos modelos de processos e dos métodos ágeis de desenvolvimento Scrum e XP;

- O Capítulo 3 apresenta os trabalhos relacionados à presente tese;
- O Capítulo 4 expõe os objetivos, planejamento, operação, resultados, e, por fim, a conclusão do primeiro experimento, onde foram avaliados os métodos Scrum, XP e Híbrido;
- O Capítulo 5 apresenta o método *eXtreme Digital Television* (XDTv), que incorpora boas práticas conhecidas da engenharia do software associadas a novos artefatos customizados para as aplicações de TVD.
- O Capítulo 6 apresenta os objetivos e o planejamento do segundo experimento, onde foi avaliado o método XDTv;
- O Capítulo 7 apresenta as conclusões e os trabalhos futuros;
- O Capítulo 8 lista as referências bibliográficas; e
- O Capítulo 9 apresenta os Apêndices da presente pesquisa.

CAPÍTULO

2.

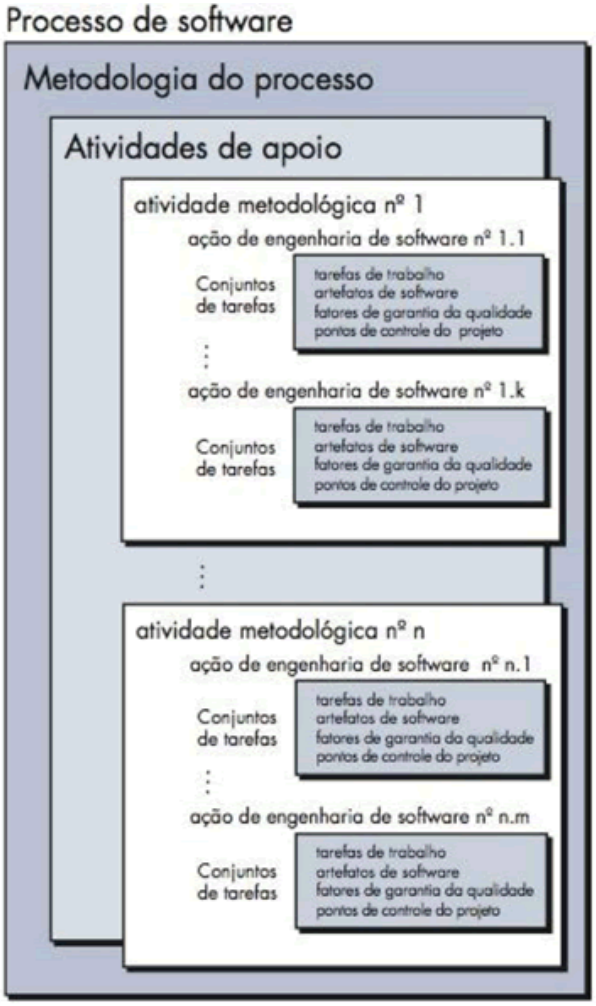
2. MODELO DE PROCESSO E MÉTODO DE DESENVOLVIMENTO

"Conversando, dialogamos, dialogando, aprendemos."
Chico Xavier

O presente capítulo apresenta a revisão bibliográfica sobre os principais processos e métodos de desenvolvimento conhecidos na engenharia de software, com foco nos métodos Scrum, XP e um método Híbrido formado por ambos.

Um processo de software corresponde a um roteiro capaz de auxiliar a entrega de um software de alta qualidade, dentro do prazo previsto, apresentando ações e tarefas necessárias para desenvolver um software de alta qualidade (Pressman, 2011). Um processo de software fornece um modelo (*template*), que pode ser definido como um conjunto de atividades de trabalho, ações e tarefas realizadas para a criação de um artefato de software. Tais atividades, ações e tarefas devem compor uma metodologia (ou modelo) capaz de determinar seu relacionamento entre si e com o processo. Cada atividade metodológica é formada por ações e cada ação é definida por um conjunto de tarefas, conforme apresentado na Figura 2.1.

Figura 2.1 – Processo, Metodologia e atividades



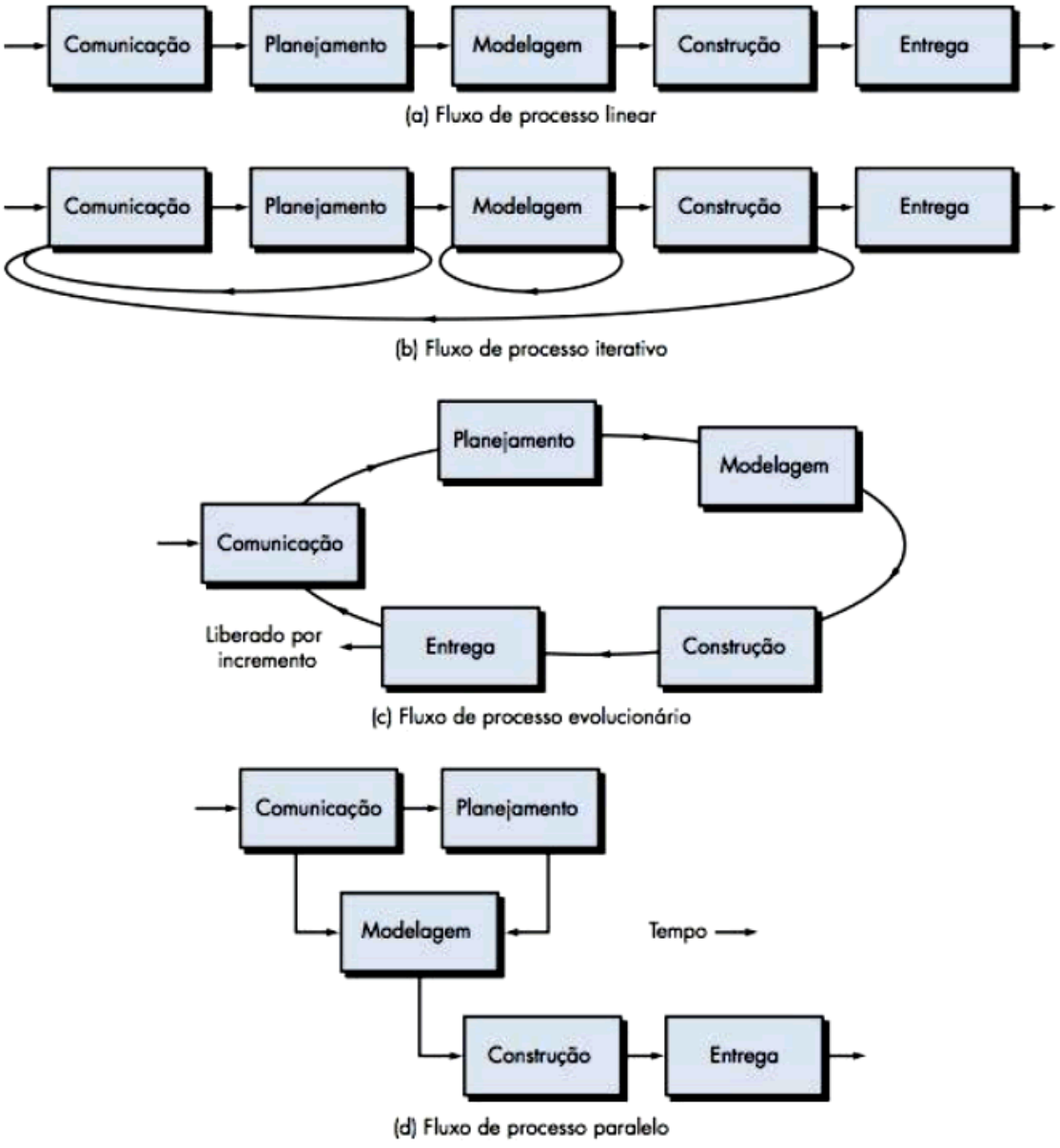
Fonte: Pressman, 2011.

A Figura 2.1 apresenta uma metodologia genérica de processo de software que estabelece cinco atividades: comunicação, planejamento, modelagem, construção e entrega. A metodologia determina o relacionamento das atividades, ações e tarefas com o processo de software. Adicionalmente, podem ser aplicadas durante o processo outras atividades de apoio, como: acompanhamento e controle do projeto, administração de riscos, garantia de qualidade, gerenciamento de configurações, revisões técnicas, entre outras (PRESSMAN, 2011).

Um processo de software possui um fluxo de processo (ou fluxo de trabalho) que corresponde à forma pela qual os elementos do processo se relacionam entre si. Este fluxo representa a organização das cinco atividades

metodológicas, suas ações e tarefas em relação à sequência de tempo (PRESSMAN, 2011). É possível resumir a classificação dos fluxos de processo em quatro tipos: a) Linear; b) Iterativo; c) Evolucionário; e d) Paralelo, conforme apresentado na Figura 2.2.

Figura 2.2 – Tipos de fluxo de processo



Fonte: Pressman, 2011.

O fluxo **Linear (a)** executa as cinco atividades metodológicas em sequência. O fluxo **Iterativo (b)** repete uma ou mais atividades antes de realizar a atividade seguinte. O fluxo **Evolucionário (c)** realiza as atividades

em ciclos, onde cada ciclo realiza as cinco atividades, e o fim de cada ciclo resulta em uma versão mais completa do software. O fluxo **Paralelo (d)** realiza uma ou mais atividades em paralelo (PRESSMAN, 2011).

Pressman (2011) apresenta também modelos de processo rotulados como "prescritivos", pois são estruturados e ordenados apresentando um conjunto de elementos de processo, atividades metodológicas, ações de engenharia de software, tarefas, produtos de trabalho, garantia da qualidade e mecanismos de controle para os projetos, bem como um fluxo de processo (fluxo de trabalho).

Diferentes autores adotam nomenclaturas sutilmente distintas para definir o processo de desenvolvimento de software. Resumidamente, Pressman (2011) adota os termos: Modelo de Processo e Metodologia. No entanto Sommerville (2011) utiliza: Modelo de Processo e Método, considerando que a palavra metodologia corresponde ao estudo (“*logia*”) do método, dentro do contexto de engenharia de software, pode se assumir que os termos “método” e “metodologia” são equivalentes.

O presente trabalho segue a nomenclatura de Sommerville (2011), onde um processo de software refere-se a uma abordagem sistemática e organizada para o desenvolvimento de software. Resumidamente, segundo Sommerville (2011), é possível definir quatro atividades fundamentais comuns a todos os processos de software:

- Especificação – clientes e engenheiros definem o software a ser produzido e suas restrições;
- Desenvolvimento – o software é projetado e implementado;
- Validação – o software é verificado a fim de confirmar que atende as especificações; e
- Evolução – após a entrega da versão final do software, este demanda por modificações para atender às mudanças de requisitos do cliente ou do mercado.

A atividade de especificação de software é subdividida em quatro fases fundamentais:

1) Estudo de viabilidade – esta primeira avaliação deve ser rápida e de baixo custo. Pretende realizar uma estimativa da possibilidade de lograr êxito com a satisfação do usuário através das tecnologias atuais disponíveis. Considera-se a rentabilidade do ponto de vista de negócio, e o orçamento disponível para o projeto. Como resultado, deve-se indicar o cancelamento ou o avanço para uma análise mais detalhada;

2) Elicitação e análise de requisitos – é o processo de levantamento dos requisitos do sistema a partir da análise do funcionamento da organização. Podem ser utilizadas técnicas de observação, entrevista com usuários, entre outras formas de levantamento de informações;

3) Especificação de requisitos – é a tradução das informações obtidas em um documento capaz de definir um conjunto de requisitos. Podem ser classificados como requisitos do usuário (abstratos) e requisitos de sistema, compostos por detalhes da funcionalidade a ser implementada; e

4) Avaliação de requisitos – verifica o realismo, consistência e completude dos requisitos, levantando e corrigindo os erros do documento de requisitos.

As atividades de Validação e Evolução estão fora do escopo da presente tese, por esse motivo, não foram detalhadas. A atividade de Desenvolvimento será apresentada no decorrer desta pesquisa.

Utilizando abstração, é possível resumir a classificação dos processos de software como baseados em plano ou em agilidade. Processos baseados em planos pregam o planejamento antecipado de todas as atividades. O progresso do projeto é avaliado de acordo com o planejamento inicial, que deve prever futuras necessidades do software. Seu planejamento antecipa e facilita alterações futuras, entretanto, muitas vezes tal esforço é desperdiçado adicionando generalidades ao projeto que possivelmente nunca serão utilizadas. No processo dirigido à agilidade tais preocupações com futuras necessidades dos usuários não fazem parte da filosofia ágil. O planejamento é gradativo e a alteração dos requisitos faz parte de sua filosofia, possibilitando que tais alterações sejam realizadas sem prejuízo ao projeto. Seu progresso é

mensurado a partir dos módulos funcionais entregues, ou seja, partes do software prontas para utilização e entregues ao usuário final. A Figura 2.3 apresenta um paralelo entre ambos os processos (SOMMERVILLE, 2011).

Figura 2.3 – Desenvolvimento orientado a planos e ágeis

Fonte: Sommerville, 2011.

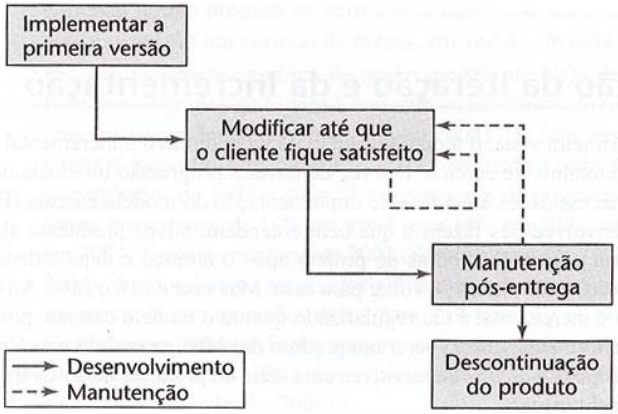
O conceito de Método de Desenvolvimento, chamado por alguns autores de Metodologia de Desenvolvimento, não é uma descrição definitiva do modelo de processo de software, e sim, uma representação simplificada, abstrata, desse modelo (SOMMERVILLE, 2011). Pode ser utilizado para explicar diferentes abordagens de desenvolvimento. Cada método representa uma perspectiva particular de um processo. Estes podem ser ampliados ou adaptados para criar processos mais específicos (SOMMERVILLE, 2011).

Existem diversos tipos e variações de métodos de desenvolvimento de software na literatura. Em Schach, (2009) são descritos cinco ciclos de vida de modelos de processo de desenvolvimento de software: Codificar e corrigir; Cascata; Iterativo e Incremental; Prototipagem Rápida; e Espiral.

O modelo rotulado como Codificar e Corrigir (SCHACH, 2009), ilustrado na Figura 2.4, trata-se de uma má prática realizada com frequência, na qual não ocorre o levantamento de requisitos e especificações adequadas, não

apresentando qualquer tentativa de projetar o desenvolvimento de software. Por meio dele, o software é reimplementado inúmeras vezes até que atenda às necessidades do cliente, o que implica na difícil manutenção e no elevado custo do desenvolvimento, se comparado a modelos que preveem atividades de coleta e especificação de requisitos de forma adequada.

Figura 2.4 – Ciclo de vida do modelo Codificar e Corrigir



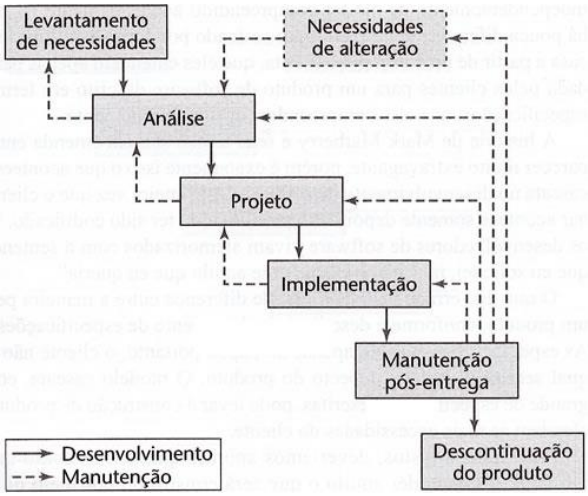
Fonte: Schach, 2009.

O modelo Cascata adota o Modelo Linear Sequencial. É um exemplo de modelo baseado em planos, proposto por W. W. Royce em 1970, década que apresentava um diferente contexto para o desenvolvimento de software em relação aos dias atuais. Essa diferença está relacionada tanto às plataformas computacionais da época, quando à necessidade dos usuários (SCHACH, 2009). Esse modelo possui entre suas características a inflexibilidade relacionada à alteração de requisitos. Suas principais atividades são: análise e definição de todos os requisitos; projeto de sistema e software; implementação e teste unitário; integração e teste de sistema; e operação e manutenção. Tais características tornam esse modelo não adaptado à modificação de requisitos.

No modelo Cascata, nenhuma fase é terminada até que sua documentação esteja completa e, em alguns casos, é possível que seja solicitada a aprovação da equipe que garante a qualidade de software (SQA), implicando em constantes solicitações de modificações ao fim de cada etapa. A especificação da documentação deve ser alterada e, quando necessário, aprovada pelo SQA para cada solicitação de modificação. Dessa forma,

contínuos testes são necessários ao longo do desenvolvimento do software. Especificamente na fase de Manutenção, é necessário garantir que o software faça o mesmo que a versão anterior de forma correta, e ainda atenda às novas solicitações de mudança determinadas pelo cliente, conforme exibido na Figura 2.5.

Figura 2.5 – Ciclo de vida do modelo Cascata



Fonte: Schach, 2009.

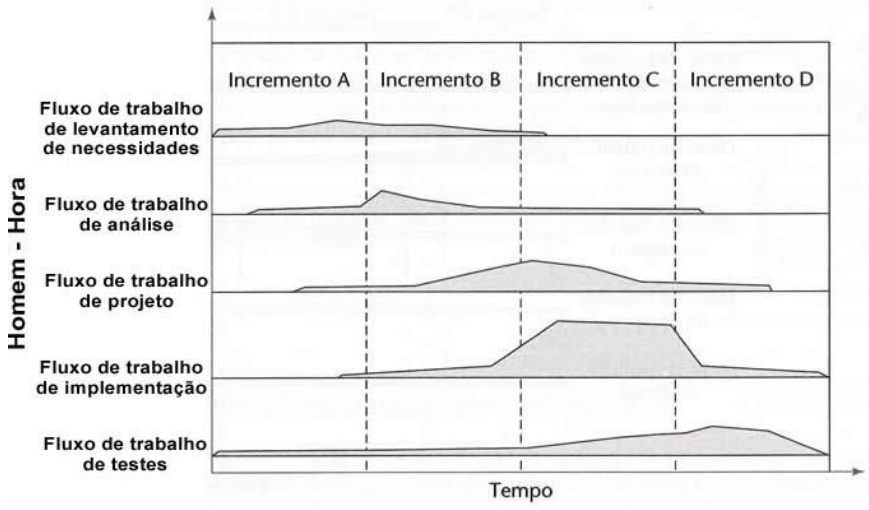
Uma característica marcante do modelo cascata é a documentação de cada fase. Por meio dela, são definidas as necessidades e atividades que devem ser satisfeitas. A verificação e avaliação dos documentos desenvolvidos em cada fase pelo SQA gera um impacto notório, que pode ser positivo ou negativo, dependendo das características do software que será desenvolvido. Pode ser considerado positivo para software cujos requisitos são fortemente definidos, sem perspectiva de mudança. Pode ser considerado negativo para software que não possui um escopo fortemente definido (SCHACH, 2009).

No entanto, o modelo Cascata apresenta as seguintes desvantagens quando comparado com o modelo Incremental: 1) custo elevado para alteração de requisitos; 2) *feedback* tardio; e, 3) entrega de software funcional apenas no fim do projeto. Porém, do ponto de vista do gerenciamento, o modelo Incremental apresenta como principal problema a dificuldade de visualização do processo, devido à redução do foco em documentação.

O modelo Iterativo (ou Iterativo e Incremental) utiliza o refinamento gradual para trabalhar com grandes quantidades de informação inerentes ao desenvolvimento de software, centrando esforços nos aspectos mais relevantes em cada momento, deixando para etapas futuras aspectos menos críticos. No entanto, tanto os aspectos mais relevantes como os menos relevantes serão tratados, porém, cada um seguindo sua ordem de prioridade.

As atividades de iteração e integração são desenvolvidas em conjunto, ou seja, um artefato é desenvolvido parte por parte (incremento), e cada incremento passa por diversas versões (iteração), conforme exibido na Figura 2.6, que apresenta um produto de software construído através de quatro incrementos.

Figura 2.6 – Ciclo de vida do modelo Iterativo



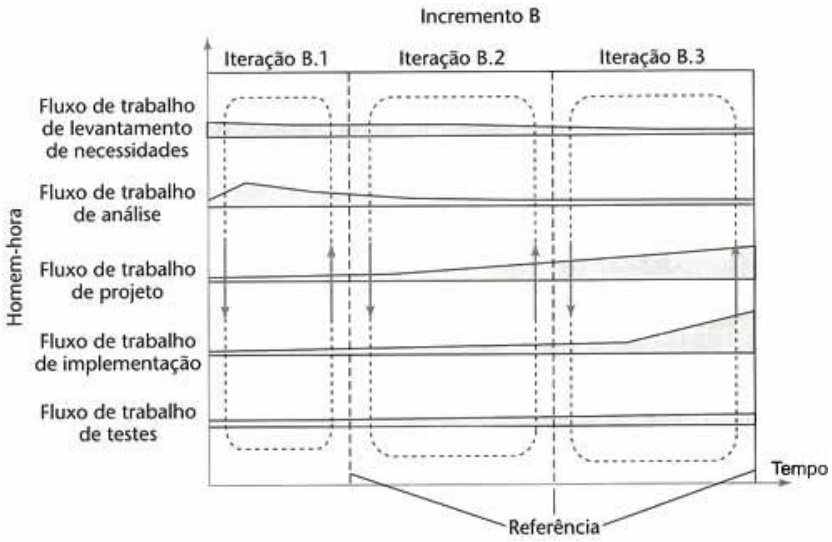
Fonte: Schach, 2009.

A **Figura 2.6** apresenta cinco diferentes fluxos de trabalho que representam as atividades realizadas ao longo do processo de desenvolvimento do produto: levantamento de necessidades; análise; projeto; implementação; e testes. Tais fluxos são realizados durante o processo de desenvolvimento, no entanto, as horas de trabalho dedicadas por cada desenvolvedor variam de um fluxo para outro no mesmo incremento. Esta variação está diretamente ligada às necessidades do incremento que está sendo desenvolvido. É natural que os incrementos iniciais possuam maior

necessidade de levantamento de requisitos. Assim que são minimizadas as demandas de requisitos eleva-se as atividades de análise, e assim sucessivamente até finalizar as demandas do fluxo de trabalho de testes.

Cada incremento possui sua iteração própria, podendo ser subdividido em ciclos menores que contemplam todos os cinco fluxos de atividades. A quantidade de ciclos menores pode variar de incremento para incremento. Na **Figura 2.7** é mostrado um exemplo do Incremento B subdividido em três iterações (SCHACH, 2009).

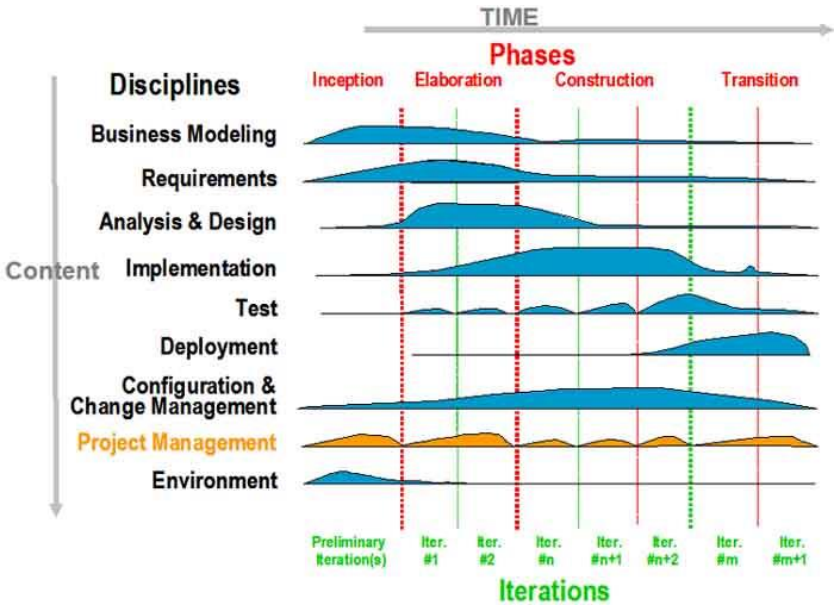
Figura 2.7 – Incremento B subdividido em três iterações



Fonte: Schach, 2009.

Um dos métodos iterativos e incrementais mais conhecidos é o *Rational Unified Process* (RUP) (IBM, 2004), que incorpora métodos padrões na indústria de gestão e técnicas para fornecer um processo de engenharia de software composto por nove disciplinas. Cada uma possui um fluxo de trabalho, cujo grau de atividade está diretamente relacionado a uma das quatro fases do processo de desenvolvimento, formando assim uma sequência única de tarefas, conforme exibido na Figura 2.8.

Figura 2.8 – Principais elementos do RUP

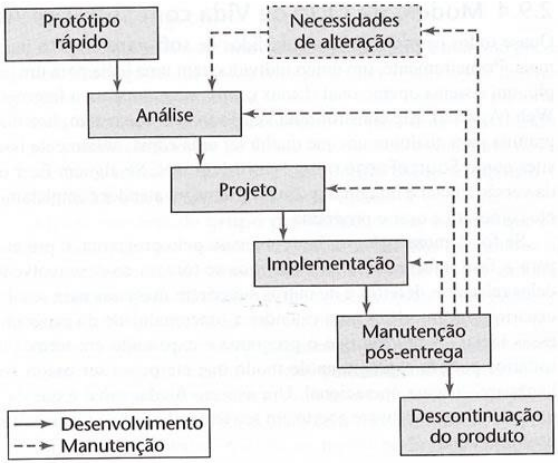


Fonte: IBM, 2004.

Outro método conhecido é denominado como Baseado em Prototipagem Rápida. Este prevê a implementação de um módulo de requisitos funcionais básicos, o mais rápido possível, sem a preocupação com detalhes de validação e verificação de entrada de dados, pois, após seu uso, o protótipo é descartado.

O protótipo visa auxiliar no levantamento das necessidades reais do cliente, oferecendo a ele e seus usuários a possibilidade de interagir e experimentar o protótipo. Assim que aqueles estiverem satisfeitos, os desenvolvedores elaboram o documento de especificações, dando sequência ao processo de desenvolvimento, conforme exibido na Figura 2.9.

Figura 2.9 – Ciclo de vida do método baseado em Prototipagem Rápida



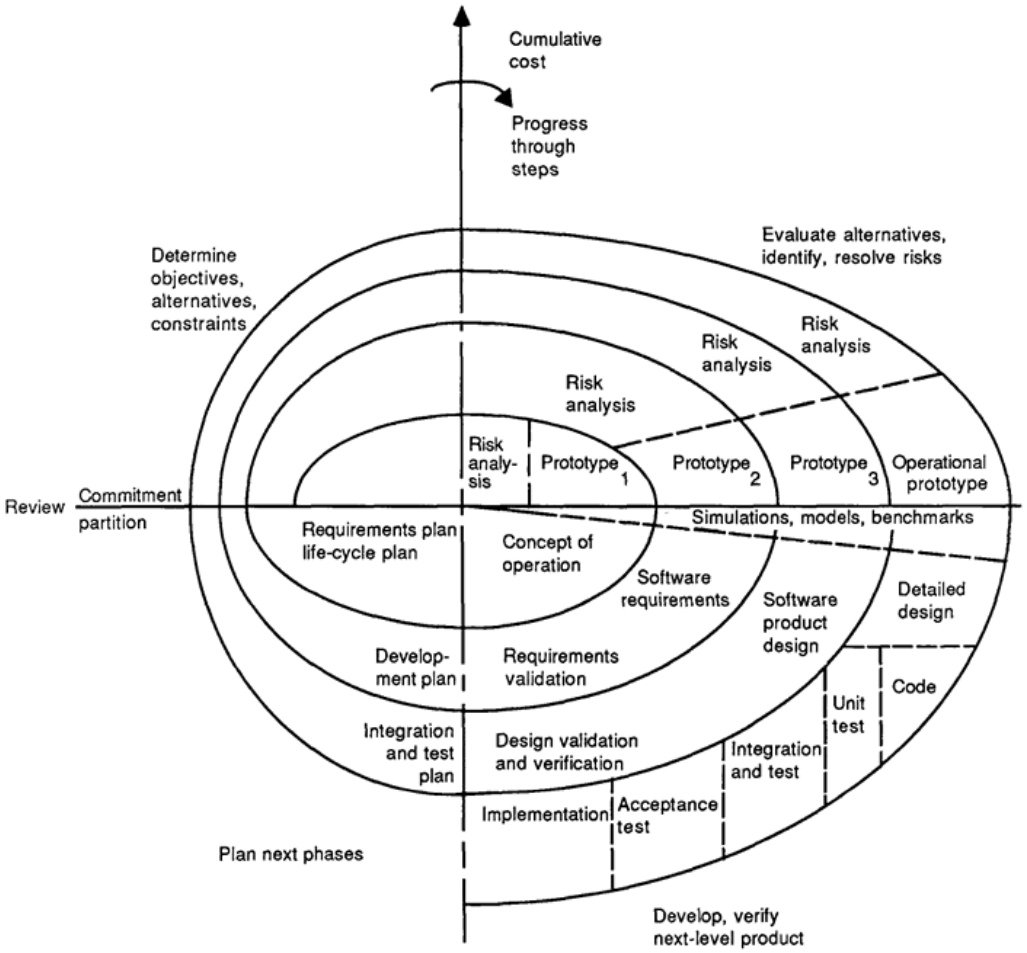
Fonte: Schach, 2009.

O desenvolvimento linear, a evolução do protótipo para o produto, é um ponto positivo da prototipagem rápida, diminuindo a probabilidade de retorno de solicitações de modificação. A implementação do protótipo pode antecipar possíveis problemas, gerando ideias de como implementar melhor o software final (SCHACH, 2009).

O modelo Espiral (BOEHM, 1988) é um modelo evolucionário e dirigido a riscos, apresenta uma abordagem cíclica, visando que o software seja incrementado, oferecendo ainda pontos de controle para manter o comprometimento dos interessados em soluções satisfatórias e viáveis. O modelo Espiral incorpora a natureza iterativa da prototipação com características sistemáticas e controladas do modelo Cascata.

O raio formado pelo espiral representa os resultados acumulados em cada quadrante, iniciando no quadrante superior esquerdo onde são levantados seus objetivos, conforme exibido na **Figura 2.10**. No quadrante seguinte são analisados os riscos, visando antecipá-los em busca de minimizar seus impactos. Em alguns casos são utilizados protótipos para atenuar tais riscos. Este modelo pode ser considerado como dirigido a riscos, porém, tais riscos necessitam de uma equipe competente para analisar pontos críticos e tomarem a decisão mais acertada para superá-los (SCHACH, 2009).

Figura 2.10 – Ciclo de vida do modelo Espiral



Fonte: Boehm, 1988.

O ciclo Espiral intercala as atividades de especificação, desenvolvimento e validação. O software é desenvolvido através de diversas versões funcionais, em forma de módulos que são incrementados às suas versões anteriores (BOEHM, 1988). Cada versão deve ser criticada pelos usuários com objetivo de obter *feedback* sobre o módulo desenvolvido.

A **Erro! Fonte de referência não encontrada.** apresenta uma síntese dos pontos fortes e fracos dos principais modelos apresentados no presente capítulo (SCHACH, 2009).

Figura 2.11 – Comparação entre modelos

Modelo de ciclo de vida	Pontos Fortes	Pontos Fracos
Modelo iterativo e incremental	Aproxima-se da produção de software no mundo real Base do Processo Unificado	
Modelo codificar-e-corrigir	Bom para programas curtos que não exigem nenhuma manutenção	Totalmente insatisfatório para programas não triviais
Modelo cascata	Método rigoroso Dirigido por documentação	Pode ser que o produto entregue não atenda às necessidades do cliente
Modelo de prototipagem rápida	Garante que o produto entregue atenda às necessidades do cliente	Ainda não foi comprovado totalmente
Modelo com software aberto	Funcionou perfeitamente bem em um número pequeno de casos	Aplicabilidade limitada Normalmente não funciona
Processos ágeis	Funciona bem quando as necessidades do cliente são vagas	Parece funcionar apenas em projetos de pequena escala
Modelo sincronizar-e-estabilizar	Necessidades futuras do cliente são atendidas Garante que os componentes possam ser integrados com sucesso	Ainda não foi usado amplamente, a não ser pela Microsoft
Modelo espiral	Dirigido por riscos	Pode ser usado apenas para produtos internos de larga escala. Os desenvolvedores têm de ser competentes na análise de riscos e na sua solução.

Fonte: Schach, 2009.

Schach (2009) frisa que o modelo de Prototipagem Rápida foi concebido a partir de um ponto fraco inerente ao modelo cascata, minimizando a possibilidade do produto entregue não satisfazer as expectativas do cliente. O modelo de código aberto não faz parte do escopo do presente trabalho, no entanto, é utilizado com sucesso no desenvolvimento de software de infraestrutura. O modelo Espiral é uma alternativa interessante, porém, necessita que os desenvolvedores sejam experientes em análise de risco e suas respectivas soluções.

A literatura apresenta outros modelos de desenvolvimento como o software Orientado a Reuso, que é baseado na existência de um número significativo de componentes reutilizáveis. Normalmente é voltado para o desenvolvimento de software em larga escala comercial, também chamado de software de prateleira (SOMMERVILLE, 2011). A estratégia do reuso de software foi proposta em 1968 por McIlroy, onde o desenvolvimento concentra-se na integração desses componentes, em vez da sua completa recodificação. Como o nome já diz, software orientado a reuso pretende reduzir custo e riscos com a codificação de funcionalidades. Este modelo de desenvolvimento possui forte compromisso com os requisitos de software, porém, o constante reuso de funcionalidades tende a dificultar o gerenciamento do controle de versões. Desta forma, apesar de oferecer características relevantes, este método não faz parte do escopo desse trabalho.

Diante das peculiaridades das aplicações de TVD apresentadas no Capítulo 1, podemos analisar os diferentes modelos processo de desenvolvimento em busca daqueles mais adequados a tais aplicações. Características como a multidisciplinaridade, possíveis alterações de requisitos, e necessidade de rapidez no desenvolvimento, reduzem a adequação dos modelos baseados em planos para grande parte das aplicações de TVD. Em contrapartida, apontam indícios de que os modelos iterativos podem auxiliar no processo de desenvolvimento, pois possuem atividades que podem auxiliar algumas dessas características elencadas.

Outros modelos oferecem características específicas que podem contribuir para o desenvolvimento de aplicações para TVD. O modelo baseado em Prototipagem Rápida, por exemplo, apesar de oferecer protótipos funcionais desenvolvidos de forma simples, prevê a avaliação desses protótipos como forma complementar as atividades de coleta, especificação e validação de requisitos. Dessa forma, diferentes modelos podem ser adaptados, utilizando protótipos de baixo nível de fidelidade, papel e canetas coloridas, para auxiliar a coleta de requisitos e comunicação com o cliente, mesclando atividades chaves para a elaboração de uma metodologia voltada para o contexto de TVD, pois, segundo Pressman (2011), é necessário adaptar

um processo de software maduro de forma apropriada aos produtos e às demandas do mercado.

Segundo Pressman (2011), uma abordagem de engenharia de software moderna deve ser ágil, oferecendo apenas atividades apropriadas para a equipe e seu projeto. O desenvolvimento de software ágil apresenta as atividades metodológicas básicas (comunicação, planejamento, modelagem, construção e entrega) minimizadas, centrando esforços no desenvolvimento e na entrega do "incremento de software" operacional no prazo combinado (PRESSMAN, 2011).

Existem vários métodos ágeis propostos pela literatura, os quais apresentam abordagens com sutis diferenças. Os adeptos aos métodos ágeis classificam tais abordagens como "ideias", que podem ser comparadas com as "tarefas de trabalho" dos processos tradicionais da engenharia de software. Grande parte dos métodos ágeis utiliza o modelo Iterativo como base do seu processo de desenvolvimento, como apresentado na próxima seção.

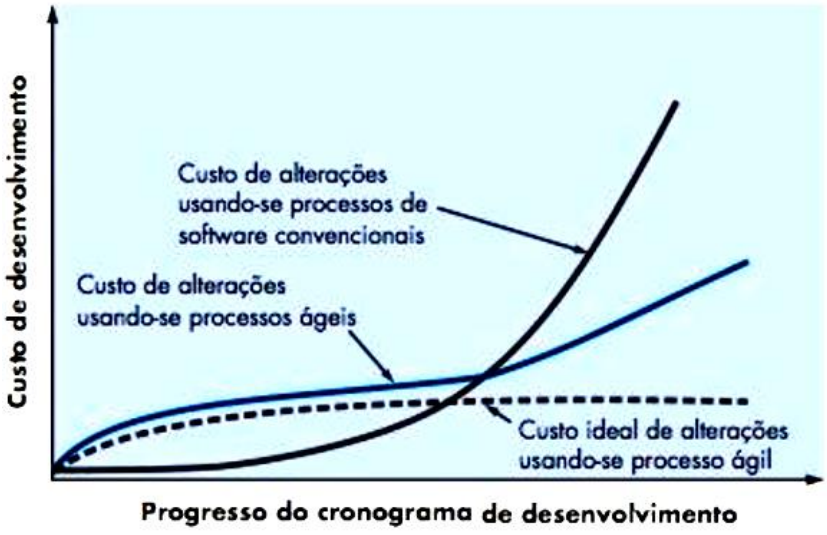
2.1. MÉTODOS ÁGEIS

Em 2001, 17 autores de métodos ágeis conhecidos e aceitos pela comunidade científica definiram formalmente os princípios filosóficos de um processo ágil genérico. Tais princípios foram publicados através do Manifesto Ágil (BECK et al., 2001), conforme apresentado abaixo:

> Estamos descobrindo maneiras melhores de desenvolver software, fazendo-o nós mesmos e ajudando outros a fazerem o mesmo. Através deste trabalho, passamos a valorizar:
>
> Indivíduos e interações mais que processos e ferramentas
>
> Software em funcionamento mais que documentação abrangente
>
> Colaboração com o cliente mais que negociação de contratos
>
> Responder a mudanças mais que seguir um plano
>
> Ou seja, mesmo havendo valor nos itens à direita, valorizamos mais os itens à esquerda.

A agilidade está relacionada à adaptação do ciclo de desenvolvimento mediante alterações dos requisitos da aplicação, minimizando os prejuízos relacionados ao custo e ao prazo de entrega do projeto de software. Esta característica torna os métodos ágeis adequados ao desenvolvimento daqueles tipos de software que não possuem requisitos fortemente definidos. O **Gráfico 1** apresenta uma estimativa de custo relacionado à modificação nos requistos de projetos convencionais e projetos ágeis.

Gráfico 1 – Custo referente às alterações de requisitos



Fonte: Pressman, 2011.

O **Gráfico 1** mostra que os custos inerentes a alterações de requisitos aumentam de forma não linear com o decorrer do projeto, pois, no início de um projeto as modificações podem ser adaptadas de forma mais natural. Porém, se a modificação ocorrer na fase de testes e validação (no fim do projeto) pode desencadear uma sequência de modificações que elevam o prazo e o custo do projeto (PRESSMAN, 2011). Dessa forma, a entrega incremental permite que as alterações de requisitos sejam controladas de forma mais eficiente que nos modelos tradicionais.

O Manifesto Ágil prevê que, para um método ser considerado ágil, este deve compartilhar de sua filosofia e princípios. Tal manifesto definiu 12 princípios fundamentais:

1. Prioridade em satisfazer o cliente através da entrega contínua e adiantada de software com valor agregado;
2. Mudanças nos requisitos são bem-vindas a qualquer momento. Processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente;
3. Entrega frequente de software funcionando, de poucas semanas a poucos meses, com preferência à menor escala de tempo;
4. Profissionais com conhecimento do negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto;
5. Desenvolvimento de projetos em torno de indivíduos motivados, oferecendo o ambiente, o suporte necessário e confiança na capacidade de realizar o trabalho;
6. O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é através de conversa face a face;
7. Software funcionando é a medida primária de progresso;
8. Os processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente;
9. Contínua atenção à excelência técnica e bom design aumenta a agilidade.
10. Simplicidade - a arte de maximizar a quantidade de trabalho que não precisou ser feito;
11. As melhores arquiteturas, requisitos e *designs* emergem de equipes auto-organizáveis; e
12. Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo.

O processo ágil genérico pode apresentar diferentes métodos, cada qual com objetivos sutilmente diferenciados, atribuindo pesos distintos para os doze (12) princípios ágeis. Alguns métodos podem ignorar um ou mais princípios, porém, defendendo a filosofia ágil (espírito ágil) (PRESSMAN, 2011).

Segundo Sommerville (2011), Versionone (2010, 2011) os métodos mais utilizados são: Scrum[7] (SCHWABER; BEEDLE, 2001) e *eXtreme Programming* (XP)[8] (BECK; ANDRES, 2004; BECK; FOWLER, 2000), detalhados nas próximas seções. No entanto, existem outros métodos ágeis disponíveis na literatura: família Crystal (COCKBURN, 2004), *Adaptative Software Development* (ASD)[9] (HIGHSMITH, 1997), (HIGHSMITH, 2004), *Feature Driven Development*[10] (PALMER; FELSING, 2002), *Lean Software Development* (POPPENDIECK; POPPENDIECK, 2003), *Dynamic System Development Method*[11] (STAPLETON, 1997), entre outros.

A família Crystal propõe um conjunto de métodos voltados para diferentes equipes e projetos de software com diferentes níveis de complexidade (COCKBURN, 2004).

O método *Adaptive Software Development* é baseado na Teoria do Caos e visa auxiliar na imprecisão dos requisitos de software, objetivando maior adaptação às suas mudanças. É dividido em: exploração, colaboração e aprendizado (HIGHSMITH, 2004).

O método *Feature Driven Development* prevê em sua abordagem as fases de Concepção e Planejamento, e a fase de Desenvolvimento Iterativo. Seus modelos são elaborados a partir da linguagem UML estereotipada (PALMER; FELSING, 2002).

O método *Lean Software Development* é baseado no sistema de produção da empresa Toyota. Seu foco é eliminar desperdício, realizando seu desenvolvimento a partir da demanda do mercado (POPPENDIECK; POPPENDIECK, 2003).

O método *Dynamic System Development Method* prevê em sua abordagem o estudo de viabilidades seguido por um estudo de negócio. Diante disso, seguindo o conceito de desenvolvimento iterativo e incremental, são desenvolvidos: o modelo funcional, o design e a construção do sistema, e a implementação (STAPLETON, 1997).

[7] www.controlchaos.com
[8] www.extremeprogramming.org
[9] www.adaptivesd.com
[10] www.featuredrivendevelopment.com
[11] www.dsdm.org

Apesar dos métodos supracitados compartilharem dos mesmos princípios, cada um deles possui características específicas que devem ser analisadas em busca daqueles que melhor se adéquam ao projeto de software. Mais detalhes sobre processos ágeis podem ser encontrado em Pressman (2011, p. 81).

2.1.1. MÉTODO SCRUM

O método Scrum refere-se ao gerenciamento ágil de projetos, com foco em ciclos iterativos (SCHWABER; BEEDLE, 2001). Este não adota técnicas de engenharia de software ou práticas de programação (MUSHTAQ; QURESHI, 2012). Neste trabalho será abordado Scrum como método de gerenciamento de projetos de software.

É possível dividir Scrum em três fases distintas: 1) a fase de planejamento geral, onde são estabelecidos os objetivos do projeto e da arquitetura do software; 2) a fase de *Sprints*, onde em cada ciclo é desenvolvido um módulo adicional para o software; e 3) a fase de encerramento do projeto, onde são elaborados os documentos exigidos, manuais do usuário e catalogadas as lições aprendidas.

Os principais atores envolvidos em um projeto Scrum são: *Product Owner*, *Scrum Master* e o Time de desenvolvimento. O *Product Owner* representa o cliente que domina as demandas do produto e suas prioridades. O Scrum Master é o líder do projeto, responsável por resolver os impedimentos, manter o time dentro dos princípios Scrum, isolando-o de interferências externas. Cabe ao Scrum Master o acompanhamento, elaboração e divulgação da produtividade do time que muitas vezes ocorre através de gráficos (*burn down chart*). O time é representado por todos os membros, que apresentam como característica seu autogerenciamento, capacidade de tomar decisões e iniciativas quando preciso.

O conjunto de requisitos que descrevem todo o software é chamado de *product backlog*. No entanto, assim como preveem os princípios ágeis, estes podem ser alterados entre os ciclos de desenvolvimento conhecidos como *Sprints* - correspondem às *releases* da metodologia XP. Sempre que possível,

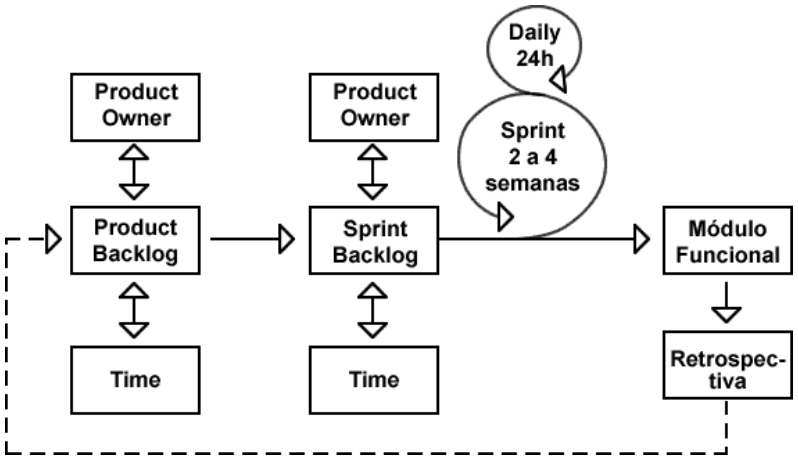
cada requisito do *product backlog* deve ser dividido em funcionalidades menores e mais simples de serem implementadas. Em seguida, todo o time deve mensurar o grau de complexidade e as horas necessárias para sua implementação. Caso existam opiniões divergentes, estas devem ser explicadas a todos em busca de um consenso (SCHWABER; BEEDLE, 2001).

Antes do início de cada *Sprint* é realizada uma reunião de planejamento chamada de *Sprint Backlog*, na qual estão presentes todos os atores envolvidos no projeto. Desta forma, são escolhidos os requisitos que serão implementados na *Sprint* de acordo com seu grau de prioridade e complexidade de desenvolvimento, buscando sempre a implementação do maior número possível de requisitos prioritários (SCHWABER; BEEDLE, 2001). É importante que todo o time acredite que a implementação dos requisitos escolhidos poderá ser realizada dentro do prazo estipulado, e que o cliente fique satisfeito com a versão que será entregue.

As *Sprints* apresentam duração fixa, normalmente de duas a quatro semanas. Ao início de cada dia é realizada uma reunião diária de 15 minutos, onde cada membro do time expõe: o que fez, qual é o plano para as próximas 24 horas, e quais são os impedimentos (SCHWABER; BEEDLE, 2001). Desta forma, o time pode replanejar o trabalho em um curto prazo. O Scrum Master é o líder democrático do projeto, não adotam a hierarquia *top down* (imperativa) como os gerentes de projetos tradicionais. Ao final de cada Sprint o time documenta as horas dedicadas ao desenvolvimento das histórias dos usuários.

Cada iteração é chamada de Sprint, a qual é responsável pelo desenvolvimento dos módulos funcionais. Em cada Sprint as funcionalidades descritas pelo cliente são avaliadas, os recursos necessários são selecionados e o projeto é implementado. Ao final de uma Sprint é entregue um módulo funcional do projeto aos *stakeholders*, como apresentado na **Figura 2.12.**

Figura 2.12 – Ciclo de vida do método Scrum



Fonte: o autor.

