UNIVERSIDADE FEDERAL DE ALAGOAS
INSTITUTO DE COMPUTAÇÃO
PROGRAMA DE PÓS GRADUAÇÃO EM INFORMÁTICA

Vanessa Pinheiro Rodrigues

Um Método para Apoiar a Engenharia de Requisitos de Qualidade que Envolvem Ajustes Dinâmicos do Software

Maceió - AL
2016

Vanessa Pinheiro Rodrigues

Um Método para Apoiar a Engenharia de Requisitos de Qualidade que Envolvem Ajustes Dinâmicos do Software

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal de Alagoas, como requisito para obtenção do grau de Mestre em Informática.

Orientador: Prof. Dr. Patrick H. S. Brito

Maceió - AL
2016

Catalogação na fonte
Universidade Federal de Alagoas
Biblioteca Central
Divisão de Tratamento Técnico
Bibliotecário Responsável Valter dos Santos Andrade

R696m Rodrigues, Vanessa Pinheiro.
Um método para apoiar a engenharia de requisitos de qualidade que envolvem Ajustes dinâmicos do software / Vanessa Pinheiro Rodrigues. – 2016.
55 f. : il.

Orientador: Patrick Henrique da Silva Brito.
Dissertação (mestrado em Modelagem Informática) - Universidade Federal de Alagoas. Instituto de Computação. Maceió, 2016.

Bibliografia: f. 49-50.
Anexo: f. 51- 55.

1. Engenharia de requisitos. 2. Requisitos não-funcionais. 3. Adaptação dinâmica. 4. Requisitos de qualidade. 5. Elicitação de requisitos. 6. Especificação de requisitos. I. Título.

CDU: 004.414.22

UNIVERSIDADE FEDERAL DE ALAGOAS/UFAL
Programa de Pós-Graduação em Informática – PpgI
Instituto de Computação
Campus A. C. Simões BR 104-Norte Km 14 BL 12 Tabuleiro do Martins
Maceió/AL - Brasil CEP: 57.072-970 | Telefone: (082) 3214-1401

Membros da Comissão Julgadora da Dissertação de Mestrado de Vanessa Pinheiro Rodrigues, intitulada: *"Um Método para Apoiar a Engenharia de Requisitos de Qualidade que Envolvem Ajustes Dinâmicos do Software"*, apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal de Alagoas em 13 de dezembro de 2016, às 14h15min, na Sala de Reuniões do Instituto de Computação da UFAL.

COMISSÃO JULGADORA

Prof. Dr. Patrick Henrique da Silva Brito
UFAL – Campus Arapiraca
Orientador

Prof. Dr. Alan Pedro da Silva
UFAL – Instituto de Computação
Examinador

Prof.ª Dr.ª Ariadne Maria Brito Rizzoni Carvalho
Unicamp – Instituto de Computação
Examinadora

AGRADECIMENTOS

Meus agradecimentos a todos os familiares, amigos, professores, que direta ou indiretamente contribuíram para a realização deste trabalho. Em especial, dedico meus agradecimentos:

- A Deus, por ter me dado força e saúde para chegar até aqui;
- Ao meu orientador, Prof. Dr. Patrick Brito, por ter aceitando a missão de me orientar mais uma vez e ajudar durante todo esse trabalho;
- Aos meus pais Cicero e Vanusa, por serem meus anjos aqui na terra;
- A minha irmã/filha Brenda, por todo incentivo;
- Aos Professores Dr. Alan Pedro e Drª Ariadne pelo acompanhamento na banca examinadora, sugestões e incentivo;
- Aos meus amigos e colegas que me acompanharam por toda essa jornada.

RESUMO

A evolução atual das plataformas de software e a adoção de serviços críticos de alta demanda mostra a grande dependência de softwares que possuem alta complexidade de desenvolvimento. Além do mais, as tendências atuais devem acentuar ainda mais a necessidade de garantir a qualidade do software. A partir de tais necessidades os requisitos de qualidade, considerados como não-funcionais, são considerados cada vez mais essenciais. A engenharia de requisitos tradicional ocorre de forma estática e com um foco fundamentalmente funcional, enquanto que sistemas com alta demanda de qualidade precisam se preocupar de forma sistemática com eventuais variações comportamentais dinâmicas, de forma a preservar restrições de qualidade. Um dos desafios para lidar com a adaptação dinâmica do software é o aumento da complexidade do seu projeto, aumentando a necessidade se ter uma preocupação sistemática com a arquitetura de software. Com base nas informações apresentadas e nos desafios da engenharia de requisitos, este trabalho propõe a construção de um processo para auxiliar nas fases de elicitação e especificação dos requisitos de qualidade que possuam impacto no comportamento do sistema em tempo de execução. Os resultados preliminares mostram que o processo proposto interferiu positivamente na qualidade dos requisitos especificados, beneficiando principalmente desenvolvedores menos experientes.

Palavras-chaves: Engenharia de requisitos, Adaptação dinâmica, requisitos não-funcionais, requisitos de qualidade, elicitação de requisitos, especificação de requisitos.

ABSTRACT

The current evolution of software platforms and the adoption of software in critical services show the increasing dependence of complex software systems. Moreover, current trends should further accentuate the need to ensure software quality. From these needs, the non-functional requirements, also known as quality requirements, are considered more and more as mandatory features. Traditional requirements engineering occurs in a static way with a fundamentally functional focus, while high quality systems must systematically concern themselves with possible dynamic behavioral variations in order to preserve quality constraints. One of the challenges to dealing with dynamic software adaptation is to increase the complexity of the project, increasing the need to have a systematic concern with the software architecture. Based on the information presented and the requirements engineering challenges, this work proposes a process to assist in the elicitation and specification phases of the quality requirements that have an impact on the behavior of the system at run time. The preliminary results show that the proposed process interfered positively in the quality of the specified requirements, benefiting mainly less experienced developers.

Key-words: Requirements engineering; dynamic adaptation; non-functional requirements; quality requirements; requirements elicitation; requirements analysis.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

1.1 PROBLEMÁTICA E MOTIVAÇÃO DA PESQUISA

A evolução atual das plataformas complexas e de serviços críticos de alta demanda expõe a disposição de se utilizar sistemas de software para desempenhar funcionalidades complexas e críticas. Além do mais, as tendências atuais devem acentuar a necessidade de garantir a qualidade do software. A partir de tais necessidades os requisitos não-funcionais (RNF) do software, são considerados cada vez mais essenciais. RNF são voltados à definição de propriedades e restrições do software, que podem afetar o sistema como um todo (SOMMERVILLE, 2007). Estes também são conhecidos como requisitos de qualidade (BOEHM; BROWN; KASPAR, 1978).

A engenharia de requisitos tradicional ocorre de forma estática e com um foco fundamentalmente funcional, enquanto que sistemas com alta demanda de qualidade precisam se preocupar de forma sistemática com eventuais variações comportamentais dinâmicas, de forma a preservar restrições de qualidade.

Um dos desafios para lidar com adaptações dinâmicas do software é o aumento da complexidade do seu projeto, aumentando a necessidade de se ter uma preocupação sistemática com a arquitetura de software. A arquitetura de software de um sistema representa elementos estruturais importantes de um software, favorecendo discussões entre as partes envolvidas no desenvolvimento e uso do sistema (*stakeholders*) (BASS; CLEMENTS; KAZMAN, 2012). Outro benefício da arquitetura de software, apontado na literatura, é o fato dela permitir o projeto e avaliação dos atributos de qualidade de um sistema de software de forma antecipada, isto é, em tempo de projeto (BASS; CLEMENTS; KAZMAN, 2012). Nesse sentido, a preocupação com os aspectos dinâmicos relacionados ao atendimento dos requisitos de qualidade do software precisam ser considerados sistematicamente, desde a elicitação e detalhamento (especificação) dos requisitos. Para promoção de sistemática, uma solução bem aceita atualmente é o uso de processos de gerenciamento de negócio (BPM[1]) (BRASIL, 2013).

BPM é uma disciplina gerencial que integra estratégias e objetivos de uma organização com expectativas e necessidades de clientes, por meio do foco em processo ponta a ponta. BPM engloba estratégias, objetivos, cultura, estruturas organizacionais, papéis, políticas, métodos e tecnologias para analisar, desenhar, implementar, gerenciar desempenho, transformar e estabelecer a governança de processos (BRASIL, 2013).

[1] Sigla do inglês *Business Process Management*.

1.2 OBJETIVO DO TRABALHO

Com base nas informações apresentadas e nos desafios da engenharia de requisitos, este trabalho propõe a construção de um processo para auxiliar nas fases de elicitação e especificação dos requisitos de qualidade que possuam impacto no comportamento do sistema em tempo de execução. A especificação do dinamismo proposto atua nas variações estruturais e funcionais do sistema em decorrência de mudanças de estado e de ambiente, que ocorrem em tempo de execução. Tais características são consideradas observáveis e precisam ser elicitadas através de atividades que serão anexadas ao método proposto. O processo tomará como base a preservação dos requisitos de qualidade do sistema, mesmo quando há mudanças dinâmicas de ambiente. Durante a especificação das adaptações dinâmicas, conflitos podem ser identificados. Nesses casos, o método proposto irá tratá-los utilizando técnicas de gestão de processos e metodologias ágeis. Para isso, as variações especificadas são classificadas em escala de prioridades, onde a mais prioritária prevalece em relação às demais.

As atividades do método proposto foram influenciadas pelo processo Scrum(SOARES, 2004) e pelo conceito de Linha de Produto de Software Dinâmica (DSPL[2]). O que motivou a escolha do Scrum é o fato dele ser um processo voltado ao acompanhamento de projetos de software e apresenta uma abordagem empírica que aplica algumas ideias da teoria de controle de processos industriais, reintroduzindo as ideias de flexibilidade, adaptabilidade e produtividade (SOARES, 2004).

O conceito de linha de produtos de software (SPL[3]), traz para a produção de software em massa a preocupação sistemática com a customização de produtos de software a partir do reúso de componentes de software de forma planejada, automatizada e sistemática, sem que sejam perdidas as vantagens da produção em massa. Estendendo estes princípios, as DSPLs possibilitam que os pontos de variação definidos na SPL sejam alterados em tempo de execução.