O Scrum não especifica como os requisitos do software serão levantados. Estes são especificados pelo *Product Owner* e são devidamente classificados de acordo com seu grau de prioridade.

Nos últimos anos o método Scrum foi o método ágil mais utilizado pela indústria de desenvolvimento de software (VERSIONONE, 2009, 2010, 2011, 2013). No entanto, ele não prevê auxílio às atividades voltadas para o desenvolvimento de software, necessitando, para esse fim, um método complementar.

2.1.2. MÉTODO XP

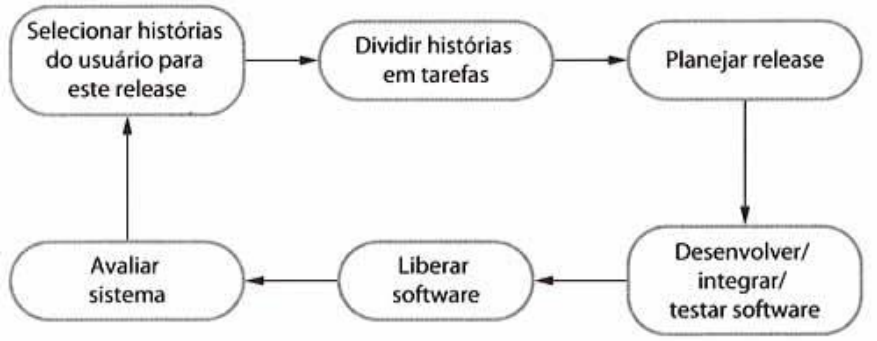
O método *eXtreme Programming* (XP) compartilha da filosofia ágil, porém, possui foco na fase de implementação e testes de software. Porém, segundo Mushtaq (2012), carece de atividades voltadas para a prática de gestão de projetos.

O autor do método XP, Kent Beck, lançou seu primeiro livro sobre este método em 1999 (BECK, 1999). No entanto, Beck e Andres (2004) detalharam o trabalho anterior, dividindo-o em práticas Primárias e Corolárias. A primeira reflete em melhorias explícitas instantâneas ao time. A segunda é baseada em resultados prévios, muitas vezes implícitos ao processo, conhecimento implícito adquirido após vivenciar a adoção do método XP, refere-se ao *know*

how, normalmente inerentes aos desenvolvedores mais experientes (BECK; ANDRES, 2004).

Esta seção descreve as boas práticas existentes no processo de desenvolvimento de software (BECK; ANDRES, 2004; BECK; FOWLER, 2000). A Figura 2.13 resume o processo de desenvolvimento XP (SOMMERVILLE, 2011).

Figura 2.13 – Fluxo de atividades do método XP



Fonte: Sommerville, 2011.

Conforme exibido na Figura 2.13, o método XP utiliza Histórias dos usuários para coletar requisitos da aplicação. Tais Histórias são subdivididas em tarefas mais simples, facilitando a estimativa de dificuldade para desenvolvê-las. Assim que definidas as tarefas, o time planeja a *release* de acordo com a prioridade do cliente e a dificuldade de implementação. Em seguida, inicia-se o processo de implementação, o qual é realizado de forma contínua. Novos requisitos são implementados diretamente na versão anterior, sem necessidade de integração (ou merge) entre diferentes versões. Dessa forma, todos os dias são geradas novas versões da aplicação. Em seguida é realizada a liberação da aplicação e a avaliação do sistema, retornando à atividade de coleta de requisitos através das Histórias do usuário.

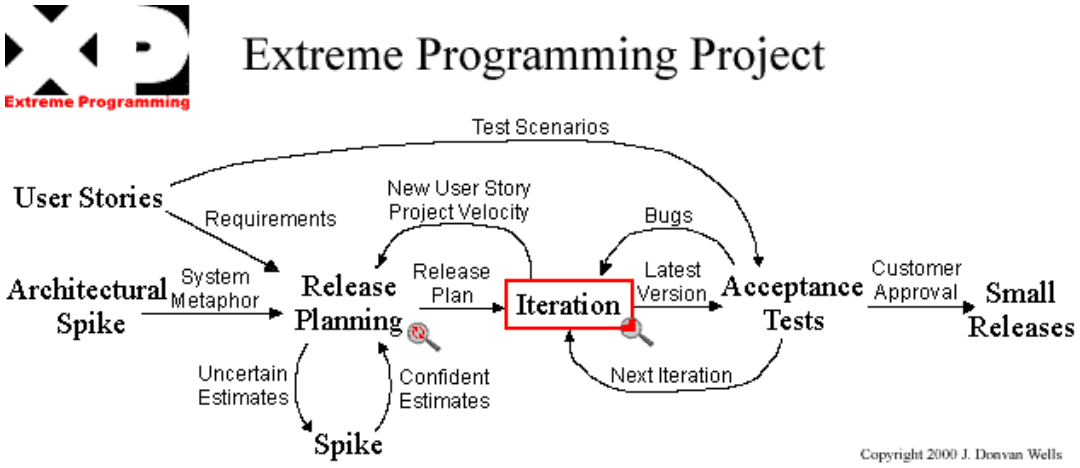
O método, no entanto, possui diversos princípios, valores, papéis e práticas fortemente definidos na literatura. Em Beck e Andres (2004) as práticas definidas em 1999 são evoluídas e adaptadas. Os princípios, valores papeis e as práticas primárias e corolárias são apresentadas em Beck e

Andres (2004). Informações adicionais também podem ser encontradas no site: *www.extremeprogramming.org*.

Um dos diferencias desse método está na programação em pares. Abaixo são descritas algumas vantagens dessa atividade (SOMMERVILLE, 2011): **a)** eleva o sentimento de propriedade coletiva do código; **b)** atua como um processo de revisão informal e mais barato; **c)** possibilita a refatoração imediata; **d)** a quantidade de erros diminui; **e)** permite a discussão da implementação, antecipando problemas, acarretando em menor retrabalho; e **f)** permite o compartilhamento do conhecimento entre os membros do time, minimizando o problema acarretado pela possível saída de um dos membros.

É possível elaborar diversos ciclos de vida do processo de desenvolvimento que contemplem os princípios, papéis, valores e práticas definidas pelo método XP. A Figura 2.14 mostra um modelo de processo de desenvolvimento composto pelas características previstas nesse método (WELLS, 2000), desde a coleta de requisitos, o planejamento do release, a iteração de desenvolvimento, que resulta na entrega de um módulo funcional ao cliente. Este módulo é devidamente testado, e, caso sejam encontradas falhas na versão apresentada, estas retornam ao ciclo de iteração. Caso contrário, a pequena versão desenvolvida é aprovada e disponibilizada para uso.

Figura 2.14 – Ciclo de vida do método XP



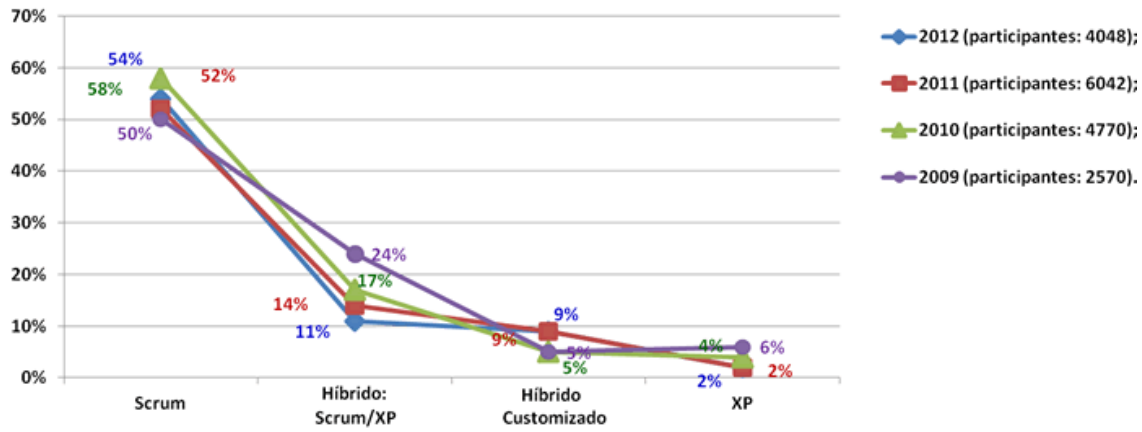
Fonte: Wells, 2000.

Todavia, vale ressaltar que as práticas apresentadas pelo método XP correspondem a uma estrutura genérica. Sua adoção integral pode não ser necessária para todos os projetos de software. Adotar integralmente tais técnicas pode acarretar em sobrecarga desnecessária ao processo de desenvolvimento. Para cada projeto e contexto do time de desenvolvimento, é importante que o engenheiro de software analise suas demandas e adapte os recursos oferecidos pelo método XP ao projeto que será desenvolvido, almejando um equilíbrio na relação custo/benefício, pois, conforme apresentado no início deste capítulo, é necessário adaptar um processo de software maduro de forma apropriada aos produtos e às demandas do mercado (PRESSMAN, 2011).

2.1.3. *SURVEY* SOBRE MÉTODOS ÁGEIS

A organização VersionOne atua na área de desenvolvimento de software empresarial, adotando processos totalmente ágeis para o desenvolvimento de software. Desde 2006 realiza anualmente um *survey* por meio de uma pesquisa onde os dados são coletados via um formulário individual publicado na web. Seu público alvo é composto por profissionais lotados em empresas de desenvolvimento de software em diferentes países. O Gráfico 2.2 apresenta uma síntese dos resultados divulgados nos quatro últimos relatórios (VERSIONONE, 2009, 2010, 2011, 2013).

Gráfico 2.2 – Síntese da adoção de métodos ágeis na indústria de software



Fonte: o autor.

O objetivo do Gráfico 2.2 é apresentar uma síntese dos métodos ágeis mais adotados nos últimos quatro (4) anos, ignorando portanto, quantidade total (valor absoluto) que variou de 2570 até 6042 participantes. Dessa forma, apresenta os quatro métodos mais adotados na indústria de software, nos últimos quatro (4) anos, cada linha representando o ano em que o *survey* foi realizado, dessa forma.

Em todos os anos, os métodos mais utilizados foram: 1º) Scrum; 2º) Método Híbrido de Scrum com XP; 3º) Método Híbrido customizado para a empresa; 4º) Método XP, exceto nas últimas pesquisas realizadas em 2011 e 2012, onde XP perdeu a 4ª posição, dividindo a 6ª posição com os métodos *Feature-Driven Development, Lean e outros*.

Entre os seis métodos ágeis mais adotados, a velocidade do desenvolvimento também é uma característica relevante, que se mostrou presente em 58% dos usuários que participaram da pesquisa Versionone (2013).

As três principais razões para adoção de métodos ágeis são: a) acelerar o tempo de lançamento do software no mercado; b) facilitar o gerenciamento de mudança de prioridades; e c) melhorar o alinhamento com o objetivo do negócio (VERSIONONE, 2012, 2013).

Os projetos que utilizam métodos ágeis oferecem resultados relevantes para a indústria de software, pois 70% dos participantes consideram que métodos ágeis apresentam o menor tempo para a conclusão de projetos (VERSIONONE, 2013).

Comparado com métodos tradicionais, 64% consideram que métodos ágeis são mais rápidos para a conclusão de projetos de desenvolvimento de software, e consideram essa característica um motivo para adotar tais métodos; 11% consideram que o tempo de desenvolvimento é equivalente; e 22% ainda não concluíram um projeto utilizando métodos ágeis. Apenas 3% consideraram os métodos ágeis mais lentos para o desenvolvimento de software (VERSIONONE, 2011).

Comparado à variedade de métodos ágeis disponíveis na literatura, Scrum possui uma relevante dianteira entre os métodos mais adotados na indústria de desenvolvimento de software. No entanto, existe uma demanda por métodos híbridos, pois aquele formado por Scrum junto com XP (Scrum/XP) ocupa segundo lugar entre aqueles mais utilizados, seguido pelo método híbrido customizado, voltado para atender a necessidade específica de cada projeto. O somatório dos métodos híbridos utilizados na indústria de desenvolvimento ágil de software alcança 23% do total de métodos utilizados, percentual que indica a demanda por métodos adaptados para os diferentes tipos de projetos e aplicações que surgiram nos últimos anos. Vale lembrar que existem diversos métodos híbridos, adaptados para atender a diferentes contextos.

2.2. CONSIDERAÇÕES SOBRE O CAPÍTULO

Cada tipo de aplicação apresenta necessidades específicas que devem ser respeitadas no momento da escolha do método que será adotado em seu desenvolvimento. Conforme apresentado no presente capítulo, cada método possui objetivos sutilmente distintos, atribuindo pesos diferentes para cada atividade, ou seja, cada método tende a concentrar esforços em atividades específicas.

Dessa forma, o presente capítulo abordou alguns dos principais métodos de desenvolvimento de software mais conhecidos na literatura, suas características e aplicabilidades. Os princípios inerentes aos métodos ágeis, como: ciclos iterativos; entrega contínua; impactos reduzidos na alteração de requisitos; foco nas necessidades do cliente e na entrega de software funcionando, potencializam a adequação de tais métodos ao processo de desenvolvimento de aplicações para TVD.

O capítulo também apresentou detalhes sobre os dois métodos ágeis mais utilizados na indústria de software, Scrum e XP. O primeiro tem foco no gerenciamento de projetos, o segundo tem foco em técnicas de engenharia de software, programação e testes de código fonte, respectivamente. Dessa forma, estes métodos apresentam características complementares.

Conforme apresentado em Versionone (2011), nos últimos anos, o método Scrum está na primeira posição entre os métodos mais utilizados na indústria de software. O método Híbrido, formados pela junção de Scrum com XP, ocupa a segunda posição. A terceira posição também pertence aos métodos híbridos, pela customização de diferentes métodos, visando atender às necessidades de projetos específicos. Tal resultado indica que adotar um método ágil híbrido é uma alternativa válida nessa indústria de desenvolvimento de software.

Diante das demandas variadas, um engenheiro de software pode especificar um método híbrido, combinando os pontos fortes originais de dois ou mais métodos já conhecidos pela literatura, customizados para atender uma demanda de mercado específica.

Scrum e XP estão entre os métodos de desenvolvimento ágeis mais adotados na literatura (MUSHTAQ; QURESHI, 2012). Segundo Mushtaq e Qureshi (2012), as etapas do ciclo de desenvolvimento de software XP podem ser totalmente alinhadas com Scrum, formando um método híbrido capaz de auxiliar aos projetos que demandam pela integração de atividades de gestão de projetos, com atividades voltadas para engenharia de software, visando melhorar a qualidade do software do ponto de vista de todos os *stakeholders* (MUSHTAQ; QURESHI, 2012).

No entanto, estruturar um método híbrido não é uma tarefa simples (MUSHTAQ; QURESHI, 2012), pois é preciso analisar diversos fatores como: as características do projeto de software que será desenvolvido, o contexto no qual a aplicação será executada, qual o perfil do cliente, como serão coletados os requisitos, quem são os usuários e como estes utilizarão a aplicação.

Diante disso, é preciso pesquisar quais métodos se aproximam mais das necessidades dos projetos, apontando quais características de cada método serão incorporadas, e por fim, especificar como suas atividades serão realizadas.

Baseado nisso, o primeiro experimento realizado pela presente tese faz uma investigação sobre a adequação ao desenvolvimento de aplicações para TVD e possíveis pontos de melhoria dos métodos Scrum, XP e do método

híbrido formado por ambos. O método híbrido avaliado é baseado na proposta de Mushtaq; Qureshi (2012), abordado no próximo capítulo, Seção 3.4.

CAPÍTULO

3.

3. TRABALHOS RELACIONADOS

"A irritação é o meio de congelar os próprios interesses."
Chico Xavier

Os trabalhos relacionados levantados durante o desenvolvimento da presente tese atravessam, de forma transversal, diferentes áreas de conhecimento. Dessa forma, o presente capítulo discute primeiramente aqueles trabalhos voltados para o **desenvolvimento de aplicações para TVD**, apresentando demandas inerentes ao processo de desenvolvimento, reforçando suas peculiaridades, conforme descrito nas seções: **3.1)** Story to Code – Modelagem de Aplicações Multimídia em TV Digital Interativa (MARQUES NETO, 2011); **3.2)** *Facilitating TV production using StoryCrate* (BARTINDALE et al., 2013); **3.3)** *Kroster-MHP: developing process, design, programming and considerations* (QUINTERO et al.; 2013); **3.4)** Utilização de métodos ágeis no desenvolvimento do conteúdo audiovisual para TVD (VEIGA, 2006) (VEIGA; TAVARES, 2006, 2007); e **3.5)** Um Processo Ágil para Especificação de Requisitos em Programas Interativos (LULA; GUIMARÃES; TAVARES, 2011). Em seguida, são apresentados os trabalhos voltados para a utilização, adaptação e avaliação de **métodos ágeis** são abordados nas seguintes seções: **3.6)** Uso eficaz de métricas em métodos ágeis de desenvolvimento de software (SATO, 2007); **3.7)** XWebProcess - um processo ágil para o desenvolvimento de aplicações web (SAMPAIO, 2004); **3.8)** XPU:

Uma Extensão do Método XP Visando à Usabilidade (SILVA, 2009); **3.9)** *Novel Hybrid Model: Integrating* Scrum *and* XP (MUSHTAQ; QURESHI, 2012); **3.10)** Análise multicritério e a ferramenta PROMETHEE II (CAMPOS, 2011); e **3.11)** Considerações sobre o capítulo.

3.1. Story to Code – Modelagem de Aplicações Multimídia em TV Digital Interativa

A tese intitulada Contribuições para a Modelagem de Aplicações Multimídia em TV Digital Interativa (MARQUES NETO, 2011) parte do princípio de que a concepção de aplicações para TVD deve responder a três questões: a) como estruturar os conteúdos das aplicações? b) como definir a dinâmica de transmissão e recepção? c) como modelar e construir aplicações voltadas para o reuso de componentes?

Segundo Marques Neto (2011), a definição de processos e métodos voltados para a modelagem de programas para TVD pode ser considerada um problema em aberto. A maior parte dos trabalhos relacionados não apresenta uma solução formal para a estruturação de requisitos e reuso de software, sendo, segundo o autor, a ausência de modelos de processos de desenvolvimento um dos principais problemas referentes ao desenvolvimento de aplicações para TVD.

Marques Neto (2011) apresenta o modelo denominado *Interactive Digital Tv Show* (IDTVS), que pretende estruturar dados adicionais ao conteúdo audiovisual da TV utilizando, para isso, um *framework* para a organização de código orientado a objeto na emissora e no receptor de TV. O IDTVS foi utilizado como base para o modelo StoryToCode (MARQUES NETO, 2011).

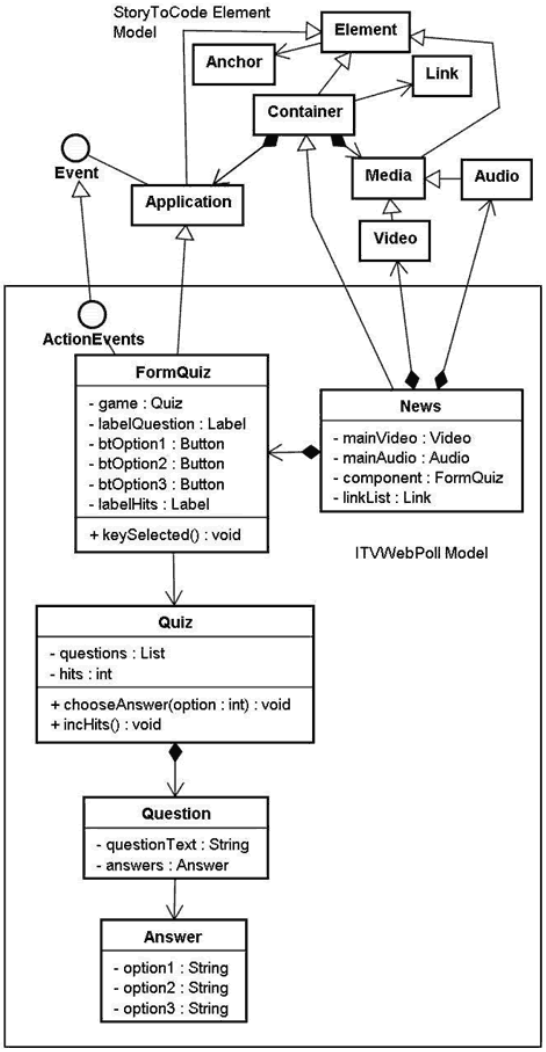
O objetivo do StoryToCode é oferecer um novo processo de concepção de componentes interativos voltados para TVD, viabilizando a especificação, construção e reaproveitamento de componentes de software para diferentes plataformas (TVD, *Mobile* e Web). Entre suas limitações está a extração de requisitos de forma não automatizada e o retrabalho necessário para completar o código gerado.

Esse modelo trata as atividades de desenvolvimento de aplicações multimídia, auxiliando entre outros pontos na definição de classes (utilizadas

no paradigma Orientado a Objeto) e sua codificação, em colaboração com a equipe de TV. Para isso, parte-se de um *storyboard* (cenário) onde são utilizados conceitos de modelagem de sistemas para definir elementos que representem mídias e componentes de software. O *storyboard* deve ser a fonte de obtenção de requisitos das aplicações. A partir de ferramentas CASE é definida a arquitetura de elementos que serve como um modelo base de estruturação dos requisitos em classes. A geração de código é realizada de forma semi-automática utilizando motores de transformação. Dessa forma, o modelo define como realizar a transformação de um *sotryborad* em elementos abstratos, e, em seguida, em código fonte.

Marques Neto (2011) ressalta a falta de um padrão capaz de auxiliar o desenvolvimento de aplicações multimídia. Tal carência implica no aumento do tempo necessário para a conclusão do desenvolvimento. Afirma ainda que linguagens de modelagem visual como UML podem ser uma alternativa válida para auxiliar na especificação de uma aplicação, pois oferecem um meio de comunicação adequado a diferentes perfis profissionais, porém a linguagem UML não é suficiente para a modelagem multimídia, pois não auxilia a modelagem da interface de usuário, nem a modelagem de mídias. A Figura 3.1 apresenta um exemplo do modelo de elementos StoryToCode.

Figura 3.1 – Exemplo do modelo de elementos StoryToCode



Fonte: Marques Neto, 2011.

A Figura 3.1 representa um diagrama completo do conjunto de classes de uma aplicação, onde, a partir do *Storyboard* e da lista de requisitos funcionais, inicia-se a fase do StoryToCode através da criação de alguns elementos, como *News*, *FormQuis*, entre outros que representam abstrações das regras de negócio de uma aplicação de perguntas e respostas (*Quiz*).

O autor apresenta três exemplos de uso do modelo proposto, entre eles, o desenvolvimento de um jogo educativo multiplataforma (TVD, Web e *mobile*). Por fim, o autor conclui que o modelo StoryToCode é viável para o desenvolvimento de aplicações multimídia, e o reaproveitamento de código é promissor.

Segundo Marques Neto (2011), o método XP apresenta algumas dificuldades para desenvolvimento de aplicações multimídias, como: testes de unidades e testes funcionais realizados primeiro (*test-first*). O autor condiciona o sucesso na adoção de métodos ágeis à realização de adaptações às características do contexto dos conteúdos multimídia.

Marques Neto (2011) afirma que, devido à abordagem menos formal e à estreita relação do time de desenvolvimento com os usuários, os métodos ágeis são candidatos a serem adotados para o desenvolvimento de aplicações multimídia. Marques Neto (2011) afirma ainda que: "*abordagens ágeis se encaixam perfeitamente com a elevada ocorrência de mudanças de requisitos no desenvolvimento de aplicações multimídia"*. No entanto, segundo as conclusões de Marques Neto (2011), ainda é um desafio em aberto escolher e adotar um método que atenda às necessidades do desenvolvimento de aplicações para TVD, principalmente por suas características multimídia e multidisciplinaridade.

A presente tese concorda com a pesquisa apresentada por Marques Neto (2011), considerando os métodos ágeis como fortes candidatos ao desenvolvimento de aplicações para TVD, porém, não é prudente generalizar. Vale lembrar que segundo Sommerville (2011), não existe um método universal, diferentes projetos demandam por métodos diferentes.

Apesar do objetivo do trabalho de Marques Neto (2011) ser distinto, voltado para a atividade de reuso de código, sua pesquisa apresenta características que motivam e contribuem com a presente tese, pois deixa em aberto o problema voltado para a carência inerente ao levantamento de requisitos multimídia, destacando a necessidade de desenvolvimento rápido de aplicações para TVD, apresentando a necessidade de adaptações dos métodos ágeis às aplicações de TVD, e por fim, afirma que um método de desenvolvimento adequado ainda é um desafio em aberto.

Diante disso a presente tese realizou experimentos com o objetivo de avaliar o desempenho dos métodos ágeis mais utilizados na literatura, conforme apresentado no Capítulo 4 e Capítulo 6.

3.2. *Facilitating TV production using StoryCreate*

A produção do conteúdo audiovisual é um processo demorado e caro que envolve um grande número de profissionais qualificados e experientes em diversas áreas do conhecimento. Produções de grande orçamento utilizam *storyboards* para planejar e reduzir os custos de refilmagem e gerenciar riscos (BARTINDALE et al., 2013). Normalmente um *storyboard* corresponde a uma sequência de desenhos estilizados, que podem representar o conteúdo que se pretende desenvolver. Um *storyboard* é um meio familiar, simples de usar, flexível e capaz de representar o conteúdo em produção de um modo bem compreensível.

A filmagem do conteúdo audiovisual é um processo criativo que apresenta limitações de tempo para sua produção. Devido à diversidade de profissionais envolvidos, deve ter sua comunicação facilitada (BARTINDALE et al., 2013). Apesar da multidisciplinaridade das equipes, estas normalmente compartilham um padrão de fluxo de trabalho comum, cujas principais etapas são apresentadas na **Figura 3.2**.

Figura 3.2 – Desenvolvimento do conteúdo audiovisual

Fonte: Bartindale et al., 2011.

Muitas tecnologias de produção visam integrar mídia e metadados em todo o fluxo de trabalho (BARTINDALE et al., 2013). A proposta do trabalho apresentado por Bartindale (2013) pretende viabilizar alterações e decisões criativas durante o desenvolvimento do projeto, antecipando modificações que, tradicionalmente, são realizadas apenas na etapa de edição, ou seja, após a produção do conteúdo.

Para isso, apresenta um protótipo colaborativo denominado StoryCreate. Este protótipo se baseia em *storybord* para oferecer um ambiente capaz de auxiliar no processo de produção de conteúdo audiovisual, contribuindo para: criatividade, comunicação da equipe, tomada de decisão e redução do tempo dedicado para a edição do conteúdo audiovisual. Sua estrutura pretende tornar

o processo de produção de conteúdo multimídia facilmente compreendido, através de uma infraestrutura tecnológica que oferece um fluxo de trabalho para melhorar a eficiência no processo de produção.

StoryCreate é uma aplicação de software desenvolvida com Microsoft .NET, executada sobre uma mesa interativa cuja infraestrutura é formada por duas telas e controlados por dois PlayStation 3 e acesso à rede. StoryCreate é pré carregado com *storyboards*, pode incluir o roteiro, descrições, imagens, filmes e *storyboard* que podem ser editados digitalmente. Esta ferramenta apresenta uma linha do tempo linear, onde o desenvolvimento do projeto ocorre de forma incremental, cada item de mídia é representado como uma miniatura da mídia no visor. Através de uma caneta digital é possível adicionar (à mão livre) novos objetos de mídia nos *storybords*, promove a visibilidade dessas ações e facilitando a entrada de dados de forma rápida e espontânea. Ao fim do projeto, é possível exportar a linha do tempo como um arquivo XML.

O StoryCreate reproduz e edita o *storyboard* em formato digital, como uma representação dinâmica compartilhada para uso colaborativo por membros da equipe de produção, facilitando a representação de uma ampla variedade de tipos e estilos de interação, apresentando mais rapidamente as ideias dos profissionais envolvidos.

Bartindale (2013) ressalta a importância em reduzir barreiras físicas e fomentar o acesso de todos os membros da equipe à ferramenta proposta, como forma de tornar explicito o objetivo do projeto que se pretende desenvolver, incentivando desse forma a comunicação e a proposta de novas ideias.

Os participantes do estudo de caso realizado por Bartindale (2013) destacaram que um *storybord* apresenta uma relação de custo e benefício satisfatória para o desenvolvimento do projeto, evitando desperdício de tempo. Viabilizar que toda a equipe envolvida no projeto visualize e interaja com os *storybords* desenvolvidos permite uma maior consciência do progresso, facilita a criatividade e propriedade sobre o conteúdo, permitindo direcionar decisões tomadas diretamente no fluxo de trabalho explicito (BARTINDALE et al., 2013).

A presente tese compartilha da importância da utilização de *storybord* sugerida por Bartindale et al. (2013), porém, utiliza tal artefato para auxiliar o desenvolvimento de software.

Conforme apresentado no Capítulo 1, a presente tese considera que aplicações para TVD são coadjuvantes, ou seja, pretende agregar valor ao conteúdo audiovisual. Dessa forma, é interessante que a equipe de desenvolvimento de software utilize ferramentas e técnicas já utilizadas pela indústria de TV. Além de *storybords* oferecer vantagens quanto à comunicação, agilidade para realizar modificações e economia de tempo para ser desenvolvido, incorporar ao processo de desenvolvimento de software ferramentas utilizadas pela equipe responsável pelo conteúdo audiovisual da TV evita o retrabalho e facilita a comunicação entre os diferentes profissionais. Diante disso, a presente tese incorpora a utilização de *storybord* ao método de desenvolvimento proposto.

3.3. *Kroster-MHP: developing process, design, programming and considerations*

Em Quintero et al. (2013) é desenvolvida a biblioteca Java TVGame cujo objetivo é auxiliar a implementação de aplicações para a plataforma MHP. Mais especificamente, esta biblioteca é voltada para o gerenciamento das quinze (15) camadas simultâneas que limitam o *middleware* MHP e a manipulação dos objetos de mídia, visando abstrair eventos comuns (loaded, paused, started e destroyed) inerentes à sincronização das mídias, tornando o desenvolvimento mais rápido e simples.

Para avaliar sua biblioteca foi realizado o desenvolvimento do t-game educativo denominado “Kroster”. A biblioteca TVGame simplifica a gestão de imagens, colisões e animações durante o desenvolvimento do jogo. Isso deu liberdade para o programador se concentrar nas ações e movimentos no cenário de jogo, em vez de colocar a atenção sobre a sincronização de diferentes elementos e suas localizações na tela, conforme apresentada pela sua interface na **Figura 3.3**.

Figura 3.3 – Interface do *t-game* Kroster

Fonte: Quintero et al., 2013.

O primeiro passo para a criação do *t-game* Kroster foi esboçar no papel suas funcionalidades e sua estrutura geral. O jogo é formado por três níveis (fases). No primeiro nível do jogo existia mais de uma centena de elementos (objetos multimídia) em cena, dificultando a sincronização do movimento e a localização dos diferentes elementos da tela.

Quintero et al. (2013) apresenta as características e as dificuldades encontradas durante o processo de desenvolvimento. Entre elas, o autor ressalta que o desenvolvimento de aplicações para TVD é limitado por vários fatores técnicos, que envolvem tanto o hardware como o software, afetando diretamente a maneira como o aplicativo é desenvolvido. Destaca ainda a necessidade de uma equipe multidisciplinar para atuar no processo de desenvolvimento.

Devido à necessidade de ajustes encontrados durante o desenvolvimento do projeto, Quintero et al. (2013) considera que o processo de concepção adotado deva ser iterativo, ajustando os problemas encontrados no decorrer do projeto sempre que necessário.

O trabalho apresentado por Quintero et al. (2013) expõe uma série de dificuldades voltadas para a obtenção, implementação e sincronismo dos diversos requisitos multimídias como: eixo horizontal (x), vertical (y) e pilha de

mídias (z), distância, largura e altura de cada objeto de mídia, reforçando os indícios que levam a necessidade de um artefato capaz de documentar e formalizar a representação visual da aplicação e seus objetos de mídia. Tais dificuldades foram importantes para reforçar a demanda e a fundamentação dos artefatos propostos pela presente tese.

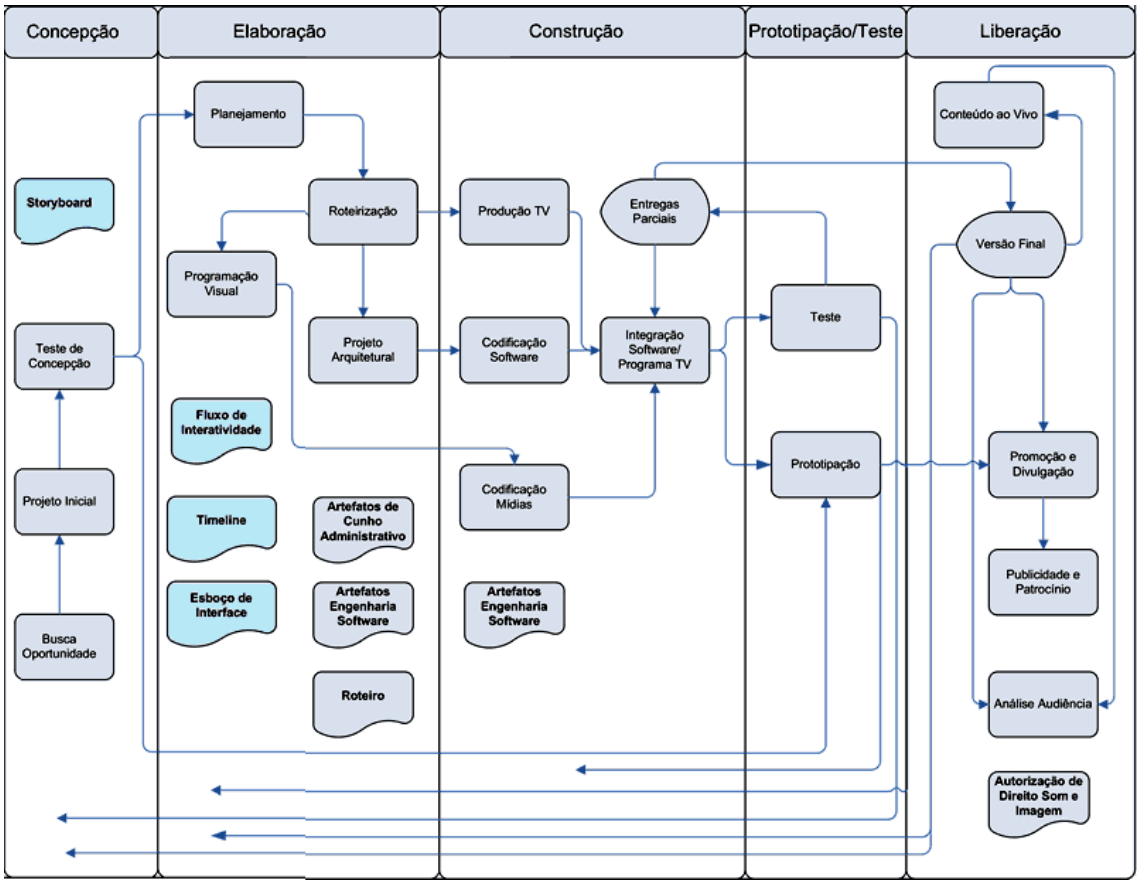
3.4. Utilização de métodos ágeis no desenvolvimento do conteúdo audiovisual para TVD

Esta seção apresenta uma síntese sobre uma sequência de publicações, que correspondem à evolução dos resultados ao longo do desenvolvimento da pesquisa por seus autores (VEIGA, 2006), (VEIGA; TAVARES, 2006, 2007). Os trabalhos apresentam a proposta de um modelo de processo cujo principal objetivo é integrar princípios dos métodos Scrum e XP em conjunto com práticas inerentes ao desenvolvimento do conteúdo audiovisual da TV, fornecendo um modelo de desenvolvimento que abrange a interseção dos dois conteúdos (audiovisual e software). Para isso, são identificadas práticas, etapas, *stakeholders*, responsabilidades e artefatos essenciais.

Veiga (2006) destaca que a necessidade de flexibilidade e agilidade inerente ao contexto de TVD resulta na iminente adoção aos métodos ágeis, viabilizando a realização de alterações de requisitos de forma eficiente.

A adoção pelos métodos Scrum e XP é baseada na contribuição inerente à integração das características de ambos, enquanto o primeiro oferece técnicas de gerenciamento, o segundo provê técnicas de engenharia de software que auxiliam no desenvolvimento de aplicações, de modo complementar possibilitando adotá-los com sucesso no contexto de TVD. O modelo e os artefatos propostos são definidos em cinco fases, exibidas na **Figura 3.4**.

Figura 3.4 – Modelo de processo proposto

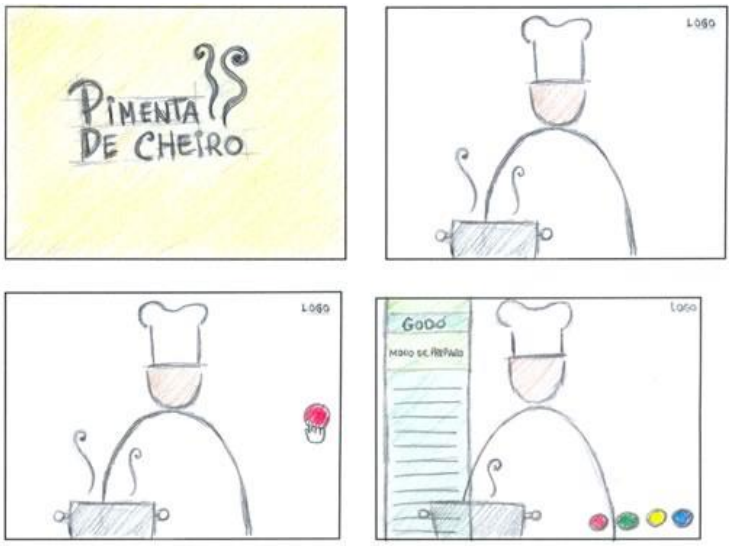


Fonte: Veiga, 2006.

O modelo proposto por Veiga (2006) prevê cinco fases apresentadas através das colunas da Figura 3.4: Concepção, Elaboração, Construção, Prototipação / Teste e Liberação.

A fase de Concepção do conteúdo audiovisual é realizada com todos os membros da equipe, desde roteiristas até os gerentes de projeto, que devem avaliar o objetivo antes de iniciar o desenvolvimento. Veiga (2006) estabelece que os objetivos da aplicação devem ser apresentados através de *storyboard*, apresentado na **Figura 3.5**, ou seja, uma sequência de quadros que descrevem um enredo. A implementação deve ser dirigida a testes (*test-first*), a validação do projeto inicial deve ser conduzida pelo produtor de TV com o apoio do engenheiro de testes. No entanto, o método indica que tal validação pode ocorrer por meio de protótipos para facilitar a compreensão e garantir a objetividade do ciclo.

Figura 3.5 – Exemplo da utilização de *Storyboard* para aplicações de TVD



Fonte: Veiga, 2006.

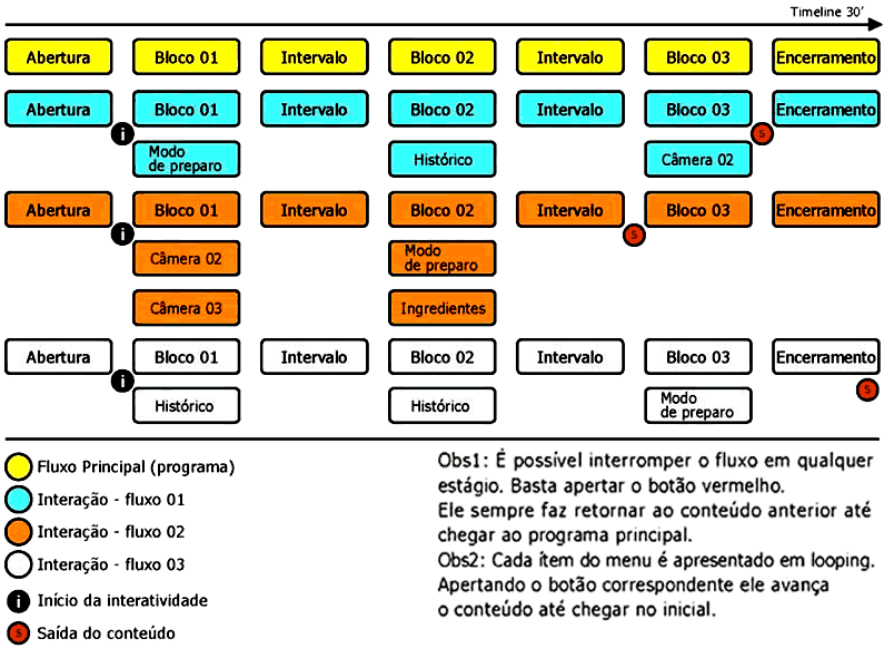
O *storybord* exibido na **Figura 3.5** é um exemplo de representação de cenas através de quadros, que pretende descrever o enredo de um filme ou uma peça. A autora adiciona os botões de interatividade e apresenta um esboço visual da interação do usuário com a aplicação.

O desenvolvimento do roteiro do vídeo deve conter, além da descrição textual do seu conteúdo, indicações que apontem as necessidades especiais da aplicação. É esperado que o artefato responsável por definir o roteiro passe por diversas revisões até sua finalização. Segundo Veiga (2006), o roteiro está para o modelo proposto, assim como a especificação de requisitos está para a engenharia de software.

O projeto arquitetural deve descrever exatamente a aplicação que será desenvolvida, contemplando suas funcionalidades, o plano de interatividade e os elementos de mídia necessários. Entre as práticas sugeridas do método XP, está a programação em pares e a participação do cliente. Veiga (2006) ressalta ainda que a prototipagem de aplicações é necessária para viabilizar a compreensão dos requisitos da aplicação por toda a equipe, reduzindo o tempo e o custo do desenvolvimento.

O modelo indica os seguintes artefatos para guiar o desenvolvimento: *storyboards*; *timeline*; fluxo de interatividade e esboço de interface. Um exemplo da utilização de *timeline* é mostrado na Figura 3.6.

Figura 3.6 – Exemplo do artefato Timelime



Fonte: Veiga, 2006.

Devido à reconhecida dificuldade em realizar experimentos que envolvam o desenvolvimento de aplicações para TVD, no experimento realizado por Veiga (2006) não foi feita a implementação de qualquer aplicação. Dessa forma, não foi possível avaliar o modelo de processo proposto. O experimento realizado envolveu doze (12) alunos de uma turma de especialização em Design de Conteúdo Digital, na qual a maioria dos participantes não possuía conhecimento sobre engenharia de software. Os alunos foram divididos em grupos, com no máximo quatro membros. Estes grupos tinham o objetivo de elaborar e avaliar os artefatos de aplicações já implementadas.

Aquele experimento iniciou com uma apresentação de quatro horas sobre o processo proposto. A avaliação dos artefatos ocorreu através de um questionário formado por vinte e três (23) questões objetivas e descritivas. As

questões objetivas ofereciam quatro alternativas de respostas: a) imprescindível; b) importante; c) necessário; e d) irrelevante (VEIGA, 2006).

Entre os resultados obtidos pode se destacar que 75% dos entrevistados indicam que é imprescindível a elaboração de um modelo conceitual da aplicação. Todos os entrevistados avaliaram o artefato *Storyboard* como “imprescindível”, no entanto, foi sugerido adicionar uma sequência de interações com o usuário.

Sobre o artefato referente ao “Fluxo de Interatividade”, 74% dos entrevistados julga este como factível, sendo para 62% imprescindível, e para 12% importante. Quanto ao artefato “Esboço de Interface”, 56% o avaliou como imprescindível e 33% o julgam importante.

Segundo Veiga (2006), para atender às necessidades do conteúdo de TVD, deve ser priorizado o envolvimento dos membros da equipe em um modelo de desenvolvimento rápido, enxuto e com o mínimo de documentação necessária.

Tanto o modelo de processo como os artefatos propostos por Veiga (2006) apresentam objetivos sutilmente diferentes da presente tese. Enquanto o escopo do trabalho de Veiga (2006) pretende auxiliar na integração e comunicação entre a equipe de desenvolvimento de software e profissionais de TV, voltados para a produção do conteúdo audiovisual, a presente tese abstrai o desenvolvimento do conteúdo audiovisual da TV, e se aprofunda no desenvolvimento de aplicações, oferecendo um método rápido e adequado às suas peculiaridades. Dessa forma, ambos os trabalhos podem ser adotados de forma complementar, cada um em seu escopo.

Além disso, o modelo proposto por Veiga (2006) prevê a realização de testes de unitários, testes de integração, testes de desempenho, testes de usabilidade, teste de Infraestrutura e, por fim, teste de revisão tipográfica. Porém, sua pesquisa não especifica como estes devem ser realizados.

Apesar das atividades de testes serem fundamentais para aplicações de TVD, a presente tese considera que a quantidade de testes proposta por Veiga (2006) pode acarretar em retrabalho do time e lentidão no processo de desenvolvimento de software, sem, necessariamente, aprimoramento da

aplicação desenvolvida, necessitando de uma avaliação criteriosa da relação custo/benefício sobre a adoção de cada tipo de teste. Como a atividade de teste de software demanda por pesquisas e experimentos específicos, tais atividades estão fora do escopo da presente pesquisa.

Apesar do experimento realizado em Veiga (2006) não realizar o desenvolvimento de aplicações e a validação do modelo proposto, sua pesquisa oferece importantes indícios sobre artefatos desejáveis no processo de desenvolvimento, que podem ser utilizados para auxiliar a realização dos objetivos da presente tese, como por exemplo, a evolução do *storybord* e a demanda pela elaboração de um modelo conceitual da aplicação.

3.5. Um Processo Ágil para Especificação de Requisitos em Programas Interativos

Em Lula, Guimarães e Tavares (2011) é apresentado um processo ágil para especificação de requisitos em conteúdos interativos, integrando o audiovisual às aplicações de software. Para isso, são adaptados conceitos de métodos ágeis em busca da agilidade necessária ao ambiente de TV.

O processo proposto adota, fortemente, as histórias de usuários, integradas aos artefatos já conhecidos das indústrias de TV e de software, e apresenta como foco os roteiros literários (cenas, diálogos entre personagens, entre outros.) e roteiros técnicos (iluminação, efeitos, câmeras, entre outros.) de TV. Ambos os roteiros são desenvolvidos através de descrições textuais. Sua abordagem utiliza tais roteiros como entrada de dados para as atividades de especificação de requisitos e criação das histórias de usuários visando sistematizar as informações dos *storyboards*.

Os roteiros literários e técnicos são elaborados exclusivamente pela equipe de TV. Ambos devem auxiliar na especificação dos requisitos da aplicação. Todas as informações necessárias para o desenvolvimento da aplicação devem ser retiradas de tais roteiros. Estes são avaliados pela equipe de TV em conjunto com a equipe de software.

A equipe de software é responsável por desenvolver as histórias dos usuários voltadas para o desenvolvimento da aplicação interativa. É possível utilizar protótipos através de imagens e animações. As histórias desenvolvidas

são validadas por ambas as equipes. Baseado nisso, a aplicação é implementada e testada. Na **Figura 3.7** é apresentado o processo proposto.

Figura 3.7 – Processo para integração das equipes de software e TV

Fonte: Lula, Guimarães e Tavares, 2011.

A avaliação do processo proposto ocorreu através de um experimento no qual foram desenvolvidos, tanto o conteúdo audiovisual, como as aplicações, de doze programas interativos. Os dados foram coletados através de dois questionários, um voltado para aplicação, o outro para a produção do conteúdo audiovisual.

Entre os resultados obtidos, foram apresentadas dificuldades para representar a aparência de algumas aplicações na tela através do roteiro: informações como posicionamento e transparência são difíceis de descrever textualmente. Para representar textualmente uma imagem localizada entre a margem esquerda e o centro da tela (eixo x), e entre o centro da tela e a margem superior (eixo y), é preciso saber o local exato da localização dessa imagem, para definir sua localização em pixels ou porcentagem na linguagem de programação multimídia.