DSPL é uma área emergente da engenharia de software, que pode facilitar o gerenciamento das reconfigurações do sistema a partir de mudanças no ambiente de execução, incluindo o estado do sistema. Mas por ser uma área de pesquisa relativamente recente, ainda encontra-se vários desafios em aberto, como por exemplo a adequação dos processos de engenharia de requisitos para especificar os cenários de variações dinâmicas que podem ocorrer em uma família de sistemas.

Segundo Carvalho (CARVALHO, 2013), as tarefas mais árduas para engenharia de requisitos de uma DSPL é a identificação e modelagem das possibilidades de configuração consistentes que um sistema pode atingir sem causar danos aos usuários. Observando

[2] Sigla do inglês *Dynamic Software Product Line*
[3] do inglês *Software Product Line*

este cenário, sentiu-se a necessidade de uma ter um processo que contribuísse para o desenvolvedor identificar os cenários de adaptação do sistema. Como ponto de partida, o trabalho proposto tem como foco os requisitos de qualidade, seu monitoramento e a especificação de ações a serem tomadas em estados da execução do sistema considerados cruciais. Tais ações podem ser desde o envio de mensagens de alerta, a especificação de adaptações ou até mesmo o desligamento do software.

1.2.1 Objetivo Geral

O objetivo geral desta pesquisa é apoiar o desenvolvedor de software nas tarefas de elicitação e especificação dos requisitos de qualidade, cuja satisfação possa implicar em variações dinâmicas no comportamento do software.

1.2.2 Objetivos Específicos

1. Propor um método para apoiar a engenharia de requisitos de qualidade que contemple a especificação de reconfigurações dinâmicas do sistema, tendo em vista a satisfação dos requisitos;

2. Adaptar um processo de engenharia de requisitos existente, voltado aos requisitos funcionais, para incorporar o método desenvolvido;

3. Avaliar o processo adaptado no contexto de um experimento.

1.3 VISÃO GERAL DA AVALIAÇÃO E RESULTADOS

O processo proposto foi avaliado em um experimento envolvendo 39 voluntários, distribuídos em três grupos de 13 voluntários cada. Cada grupo utilizou um processo diferente, identificado durante a fase de revisão da literatura.

Os resultados comparativos preliminares mostraram que o processo proposto interferiu positivamente na qualidade e detalhamento dos requisitos de qualidade especificados, beneficiando principalmente desenvolvedores menos experientes.

De forma complementar, foi percebido que o trabalho desenvolvido tem potencial para contribuir na busca por um processo mais geral, voltado para a engenharia de requisitos de DSPLs, podendo assim contribuir na resolução do problema de adaptação dinâmica de requisitos funcionais e não-funcionais em tempo de execução, ao longo do ciclo de vida do software.

1.4 ESTRUTURA DO DOCUMENTO

O restante deste documento está organizado como segue. O Capítulo 2 expõe os principais conceitos utilizados nessa proposta. O Capítulo 3 traz alguns trabalhos que mostram alguma similaridade com a abordagem proposta, destacando a diferença entre eles. O Capítulo 4 Mostra o processo proposto, seus conceitos, além de detalhes sobre os passos que devem ser seguidos para sua aplicação. O Capítulo 5 apresenta como foi realizado o experimento para validar o processo proposto, assim como os resultados obtidos. Finalmente, o Capítulo 6 apresenta as considerações finais desse trabalho, contribuições, limitações e direcionamentos para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 LINHAS DE PRODUTO DE SOFTWARE DINÂMICAS

O conceito de Linha de produto de software (SPL[1]) traz para a produção de software em massa a possibilidade de customização de produtos a partir do reúso de componentes de software de forma planejada, automatizada e sistemática, sem que sejam perdidas as vantagens da produção em massa.

> Uma linha de produtos pode ser vista como sendo um conjunto de produtos em um portfólio de produtos de um fabricante que compartilham semelhanças substanciais e que são, idealmente, criados a partir de um conjunto de partes reutilizáveis. Em vez de proporcionar um único produto padronizado, fabricantes são capaz de oferecer vários produtos similares (APEL et al., 2013).

A abordagem de desenvolvimento de uma SPL é diferente das abordagens tradicionais, pois tem-se olhado para a variedade de funcionalidades que são similares mas não idênticas. Com isso, para obter o sucesso dessa abordagem é necessário definir foco na análise do domínio. A variabilidade entre os requisitos desses cenários trazem o conceito de família de produtos. Uma família de produtos é uma coleção de produtos semelhantes, que compartilham a maioria das funcionalidades. Assim, existem inúmeros requisitos que são comuns em toda a família, mas outros são únicos para cada produto. Portanto, um processo de engenharia de requisitos bem estabelecido é essencial para qualquer linha de produtos (EBERLEIN, 2001).

A variabilidade de uma SPL é a capacidade que um sistema possui de ser eficientemente aumentado, alterado, personalizado ou configurado para o uso em um contexto particular (ASIKAINEN; MÄNNISTÖ; SOININEN, 2007). Através dos pontos de variação definidos, a SPL ganha flexibilidade para cumprir outros requisitos exigidos. Tais variações são definidas em termos das características[2]. do sistema. Uma característica pode ser definida como uma propriedade relevante do sistema (FERNANDES; WERNER; MURTA, 2008).

O desenvolvimento de uma SPL é dividido em dois principais processos que são os processos de engenharia de domínio e engenharia de aplicação. O primeiro trata da forma como será estabelecido o reúso, definindo as partes comuns e variáveis. Já o processo de engenharia de aplicação trata de construir diversos produtos a partir das variabilidades estabelecidas na engenharia de domínio. No contexto do presente trabalho, a principal motivação para o estudo de SPL relaciona-se à fase de análise de requisitos, que está inclusa no processo de engenharia de domínio.

[1] Sigla do inglês *Software Product Line*
[2] Do inglês *features*

Alguns conceitos inerentes à definição de variabilidades na execução da engenharia de domínio são (OLIVEIRA, 2006):

- **Pontos de variação**: são as características que refletem as características que podem variar no domínio. As possibilidades são definidas por meio das variantes;
- **Variantes**: são características que atuam como possibilidades para se configurar um ponto de variação;
- **Invariantes**[3]: são as características fixas, que representam elementos comuns e não configuráveis em um domínio.
- *Características Opcionais*: descrevem características que podem ou não estar presentes em produtos desenvolvidos em uma SPL, ou em aplicações instanciadas a partir de um domínio;
- **Características Alternativas**: descrevem características que representam possibilidades de escolha, permitindo a seleção de uma ou várias características.
- **Características Mutuamente Exclusivas**: descrevem características que representam possibilidades de escolha, permitindo a seleção de apenas uma característica do grupo.

O conceito de Linha de produto de software dinâmica (DSPL[4]) estende a definição de SPL para que a escolha das variabilidades possa ocorrer em tempo de execução. Sendo assim, DSPLs são voltadas aos sistemas que necessitam de modificações e adaptações ao ambiente, também conhecidos como sistemas adaptativos (PARRA, 2011; BENCOMO; LEE; HALLSTEINSEN, 2010).

Para viabilizar uma DSPL é necessário considerar cenários envolvendo adaptação dinâmica tanto no funcionamento interno dos componentes, quanto da configuração arquitetural, conforme as mudanças de requisitos (COPETTI, 2010). Contudo, o tratamento de uma DSPL pode reduzir consideravelmente a complexidade de se lidar com o gerenciamento de reconfiguração dinâmica, uma vez que boa parte do tratamento sistemático dessa questão pode ser herdado do conceito de SPL. Porém, a adaptação dinâmica é considerada um problema de primeira classe que deve ser obtido por conta do longo ciclo de vida de software.

Um das tarefas mais árduas para a DSPL é a identificação e modelagem de todas as possibilidades de configuração consistentes que um sistema pode atingir sem causar danos aos usuários. Para um Sistema Adaptativo, cada produto da DSPL corresponde a uma configuração possível que o sistema pode alcançar enquanto está em execução. Com essa perspectiva, um sistema pode ser modelado definindo os elementos que variam de uma

[3] Invariantes são também conhecidos como elementos mandatórios.
[4] Sigla do inglês *Dynamic Software Product Line*

configuração para outra, ou seja, entre as transições. Estes elementos correspondem às variabilidades dinâmicas do software (COPETTI, 2010).

Vale salientar que mesmo no contexto de uma DLPS, o processo de engenharia de domínio é executada de forma estática; apenas as atividades da engenharia de aplicação são realizadas em tempo de execução. Para isso, possíveis mudanças nos objetivos da DSPL precisam ser definidos por um especialista no domínio.

Apesar da necessidade conhecida de se antecipar as variações dinâmicas do software, pouco trabalho foi identificado que aborda esse assunto (COPETTI, 2010). Ressalta que mudanças dos requisitos em uma DSPL também podem ser ocasionadas por um usuário durante a execução do sistema. Nesses casos, o gatilho da adaptação é manual e denominada dirigida por usuário[5]. Porém, o trabalho proposto na presente dissertação pretende focar também condições ambientais que possam engatilhar adaptações automaticamente, isto é, sem a interferência do usuário.

2.2 ENGENHARIA DE REQUISITOS PARA DSPL

Nesta seção serão tratados conceitos relacionados à engenharia de requisitos de software e sua adequação e dificuldades para aplicação em uma DSPL.

2.2.0.1 Engenharia de Requisitos de Software

Engenharia de requisito é o processo para compreender e definir quais serviços são necessários e identificar as restrições de operação e de desenvolvimento do sistema (SOMMERVILLE, 2007). A engenharia de requisitos é uma fase particularmente crítica do processo de desenvolvimento de software, pois os erros cometidos nesse estágio conduzem inevitavelmente a problemas posteriores no projeto e na implementação do sistema, sendo identificados tardiamente (SOMMERVILLE, 2007). Vale salientar ainda que a engenharia de requisitos deve ser adaptada às necessidades do processo, do projeto, do produto e das pessoas que estão realizando o trabalho (PRESSMAN, 2011).

O processo de engenharia de requisitos é composto de quatro subprocessos de alto nível que são (SOMMERVILLE, 2007):

1. estudo de viabilidade;
2. elicitação e análise de requisitos[6];
3. especificação;
4. validação.

[5] Do inglês *User-Driven Adaptation Systems*
[6] Essa etapa também é conhecida como: obtenção e compreensão de requisitos

Cada subprocesso gera documentação e verificação de requisitos.

Para construção dos modelos de requisitos, sempre que possível é recomendado o uso da notação UML[7], que se tornou padrão para modelagem de programas orientados a objetos (PRESSMAN, 2011).

2.2.1 Engenharia de Requisitos para DSPL

As técnicas utilizadas na fase de análise de domínio de uma DSPL visam coletar informações sobre o domínio e explicitar suas características similares e diferentes. Porém, como citado anteriormente, uma das tarefas mais árduas para a DSPL é a identificação e modelagem de todas as possibilidades de configuração consistentes que um sistema pode atingir. Para um sistema adaptativo, cada produto da DSPL corresponde a uma configuração possível que o sistema pode alcançar em tempo de execução. Com essa perspectiva, um sistema pode ser modelado definindo os elementos que variam de uma configuração para outra, ou seja, as transições entre produtos (CARVALHO, 2013).

Por ser uma abordagem recente da SPL não há na literatura muito sobre o processo de engenharia de requisitos ou especificação de domínio de uma DSPL. Porém, nos trabalhos revisados ficam evidentes as dificuldades de representar alguns pontos críticos e pontos de variação que devem ser monitorados e/ou alterados em tempo de execução. Por exemplo, apesar de não haver propostas sistemáticas de como fazer, sabe-se que durante a especificação do domínio é necessário considerar tanto adaptações internas aos componentes, quanto adaptações estruturais, que envolvam reconfiguração da arquitetura de software (PESSOA, 2014).

O trabalho apresentado na presente dissertação pode ser visto como um passo no sentido de alcançar um processo que atenda as demandas da engenharia de domínio em uma DSPL.

2.3 MODELO DE PROCESSO DE NEGÓCIO

O gerenciamento de processo de negócio (BPM[8]) conduz a uma nova forma de visualizar os processos realizados por uma instituição visando torná-los mais eficazes e eficientes. O CBOK traz a seguinte definição para BPM (BRASIL, 2013):

> "BPM é uma disciplina gerencial que integra estratégias e objetivos de uma organização com expectativas e necessidades de clientes, por meio do foco em processo ponta a ponta. BPM engloba estratégias, objetivos, cultura, estruturas organizacionais, papéis, políticas, métodos e tecnologias para analisar, desenhar, implementar, gerenciar desempenho, transformar e estabelecer a governança de processos."

[7] Sigla do inglês *Unified Modeling Language*
[8] sigla do inglês *Business Process Management*

O BPM é utilizado em várias áreas do conhecimento, necessitando de adequação à finalidade do domínio, o que deve ser realizado por um especialista. Por exemplo, a área de tecnologia da informação normalmente utiliza BPM para descrever as capacidades de um produto ou tecnologia em particular (BRASIL, 2013).

A valorização de BPM se deve também ao fato dele agregar valor tanto à organização, quanto aos seus projetos(CAPOTE, 2012). Com a identificação dos processos e a importância estratégica de cada um deles para a organização, BPM promove maior vantagem competitiva, pois possibilita ao administrador e gerente de projetos conhecer melhor o processo, a ponto de identificar mais facilmente pontos que podem ser melhorados (CAPOTE, 2012). Outras vantagens também são apontadas na literatura especializada, tais como (CAPOTE, 2012; BRASIL, 2013): transparêcia em todas as etapas desenvolvidas no projeto, o aumento da produtividade, possível redução de custos e maior controle administrativo das atividades. Isso acontece porque o BPM não se limita apenas a conhecer como o processo é realizado, mas também orienta para as respostas de perguntas estratégicas, utilizando uma técnica conhecida como 5W2h[9] (BRASIL, 2013). Essa é uma técnica ágil e simples, comumente usada em projetos tradicionais.

2.3.1 Notação de Modelagem de Processos de Negócio

A Notação de Modelagem de Processos de Negócio (BPMN)[10] é uma notação gráfica que possui um conjunto de símbolos robustos para representar processos de negócios através de diagramas contendo uma representação gráfica do fluxo sequencial e do controle lógico de um conjunto de atividades. Seu objetivo é apoiar o uso de BPM por usuários especialistas ou não especialistas.

O propósito da modelagem é criar uma representação do processo de maneira completa e precisa sobre seu funcionamento. Por esse motivo, o nível de detalhamento e o tipo específico de modelo tem como base o que é esperado da execução do processo (BRASIL, 2013). Para isso, BPMN define um conjunto padronizado de símbolos e regras que determinam o significado desses símbolos (BRASIL, 2013). Como na maioria das notações, os símbolos descrevem relacionamentos claramente definidos, tais como fluxo de atividades e ordem de precedência (BRASIL, 2013).

Nesta notação temos um modelo composto por uma ou mais *swimlanes*, que tem como finalidade organizar e definir o escopo das atividades do processo através de um diagrama. A organização visual das *swimlane* se assemelha a uma piscina delimitada por suas raias. Nessa analogia, a piscina seria o processo e cada raia uma *swimlane*. *Swimlanes* englobam, normalmente, atividades executadas por um determinado papel ou usuário, também conhecido como ator.

[9] Sigla do inglês *What, Where, When, Why, hoW, How much and Who*
[10] Sigla do inglês *Business Process Modelling Language*.

Os processos de negócio modelados com PBMN são normalmente gerenciados por sistemas de software especializados, denominados sistemas de gerenciamento de processos de negócio (BPMS[11]). Tais sistemas são responsáveis pela automação, controle e monitoração das atividades e das tarefas executadas em um processo. Eles gerenciam desde a sequência de atividades a serem executadas, quanto o estado de execução de cada uma delas, e quem as executou (ator).

A ferramenta escolhida para a modelagem do BPMN foi o *Together Workflow Editor* (TWE) (PROJECT, 1996). Trata-se de um editor gráfico para criar e editar processos de negócio e contempla todos os elementos e componentes utilizados em BPMN, dando assim ao usuário uma visão mais intuitiva do processo. Os arquivos são gravados no formato XPDL, que se tornou um padrão bem aceito na comunidade científica e empresarial (PROJECT, 1996).

2.4 O PROCESSO SCRUM

A engenharia de software voltada ao desenvolvimento ágil adota uma filosofia combinada com um conjunto de princípios de desenvolvimento presentes no manifesto ágil (ÁGIL, 2011). A filosofia defende a satisfação do cliente e a entrega de incrementos sucessivos; equipes de projetos pequenas e altamente motivadas; métodos informais; artefatos de engenharia de software mínimos e, acima de tudo, simplicidade no desenvolvimento geral. Os princípios de desenvolvimento priorizam a entrega mais que a análise e projeto (embora essas atividades não sejam desencorajadas); também priorizam a comunicação ativa e contínua entre desenvolvedores e clientes (PRESSMAN, 2011).

Entre as metodologias ágeis, um processo de destaque é o Scrum, voltado para gestão e planejamento de projetos de software. Ele foi escolhido como base para o trabalho proposto pois segundo seu autor, o Scrum não é um processo previsível, uma vez que não define o que fazer em todas as circunstâncias (SCHWABER, 2004). Dessa forma, o Scrum pode ser usado em trabalhos complexos nos quais não é possível prever tudo o que irá ocorrer, oferecendo um *framework* e um conjunto de práticas de acompanhamento constante dos trabalhos. Isso permite aos praticantes do Scrum tomar conhecimento do que está acontecendo ao longo do projeto e fazer os devidos ajustes para manter o projeto se movendo ao longo do tempo, visando alcançar os seus objetivos.

O Scrum baseia-se em algumas características como a flexibilidade dos resultados e prazos, times pequenos, revisões frequentes, iterações curtas e colaboração (SOARES, 2004).

O primeiro artefato criado no Scrum é o Backlog do produto, que contém uma lista de itens priorizados que incluem tudo o que precisa ser realizado, que possa ser associado com valor de negócio, para a finalização do projeto; sejam eles requisitos funcionais ou

[11] Sigla do inglês *Business Process Management Systems*.

não (PEREIRA; TORREÃO; MARÇAL, 2007). A partir do Backlog do produto os requisitos podem ser priorizados.

Os papéis e responsabilidades do Scrum são bem definidos e são caracterizados pelos seguintes papeis (SCHWABER, 2004):

- ***Product Owner*** - Este membro do time geralmente representa o cliente. Ele define quais são os requisitos e qual é o grau de importância e prioridade de cada um deles. Este membro necessita conhecer muito bem as regras de negócios do cliente, de forma que ele possa tirar qualquer dúvida que o time possa ter em relação às funcionalidades do produto;

- ***Scrum Master*** - O *Scrum master* trabalha para que o processo Scrum aconteça e para que não existam impedimentos para os membros da equipe realizarem seus trabalhos. Seu principal dever é remover os obstáculos apontados nas reuniões diárias, como duração máxima de 15 minutos, de modo que os desenvolvedores se concentrem apenas nas questões técnicas. Esses obstáculos são colocados em uma lista chamada de *Backlog* de Impedimentos, que fica à vista de todos;

- ***Scrum Team*** - Grupo de desenvolvimento Multi-funcional que possui até 10 membros. Seleciona, entre os itens priorizados, os que irão ser executados durante cada iteração, denominada *sprint*. Tem liberdade de tomar as decisões de projeto relacionadas à *sprint*.

Cada *sprint* tem duração entre 2 e 4 semanas e no fim de cada uma delas deve ser entregue um produto definido no inicio da interação.

O último artefato do Scrum é o gráfico *burndown*. Trata-se de uma representação gráfica do trabalho restante em comparação com o trabalho já realizado. Geralmente, coloca-se a quantidade de trabalho no eixo vertical e o tempo no eixo horizontal. Ele é muito útil para identificar potenciais atrasos. No fim de cada *sprint* é realizada uma reunião de revisão, onde é mostrado o produto gerado e acontece a sua validação, tendo em vista o objetivo a ser alcançado. Ao final de todas as *sprints*, é recomendado haver uma reunião de retrospectiva, onde podem ser expostas as lições aprendidas durante o processo.