As principais conclusões obtidas pelos autores referem-se à necessidade de agilidade no desenvolvimento das aplicações interativas, e indicam a necessidade de ter um profissional de software junto à equipe responsável pelo audiovisual da TV. A interatividade deve ser planejada desde o início da produção do audiovisual. O roteiro deve indicar o momento que a

aplicação interativa será iniciada. Ressaltam ainda que as histórias dos usuários podem complementar a especificação da interatividade dos roteiros, e pode guiar a fase de testes (LULA; GUIMARÃES; TAVARES, 2011).

Entre as dificuldades foi destacado que profissionais de TV julgam difícil descrever de forma sistemática as atividades de interatividade. A equipe de software deve se adequar às solicitações de mudança de requisito, atendendo de forma rápida e adequada ao contexto do ambiente de TV e a entrega do software funcional.

Segundo Lula, Guimarães e Tavares (2011), mesmo diante das diversas abordagens disponíveis na literatura, saber como executar as etapas do processo de desenvolvimento de software com eficiência ainda é um desafio em aberto.

A presente tese pretende atuar no ponto em que Lula, Guimarães e Tavares (2011) classifica como “um desafio em aberto”, oferecendo atividades bem definidas de um método de desenvolvimento de software rápido e adequado ao contexto das aplicações de TVD.

O procedimento adotado por Lula, Guimarães e Tavares (2011) para realizar a coleta de requisitos e demais características dos objetos de mídia através de descrições textuais utilizando roteiros, não é a melhor abordagem a ser adotada, pois estes apresentam forte característica visual. Informações como "*canto superior direito*" não é eficiente e dificulta a apresentação de objetos de mídia que não se localizam nas extremidades. Para realizar a coleta de requisitos multimídia, a presente tese elaborou o artefato denominado Protótipo de Aplicações para Tv (PATv), que corresponde a um protótipo de baixo nível de fidelidade, baseado em *storyboards*, visando atender de forma simples, rápida e de fácil entendimento tanto para os profissionais de TV como também para os engenheiros de software, auxiliando na comunicação do time.

3.6. Uso eficaz de métricas em métodos ágeis de desenvolvimento de software

Sato (2007) investiga o uso de métricas para o acompanhamento de projetos que adotam métodos ágeis no desenvolvimento de software. Sua dissertação pretende auxiliar o *tracker*, papel definido no método XP, que visa

coletar métricas de produtividade para auxiliar sua equipe a entender e aprimorar o desenvolvimento do software.

Para isso, o autor realiza um estudo de caso avaliando as métricas inerentes a sete projetos acadêmicos e governamentais que adotaram o método XP. Os resultados foram coletados de modo quantitativo e qualitativo por meio de: código-fonte, repositório, entrevistas e questionários. O autor (Sato) participou como *coach* e monitor de cinco projetos acadêmicos nos quais cada estudante trabalhava entre 6 e 8 horas por semana. O autor atuou também como consultor de outros dois projetos governamentais, nos quais cada membro trabalhava, em média, 30 horas por semana.

Antes da realização dos estudos de caso sobre os projetos acadêmicos, o autor ministrou um treinamento a fim de nivelar o conhecimento dos membros de cada time. A maioria dos membros não possuía conhecimento sobre métodos ágeis, e esse treinamento envolveu: práticas XP e ferramentas utilizadas.

Estes projetos tinham características variadas, com diferentes focos, voltados para: web, open source, *stand-alone ou mobile*. As seguintes variáveis do processo de desenvolvimento não foram controladas: quantidade de estudantes por equipe, que variou de 4 a 9; quantidade de meses para análise de projeto; nível do estudante (graduação e pós-graduação) e sua experiência.

Os resultados obtidos apontam que os projetos que adotaram as menores quantidades de práticas ágeis, apresentaram maior quantidade de falhas e maior esforço na realização de testes e manutenção.

Algumas características utilizadas por Sato (2007) foram adaptadas e incorporadas aos experimentos realizados pela presente pesquisa, tais como: adoção e treinamento de alunos para realizar experimentos, bem como os instrumentos de coleta de dados de modo quantitativo e qualitativo através de: código-fonte, entrevistas e questionários.

No entanto, ao contrário da pesquisa realizada por Sato (2007), a presente tese avalia que a comparação da produtividade entre diferentes times não pode se basear em *story-points*, pois essa técnica é baseada no conhecimento prévio dos membros de cada time, permitindo que, enquanto um

time estima 1000 pontos, o outro pode estimar 600 para realizar a mesma atividade. Dessa forma, não é adequado utilizar *story-points* para mensurar a velocidade do desenvolvimento de um time formado por alunos, pois, é possível que essa variável seja supervalorada atribuindo mais pontos do que o necessário para a atividade que será desenvolvida, aumentando, em teoria, a velocidade do desenvolvimento.

3.7. XWebProcess - um processo ágil para o desenvolvimento de aplicações web

Em Sampaio (2004) foi elaborado um processo ágil para o desenvolvimento de aplicações web denominado XWebProcess, baseado no método XP. O principal foco daquele trabalho foi oferecer aos desenvolvedores de aplicações web um método ágil que visa auxiliar o desenvolvimento de qualidade de forma rápida e eficiente. Visando maior contribuição com o desenvolvimento Web, o método XWebProcess mantém algumas características de XP inalteradas, outras modificadas e algumas adicionadas.

Para avaliar o método XWebProcess, foi realizado um experimento com quarenta (40) estudantes de graduação, formando oito grupos, cada um com cinco (5) pessoas. A distribuição dos alunos entre os grupos foi aleatória. Quatro grupos utilizaram XWebProcess e os outros quatro grupos XP para implementar a mesma aplicação que corresponde a uma livraria virtual web. O cliente desse sistema foi o próprio autor daquela dissertação.

Todos os grupos tiveram iguais condições de trabalho, ferramentas e adotaram a mesma linguagem de programação. Dessa forma, o autor reduziu as variáveis de interferência, coletando o esforço que cada time dedicou na implementação da aplicação. Para isso, cada membro do time preencheu um formulário constando as horas de trabalho dedicadas na implementação das funcionalidades da aplicação.

Ao fim da implementação do projeto foi calculada a média aritmética das horas de trabalho dos membros de cada time. Os valores absolutos foram próximos, totalizando XWebProcess = 113,9 horas e XP = 104,1 horas, com poucas horas adicionais para os times que utilizaram o XWebProcess. No

entanto, ao utilizar o Teste “t” para analisar o esforço gasto entre ambos os métodos, o autor apresenta indícios que o esforço exercido foi equivalente.

Assim que finalizada a fase de implementação do projeto, foi aplicado um questionário com objetivo de obter um *feedback* a respeito do método XWebProcess. Como resultado, foi constatado que as fases de análise de requisitos, projeto de navegação e interface gráfica, e testes, o método XWebProcess é adequado ao desenvolvimento web, satisfazendo as principais demandas dos desenvolvedores que utilizaram esse método.

O experimento realizado por Sampaio (2004) distribuiu os participantes (desenvolvedores) dos times igualmente para ambos os métodos, estratégia que foi adaptada e incorporada à presente pesquisa. No entanto, diferente de Sampaio (2004), a presente tese julga que o Teste “t” não é uma técnica interessante para ser adotada como forma de validação estatística dos dados coletados em seus experimentos, pois a precisão do Teste “t” está diretamente relacionada a quantidade de dados obtidos, ou seja, à quantidade de réplicas do experimento, necessitando da realização de mais experimentos. Diante disso, diferente de Sampaio (2004), a presente tese utilizou a técnica denominada Quadrado Latino para estruturar o segundo experimento, responsável por avaliar o método (XDTv) proposto pela presente pesquisa. Conforme apresentado no Capítulo 1, Seção 1.3, o Quadrado Latino pode ser utilizado para estruturar e avaliar os resultados dos experimentos estatisticamente, auxiliando a avaliação do desempenho de times de desenvolvimento de software, estruturados através de uma matriz *N* x *N*, cujo conteúdo é formado por *N* variáveis, de forma que estas estejam localizadas uma única vez em cada linha e coluna (SÁNCHEZ, 2011).

3.8. XPu: Uma Extensão do Método XP Visando à Usabilidade

A dissertação de Silva (2009) adiciona práticas de usabilidade denominada Praxis-u ao método XP. Este processo adaptado foi denominado XPu. Apesar dessa integração, foi mantido, a característica de agilidade do método XP.

O experimento realizado tem o objetivo de verificar a viabilidade de utilização conjunta das práticas do Praxis-u e com as práticas do método XP e

analisando uma possível sobrecarga de trabalho ao método XP devido à adição de atividades de usabilidade. Para isso, foi desenvolvida uma aplicação cujo objetivo era auxiliar usuários de uma videolocadora em terminais de autoatendimento.

Silva (2009), autor daquela pesquisa, atuou como cliente durante o desenvolvimento do projeto, guiando e verificando se o time cumpria as práticas dos métodos. A equipe de desenvolvimento era formada por dois desenvolvedores recém-graduados no curso de Sistemas de Informação, sendo um deles notoriamente mais experiente. No entanto, nenhum deles possuía conhecimento prévio de desenvolvimento ágil ou Interface Humano Computador (IHC), diante disso, foi realizado um treinamento da equipe com foco nos assuntos: desenvolvimento ágil, XP e planejamento.

A aplicação foi desenvolvida pela equipe durante uma semana em período integral. O desenvolvimento da aplicação foi dividido em duas partes, uma utilizando XP, seguida pela outra utilizando XPu. Ao fim dos trabalhos inerentes à primeira parte, o autor realizou um novo treinamento referente à segunda parte das atividades, com foco em: processos de usabilidade, o método XPu, e técnicas de IHC.

Segundo Silva (2009), o treinamento em duas etapas foi uma prevenção às interferências entre as atividades, pois caso o time conhecesse previamente as técnicas do método XPu, possivelmente haveria uma influência nas atividades realizadas através do método XP. Segundo o autor, o experimento apontou que o método XPu obteve um considerável ganho de qualidade na interface da aplicação a um custo de desenvolvimento relativamente baixo se comparado ao XP.

No entanto, o contrário não foi previsto pelo autor. É possível que a primeira experiência com o método XP tenha servido de aprendizado para facilitar as atividades realizadas com o método XPu, pois, a avaliação de dois métodos em série, permite que o aprendizado inerente ao primeiro método “*A*” (XP), auxilie o time a realizar o experimento seguinte, influenciando positivamente no desempenho do método “*B*” (XPu).

Diferente do trabalho apresentado por Silva (2009), a presente tese avaliou o conhecimento dos membros dos times, formando grupos de diferentes perfis de conhecimento. Visando minimizar a possibilidade de o aprendizado gerar anomalias nos resultados obtidos, os experimentos realizados pela presente tese foram executados paralelamente, seguindo o plano experimental do Quadrado Latino (2 x 2) (SÀNCHES, 2011), iniciando paralelamente um dos métodos investigados para desenvolver uma aplicação distinta, viabilizando assim, a comparação do desempenho do método adotado pelos diferentes times.

3.9. *Novel Hybrid Model: Integrating Scrum and XP*

Em Mushtaq e Qureshi (2012) é apresentado um modelo híbrido que se propõe a alcançar a integração dos métodos de gestão de projetos inerentes ao método Scrum, mesclados com as práticas de engenharia de software do método XP, envolvendo as fases de projeto, codificação, testes e integração de software, com o objetivo de auxiliar no desenvolvimento de software atendendo a necessidade dos *stakeholders* do projeto. O método híbrido proposto é formado por boas práticas de ambos os métodos, alinhando o produto de software aos requisitos da empresa, aumentando o desempenho dos times (MUSHTAQ; QURESHI, 2012).

Conforme apresentado anteriormente, existem diversos métodos ágeis na literatura, no entanto, Mushtaq e Qureshi (2012) optaram por escolher os métodos Scrum e XP, pois segundo o autor, Scrum é um dos métodos ágeis mais eficazes para a atividade de gestão de projetos, já o método XP apresenta ampla utilização, simplicidade, flexibilidade e capacidade de adaptação a modificações inesperadas.

O método proposto em Mushtaq e Qureshi (2012) inclui *sprint* zero, *product backlog, sprint backlog*, reuniões de planejamento de *sprint*, reunião diária Scrum, reunião de revisão da *sprint*, e retrospectiva da *sprint*. As principais práticas do XP introduzidas envolvem: *design* simples, código coletivo, programação em pares, padrão para codificação, testes contínuos, integração contínua e refatoração de código.

O processo de planejamento é a primeira etapa do ciclo de desenvolvimento de *sprint*, envolvendo a definição do projeto. As tarefas definidas no *sprint backlog* são planejadas e classificadas de acordo com suas prioridades, demandando a atenção do *scrum master*, *product owner* e o time. Ainda na fase de planejamento é estimado o esforço necessário para implementar uma tarefa do *sprint backlog* (MUSHTAQ; QURESHI, 2012).

Na fase de concepção, dois tipos de diagramas são desenvolvidos a partir de tarefas do *sprint backlog*: o diagrama de classe e o diagrama de objeto. O método também prevê a necessidade da elaboração de um projeto de testes (MUSHTAQ; QURESHI, 2012).

A atividade de codificação incorporada ao método híbrido proposto por Mushtaq e Qureshi (2012) requer padrões de codificação, compartilhamento do código, programação em pares, integração e testes contínuos e refatoração do código. Segundo o autor, a programação em par contribui para otimizar o desempenho e elevar a qualidade da aplicação. São realizadas também as atividades de desenvolvimento dirigido a testes (*test first*) e testes unitários durante o processo de desenvolvimento.

Cada característica do produto é projetado, implementado e testado individualmente dentro do ciclo de desenvolvimento da *sprint*. Um novo código é integrado continuamente no código já existente, assim que este é aprovado pelos testes. A integração contínua garante que o módulo de trabalho está disponível para usar com novos recursos (MUSHTAQ; QURESHI, 2012).

A avaliação do método híbrido proposto por Mushtaq e Qureshi (2012) foi realizada através de um experimento envolvendo um time formado por seis membros, os quais não possuíam experiência com métodos ágeis. Dessa forma, foi realizado um treinamento sobre métodos ágeis e o modelo proposto. Aos participantes foram atribuídos os papeis de programadores, testadores e um Scrum Master.

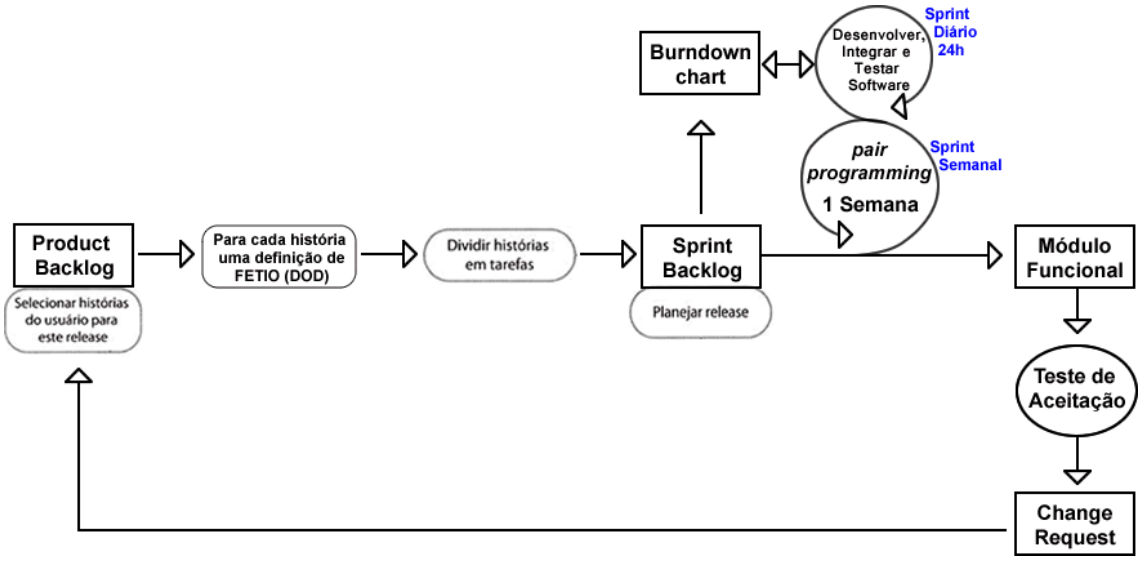
Para realizar o experimento de avaliação do método foi solicitado ao time o desenvolvimento de uma aplicação referente a uma gestão de folha de pagamento. Esta aplicação foi desenvolvida através de quatro iterações.

Entre as conclusões obtidas, Mushtaq e Qureshi (2012) ressaltam que sua proposta de método híbrido mostrou-se adequada, eliminando as carências dos métodos puros XP e Scrum. Segundo o autor, a sua proposta de método híbrido pode ser considerada uma versão melhorada do método XP e do método Scrum, pois entre outros pontos, a satisfação do cliente atingiu 85% em média.

A presente tese considera que o trabalho apresentado em Mushtaq e Qureshi (2012) não permite afirmar qualquer relação de comparação entre os métodos Scrum, XP e o método híbrido proposto por Mushtaq e Qureshi (2012). Vale destacar que aquela pesquisa não realizou qualquer experimento com o método Scrum e/ou o método XP, inviabilizando qualquer comparação entre estes três métodos. Além de não apresentar as variáveis de controle do experimento realizado, inviabilizando sua replicação.

Contudo, a presente tese considera que as características de customização e adaptação inerente aos métodos híbridos fazem destes métodos uma das alternativas em potencial para o desenvolvimento de aplicações para TVD. Diante da necessidade de avaliar o desempenho de um método híbrido formado por Scrum junto com XP, a presente tese utilizou como base a proposta definida por Mushtaq e Qureshi (2012). A proposta de Mushtaq e Qureshi (2012) foi escolhida devido ao seu desempenho positivo nos experimentos realizados por aqueles autores, bem como pela possibilidade de replicação do método. Sua proposta foi adaptada ao contexto do primeiro e do segundo experimento realizados pela presente tese, conforme apresentado na Figura 3.8.

Figura 3.8 – Exemplo de método híbrido combinando Scrum com XP



Fonte: o autor.

A Figura 3.8 apresenta as atividades do método XP intercaladas com as atividades do método Scrum, mesclando práticas de gestão de projetos com práticas de desenvolvimento de software. No entanto, inúmeras outras combinações podem ser possíveis, para atender diferentes demandas e contextos onde se pretende utilizar tal método.

Vale ressaltar que não existe um único modelo de formalizar um método híbrido. Como o próprio nome sugere, este método foi mesclado com as principais técnicas, papéis, artefatos e atividades de gestão de projetos do método Scrum, com principais técnicas, papéis, artefatos e atividades de engenharia de software do método XP, visando formar um método que atenda aos focos de ambos os métodos.

3.10. ANÁLISE MULTICRITÉRIO E A FERRAMENTA PROMETHEE II

A escolha pelo método de desenvolvimento de software ideal não deve considerar apenas a velocidade do processo de desenvolvimento. O método também precisa se mostrar adequado às atividades de especificação de requisitos, planejamento, implementação e testes de software, o que caracteriza a escolha do método como um problema de decisão multicritério.

A velocidade do desenvolvimento de software e a adequação do processo de desenvolvimento podem ser conflitantes, pois o método mais rápido pode não atender as necessidades do processo de desenvolvimento de forma adequada às peculiaridades das aplicações de TVD. Segundo Almeida (2011) “Um problema de decisão multicritério consiste numa situação, onde há pelo menos duas alternativas de ação para se escolher e esta escolha é conduzida pelo desejo de se atender a múltiplos objetivos, muitas vezes conflitantes entre si”.

O apoio multicritério de auxílio a tomada de decisão corresponde a diversos métodos e técnicas que se baseiam em preferências para realizar a análise dos critérios relevantes que resultam em informações que auxiliam a tomada de decisão (LOPES; COSTA, 2007), os critérios utilizados para a tomada de decisão podem ser tangíveis, intangíveis, objetivos, subjetivos, qualitativos ou quantitativos.

Basicamente, os métodos multicritérios são classificados como: não compensatórios ou compensatórios. Os métodos não compensatórios conhecidos como Escola Europeia / Francesa, consideram que há incompatibilidade entre as alternativas (não existe transitividade), inviabilizando a elaboração de uma relação entre as alternativas. Os métodos multicritérios compensatórios, também conhecido como Escola Americana, consideram que há relação entre as alternativas (existe transitividade de preferências), onde o decisor define as preferências, estas podem agrupar diversos critérios em um único, através de síntese. Uma das abordagens adotadas é o uso de médias ponderadas (LOPES; COSTA, 2007).

Entre os diversos métodos multicritérios existentes, um dos mais utilizados é o *Preference Method for Enrichment Evaluation* (PROMETHEE) (BRANS; VINCKE, 1985). A ferramenta PROMETHEE II implementa o método PROMETHEE permitindo realizar uma análise de sobreclassificação de valores e estabelece uma relação entre as alternativas superlativas e as demais alternativas, apontando ações cujos critérios são balanceados (equilibrados) (LOPES; COSTA, 2007).

A ferramenta PROMETHEE II foi adotada para avaliar os resultados dos experimentos da presente pesquisa devido às condições abaixo:

- soluciona problemas de ordenamento de alternativas, ou seja, classifica a melhor opção entre os métodos de desenvolvimento de software;
- permite o processamento de informações imprecisas decorrente da atribuição dos diferentes critérios;
- é vastamente utilizado como ferramenta de escolha de sistemas de informação e gestão de tecnologia da informação, em busca da solução com maior custo-benefício, ou seja, o mais adequado às diversas características e prioridades da indústria (ALMEIDA; COSTA, 2010; LOPES; COSTA, 2007); e
- pertence à família de métodos de sobreclassificação.

Baseado nos pesos de cada critério, a ferramenta PROMETHEE II estabelece o grau de sobreclassificação de “a” sobre “b” (ARAÚJO, 2012), característica fundamental para a escolha desta ferramenta, pois ela fornece uma pré-ordem completa, ou seja, uma lista em ordem completa sobre as melhores alternativas (métodos) segundo um conjunto de critério.

O PROMETHEE II gera uma classificação global (*ranking*) (ARAÚJO, 2012). Os detalhes sobre as fórmulas utilizadas pela ferramenta PROMETHEE II podem ser encontrados em (CAMPOS, 2011, p. 72).

3.11. CONSIDERAÇÕES SOBRE O CAPÍTULO

Propor um novo método de desenvolvimento implica em definir papéis, artefatos, modelo de processo e comprovar estatisticamente sua eficácia. Alguns dos trabalhos relacionados apresentaram adaptações e/ou elaboração de novos métodos de desenvolvimento de software, porém, vale destacar que realizar uma análise comparativa que prove estatisticamente que um método “*A*” atende melhor as expectativas do que um método “*B*”, não é uma tarefa trivial.

Realizar experimentos controlados é a forma mais comum de coletar e avaliar as variáveis dependentes, e ainda controlar as variáveis independentes, para cumprir esse objetivo. Os trabalhos relacionados utilizam o ambiente acadêmico, realizando experimentos controlados com alunos de graduação ou pós-graduação. Se comparado com a realização de experimento na indústria, os experimentos no ambiente acadêmico viabiliza estabelecer igualdade de condições de desenvolvimento entre dois ou mais times, como: análise de currículo, distribuição homogênea dos envolvidos, ferramentas utilizadas, especificar os requistos da aplicação que será desenvolvida, estabelecer prazo, definir o cliente e realizar treinamentos.

Estabelecer tais condições de controle na indústria de software é pouco viável, pois implica na dedicação de tempo dos profissionais e no desvio de suas atividades prioritárias, acarretando no aumento significativo de custo com mão de obra. Caso não seja possível controlar alguma dessas variáveis, o resultado obtido pode sofrer influência de elementos externos, favorecendo ou prejudicando algum dos times de desenvolvimento. Desta forma, para atender aos objetivos desta tese, a realização de experimentos controlados mostrou-se mais adequada do que a realização de estudos de caso.

De forma geral, os trabalhos relacionados apresentados destacam a dificuldade de realizar as atividades de coleta e especificação de requisitos de aplicações multimídias, indicando a necessidade de uma linguagem visual, simples e objetiva para esse fim. Apresentam ainda a necessidade do rápido desenvolvimento das aplicações de TV Digital, indicando que, entre os métodos existentes, os métodos ágeis estão mais próximos de atenderem às peculiaridades do processo de desenvolvimento de tais aplicações.

Outros trabalhos destacam a necessidade de desenvolver aplicações de TVD de forma rápida, para isso, oferecem reuso de código (MARQUES NETO, 2011), ou desenvolvimento orientado a arquétipos (FERNANDES; TAVARES; PÔRTO, 2011). Tais trabalhos apresentam contribuições que visam auxiliar a implementação das aplicações, porém, não especificam a integração com um método de desenvolvimento ou seu modelo de processo. Dessa forma, tais

trabalhos relacionados podem ser considerados complementares a presente tese.

Alguns trabalhos relacionados destacam que métodos ágeis híbridos, formados por Scrum e XP, são adequados ao desenvolvimento de aplicações para TVD (LULA; GUIMARÃES; TAVARES, 2011), (VEIGA, 2006), (VEIGA; TAVARES, 2006, 2007). Porém, diferente da presente tese, tais trabalhos estão voltados para a confecção do conteúdo audiovisual, com foco na integração da equipe multidisciplinar, integrando os profissionais de TV, com os profissionais responsáveis pelo desenvolvimento de software.

Conforme apresentado anteriormente, os trabalhos relacionados auxiliaram na estruturação dos experimentos, apresentados nos capítulos 4 e 6, bem como na especificação do método XDTv, proposto pela presente tese, apresentado no Capítulo 5. Os trabalhos apresentados neste capítulo estão relacionados com a presente tese de acordo com o Quadro 3.1.

Quadro 3.1 – Síntese sobre a utilização de características dos trabalhos relacionados

TÉCNICAS ADOTADAS EM TRABALHOS RELACIONADOS INCORPORADAS A PRESENTE TESE	SEÇÃO DO TRABALHO										TESE XDTV
	3.1	3.2	3.3	3.4	3.5	3.6	3.7	3.8	3.9	3.10	
Apresenta a demanda por modelagem das aplicações (TVD)	X			X	X						X
Apresenta a demanda por velocidade no desenvolvimento de aplicações de TVD.	X	X	X	X	X						X
Utiliza *storybord* no desenvolvimento de aplicações ou conteúdo multimídia para TVD.	X	X		X	X						X
Apresenta a dificuldade representar objetos multimídia sua sincronização.	X		X								X
Apresenta a demanda por recursos capazes de realizar a coleta de requisitos multimídia.	X		X	X	X						X
Aponta os métodos ágeis como uma provável opção para desenvolver aplicações de TVD.	X		X	X	X						X
Aponta a importância da comunicação do time de desenvolvimento de aplicações de TVD.		X		X	X						X
Apresenta peculiaridades das aplicações de TVD.	X	X	X	X	X						X
Utilizou estudantes em seus experimentos controlados para avaliar sua proposta.				X		X	X		X		X
Utilizou profissionais para avaliar sua proposta.		X		X	X	X		X			X
Realizou experimentos e apresentou métricas sobre métodos ágeis para desenvolvimento de software em geral.						X	X	X			X
Técnicas para controle de experimentos: treinamento, análise de código fonte e questionários.						X	X				X
Propõe e avalia um método ágil customizado para o desenvolvimento de software.				X	X		X	X			X
Realiza a distribuição do time de desenvolvimento com base no perfil dos participantes.								X			X
Apresenta um método híbrido com características inerentes dos métodos Scrum e XP.				X	X				X		X
Realiza uma análise estatística (qualitativa e quantitativa) através do método multicritério de apoio à tomada de decisão e a ferramenta PROMETHEE II.										X	X
Especifica um método ágil para o desenvolvimento de aplicações para TVD.											X
Apresenta métricas referentes ao tempo e a adequação dos métodos de desenvolvimento investigados.											X

Fonte: o autor.

CAPÍTULO

4.

4. INVESTIGAÇÃO SOBRE SCRUM, XP E HÍBRIDO

"A língua revela o conteúdo do coração."
Chico Xavier

O presente capítulo apresenta uma investigação sobre os métodos Scrum, XP e um método híbrido formado por Scrum junto com XP (Scrum/XP). Conforme citado no Capítulo 1, este experimento foi baseado na proposta de experimentação definida em Travassos (2002), que estabelece as seguintes etapas: Definição, Planejamento, Instrumentação e Execução do experimento.

O Quadro 4.1 apresenta uma síntese com as principais características (Foco, Papéis, Artefatos e Atividades) dos métodos investigados neste experimento: Scrum, XP e Híbrido (XP/Scrum).

Considerando os métodos Scrum, XP e o método híbrido (Scrum/XP), com base na análise do referencial teórico e dos trabalhos relacionados, este experimento partiu das seguintes questões de pesquisa a serem investigadas:

- **Q1**. Qual dos métodos investigados é mais adequado ao desenvolvimento de aplicações de TVD?
- **Q2**. Qual dos métodos investigados possibilita a implementação de aplicações de TVD em menos tempo?

Para guiar a investigação das questões de pesquisa levantadas no primeiro experimento, foram definidas as seguintes hipóteses sobre os métodos investigados:

- Hipótese nula (**H0**): os métodos avaliados não apresentam pontos de melhoria e são igualmente adequados para o desenvolvimento de aplicações para TVD.
- Hipótese nula (**H1**): o tempo dedicado ao desenvolvimento de aplicações é igual para os três métodos investigados.
- Hipótese alternativa (**H2**): é possível realizar melhorias nos métodos investigados para melhor atender às peculiaridades das aplicações de TVD.
- Hipótese alternativa (**H3**): entre os métodos investigados o método Scrum é o mais adequado às peculiaridades de TVD.
- Hipótese alternativa (**H4**): entre os métodos investigados o método XP é o mais adequado às peculiaridades de TVD.
- Hipótese alternativa (**H5**): entre os métodos investigados o método híbrido (Scrum/XP) é o mais adequado às peculiaridades de TVD.

Quadro 4.1 – Comparação funcional entre os métodos Scrum, XP e Híbrido

	Scrum	XP	Híbrido (Scrum/XP)
FOCO	Gestão de projeto	Implementação	Gestão de projeto
	Agilidade e Rapidez	Testes de software	Implementação
	Escopo: projetos em geral	Agilidade e Rapidez	Agilidade e Rapidez
		Escopo: desenvolvimento de software	Testes de software
			Escopo: desenvolvimento de software
PAPÉIS	Product Owner	Cliente/Usuário/Executivo	Cliente/Usuário/Executivo
	Scrum Master	Gerente de projeto	Scrum Master
		Gerente de produto	Tracker
		Arquitetos	Programador / Testador
		Projetista	Gerente produto/Requisitos
		Programador / Testador	
		Coach	
		Tracker	
		Analista de negócio	
		Redatores	
		Recursos humanos	
ARTEFATOS	Burn down chart	Histórias	Burn down chart
	Product Backlog	Tarefas	Product Backlog
	Sprint Backlog	Módulos funcionais	Sprint Backlog
			Histórias
			Tarefas
			Módulos funcionais
ATIVIDADES	Iteração / Sprint	Iteração semanal/trimestral	Iteração / Sprint
	Sprint Planning	Colaboração	Sprint Daily
	Sprint Daily	Stand up meeting	Stand up meeting
	Stand up meeting	Programação em par	Sprint Retrospective
	Sprint Review	Envolvimento real com o cliente	Definition of Done
	Sprint Retrospective	Código compartilhado	Colaboração
	Definition of Done	Repositório de código único	Programação em par
		Implementação contínua	Envolvimento real do cliente
		Padronizar código fonte	Código compartilhado
		Test-first	Repositório de código único
		Manter apenas código e testes	Implementação contínua
			Padronizar código fonte
			Teste de Aceitação
			Evoluir apenas código e teste

Fonte: o autor.

Visando realizar uma avaliação sobre tais métodos, o presente capítulo está subdivido nas seguintes seções: 4.1) objetivos; 4.2) planejamento; 4.3) operação; 4.4) resultados do experimento realizado; 4.5) *survey* realizado com desenvolvedores experientes em TVD; 4.6) discussão dos resultados; 4.7) análise dos resultados através do método PROMETHEE; e 4.8) conclusão.

Dessa forma, contempla-se as Fases 2, 3 e 4 do desenho metodológico, apresentado na Seção 1.3 do Capítulo 1.

4.1. Definição do objetivo global do experimento

O objetivo desse primeiro experimento foi mensurar a velocidade (tempo) e a adequação da adoção dos métodos Scrum, XP e o método híbrido formado por Scrum junto com XP (Scrum/XP) baseado em Mushtaq e Qureshi (2012), sobre o desenvolvimento de aplicações para TVD, visando coletar possíveis pontos de melhoria para atender às peculiaridades das aplicações de TVD. O objetivo global do presente experimento é apresentado no Quadro 4.2, onde são descritos os atributos e valores esperados.

QUADRO 4.2 – INVESTIGAÇÃO DE SCRUM, XP E HÍBRIDO

ATRIBUTOS	VALORES PARA O EXPERIMENTO
Objetos investigados	Os métodos de desenvolvimento: Scrum, XP e Híbrido (Scrum/XP).
Propósito	Avaliar o desempenho dos métodos investigados em busca pontos de melhoria para o processo de desenvolvimento de aplicações para TVD.
Em respeito a	Adequação e ao tempo dedicado ao processo de desenvolvimento.
Perspectiva	Desenvolvedores de aplicações para TVD.
Contexto	Alunos matriculados na disciplina Tópicos Avançados em Engenharia de Software oferecida no décimo período do curso de Engenharia da Computação (UNIVASF).

Fonte: o autor.

Conforme apresentado no Capítulo 1, a presente tese considera que um método adequado ao desenvolvimento de aplicações para TVD deve contemplar as atividades de Planejamento, Especificação, Desenvolvimento e Testes, com o objetivo de auxiliar no trato de suas peculiaridades por meio de artefatos customizados e um ciclo de vida que permita um rápido desenvolvimento.

4.2. PLANEJAMENTO

Visando investigar as questões de pesquisa e hipóteses apresentadas na Seção 1.3 (Procedimentos Metodológicos), este experimento adota diferentes perspectivas para realizar a coleta dos resultados, foram utilizados os seguintes instrumentos: questionário formado por questões objetivas e

descritivas, disponível no Apêndice D, seguido pela entrevista semiestruturada individual e posteriormente entrevista em grupo (FLICK, 2009).

Para cada questão objetiva são oferecidas cinco alternativas de resposta, conforme sugere a escala de Rensis Likert em seu trabalho sobre técnicas para mensurar atitudes, apresentado originalmente em 1932 (BERTRAM, 2009). As questões descritivas são analisadas utilizando a técnica de palavra-chave, condensando os resultados similares levantados. Os detalhes da coleta de resultados serão descritos nas próximas seções.

Visando minimizar a desigualdade entre os times, foram tomados os seguintes cuidados para a realização do experimento:

1. Os times de desenvolvimento foram formados por alunos do décimo período do curso de Engenharia da Computação (UNIVASF);
2. Com o objetivo de formar times de desenvolvimento com conhecimento homogêneo, foram levantadas as experiências pessoal e acadêmica de cada membro. Os principais pontos considerados foram: TV Digital, programação, desenvolvimento de software e métodos ágeis;
3. Os perfis dos alunos que participaram do experimento foram levantados por meio de:
    3.1. Análise das disciplinas previamente cursadas por cada aluno;
    3.2. Aplicação de um questionário, disponível no Apêndice A;
    3.3. Entrevista individual; e
    3.4. Uma planilha com a pontuação para cada participante.
4. Apesar da pontuação que media o conhecimento dos participantes não apresentar diferenças significativas, estes foram distribuídos de acordo com sutis variações levantadas na planilha de análise dos participantes, buscando assim, formar times cujos membros apresentam perfis balanceados de acordo com a pontuação obtida; e
5. A definição do método adotado por cada time ocorreu através de sorteio, seguindo o princípio da aleatoriedade.

Tendo em vista que este experimento foi realizado com alunos de graduação, este trabalho se preocupou em minimizar a possibilidade da nota da disciplina influenciar (evitando a supervalorização) nos dados a serem coletados. Desta forma, além de alertar aos alunos que os dados coletados não interferem na nota, tais notas foram entregues antes do início da coleta dos dados. O Quadro 4.3 sintetiza as variáveis investigadas nesse experimento.

QUADRO 4.3 – VARIÁVEIS INDEPENDENTES: INVESTIGAÇÃO SOBRE SCRUM, XP E HÍBRIDO

Variáveis Independentes	Valores
Quantidade total de desenvolvedores	9 participantes.
Quantidade de times	Três times (A, B e C).
Quantidade de desenvolvedores	3 participantes para cada time; Distribuição balanceada através de currículo.
Modificação dos times	Não houve modificação. Cada time foi composto pelos mesmos membros durante todo o experimento, desenvolvendo as duas aplicações.
A formação dos times ocorreu em duas etapas	1) Classificação dos desenvolvedores em três diferentes níveis de conhecimento através da análise de perfil; 2) Através de sorteio, elencar um membro de cada grupo de conhecimento para cada time.
Conhecimento dos times sobre TVD	Inexperientes. Foram dedicadas 30 horas de treinamento.
Métodos de desenvolvimento adotados	Scrum; XP; e Híbrido (Scrum+XP).
Método adotado pelo time	Aleatório (sorteio).
Aplicação desenvolvida	Arquibancada virtual de futebol.
Requisitos funcionais	Idênticos para todos os times.
Requisitos não funcionais	Para reduzir a possibilidade de cópia de código fonte, foram diferentes para cada time. Exemplo: posição, tamanho, tempo de surgimento e finalização das mídias.
Implementação dos requisitos	Foi implementado um requisito de cada vez, em série.
Realização das atividades	As atividades abaixo foram realizadas em equipe, com a participação de todos os membros. a) Reunião com o cliente; b) Coleta e especificação de requisitos; e c) Planejamento.
Prazo de entrega para cada aplicação	3 semanas.
Cliente	O autor da tese.
Reuniões com o cliente	Tempo fixo e inflexível para todos os times: 6 reuniões de 30 minutos.
Quantidade de iterações (ciclo)	3 iterações.
Duração de cada iteração	Uma semana.
Ambiente de desenvolvimento	Eclipse Helios; Plugins: RSE, Lua, NCL; Maquina virtual VMware Player; Ginga Fedora.

Fonte: o autor.

Para evitar interferência nos resultados obtidos, foi realizado um forte controle sobre as variáveis externas. Entre elas, se destaca o contato entre o cliente e o time de desenvolvimento. Conforme apresentado em CHAOS (2010), a presença do cliente é um fator relevante, influenciando diretamente no sucesso do projeto. Quando esta relação é falha, há alto índice de fracasso, conforme apresentado no Capítulo 1. Dessa forma, o presente trabalho fixou igualmente o tempo de participação do cliente com todos os times.

Sabendo que futebol é um assunto de interesse de grande parte dos envolvidos, visando estimular os times, a aplicação solicitada para a implementação correspondia à simulação de uma arquibancada virtual de um campeonato de futebol. O objetivo desta aplicação é oferecer aos usuários/telespectadores de torneios esportivos informações adicionais relacionadas ao campeonato, visando contribuir com seu entretenimento. Resumidamente, a aplicação solicitada possuía os seguintes Requisitos Funcionais (RF):

RF-01. Possibilidade de visualizar o jogo por 4 ângulos diferentes;

RF-02. Visualização em tela cheia;

RF-03. Visualização simultânea;

RF-04. Pausar e retomar vídeo;

RF-05. Visualizar a classificação do campeonato;

RF-06. Adquirir as seguintes informações sobre as equipes: escalação, placar, técnico e quantidade de cartões amarelos e vermelhos atribuídos para cada jogador;

RF-07. Veiculação de publicidade;

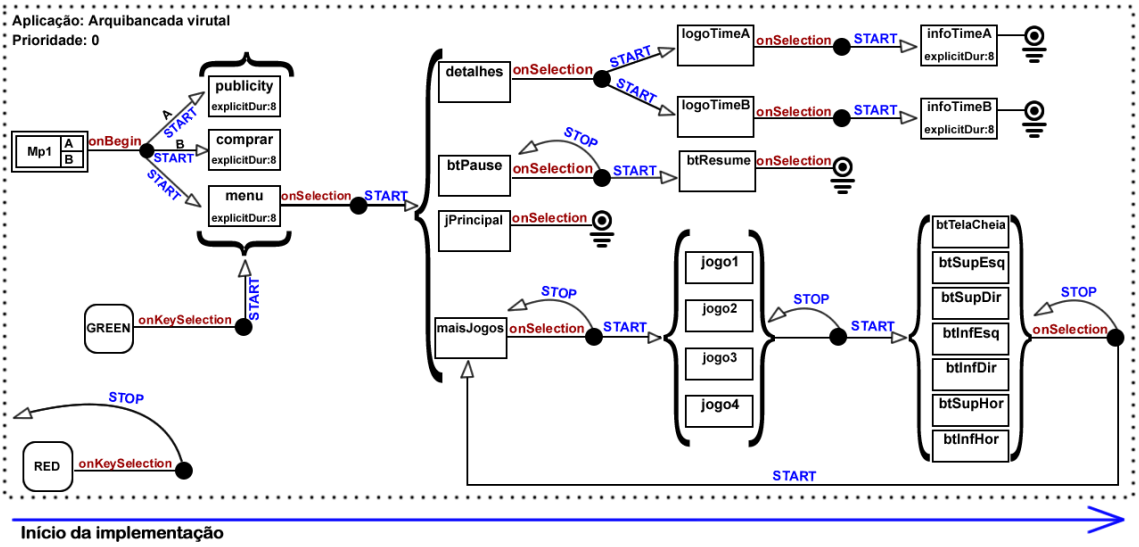
RF-08. Venda de produtos.

Os três times receberam um documento, disponível no Apêndice B, que descrevia superficialmente os requisitos supracitados. A superficialidade do documento foi necessária para que os times exercitassem as atividades de Elicitação e Especificação de requisitos, possibilitando a coleta do desempenho dos métodos sobre essas atividades.

Certamente a implementação dos requisitos solicitados poderia ser realizada de diversas maneiras. Visando maior compreensão, a **Figura 4.1**

apresenta uma das possibilidades de implementação da aplicação solicitada. Este diagrama corresponde ao Modelo de Fluxo e Sincronismo de Mídia (MFSM), artefato proposto pela presente tese como um modelo de documentação, publicado anteriormente (GODOY NETO; FERRAZ, 2012, 2013a, 2013b). O metamodelo, a legenda e os objetivos desse modelo de documentação são apresentados com mais detalhes no Capítulo 6.

Figura 4.1 – Representação da aplicação desenvolvida no experimento



Fonte: o autor.

Vale ressaltar que a **Figura 4.1** foi desenvolvida após os times finalizarem a aplicação, dessa forma, nenhum dos times teve acesso a este diagrama. Os participantes receberam apenas um documento básico, disponível no Apêndice B. A **Figura 4.1** pretende apresentar mais detalhadamente o funcionamento da aplicação solicitada, visando maior compreensão sobre o seu grau de complexidade.

O elemento (`Mp1`) representa a porta (`port`), mídia principal da aplicação e o mapeamento de suas âncoras (`A` e `B`). Ao iniciar a âncora `A`, é exibida a mídia de publicidade. Ao iniciar a âncora `B`, é exibida a mídia de *t-commerce*. Ambas as âncoras são exibidas durante 8 segundos (`explicitDur: 8`), tornando-as disponíveis para a interação do usuário durante esse período; após esse tempo, tais mídias são finalizadas.

O botão `menu`, quando selecionado, oferece os seguintes recursos: pausar jogo (`btPause`); finalizar a aplicação e retornar ao jogo principal (`jPrincipal`); apresentar detalhes dos times (`detalhes`), permitindo ao usuário escolher qual dos times quer mais informações.

A opção `maisJogos` possibilita alterar o fluxo de vídeo exibido, permitindo ao usuário escolher outros jogos em andamento. Ao selecioná-la, são apresentados ao usuário quatro imagens de botões que correspondem às opções de jogos (`jogo1`, `jogo2`, `jogo3` e `jogo4`). Ao escolher um jogo, o usuário pode optar pelo local da tela onde este será exibido, com as opções: tela cheia, cantos superiores esquerdo ou direito, cantos inferiores esquerdo ou direito, ou ainda, é possível optar pela tela dividida em duas, horizontal e vertical. Após a definição do local, o usuário é redirecionado de volta para a opção `maisJogos`, onde poderá escolher o segundo, terceiro e até um quarto jogo para assistir paralelamente.

O evento "`onKeySelection`" habilita que, a qualquer momento, dentro do contexto da aplicação "Arquibancada Virtual", ao pressionar o botão interativo vermelho (`RED`) do controle remoto, todos as mídias ativas devem ser paradas (`stop`), finalizando a aplicação.

Conforme apresentado anteriormente, este experimento pretende analisar o processo de desenvolvimento de forma ampla. Para isso, foi mensuradas das horas trabalhadas individualmente por cada participante, durante as atividades de Planejamento e Implementação da aplicação, definindo assim as variáveis dependentes desse experimento, conforme mostra o Quadro 4.4.

QUADRO 4.4 – VARIÁVEIS DEPENDENTES: INVESTIGAÇÃO SOBRE SCRUM, XP E HÍBRIDO

Variáveis Dependentes	Atividades do Processo de Software
Tempo dedicado	Planejamento e Implementação.
Pontos de melhoria/adequação dos métodos	Requisitos, Planejamento e Implementação.
Desgaste / cansaço do time	Planejamento e Implementação.

Fonte: o autor.

O tempo dedicado à execução do Experimento 1 foi sintetizado em ordem cronológica, conforme apresentado no Quadro 4.5.

QUADRO 4.5 – SÍNTESE DO TEMPO DEDICADO ÀS ATIVIDADES REALIZADAS

Atividade	Responsável	Tempo dedicado
Revisão e nivelamento: métodos Scrum, XP.	O autor	04:00 horas
Treinamento: NCL, Lua, Ambiente de TVD.	O autor	30:00 horas
Elaboração do questionário para coleta do nível de experiência.	O autor	12:00 horas
Preenchimento do questionário de experiência.	Desenvolvedores	00:15 minutos
Nível de experiência: Análise currículo.	O autor	06:00 horas
Nível de experiência: Entrevista individual.	O autor	03:00 horas
Analise do perfil dos desenvolvedores e distribuição entre times.	O autor	04:00 horas
Sorteio do método para cada time.	O autor / desenvolvedores	00:15 minutos.
Elaboração do formulário para coleta de horas trabalhadas.	O autor	08:00 horas
Planejamento da aplicação a ser desenvolvida.	O autor	12:00 horas
Desenvolvimento da aplicação.	Desenvolvedores	3 semanas
Preenchimento do formulário de horas trabalhadas.	Desenvolvedores	
Elaboração do questionário sobre o desenvolvimento.	O autor	06:00 horas
Preenchimento do questionário sobre o desenvolvimento.	Desenvolvedores	00:20 minutos
Entrevista semiestruturada.	O autor / Desenvolvedores	02:30 horas
Análise dos resultados obtidos: formulário horas trabalhadas, questionário sobre o desenvolvimento, entrevista.	O autor	24:00 horas
Elaboração do *Survey.*	O autor	12:00 horas
Preenchimento do *Survey.*	Desenvolvedores profissionais de TVD	00:10 minutos
Análise do *Survey*.	O autor	8 horas
Planejamento e definição do método XDTv.	O autor	8 meses

Fonte: o autor.

A execução das atividades planejadas para este experimento é apresentada na próxima seção.

4.3. OPERAÇÃO E EXECUÇÃO

Conforme planejado, foram formados três (3) times de desenvolvimento, compostos por três (3) membros cada, igualmente distribuídos de acordo com seu grau de conhecimento, conforme explicado anteriormente. Cada time foi orientado a desenvolver a aplicação proposta, adotando por sorteio, um dos seguintes métodos: Scrum, XP ou híbrido (Scrum juntamente com XP).

Devido à inexperiência dos membros dos times no desenvolvimento de aplicações para TV Digital, foram dedicadas 30 horas de aula, durante 2 meses, voltadas exclusivamente para o seu treinamento. Foram abordadas as linguagens NCL e Lua, a IDE Eclipse[12] versão Helios, e a configuração do ambiente de desenvolvimento e simulação de aplicações através da maquina virtual VMware Player[13] emulando o Ginga Fedora[14].