3 TRABALHOS RELACIONADOS

Os trabalhos que mais se assemelham à pesquisa desenvolvida nesta dissertação de mestrado foram agrupados em duas categorias: (1) Engenharia de requisitos de linhas de produtos de software dinâmicas; e (2) Abordagens envolvendo dinamismo na gestão de processos de negócio.

3.0.1 Engenharia de requisitos de linhas de produtos de software dinâmicas

O trabalho de (TALIB et al., 2010) busca mostrar um maior intensidade o dinamismo, automação e evolução que possa ser utilizado na DAS e nas DSPL. Ele trata dinamismo como o grau que uma alteração pode ser aplicada ao sistema em tempo de execução sem que haja intervenção humana. A abordagem proposta tenta adequar a evolução das DAS e a potência de automatização das DSPL para a situação.

Para expressar a arquitetura de software auto-organizadas baseadas em regras utiliza-se uma linguagem específica de domínio chamada de ROAD(*Role Adaptative Design*). A arquitetura terá um modelo a representação de uma linha de produtos e um modelo de tempo de execução que pode ser alterado em tempo de execução. O próprio modelo é simples, em que ele constitui alguns conceitos, mas é suficientemente poderoso para capturar aspectos detalhados da estrutura e do comportamento do sistema.

Modelos de recurso seriam usados para caracterizar a estrutura do sistema e o comportamento em um nível mais alto. Definindo os mapeamentos do modelo de recurso para o modelo arquitetural, pode-se automatizar a síntese de configurações de produto. Conhecendo essas configurações do produto, a linha de produtos permitirá associar os componentes de arquitetura. A evolução será auxiliada por uma interface de gerenciamento de tempo de execução que permitirá alterações de tempo de execução na linha de produtos.

Apesar de nosso trabalho também tratar de variações dinâmicas do sistema, ele possui um foco nos requisitos de qualidade do software. Além disso, o trabalho de (TALIB et al., 2010) possui maior foco no mapeamento entre as variabilidades já especificadas e regras arquiteturais, enquanto o nosso trabalho possui foco na engenharia de requisitos em si.

SoProL-ws é proposto por (QUEIROZ, 2009) com o objetivo de reduzir custos e prazos de desenvolvimento de uma LPS e facilitar a sua manutenção, evolução e derivação de seus membros. Esse tem como foco a fase de engenharia de domínio, em que desenvolve-se uma arquitetura de LPS baseada em serviços, a partir da qual pode-se derivar produtos na fase de engenharia de aplicações. O processo proposto utiliza o processo de linha de produtos que é composto pelas duas fases a seguir:

Engenharia de domínio trata da Modelagem com o uso da UML adaptada para as

necessidades de LPS e web services, modelagem das features (características), definição dos serviços, análise das variabilidades em processos de negócios e em serviços, definição da arquitetura da linha, implementação de web services reutilizaveis e construção de um gerador de aplicações para a derivação dos produtos da linha.

Engenharia de aplicações trata da Configuração da modelagem de um membro (sistema alvo) da linha de produtos a partir da modelagem da linha. O engenheiro de aplicações seleciona as features desejadas para o membro da linha. A partir das features selecionadas, adapta-se a arquitetura da linha para o sistema alvo e determina-se quais serviços são necessários para a configuração do sistema.

A atividade de engenharia de requisitos é composta por algumas fases que são a Elicitação de requisitos, com a Identificação dos processos de negócios e suas variabilidades, modelagem de caso de uso e Modelagem de características. A outra fase do processo é a Atividade de Análise e projeto com a modelagem de estática, identificação dos Web services, Modelagem de navegação de interface, diagrama de comunicação, projeto de classes, definição de arquitetura e projeto de banco de dados. A metodologia ainda é composta por implementação, teste e manual para Geração dos Membros da LPS.

O método de (QUEIROZ, 2009) contribuiu no desenvolvimento de LPS baseadas em Web services, guiou o desenvolvimento de LPS com arquitetura orientadas a serviços. O principal diferencial do trabalho proposto na presente dissertação é seu foco voltado para as variações dinâmicas das linhas de produtos de softwares (DSPL); além de tais variações serem voltadas à satisfação de requisitos de qualidade.

O artigo apresentado em (CAPILLA et al., 2014) fornece uma visão geral do estado da arte e principais desafios relativos ao projeto de variabilidade de tempo de execução no contexto da DSPL. Apesar do trabalho não apresentar diretrizes para apoiar a fase de engenharia de requisitos de DSPLs, conhecer as soluções de projeto que realizam as reconfigurações dinâmicas é um passo importante para a definição da sistemática.

3.0.2 Abordagens envolvendo dinamismo na gestão de processos de negócio

PL4BPM (ROCHA, 2012) define um processo de gerenciamento de negócios voltado ao desenvolvimento de linha de produtos de software. O método proposto possui foco em atividades de gestão de projetos de softwares orientados a serviço. O DynPL4BPM (ROCHA, 2012) é uma evolução desse trabalho, que contempla aspectos dinâmicos da gestão de processos por meio da composição e gestão de serviços. Porém, o método proposto tem foco gerencial e não contempla atividades de desenvolvimento, tais como engenharia de requisitos.

Abordagem de (RAMOS, 2014) propõe um processo de suporte, que melhore a capacidade efetiva de realizar os processos de negócio primários na instituição envolvendo

ajustes dinâmicos. Processos de suporte são formalmente definidos e dão suporte aos processos primários, ou seja, são processos que não interagem diretamente com os clientes finais. Apesar do trabalho proposto não possuir um foco no desenvolvimento de DSPLs, características utilizadas na definição de características dinâmicas dos modelos de processo de negócio podem ajudar na definição do método proposto nesta dissertação, que é voltado à elicitação e especificação de requisitos de DSPLs.

O trabalho de (ASADI et al., 2012) aborda diversos métodos Orientados a Features para modelagem de linhas de produto de software e os avaliando através de um conjunto de critérios estabelecidos, entre eles estão os métodos Feature-Oriented Domain Analysis (FODA), Feature-Oriented Reuse Method (FORM) , Cardinality-based Feature Modeling (CBFM), Goal and Scenario Based Domain requirements analysis, Goal-driven software product line engineering, AoURN-based Software Product Line, FeatuRSEB, Product Line Use-case for Software and System engineering (PLUSS), AMPLE.

Os critérios de avaliação foram divididos em três áreas coberturas que são a engenharia de requisitos, cobertura de variabilidade e Análise de comunalidade e cobertura do suporte de ferramentas. A área de destaque é a engenharia de requisitos. Para a avaliação foram utilizados os critérios de usados pela Literatura de engenharia de requisitos para representar requisitos. Entre os artefatos existentes, selecionamos os artefatos mais comumente usados, ou seja, modelo de objetivo, modelo de caso de uso, modelo baseado em cenário, e modelo não-funcional como critérios. Além disso, de acordo com Documentos de engenharia de requisitos, as necessidades dos requisitos funcionais e não-funcionais, bem como sobre os requisitos funcionais e não funcionais.

Este trabalho mostra que os métodos de engenharia de requisitos abrangem mais técnicas para elicitação de requisitos funcionais, enquanto os requisitos não-funcionais e variabilidades não receberam atenções dos métodos mais populares. Mostrando a deficiência dos métodos, pois os requisitos não-funcionais tem grandes importância no desenvolvimento da linha de produto de software e podem entrar em conflito entre eles.

4 UM MÉTODO PARA APOIAR A ENGENHARIA DE REQUISITOS DE QUALIDADE QUE ENVOLVEM AJUSTES DINÂMICOS DO SOFTWARE

A abordagem proposta traz um processo para apoiar a engenharia de requisitos com foco na elicitação e especificação de variações dinâmicas no software, tendo em vista a satisfação de requisitos de qualidade. Durante a especificação das adaptações, conflitos podem ser identificados e devidamente tratados através de serialização.

A Figura 1 traz uma visão geral do método que está sendo proposto, de acordo com a notação de BPMN, apresentada na Seção 2.3.1.

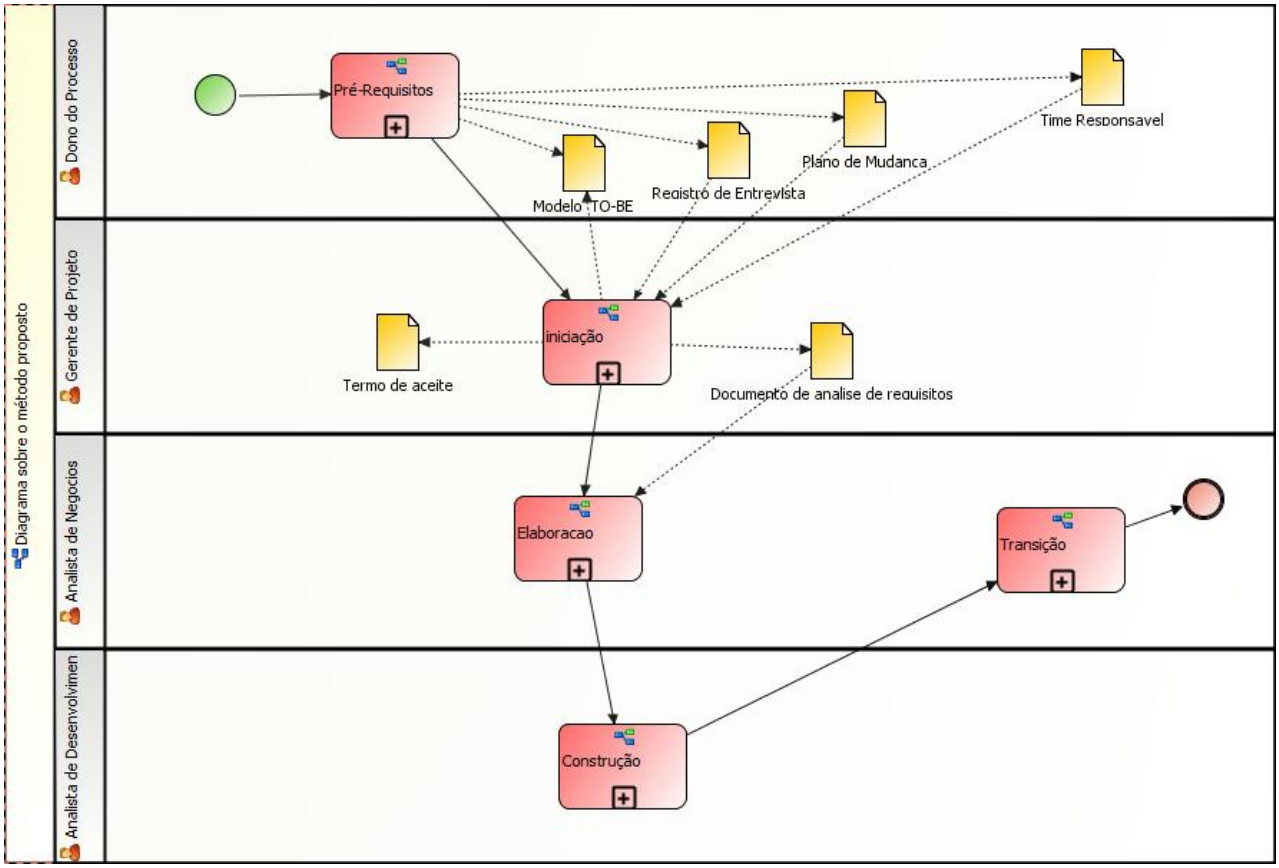


Figura 1 – Visão Geral do processo

O método proposto inicia-se com a atividade de pré-requisitos, onde terá como participantes os donos do processo. Estes podem ser compostos pela diretoria, os participantes da área de negócio e do escritório de processo. Nesta fase, busca-se definir a missão corporativa, analisar a situação atual, formular objetivos e estratégias. Também é um local oportuno para aperfeiçoar o processo em si, a partir de experiências anteriores. A análise é realizada levando

em consideração as condições da empresa e a evolução esperada. O fluxo detalhado de tarefas da atividade de pré-requisitos é apresentado na Figura 2.

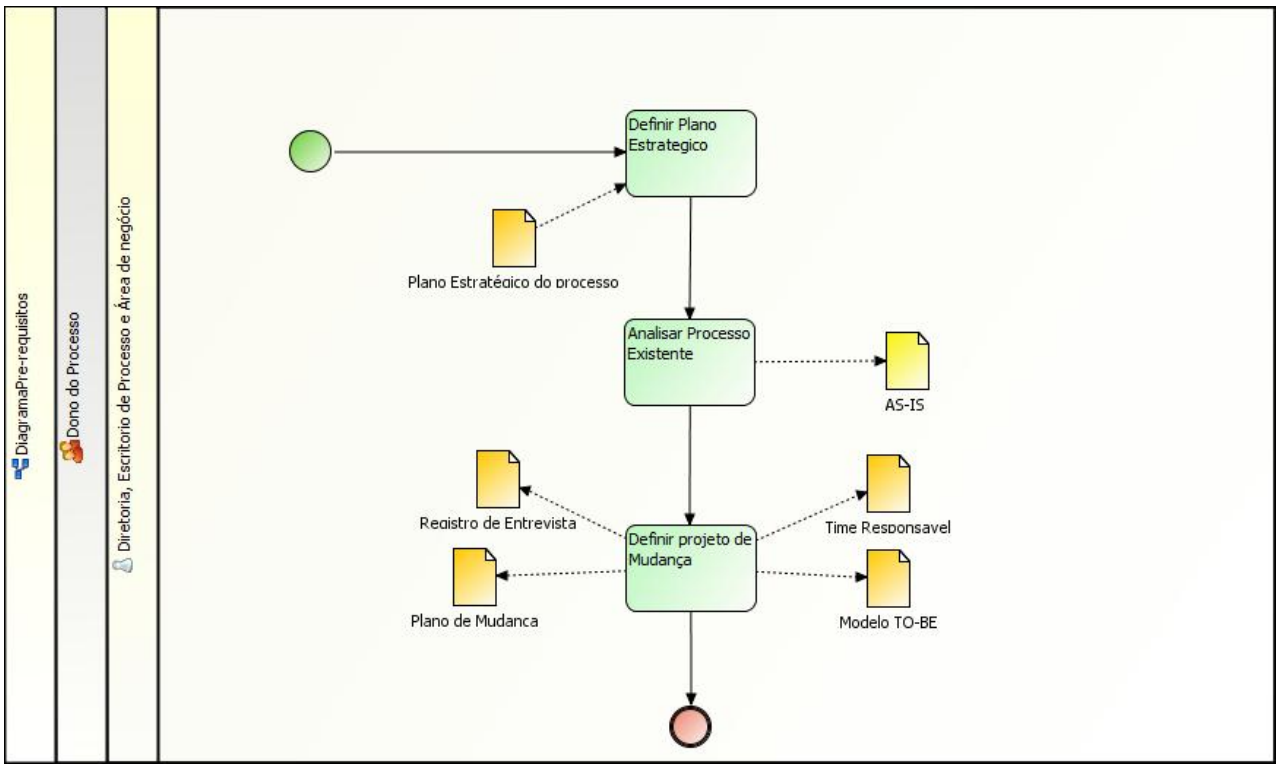


Figura 2 – Diagrama sobre a fase de pré-requisitos do processo

A primeira tarefa da atividade de pré-requisitos inicia-se com a entrada do plano estratégico da organização, onde será possível obter a missão corporativa, formular objetivos e estratégias para criação do modelo *AS-IS*. O modelo *AS-IS* aborda o levantamento e documentação da situação atual do processo. Além de, mostrar os problemas, fragilidade e oportunidades de melhorias. As técnicas mais usadas para coletar as informações para esse modelo são as de entrevista, observações, questionários e reuniões JAD. Para a verificação das informações mais importantes recomenda-se a utilização da técnica 5w2H[1](BRASIL, 2013) (O que, por que, onde, quando, por quem, como e quanto).

Segundo o CBOK (BRASIL, 2013), o modelo *TO-BE* representa o estado futuro de processo de negócio. Visa produzir alternativas para o estado atual e incorpora melhores práticas, redesenho, reengenharia e/ou mudança de paradigma. Este será construído baseado no modelo *AS-IS* e na ideia de que este deve ser desafiado para incorporar as mudanças. Todo o modelo AS-IS deve ser questionado, pois tudo pode ser oportunidade de simplificar o processo sem prejudicar nenhum dos níveis anteriores, trazendo assim importantes benefícios. Os questionamentos da técnica 5W2H podem ser complementados com questões adicionais sugeridas pelo CBOK (BRASIL, 2013):

[1] Sigla do inglês: *What, Why, Where, When, Who, How and How much*

- Qual é o propósito do processo e subprocesso e respectivas conexões com função, atividade e fluxo de trabalho?
- É redundante ou semelhante a outra atividade que já está sendo realizada?
- Quais são os problemas, questões de qualidade e governança e por que estão ocorrendo?
- por que esse passo é necessário?
- Qual é o seu propósito?
- Onde deve ser feito?
- Quando deve ser feito?
- Quem está mais bem qualificado para executá-lo?
- Está devidamente apoiado por automação?
- Quais são seus principais problemas?
- Como os problemas podem ser eliminados?
- Como a operação pode ser realizada com maior eficácia possível(apenas fazer o que tem de ser feito)?
- Como a operação pode ser realizada com maior eficiência possível(eliminando as tarefas desnecessárias)?
- Como desperdícios percebidos podem ser removidos?
- Existem padrões que precisam ser atendidos?
- Como podemos monitorar a atividade e assegurar que alvos de desempenho sejam atingidos?
- Quais são os fatores limitantes sobre as mudanças em processo, subprocesso, função, fluxo de trabalho, atividade ou tarefa?

A partir do conhecimento sobre o processo obtido através do modelo AS-IS as mudanças no processo podem ser planejadas com um esforço proativo, organizadas e documentadas em um plano de mudança. O CBOK (BRASIL, 2013) cita que a gestão da mudança está associada à modificação ou transformação da organização, visando manter ou melhorar sua eficácia por meio do gerenciamento, no que tange aos processos de mudança. O gerenciamento de mudança pode seguir uma estratégia de mudança de Kotter (BRASIL, 2013), que pode foi incorporado ao processo através das práticas do CBOK (BRASIL, 2013), que apresenta oito passos para as mudanças em uma organização:

Passo 1: Estabelecer um grande senso de urgência;
Passo 2: Formar uma aliança de orientação forte o suficiente;
Passo 3: Criar visão;
Passo 4: Comunicar a visão;
Passo 5: Investir os funcionários de empowerment para que agir com relação à visão;
Passo 6: Planejar e criar vitórias de curto prazo;
Passo 7: Consolidar as melhorias e produzir mais mudanças;
Passo 8: Institucionalizar novas abordagens.

Os artefatos de saída da atividade de pré-requisitos é o conjunto de documentos gerados, que permitem conhecer a organização, seus sistemas e processos, podendo assim trazer mudanças sem causar danos aos métodos já estabelecidos.

A partir do conhecimento obtido sobre a organização, entra-se na Atividade de iniciação (ver Figura 1). Esta trata de analisar o processo de negócio que foi revisado, planejar interações, definir variáveis de ambiente a serem monitoradas e iniciar a elicitação de requisitos. Os requisitos são organizados, priorizados, especificados, modelados e verificados. A Figura 3 mostra o fluxo interno da atividade de iniciação.