Vale salientar que nos experimentos realizados durante essa pesquisa, não foi utilizada qualquer ferramenta assistente responsável por auxiliar a implementação através de objetos gráficos visuais (interface gráfica), todas as atividades desenvolvidas foram implementadas através de codificação NCL e Lua diretamente na IDE Eclipse. Além das 30 horas com aulas presenciais, foram realizados diversos exercícios de fixação como atividades extraclasse.

A fim de viabilizar a comparação entre os resultados obtidos, as seguintes condições foram idênticas para todos os times: a aplicação solicitada, os requisitos, o cliente e o prazo de entrega. Foi estabelecido o prazo de 3 semanas para o desenvolvimento. Cada time realizou duas reuniões semanais com o cliente com o mesmo tempo de duração (30 minutos), totalizando três horas de reuniões com o cliente para cada time. Tais reuniões foram conduzidas através do Scrum Master ou pelos líderes dos projetos XP.

Para verificar se os times cumpriam as metodologias, foi solicitado a cada time um relatório técnico semanal descrevendo o ocorrido em cada fase da metodologia utilizada, bem como os papéis e atividades de cada membro. O cliente era representado pelo professor da disciplina, autor da presente tese, que nesse momento, comportava-se como um cliente típico, porém, observando se os grupos seguiam, de fato, os métodos.

Ao fim de um ciclo, cada membro do time entregava um formulário (apresentado no Apêndice C) com as métricas de desenvolvimento, horas trabalhadas e as atividades desenvolvidas.

[12] http://www.eclipse.org
[13] http://www.vmware.com
[14] http://www.ginga.org.br

Para verificar as dificuldades enfrentadas e os possíveis pontos de melhoria sobre o método utilizado, foi elaborado um questionário composto por perguntas objetivas e descritivas, disponível no Apêndice D, respondido individualmente pelos membros de cada equipe.

Após a análise dos questionários individuais, foi realizada uma entrevista em grupo com cada time, na qual foram debatidas algumas respostas que destoaram, levemente, entre os membros de cada time, visando obter perspectivas variadas e os fatores que justificassem as respostas. Assim que terminada a entrevista em grupo, o time se reuniu novamente, sem a presença do cliente, e respondeu ao mesmo questionário, dessa vez, em equipe, e para tal foi necessário buscar um consenso. Os resultados desse experimento são exibidos na próxima seção.

4.4. RESULTADOS PARCIAIS

Os três times desenvolveram a aplicação solicitada de forma independente, em paralelo com os demais times. Os três times conseguiram implementar todas as funcionalidades solicitadas pelo experimento. Parte destes resultados foi publicada em Godoy Neto e Ferraz (2011).

Após a finalização do Experimento 1, os três times foram convidados a escrever um artigo sobre a aplicação desenvolvida, porém, apenas dois artigos foram escritos e apenas um foi publicado. O time que adotou o método híbrido (Scrum/XP) publicou a aplicação desenvolvida como artigo completo (*full paper*) no Workshop Tecnologias da Informação e Comunicação nos Grandes Eventos Esportivos (WTICEE) Godoy Neto; Amorim; Ferraz (2012). Esta aplicação é *open source,* sob licença *General Public License* (GNU GPL), seu código fonte está disponível no endereço: *https://sourceforge.net/p/gtvd/*.

Resultados Quantitativos

Os valores absolutos obtidos através do formulário utilizado para coleta das horas trabalhadas são apresentados no Quadro 4.6.

Quadro 4.6 – Resultados Quantitativos - Horas trabalhadas por time

Times (métodos)	Horas dedicadas			
	Reuniões de planejamento		Implementação (NCL/Lua) (TI)	Soma das Horas
	Time (TP)	Cliente		
Scrum	16:00	03:00	59:00	78:00
XP	15:30		46:09	64:39
Híbrido (Scrum/XP)	17:30		38:29	58:59

Fonte: o autor.

Os métodos (XP e Híbrido) reduziram, em média, 21% (17 horas) da carga horária necessária para finalizar os projetos.

Resultados Qualitativos

Abaixo serão exibidos os resultados do questionário preenchido em consenso por cada time, separado em questões objetivas e questões descritivas. Vale ressaltar que os resultados das Questões 8 e 9 não foram exibidos por serem inconclusivos.

Questões Objetivas

A **Pergunta 1 (P1)** solicitou a classificação do grau de desgaste do desenvolvedor, ou seja, o esforço dedicado para realizar as atividades solicitadas.

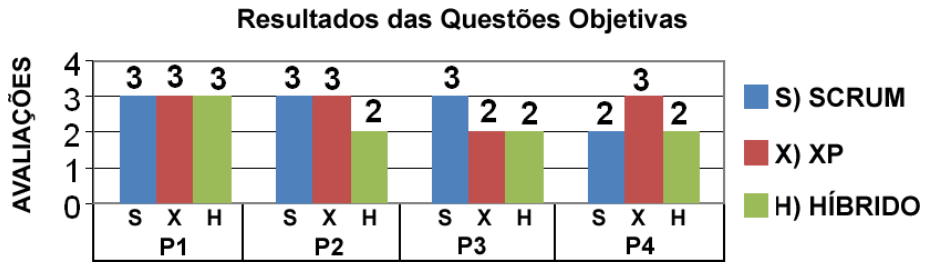
A **Pergunta 2 (P2)** buscou o quão o método utilizado auxiliou no processo de desenvolvimento.

A **Pergunta 3 (P3)** solicitou que seja definido o grau de complexidade da aplicação.

A **Pergunta 4 (P4)** indagou sobre a complexidade de codificar suas funcionalidades.

Para cada questão objetiva são oferecidas cinco opções de resposta que seguem a escala de Likert, que corresponde aos valores de zero (0) até quatro (4), sendo respectivamente a menor e a maior nota. O eixo horizontal representa as questões objetivas, o eixo vertical o valor que cada time atribuiu às repostas. Os resultados das perguntas objetivas P1, P2, P3 e P4 são apresentados no **Gráfico 4.1**.

Gráfico 4.1 – Resultado das questões objetivas



Fonte: o autor.

As respostas sobre as Perguntas 5, 6 e 7 são sintetizadas a seguir:

Pergunta 5. O método utilizado foi adequado ao desenvolvimento da aplicação solicitada?

Os três grupos foram unânimes confirmando tal adequação mediante pequenas modificações no método.

Pergunta 6. Caso trabalhasse em uma fábrica de aplicações para TVD e pudesse escolher um método de desenvolvimento, qual seria sua decisão?

Os times XP e híbrido (Scrum/XP) confirmaram a preferência pelo método utilizado pelo seu time. Já o time Scrum considera que a característica de programação em pares, se adicionada a Scrum, pode auxiliar no processo de desenvolvimento.

A **Pergunta 7** (descritiva) solicitou que fossem relatados três pontos fortes e fracos sobre o método adotado. As respostas foram sintetizadas utilizando a técnica de palavras chave (FLICK, 2009), e estão disponíveis no Quadro 4.7.

Quadro 4.7 – Questão subjetiva - pontos fortes e fracos

Times (métodos)	Pontos Fortes	Pontos Fracos
Scrum	Entrega rápida de módulos funcionais torna o *feedback* eficiente; A alteração de requisitos não implica em refatorar módulos já entregues.	Manter os horários das reuniões.
XP	Programação em par une a equipe; Reuniões diárias auxiliam o planejamento.	Dificuldade em realizar o desenvolvimento orientado a testes. Integrar os requisitos ao longo do desenvolvimento.
Híbrido (Scrum/XP)	Liberdade, comprometimento do time, simplicidade de implementação.	Excesso de atividades. Desenvolvimento cansativo.

Fonte: o autor.

Entrevista semiestruturada individual

A entrevista individual foi aplicada com o intuito de refinar o questionário aplicado. Seu objetivo era esclarecer os seguintes questionamentos:

- Houve dificuldade do líder durante o gerenciamento do projeto?
- Qual o motivo do elevado desgaste (esforço dedicado às atividades), apresentado no Gráfico 4.1, tendo em vista que dois (2) times avaliaram a complexidade da aplicação e a dificuldade de sua implementação inferior ao nível do desgaste?

As respostas à primeira questão foram similares para os líderes dos três times. Foi relatado que cada membro cumpriu o seu papel, reforçando a ideia de que times pequenos, três (3) membros, conseguem ser auto-gerenciáveis. Entretanto, o líder do time que utilizou o método híbrido salientou que a programação em pares reforça a produtividade e o compromisso com horários.

As respostas referentes à segunda questão apontaram um elevado esforço repetitivo para a codificação da linguagem NCL e para a realização do sincronismo de mídias.

Entrevista semiestruturada em grupo

A entrevista em grupo, aplicada separadamente para cada time, teve o intuito de refinar o questionário e a entrevista individual. O foco era levantar

sugestões de melhoria diante das dificuldades enfrentadas, o resultado é apresentado no Quadro 4.8.

QUADRO 4.8 – RESULTADO DA ENTREVISTA EM GRUPO

Times	Motivos do desgaste / esforço dedicado para realizar as atividades
Scrum, XP e Híbrido	A partir da segunda iteração, o aumento significativo do código dificulta a compreensão do fluxo e sincronismo de mídia, tornando difícil a implementação de novas funcionalidades.
XP	Necessidade de encontro presencial para programação em pares; Viabilizar reuniões síncronas a distância pode diminuir o desgaste.
Híbrido (Scrum/XP)	A associação de tais metodologias aumenta significativamente os processos a serem seguidos, o que demanda muita atenção para que todas as características de cada metodologia sejam realizadas. As reuniões diárias presenciais precisam que os membros dos times tenham suas agendas sincronizadas.

Fonte: o autor.

No entanto, tais dados ainda precisavam ser comparados com a opinião de desenvolvedores experientes em aplicações para TVD. Diante isso, foi planejado e realizado o *survey* apresentado na próxima seção.

4.5. *SURVEY* SOBRE DESENVOLVIMENTO DE APLICAÇÕES PARA TVD

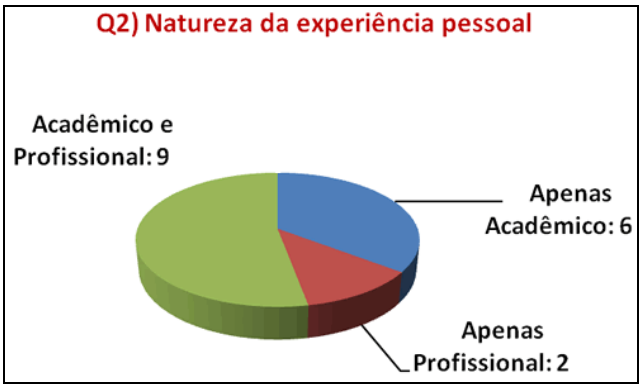
Este *survey* visou conhecer a perspectiva da industria, com o objetivo de levantar a opinião de profissionais experientes no desenvolvimento de aplicações para TVD, de modo a complementar os resultados oriundos do primeiro experimento, concentrando-se nos seguintes pontos:

- Quais os métodos, boas práticas e adaptações utilizadas;
- Qual o método mais adequado;
- Quais as práticas ágeis adequadas;
- Quais documentações são necessárias; e
- Qual o método mais adequado Scrum, XP ou Híbrido.

O *survey* foi colhido através de um questionário, disponível no Apêndice E, disponibilizado online pela ferramenta Google Forms. Além das informações pessoais sobre os desenvolvedores, era composto por 9 questões objetivas e descritivas que foram sintetizadas através da técnica de palavra chave (FLICK, 2009).

Responderam ao questionário 17 desenvolvedores. Como já era esperada, a área de atuação dos participantes foi Ciência da Computação ou áreas a fins. A natureza dessa experiência foi tanto profissional, como acadêmica, porém, grande parte está relacionada a atividades profissionais como mostra o **Gráfico 4.2**.

Gráfico 4.2 – Natureza da experiência com o desenvolvimento de aplicações para TVD



Fonte: o autor.

A média do tempo de experiência foi de 2 anos e 8 meses, considerando aqueles que possuem mais de 4 anos de experiência, com apenas 4,5 anos. No **Gráfico 4.3** é exibido o tempo de experiência sobre o assunto.

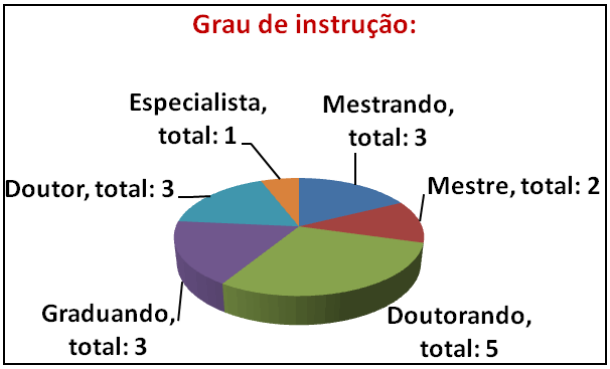
Gráfico 4.3 – Tempo de experiência no desenvolvimento de aplicações de TVD

Fonte: o autor.

Apenas três participantes estavam cursando a graduação, porém, estes indicaram alto grau de conhecimento e experiência no desenvolvimento de aplicações para TVD. Os demais participantes possuíam elevado grau de

instrução. O Gráfico 4.4 apresenta apenas o maior nível de instrução de cada participante (não acumula títulos).

Gráfico 4.4 – Grau de instrução dos participantes

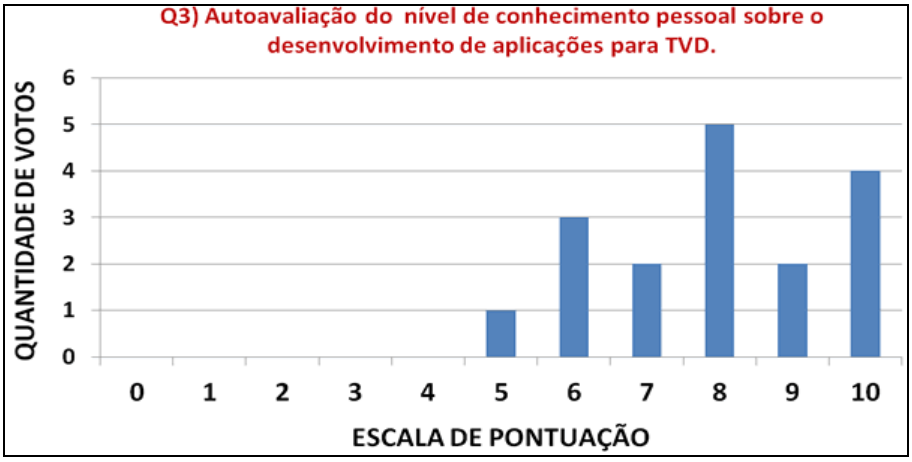


Fonte: o autor.

As questões objetivas (Q3, Q8, Q9 e Q10) apresentam as suas alternativas de respostas em uma escala que corresponde de avaliação que variou de: extremamente inadequado “0”, até extremamente adequado “10” (pontos), dessa forma uma pontuação neutra atingiria nota, sendo a pontuação neutra 5.

O Gráfico 4.5 apresenta, por meio de uma autoavaliação, o nível de conhecimento dos participantes sobre o desenvolvimento de aplicações para TVD. A média de conhecimento obtida entre todos os participantes foi 7,94 pontos.

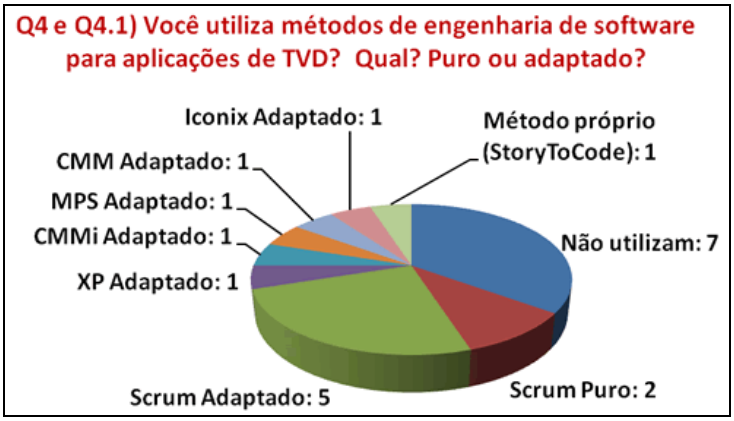
Gráfico 4.5 – Q3: autoavaliação do nível de conhecimento sobre o desenvolvimento



Fonte: o autor.

As Questões 4 e 4.1 foram sintetizadas no **Gráfico 4.6**, onde é possível observar que diversos métodos são utilizados para o desenvolvimento de aplicações para TVD. Porém, (41%) 7 usuários não utilizam qualquer método, e aquele que aparece com maior frequência é o método Scrum. A soma dos métodos ultrapassa a quantidade de desenvolvedores, pois, alguns participantes utilizam mais de um método.

Gráfico 4.6 – Métodos utilizados pelos participantes



Fonte: o autor.

O **Gráfico 4.7** apresenta as práticas relacionadas à engenharia de software que, segundo os participantes, são essenciais ao desenvolvimento de aplicações para TVD.

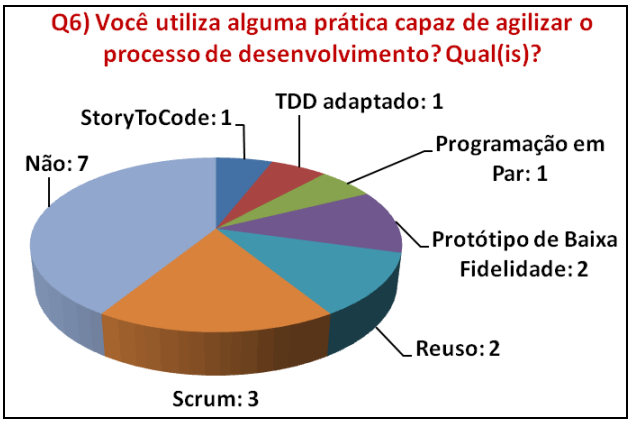
Gráfico 4.7 – Práticas de engenharia de software essenciais para TVD

Fonte: o autor.

O **Gráfico 4.8** apresenta as técnicas utilizadas pelos desenvolvedores com o objetivo de agilizar o processo de desenvolvimento. Sete (7)

entrevistados, ou seja, 41% não se preocupam com agilidade. Apesar de não existir um consenso, os demais, procuram algum meio de viabilizar essa característica.

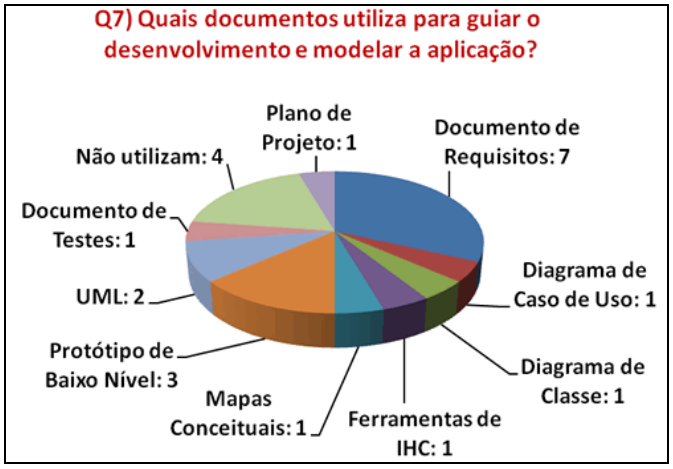
Gráfico 4.8 – Prática utilizada para agilizar o desenvolvimento



Fonte: o autor.

O **Gráfico 4.9** apresenta os documentos conhecidos na Engenharia de Software utilizados para guiar o processo de desenvolvimento e modelar a aplicação. Similar ao gráfico anterior, o **Gráfico 4.9** apresenta a diversidade de documentos adotados, porém, grande parte utiliza o documento de requisitos e protótipos de baixo nível para esse fim. A soma dos tipos de documentos ultrapassa a quantidade de desenvolvedores, pois, alguns deles utilizam mais de um documento.

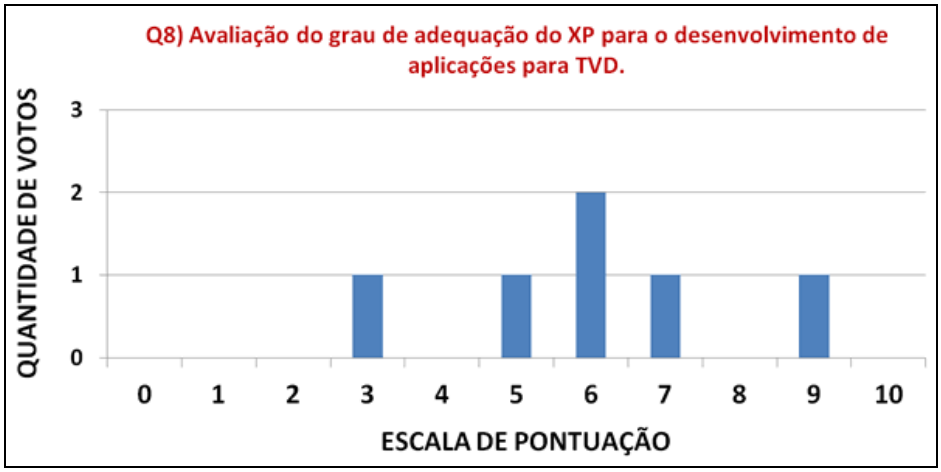
Gráfico 4.9 – Documentos adotados para o desenvolvimento



Fonte: o autor.

O **Gráfico 4.10** apresenta o grau de adequação do método XP ao desenvolvimento de aplicações para TVD. Esta pergunta foi opcional, voltada apenas para aqueles que já haviam adotado esse método, totalizando 6 participantes. A média das notas obtidas para essa questão foi 6,0 pontos.

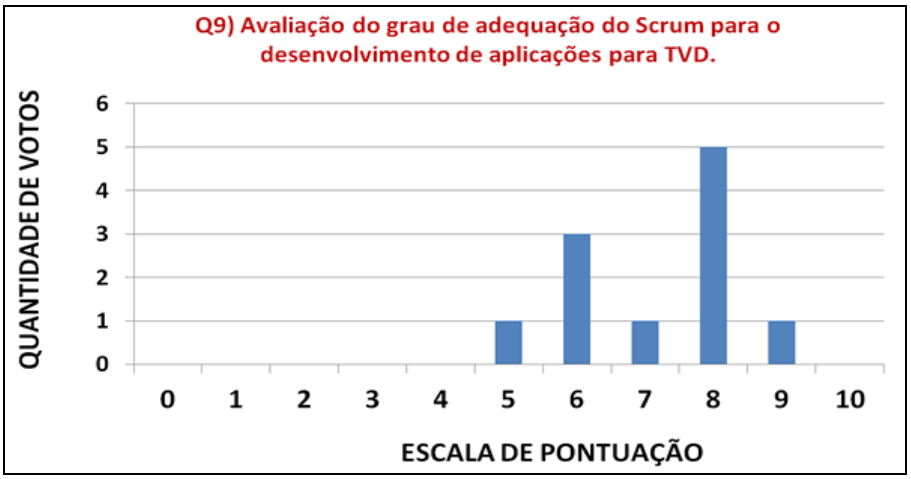
Gráfico 4.10 – Q8: grau de adequação do XP ao desenvolvimento para TVD



Fonte: o autor.

O **Gráfico 4.11** apresenta o grau de adequação do método Scrum ao desenvolvimento de aplicações para TVD. Esta pergunta foi opcional, voltada apenas para aqueles que já haviam adotado esse método, totalizando dez (10) desenvolvedores. A média das notas obtidas para essa questão foi 7,18 pontos.

Gráfico 4.11 – Q9: grau de adequação do Scrum ao desenvolvimento para TVD



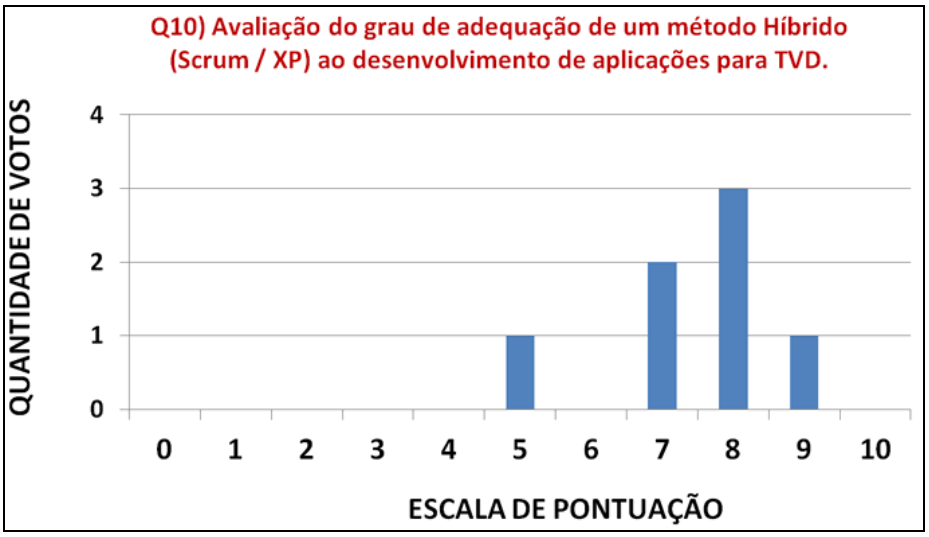
Fonte: o autor.

Devido ao fato da maior quantidade de programadores já terem utilizado o método Scrum, foi possível levantar quatro pontos de melhoria para esse método, tais melhorias se referem a:

- Adaptar as Sprints para oferecer maior auxílio à atividade de implementação;
- Oferecer recursos que permitam auxiliar na documentação das aplicações de forma simples e eficiente; e
- Oferecer ferramentas que facilitem a realização das atividades de testes de código.

O **Gráfico 4.12** apresenta o grau de adequação de um método Híbrido, formado por Scrum junto com XP, ao desenvolvimento de aplicações para TVD.

Gráfico 4.12 – Q10: grau de adequação de um método Híbrido (Scrum junto com XP)



Fonte: o autor.

Esta pergunta foi opcional, voltada apenas para aqueles que já haviam adotado esse método, totalizando sete (7) respostas. A média das notas obtidas para essa questão foi 7,43 pontos.

4.6. ANÁLISE E DISCUSSÃO DOS RESULTADOS INICIAIS

Apesar dos métodos XP e Híbrido adotarem programação em par, contabilizando 1 hora para cada programador, ambos os métodos permitiram realizar a atividade de implementação em menos tempo que o método Scrum.

Os resultados obtidos através dos questionários forneceram indícios de que, apesar das dificuldades enfrentadas, como desgaste, pouca experiência no desenvolvimento de aplicações para TVD e pouca prática em metodologias ágeis, os times foram unânimes e julgaram os três métodos utilizados adequados ao contexto de TVD, apontando como pontos fortes do processo de desenvolvimento as principais características das metodologias ágeis: entrega frequente de módulos funcionais, programação em pares e reuniões diárias.

O time que utilizou apenas Scrum identificou que se incorporada a característica de desenvolvimento em pares, poderia reduzir dificuldades inerentes à atividade de programação, permitindo desenvolver a aplicação em menos tempo, alcançando um melhor resultado final. Esta possibilidade foi reforçada pelos resultados coletados nos demais times.

A entrevista individual semiestruturada foi fundamental para retirar eventuais dúvidas sobre a participação de cada membro, bem como minimizar dúvidas que surgiram no questionário. Por meio dessa entrevista puderam ser levantados indícios de pontos de melhoria, como: dificuldade de entender o fluxo e o sincronismo de mídia, e a coleta de requisitos de mídias.

Através da entrevista em grupo foi constatado que os três times foram unânimes a respeito da dificuldade de coordenar o sincronismo de mídias a partir da segunda iteração, pois, há um aumento significativo do código, o que dificulta a compreensão do fluxo de mídia, tornando difícil a implementação de novas funcionalidades. O time que utilizou o método Híbrido (Scrum/XP) ressaltou que a elevada quantidade de processos definida pela junção dos dois métodos também eleva o desgaste do time.

Considerando a pouca quantidade de desenvolvedores de aplicações para TVD existentes no mercado e o elevado grau de experiência dos desenvolvedores que participaram do *survey* (a maior parte possui conhecimento acadêmico e profissional sobre TVD), os resultados obtidos pelo *survey* serviram como fonte complementar de dados, oferecendo indícios relevantes para a presente pesquisa.

Através do *survey* foi possível constatar que não existe um consenso sobre o melhor método a ser adotado. Foram levantados sete (7) métodos distintos, onze (11) desenvolvedores entre dezessete (17) utilizam métodos adaptados, e apenas dois (2) métodos puros. O método Scrum foi o único que apresentou reincidência, totalizando sete (7) ocorrências. Porém, sete (7) desenvolvedores não utilizam qualquer método de desenvolvimento.

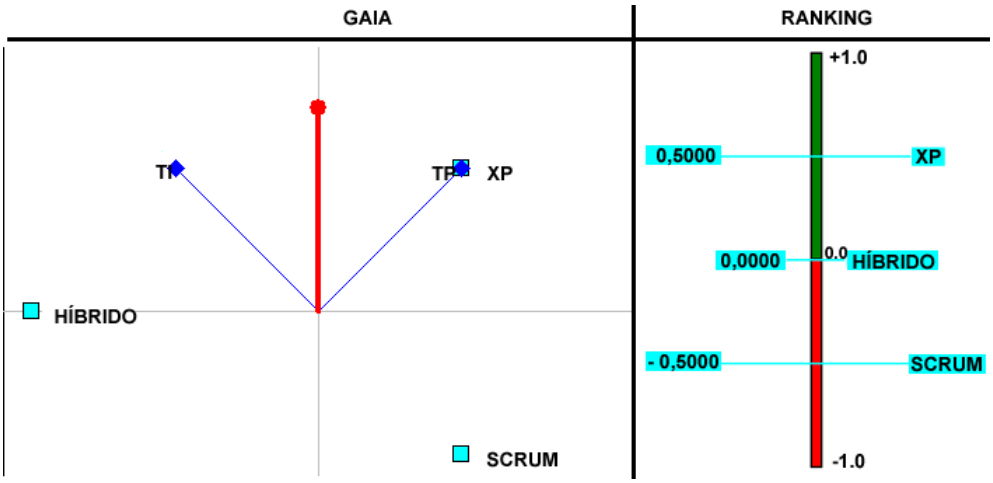
Sobre os documentos utilizados para guiar o processo de desenvolvimento e modelar as aplicações, também não houve consenso, totalizando 9 diferentes tipos de documentos utilizados. Apenas três (3) são reincidentes: UML, protótipos de baixo nível de fidelidade e documento de requisitos, com dois (2), três (3) e sete (7) ocorrências respectivamente. Isto indica que protótipos de baixo nível são interessantes e reforça a preocupação dos desenvolvedores com a coleta de requisitos.

Considerando que a escala de avaliação varia entre 0 e 10 pontos, a avaliação dos métodos foi um pouco acima da nota média (5), indicando que os três métodos investigados foram avaliados positivamente, porém, necessitam de adaptações para o contexto de TVD para alcançar avaliações mais elevadas.

4.7. ANALISE DOS RESULTADOS ATRAVÉS DO MÉTODO MULTICRITÉRIO (PROMETHEE II)

Para realizar a análise estatística dos dados quantitativos e qualitativos, foi adotada a Metodologia Multicritério de Apoio à Decisão (ENSSLIN et al., 2010), por meio do software Visual PROMETHEE II, versão acadêmica 1.4. Os valores relacionados ao Tempo do Planejamento (**TP**) e Implementação (**TI**), apresentadas no Quadro 4.6**,** foram utilizados como critérios de decisão para a realização da análise dos dados quantitativos de forma isolada, ou seja, sem interferência dos dados qualitativos, conforme apresentado na Figura 4.2, por meio do gráfico de Gaia e da classificação (*ranking*), estabelecidos pelo PROMETHEE II.

Figura 4.2 – Análise multicritério sobre os dados quantitativos (tempo)



Fonte: o autor.

O gráfico de Gaia estabelece uma relação entre os valores de cada critério, tanto os quantitativos como os qualitativos, seus respectivos pesos. O gráfico de Gaia posiciona seu eixo decisor (cor vermelha e ponta redonda) de

acordo com a adequação dos objetos de estudo. Dessa forma, o objeto de estudo mais próximo o eixo decisor é aquele mais adequado. Em (CAMPOS, 2011, p. 72) são apresentados mais detalhes sobre a ferramenta PROMETHEEII.

Os resultados apresentados pela Figura 4.2 correspondem ao cenário onde os pesos são iguais (peso = 1) para ambos os critérios (TP e TI). Apesar da soma dos tempos (TP e TI) serem menores para o método Híbrido, a sobreclassificação do TP e TI realizada pelo PROMETHEE aponta o método XP como mais interessante, seguido pelo Híbrido e, em última posição, o método Scrum, pois o eixo decisor fica dividido entre o método Híbrido e o método XP, com leve tendência para o método XP.

No entanto, considerar apenas os critérios quantitativos não atende à necessidade avaliar a "adequação" sobre métodos investigados. Por esse motivo, além dos dados quantitativos (tempo), foi realizada uma nova análise considerando também os critérios qualitativos, conforme apresentados no Quadro 4.9.

Quadro 4.9 – Critérios qualitativos e quantitativos.

Critérios	Tipo	Cenários (peso)					Impacto no desempenho geral	RESULTADOS		
		A	B	C	D	E		Scrum	XP	Híbrido
TP) Tempo do Planejamento	Quantitativo	1	2	1	1	2	Negativo	16,0h	15,5h	17,5h
TI) Tempo da Implementação	Quantitativo	1	1	2	1	2	Negativo	59,0h	46,15h	38,5h
C3) Adequação do método	Qualitativo	1	1	1	2	1	Positivo	7,18	6,00	7,43

Fonte: o autor.

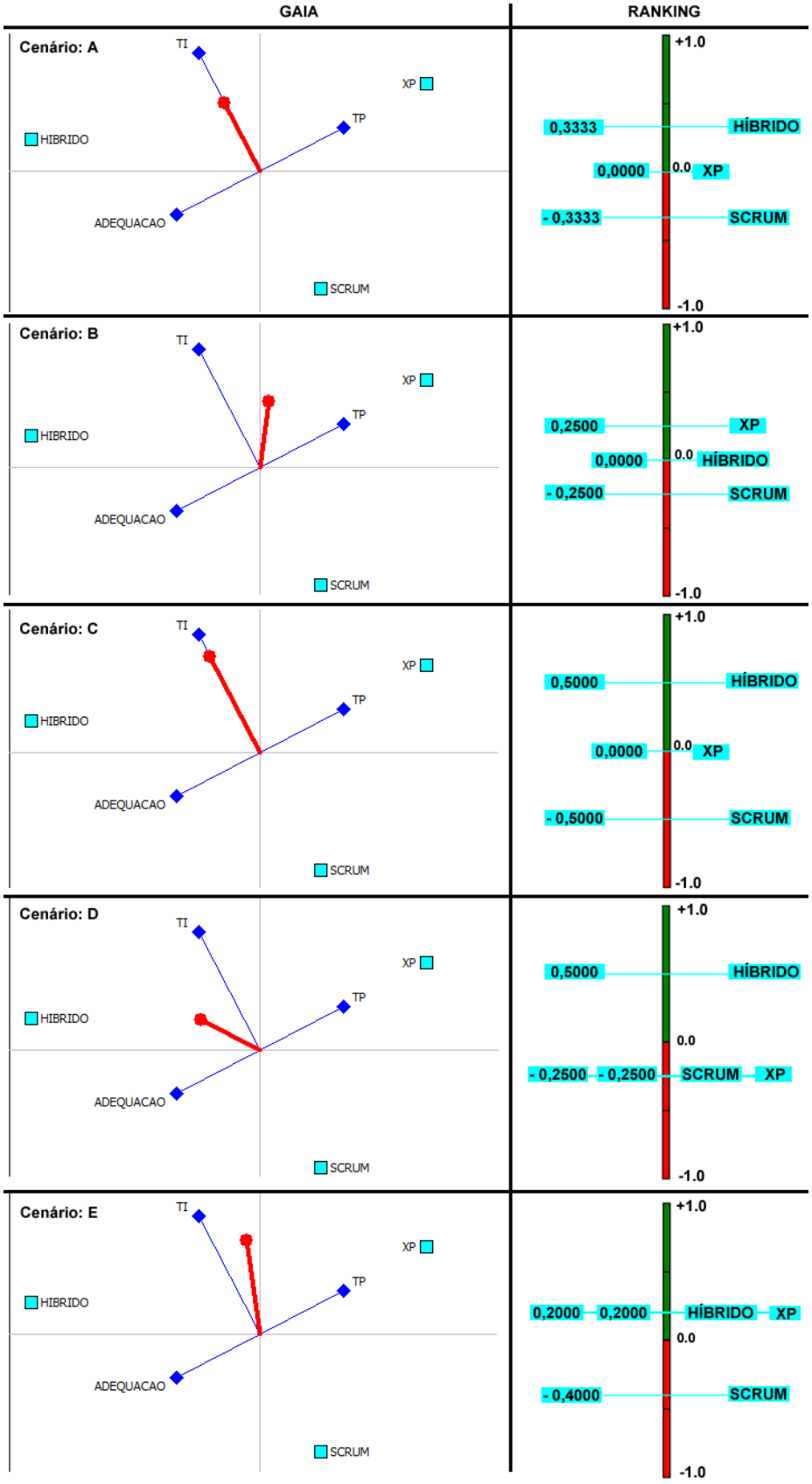
Os valores referentes ao critério qualitativo (C3) foram coletados através de um questionário aplicado com profissionais experientes no desenvolvimento de aplicações para TVD, cujas opções de respostas obedeciam à escala de Likert, com os seguintes valores: (1) muito ruim; (2) ruim; (3) regular; (4) bom; (5) muito bom.

No Quadro 4.9 a coluna "Cenários" apresenta os diferentes pesos atribuídos para cada critério (TP, TI e C3). Os diferentes pesos são utilizados

para analisar a viabilidade dos métodos investigados através de diferentes contextos (cenários). Por exemplo, caso algum time de desenvolvimento valorize mais o critério “Adequação do Método”, poderá considerar o Cenário D. A coluna “Impacto no desempenho geral” apresenta o impacto (positivo ou negativo) dos valores de cada critério. Por exemplo, quanto maior o tempo do planejamento ou da implementação, mais negativo é o desempenho do método. Quanto maior a adequação, mais positivo é o desempenho do método.

A variação dos pesos apresentada no Quadro 4.9 configura cinco possíveis cenários (A, B, C, D e E), cada cenário apresenta um peso maior para algum dos critérios, cada cenário pode representar que, em um determinado contexto do processo de desenvolvimento, aquele critério possui maior grau de prioridade que os demais. O gráfico do plano Gaia, apresentado na Figura 4.3, mantém as informações preservadas em 100%, onde o eixo decisor indica que, em todos os cenários, exceto no B, o melhor método é o Híbrido.

Figura 4.3 – Análise multicritério dos resultados quantitativos e qualitativos



Fonte: o autor.

A classificação (*ranking*) apresenta uma escala que varia de “+1.0” (melhor avaliação) até “-1.0” (pior avaliação). Entre estes extremos existe uma avaliação neutra “0.0”. Dessa forma, quanto mais próximo de “+1.0” estiver o método “W”, melhor o seu desempenho, ou seja, mais o método “W” sobreclassifica o método “Y”.

O método Híbrido foi considerado positivo para os cinco cenários analisados. Nos Cenários (A, C e D) o método Híbrido ficou em primeiro lugar, seguido pelo método XP. No Cenário E os métodos Híbrido e XP foram considerados equivalentes, pois foram atribuídos pesos maiores aos critérios quantitativos (TP e TI), onde o método XP apresentou mais rapidez sobre o critério TP e o método Híbrido apresentou maior rapidez sobre o critério TI, porém, a diferença entre tal eficiência aponta vantagem de 5% para o método Híbrido. No entanto, esta vantagem pode ser considerada pequena, pois, quando comparada com o método Scrum, o método Híbrido mostrou-se 34,75% mais rápido sobre o critério TI.

A pior avaliação do método Híbrido ocorreu no cenário B, onde o método XP ficou em primeiro lugar, pois atingiu o menor tempo para realizar a atividade de Planejamento, critério este que foi considerado prioritário neste cenário (peso = 2), seguido pelo método Híbrido que ficou em segundo lugar no *hanking*, atingindo uma pontuação intermediária (regular) igual a 0, porém, este foi o cenário onde as avaliações dos três métodos foi mais aproximada, variando +0,25 até -0,25, ou seja, neste cenário o desempenho dos métodos mostrou-se mais equilibrado.

Dentre todos os cenários, o Cenário D foi aquele que apresentou o melhor resultado para o método Híbrido (+0,50), apresentando uma diferença de 0,75 pontos com relação aos demais métodos, que foram avaliados igualmente negativos (-0,25). O método Scrum foi considerado negativo para os cinco cenários, ou seja, o último método classificado no *hanking*.

Para realizar a atividade de planejamento (TP) o método Híbrido dedicou 11,43% (2 horas) mais tempo que o método XP, e 8,57% (1,5 horas) mais tempo que o método Scrum. No entanto, o tempo dedicado à

implementação (TI) através do método Híbrido foi 16,58% (7,65 horas) menor que o método XP e 34,75% (20,5 horas) menor que o método Scrum.

A diferença entre a perda relacionada ao TP e o ganho relacionado ao TI aponta vantagem para o método Híbrido, pois, o tempo que o método Híbrido economizou durante a atividade de implementação (TI) foi 3,82 vezes superior ao tempo que o método XP economizou durante a atividade de planejamento (TP). Esta diferença é ainda maior se comparada com o método Scrum, onde o tempo que o método Híbrido economizou durante a atividade de implementação (TI) foi 13,67 vezes superior ao tempo que o método Scrum economizou durante a atividade de planejamento (TP).

Outro fator que favoreceu o método Híbrido foi a maior pontuação na avaliação (*survey*) realizada por profissionais experientes em TVD, onde a adequação do método Híbrido foi 3,36% melhor avaliado que o método Scrum e 19,25% melhor avaliado que o método XP.

4.8. CONSIDERAÇÕES SOBRE O EXPERIMENTO

Este experimento controlado realizou o desenvolvimento de três aplicações iguais, por três times distintos. Cada time adotou, por sorteio, um dos seguintes métodos: Scrum, XP e um método híbrido de Scrum com XP. Em seguida foi realizado um *survey* com profissionais experientes em TVD visando coletar a opinião da indústria sobre o processo de desenvolvimento de aplicações para TVD.

Os times envolvidos foram nivelados de forma homogênea, de acordo com seu grau de conhecimento e experiência sobre as tecnologias utilizadas. Esse nivelamento minimiza as variáveis que influenciam nos resultados, permitindo sua comparação.

Os resultados obtidos fornecem indícios de que os métodos ágeis investigados podem auxiliar no desenvolvimento de aplicações para TVD, pois, de modo geral, sua avaliação foi positiva (acima da média). Dentre os métodos investigados, o método Híbrido (Scrum junto com XP) foi aquele que obteve melhor desempenho sobre as variáveis quantitativas (tempo) e qualitativas (adequação), respondendo, desta forma, às questões de pesquisa (Q1 e Q2).

Porém, ainda existem pontos de melhoria que devem ser adaptados aos métodos ágeis para suprir demandas específicas dessa plataforma.

Dentre os pontos de melhoria levantados destacaram-se as necessidades de apoio às atividades de coleta de requisitos de mídia e definição de fluxo e sincronismo de mídias. Para isso, é necessário que tais conteúdos sejam devidamente coletados como requisitos da aplicação. Devido à sua especificidade, tais características não são previstas nas metodologias levantadas na literatura, fazendo-se necessário uma atenção especial no processo de desenvolvimento.

Dentro das limitações deste experimento, os resultados obtidos fornecem indícios da possibilidade de refutar as Hipóteses Nulas H0 e H1 e as Hipóteses alternativas H3 e H4, corroborando com as Hipóteses alternativas H2 e H5, apresentadas no início do presente capítulo. No entanto, tais resultados não são absolutos. Dependendo do contexto do time de desenvolvimento e da necessidade do projeto de software, é possível que o desempenho dos métodos obtenha resultados diferentes.

Vale ressaltar que os times envolvidos neste primeiro experimento eram relativamente inexperientes com as linguagens NCL, Lua e as demais ferramentas utilizadas para desenvolver aplicações para TVD. Diante disso, é provável que times mais experientes contabilizem valores absolutos menores sobre o tempo de desenvolvimento. No entanto, é provável que a diferença entre os métodos mantenha-se proporcional aos valores apresentados neste experimento.

A experiência e conhecimento dos desenvolvedores da indústria que participaram do *survey* foram fundamentais para coletar uma perspectiva profissional a respeito do desenvolvimento de aplicações para TVD, oferecendo dados complementares a este experimento. Através do *survey* foi possível constatar que não existe um consenso sobre um método de desenvolvimento preferido pelos desenvolvedores profissionais. A diversidade dos métodos utilizados, associada à elevada quantidade de desenvolvedores que não utiliza qualquer método de desenvolvimento, reforça os indícios que os métodos disponíveis na literatura não oferecem suporte suficiente para

serem adotados, fomentando a necessidade de um método que atenda às peculiaridades do desenvolvimento de aplicações para TVD.

O *survey* apontou que a grande variedade de documentos utilizados para representar aplicações para TVD também pode indicar que não existe um documento preferido. Os documentos utilizados são tradicionais na indústria de software, como documentos de requisitos e técnicas de prototipagem de baixo nível de fidelidade, mostrando uma lacuna que pode ser preenchida por um documento específico, customizado para aplicações de TVD.

A experiência prática dos participantes do *survey* sobre desenvolvimento de aplicações para TVD, juntamente com os resultados deste experimento e a revisão da literatura, forneceram indícios que possibilitaram a especificação e a formalização do método XDTv apresentado no próximo capítulo.

CAPÍTULO

5.

5. O MÉTODO XDTv

"Minuto de queixa é minuto perdido, arruinando talentos preciosos para a solução do problema."
Chico Xavier

O método *eXtreme Digital Television* (XDTv) foi especificado através da análise, formalização e estruturação de boas práticas existentes no desenvolvimento de software. Para isso, conforme apresentado anteriormente, foram coletados dados de três diferentes perspectivas: a) levantamento bibliográfico; b) experimentos controlados; e c) *survey* com desenvolvedores experientes.

O XDTv pode ser classificado como um método ágil, híbrido e customizado. Trata-se de um método ágil, pois segue a filosofia estabelecida pelo Manifesto Ágil (BECK et al., 2001). Híbrido, pois agrega técnicas utilizadas por métodos já existentes na engenharia de software. Customizado, pois seu modelo de processo, ciclo de vida e artefatos são adaptados com o objetivo de auxiliar no trato das peculiaridades inerentes ao desenvolvimento de aplicações para TVD, tais como: coleta de requisitos multimídias, rapidez no desenvolvimento, agilidade na alteração de requisitos e auxilio na comunicação de profissionais com diferentes perfis, conforme apresentado no Capítulo 1. Diante disso, o método XDTv estabelece os seguintes requisitos:

- Auxiliar nas atividades de planejamento, coleta e especificação de requisitos e implementação;
- Auxiliar na implementação da aplicação;
- Auxiliar na velocidade do desenvolvimento do projeto;
- Oferecer um ciclo de vida que permita antecipar a identificação de falhas durante as atividades desenvolvidas; e
- Oferecer um modelo de processo capaz de se adaptar a possíveis alterações de requisitos de modo a não inviabilizar a entrega do projeto ou extrapolações do seu orçamento.

Visando atender aos requisitos supracitados, a especificação do método XDTv apresenta as seguintes seções: **5.1)** artefatos específicos, sendo um protótipo de baixo nível de fidelidade (Protótipo de Aplicações para TV - PATv), e um modelo utilizado para especificar detalhes da implementação das aplicações multimídia (Modelo de Fluxo e Sincronismo de Mídias - MFSM); **5.2)** papéis/atores envolvidos e suas atribuições; e **5.3)** modelo de processo e o ciclo de vida do método, responsáveis por estruturar suas atividades.

Conforme apresentado nos Capítulos 1 e 2, segundo Sommerville (2011), existem quatro atividades básicas, comuns a todos os processos de software: 1) Especificação; 2) Desenvolvimento; 3) Validação; e 4) Evolução. O método XDTv pretende atuar principalmente nas atividades de Especificação e Desenvolvimento, considerando que a atividade de desenvolvimento contempla o planejamento e a implementação do software.