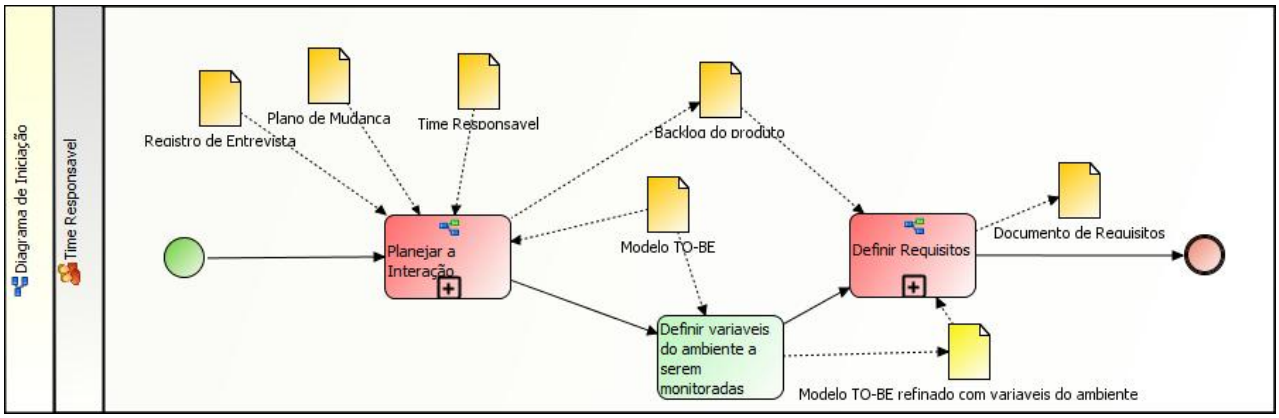


Figura 3 – Diagrama sobre a fase Iniciação do processo

O método proposto possui uma dinâmica baseada no processo Scrum, apresentado na Seção 2.4. Sendo assim, a especificação deve acontecer de forma iterativa e incremental, onde cada iteração é denominada *sprint*. Durante o planejamento dos *sprints*, deve-se identificar os riscos e pendências.

A primeira tarefa de iniciação é chamada de Planejar a Interação. Como o nome já descreve, a atividade a ser realizada consiste na compreensão da dinâmica do processo, que inclui suas atividades, quem deve realizá-las e o que deve ser entregue ao chegar a seu fim. Esta fase pode ser executada através de reuniões e diálogo entre o time responsável e o gerente do projeto. Para facilitar a sua execução e se ter uma boa compreensão do processo, os detalhes internos da atividade de planejar interação é apresentado na Figura 4. Como pode ser observado, a tarefa de planejar a interação possui três sub-atividades:

- **Definir papeis**- Onde será definido quem pode realizar o processo;
- **Planejar *sprint*** - Será planejado como a atividade será realizada;
- **Identificar riscos e pendências**- será definidos os riscos e pendências que podem ocorrer antes, durante e depois da realização do processo.

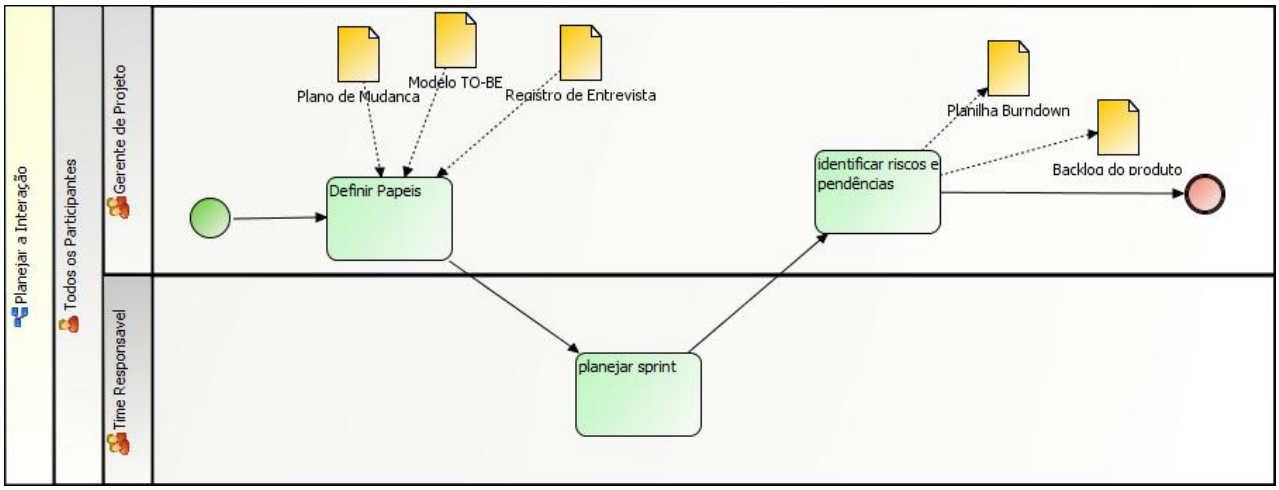


Figura 4 – Diagrama sobre as atividades de Planejar interação do processo

Dois artefatos são produzidos durante o planejamento da interação: (1) o *backlog* do produto; e (2) a planilha *Burndown chart*; esta última ajudará a acompanhar o desenvolvimento e entregas do produto. O *backlog* do produto pode ter a seguinte composição:
- ID - identificação do requisito;
- Nome - nome do requisito;
- Importância - o nível de importância do requisito;
- Estimativa Inicial - custos iniciais com o produto;
- Como demonstrar - como deve funcionar e como deve ser a interação com o requisito;
- Notas - observações.

Após o planejamento da interação é necessário definir variáveis do ambiente a serem monitoradas e seus pontos de variação. Além de, definir onde estará presente o dinamismo do ambiente (ver Figura 3). Nesta fase são definidas características críticas do ambiente, que variam seu estado durante a execução do sistema. Vários fatores internos e externos podem fazer com que seu estado mude. A definição dessas variáveis trará ao método formas de monitorar e tratar as variações do ambiente de modo eficiente e eficaz durante a execução do software. Exemplos de variáveis são: (1) Disponibilidade de recursos redundantes; (2) Número de acessos simultâneos; (3) Tempo de resposta; (4) Número de exceções lançadas por minuto; etc.

Após a definição das variáveis de ambiente a serem monitoradas pode-se iniciar a definição dos requisitos em si, que tem o intuito de identificar as características requeridas da

aplicação, definir a prioridade entre elas, compreender as características de forma detalhada e especificar o comportamento da solução, permitindo o desenvolvimento do software de forma satisfatória. Participarão dessa fase os analistas de negócio, gerente de processo e o gerente de projeto. Os detalhes internos da atividade de "definir requisitos"são apresentados na Figura 5.

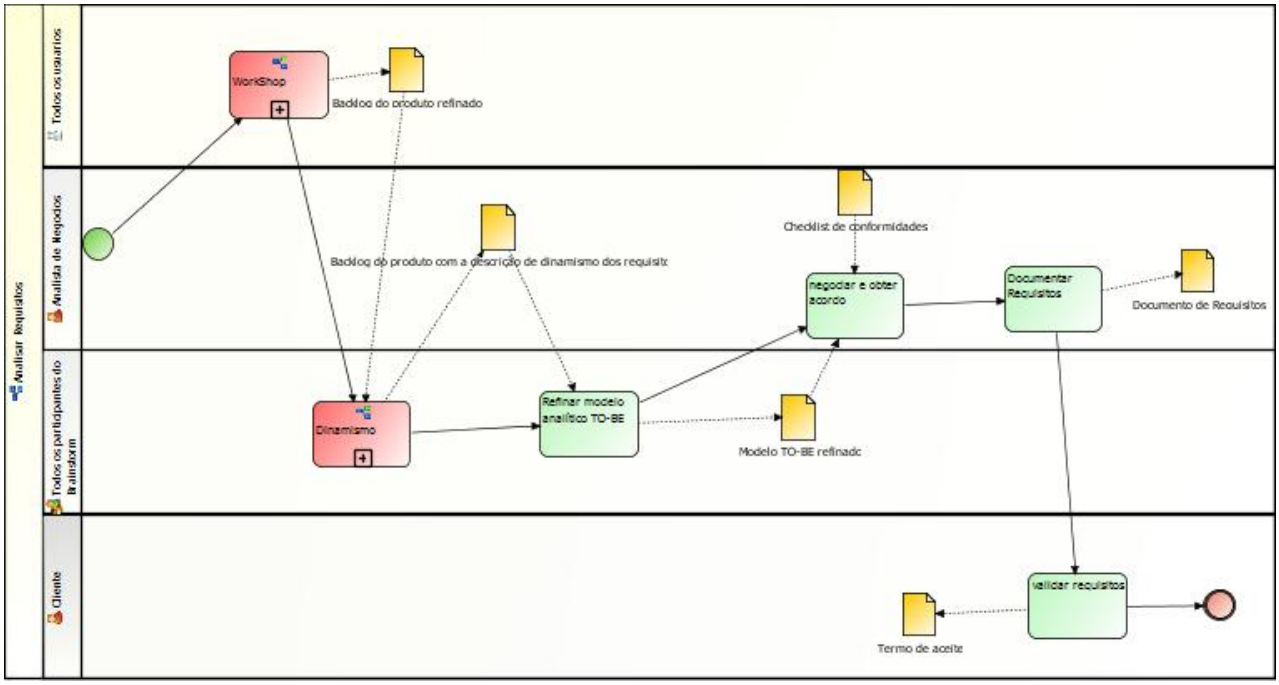


Figura 5 – Diagrama sobre as atividades da fase de Definição de requisitos

O processo de Definição de requisitos inicia-se com um *workshop*, que consiste na elaboração de uma reunião estruturada com todos os participantes da atividade. O objetivo é listar um conjunto bem definido de requisitos. O *workshop* deve seguir um roteiro bem definido e possuir um facilitador, que pode estar representado no papel do gerente de projeto.

Como pode ser visto na Figura 6, o roteiro do *workshop* inicia-se com as tarefas de introdução feita pelo facilitador, este apresenta a necessidade do projeto a partir das informações do planejamento estratégico e das outras informações. Em seguida, o facilitador realiza a contextualização, onde serão revisadas as variáveis de ambiente e os demais artefatos produzidos anteriormente.