Na presente tese, a atividade de Validação não é abordada em todo seu potencial. Está previsto, apenas, o teste de aceitação junto ao usuário. As demais práticas, como: testes unitários e desenvolvimento dirigido a testes necessitam de pesquisas específicas, que estão fora do escopo da presente tese. Em Tekcan et al. (2012) é proposta uma abordagem para geração de casos de testes automáticos para TVD, viabilizando a verificação funcional baseada no perfil do usuário, com o objetivo de reduzir o tempo dos testes.

Existem pesquisas com foco na elaboração de estratégias eficientes para a realização de testes de software em aplicações para TVD. Em Araújo et al. (2011) é proposto um padrão de suíte de testes de conformidade para

aplicações declarativas *Nested Context Language* (NCL), disponível em: http://testsuite.gingancl.org.br. Em Silva (2010) é proposto o ambiente Xtation, cujo principal atrativo é realizar a validação confiável de aplicações para TVD.

Autores ressaltam que a atividade de teste está entre as mais onerosas atividades inerentes ao processo de desenvolvimento de software (ABRANTES; TRAVASSOS, 2012). Foi constatado em Marques Neto (2011) que existem dificuldades em adotar práticas de testes unitários e programação orientada a testes (*test-first*) para o desenvolvimento de aplicações multimídia.

Visto que as atividades de testes demandam profissionais especializados e implicam em mais tempo de dedicação ao processo de desenvolvimento (SOMMERVILLE, 2011), apesar de ser uma atividade relevante para o contexto de TVD, o método XDTv não especifica atividades relacionadas a testes unitários e desenvolvimento dirigido a testes, por considerar uma atividade específica, fora de seu escopo. No entanto, não impede que tais atividades, desenvolvidas em outros trabalhos (ARAÚJO et al., 2011), (SILVA, 2010) possam ser incorporadas ao método XDTv.

A atividade de Evolução também está fora do escopo da presente tese. Normalmente, as aplicações executadas sobre plataformas tradicionais (PC, Web, entre outras) são utilizadas constantemente por diversos anos, e com o passar do tempo, é esperado existam mudanças no contexto onde o software tradicional é utilizado, seja no âmbito empresarial ou pessoal, necessitando assim de ajustes (evolução) para atender as atuais regras de negócio. Conforme apresentado no Capítulo 1, aplicações para TVD possuem curto tempo de vida, pois estão associadas à exibição do conteúdo audiovisual da TV que pode durar poucos minutos e ser veiculado durante poucos dias, principalmente para aplicações voltadas às campanhas publicitárias. Dessa forma, a fase de Evolução de software em aplicações de TVD não possui a mesma demanda que as demais plataformas tradicionais.

Segundo Sommerville (2011), constantes modificações e a entrega incremental de novos requisitos inerentes à fase de Evolução podem acarretar em falhas ao longo do ciclo de vida do software. Por esse motivo, métodos

iterativos, que possuem constantes integrações de novas implementações de requisitos, são ideais para aplicações que possuem um ciclo de vida curto.

5.1. ARTEFATOS PROPOSTOS

Os artefatos do método XDTv foram elaborados a partir do levantamento bibliográfico, seguido pelos resultados obtidos através do primeiro experimento, associados ao primeiro *survey* realizado com desenvolvedores experientes.

A problemática da coleta e priorização de requisitos (ASFORA, 2009), apresentada no Capítulo 1, bem como as peculiaridades das aplicações para TVD, serviram de base motivacional para a elaboração do Protótipo de Aplicações para Tv (PATv) e o Modelo de Fluxo e Sincronismo de Mídias (MFSM), que utilizam de diferentes níveis de abstração para facilitar a compreensão de usuários não experientes, clientes e do time de desenvolvimento, auxiliando na coleta de requisitos multimídia e na implementação da aplicação.

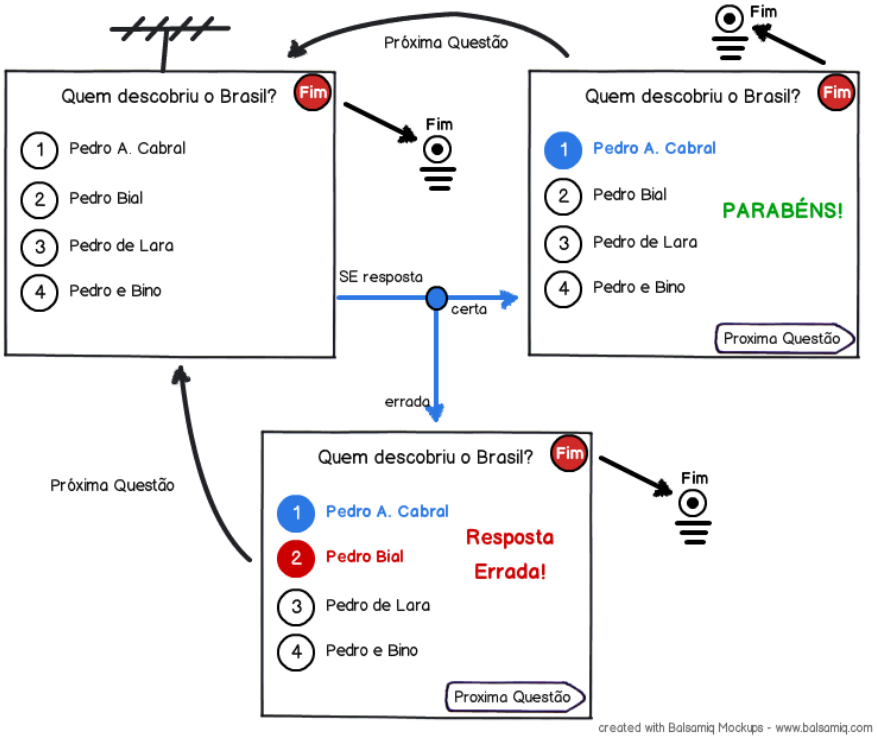
Esses artefatos possuem similaridade com linguagens de modelagem visual, as quais, segundo a literatura especializada, podem ser uma alternativa válida para auxiliar na especificação de uma aplicação, visto que oferecem um meio de comunicação adequado a diferentes perfis profissionais. Conforme apresentado no Capítulo 1, os times de desenvolvimento de aplicações para TVD tendem a ser multidisciplinares, envolvendo: diretor de arte, diretor de áudio, roteiristas, produtor, designer, atores, diferentes artistas, jornalistas e também engenheiros de software (MARQUES NETO, 2011), (VEIGA, 2006), (VEIGA; TAVARES, 2006, 2007).

Conforme apresentado no Capítulo 3, a técnica de *Storyboard* foi uma boa prática utilizada para auxiliar o desenvolvimento de aplicações por diversos trabalhos relacionados (BARTINDALE et al., 2013; MARQUES NETO, 2011; VEIGA, 2006; VEIGA; TAVARES, 2006, 2007). No experimento realizado em (VEIGA; TAVARES, 2006) essa técnica é avaliada de forma positiva. No entanto, é sugerido por aqueles entrevistados que seja adicionada ao *Storyboard* uma forma de indicar a sequência lógica de interação dos usuários.

Sob tal perspectiva o artefato PATv corresponde a uma prototipação de baixo nível de fidelidade (NIELSEN, 1993; SNYDER, 2003), baseado em *Storyboard* e oferece uma alternativa que atende a demanda levantada nos experimentos de (VEIGA; TAVARES, 2006), por meio da representação de uma sequência lógica baseada em eventos e na interação do usuário.

O artefato PATv pretende oferecer um sutil aprimoramento sobre a prototipagem de baixa fidelidade tornando-a uma prática adequada à integração e à comunicação de profissionais com diferentes perfis de conhecimento, mantendo características como: agilidade, eficiência, simplicidade e objetividade na comunicação entre os membros do time (VEIGA, 2006), (VEIGA; TAVARES, 2006, 2007). Esta técnica é capaz de descrever intuitivamente a quantidade de objetos de mídia, suas regiões, reutilização de regiões e conectores, links, mudança de estado, seu surgimento, finalização, estruturas condicionais e a sequência de eventos, entre outras possibilidades. Um exemplo do PATv é apresentado na **Figura 5.1**.

Figura 5.1 – Exemplo de utilização do artefato PATv



Fonte: o autor.

Apesar do exemplo da Figura 5.1 ser desenvolvido através do software Balsamiq[15], a utilização de software para prototipagem da interface é facultativo ao time, dependendo da demanda do projeto envolvido. O método XDTv consente a utilização de papel/cartolina e canetas coloridas para simplificar a coleta de requisitos e desenvolver o protótipo de baixo nível de fidelidade, mantendo a ideia de simplicidade, eficiência e sua adequação à comunicação dos envolvidos. Se necessário, a versão final do protótipo pode, posteriormente, ser desenvolvida através de um software de prototipagem de interfaces.

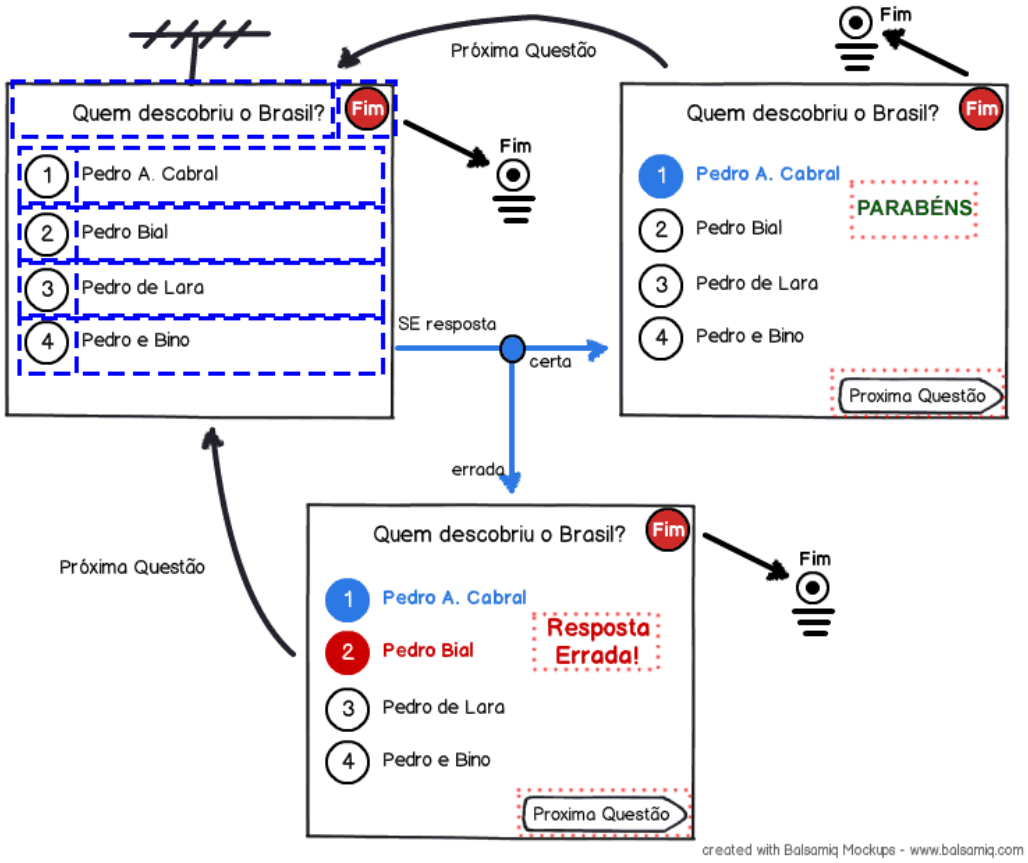
Conforme apresentado no Capítulo 1, aplicações para TVD estão fortemente voltadas para exibição de conteúdo multimídia, tais como: imagem, vídeo, áudio e texto. Para isso, é necessário que tais conteúdos sejam devidamente coletados como requisitos da aplicação, conforme apresentado nos trabalhos relacionados. A literatura ressalta a carência inerente ao levantamento de requisitos de aplicações multimídia (MARQUES NETO, 2011).

Entre suas características pode se destacar: altura, largura, início de exibição, duração de exibição, posição na pilha de mídias, ordem da sequência de exibição, entre outras.

A prática da prototipagem pretende contribuir significativamente para a coleta e especificação de tais requisitos, como regiões (elemento da linguagem NCL utilizado para diagramar as mídias na TV), suas propriedades conforme mostra o exemplo da Figura 5.2. As regiões tracejadas serão reutilizadas após a interação do usuário, independente se a resposta está certa ou errada. As novas regiões destacadas com pontilhado serão utilizadas apenas para exibição da resposta da questão.

[15] http://www.balsamiq.com

Figura 5.2 – Exemplo da coleta de requisitos e reuso de objetos de hipermídia com PATv



Fonte: o autor.

Conforme apresentado em trabalhos relacionados, uma aplicação de TVD pode conter mais de uma centena de objetos multimídia, dificultando sua compreensão e a implementação dos eventos que ocorrem sobre tais mídias (QUINTERO et al., 2013). Diante disso, a presente tese elaborou outro artefato denominado Modelo de Fluxo e Sincronismo de Mídias (MFSM), correspondendo a uma evolução da visão estrutural disponível em Soares Neto et al. (2007).

O objetivo do MFSM é representar qualquer aplicação de TVD através da projeção do seu diagrama, oferecendo uma visão completa dos objetos de mídia e os eventos que os manipulam, tornando explicito o objetivo do projeto através do planejamento destas aplicações no papel, ou seja, antes da fase de implementação. A diagramação do MFSM utiliza os elementos da linguagem de hipermídia para auxiliar o time a prever como a aplicação será

implementada. Considerando que esta pesquisa adota o padrão ISDB para realizar seus experimentos, a linguagem utilizada para viabilizar o experimento foi a NCL. No entanto, o método XDTv e seus artefatos podem ser adotados para as demais plataformas, necessitando apenas da alteração dos nomes dos eventos e as ações da linguagem adotada. Dessa forma, os principais eventos (conectores) da linguagem NCL são apresentados no Quadro 5.1.

QUADRO 5.1 – TIPOS DE EVENTOS PARA CONECTORES DA LINGUAGEM NCL

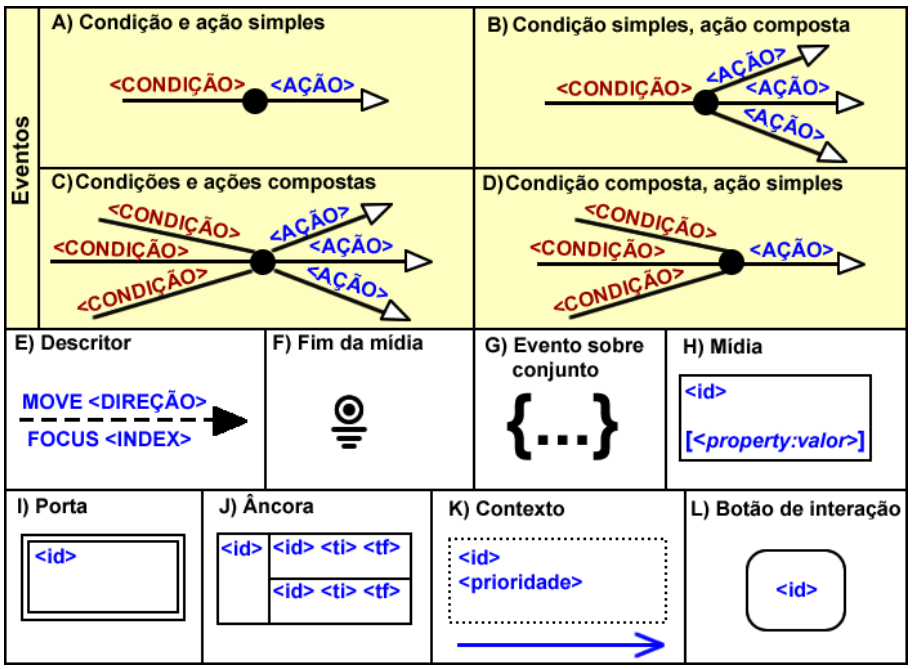
Condições:	onBegin; onEnd; onAbort; onPause; onResume; onSelection; onBeginAttribution; onEndAttribution; onKeySelection.
Ações:	Start; Stop; Abort; Pause; Resume; Set (variável).

Fonte: o autor.

Conforme apresentado anteriormente no Capítulo 1, a presente tese realiza a avaliação do método XDTv e seus artefatos através do Sistema Brasileiro de TV Digital (SBTVD). Por esse motivo, os exemplos apresentados correspondem a linguagens NCL e suas características. Além dos eventos (conectores), o MFSM também é capaz de representar graficamente as seguintes características essenciais de uma aplicação de TVD: mídias, âncoras, propriedades, script Lua, conectores, descritores, atributos, portas, contexto e *links*.

A representação gráfica referente ao diagrama do MFSM é apresentada na Figura 5.3. Esta representação visual possibilita aprofundar a coleta de requisitos do modelo PATv e documentar as características do comportamento de qualquer documento hipermídia. A simplicidade do modelo deve ser utilizada para documentar apenas o necessário, visando auxiliar o time de desenvolvimento, de acordo com os princípios dos métodos ágeis apresentados no Capítulo 2.

Figura 5.3 – Representação gráfica dos elementos que compõe o artefato MFSM



Fonte: o autor.

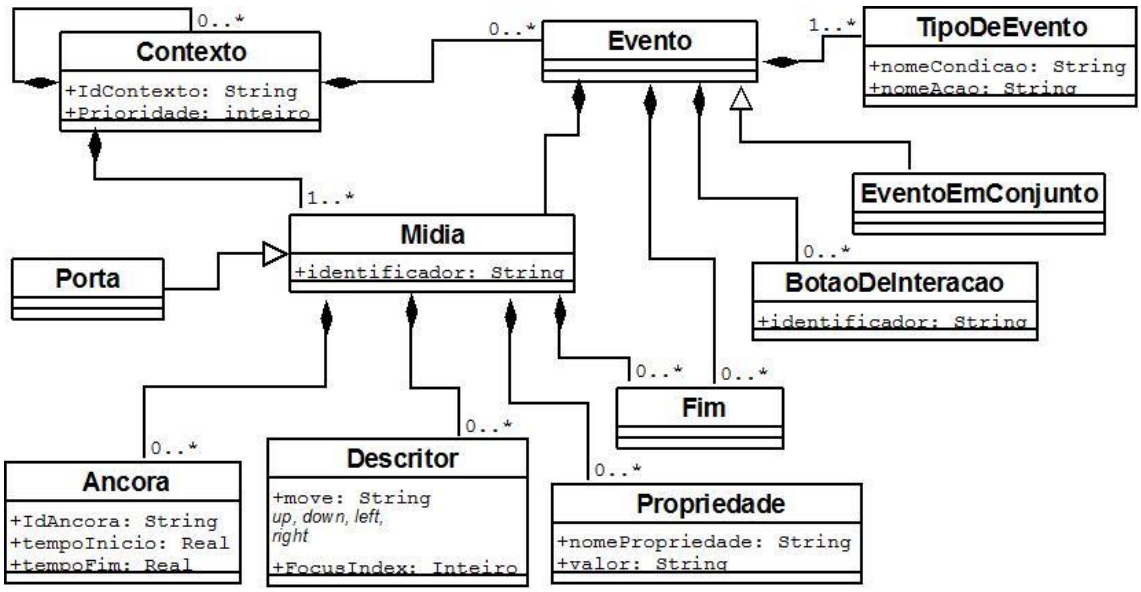
A Figura 5.3 apresenta as ilustrações (de “**A**” até “**L**”) responsáveis por realizar a modelagem da aplicação, especificando as principais características das aplicações multimídia de forma clara e objetiva. Os itens **(A, B, C, D)** correspondem aos conectores, representam qualquer tipo de evento (seus atributos: “condição” e “ação”) da linguagem NCL, desde **A)** condição simples e ação simples, até **D)** condição composta e ação simples. Os pontos pretos que dividem os conectores (condição/ação) correspondem ao momento de transição de mídias, que pode ocorrer através da interação do usuário ou de forma automática.

As demais representações correspondem a: **E) Descritor** e seus atributos “*move*” e “*focus*”, responsáveis por tornar imagens interativas, ou seja, selecionáveis como botões; **F)** sinaliza o fim da aplicação; **G)** auxilia a indicação de uma ação em comum para um conjunto de mídias, diminuindo a redundância de setas e a poluição visual do diagrama; **H)** uma mídia, seu identificador e possíveis propriedades; **I)** a porta, mídia principal do contexto; **J)**

uma mídia e suas âncoras, com seus respectivos identificadores, apontando tempo inicial e o tempo final; **K)** delineamento de um contexto; e **L)** corresponde a qualquer botão de interação disponível no controle remoto. Tal botão deve ser identificado (id) pelo nome. É possível representar todos os botões de interação como os botões coloridos (`Red`, `Green`, `Yellow`, `Blue`), numéricos (`0` até `9`), "`Ajuda`", "`Info`", "`Guia`", "`Menu`", "`Sair`", "`Volume`", "`Canal`", "`*`", "`#`", "`Ok`" ou uma das quatro setas direcionais.

Conforme citado anteriormente, o MFSM oferece informações abstratas sobre a implementação da aplicação. Dessa forma, não são apresentados todos os detalhes do código. A Figura 5.4 apresenta o metamodelo correspondente ao MFSM. Por meio dele é possível estruturar qualquer diagrama de acordo com a aplicação, atendendo a demanda por um modelo conceitual para aplicações de TVD (VEIGA, 2006).

Figura 5.4 – Metamodelo referente ao Modelo de Fluxo e Sincronismo de Mídias



Fonte: o autor.

O metamodelo apresentado na Figura 5.4 estabelece a regra de criação do MFSM, na qual um **Contexto** pode ser parte de *N* **Contextos** aninhados. Um **Evento** e uma **Mídia** são partes de um **Contexto**. Um **Evento** pode ser formado por diversos **TiposdeEvento** (atributos: condição e ação), **Mídias,**

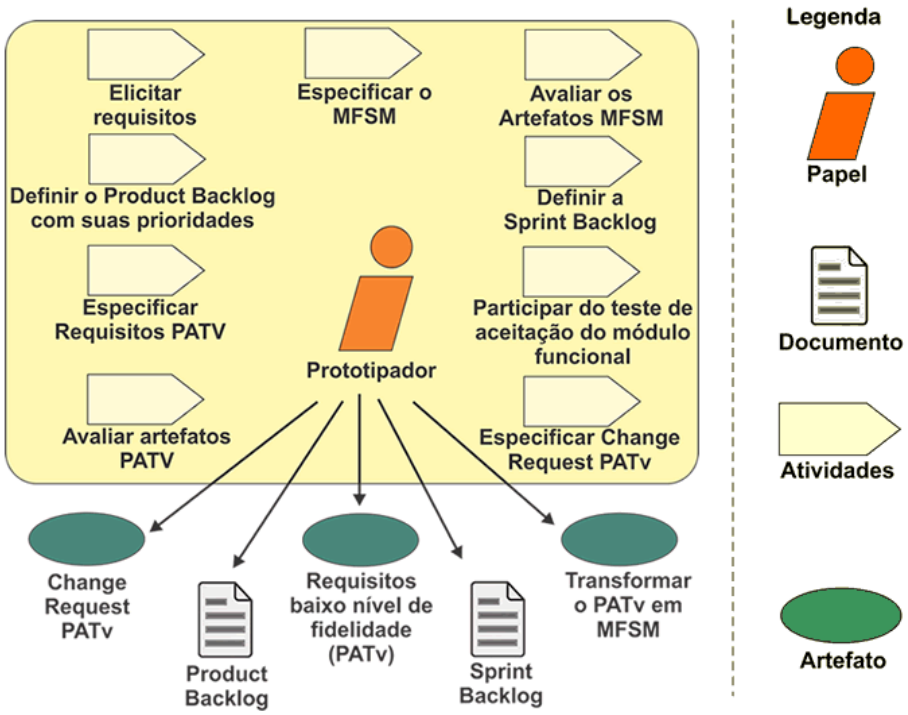
BotoesDeInteracao ou uma representação gráfica do **Fim** da aplicação. Um **EventoSobreConjunto** é um tipo de **Evento** e uma **Porta** é um tipo de **Mídia**. Uma **Mídia** pode ser formada pelas seguintes representações gráficas: **Fim**, **Propriedade**, **Descritor** e **Âncora**.

Usando abstração é possível utilizar o MFSM para modelar qualquer aplicação multimídia sem prejuízo às suas características, como sua simples representação, bem como a clareza na comunicação entre o time e o cliente, permitindo a visualização da aplicação como um todo em forma de grafo e documentar o projeto da aplicação, as características dos objetos multimídia e seus requisitos.

5.2. PAPÉIS

O time de desenvolvimento é composto por quatro papéis: Líder do projeto, Prototipador, Programador e o Cliente. O Prototipador é o principal responsável pela engenharia de requisitos. Ele realiza a coleta e especificação dos requisitos, utilizando a diagramação e avaliação dos artefatos PATv e MFSM, conforme apresentado na **Figura 5.5**.

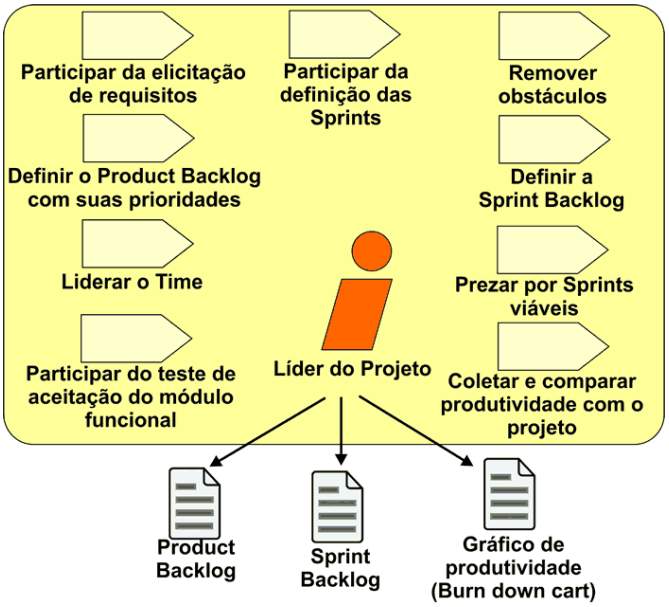
Figura 5.5 – O papel do Prototipador



Fonte: o autor.

O Líder do projeto possui responsabilidades similares ao Scrum Master, original do método (Scrum), garantindo a infraestrutura do ambiente de desenvolvimento e removendo obstáculos. Ele acumula, ainda, a função do Tracker, cujas atribuições são definidas originalmente pelo método (XP), conforme apresentado na **Figura 5.6**.

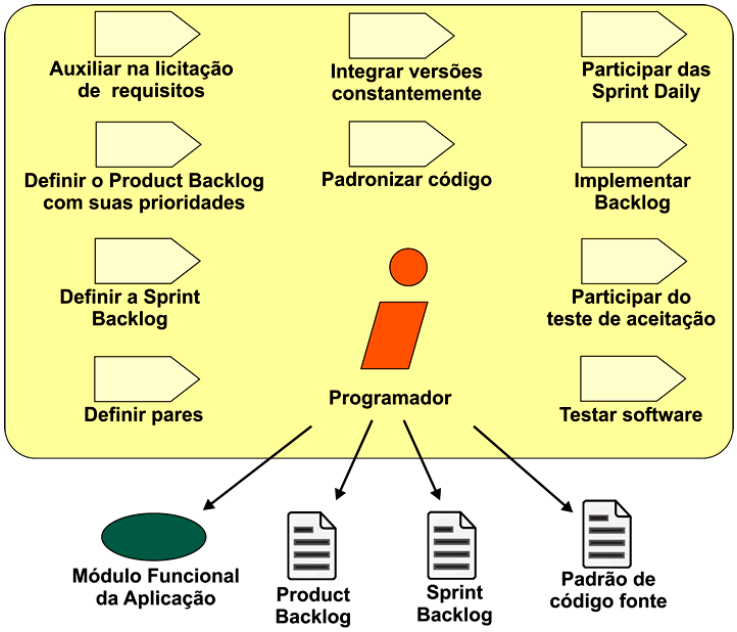
Figura 5.6 – O papel do Líder do Projeto



Fonte: o autor.

O programador é o responsável por implementar/codificar os módulos funcionais da aplicação. Este ator também acumula a função de testador, realizando testes de código e testes aceitação da aplicação implementada junto ao Cliente, antes da liberação de cada módulo funcional, conforme apresentado na **Figura 5.7**.

Figura 5.7 – O papel do Programador



Fonte: o autor.

Conforme apresentado no início do presente capítulo, testes de código estão fora do escopo da presente pesquisa. Entre tanto, existem trabalhos específicos voltados para testes de aplicações de TVD (TEKCAN et al., 2012; ARAÚJO et al., 2011).

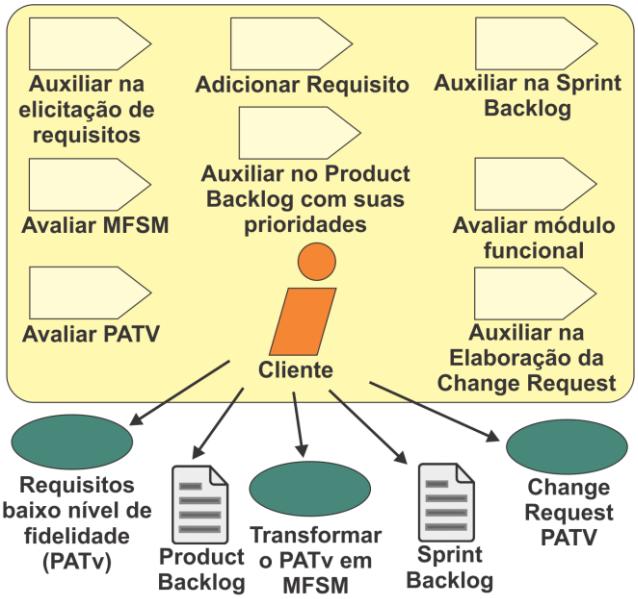
Tanto o Líder do projeto, como o Prototipador, quando houver tempo disponível, podem acumular a função de Programador. O presente método sugere que, tanto o Líder do projeto, como o Prototipador, tenham a habilidade de programar aplicações de TVD. Tal habilidade, além de auxiliar na velocidade da entrega da aplicação, contribui também para: a) comunicação entre os membros do time; b) estimativa de prazos mais precisa; e, c) compreensão das dificuldades a serem enfrentadas no decorrer do projeto.

Conforme apresentado no Capítulo 1, aplicações de TVD são coadjuvantes e visam agregar valor ao conteúdo audiovisual. Diante disso, o método XDTv recomenda que o papel do Cliente seja ocupado por algum profissional atuante na área de TV, com perfil profissional voltado para o desenvolvimento do conteúdo audiovisual.

O papel do Cliente é fundamental para o bom andamento do projeto, pois participa ativamente da elaboração do *Produckt Backlog*, na definição das

prioridades de cada requisito, porém, deve estar disponível para quaisquer esclarecimentos durante o desenvolvimento, conforme apresentado na **Figura 5.8**.

Figura 5.8 – O papel do Cliente



Fonte: o autor.

O Cliente também é o responsável pela avaliação dos artefatos desenvolvidos, como: o *produckt bcklog*, os diagramas PATV e MFSM, o módulo funcional e possíveis solicitações de mudança (*change request*). Cabe a ele verificar se existe alguma diferença entre o que está sendo desenvolvido e a expectativa da entrega.

5.3. MODELO DE PROCESSO E CICLO DE VIDA DO MÉTODO XDTV

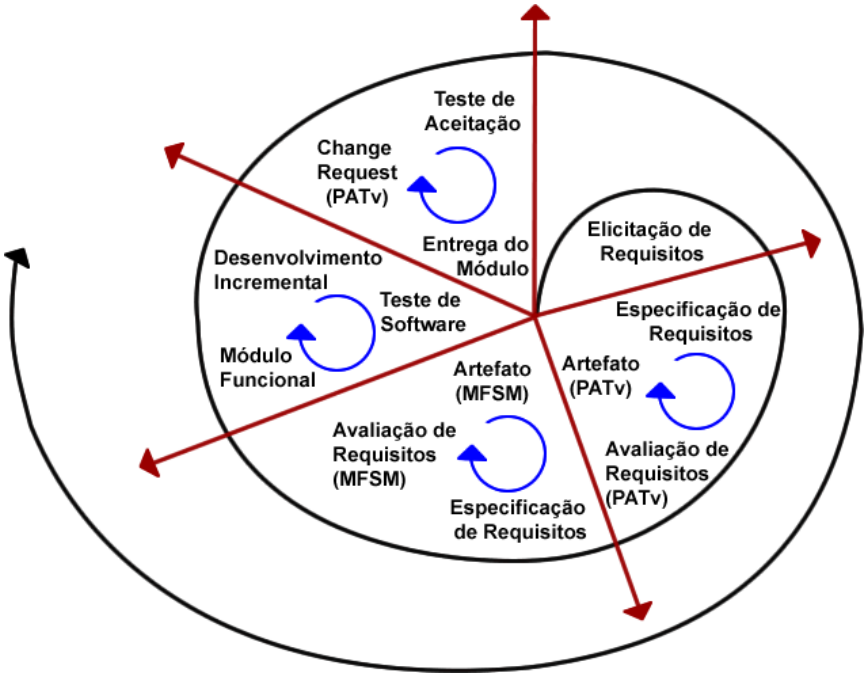
Além do projeto dos artefatos PATv e MFSM, o método XDTv também utiliza algumas atividades que são originais dos métodos mais utilizados na literatura especializada (Scrum e XP). Atividades relacionadas à gestão de projetos do método Scrum, como: *Product Backlog, Sprint Planning* e *Sprint Backlog,* foram incorporadas ao método XDTv pela simplicidade e eficiência de execução. O *Product Backlog* oferece uma lista com elevado nível de abstração de todos os requisitos da aplicação. A *Sprint Backlog* define aqueles

requisitos prioritários que serão implementados na próxima iteração, atividades que colaboram com a realização dos requisitos do método. Tais atividades são associadas às seguintes práticas do método XP: programação em par, padronização do código fonte e constante integração de versões.

Durante todos as fases do ciclo de desenvolvimento de software do método XDTv (especificação, planejamento, implementação e teste), os artefatos são constantemente avaliados assim que produzidos, antes de realizar a atividade seguinte. Se necessário, são corrigidos e reavaliados até que sejam aprovados. Nenhum artefato produzido pelo método XDTv é implementado sem ser avaliado pelo cliente em seguida.

Conforme apresentado na Figura 5.9, o método XDTv adota um modelo de processo de fluxo Iterativo e Incremental baseado em agilidade, características apresentadas no Capítulo 2 (BECK et al., 2001; PRESSMAN, 2011; SOMMERVILLE, 2011).

Figura 5.9 – Modelo de processo do método XDTv



Fonte: o autor.

Conforme apresentado na Figura 5.9, cada ciclo do método XDTv é subdividido nas seguintes etapas: **1ª)** Elicitação de requisitos; **2ª)** Especificação de requisitos com baixo nível de fidelidade (PATv); **3ª)** Transformação do PATv

em MFSM, elaborando um diagrama mais próximo da linguagem de desenvolvimento; **4ª)** Desenvolvimento/implementação do software; e **5ª)** Teste de aceitação. Assim que finalizada a 5ª e última etapa do ciclo do espiral, na qual é realizada a aprovação do módulo funcional, inicia-se um novo ciclo, retomando à atividade de elicitação de novos requisitos que serão incrementados, gerando uma nova versão do módulo funcional.

Na primeira etapa de elicitação de requisitos o time de desenvolvimento obtém junto ao cliente os primeiros requisitos da aplicação. Em seguida, na segunda e terceira etapas, ocorre a especificação de requisitos, onde o papel do Prototipador entra em ação e os artefatos PATv e MFSM são produzidos.

A quarta etapa recebe os artefatos PATv e MFSM como entrada de dados para realizar a implementação do módulo funcional do software. Assim que este for testado, será iniciada a quinta etapa, para a realização do teste de aceitação, possíveis solicitações de mudança (*change request*) e a entrega do módulo funcional ao cliente.

Todas as etapas do método XDTv apresentam um fluxo iterativo (PRESSMAN, 2011), responsável por avaliar os artefatos desenvolvidos, conforme apresentado no Capítulo 2, Figura 2.2 (b). Esta característica oferece oportunidade de antecipar possíveis falhas relacionadas à atividade de coleta de requisitos antes da implementação da aplicação, através da prototipagem de baixo nível de fidelidade (PATv) e da especificação do Modelo de Fluxo e Sincronismo de Mídias (MFSM), minimizando o custo de desenvolvimento, pois, conforme apresentado no Capítulo 1, uma alteração com custo de “1x” realizada na atividade de coleta e especificação de requisitos terá custo de 1,5x até 6x se realizada na fase de implementação, e “60x a 100x” se realizada na fase de manutenção do software (PRESSMAN, 2011). Portanto, quanto mais rápido ocorrer a detecção de falhas durante o processo de desenvolvimento, menor será o custo e o tempo dedicado para sua correção (ABRANTES; TRAVASSOS, 2012; PRESSMAN, 2011; SCHACH, 2009).

Dessa forma, atende aos requisitos e aos objetivos do método, possibilitando a entrega rápida de software funcionando, a alteração de requisitos e oferecendo artefatos (PATv e MFSM) customizados às

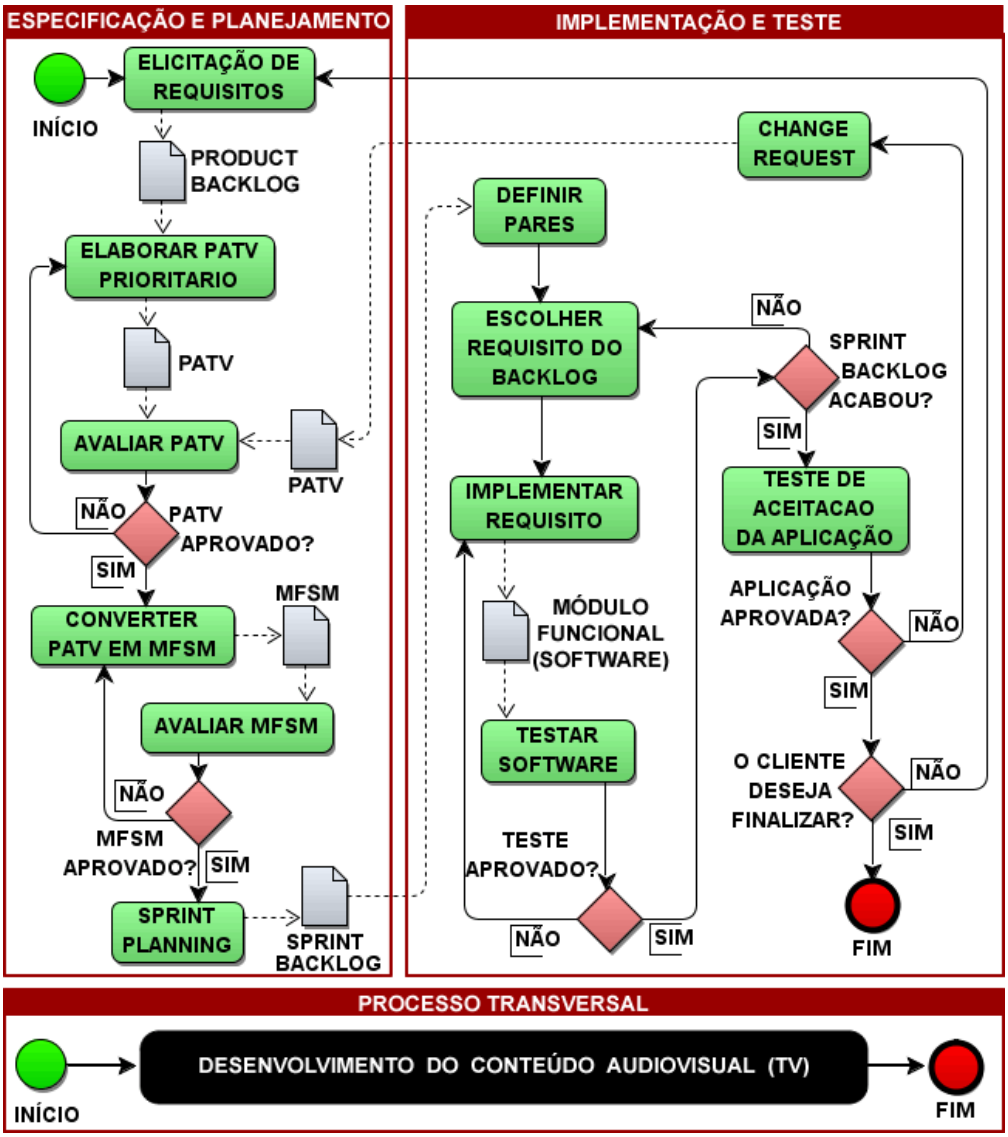
peculiaridades das aplicações de TVD, como rapidez no desenvolvimento, coleta de requisitos multimídias, documentação da aplicação, agilidade para alteração de requisitos e a auxílio na comunicação entre times multidisciplinares.

O método XDTv considera que os artefatos PATv e MFSM devem ser aprimorados e mantidos no decorrer de cada iteração, diferente, portanto, da prática corolária do método XP que define que o código fonte e os testes de software são os únicos artefatos a serem mantidos.

O PATv e o MFSM devem ser utilizados para auxiliar o time de desenvolvimento durante a realização dos testes e a implementação dos requisitos funcionais e não funcionais, auxiliando na comunicação entre a equipe multidisciplinar e o cliente.

Conforme apresentado no Capítulo 2, a ausência de modelos de processos de desenvolvimento é um dos principais problemas referentes ao desenvolvimento de aplicações para TVD (MARQUES NETO, 2011). Decidir de modo eficiente como executar as atividades do processo de desenvolvimento de software é um desafio para o desenvolvimento de aplicações para TVD (LULA; GUIMARÃES; TAVARES, 2011). Diante de tais dificuldades, a presente tese apresenta na Figura 5.10 a sequencia das atividades inerentes ao ciclo de desenvolvimento do método XDTv.

Figura 5.10 – Ciclo de vida do método XDTv



Fonte: o autor.

Conforme apresentado na **Figura 5.10**, o método inicia-se através da elicitação de requisitos, onde o cliente junto com o time utiliza alto nível de abstração para definir de forma objetiva, através de uma lista de tópicos, as funcionalidades desejadas na aplicação, bem como as suas prioridades. Resultando em uma tabela com todos os requisitos inicialmente previstos para o desenvolvimento da aplicação, formando o *Product Backlog* do projeto da aplicação.

Para minimizar a dificuldade de comunicação entre o time e o cliente, nas atividades que envolvem sua participação, como: elicitação de requisitos, avaliação do PATv, avaliação do MFSM, definição do *sprint backlog*, teste do módulo funcional e elaboração de solicitação de mudança (*change request*), é necessário que estejam presentes nesta etapa, ao menos, o Líder do projeto, um Prototipador e um Programador.

Baseado nos requisitos prioritários do *Product Backlog* é desenvolvido o primeiro artefato, o modelo PATv, resultando em um conjunto de protótipos de baixo nível de fidelidade, que serão validados junto ao cliente. Se reprovado, volta à etapa de elaboração; se aprovado, inicia a etapa de conversão do protótipo PATv em MFSM, aprofundando minuciosamente os eventos e as demais características da linguagem NCL conforme apresentado na **Figura 5.3**.

O cliente deve estar presente sempre que os artefatos forem avaliados, o que deve ocorrer primeiramente sobre os modelos PATv e MFSM, em seguida, sobre o testes de aceitação do módulo funcional da aplicação.

Os artefatos aprovados devem ficar acessíveis para todo o time. Fixar os protótipos na parede foi uma prática de sucesso utilizada no segundo *survey* realizado pela presente tese.

Baseado no PATv e no MFSM, faz-se a seleção dos modelos que serão implementados, resultando no *Sprint Backlog*. A cada ciclo são realizadas três atividades de validação: 1) PATv; 2) MFSM; e, 3) Módulo funcional. Essa sequência de validações pretende evitar falhas na coleta de requisitos, pois caso estas ocorram, serão detectadas antes da etapa de implementação, minimizando o tempo dedicado à refatoração de código e o aumento do custo do projeto, conforme explicado no Capítulo 1 (ABRANTES; TRAVASSOS, 2012; PRESSMAN, 2006, 2011; SCHACH, 2009).

Diante disso, o time inicia a fase de desenvolvimento. Sua duração deve ser fixa, porém, a quantidade de dias do ciclo depende do projeto que será desenvolvido, que deverá ser definido pelo time de desenvolvimento de acordo com as necessidades do cliente. A primeira atividade é a definição dos pares que irão atuar na implementação dos requisitos, onde cada par de

programadores implementará uma parte do MFSM a sua escolha, incrementando e gerando uma nova versão do módulo funcional, até que todos os itens existentes no *Sprint Backlog* sejam implementados. Devido aos resultados favoráveis obtidos através do Experimento 1, este método sugere que a implementação ocorra através de *pair programming*.

Assim que implementados os requisitos da *sprint backlog*, é realizado o teste de aceitação do módulo funcional junto ao cliente. Caso o módulo seja reprovado, é elaborada uma solicitação de alteração (*change request*) utilizando o artefato PATv, que será encaminhada para o próximo ciclo. Caso aprovado, é verificado junto ao cliente se há necessidade do desenvolvimento dos requisitos existentes no *Product Backlog*, possibilitando a reestruturação de prioridades, remover ou adicionar novos requisitos, reiniciando o ciclo de planejamento. Fica a critério da necessidade do cliente finalizar o projeto.

Vale destacar que, de acordo com a **Figura 5.10**, o desenvolvimento do conteúdo audiovisual envolvendo os profissionais de TV é uma atividade paralela, apresentada como uma “caixa preta”, pois está fora do escopo do presente trabalho. No entanto, os trabalhos relacionados apresentam pesquisas específicas cujo objetivo é integrar equipes de desenvolvimento de software com profissionais de desenvolvimento do conteúdo audiovisual da TV (LULA; GUIMARÃES; TAVARES, 2011) (VEIGA, 2006), (VEIGA; TAVARES, 2006, 2007).

A seguir, para ilustrar o funcionamento do método XDTv, é apresentado um exemplo de uma aplicação do tipo Quiz, passando por todo o ciclo de vida do método XDTv e os seus artefatos correspondentes a esta aplicação. Vale lembrar que esse é um pequeno exemplo, cuja aplicação possui poucos requisitos visando apenas ilustrar o funcionamento do método. A implementação completa desta aplicação em NCL está disponível no Apêndice O.

5.4. SÍNTESE DO DESENVOLVIMENTO DE UMA APLICAÇÃO QUIZ ATRAVÉS DO XDTV

Passo 1: elaboração do *Produckt Backlog*

O Quadro 5.2 apresenta um exemplo da lista correspondente ao *Product Backlog* e a classificação das prioridades de cada requisito. Os requisitos devem ser decompostos (divididos em requisitos mais simples) o máximo possível, para facilitar a atribuição do grau de prioridade. Tais informações devem ser definidas em reuniões entre o cliente e o time de desenvolvimento.