Após a revisão dos artefatos anteriores, deve-se realizar uma seção de *brainstorm* com o objetivo de obter respostas para um questionário com perguntas-chave, previamente construído através de duvidas que surgiram nas atividades anteriores. Todos os participantes podem colaborar na resolução do questionário, incluindo desenvolvedores e analista de negócios. Nesse tempo, cada um pode especificar suas dúvidas, que serão adicionadas como novas perguntas a serem respondidas no documento com as respostas obtidas. Ainda durante o *workshop*. Após sanar as dúvidas e responder o questionário, o *brainstorm* deve tratar da geração de ideias, por parte de toda a equipe, sobre as possíveis funcionalidades do sistema.

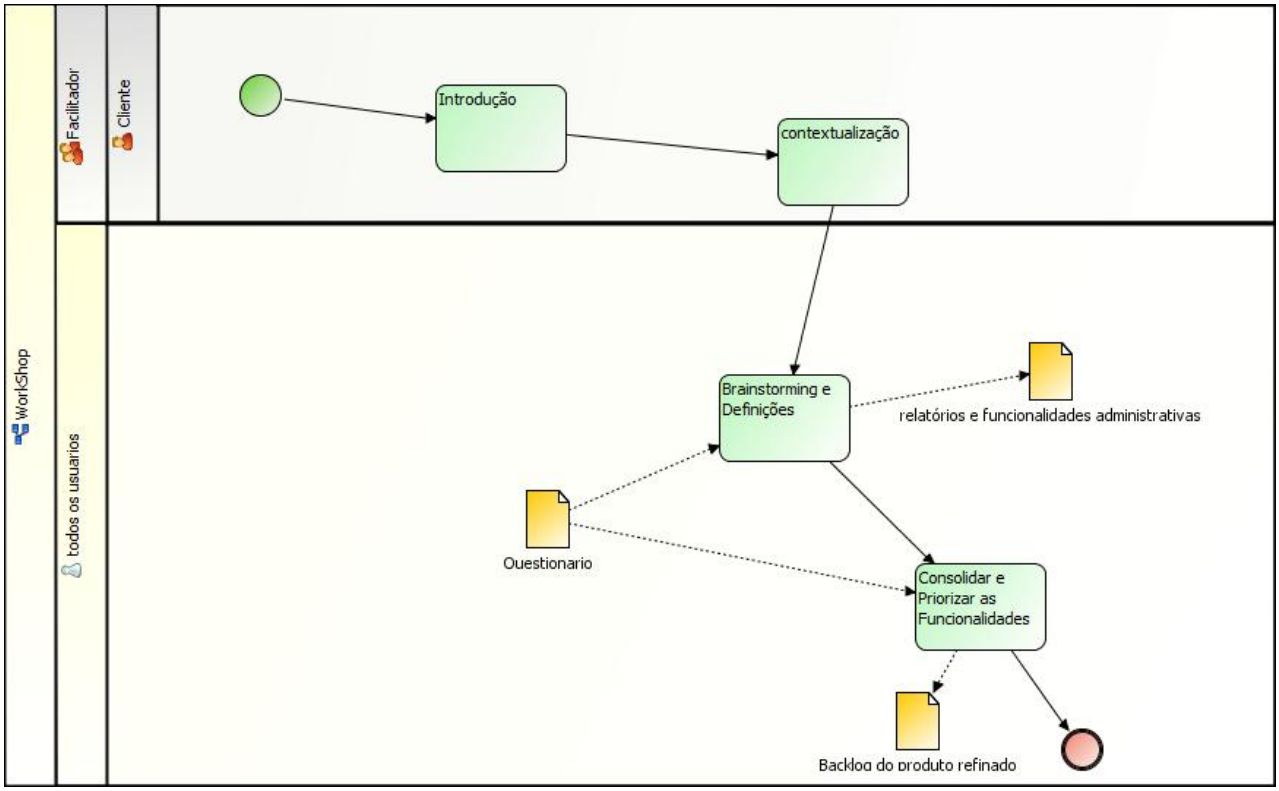


Figura 6 – Diagrama sobre as atividades da fase do Workshop

A lista de perguntas e respostas, assim como a lista de funcionalidades identificadas, deve ser incorporada a um relatório que deve ser gerado. Tais funcionalidades darão origem ao *product backlog*.

Após o *brainstorm* deve-se consolidar e priorizar os requisitos identificados, onde será definida a prioridade dos itens do *backlog* do produto. Os desenvolvedores e gerentes de projeto podem aplicar técnicas para medir o tamanho do projeto para os cenários identificados no *backlog*, a partir do modelo TO-BE (BRASIL, 2013). As questões abaixo devem ser consideradas nessa etapa:

1. Quais sistemas ou softwares se encaixam melhor às necessidades do processo?
2. O modelo pode ser implementado rapidamente?
3. Qual será o impacto para a organização?
4. Pode ser empregada uma abordagem por estágios?
5. Qual será o custo da nova implementação (incluindo treinamento, tecnologia e etc.)?
6. Existem fornecedores que podem auxiliar na implementação?

A estimativa do tamanho do projeto pode ser realizada através da técnica *Planning Poker*, adotada no processo Scrum (Seção 2.4). Nessa técnica, a estimativa de esforço para im-

plementação do requisito é definida de forma individual por cada participante do *brainstorm*, que pode atribuir pontos, de acordo com a sequência de Fibonaccci (SCRUMSTUDY, 2016), de acordo com a complexidade que se vislumbra naquele requisito. A pontuação de cada funcionalidade do *product backlog* é então estipulada pela média da pontuação atribuída pelo time de desenvolvimento.

Outra técnica que pode ser utilizada é a *ideal day*, que mede a quantidade de trabalho que um profissional da área consegue concluir em um dia de trabalho (PEREIRA; TORREÃO; MARÇAL, 2007). Segundo (PEREIRA; TORREÃO; MARÇAL, 2007), a velocidade é calculada a partir do número de horas que a equipe gasta para implementar um trabalho equivalente ao *ideal day*. Caso passe de um dia de trabalho, é sugerido decompor esse item em itens menores que possam ser implementados em um dia, de forma a facilitar o acompanhamento. A partir do resultado das técnicas de estimativas do projeto pode-se refinar o *product backlog*.

Após a condução do *workshop*, é chegada a hora de se definir os requisitos de qualidade; em especial o impacto de cada um nas adaptações dinâmicas do software. Esta etapa é denominada Dinamismo (ver Figura 5). A atividade dinamismo deve ser executada logo após a escolha e priorização dos requisitos na atividade de *workshop* e participa dela todos os envolvidos com a fase do workshop. Durante essa atividades podem ser identificados conflitos entres os requisitos de qualidade. Tais conflitos devem ser identificados e devidamente tratados.

A Figura 7 apresenta os detalhes internos da atividade de dinamismo. Para cada uma das funcionalidades já identificadas, deve-se identificar os requisitos de qualidade desejados, assim como potenciais adaptações de comportamento.

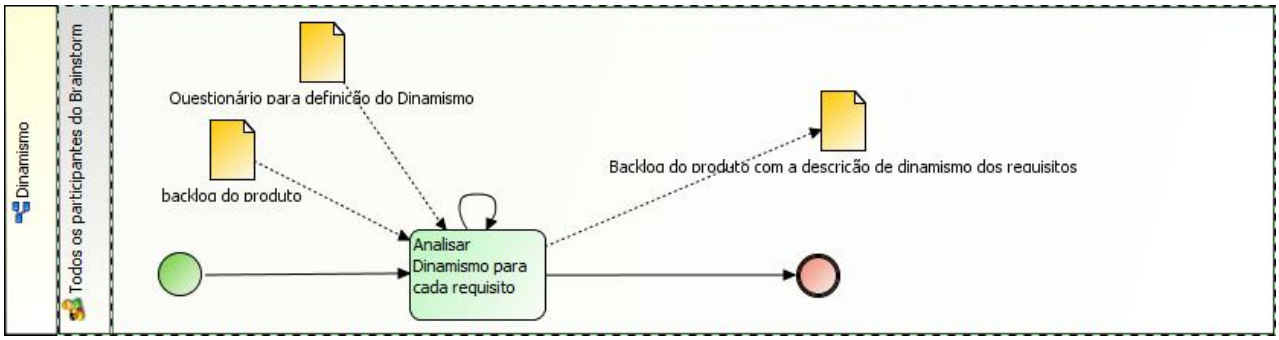


Figura 7 – Diagrama sobre as atividades da fase de Dinamismo

Para guiar a identificação dos requisitos de qualidade, o método proposto apresenta um questionário com diretrizes concretas para os requisitos de desempenho, escalabilidade, disponibilidade, segurança e confiabilidade. É recomendado que tal questionário seja respondido por todos os participantes, através de um *brainstorm* modelado pelo gerente de projetos. Vale salientar que algumas perguntas do questionário fazem referência a valores de limiares que não estão definidos. Tais limiares também devem ser definidos durante a

resolução do questionário. O questionário fornecido pelo processo é apresentado a seguir:

1. Para cada funcionalidade
    a) Ela possui limite de tempo?
        i. 1. Se sim: Definir o limite (em milissegundos)
            A. Qual o período crítico do dia para monitorar?
                - O que fazer se esse limite for excedido em até X% ?
                - O que fazer se esse limite for excedido em mais de X% ?
    b) Essa funcionalidade terá grande fluxo de acesso (acessos simultâneos)?
        i. Se sim: Qual o pico máximo a ser tolerado (número máximo de acessos simultâneos)?
            A. Qual o período crítico do dia para monitorar
                - O que fazer se esse limite for prejudicado em até X%?
                - O que fazer se esse limite for prejudicado em mais de X%?
    c) Essa funcionalidade deve estar disponível o máximo de tempo possível?
        i. Se sim: A indisponibilidade da funcionalidade pode acarretar prejuízos financeiros?
        ii. sim: A indisponibilidade da funcionalidade pode acarretar prejuízos estratégicos para a instituição?
        iii. sim: A indisponibilidade da funcionalidade pode acarretar perda de vidas humanas?
        iv. sim: Qual o tempo máximo que a funcionalidade pode ficar “off-line” para eventuais manutenções (minutos por dia/semana/mês/ano)?
            A. Qual o período crítico do dia para monitorar
                - O que fazer se o tempo for excedido em até X%?
                - O que fazer se o tempo for excedido em mais de X%?
    d) Essa funcionalidade lida com dados sigilosos, que devem ser mantidos cifrados para aumentar a segurança?
        i. Se sim: Qual o tempo mínimo exigido (seguro) para que alguém com acesso eventual aos seus dados cifrados consigam decifrá-los? (em horas, dias, meses ou anos)
            A. Qual o período crítico do dia para monitorar?
                - O que fazer se o tempo de decifrar for reduzido em até X%?
                - O que fazer se o tempo de decifrar for reduzido em mais de X%?
    e) Essa funcionalidade será acessada remotamente, por exemplo, via Web?