QUADRO 5.2 – EXEMPLO DE APLICAÇÃO DO TIPO QUIZ - PRODUCT BACKLOG

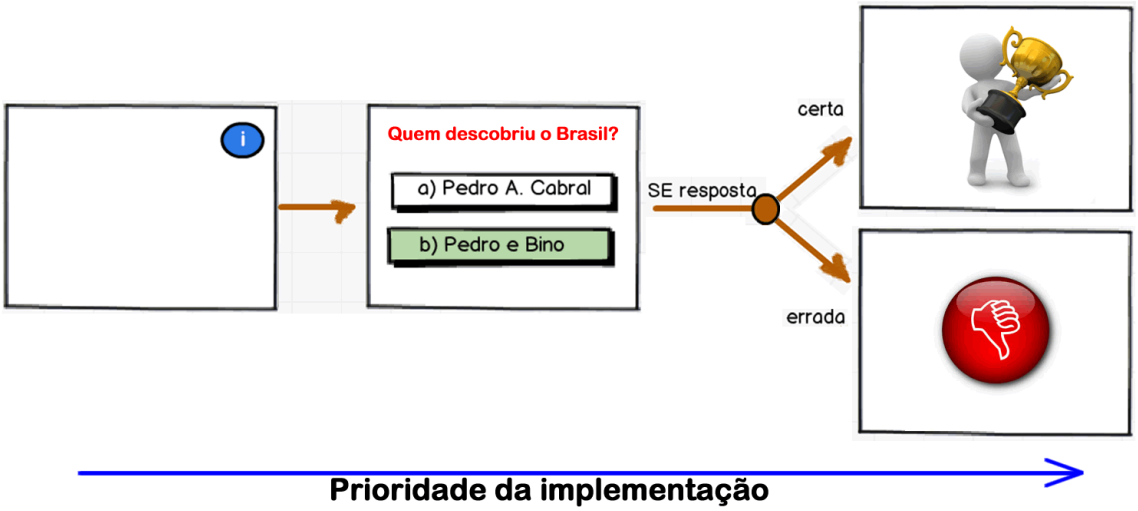
RF	NOME	DESCRIÇÃO	PRIORIDADE
RF01	**Ícone de interatividade**	**O ícone de interatividade é o primeiro objetivo de mídia da aplicação. Caso o usuário selecione este ícone, iniciará a primeira pergunta do Quiz.**	**1**
RF02	**Ícone desaparece**	**Se o usuário não selecionar o ícone de interatividade em 30 segundos, a aplicação deve ser finalizada.**	**1**
RF03	**Pergunta**	**Cada pergunta deve ter duas opções de respostas, uma certa e uma errada.**	**2**
RF04	**Resposta correta**	**Caso o usuário selecione a resposta correta, apresentar uma imagem de um troféu.**	**3**
RF05	**Resposta errada**	**Caso o usuário selecione a resposta errada, apresentar uma imagem do símbolo negativo.**	**3**
RF06	**Figura desaparece**	**Tanto a imagem do "Troféu" quanto a imagem do símbolo "Negativo" devem ser exibidas por 15 segundo, depois desse tempo a imagem desaparece.**	**4**
RF07	**...**	**...**	**5**
RF08	**...**	**...**	**5**

Fonte: o autor.

Passo 2: elaboração do artefato PATv

Os requisitos do *Produckt Backlog* que possuem maior grau de prioridade são selecionados para a elaboração do artefato PATv, que corresponde à representação da interface gráfica da compreensão do time de desenvolvimento sobre tais requisitos, conforme apresentado na **Figura 5.11**.

Figura 5.11 – Exemplo de Aplicação do tipo Quiz – artefato PATV.



Fonte: o autor.

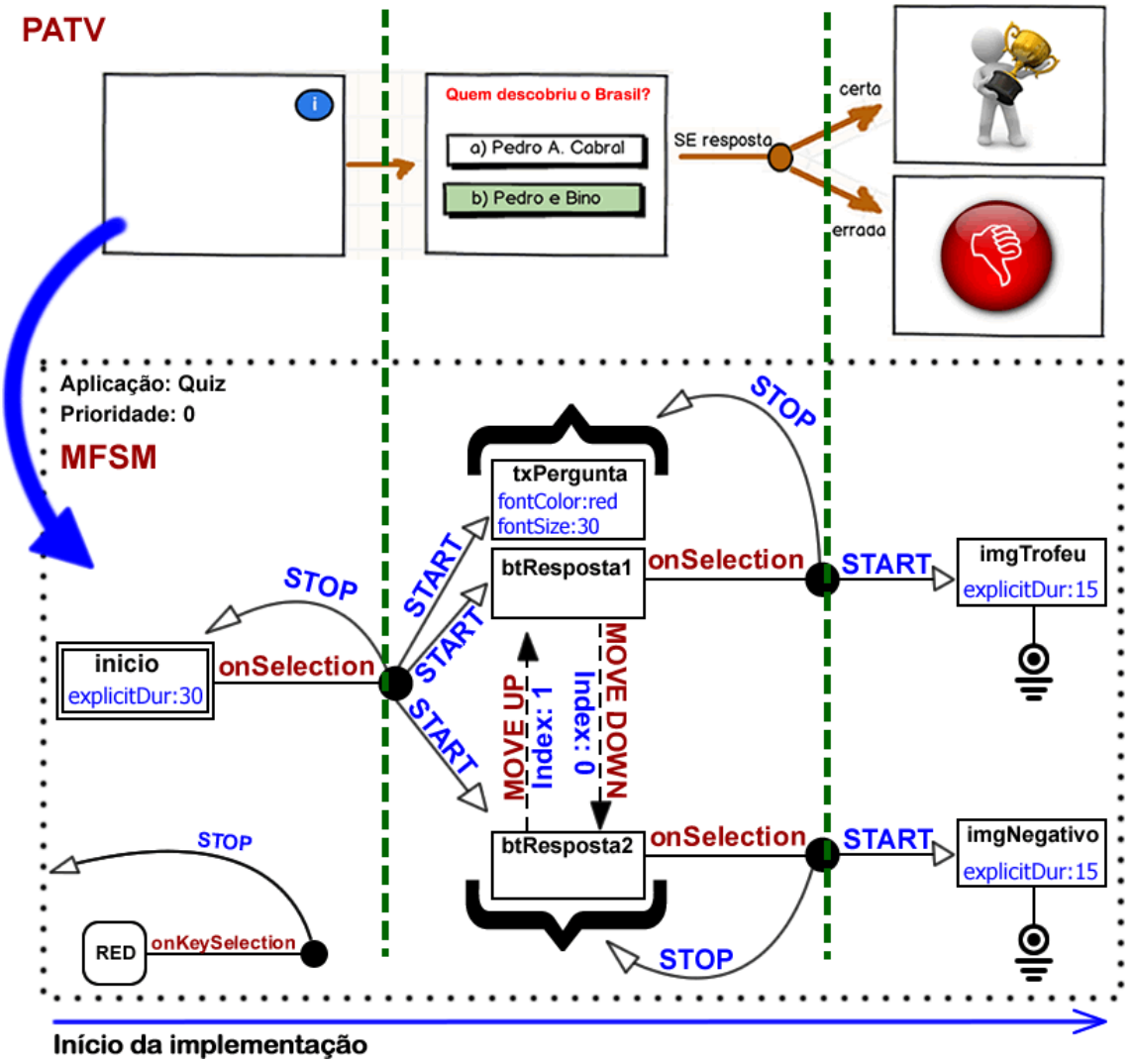
Este artefato permite visualizar todos os objetos de mídia e suas características, como: altura, largura, posição horizontal e vertical, formato, cor, entre outras. Tais características são importantes para a confecção das próprias mídias e também para auxiliar na implementação do código NCL, conforme apresentado no decorrer desta seção.

Na linguagem NCL, a mídia classificada como porta (`port`) é, excepcionalmente, iniciada espontaneamente dentro de um contexto, ou seja, é a primeira mídia a aparecer. Nos demais casos, a existência de uma mídia (*B*) está relacionada à existência de uma mídia (*A*) exibida anteriormente, ou seja, apenas uma mídia (*A*) é capaz de executar um evento (`start`, `stop`, `pause`, `resume`, entre outros.) sobre outra mídia (*B*). Diante disso, para evitar falhas na integração de versões, é importante que o Prototipador deixe claro em seus diagramas a prioridade do que deverá ser implementado, auxiliando os Programadores no momento da implementação, conforme apresentado na **Figura 5.11**.

Passo 3: elaboração do artefato MFSM

A partir do PATv é elaborado o diagrama correspondente ao MFSM, conforme apresentado na **Figura 5.12**. A linha tracejada na posição vertical é apenas ilustrativa, pretende indicar o mapeamento entre a interface do protótipo de baixo nível apresentado pelo modelo PATv e os eventos da linguagem NCL explícitos no modelo MFSM.

Figura 5.12 – Exemplo de aplicação do tipo Quiz – transformando o PATV no MFSM

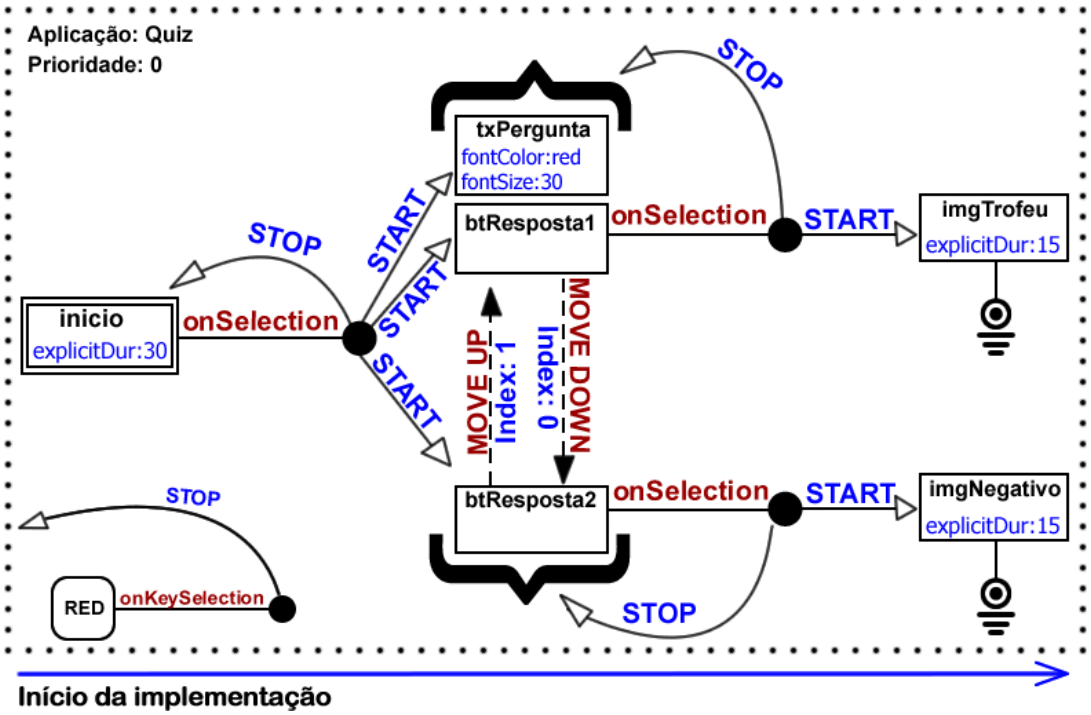


Fonte: o autor.

Elaboração do MFSM exige primeiramente que o prototipador e o time de desenvolvimento planejem como a aplicação será implementada, e

especifique sua implementação utilizando papel e canetas coloridas através do MFSM. Dessa forma, são identificados antes da implementação as propriedades das mídias, os conectores e descritores, portas e demais elementos da linguagem NCL, oferecendo uma visão detalhada do grau de complexidade da aplicação que será desenvolvida, conforme apresentado na **Figura 5.13**.

Figura 5.13 – Exemplo de Aplicação do tipo Quiz – MFSM completo



Fonte: o autor.

Conforme apresentado na **Figura 5.13**, todos os componentes do exemplo da aplicação Quiz são objetos de mídia distintos. Para melhor compreensão, no início do presente capítulo, a Figura 5.4 apresenta o metamodelo proposto pela presente tese, sob o qual foi estabelecido o MFSM, e a **Figura 5.3** apresenta o significado (legenda) dos elementos visuais utilizados para diagramar o MFSM.

O contexto, delineado pela linha pontilhada, apresenta seus atributos (nome da aplicação e prioridade da implementação). A mídia Início é a porta (`port`), ou seja, primeira mídia a ser exibida no contexto.

O evento "OnSelecion**SotpStartN**" eleva o nível de abstração, pois, por meio dele também é apontado o elemento `link` (da linguagem NCL) indicando as mídias que estão submetidas aos eventos `start` e `stop`.

Os eventos `onSelection`**`StopN`**`Start` realizam a ação `stop` sobre um conjunto de mídias, diminuindo a quantidade de setas. As mídias "`Inicio`", "`txPergunta`", "`imgTrofeu`" e "`imgNegativo`", possuem propriedades (`explicitDur`, `fontSize`, `fontColor`). No entanto, a linguagem NCL possui uma grande variedade de outras propriedades que, se necessário, podem ser adicionadas as mídias.

É possível definir também os efeitos dos descritores, conforme apresentado entre as mídias "`btResposta1`" e "`btResposta2`", apontando diretamente para as mídias e os índices (`index`) que receberão o foco (`focus`) sobre o atributo (`move`).

O evento "`onKeySelection`" define que, a qualquer momento, dentro do contexto da aplicação Quiz, ao pressionar o botão interativo vermelho (`RED`) do controle remoto, todos as mídias ativas devem ser paradas (`stopN`), finalizando a aplicação. Tais características possibilitam que, através de um pequeno diagrama, o time de desenvolvimento compreenda o funcionamento e a implementação da aplicação.

Passo 4: definição da *Sprint Backlog*

A *Sprint Backlog* é definida com base na prioridade dos requisitos e na visão detalhada do grau de complexidade apresentado pelo MFSM, onde são elencados os requisitos que serão implementados na próxima iteração (ciclo), conforme apresentado no Quadro 5.3.

QUADRO 5.3 – EXEMPLO DE APLICAÇÃO DO TIPO QUIZ - SPRINT BACKLOG

RF	NOME	DESCRIÇÃO	PRIORIDADE
RF01	**Ícone de interatividade**	**O ícone de interatividade é o primeiro objetivo de mídia da aplicação. Caso o usuário selecione este ícone, iniciará a primeira pergunta do Quiz.**	**1**
RF02	**Ícone desaparece**	**Se o usuário não selecionar o ícone de interatividade em 30 segundos, a aplicação deve ser finalizada.**	**1**
RF03	**Pergunta**	**Cada pergunta deve ter duas opções de respostas, uma certa e uma errada.**	**2**
RF04	**Resposta correta**	**Caso o usuário selecione a resposta correta, apresentar uma imagem de um troféu.**	**3**
FR05	**Resposta errada**	**Caso o usuário selecione a resposta errada, apresentar uma imagem do símbolo negativo.**	**3**
RF06	**Figura desaparece**	**Tanto a imagem do Troféu quanto a imagem do símbolo "Negativo" devem ser exibidas por 15 segundo, depois desse tempo a imagem desaparece.**	**4**

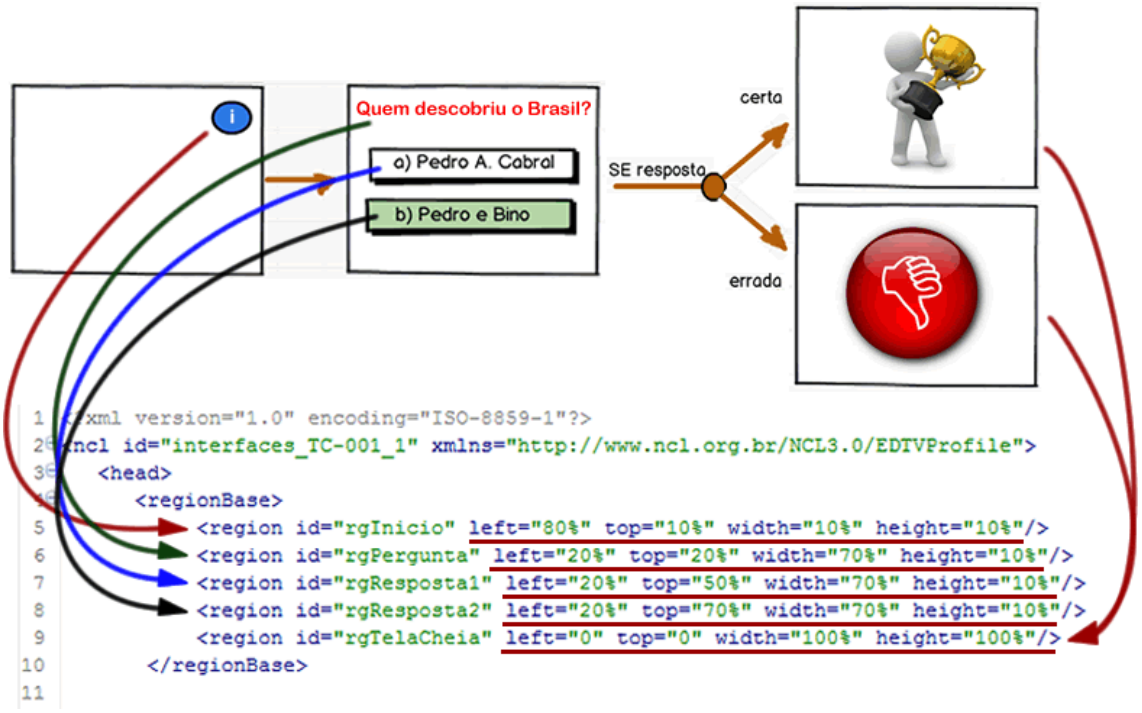
Fonte: o autor.

Todos os requisitos do Quadro 5.3 foram implementados. O código fonte NCL completo é apresentado no Apêndice O. Com o intuito de apresentar uma visão didática, a seguir, é traçado um paralelo entre a utilização dos artefatos PATv e MFSM e a implementação do código NCL da aplicação Quiz.

Passo 5 – a implementação do módulo funcional.

Baseado no artefato PATv é possível identificar a quantidade de mídias utilizadas, sua localização na tela (`region`) e as propriedades das regiões como: posição esquerda (`left`), posição superior (`top`), largura (`width`), altura (`height`) e a posição na pilha de mídias (`zIndex`), e a mídia que será utilizada com porta (`port`), conforme apresentado na **Figura 5.14**.

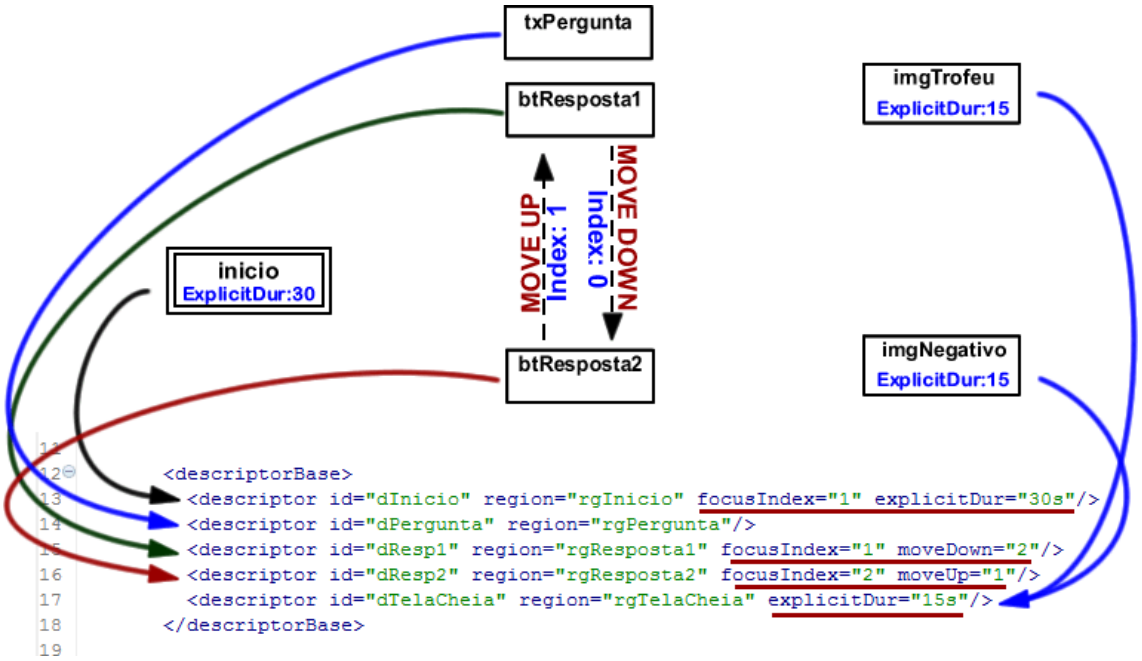
Figura 5.14 – Exemplo Quiz: utilizando PATv para implementar Regiões NCL



Fonte: o autor.

O artefato MFSM apresentado na **Figura 5.15** oferece informações adicionais importantes para a implementação dos descritores (`descriptor`) NCL, como: identificador (`id`), região (`region`) onde será exibida a mídia, foco para cada botão (`focusIndex`), controle sobre as setas direcionais (`moveUp`, `moveDown`), tempo de exibição da mídia (`explicitDur`) e demais características.

Figura 5.15 – Exemplo Quiz: utilizando MFSM para implementar Descritores NCL



Fonte: o autor.

O MFSM auxilia na visualização dos elementos que podem ser reutilizados, como por exemplo, as regiões `rgTelaCheia`, esta região é definida apenas uma vez, porém, é reutilizada para apresentar as mídias que ocupam toda a tela. O MFSM contribui também para a implementação dos conectores (eventos) da aplicação. Para implementar a aplicação Quiz foi necessário apenas o conector (`onSelectionStartNStopN`), conforme apresentado na **Figura 5.16**.

Figura 5.16 – Exemplo Quiz: Utilizando MFSM para implementar Conectores NCL


```
19
20      <connectorBase>
21        <causalConnector id="onSelectionStartNStopN">
22              <simpleCondition role="onSelection"/>
23              <compoundAction operator="seq">
24                  <simpleAction role="start" max="unbounded" qualifier="par"/>
25                  <simpleAction role="stop" max="unbounded" qualifier="par"/>
26              </compoundAction>
27        </causalConnector>
28      </connectorBase>
29  </head>
```

Fonte: o autor.

O artefato PATv também contribui para a implementação das mídias (`media`) facilitando a identificação da mídia porta (`port`) e suas propriedades (`property`), como (`fontColor` e `fontSize`), conforme apresentado na **Figura 5.17**.

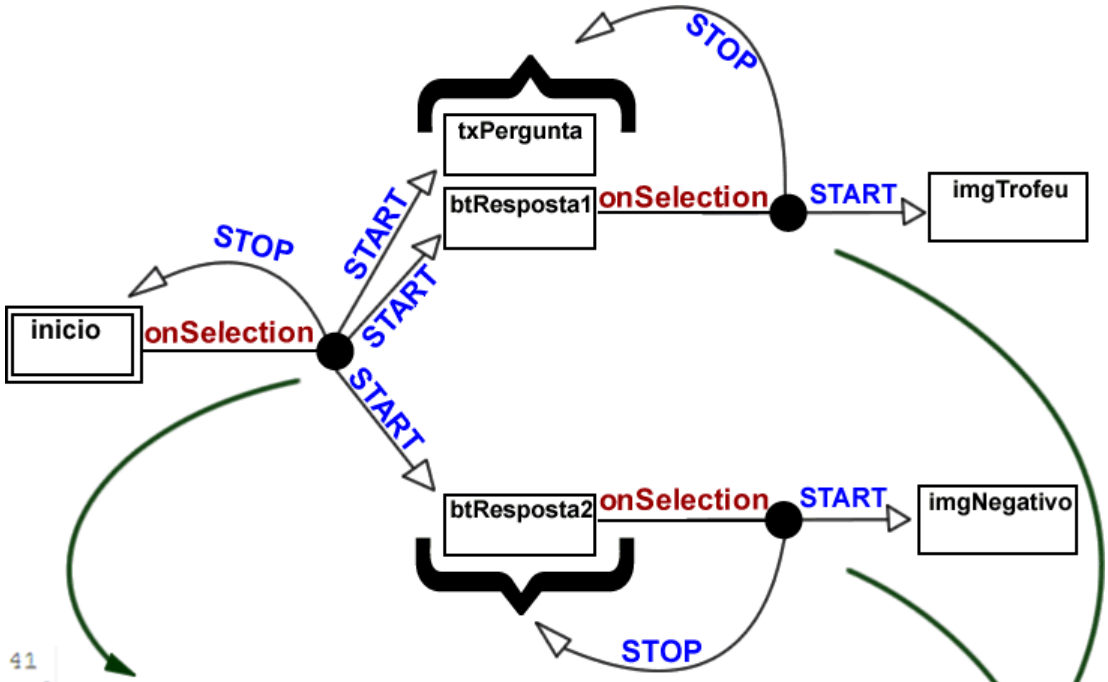
Figura 5.17 – Exemplo Quiz: utilizando PATv para implementar Mídias NCL


```
30  <body>
31      <port id="pInicio" component="inicio"/>
32      <media type="image/png" id="inicio" src="media/btInicio.png" descriptor="dInicio"/>
33      <media type="text/plain" id="txPergunta" src="media/pergunta.txt" descriptor="dPergunta">
34          <property name="fontColor" value="red"/>
35          <property name="fontSize" value="30"/>
36      </media>
37      <media type="image/png" id="btResposta1" src="media/resposta1.png" descriptor="dResp1"/>
38      <media type="image/png" id="btResposta2" src="media/resposta2.png" descriptor="dResp2"/>
39      <media type="image/png" id="imgTrofeu" src="media/imgTrofeu.png" descriptor="dTelaCheia"/>
40      <media type="image/png" id="imgNegativo" src="media/imgNegativo.png" descriptor="dTelaCheia"/>
41
```

Fonte: o autor.

O MFSM auxilia a implementação dos elos (`links`), pois apresenta como o conector (`onSelectionStartNStopN`) será utilizado, apontando quais as mídias (`inicio`, `txPergunta`, `btResposta1`, `btResposta2`, `imgTrofeu` e `imgBurinho`) estão relacionadas as ações (`role`), conforme é apresentado na **Figura 5.18**.

Figura 5.18 – Exemplo Quiz: Utilizando MFSM para implementar Links NCL




```
41
42      <link xconnector="onSelectionStartNStopN">
43            <bind component="inicio" role="onSelection"/>
44            <bind component="inicio" role="stop"/>
45            <bind component="txPergunta" role="start"/>
46            <bind component="btResposta1" role="start"/>
47            <bind component="btResposta2" role="start"/>
48      </link>
49      <link xconnector="onSelectionStartNStopN">
50            <bind component="btResposta1" role="onSelection"/>
51            <bind component="txPergunta" role="stop"/>
52            <bind component="btResposta1" role="stop"/>
53            <bind component="btResposta2" role="stop"/>
54            <bind component="imgTrofeu" role="start"/>
55      </link>
56      <link xconnector="onSelectionStartNStopN">
57            <bind component="btResposta2" role="onSelection"/>
58            <bind component="txPergunta" role="stop"/>
59            <bind component="btResposta1" role="stop"/>
60            <bind component="btResposta2" role="stop"/>
61            <bind component="imgNegativo" role="start"/>
62      </link>
63    </body>
64 </ncl>
```

Fonte: o autor.

O modelo PATv, assim como o MFSM, são artefatos projetados pela presente tese com o objetivo de auxiliar as fases de Elicitação e Especificação de Requisitos, Planejamento e Desenvolvimento, visando atender de forma

adequada às peculiaridades das aplicações para TVD. A simplicidade da sua elaboração auxilia na comunicação com o cliente facilitando a gestão do fluxo de mídias, principal dificuldade encontrada no primeiro experimento e também em trabalhos relacionados (MARQUES NETO, 2011).

5.5. CONSIDERAÇÕES SOBRE O CAPÍTULO

O método ágil *eXtreme Digital Television* foi especificado visando auxiliar nas peculiaridades das aplicações para TVD, com foco na velocidade do desenvolvimento, visando minimizar as seguintes dificuldades: coleta de requisitos das mídias, compreensão do fluxo das mídias, comunicação entre os membros do time possibilidade de alterar requisitos sem prejuízo ao prazo de entrega e custo do projeto.

Para isso, o método XDTv oferece dois artefatos, o Protótipo de Aplicações para TV (PATv), de baixa fidelidade, e o Modelo de Fluxo e Sincronismo de Mídia (MFSM), projetados para coletar os requisitos de mídia de forma simples e eficiente.

O PATv evolui o conceito do tradicional *storyboard,* técnica frequentemente utilizada em trabalhos relacionados (BARTINDALE et al., 2013; MARQUES NETO, 2011; VEIGA, 2006; VEIGA; TAVARES, 2006, 2007), adicionando estruturas condicionais que permitem indicar a transição de eventos e mudanças de tela, conforme sugerido em (VEIGA; TAVARES, 2006). Por meio destas, é possível identificar de forma intuitiva, regiões, quantidade de objetos de mídia, formato, entre outras características da aplicação.

Os artefatos PATv e MFSM não se restringem a aplicações de TVD, estes podem ser utilizados em outros contextos que necessitem da representação visual para representar iteração do usuário e eventos da linguagem de programação, pois auxiliam o time de desenvolvimento na atividade de coleta e especificação requisitos, bem como, na comunicação do time com o cliente.

O MFSM é um modelo mais detalhado, que apresenta características da linguagem NCL, como: mídias e suas propriedades, o fluxo e os eventos que ocorrem sobre cada uma delas. Este modelo pretende auxiliar tanto na coleta

de requisitos, como na implementação da aplicação. O MFSM pode ser adaptado para outras linguagens baseado no metamodelo proposto pela presente tese, alterando apenas os nomes dos eventos (condição/ação), de acordo com as características oferecidas pela sintaxe da linguagem utilizada.

Apesar de o presente trabalho adotar o padrão ISDB-Tb para realizar seus experimentos, conforme apresentado no Capítulo 1, as contribuições do método XDTv não se restringem ao sistema ISDB-Tb, podendo ser utilizado também para auxiliar o desenvolvimento de aplicações no sistema Europeu, *Digital Video Broadcasting* (DVB) Transmissão de Vídeo Digital (ETSI, 2005), e o sistema Norte-Americano *Advanced Television System Committee* (ATSC) (ATSC, 2006; SONG; PARK, 2006; SHELLHAMMER, 2008).

Além dos artefatos supracitados, o presente capítulo apresentou os papeis, o modelo de processo e o ciclo de vida do método XDTv, que especifica a realização das atividades e a utilização dos artefatos, de forma customizada para atender as peculiaridades do desenvolvimento de aplicações para TVD, minimizando a dificuldade relatada por (MARQUES NETO, 2011). O método XDTv, seus artefatos, modelo de processo e ciclo de vida são avaliados no experimento apresentado no próximo capítulo.

CAPÍTULO

6.

6. AVALIAÇÃO DO MÉTODO XDTV

"Toda prova aparece para elastecer-nos a força e aperfeiçoar-nos a experiência."
Chico Xavier

O objetivo deste capítulo é apresentar a avaliação do método XDTv, comparando-o com o método híbrido, formado pela combinação de Scrum e XP, baseado em trabalhos relacionados (MUSHTAQ; QURESHI, 2012). Conforme citado no Capítulo 1, este experimento foi baseado na proposta de experimentação definida em TRAVASSOS (2002), que estabelece as seguintes etapas: Definição, Planejamento, Instrumentação e Execução do experimento.

O Quadro 6.1 apresentada uma síntese com as principais características (Foco, Papéis, Artefatos e Atividades) do método Híbrido (XP/Scrum) e do método XDTv, proposto pela presente tese.

Baseado no referencial teórico, nos trabalhos relacionados e nos resultados obtidos pelo primeiro experimento, este segundo experimento partiu das seguintes questões de pesquisa a serem investigadas:

- **Q1.** Qual método melhor atende às atividades de coleta e especificação de requisitos, planejamento e implementação de aplicações para TVD?

- **Q1.** Qual método permite o desenvolvimento mais rápido de aplicações para TVD?

Para guiar a investigação das questões de pesquisa citadas anteriormente, foram definidas as seguintes hipóteses:

- Hipótese nula **(H0)**: o tempo utilizado para o desenvolvimento das aplicações através do método híbrido formado por Scrum junto com XP (Scrum/XP) é igual ao tempo utilizado pelo método XDTv.
- Hipótese nula **(H1)**: as atividades de coleta e especificação de requisitos, planejamento e implementação estabelecidas no método Híbrido (Scrum/XP) são igualmente adequadas às atividades previstas no método XDTv.
- Hipótese alternativa **(H2)**: o método XDTv torna o desenvolvimento de aplicações de TVD mais rápido que o método híbrido (Scrum/XP);
- Hipótese alternativa **(H3)**: o tempo dedicado ao planejamento é menor utilizando o método XDTv;
- Hipótese alternativa **(H4)**: o tempo dedicado à coleta e especificação de requisitos é menor utilizando o método XDTv;
- Hipótese alternativa **(H5)**: o tempo dedicado à implementação é menor utilizando o método XDTv;
- Hipótese alternativa **(H6)**: de forma geral, o método XDTv e seus artefatos são adequados às atividades de planejamento, especificação e implementação de aplicações para TVD.

QUADRO 6.1 – COMPARAÇÃO FUNCIONAL ENTRE OS MÉTODOS HÍBRIDO E XDTV

	Híbrido (Scrum/XP)	XDTv
FOCO	Gestão de projeto	Gestão de projeto
	Implementação	Implementação
	Testes	Testes
	Agilidade e Rapidez	Agilidade e Rapidez
	Escopo: desenvolvimento de software em geral	Coleta de requisitos multimídia
		Escopo: desenvolvimento de software para TVD
PAPÉIS	Cliente/Usuário/Executivo	Cliente/Usuário/Executivo
	Scrum Master	Scrum Master / Tracker
	Tracker	Programador / Testador
	Programador / Testador	Prototipador / Engenheiro de Requisitos
	Gerente produto/Requisitos	
ARTEFATOS	Burn down chart	-
	Product Backlog	Product Backlog
	Sprint Backlog	Sprint Backlog
	Histórias	Artefatos: PATv e MFSM
	Tarefas	Change Request (PATv)
	Módulos funcionais	Módulos funcionais
ATIVIDADES	Iteração / Sprint	Iteração / Sprint
	Daily Sprint	Protótipos de baixa fidelidade
	Stand up meeting	Documentar/modelar primeiro
	Colaboração	Colaboração
	Programação em par	Programação em par
	Envolvimento real do cliente	Envolvimento real com o cliente
	Código compartilhado	Código compartilhado
	Repositório de código único	Repositório de código único
	Implementação contínua	Implementação contínua
	Padronizar código fonte	Padronizar código fonte
	Teste de Aceitação	Teste de aceitação sobre todos os artefatos
	Evoluir apenas código e teste	Evoluir todos os artefatos
	Sprint Retrospective	-
	Definition of Done	-

Fonte: o autor.

Visando realizar uma avaliação sobre tais métodos, o presente capítulo está subdivido nas seguintes seções: **6.1)** objetivos; **6.2)** planejamento; **6.3)** operação e execução; **6.4)** resultados obtidos; **6.5)** analise e discussão dos resultados; e **6.6)** *survey* realizado com profissionais experientes em TVD, contemplando as Fases 6, 7, 8 e 9 do desenho metodológico, apresentado na Seção 1.3 do Capítulo 1.

6.1. DEFINIÇÃO DO OBJETIVO GLOBAL

Além da variável “Tempo”, foi investigada também a “Adequação” sobre os métodos Híbrido (Scrum/XP) e XDTv para o desenvolvimento de aplicações de TVD. O objetivo global do presente experimento é apresentado no Quadro 6.2, onde são descritos os atributos e valores esperados para o experimento.

QUADRO 6.2 – CARACTERÍSTICAS SOBRE A AVALIAÇÃO DO MÉTODO XDTV

ATRIBUTOS	VALORES PARA O EXPERIMENTO
Objetos Investigados	Os métodos de desenvolvimento Híbrido (Scrum/XP) e o método XDTV.
Propósito	Mensurar o “Tempo” e a “Adequação” sobre o desempenho de ambos os métodos, indicando aquele mais rápido e aquele mais adequado ao desenvolvimento de aplicações para TVD.
Em respeito a	Adequação e ao tempo dedicado ao processo de desenvolvimento.
Perspectiva	Time de desenvolvimento de aplicações.
Contexto	Alunos matriculados na disciplina Tópicos Avançados em Engenharia de Software oferecida no décimo período do curso de Engenharia da Computação (UNIVASF).

Fonte: o autor.

Conforme apresentado no Capítulo 1, a presente tese considera que um método adequado ao desenvolvimento de aplicações para TVD deve auxiliar este processo, com ênfase no planejamento e na execução das atividades de Especificação, Desenvolvimento e Testes, apoiando o trato de suas peculiaridades através de artefatos customizados e um ciclo de vida ágil e rápido.

6.2. PLANEJAMENTO

Visando investigar as questões de pesquisa e hipóteses apresentadas na Seção 1.3 (Procedimentos Metodológicos), este segundo experimento adota diferentes perspectivas para realizar a coleta dos resultados, através dos seguintes instrumentos: **a)** questionário formado por questões objetivas e descritivas, disponível no Apêndice K; **b)** entrevista semiestruturada com cada time; e **c)** grupo focal envolvendo todos os times (FLICK, 2009), cujo roteiro está disponível no Apêndice L.

A análise do perfil dos alunos para este segundo experimento foi similar ao primeiro experimento, realizada por meio do levantamento da experiência profissional e acadêmica dos participantes, utilizando o questionário disponível no Apêndice F.

Após a análise de currículo, os participantes foram distribuídos em três diferentes grupos, cada grupo com um grau de conhecimento específico. Os times foram formados por sorteio, selecionando 1 (um) desenvolvedor de cada grupo, viabilizando assim o principio da aleatoriedade visando tornar os times homogêneos (SÁNCHEZ, 2011).

Tendo em vista que o experimento foi realizado com alunos de graduação, este trabalho se preocupou em minimizar a possibilidade da nota da disciplina influenciar nos dados a serem coletados. Desta forma, para minimizar a possibilidade de supervalorização de resultados, além de alertar aos alunos que os dados coletados não interferem na nota, tais notas foram entregues antes do início da fase de coleta dos dados. O Quadro 6.3 sintetiza as variáveis de controle utilizadas neste experimento.

QUADRO 6.3 – VARIÁVEIS INDEPENDENTES: AVALIAÇÃO DO XDTV

Variáveis Independentes	Valores
Quantidade total de desenvolvedores	12
Quantidade de times	Quatro Times (A, B, C e D).
Quantidade de desenvolvedores para cada time	3
Métodos de desenvolvimento adotados	XDTv; Híbrido (Scrum / XP)
Modificação dos times	Não houve modificação. Cada time foi composto pelos mesmos membros durante todo o experimento, desenvolvendo as duas aplicações.
A formação dos times ocorreu em duas etapas	1) Classificação dos desenvolvedores em três diferentes níveis de conhecimento; 2) Através de sorteio, elencar um membro de cada grupo de conhecimento para cada time.
Quantidade de aplicações desenvolvidas	Aplicação A: Concessionária de veículos. Aplicação B: Construtora de imóveis.
Requisitos funcionais	Idênticos para todos os times.
Requisitos não funcionais	Diferentes para cada time. Exemplo: posição, tamanho, tempo de surgimento e finalização das mídias.
Implementação dos requisitos	Foi implementado um requisito por vez, de forma serial.
Realização das atividades	As atividades abaixo foram realizadas em equipe, com a participação de todos os membros. a) Reunião com o cliente; b) Coleta e especificação de requisitos; e c) Planejamento.
Ordem de desenvolvimento das aplicações	Definida através de sorteio: Aplicação A ou aplicação B.
Prazo de entrega para cada aplicação	3 semanas.
Cliente	O autor da tese.
Reuniões com o cliente	Tempo fixo e inflexível para todos os times: 6 reuniões de 30 minutos.
Quantidade de iterações (ciclo)	3 iterações.
Duração de cada iteração	Uma semana.
Ambiente de desenvolvimento	Eclipse Helios; Plugins: RSE, Lua, NCL; Maquina virtual VMware Player; Ginga Fedora.
Conteúdo multimídia utilizado (vídeos, figuras e botões)	Disponibilizado previamente; Idêntico para todos os times.

Fonte: o autor.

As variáveis independentes apresentadas no Quadro 6.3 foram necessárias para controlar o experimento, oferecendo condições iguais para todos os times, evitando que estas variáveis influenciem e alterem os valores referentes à “Adequação” e ao “Tempo”, alvos deste experimento.

O presente trabalho fixou igualmente o tempo de participação do cliente para cada time, pois, de acordo com o relatório CHAOS (2010) apresentado no Capítulo 1, a presença do cliente/usuário é um fator relevante e está

diretamente relacionada com o sucesso do projeto, pois quando esta relação é falha, há alto índice de fracasso.

Cada time desenvolveu duas aplicações. A **Aplicação A** tem como objetivo agregar valor à campanha publicitária de uma concessionária de veículos importados, apresentando as marcas e os diferentes modelos comercializados. Os requisitos funcionais referentes a esta aplicação estão disponíveis no Apêndice M. A **Aplicação B** tem como objetivo agregar valor à campanha publicitária de uma construtora, apresentando três prédios e seus tipos de apartamentos. Os requisitos funcionais referentes a esta aplicação estão disponíveis no Apêndice N.

A fim de evitar resultados tendenciosos, foi aplicado o princípio da aleatoriedade. Dessa forma, foram definidas através de sorteio as seguintes variáveis:

- Os membros de cada time;
- O primeiro método utilizado por cada time;
- A primeira aplicação a ser desenvolvida.

Dessa forma, foram distribuídos 3 desenvolvedores em cada time (**A**, **B**, **C** e **D**). O experimento foi delineado através da técnica denominada Quadrado Latino, do tipo (2x2), conforme apresentado em SÁNCHEZ (2011). O quadrado (2x2) viabilizou estruturação do experimento, auxiliando na igualdade de oportunidades, permitindo a troca de métodos entre os times (A e B) e (C e D), tornando o impacto do aprendizado igual para ambos os métodos.

A primeira aplicação sorteada (Aplicação 1) foi desenvolvida paralelamente por todos os times. Para o desenvolvimento desta aplicação, foram sorteados os métodos investigados, de modo que, cada Quadrado Latino contenha dois métodos distintos. Para o desenvolvimento da segunda aplicação, os times adotaram automaticamente o método concorrente (oposto). Dessa forma, um time de cada quadrado adotou inicialmente o método híbrido (XP junto com Scrum), e simultaneamente, o outro time que formou o mesmo quadrado adotou o método XDTv, conforme exibido no Quadro 6.4.

QUADRO 6.4 – PLANEJAMENTO DO QUADRADO LATINO PARA OS QUATRO TIMES

	Time **A**	Time **B**
Qtd. membros:	**3**	**3**
Aplicação 1	Híbrido (Scrum/XP)	XDTv
Aplicação 2	XDTv	Híbrido (Scrum/XP)

	Time **C**	Time **D**
Qtd. membros:	**3**	**3**
Aplicação 1	Híbrido (Scrum/XP)	XDTv
Aplicação 2	XDTv	Híbrido (Scrum/XP)

Fonte: o autor.

Visando reduzir o risco de compartilhamento de código entre os times, para ambas as aplicações, os seguintes requisitos não funcionais foram diferentes para cada time: localização (região), tamanho e o tempo de transição e exibição das mídias. Desta forma, todos os times desenvolveram os mesmos requisitos funcionais, porém, com características específicas tornando cada aplicação única, sem alterar o grau de dificuldade do desenvolvimento da aplicação.

Como um dos objetivos deste experimento é mensurar o tempo do desenvolvimento das aplicações, todos os objetos de mídia a serem utilizados foram pré-definidos e fornecidos a todos os times, para que o tempo dedicado à elaboração do *design* da aplicação não interfira no resultado final.

Este experimento mensurou a quantidade de horas trabalhadas e o nível de adequação de ambos os métodos durante as atividades de especificação, planejamento e implementação. As variáveis dependentes são apresentadas no Quadro 6.5.

QUADRO 6.5 – VARIÁVEIS DEPENDENTES DO SEGUNDO EXPERIMENTO

Atividades	Variável Dependente
Especificação	Tempo dedicado à Elicitação e Especificação de Requisitos.
	Método mais adequado à Elicitação e Especificação de Requisitos.
	Método que apresenta os artefatos mais adequados ao contexto de TVD.
Planejamento	Tempo dedicado ao Planejamento do time.
	Método mais adequado ao Planejamento.
	Adequação do ciclo de vida do método XDTv.
	Adequação do modelo de processo do método XDTv.
Implementação	Tempo dedicado à Implementação de requisitos em série.
	Método mais adequado à Implementação.
Teste	Adequação das atividades de teste.

Fonte: o autor.

Os resultados foram coletados de forma quantitativa e qualitativa. Quantitativa, no que se refere às horas trabalhadas. Qualitativa, no que se refere à análise subjetiva da adequação dos métodos investigados, suas atividades e artefatos. Visando reduzir a possibilidade de falhas durante a coleta das horas trabalhadas foi utilizada a ferramenta ManicTime[16], livre, portátil, instalada no *pendrive* de cada desenvolvedor. Tal ferramenta tem o propósito de monitorar o funcionamento do computador em tempo real, considerando apenas as aplicações ativas (em execução), contabilizando o tempo real que cada aplicação é realmente utilizada pelo usuário.

6.3. OPERAÇÃO E EXECUÇÃO

Os participantes do experimento não possuíam experiência sobre o desenvolvimento de aplicações para TVD. Diante disso, foram dedicadas 38 horas de aulas teóricas e práticas, ministradas presencialmente pelo autor da presente tese, durante dois (2) meses, além de diversos exercícios de fixação e atividade extraclasse voltadas, exclusivamente, para treinamento sobre o desenvolvimento de aplicações para TVD. Durante o treinamento foi abordada a linguagem NCL, Lua, IDE Eclipse versão Helios, a configuração do ambiente de desenvolvimento e simulação de aplicações através da máquina virtual VMware Player emulando o Ginga Fedora.

Antes do início do desenvolvimento de cada aplicação foram dedicadas quatro (4) horas adicionais de treinamento para os métodos adotados. Este treinamento ocorreu isoladamente para cada método, sendo que cada time só participou do treinamento do método que iria adotar. Antes do desenvolvimento da segunda aplicação, foi realizada uma nova rodada de treinamento onde cada time foi treinado para utilizar o método oposto.

Visando auxiliar no planejamento e desenvolvimento dos projetos, foi entregue para todos os participantes um *kit* contendo uma pasta com folhas de papel pautado, canetas coloridas, lápis, borracha e os formulários para preenchimento das horas dedicadas às reuniões do time.

[16] http://www.manictime.com

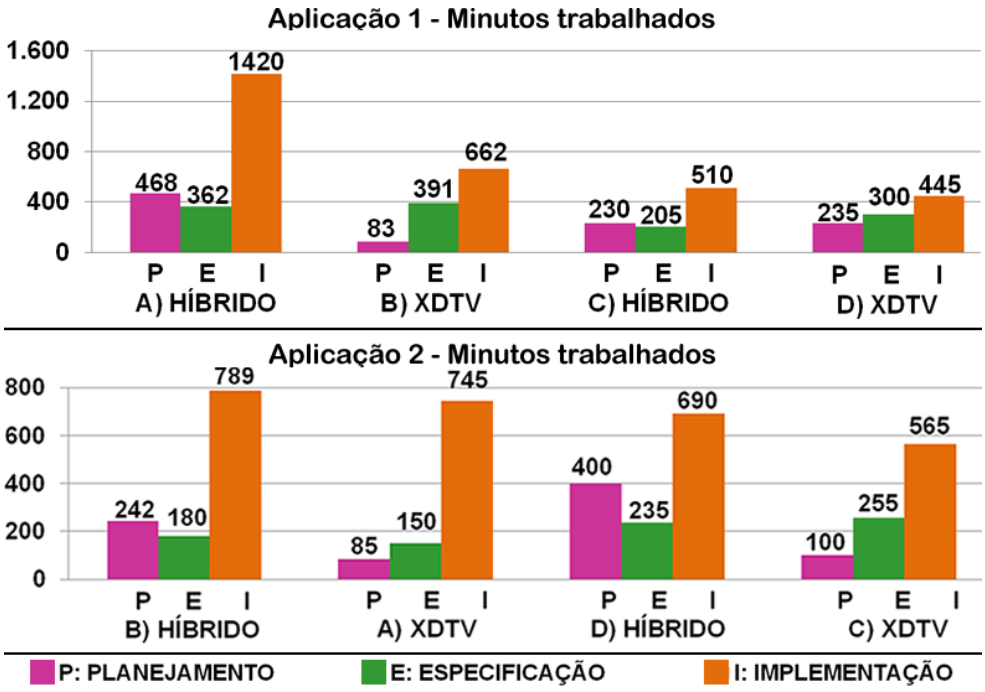
A ferramenta ManicTime monitorou a utilização das ferramentas utilizadas no ambiente de desenvolvimento, mais precisamente, a IDE Eclipse Helios e o VMware Player (responsável por emular o Ginga Fedora). Foram adotados também dois modelos de formulários individuais em papel, como uma forma complementar para a coleta das horas trabalhadas, evitando uma possível perda das informações armazenadas no *pendrive* através da ferramenta ManicTime. O formulário disponível no Apêndice G foi utilizado pelos times durante a experimentação do método híbrido (Scrum/XP). O formulário disponível no Apêndice H foi utilizado pelos times durante a experimentação do método XDTv.

Foi utilizado também um formulário específico para reuniões de planejamento, disponível no Apêndice I para ser preenchido pelos times que adotaram o método híbrido (Scrum/XP) e no Apêndice J para ser preenchido pelos times que adotaram o método XDTv.

6.4. RESULTADOS OBTIDOS

Os quatro times, representados pelas letras: **A**, **B**, **C** e **D,** implementaram todas as funcionalidades solicitadas. A **Figura 6.1** apresenta os valores absolutos totais referentes ao tempo (em minutos) dedicado às atividades de Planejamento (**P**), coleta e Especificação de requisitos (**E**) e Implementação (**I**), para o desenvolvimento de cada aplicação.

Figura 6.1 – Tempo dedicado a cada atividade do desenvolvimento



Fonte: o autor.

Vale destacar que além do tempo apresentado na **Figura 6.1**, cada time realizou igualmente seis reuniões com o cliente, totalizando 180 minutos, para a coleta de requisitos de cada aplicação.

Os resultados correspondentes à variável qualitativa “Adequação”, foram coletados em um questionário objetivo, disponível no Apêndice K, seguido por um questionário descritivo, disponível no Apêndice L, pela realização de entrevistas em grupo com cada time, e, por fim, pela realização de um grupo focal (Flick, 2009).

Questões objetivas

O questionário objetivo (disponível no Apêndice K) era composto por oito questões para avaliar os critérios sobre cada método. Dentre estas, as sete primeiras possuíam cinco alternativas de resposta, conforme apresentado no primeiro experimento (Capítulo 4), seguindo a escala de Rensis Likert (BERTRAM, 2009), sendo: (1) Muito inadequado; (2) Inadequado; (3) Regular; (4) Adequado; e (5) Muito adequado ao desenvolvimento de aplicações para TVD. Visando obter maior granularidade, as alternativas da última (oitava)

questão objetiva, apresentava as alternativas de resposta através de uma escala que variou de: “Muito inadequado” (0) até “Muito adequado” (10). Caso houvesse uma avaliação ótima (com todas as respostas: “Muito adequada”), ter-se-ia 45 pontos.

Quadro 6.6 – Pontuação média obtida através do questionário objetivo.

Questionário Objetivo (Apêndice K)	Média	
	XDTv	Scrum/XP
C1) Velocidade do desenvolvimento	4,0	3,0
C2) Adequação dos artefatos	4,0	2,5
C3) Adequação da Coleta de Requisitos multimídia	4,0	3,0
C4) Adequação da atividade de Especificação de Requisitos	4,0	3,5
C5) Adequação da atividade de Planejamento	4,0	4,0
C6) Adequação da atividade de Implementação	4,5	3,0
C7) Adequação da atividade do Teste de aceitação	4,0	3,5
C8) Adequação do método ao desenvolvimento de aplicações para TVD	9,18	6,36
Soma:	**37,68**	**28,86**

Fonte: o autor.

O Quadro 6.6 apresenta a média das avaliações dos times sobre cada método. Ao considerar a soma das médias de todas as oito questões foram obtidos os seguintes valores: XDTv = 37,68 pontos e Híbrido = 28,86 pontos. Caso houvesse uma avaliação mínima (com todas as respostas: “Muito inadequado”) para as oito questões objetivas, com escala de 1 até 5, ter-se-ia 7 pontos.

Questionário descritivo

O questionário descritivo, disponível no Apêndice L, foi aplicado individualmente para cada participante, no entanto, entre os doze participantes, um optou por não respondê-lo. Este questionário era composto por seis perguntas abertas, viabilizando coletar informações variadas, subjetivas e fornecendo liberdade aos participantes para expressarem satisfação ou insatisfação sobre tais temas. Dessa forma, os participantes dissertaram sobre a adequação dos métodos híbrido (Scrum/XP) e do método XDTv. As perguntas são apresentadas abaixo:

P1) Qual dos métodos melhor contribuiu para as atividades de coleta e especificação de requisitos?

P2) Qual dos métodos melhor contribuiu para as atividades de planejamento?

P3) Qual dos métodos melhor contribuiu para as atividades de implementação?

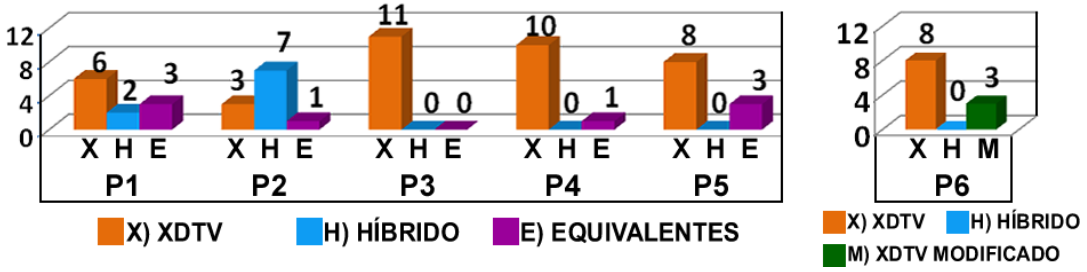
P4) Qual método possui os artefatos que mais contribuíram no processo de desenvolvimento?

P5) De forma geral, algum dos métodos contribuiu mais para o desenvolvimento? Qual?

P6) Qual dos métodos você adotaria para desenvolver uma aplicação para TVD, hibrido (Scrum/XP) ou (XDTV)?

As seis perguntas supracitadas visam avaliar as variáveis dependentes, através da opinião individual do participante. Para cada pergunta foi realizada uma síntese através da técnica de palavras chaves (FLICK, 2009), com objetivo de sintetizar o levantamento daquele método que mais contribuiu com o processo de desenvolvimento. Dessa forma, os resultados das perguntas (**P1**, **P2**, **P3**, **P4** e **P5**) foram sintetizados nas seguintes respostas: “XDTv”, “Híbrido” ou “Equivalentes”, indicando a preferência dos participantes pelos métodos adotados. Foram classificados como “equivalentes” as respostas que avaliaram ambos os métodos como iguais. As respostas da pergunta **P6** foram sintetizadas em: “XDTv”, “Híbrido” ou “XDTv Modificado”. Foram classificados como “XDTv modificado” aquelas respostas que julgaram o método XDTv mais adequado que o método híbrido, porém, mediante adaptações. O resultado é apresentado na **Figura 6.2**.

Figura 6.2 – Questionário 3: adequação dos métodos XDTv e Híbrido



Fonte: o autor.

Com base nos resultados obtidos pelos questionários, foram realizadas entrevistas semiestruturadas em grupo, e, por fim, a realização de um grupo focal, visando realizar um debate criativo em busca de pontos de melhoria, aumentando assim as perspectivas investigadas. O roteiro de tais entrevistas está disponível no Apêndice P.

Entrevistas e grupo focal

Os resultados obtidos através do grupo focal indicaram que a atividade de Planejamento do método XDTv pode ser aprimorada com a adição de técnicas voltadas ao gerenciamento das atividades. A principal contribuição obtida através das entrevistas e do grupo focal foi a sugestão da utilização de alguma técnica similar ao Kanban (COCCO et al., 2011) para auxiliar as atividades de planejamento do método XDTv. Os participantes reforçaram que a programação em par traz benefícios para o time de desenvolvimento, principalmente para a comunicação, cobrança das obrigações e motivação do trabalho em equipe.

6.5. SURVEY COM PROFISSIONAIS EXPERIENTES EM TVD

O objetivo deste *survey* foi coletar a opinião de profissionais atuantes na indústria de desenvolvimento de aplicações para TVD sobre o grau de adequação do método XDTv para o desenvolvimento dessas aplicações. A variável "Tempo" de desenvolvimento estava fora do escopo deste *survey*, pois, para isso, seria necessária a participação de, pelo menos mais 2 profissionais, por um período duas vezes superior ao concedido pela empresa.

Este *survey* foi aplicado no Centro de Estudos e Sistemas Avançados do Recife (C.E.S.A.R.) com duração de uma semana (5 dias corridos). Quatro profissionais participaram do *survey*, sendo: 2 mestres, 1 graduado e 1 que está finalizando sua graduação. Tais participantes possuem, em média, 2 anos e 9 meses de experiência no desenvolvimento de aplicações para TVD.

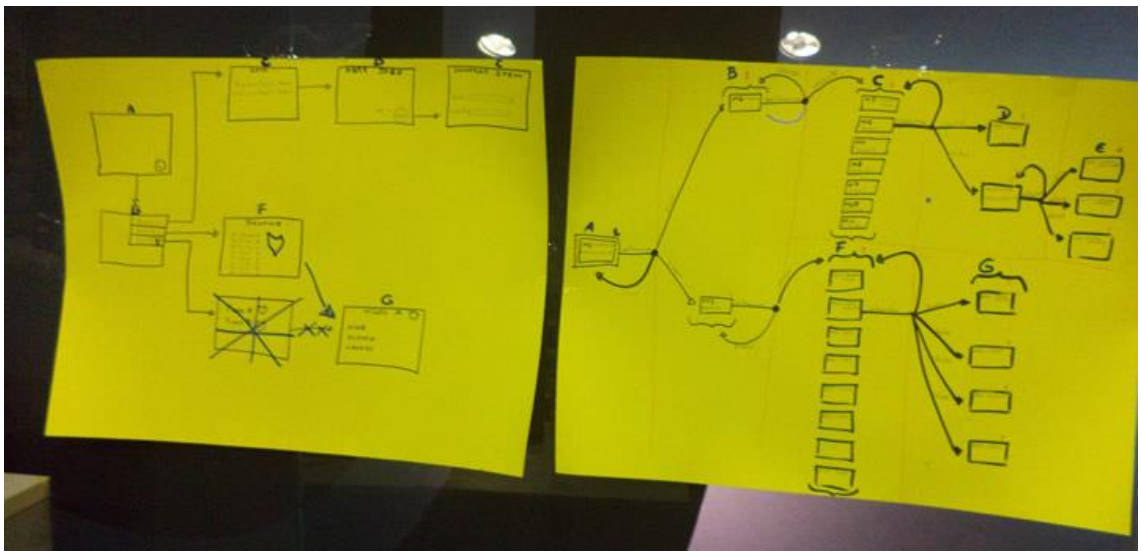
Foi disponibilizado para todos os participantes: papel, canetas coloridas, lápis, borracha, pincel atômico para quadro branco e um guia básico[17] sobre o método XDTv, para eventuais consultas.

[17] http://www.univasf.edu.br/~mario.godoy/xdtv/

No primeiro dia foi apresentado aos profissionais o método XDTv, contemplando: objetivos, artefatos, papéis, e modelo de processo e ciclo de vida.

Após esta apresentação, visando oferecer também uma experiência prática sobre o método XDTv, foi solicitado que o time desenvolvesse uma aplicação do tipo arquibancada virtual de futebol, com funcionalidades similares à aplicação solicitada no primeiro experimento. Dessa forma, os participantes realizaram todas as atividades previstas no ciclo de vida do método XDTv, apresentado na Figura 5.10. A Figura 6.3 apresenta os artefatos PATv e MFSM desenvolvidos neste *survey*.

Figura 6.3 – Artefatos desenvolvidos durante o *survey* realizado no C.E.S.A.R.



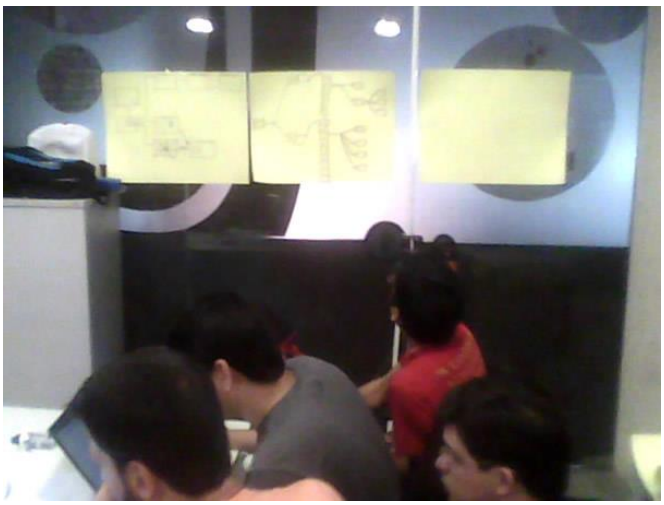
Fonte: o autor.

O autor do presente trabalho participou ativamente do desenvolvimento da aplicação solicitada, assumindo o papel de Cliente. Assim como nos experimentos anteriores, as mídias utilizadas no desenvolvimento da aplicação foram disponibilizadas para os times.

No decorrer do desenvolvimento da aplicação solicitada, o autor da presente pesquisa adotou também a técnica de observação. Por meio dela, pode-se notar que durante a definição dos pares envolvidos na implementação (*pair programming*), houve certa disputa na definição do programador que atuaria ativamente na implementação.

Outro ponto coletado através da observação foi a localização do PATv e do MFSM durante o desenvolvimento da aplicação. A princípio, tais artefatos foram deixados sobre a bancada de trabalho, dificultando sua visualização e manipulação por todos os membros. Visando tornar as informações dos artefatos mais acessíveis, estes foram fixados na parede, facilitando sua utilização. O desenvolvimento deste *survey* pode ser observado na **Figura 6.4**.

Figura 6.4 – Implementação (programação em par) realizada durante o *survey* no C.E.S.A.R.



Fonte: o autor.

Ao fim da implementação dos requisitos, foi aplicado um questionário objetivo, disponível no Apêndice Q, sobre a adequação do método XDTv, cujas alternativas de resposta variam de zero (0) “Totalmente inadequado” até dez (10) “Totalmente adequado”. As médias da pontuação das respostas dos quatro usuários são apresentadas no Quadro 6.7.

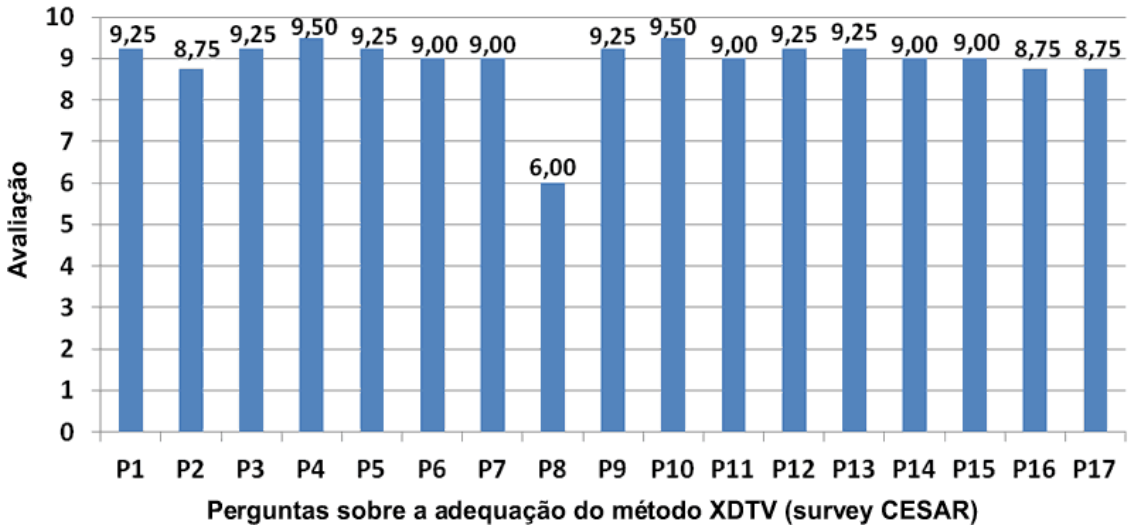
Quadro 6.7 – Questionário e média de notas atribuídas ao *survey* (C.E.S.A.R.).

PERGUNTA	MÉDIA
1) Avalie o quão adequado é o método XDTv para a atividade de COLETA de requisitos:	**9,25**
2) Avalie o quão adequado é o método XDTv para a atividade de ESPECIFICAÇÃO de requisitos:	**8,75**
3) Avalie o quão adequado é o método XDTv para a atividade de PLANEJAMENTO:	**9,25**
4) Avalie o quão adequado é o método XDTv para a atividade de IMPLEMENTAÇÃO:	**9,50**
5) Avalie o quão adequado é o CICLO DE VIDA do método XDTv:	**9,25**
6) Avalie o quão adequado são os TESTES DE ACEITAÇÃO no ciclo de vida do método XDTv:	**9,00**
7) Avalie o quão adequado é o MODELO DE PROCESSO do método XDTv:	**9,00**
8) Avalie o quão adequado é a PROGRAMAÇÃO em PAR para o desenvolvimento de aplicações de TVD:	**6,00**
9) Avalie o quão adequado é o artefato Protótipo de Aplicações para TV(PATV):	**9,25**
10) Avalie o quão adequado é o artefato Modelo de Fluxo e Sincronismo de Mídia (MFSM):	**9,50**
11) Avalie o grau de adequação do papel do LIDER do projeto e as atividades a ele atribuídas:	**9,00**
12) Avalie o grau de adequação do papel do PROTOTIPADOR e as atividades a ele atribuídas:	**9,25**
13) Avalie o grau de adequação de papel do PROGRAMADOR e as atividades a ele atribuídas:	**9,25**
14) Avalie o grau de adequação do papel do CLIENTE e as ATIVIDADES a ele atribuídas:	**9,00**
15) De forma geral, avalie o grau de adequação do método XDTV para o desenvolvimento de aplicações de TVD:	**9,00**
16) Avalie o quão o método XDTV contribui para tornar o desenvolvimento mais rápido:	**8,75**
17) Avalie o grau de adequação da filosofia ágil (independente do método adotado) para o desenvolvimento de aplicações para TVD:	**8,75**

Fonte: o autor.

A menor avaliação foi referente à Pergunta 8 (adequação da programação em par), que atingiu média de seis (6,0) pontos. As respostas para as demais perguntas variaram entre 8,75 e 9,5, conforme apresentado no **Gráfico 6.1**.

Gráfico 6.1 – Adequação do método XDTv segundo *survey* realizado no CESAR



Fonte: o autor.

Conforme apontado pela técnica de observação, alguns profissionais experientes podem ficar impacientes atuando como observador/revisor de código. Por outro lado, estudantes ou profissionais com pouca experiência se sentem mais amparados ao compartilhar dúvidas e decisões sobre a implementação com um parceiro.

Depois do preenchimento do questionário, visando coletar prontos de melhoria sobre o método XDTv, foi realizado um grupo focal. A principal sugestão de melhoria foi a criação de uma nova representação gráfica para o MFSM, com o objetivo de representar diversas mídias de forma mais genérica, ocupando menos espaço no diagrama.

6.6. ANALISE DOS RESULTADOS ATRAVÉS DO MÉTODO MULTICRITÉRIO (PROMETHEE II)

Conforme apresentado anteriormente, a avaliação do método XDTv ocorreu através da implementação de oito aplicações, sendo 4 aplicações desenvolvidas através do método Híbrido e 4 aplicações desenvolvidas através do método XDTv. Para realização da análise multicritério, foi calculada a média dos tempos inerentes a cada método. O cálculo da média é uma técnica frequentemente utilizada para análise e síntese de grupo, conforme apresentado na Seção 1.4.

Os critérios quantitativos (TP, TE e TI) correspondem ao tempo (em minutos) dedicado à atividade de Planejamento, Especificação e Implementação, respectivamente. Dessa forma, a média dos valores obtidos para cada método é apresentado no Quadro 6.8:

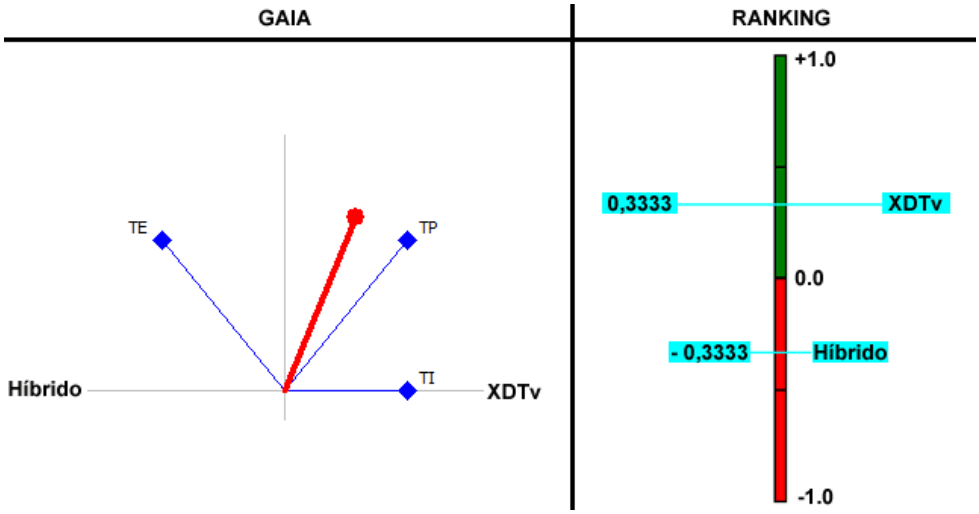
Quadro 6.8 – Tempo médio (em minutos) referente a cada método

			MEDIA			Soma das médias
Aplicação	Método	Times	TP	TE	TI	
1 e 2	Híbrido	A;C;B;D	335,00	245,50	852,25	1.432,75
1 e 2	XDTv	B;D;A;C	125,75	274,00	604,25	1.004,00

Fonte: o autor.

Para realizar a análise estatística dos dados quantitativos e qualitativos, foi adotada a Metodologia Multicritério de Apoio à Decisão, conforme apresentado no Capítulo 1, Seção 1.5 (ENSSLIN et al., 2010), através do software Visual PROMETHEE II[18], versão acadêmica 1.4. As médias (TP, TE e TI) apresentadas no Quadro 6.8 foram utilizadas como critérios de decisão para a realização da análise dos dados quantitativos de forma isolada (a princípio), ou seja, sem interferência dos dados qualitativos, conforme apresentado na **Figura 6.5**.

Figura 6.5 – Análise multicritério sobre os dados quantitativos (tempo)



Fonte: o autor.

[18] http://www.promethee-gaia.com/

Foram atribuídos pesos iguais (peso 1) para os critério TP, TE e TI. Apesar do critério (TE) ser favorável ao método Híbrido, os outros dois critérios são favoráveis ao método XDTv, com destaque para o TI, que apresenta forte favorecimento ao método XDTv. Considerando tal cenário, o eixo decisor indica preferência pelo método XDTv. No entanto, considerar apenas os critérios quantitativos não atende à necessidade de avaliar a "adequação" de ambos os métodos. Por esse motivo, além dos dados quantitativos (tempo), foi realizada uma nova análise considerando também os critérios qualitativos, conforme apresentados no Quadro 6.9.

Quadro 6.9 – Critérios qualitativos e quantitativos.

Critérios	Tipo	Valores		Cenário (peso)			Impacto no desempenho geral
		XDTv	Híbrido Scrum/XP	A	B	C	
TP) Tempo do Planejamento	Quantitativo	125,75	335,00	1	2	1	Negativo
TE) Tempo da Coleta e Especificação de Requisitos	Quantitativo	274,00	245,50	1	2	1	Negativo
TI) Tempo da Implementação	Quantitativo	604,25	852,25	1	2	1	Negativo
C1) Velocidade do desenvolvimento	Qualitativo	4,0	3,0	1	1	2	Positivo
C2) Adequação dos artefatos	Qualitativo	4,0	2,5	1	1	2	Positivo
C3) Adequação da Coleta de Requisitos multimídia	Qualitativo	4,0	3,0	1	1	2	Positivo
C4) Adequação da atividade de Especificação de Requisitos	Qualitativo	4,0	3,5	1	1	2	Positivo
C5) Adequação da atividade de Planejamento	Qualitativo	4,0	4,0	1	1	2	Positivo
C6) Adequação da atividade de Implementação	Qualitativo	4,5	3,0	1	1	2	Positivo
C7) Adequação da atividade do Teste de aceitação	Qualitativo	4,0	3,5	1	1	2	Positivo
C8) Adequação ao desenvolvimento de aplicações para TVD	Qualitativo	9,18	6,36	1	1	2	Positivo

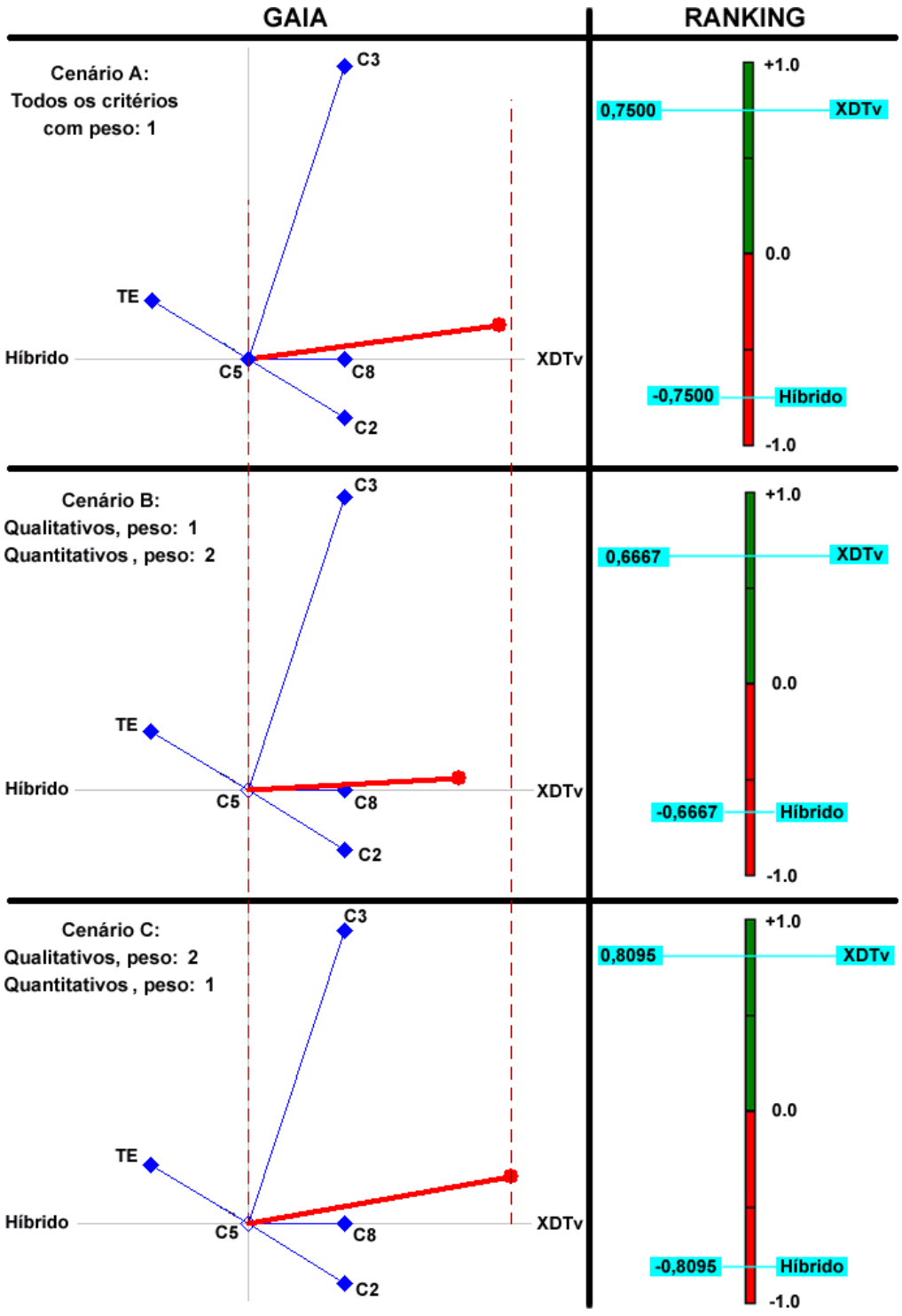
Fonte: o autor.

Os critérios qualitativos (C1, C2, C3, C4, C5, C6 e C7) apresentados no Quadro 6.9 foram coletados através do questionário disponível no Apêndice K, aplicado com os times de desenvolvimento, cujas opções de respostas obedeciam à escala de Likert, com os seguintes valores: (1) muito ruim; (2)

ruim; (3) regular; (4) bom; e (5) muito bom. Visando oferecer maior granularidade para a avaliação geral dos métodos, a escala utilizada pelo critério C8 variou de: zero (0) “Totalmente inadequado”, até dez (10) “Totalmente adequado”.

O gráfico do plano Gaia, apresentado na Figura 6.6, mantém as informações preservadas em 100%, onde o eixo decisor indica que, mesmo com a variação dos pesos apresentada no Quadro 6.9, configurando três cenários (A, B e C), o método XDTv apresenta os critérios mais adequados que o método Híbrido (Scrum/XP), apesar do critério Tempo da Especificação (TE) de requisitos ser maior no método XDTv.

Figura 6.6 – Análise dos resultados quantitativos e qualitativos através do plano Gaia



Fonte: o autor.

A classificação (*ranking*) entre ambos os métodos apresenta o quão diferentes estes foram avaliados: quanto mais próximo de zero (0.0) mais semelhantes são, e, quanto mais próximo de um (+1.0) estiver o método,

melhor a sua adequação. Dessa forma, os resultados apontam que o método XDTv é mais adequado nos três cenários analisados, ou seja, mais o XDTv sobreclassifica o Híbrido. As considerações finais sobre os resultados obtidos neste experimento são apresentadas no Capítulo 7.

CAPÍTULO

7.

7. CONCLUSÃO E TRABALHOS FUTUROS

"Teus atos possuem a voz que fala
mais alto que todas tuas palavras."
Chico Xavier

As peculiaridades inerentes às aplicações de TVD implicam em necessidades específicas durante seu processo de desenvolvimento, demandando por um método customizado com o objetivo de auxiliar o time de desenvolvimento de forma adequada e rápida.

Durante os dois experimentos controlados, realizados pela presente tese, no total, foram formados sete times (sem contar com o *survey*) de desenvolvimento, sendo três (3) para o primeiro experimento e quatro (4) para o segundo experimento, totalizando vinte e um (21) desenvolvedores atuando diretamente na implementação de onze (11) aplicações de TVD.

Ao fim do primeiro experimento, sobre a investigação dos métodos ágeis Scrum, XP e Híbrido (Scrum/XP), foram apresentaram indícios positivos da adequação de tais métodos sobre o desenvolvimento de aplicações de TVD. Entre eles, o método Híbrido lidera os resultados obtidos. No entanto, tais métodos apresentam pontos de melhoria que podem ser trabalhados para atender às peculiaridades que configuram o ambiente de TVD. Diante disso, a presente pesquisa especificou o método denominado *eXtreme Digital Television* (XDTv), que compartilha da filosofia ágil para auxiliar de forma

rápida e adequada ao tratamento de tais peculiaridades. O método XDTv adota o modelo de processo Iterativo, apresentando em seu ciclo de vida atividades e artefatos próprios, exclusivamente projetados para atender às especificidades das aplicações de TVD.

Ao fim do segundo experimento, que realizou a investigação das variáveis “tempo” e “adequação” sobre os métodos Híbrido (Scrum/XP) e XDTv, foram apresentados resultados que apontam o método XDTv como melhor alternativa nos três cenários investigados, ou seja, atribuindo peso: a) igual para ambas as variáveis; b) maior para a variável “tempo”; e c) maior para a variável “adequação”.

Entre os resultados quantitativos (horas trabalhadas) apresentados no Capítulo 6, Figura 6.1, vale ressaltar que as diferenças entre as médias obtidas pelos times apontam que, durante o desenvolvimento da Aplicação 1, os times que adotaram o método híbrido (Scrum/XP) demoraram 51% (540 minutos) a mais que os times que adotaram o método XDTv. Durante o desenvolvimento da Aplicação 2, os times que adotaram o método híbrido demoraram 33% (318 minutos) a mais que os times que adotaram o método XDTv.

Ao analisar as horas trabalhadas individualmente para cada time, considerando ambas as aplicações, todos os times obtiveram melhor eficiência ao adotar o método XDTv. Dessa forma, todos os times alcançaram maior rapidez no desenvolvimento ao adotar XDTv. O Time C obteve melhor desempenho em ambas as aplicações, indicando maior eficiência dos seus membros com relação aos demais.

A média do tempo dedicado ao Planejamento e à Implementação através do método XDTv foi menor para ambas as aplicações desenvolvidas. No entanto, a média do tempo da atividade de coleta e especificação de requisitos foi 11,60**%** (28,5 minutos) maior quando adotado o método XDTv. Já era esperada uma dedicação maior para estas atividades devido à forte utilização de protótipos de baixo nível de fidelidade PATv e o diagrama do MFSM. Porém, o tempo dedicado à elaboração de tais artefatos viabilizou que o método XDTv atingisse um desempenho melhor na etapa de Planejamento e Implementação, pois o tempo dedicado à atividade de implementação quando

adotado o método Híbrido, foi, em média, 41,04% (248 minutos) maior que o método XDTv. Segundo relato dos participantes, através da entrevista semiestruturada, os artefatos PATV e MFSM do método XDTV ofereceram um significativo auxílio para a realização da implementação da aplicação.

Os resultados apresentados no Capítulo 6, Figura 6.2 apontam que o método XDTv obteve melhor desempenho e adequação sobre as atividades Coleta de requisitos, Implementação e Artefatos. Vale destacar que a avaliação da atividade de Implementação foi unânime em prol do método XDTv, refutando as Hipóteses Nulas H0 e H1, e corroborando as Hipóteses Alternativas H2, H4 e H5. No entanto, 63,6% dos participantes julgaram as atividades de Planejamento (P2) do método híbrido (Scrum/XP) mais adequadas, refutando a Hipótese Alternativa H3, referentes ao segundo experimento.

Para 72,7% dos participantes, o método XDTv foi aquele que mais contribuiu para o desenvolvimento. Essa parcela de desenvolvedores adotaria o método XDTv para desenvolver aplicações para a indústria. O restante (27,3%) identificou que ambos os métodos são equivalentes, e adotariam o método XDTv com algumas modificações para desenvolver para a indústria. Nenhum dos participantes julgou o método híbrido (Scrum/XP) mais apropriado.

Por fim, a média referente à adequação do método híbrido (Scrum/XP) foi 6,36. No entanto, a média referente à adequação do método XDTv foi 9,18, o que corroborou a Hipótese Alternativa H6 e reforçou as avaliações das hipóteses citadas anteriormente.

Os resultados obtidos pelos times de desenvolvimento do segundo experimento, formados por alunos de graduação e os resultados obtidos através do *survey* com profissionais experientes, apresentaram duas diferenças. A primeira está relacionada à programação em par (*pair programming*). Esta atividade foi bem avaliada segundo o experimento com alunos de graduação, naturalmente, devido à pouca experiência dos mesmos, pois esta atividade pode oferecer mais apoio aos envolvidos. Já para os profissionais experientes, dentre todas as avaliações, está foi a mais baixa seis

(6) pontos, pouco acima da média regular cinco (5) pontos, em uma escala de zero (0) até dez (10) pontos. Tal avaliação foi reforçada através da observação do pesquisador, que constatou certa ansiedade e disputa pelo papel ativo na codificação, e descontentamento daquele que permaneceu na função de auxiliar (*backup*). Dessa forma, a adoção da atividade de programação em par deve ser facultativa ao time de desenvolvimento. A segunda diferença está relacionada à atividade de planejamento, onde a maior parte, sete (7) dos doze (12) participantes do experimento, classificou o método Híbrido (Scrum/XP) como mais adequado que o método XDTv. O grupo focal realizado com os times apontou que tal avaliação pode ter sido influenciada negativamente devido à falta de um local fixo, específico para o desenvolvimento do projeto, dificultando a visualização e manutenção dos artefatos PATv e MFSM. Além disso, os sete (7) participantes mudariam sua escolha, classificando o método XDTV como mais adequado, caso fosse incorporada às suas atividades a técnica de Kanban. Já os profissionais experientes em TVD avaliaram positivamente (9,25), atribuindo a terceira nota mais alta para a atividade de planejamento. Tal avaliação pode ser sido influenciada pelo local fixo e a disponibilização da infraestrutura necessária para as reuniões.

Os resultados qualitativos coletados através do *survey* com profissionais experientes no desenvolvimento de aplicações para TVD confirmaram os indícios levantados pelos questionários aplicados no segundo experimento. Considerando todas as 17 questões realizadas no *survey*, a avaliação do método XDTv foi positiva, alcançou média de 8,93 pontos. Considerando a prática de programação em par opcional, desconsiderando a pontuação referente a Questão 8 do *survey*, o método XDTv mostra-se ainda mais positivo segundo os profissionais participantes, sendo atribuído à sua adequação a média de 9,11 pontos, resultado que reforça a adequação do método XDTV para o desenvolvimento de aplicações de TVD.

Vale ressaltar que ao adotar o método XDTv, é necessário deixar seus artefatos visíveis e acessíveis, eliminando qualquer obstáculo visual entre os artefatos e o time, obtendo maior facilidade em sua utilização, conforme

observado durante o *survey,* após a fixação do PATv e do MSFS na parede da sala onde trabalhava o time de desenvolvimento.

O método XDTv não se restringe ao sistema ISDB-Tb, este método pode ser aplicado para os demais padrões de TVD como o ATSC ou DVB. Além disso, os artefatos PATv e MFSM não se restringem ao desenvolvimento de aplicações de TVD, estes podem ser utilizados em outros contextos que necessitem da representação visual para representar iteração do usuário e os eventos da linguagem de programação, pois auxiliam o time de desenvolvimento na atividade de coleta e especificação requisitos, bem como, na comunicação do time com o cliente e documentação de aplicações. Porém, o desempenho do método XDTv e seus artefatos em outros contextos ainda demanda por avaliações.

7.1. CONTRIBUIÇÕES

As principais contribuições deste trabalho são: o levantamento das peculiaridades das aplicações para TVD; os possíveis pontos de melhoria no processo de desenvolvimento de aplicações para TVD; a avaliação dos métodos ágeis (Scrum, XP e Híbrido (Scrum/XP) realizada através de experimentos controlados e replicáveis, realizados em busca do método que melhor atende às suas peculiaridades. O método de desenvolvimento XDTv, seus artefatos e o ciclo de vida do método, completam as principais contribuições da presente tese.

O presente trabalho pode ser classificado como uma contribuição ainda inexistente na comunidade científica. Sua originalidade pode ser encontrada através dos seguintes pontos:

- levantamento das peculiaridades inerentes ao desenvolvimento de aplicações para TVD;
- levantamento das métricas sobre tempo de desenvolvimento e a adequação dos métodos ágeis Scrum, XP, Híbrido (Scrum/XP) e XDTv através dos experimentos controlados (que podem ser replicados) e os *surveys* realizados com profissionais experientes em TVD;

- a definição do método ágil e híbrido denominado XDTv que incorpora características metodológicas de Gestão de Projetos e de Engenharia de Software, cuja customização permite auxiliar o desenvolvimento de aplicações para TVD; e
- os artefatos do PATV e o Modelo de Fluxo e Sincronismo de Mídias (MFSM).

7.2. LIMITAÇÕES E RISCOS A VALIDADE DA PESQUISA

A presente pesquisa apresenta as seguintes limitações no seu escopo:

- a realização de experimentos controlados restringe a aplicação desenvolvida às variáveis de controle, deste modo, é possível que os resultados obtidos não reflitam a indústria de software real;
- o distanciamento da indústria de desenvolvimento de software é caracterizado pelas seguintes variáveis de controle: a) a experiência exclusivamente acadêmica dos participantes dos experimentos; b) a invariável quantidade de tempo dedicado para interação entre o cliente e os times; c) os requisitos das aplicações; e d) ferramentas utilizadas;
- as conclusões obtidas ficam restritas à amostra, ou seja, aos métodos de desenvolvimento investigados;
- a qualidade das aplicações desenvolvidas não foi considerada; e
- ausência de métricas relacionadas as alterações de requisitos. Porém, tal demanda não se mostrou prioritária devido à simplicidade e o curto ciclo de vida das aplicações de TVD.

7.3. TRABALHOS FUTUROS

Esta pesquisa serve como base para a elaboração dos seguintes trabalhos futuros:

1. Realizar um novo ciclo de experimentos, similar ao segundo experimento, visando reforçar os resultados obtidos, incorporando às duas repetições apresentadas pela presente pesquisa, outras duas

novas repetições, totalizando quatro repetições para o segundo experimento. Este trabalho pode ser realizado através de uma dissertação de mestrado;

2. Incorporar ao método XDTv a atividade de evolução de software, realizado experimentos que avaliem o desempenho desta incorporação. Este trabalho pode ser realizado através de uma pesquisa de mestrado ou doutorado, de acordo com o aprofundamento da pesquisa;
3. Realizar um levantamento de técnicas de Linhas de Produto de Software, buscando aquelas mais adequadas às peculiaridades inerentes às aplicações para TVD, auxiliando, possivelmente, na velocidade do seu desenvolvimento. Este trabalho pode ser realizado através de uma pesquisa de mestrado ou doutorado, de acordo com o aprofundamento da pesquisa;
4. Adotar o método no desenvolvimento de aplicações para IPTV, Web TV e TV Híbrida, mensurando seu desempenho e possíveis pontos de melhoria ou adaptação. Este trabalho pode ser realizado através de uma pesquisa de mestrado ou doutorado, de acordo com o aprofundamento da pesquisa;
5. Incorporar profissionais da área de TV, voltados para a elaboração do conteúdo audiovisual, ao time de desenvolvimento. Este trabalho pode ser realizado através de uma dissertação de mestrado;
6. Incorporar e avaliar a atividade de desenvolvimento dirigido a testes (*test first*), revisões e inspeções de códigos, testes unitários em busca da melhor relação de custo e benefício para otimizar o tempo dedicado para a atividade de testes. Este trabalho pode ser realizado através de uma pesquisa de mestrado ou doutorado, de acordo com o aprofundamento da pesquisa;
7. Implementar uma ferramenta visual para modelar o MFSM, permitindo a converter automaticamente o MFSM em código fonte NCL. Este trabalho pode ser realizado através de uma pesquisa de mestrado ou doutorado, de acordo com o aprofundamento da pesquisa; e

8. Avaliar o desempenho do método XDTv sobre a atividade de especificação das alterações de requisitos (*change request*), comparando os resultados obtidos com os demais métodos existentes. Este trabalho pode ser realizado através de uma dissertação de mestrado.

Referências

"Não há felicidade sem dever cumprido."

Chico Xavier

ABNT NBR - Associação Brasileira de Normas Técnicas, Norma Brasileira. *Televisão digital terrestre* – Codificação de dados e especificações de transmissão para radiodifusão digital - Parte 2: Ginga-NCL para receptores fixos e móveis – Linguagem de aplicação XML para codificação de aplicações. Número de referência: ABNT NBR 15606-2:2007. 2007 [on-line]. www.abnt.org.br/imagens/Normalizacao_TV_Digital/ ABNTNBR15606-1_2007Vc_2008.pdf.

ABRANTES, J. F.; TRAVASSOS, G. H. *Avaliação de Características de Agilidade e Práticas Ágeis para Processos de Software*. In: Congresso Ibero-Americano em Engenharia de Software, Buenos Aires, Argentina, 2012.

ABRANTES, J. F.; TRAVASSOS, G. H. *Common Agile Practices in Software Processes*. In: International Symposium on Empirical Software Engineering and Measurement, pp.355-358, 2011.

ABRANTES, J. F.; TRAVASSOS, G. H. *Caracterização de Métodos Ágeis de Desenvolvimento de Software*. In: Primeiro Workshop de Desenvolvimento Rápido de Aplicações – VI Simpósio Brasileiro de Qualidade de Software, Porto de Galinhas-PE, Brasil, 2007.

ALMEIDA, J. A. DE; COSTA, A. P. C. S. Sistema de Apoio a Decisão Multicritério para Seleção de Portfólio de Sistemas de Informação. In: XXX Encontro Nacional de Engenharia de Produção. São Carlos, SP, 2010.

ARAÚJO E. C; REDLICH, L. R; LAGO V. C; MORENO M. F; SOARES, L. F. G. *Suíte de Testes de Conformidade para o Ginga-NCL*. In: Simpósio Brasileiro de Sistemas Multimídia e Web, 2011.

ARAÚJO, M. C. B. de; ALENCAR, L. H. Modelo de Seleção de Fornecedores Utilizando O PROMETHEE para Decisão em Grupo. In: XXXII Encontro Nacional de Engenharia de Produção. Bento Gonçalves - RS, 2012.

ASFORA, D. M. *Uma abordagem para a priorização de requisitos em ambientes ágeis*. Dissertação (mestrado) - Universidade Federal de Pernambuco, Recife, 2009.

ATSC - Advanced Television Systems Committee. *Digital Television Standard* (A/53) Revision E, with Amendments No. 1 and 2, Sept. 2006.

BARBOSA, S. D. J.; SOARES, L. F. G. *TV digital interativa no Brasil se faz com Ginga*: Fundamentos, Padrões, Autoria Declarativa e Usabilidade. In: Tomasz Kowaltowski and Karin Breitman (orgs.) atualizações em informática. XXVIII Congresso da Sociedade Brasileira de Computação. Jornadas de Atualização em Informática (JAI/SBC), 2008.

BARTINDALE, T.; VALENTINE, E.; GLANCY, M.; KIRK, D.; WRIGHT, P.; OLIVIER, P. Facilitating TV production using StoryCrate. In: Proceedings of the 9th ACM Conference on Creativity & Cognition, June 17-20, Sydney, Australia, 2013.

BECK, K. *Extreme Programming Explained*: Embrace Change. Addison-Wesley, 1ª edition, 1999.

BECK, K; ANDRES, C. *Extreme Programming Explained*: Embrace Change. Addison-Wesley Professional, 2a. edição, 2004.

BECK, K.; BEEDLE, M.; VAN BENNEKUM, A.; COCKBURN, A.; CUNNINGHAM, W.; FOWLER, M.; GRENNING, J.; HIGHSMITH, J.; HUNT, A.; JEFFRIES, R.; KERN, J.; MARICK, B.; MARTIN, R. C.; MELLOR, S.; SCHWABER, K.; SUTHERLAND, J.; THOMAS, D. *Manifesto for Agile Software Development* [online]. 2001. http://www.agilemanifesto.org.

BECK, K; FOWLER, M. *Planning Extreme Programming*. Publisher: Addison Weley, First Edition October 12, ISBN: 0-201-71091-9, 160 páginas, 2000.

BELLOTTI, F.; BERTA, R.; DE GLORIA, A.; OZOLINA, A. Investigating the added value of interactivity and serious gaming for educational TV. In: Computers & education [0360-1315], vol:57 iss:1 pg:1137-1148, 2011.

BERTRAM, D. *Likert Scales.* Topic Report, The Faculty of Mathematics – University of Belgrade – Servia, 2009.

BOEHM, B. W. *A Spiral Model of Software Development and Enhancement*. IEEE Computer, Vol. 21, No. 5, pp. 61-72, 1988.

BRANS, J. P.; VINCKE, P.H. A Preference Ranking Organization Method, the PROMETHEE Method for MCDM. Mgmt. Sci., v.31, p647-656, 1985.

CAMPOS, V. R. Modelo de apoio à decisão multicritério para priorização de projetos em saneamento. Tese (Doutorado) – Escola de Engenharia de São Carlos. Universidade de São Paulo, São Paulo, 2011.

CHAPMAN, J.; CHAPMAN, N. Digital Multimedia. In: Chichester. Editora: John Wiley & sons, 2004.

CHAOS. SUMMARY FOR 2010. *The Standish Group International*, *Inc*. In: www.standishgroup.com, 2010.

CHAOS. THE CHAOS MANIFESTO. Think Big, Act Small. *The Standish Group International*, *Inc*. In: www.standishgroup.com, 2013.

COCKBURN, A. *Crystal Clear*: a Human-Powered Methodology for Small Teams. Addison-Wesley Professional, 2004.

COCCO, L.; MANNARO, K.; CONCAS, G.; MARCHESI, M. Simulating Kanban and Scrum vs. Waterfall with System Dynamics. Agile Processes in Software Engineering and Extreme Programming. Lecture Notes in Business Information Processing Volume 77, pp 117-131, 2011.

COTTRELL, W. *Standards, compliance, and Rational Unified Process* [on-line]. 2004. http://www.ibm.com/developerworks/rational/library/ 4763.html.

ENSSLIN, L.; GIFFHORN, E.; ENSSLIN, S. R.; PETRI, S. M.; VIANNA, W. B. *Avaliação do desempenho de empresas terceirizadas com o uso da metodologia multicritério de apoio à decisão* - construtivista. Pesqui. Oper., Rio de Janeiro, v. 30, n. 1. 2010.

ETSI - European Telecommunications Standards Institute. *Digital Video Broadcasting (DVB)*: Globally Executable MHP (GEM) Specification 1.0.2. ETSI TS 102 819 V1.3.1. 2005.

FERNANDES, M.; TAVARES, T. A.; PÔRTO, E. *iTVnews*: Uma Ferramenta para Construção de Aplicações Telejornalísticas em TVDI. In: WebMedia: XVII Simpósio Brasileiro de Multimídia e Web, 2011.

FERREIRA, T. P.; KULESZA, R; SOUZA FILHO, G. L. Uma Ferramenta de Autoria para Aplicações NCL e NCLua: uma Abordagem Orientada a Templates e com Suporte a Serviços Web. In: XXVIII Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia). Workshop de Teses e Dissertações, São Paulo, 2012.

FLICK, U. *Introdução à pesquisa qualitativa*. Artmed, Porto Alegre, 2009.

GODOY NETO, M.; FERRAZ, C. A. G. *Uma Abordagem de Desenvolvimento de Aplicações para TV Digital baseada em Metodologias Ágeis*. In: WebMedia'11: XVII Simpósio Brasileiro de Multimídia e Web. XI Workshop de Teses e Dissertações, Florianópolis, 2011.

GODOY NETO, M.; AMORIM A. S.; FERRAZ C. A. G. *GTvD*: um gerenciador de torneios esportivos para TV Digital desenvolvido através de metodologias ágeis. Workshop Tecnologias da Informação e Comunicação nos Grandes Eventos Esportivos, 2012.

GODOY NETO, M.; FERRAZ C. A. G. *XDTv*: uma Adaptação de Métodos Ágeis para o Desenvolvimento de Aplicações para TV Digital. Workshop de Desenvolvimento Rápido de Aplicações - WDRA. Simpósio Brasileiro de Qualidade de Software – SBQS, Fortaleza, 2012.

GODOY NETO, M.; FERRAZ, C. A. G. XDTv: um método Ágil para o Desenvolvimento de Aplicações para TV Digital. In: Congresso Brasileiro de Software: Teoria e Prática (CBSoft), Brasília - DF, 2013.

GODOY NETO, M.; FERRAZ, C. A. G. XDTv: Agile Development of Applications for Digital TV. In: Brazilian Symposium on Multimedia and the Web, Salvador – BA, 2013.

HIGHSMITH, J. M. *Exciting and anxiety-ridden*: Adaptive Software Development. In: American Programmer, 23–29, 1997.

HIGHSMITH, J. *Agile Project Management*: Creating Innovative Products. Addison Wesley, 2004.

ITU - International Telecommunication Union. *Recommendation ITU-T H.761* [on-line]. 2011. http://www.itu.int/rec/T-REC-H.761-201106-I/en.

LEFFINGWELL, D.; WIDRIG, D. *Managing Software Requirements*: A Unified Approach – Addison-Wesley object technology series, Addison Wesley, ISBN: 0-201-61593-2, 2000.

LOPES, Y.; Costa, A. P. C. S. Modelo de Decisão Para Seleção de Sistemas de Informação Baseado em Decisão Multicritério e Programação Inteira 0-1. In: Revista Gestão Industrial. ISSN 1808-0448 / v. 03, n. 04: p. 135-146. D.O.I.: 10.3895/S1808-04482007000400011, 2007.

LULA, M. M. M; GUIMARÃES, A. P; TAVARES, T. A. *Um Processo Ágil para Especificação de Requisitos em Programas Interativos com foco em Roteiros de TV*. In: V Workshop de Desenvolvimento Rápido de Aplicações, 2011.

MARCONI, M. A.; LAKATOS, E. *Fundamentos de metodologia científica*. Editora Atlas, São Paulo/SP, 2003.

MARQUES NETO, M. C. *Contribuições para a Modelagem de Aplicações Multimídia em TV Digital Interativa*. Tese (Doutorado multi-institucional) –

Universidade Federal da Bahia, Universidade Estadual de Feira de Santana e Universidade Salvador, 2011.

MUSHTAQ, Z; QURESHI, M. R. J. *Novel Hybrid Model*: Integrating Scrum and XP. I.J. Information Technology and Computer Science, 2012, 6, 39-44, 2012.

MATTOS, D. P. de; SILVA, J. V. da; MUCHALUAT-SAADE, D. C. NEXT: graphical editor for authoring NCL documents supporting composite templates. In: Proceedings of the 11th european conference on Interactive TV and video (EuroITV '13). ACM, New York, NY, USA, 89-98, 2013.

NIELSEN, JACOB. *Usability Engineering*. Boston, Academic Press, 1993.

OLIVEIRA, S. R. B. *ProDefiner*: Uma Abordagem Progressiva para a Definição de Processos de Software no Contexto de um Ambiente Centrado no Processo. Tese (Doutorado) – Universidade Federal de Pernambuco, Recife, 2007.

PAIVA, M. C. de. Uma implementação em software do subsistema de transmissão do padrão ISDB-TB. Dissertação (Mestrado) – Instituto Nacional de Telecomunicações (INATEL), Santa Rita do Sapucaí – MG, 2010.

PALMER, S. R.; FELSING, J. M. *A Practical Guide to Feature Driven Development*. Prentice Hall, 2002.

PEQUENO, H.S.L.; GOMES, G.A.M.; ANDRADE, R.M.C.; DE SOUZA, J.N.; DE CASTRO, M.F. *FrameIDTV*: a framework for developing interactive applications on digital television environments. In: Journal of Network and Computer Applications 33, 503–511. 2010.

POPPENDIECK, M.; POPPENDIECK, T. *Lean Software Development*: an Agile Toolkit for Software Development Managers. Addison-Wesley Professional, 2003.

PRESSMAN, R. S. *Engenharia de Software*, 6ª edição. ISBN: 85-86804-57-6. São Paulo, McGraw-Hill, 2006.

PRESSMAN, R. S. “Engenharia de Software”, Mc Graw Hill, 7ª edição, 2011.

QUINTERO, I. A.; RODRÍGUEZ, M. M.; BARAJAS, C. O.; CERÓN, J. V. P.; RODRIGUEZ, P. M.; CADAVID, A. N. Kroster-MHP Game for Digital TV. Developing Process, Design, and Programming Considerations Against Technical Issues. In: IEEE Revista Iberoamericana de Tecnologias del Aprendizaje, Vol. 8, No. 4, 2013.

ROBERTO M.; A. FER; C. BOTELHO. iTV project: an authoring tool for mhp and ginga-j based on a web environment. In: Proceedings of the 1st international conference on Designing interactive user experiences for TV and video (UXTV '08). ACM, New York, NY, USA, 179-182, 2008.

SAITO, Y.; SATO, J.; AOKI, H.; KHUONG, D. C. V. *Software Framework for Embedded Digital Consumer Appliance*. In: Consumer Electronics (ISCE). IEEE 15th International Symposium. Pg. 528 - 531. 2011.

SAMPAIO, A. T. F. *XWebProcess*: Um processo ágil para o desenvolvimento de aplicações web. Dissertação (Mestrado) – Universidade Federal de Pernambuco, Recife, 2004.

SÁNCHEZ, I. F. H. *Quadrados Latinos com Aplicações em Engenharia de Software*. Dissertação (Mestrado) – Universidade Federal de Pernambuco, Recife, 2011.

SANZ, E. *Statistical, Ecosystems and Competitiveness Analysis of the Media and Content Industries*: European Television in the New Media Landscape. In: JRC Technical Reports. 2012.

SATO, D. T. *Uso eficaz de métricas em métodos ágeis de desenvolvimento de software*. Dissertação (Mestrado) [online] – Universidade de São Paulo, São Paulo, 2007. http://www.ime.usp.br/~gold/.

SCHACH, S. R. *Engenharia de Software os Paradigmas Clássico e Orientado a Objetos.* Editora: Mc Graw Hill, 7ª Edição. ISBN: 9788563308443, 2009.

SCHWABER, K; BEEDLE, M. *Agile Software Development with Scrum*. Prentice Hall, 2001.

SHELLHAMMER, S. J. *Spectrum Sensing*. In: IEEE 802.22. IAPR Wksp. Cognitive Info. Processing, 2008.

SILVA, T. R. *XPU*: Uma Extensão do Método XP Visando à Usabilidade. Dissertação (Mestrado) – Universidade Federal de Minas Gerais, Belo Horizonte, 2009.

SILVA, F. P. R. *Xtation*: um ambiente para execução e teste de aplicações interativas para o Middleware Ginga. Dissertação (Mestrado) – Universidade Federal da Paraíba, João Pessoa, 2010.

SNYDER, C. *Paper Prototyping*: The Fast and Easy Way to Design and Refine User Interfaces, Morgan Kaufmann, 2003.

SOARES NETO, A. S.; SOARES, L. F. G.; RODRIGUES, R. F.; BARBOSA, S. D. J. *Construindo Programas Audiovisuais Interativos Utilizando a NCL 3.0 e a Ferramenta Composer*. Tutorial [online]. Telemídia / PUC – Rio, 2ª edição, 2007. http://www.ncl.org.br.

SOARES, L.F.G.; MORENO, M.F.; SOARES NETO, C. DE S.; MORENO, M. F. *Ginga-NCL*: Declarative Middleware for Multimedia IPTV Services. In: Communications Magazine, IEEE, 2010.

SOMMERVILLE, I. *Engenharia de Software*, Editora Pearson, 9ª edição, 2011.

SONG, H.Y.; PARK, J. *Design of an interoperable middleware architecture for digital data broadcasting*. In: Consumer Electronics, IEEE Transactions on, 2006.

STAPLETON, J. *DSDM*: a framework for business centered development. Addison-Wesley Professional, 1997.

TEKCAN, T.; ZLOKOLICA, V.; PEKOVIC, V.; TESLIC, N.; GÜNDÜZALP, M. *User-driven Automatic Test-case Generation for DTV/STB Reliable Functional Verification*. In: Consumer Electronics, IEEE Transactions. Volume: 58, Issue: 2, Page(s): 587 - 595, 2012.

TRAVASSOS, G. H. *Introdução à Engenharia de Software Experimental*. Relatório Técnico RT-ES-590/02 do Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ, Rio de Janeiro, Brasil, 2002.|

VAVILOV, D.; MELIKHOVA, L.; LOGUNOV, A. Perspectives of Digital TV applications development in Russia. In: 5th Central and Eastern European - Software Engineering Conference in Russia (CEE-SECR). Pg. 173 - 176. 2009.

VEIGA, E. G. *Um Modelo de Processo para o Desenvolvimento de Programas para TV Digital e Interativa*. Dissertação (Mestrado) – Universidade Federal da Paraíba, João Pessoa, 2006.

VEIGA, E. G.; TAVARES, T. A. *Um Modelo de Processo para o Desenvolvimento de Programas para TV Digital e Interativa*. In: Workshop de Teses e Dissertações do WebMedia, 2006.

VEIGA, E. G.; TAVARES, T. A. *Um Modelo de Processo para o Desenvolvimento de Programas para TV Digital e Interativa baseado em Metodologias Ágeis*. In: Workshop de Desenvolvimento Rápido de Aplicações, 2007, Porto de Galinhas, 2007.

VERSIONONE. *The State of Agile Development Survey Results* [online]. 2009. Disponível em: http://www.versionone.com/pdf/2009_State_of_Agile_Development_Survey_Results.pdf.

VERSIONONE. *The State of Agile Development Survey Results* [online]. 2010. Disponível em: http://www.versionone.com/pdf/2010_State_of_Agile_Development_Survey_Results.pdf.

VERSIONONE. *The State of Agile Development Survey Results* [online]. 2011. Disponível em: http://www.versionone.com/pdf/2011_State_of_Agile_Development_Survey_Results.pdf.

VERSIONONE. *7th Annual State of Agile Development Survey* [online]. 2013. Disponível em: http://www.versionone.com/pdf/7th-Annual-State-of-Agile-Development-Survey.pdf

WELLS, J. D. *Extreme Programming*: A Gentle Introduction, [On-line]. 2000. http://www.extremeprogramming.org.

WILLIAMS, L. *Pair Programming Illuminated*. Addison-Wesley Professional, 2002.

Apêndices

A seguir são apresentados os documentos, formulários, questionários que contribuíram para a elaboração dessa tese.

Apêndice A

Questionário utilizado no início do primeiro experimento para a realização da análise de currículo e experiência dos desenvolvedores participantes.

Nome:				
E-mail:				
1) Disciplinas de engenharia de software e programação cursadas:				
2) Classifique seu conhecimento sobre desenvolvimento de aplicações em NCL para TV Digital:				
Desconheço	Fraco	Básico	Intermediário	Avançado
[ ]	[ ]	[ ]	[ ]	[ ]

3) Informe se conhece outra linguagem de programação, classifique ao lado seu grau de conhecimento sobre ela: (Muito Fraco, Fraco, Básico, Intermediário ou Avançado**).**

Linguagem	**Classificação**

4) Classifique seu conhecimento sobre a metodologia SCRUM:

Desconheço	Fraco	Básico	Intermediário	Avançado
[ ]	[ ]	[ ]	[ ]	[ ]

5) Classifique seu conhecimento sobre a metodologia XP:

Desconheço	Fraco	Básico	Intermediário	Avançado
[ ]	[ ]	[ ]	[ ]	[ ]

6) Informe se conhece outra metodologia de desenvolvimento de software, classifique ao lado seu grau de conhecimento sobre ela: (Muito Fraco, Fraco, Básico, Intermediário ou Avançado**).**

Linguagem	Classificação

7) Você já desenvolveu alguma aplicação sem utilizar uma metodologia conhecida na literatura? [ ] SIM [ ] NÃO

Caso afirmativo, o quê achou dessa experiência?

APÊNDICE B

Resumo geral da aplicação, fornecido para todos os times envolvidos no primeiro experimento. Descreve de forma abstrata os requisitos básicos e superficiais de cada funcionalidade solicitada.

<table>
<tr><th colspan="2">Time A, B ou C – Scrum, XP ou Híbrido (Scrum/XP)</th></tr>
<tr><td colspan="2">Metodologia: seguir rigorosamente o método Y.</td></tr>
<tr><th>Nome</th><th>Papel / Papeis</th></tr>
<tr><td>Aluno J</td><td></td></tr>
<tr><td>Aluno D</td><td></td></tr>
<tr><td>Aluno M</td><td></td></tr>
<tr><td colspan="2">Visão Geral da Aplicação:

A aplicação deve:

RF.01. Manipular informações sobre o campeonato brasileiro, os times, jogos, classificação e patrocinadores desse campeonato.

RF.02. Ser executada sobre um jogo de futebol, onde, além do vídeo contendo o jogo principal, o usuário poderá escolher assistir um novo jogo.

RF.03. Possibilitar visualizar diferentes vídeos de jogos tanto simultaneamente (em paralelo), como também, visualizar um jogo diferente do principal exclusivamente (em tela cheia).

RF.04. Permitir pausar/continuar cada jogo que esteja em execução.

RF.05. Exibir a quantidade de jogos disponíveis.

RF.06. Possibilitar a compra de um ou mais produtos (como bola, chuteira, camisa, entre outros) quando forem estes produtos forem exibidos durante o jogo.

RF.07. Oferecer informações adicionais (como nomes dos times,</td></tr>
</table>

classificação dos times no campeonato, nomes dos jogadores, técnico, entre outras.) sobre cada jogo.

RF.08. Mostrar informações adicionais sobre os patrocinadores no momento em que esses aparecem na tela.

O segredo industrial é uma forte característica do desenvolvimento de software, por isso, é expressamente proibido a troca qualquer informação, código ou documentos sobre o desenvolvimento desse projeto com os membros dos outros projetos.

1ª Iteração **(08/04)** **18:00h**	Objetivo: colher requisitos. Os papéis de cada membro devem estar definidos. A equipe deve estar pronta para colher os requisitos do cliente (prof. Mario) **rigorosamente de acordo** com a metodologia **SCRUM**.
2ª Iteração **(19/04)** **14:00h**	Objetivo: o cliente estará disponível a critério da equipe.
3ª Iteração **(26/04)**	Objetivo: o cliente estará disponível a critério da equipe.
Entrega da Aplicação **(03/05)**	

APÊNDICE C

Formulário de coleta das horas trabalhadas utilizado por todos os times envolvidos no primeiro experimento, cujo objetivo foi investigar os métodos Scrum, XP e Híbrido (Scrum/XP).

Dados sobre o projeto desenvolvido e o desenvolvimento das funcionalidades:								
Obs: o preenchimento dessa tabela deve obedecer rigorosamente a realidade enfrentada no desenvolvimento do projeto. As horas trabalhadas não irão influenciar em hipótese nenhuma a nota do grupo, nem tão pouco a nota individual. A nota será atribuída ao grupo e não individualmente.								
Nome:	Aluno M.							
Data de realização:	Nome do(s) membro(s) envolvido(s):	Hora de início:	Hora de fim:	Tempo de pausa:	Iteração nº:	Requisito nº:	Atividades	Observações relevantes:
08/abr	Aluno M.	09:00	10:30	00:00	1	1, 2 e 3	Planejamento	xxx
11/abr	Aluno M.	18:00	20:00	00:15	1	1, 2 e 3	Implementação	yyy
...	...	...	...	...	...	...	...	...
12/abr	Aluno M.	18:00	20:00	00:15	1	3	Implementação	www

APÊNDICE D

Questionário utilizado por todos os times envolvidos no primeiro experimento, cujo objetivo foi investigar os métodos Scrum, XP e Híbrido (Scrum/XP).

ATENÇÃO: as respostas, positivas ou negativas, não influenciarão em hipótese alguma em sua avaliação nesta disciplina.

Nome: _____ .
Método utilizado: _____ **Papel:** _____ .

1) Classifique seu grau de desgaste/cansaço (quanto programador/lider) no desenvolvimento dessa aplicação.

Extremo Desgaste	Desgaste alto	O que era esperado	Pouco desgaste	Não houve desgaste
[ ]	[ ]	[ ]	[ ]	[ ]

2) O método ágil utilizado por seu grupo foi satisfatório? O quanto que este método facilitou o desenvolvimento da aplicação?

[ ]Facilitou 100%. [ ]Facilitou 75%. [ ]Facilitou 50%. [ ]Facilitou apenas 25%. [ ]Não Facilitou.

3) Classifique de maneira geral o grau de complexidade da aplicação (funcionalidades e recursos) solicitada pelo cliente?

[ ]Muito complicado. [ ]Complicado. [ ]Regular. [ ]Simples. [ ]Muito Simples.

4) Classifique de maneira geral o grau de complexidade da programação/codificação das funcionalidades e recursos solicitados pelo cliente?

[ ]Muito complicado. [ ]Complicado. [ ]Regular. [ ]Simples. [ ]Muito Simples.

5) O método ágil utilizado por você é adequado para o desenvolvimento de aplicações para TV Digital?

	Este método **deve** ser utilizado **sem qualquer alteração**, atende 100% das necessidades.
	Este método **deve** ser utilizado com **pequenas alterações** em seus procedimentos.
	Este método **pode** ser utilizado, porém são necessárias **muitas alterações** em seus procedimentos.
	Este método **não deve** ser utilizado em TVD.

6) Caso você fosse contratado para trabalhar em uma fábrica de aplicações para TVD e pudesse escolher um método para desenvolver suas aplicações, qual seria a sua decisão?

	Escolheria a método: _____.
	Escolheria o mesmo método utilizado por mim no projeto desta disciplina.
	Escolheria o mesmo método utilizado por mim no projeto com pequenas alterações.
	Escolheria o mesmo método utilizado por mim no projeto com muitas alterações.
	Elaboraria, junto com meu time de desenvolvimento, meu próprio método.

7) Relate três pontos fortes e três pontos fracos sobre o método utilizado por você.

	Pontos Fortes	Pontos Fracos
1		
2		
3		

8) Houve alguma funcionalidade solicitada que não foi implementada? Caso afirmativo, qual foi o motivo. Caso negativo, informe o que foi preciso fazer para desenvolver a aplicação solicitada.

[ ]Sim. [ ]Não. Justifique:

9) Relate qual foi a maior dificuldade encontrada no desenvolvimento dessa aplicação?

APÊNDICE E

Questionário utilizado ao fim do primeiro experimento para a realização do *survey* com desenvolvedores de aplicações de TVD.

Questionário: metodologias de engenharia de software para o desenvolvimento de aplicações para TV Digital.

Caro colega, este questionário se destina a todos aqueles que já desenvolveram alguma aplicação para TV Digital. O objeto é realizar um levantamento sobre a utilização de métodos de engenharia de software (MPS.br, CMM, Scrum, XP, Crystal, entre outros) no desenvolvimento de aplicações para esta tecnologia.

Esta pesquisa conta com a orientação do Prof. Dr. Carlos Ferraz e faz parte de uma tese de doutoramento no CIn/UFPE, tendo como título provisório: "Uma Proposta de Melhoria para o Desenvolvimento de Aplicações para TV Digital baseada em Métodos Ágeis".

Você dedicará de 4 e 6 minutos para responder esse questionário. Certo de sua valiosa contribuição, desde já agradeço. Mario Godoy mario.godoy@univasf.edu.br

* *Required*

Nome [Desejável, porém opcional]

E-mail [Desejável, porém opcional]

Instituição [Desejável, porém opcional]

Qual sua área de formação? [Desejável, porém opcional]

Marque seu grau de instrução: * (é possível marcar mais de uma

opção)

[ ] Técnico;
[ ] Graduação em andamento;
[ ] Graduado;
[ ] Especialização em andamento;
[ ] Especialista;
[ ] Mestrado em andamento;
[ ] Mestre;
[ ] Doutorado em andamento;
[ ] Doutor;
[ ] Other: _____ .

1) Quanto tempo possui de experiência no desenvolvimento de aplicações para TVD? *

[ ] menos de 6 meses;

[ ] mais de 6 meses e menos de 1 ano;

[ ] mais de 1 ano e menos de 2 anos;

[ ] mais de 2 anos e menos de 3 anos;

[ ] mais de 3 anos e menos de 4 anos;

[ ] mais de 4 anos.

2) Seu contato com o desenvolvimento de aplicações foi: *

[ ] Apenas acadêmico

[ ] Apenas profissional

[ ] Acadêmico e profissional

[ ] Other: _____ .

3) Avalie a extensão do seu conhecimento sobre o desenvolvimento de aplicações para TVD. *

Não conheço	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	Especialista no desenvolvimento.

4) Você utiliza algum método de engenharia de software para o desenvolvimento de aplicações de TVD? Qual? * Exemplo: MPS.br,

Scrum, XP, Crystal, CMM, entre outros.

4.1) Caso afirmativo, utiliza o método puro ou realizou adaptações?

4.2) Caso existam, quais adaptações são indispensáveis?

5) Na sua opinião quais práticas relacionadas a engenharia de software são essenciais no processo de desenvolvimento de aplicações para TVD? * Entenda como práticas as atividades associadas as fases de: especificação, projeto, implementação ou testes.

6) Você utiliza alguma prática capaz de agilizar este processo de desenvolvimento? Qual(is)? *

7) Quais documentos conhecidos na engenharia de software você utiliza para guiar o processo de desenvolvimento e modelar a aplicação? *

8) Caso você já tenha utilizado o método XP, classifique o grau de adequação do mesmo ao desenvolvimento de aplicações para TVD.

Extremamente Inadequado	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	Extremamente adequado.

8.1) Caso existam, na sua opinião, quais são os pontos de melhoria do método XP?

9) Caso você já tenha utilizado o método Scrum, classifique o grau de adequação do mesmo ao desenvolvimento de aplicações para TVD.

Extremamente Inadequado	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	Extremamente adequado.

9.1) Caso existam, na sua opinião, quais são os pontos de

melhoria do método Scrum?

APÊNDICE F

Questionário utilizado no início do segundo experimento para análise de currículo e experiência dos desenvolvedores.

Nome:

E-mail:

1) Disciplinas de engenharia de software e programação cursadas:

2) Classifique seu conhecimento sobre desenvolvimento de aplicações em NCL para TV Digital:

Muito Fraco	Fraco	Regular	Bom	Muito Bom
[ ]	[ ]	[ ]	[ ]	[ ]

3) Informe quais linguagens de programação você já utilizou, classifique ao lado seu grau de conhecimento sobre ela: Muito Fraco, Fraco, Básico, Intermediário ou Avançado**.**

Linguagem	Classificação

4) Classifique seu conhecimento sobre a metodologia SCRUM:

Muito Fraco	Fraco	Regular	Bom	Muito Bom
[ ]	[ ]	[ ]	[ ]	[ ]

5) Classifique seu conhecimento sobre a metodologia XP:

Muito Fraco	Fraco	Regular	Bom	Muito Bom
[ ]	[ ]	[ ]	[ ]	[ ]

6) Informe se conhece outra metodologia de desenvolvimento de software. Classifique ao lado seu grau de conhecimento sobre ela: Muito Fraco, Fraco, Regular, Bom ou Muito Bom.

Metodologia	Classificação

7) Você já desenvolveu alguma aplicação sem utilizar uma metodologia conhecida na literatura? [ ] SIM [ ] NÃO

O quê achou dessa experiência?

APÊNDICE G

Formulário utilizado no Experimento 2, exclusivamente para preenchimento das horas trabalhadas em cada atividade durante a adoção do método híbrido (Scrum/XP).

Formulário Individual para Desenvolvimento do Projeto 1				
Obs: o preenchimento dessa tabela deve obedecer rigorosamente a realidade enfrentada no desenvolvimento do projeto. As horas trabalhadas não irão influenciar em hipótese nenhuma a nota do grupo, nem tão pouco a nota individual. A nota será atribuída ao grupo e não individualmente.				
1ª Interação:	**Nome:**			**Time: Híbrido (Scrum/XP)**
	Papel / Papeis:			
	Duração: 09/10 até 16/10			
DATA	**Atividade**	**Objetivo**	**Nome do(s) membro(s) envolvido(s):**	**Duração (horas)**

Atividades a serem realizadas
Elicitação e Análise de Requisitos
Elaborar Artefato: Histórias do usuário (Especificação de Requisitos);
Avaliação e correções de Requisitos (com o cliente);
Implementação em Par do Requisito (X);
Teste de Aceitação com o Cliente
Correção após os Testes de Aceitação;
Elaborar Artefato: Diagrama de Atividades UML - representar a aplicação;
Elaborar Artefato: Product Backlog;
Elaborar Artefato: Sprint Backlog;
Elaborar Artefato: burn down chart;
Elaborar Artefato: definição de feito para cada história;
Elaborar Artefato: padrão de codificação;
Elaborar Artefato: outro documento/artefato necessário (X);

Glossário

Elicitação e análise de requisitos – é o processo de levantamento dos requisitos do sistema a partir da análise do funcionamento da organização. Podem ser utilizadas técnicas de observação, entrevista com usuários, entre outras formas de levantamento de informações.

Especificação de requisitos – é a tradução das informações obtidas em um documento capaz de definir um conjunto de requisitos. Podem ser classificados como requisitos do usuário (abstratas) e requisitos de sistema, composta por detalhes da funcionalidade a ser implementada.

Avaliação de requisitos – verifica o realismo, consistência e completude dos requisitos, levantando e corrigindo os erros do documento de requisitos.

APÊNDICE H

Formulário utilizado no Experimento 2 exclusivamente para preenchimento das horas trabalhadas em cada atividade durante a adoção do método XDTv.

Formulário Individual para Desenvolvimento do Projeto 1				
Obs: o preenchimento dessa tabela deve obedecer rigorosamente a realidade enfrentada no desenvolvimento do projeto. As horas trabalhadas não irão influenciar em hipótese nenhuma a nota do grupo, nem tão pouco a nota individual. A nota será atribuída ao grupo e não individualmente.				
1ª Interação:	**Nome:**			**Time: XDTv**
	Papel / Papeis:			
	Duração: 09/10 até 16/10			
DATA	**Atividade**	**Objetivo**	**Nome do(s) membro(s) envolvido(s):**	**Duração (horas)**

Atividades a serem realizadas
Elicitação e Análise de Requisitos
Elaborar Artefato: PATv (Especificação de Requisitos);
Avaliação e correções de Requisitos (com o cliente);
Implementação em Par do Requisito (X);
Teste de Aceitação com o Cliente
Correção após os Testes de Aceitação;
Elaborar Artefato: MFSM - que represente a aplicação;
Elaborar Artefato: Product Backlog;
Elaborar Artefato: Sprint Backlog;
Elaborar Artefato: outro documento/artefato necessário (X);

Glossário

Elicitação e análise de requisitos – é o processo de levantamento dos requisitos do sistema a partir da análise do funcionamento da organização. Podem ser utilizadas técnicas de observação, entrevista com usuários, entre outras formas de levantamento de informações.

Especificação de requisitos – é a tradução das informações obtidas em um documento capaz de definir um conjunto de requisitos. Podem ser classificados como requisitos do usuário (abstratas) e requisitos de sistema, composta por detalhes da funcionalidade a ser implementada.

Avaliação de requisitos – verifica o realismo, consistência e completude dos requisitos, levantando e corrigindo os erros do documento de requisitos.

APÊNDICE I

Formulário utilizado no Experimento 2 exclusivamente para preenchimento das reuniões de planejamento durante a adoção do método Híbrido (Scrum/XP).

Formulário sobre o Projeto 1				
Obs: o preenchimento dessa tabela deve obedecer rigorosamente a realidade enfrentada no desenvolvimento do projeto. As horas trabalhadas não irão influenciar em hipótese nenhuma a nota do grupo, nem tão pouco a nota individual. A nota será atribuída ao grupo e não individualmente.				
1ª Iteração	Método: **Híbrido (Scrum / XP)**		**Time:**	
	Contabilizar apenas Reuniões			
	09/10 até 16/10			
DATA	**Tipo de REUNIÃO Tipos: Release Planning, Sprint Planning e Daily Scrum.**	**Nome do(s) membro(s) envolvido(s):**	**Assunto / objetivo**	**Duração (horas)**

APÊNDICE J

Formulário utilizado no Experimento 2 exclusivamente para preenchimento das reuniões de planejamento durante a adoção do método XDTv.

Formulário sobre o Projeto 1				
Obs: o preenchimento dessa tabela deve obedecer rigorosamente a realidade enfrentada no desenvolvimento do projeto. As horas trabalhadas não irão influenciar em hipótese nenhuma a nota do grupo, nem tão pouco a nota individual. A nota será atribuida ao grupo e não individualmente.				
1ª iteração	Método: **XDTv**		**Time:**	
	Contabilizar apenas Reuniões			
	09/10 até 16/10			
DATA	**Tipo de REUNIÃO Tipos: Sprint Planning e Daily Scrum.**	**Nome do(s) membro(s) envolvido(s):**	**Assunto / objetivo**	**Duração (horas)**

APÊNDICE K

Questionário utilizado no segundo **Experimento** para avaliação dos métodos híbrido (Scrum/XP) e XDTv, aplicado após a implementação das aplicações.

QUESTIONÁRIO INDIVIDUAL

30/10/2012

Nome: _____ **Time:** ____ .

Metodologia utilizada: _____ **Papel:** _____ .

ATENÇÃO: As respostas, positivas ou negativas, não influenciarão, em hipótese alguma, em sua avaliação nesta disciplina. Porém, são necessárias para levantar dados para futura publicação científica.

C1) Qual a influência do método adotado na velocidade do desenvolvimento da aplicação?				
Atrasou muito [ ]	**Atrasou** [ ]	**Regular** [ ]	**Acelerou** [ ]	**Acelerou muito** [ ]
Justifique sua resposta:				

C2) Qual o nível de adequação dos artefatos do método ao desenvolvimento de aplicações para TVD?				
Muito inadequado [ ]	**Inadequado** [ ]	**Regular** [ ]	**Adequado** [ ]	**Muito adequado** [ ]
Justifique sua resposta:				

C3) Qual o nível de adequação dos artefatos do método para a coleta de requisitos multimídia?				
Muito Inadequado [ ]	**Inadequado** [ ]	**Regular** [ ]	**Adequado** [ ]	**Muito adequado** [ ]
Justifique sua resposta:				

C4) Qual o nível de adequação do método e seus artefatos para a atividade de Especificação de requisitos?				
Muito Inadequado [ ]	**Inadequado** [ ]	**Regular** [ ]	**Adequado** [ ]	**Muito adequado** [ ]
Justifique sua resposta:				

C5) Qual o nível de adequação do método e seus artefatos para a atividade de Planejamento?				
Muito Inadequado [ ]	**Inadequado [ ]**	**Regular [ ]**	**Adequado [ ]**	**Muito adequado [ ]**
Justifique sua resposta:				

C6) Qual o nível de adequação do método e seus artefatos para a atividade de Implementação?				
Muito Inadequado [ ]	**Inadequado [ ]**	**Regular [ ]**	**Adequado [ ]**	**Muito adequado [ ]**
Justifique sua resposta:				

C7) Qual o nível de adequação do método e seus artefatos para a atividade de testes de aceitação junto ao usuário?				
Muito Inadequado [ ]	**Inadequado [ ]**	**Regular [ ]**	**Adequado [ ]**	**Muito adequado [ ]**
Justifique sua resposta:				

C8) Avalie, de maneira geral, o quão adequado foi o método utilizado para o da aplicação solicitada.

Método: Híbrido (Scrum / XP)

Muito inadequado **0 1 2 3 4 5 6 7 8 9 10** **Muito adequado**

Método: XDTv

Muito inadequado **0 1 2 3 4 5 6 7 8 9 10** **Muito adequado**

Apêndice L

Roteiro semiestruturado utilizado para conduzir o grupo focal, adotado ao fim do segundo experimento, visando pontos de melhoria sobre os métodos híbrido (Scrum/XP) e (XDTV).

1) Qual método mostrou-se mais adequado para a atividade de coleta de requisitos? Porquê?

2) Qual método mostrou-se mais adequado para a atividade de planejamento? Porquê?

3) Qual método mostrou-se mais adequado para a atividade de implementação? Porquê?

4) Qual método apresenta os artefatos mais adequados para o desenvolvimento das aplicações solicitadas? Porquê?

5) De forma geral, algum dos métodos mostrou-se mais adequado ao desenvolvimento das aplicações solicitadas? Porquê?

6) Caso vocês fossem contratados para desenvolver uma aplicação de TVD, qual método vocês adotariam?

7) Relate três pontos fortes e três pontos fracos do método adotado pelo seu time.

	Pontos Fortes	Pontos Fracos
1		
2		
3		

APÊNDICE M

A **Aplicação A** tem como objetivo agregar valor à campanha publicitária de uma concessionária de veículos importados, apresentando as marcas e os diferentes modelos comercializados. Os requisitos funcionais referentes a esta aplicação foram:

RF-01. Apresentar o vídeo publicitário principal e o botão de interação.

RF-02. Ao selecionar o ícone de interação, devem ser exibidas as logomarcas das três marcas (Mercedes, BMW e Lamborghini) dos veículos em forma de ícones interativos.

RF-03. Caso seja selecionada a marca A, deve ser exibida automaticamente, em série (um por vez), a foto de cada modelo disponível, durante “Y” segundos.

RF-04. Caso seja selecionada a marca B, deve ser exibida a foto de um modelo dessa marca, juntamente com um ícone em forma de seta, quando o usuário desejar visualizar o próximo modelo, deve selecionar o ícone com o botão “ok”.

RF-05. Caso seja selecionada a marca C, os modelos disponíveis devem ser disponibilizados em forma de ícones na posição “X”. Quando selecionado um modelo, através das setas direcionais e o botão “ok”, a imagem do modelo deve ser redimensionada ocupando 80% da TV. Caso já exista um modelo sendo exibido, este deverá ser redimensionado para a sua posição inicial de ícone.

RF-06. Para visualizar mais informações sobre qualquer modelo, o usuário deve pressionar o botão verde durante sua exibição.

RF-07. Para qualquer marca, as informações sobre os modelos disponíveis devem ser: imagens do motor e do modelo (reduzido), textos sobre o preço, consumo e o ícone de interatividade.

RF-08. O agendamento de *test drive* corresponde a um formulário contendo os seguintes campos de preenchimento: nome, CPF e telefone.

RF-09. Os dados do agendamento do test drive deverão ser armazenados em arquivo no Set-top Box**.**

RF-10. O botão vermelho (F1), a qualquer momento, deve encerrar (fechar) a aplicação, continuando apenas exibição do vídeo publicitário.

RF-11. Todos os requisitos da aplicação devem ser exibidos sobre o vídeo da campanha publicitária, sem ocultá-lo ou pará-lo.

APÊNDICE N

A Aplicação B tem como objetivo agregar valor à campanha publicitária de uma construtora, apresentando três prédios e seus tipos de apartamentos. Os requisitos funcionais referentes a esta aplicação foram:

RF-01. Apresentar o vídeo publicitário principal e o botão de interação.

RF-02. Ao selecionar o ícone de interação, deve ser apresentada a imagem do Prédio 1 ocupando “y%” do centro da TV durante “x” segundos. Após esse tempo, a imagem deve ser redimensionada ocupando “w%” da lateral TV, tornando-se um botão interativo. Esse procedimento deve se repetir até que os três prédios sejam exibidos.

FR-03. Ao selecionar qualquer prédio, aparecerá automaticamente a imagem correspondente a sua localização em forma de um ícone.

RF-04. Ao pressionar o botão “+”, a imagem da localização deverá aumentar seu tamanho em “n%”, até ocupar 90% da área visual. Ao pressionar o botão “-”, a imagem da localização deverá diminuir seu tamanho em “n%”, até ocupar o tamanho inicial do ícone.

RF-05. Ao selecionar o Prédio 1, a imagem do prédio deve ocupar “q%” da TV, e os apartamentos disponíveis devem ser exibidos em forma de ícones.

RF-06. Quando selecionado algum apartamento, este deve ser exibido no tamanho “j” na posição “s”.

RF-07. Aos “k” segundos de início do vídeo principal, deve aparecer o ícone “Conforto”, quando selecionado, o vídeo deve ser posicionado no tempo de duração “p” que está relacionado ao assunto. O mesmo ocorre para os temas: “Imponente”, “*Pay per use*” e “Academia”.

RF-08. Aos “t” segundos a partir do início do vídeo publicitário, deverá ser exibido o ícone do “Telefone” durante “r” segundos, sobreposto a todas as demais mídias. Se o ícone for selecionado, deve aparecer o número do telefone para contato. O mesmo deve ocorrer para os ícones “Endereço” e “Corretor”.

RF-09. Para qualquer prédio, durante a apresentação dos detalhes de cada apartamento, deve ser exibido um ícone “Proposta”, onde o usuário poderá fazer uma proposta de compra sobre o imóvel apresentado. O formulário deverá armazenar: número do prédio, código do apartamento, valor da oferta, CPF, nome e telefone do usuário.

RF-10. Os dados da “Proposta” devem ser armazenados em arquivo no Set-top Box**.**

RF-11. O botão vermelho (F1), a qualquer momento, deve encerrar (fechar) a aplicação, continuando apenas exibição do vídeo publicitário.

RF-12. A qualquer momento, o botão amarelo deve exibir aos três prédios.

RF-13. Todos os requisitos da aplicação devem ser exibidos sobre o vídeo da campanha publicitária, sem ocultá-lo ou pará-lo.

APÊNDICE O

Código fonte NCL sobre a aplicação Quiz.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<ncl id="interfaces_TC-001_1" xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile">
   <head>
      <regionBase>
         <region id="rgInicio" left="80%" top="10%" width="10%" height="10%"/>
         <region id="rgPergunta" left="20%" top="20%" width="70%" height="10%"/>
         <region id="rgResposta1" left="20%" top="50%" width="70%" height="10%"/>
         <region id="rgResposta2" left="20%" top="70%" width="70%" height="10%"/>
         <region id="rgTelaCheia" left="0" top="0" width="100%" height="100%"/>
       </regionBase>

       <descriptorBase>
         <descriptor id="dInicio" region="rgInicio" focusIndex="1" explicitDur="30s"/>
         <descriptor id="dPergunta" region="rgPergunta"/>
         <descriptor id="dResp1" region="rgResposta1" focusIndex="1" moveDown="2"/>
         <descriptor id="dResp2" region="rgResposta2" focusIndex="2" moveUp="1"/>
         <descriptor id="dTelaCheia" region="rgTelaCheia" explicitDur="15s"/>
       </descriptorBase>

       <connectorBase>
         <causalConnector id="onSelectionStartNStopN">
                <simpleCondition role="onSelection"/>
                <compoundAction operator="seq">
                    <simpleAction role="start" max="unbounded" qualifier="par"/>
                    <simpleAction role="stop" max="unbounded" qualifier="par"/>
                </compoundAction>
         </causalConnector>
       </connectorBase>
   </head>

   <body>
      <port id="pInicio" component="inicio"/>
      <media type="image/png" id="inicio" src="media/btInicio.png" descriptor="dInicio"/>
      <media type="text/plain" id="txPergunta" src="media/pergunta.txt" descriptor="dPergunta">
            <property name="fontColor" value="red"/>
            <property name="fontSize" value="30"/>
      </media>
      <media type="image/png" id="btResposta1" src="media/resposta1.png" descriptor="dResp1"/>
      <media type="image/png" id="btResposta2" src="media/resposta2.png" descriptor="dResp2"/>
      <media type="image/png" id="imgTrofeu" src="media/imgTrofeu.png" descriptor="dTelaCheia"/>
      <media type="image/png" id="imgBurro" src="media/imgBurro.png" descriptor="dTelaCheia"/>

      <link xconnector="onSelectionStartNStopN">
            <bind component="inicio" role="onSelection"/>
            <bind component="inicio" role="stop"/>
            <bind component="txPergunta" role="start"/>
            <bind component="btResposta1" role="start"/>
            <bind component="btResposta2" role="start"/>
      </link>
      <link xconnector="onSelectionStartNStopN">
            <bind component="btResposta1" role="onSelection"/>
            <bind component="txPergunta" role="stop"/>
            <bind component="btResposta1" role="stop"/>
            <bind component="btResposta2" role="stop"/>
            <bind component="imgTrofeu" role="start"/>
      </link>
      <link xconnector="onSelectionStartNStopN">
            <bind component="btResposta2" role="onSelection"/>
            <bind component="txPergunta" role="stop"/>
            <bind component="btResposta1" role="stop"/>
            <bind component="btResposta2" role="stop"/>
            <bind component="imgBurro" role="start"/>
      </link>
   </body>
</ncl>
```

APÊNDICE P

Questionário aplicado no Survey realizado no CESAR, com profissionais experientes em TVD.

Survey CESAR - 2013

Prezados colegas do CESAR,
Esta pesquisa conta com a orientação do Prof. Carlos Ferraz e faz parte de uma tese de doutoramento no CIn/UFPE, tendo como título provisório: "Uma Proposta de Melhoria para o Desenvolvimento de Aplicações para TV Digital baseada em métodos ágeis".
Este questionário visa coletar a sua opinião sobre a adequação do método XDTv para o desenvolvimento de aplicações para TV Digital.
Você dedicará de 6 e 8 minutos para responder esse questionário.
Se necessário, consulte o guia básico do método XDTV em:
http://www.univasf.edu.br/~mario.godoy/xdtv/

Desde já, agradeço a participação e valorosa contribuição de todos na realização dessa pesquisa.
Mario Godoy
mario.godoy@univasf.edu.br
*Perguntas obrigatórias.

Informe seu nome (opcional):

Informe seu e-mail (opcional):

Qual foi o papel exercido por você durante o survey? *
[ ] Líder do Projeto
[ ] Prototipador
[ ] Programador

Marque seu grau de instrução: *
(é possível marcar mais de uma opção)
[ ] Técnico.
[ ] Graduação em andamento.
[ ] Graduado.
[ ] Especialização em andamento.
[ ] Especialista.
[ ] Mestrado em andamento.
[ ] Mestre.
[ ] Doutorado em andamento.
[ ] Doutor.

Quanto tempo de experiência no desenvolvimento de aplicações para TVD você possui?*
[ ] Menos de 6 meses;
[ ] Mais de 6 meses e menos de 1 ano;
[ ] Mais de 1 ano e menos de 2 anos;
[ ] Mais de 2 anos e menos de 3 anos;
[ ] Mais de 3 anos e menos de 4 anos;
[ ] Mais de 4 anos.

1) Avalie o quão adequado é o método XDTv para a atividade de COLETA de requisitos: **Muito inadequado 0 1 2 3 4 5 6 7 8 9 10 Muito adequado**
2) Avalie o quão adequado é o método XDTv para a atividade de ESPECIFICAÇÃO de requisitos: **Muito inadequado 0 1 2 3 4 5 6 7 8 9 10 Muito adequado**
3) Avalie o quão adequado é o método XDTv para a atividade de PLANEJAMENTO: **Muito inadequado 0 1 2 3 4 5 6 7 8 9 10 Muito adequado**
4) Avalie o quão adequado é o método XDTv para a atividade de IMPLEMENTAÇÃO: **Muito inadequado 0 1 2 3 4 5 6 7 8 9 10 Muito adequado**
5) Avalie o quão adequado é o CICLO DE VIDA do método XDTv: **Muito inadequado 0 1 2 3 4 5 6 7 8 9 10 Muito adequado**
6) Avalie o quão adequado são os TESTES DE ACEITAÇÃO no ciclo de vida do método XDTv: **Muito inadequado 0 1 2 3 4 5 6 7 8 9 10 Muito adequado**
7) Avalie o quão adequado é o MODELO DE PROCESSO do método XDTv: **Muito inadequado 0 1 2 3 4 5 6 7 8 9 10 Muito adequado**
8) Avalie o quão adequado é a PROGRAMAÇÃO em PAR para o desenvolvimento de aplicações de TVD: **Muito inadequado 0 1 2 3 4 5 6 7 8 9 10 Muito adequado**
9) Avalie o quão adequado é o artefato Protótipo de Aplicações para TV(PATV): **Muito inadequado 0 1 2 3 4 5 6 7 8 9 10 Muito adequado**
10) Avalie o quão adequado é o artefato Modelo de Fluxo e Sincronismo de Mídia (MFSM): **Muito inadequado 0 1 2 3 4 5 6 7 8 9 10 Muito adequado**
11) Avalie o grau de adequação do papel do LIDER do projeto e as atividades a ele atribuídas: **Muito inadequado 0 1 2 3 4 5 6 7 8 9 10 Muito adequado**
12) Avalie o grau de adequação do papel do PROTOTIPADOR e as atividades a ele atribuídas: **Muito inadequado 0 1 2 3 4 5 6 7 8 9 10 Muito adequado**
13) Avalie o grau de adequação de papel do PROGRAMADOR e as atividades a ele atribuídas: **Muito inadequado 0 1 2 3 4 5 6 7 8 9 10 Muito adequado**
14) Avalie o grau de adequação do papel do CLIENTE e as ATIVIDADES a ele atribuídas: **Muito inadequado 0 1 2 3 4 5 6 7 8 9 10 Muito adequado**
15) De forma geral, avalie o grau de adequação do método XDTV para o desenvolvimento de aplicações de TVD: **Muito inadequado 0 1 2 3 4 5 6 7 8 9 10 Muito adequado**
16) Avalie o quão o método XDTV contribui para tornar o desenvolvimento mais rápido: **Muito inadequado 0 1 2 3 4 5 6 7 8 9 10 Muito adequado**
17) Avalie o grau de adequação da filosofia ágil (independente do método adotado) para o desenvolvimento de aplicações para TVD: **Muito inadequado 0 1 2 3 4 5 6 7 8 9 10 Muito adequado**