i. Se sim: Há preocupação com eventuais invasões ou ataques maliciosos nos servidores de aplicação?
   A. Se sim: Qual o tempo esperado para identificar e solucionar o problema, a partir do início do ataque? (em segundos, minutos, horas, dias)
   B. Qual o período crítico do dia para monitorar?
      - O que fazer se o tempo for excedido em até X%?
      - O que fazer se o tempo for excedido em mais de X%?

ii. Qual o máximo de tempo para se implementar mudanças nas funcionalidades existentes (minutos/dia/semana/mês/ano)?

iii. Qual o tempo máximo para se implementar novas funcionalidades (minutos/dia/semana/mês/ano)?

f) Na funcionalidade pode existir algum tipo de falha?

i. A funcionalidade deve funcionar, ainda que parcialmente, quando houver algum tipo de falha?
   A. Se sim: A Falha na funcionalidade pode acarretar prejuízos financeiros?
   B. Se sim: A falha na funcionalidade pode acarretar prejuízos estratégicos para a instituição?
   C. Se sim: A falha na funcionalidade pode acarretar perda de vidas humanas?
   D. Quais os principais cenários de falhas?
   E. Para cada cenário de falha:
      - Qual o período crítico do dia para monitorar?
      - O que deve ser feito se a falha ocorrer?
      - Qual o tempo máximo para solucioná-la? (em minutos, horas, dias, meses ou anos)
   F. A funcionalidade deve ter previsão de falha?
      - Se sim:
      - Qual o período crítico do dia para monitorar
      - Qual o tempo mínimo de antecedência para previsão da falhas?
      - O que deve ser feito se uma falha for prevista?
   G. Deseja monitorar o tempo médio entre duas falhas?
      - Qual o período crítico do dia para monitorar
      - Qual o tempo médio mínimo esperado entre duas falhas?
      - O que fazer se o tempo for menor que x?

ii. Quais as exceções esperadas para a funcionalidade atual (exceções do negócio)?
   A. Qual o período crítico do dia para monitorar?

B. O que fazer quando uma exceção de negócio for lançada?

C. que fazer quando uma outra exceção for lançada?

iii. Existe alguma fonte de informação/dados para alimentar a funcionalidade?

A. Se sim: especifique as fontes.

B. As fontes de informação possuem uma formatação de acordo com algum padrão predefinido?

- Se sim:
- Qual o período crítico do dia para monitorar?
- O que fazer quando esse padrão não for obedecido?

C. A execução de testes na aplicação é importante?

- Se sim:
- Qual o período crítico do dia para monitorar?
- Qual a porcentagem mínima do código valor que deve ser coberto pelos casos de teste?
- O que fazer se a cobertura dos testes estiver inferior a x%?

Com as informações coletadas durante a resolução do questionário é possível refinar o documento de requisitos através do modelo analítico *TO-BE* (ver Figura 5), de forma que os seguintes pontos possam ser preenchidos:

- Definição das atividades (automáticas e manuais) que são partes do processo primário, suporte e gerenciamento/monitoração;
- Definição de *handoffs*[2] de processos entre grupos funcionais;
- Definição de métricas e indicadores desejados para cada variável de ambiente;
- Definição de regras de negócio;
- Definição de caminhos de exceção com o tratamento de erro de negócio, quando não é possível adaptar o sistema para satisfazer os requisitos (caminhos tristes).
- Definição de informações de fluxo de dados;

Após a atualização do modelo TO-BE os participantes do processo devem realizar um *checklist* de conformidade para todos os requisitos definidos, que consiste em um questionário para identificar informações sobre a atividade através de consultas aos artefatos produzidos anteriormente e também das respostas dos *stakeholders* envolvidos no questionário. O *checklist* deve responder as seguintes perguntas:

[2] *Handoffs* são qualquer ponto em um processo no qual o trabalho ou a informação passa de uma função para outra.

1. Os requisitos podem ser entendidos claramente ?
2. Os requisitos não possuem informação repetida desnecessariamente ?
3. Os requisitos atendem completamente as necessidades do cliente ?
4. Existe alguma informação faltante que deveria estar descrita no documento?
5. Os requisitos podem ser interpretados de forma diferente por diferentes usuários?
6. Os requisitos não geram contradição entre si ?
7. Os requisitos estão organizados de forma adequada ?
8. Os requisitos estão em conformidade com os padrões estabelecidos ?
9. Os requisitos podem ser rastreados, possuem ligações com outros requisitos que possuem relação e a razão de sua existência está documentada ?

Após o refinamento dos requisitos, com o modelo TO-BE e o *checklist,* é necessário apresentar os requisitos detalhados e as características dinâmicas dos requisitos de qualidade do sistema aos clientes para que sejam identificados eventuais pontos de discordância que necessitem de negociação. As discordâncias podem ocorrer por quatro razões principais: (1) Discordância do custo necessário para implementar a adaptação dinâmica, tais como a necessidade de recursos redundantes de *hardware* e software; (2) Discordância dos limiares definidos; (3) Discordância das prioridades dos requisitos; e (4) Discordância dos comportamentos dinâmicos previstos.

Após a negociação e consequente atualização dos requisitos, dá-se continuidade com a documentação final dos requisitos especificados (ver Figura 5). O método proposto propõe uma estrutura para o documento de requisitos, que deve refletir o que foi decidido e acordado durante o processo para definição de requisitos, documentando inclusive as dúvidas esclarecidas e as decisões fruto do processo. O documento dessa fase pode seguir a estrutura sugerida abaixo:

1. INTRODUÇÃO
2. DESCRIÇÃO DO MINI-MUNDO (DESCRIÇÃO TEXTUAL DO SISTEMA)
3. DESCRIÇÃO FUNCIONAL
4. DESCRIÇÃO COMPORTAMENTAL ESTÁTICA
5. DESCRIÇÃO COMPORTAMENTAL DINÂMICA
6. LISTA DE PRIORIDADES

7. CRITÉRIOS DE VALIDAÇÃO

8. DÚVIDAS ESCLARECIDAS

Após a fase de documentação de análise de requisitos passa-se a atividade de validar requisitos (ver Figura 5), onde os requisitos que foram especificados devem ser avaliados com a participação do cliente. Com a validação desses requisitos será elaborado o termo de aceite, que evidencia o consentimento do cliente em relação aos requisitos especificados.

Para faciliar eventuais consultas durante a fase de elicitação dos requisitos, a Tabela 1 apresenta um resumo das atividades definidas no método proposto.

Atividade recomendada	**Atividade base**
Modelo de referência	(RAMOS, 2014)
Modelo AS-IS	(BRASIL, 2013)
Reuniões JAD	(SCRUMSTUDY, 2016)
Modelo TO-BE	(BRASIL, 2013)
Planos de mudança de Kotter	(BRASIL, 2013)
Sprint	(SCRUMSTUDY, 2016)
Planilha *Burndown Chart*	(SCRUMSTUDY, 2016)
Workshop	(SOMMERVILLE, 2007)
Brainstorm	(SOMMERVILLE, 2007)
Product Backlog	(SCRUMSTUDY, 2016)
Planning Poker	(SCRUMSTUDY, 2016)
Ideal Day	(SCRUMSTUDY, 2016)
Refinamento do modelo TO-BE	(BRASIL, 2013)
Checklist de Conformidade	(BRASIL, 2013)
Modelo de documentação de engenharia de requisitos	(SOMMERVILLE, 2007)
Termo de Aceite	(SOMMERVILLE, 2007)

Tabela 1 – Relação entre os processos sugeridos e o processos em que foram baseados.

5 AVALIAÇÃO DA SOLUÇÃO

5.1 PLANEJAMENTO DA AVALIAÇÃO

5.1.1 Método *Goal-Question-Metric* (GQM)

O método escolhido para medição do processo foi o *Goal-Question- Metric* (GQM). O GQM é uma abordagem *top-down* para estabelecer um sistema de medição direcionado a metas e bastante utilizado para avaliações relacionadas ao desenvolvimento de software (SOLINGEN et al., 2002; WANGENHEIM; RUHE,).

O método GQM define que inicialmente deve-se definir os objetivos da medição são produtos (*goals*). As questões (*questions*) estão no nível operacional e são um conjunto de questões sobre as características e o modo como os objetivos estão sendo especificados; as questões devem abranger e caracterizar os objetivos da medição. As métricas (*metrics*) estão no nível quantitativo e são o conjunto de dados que está associado às respostas das questões e a partir delas deve ser possível realizar a análise objetiva e subjetiva do conjunto.

5.1.2 Especificação do GQM no Contexto do Método Proposto

Os objetivos, questões e métricas do método GQM foram organizados da seguinte forma, para validar o método proposto:

G1 Aferir a viabilidade e facilidade de execução do método proposto.

Q1.1. O método sugerido pode ser seguido?

M1 Proporção das respostas:

- Concordo Fortemente; Concordo; Discordo; Discordo Fortemente.

Q1.2. O método apresentado é intuitivo?

M1 Proporção das respostas:

- Concordo Fortemente; Concordo; Discordo; Discordo Fortemente.

Q1.3. O método apresentado facilita a descoberta dos requisitos de qualidade?

M1 Proporção das respostas:

- Concordo Fortemente; Concordo; Discordo; Discordo Fortemente.

Q1.4. O método apresentado facilita a compreensão e detalhamento dos requisitos não-funcionais?

M1 Proporção das respostas:

- Concordo Fortemente; Concordo; Discordo; Discordo Fortemente.

Q1.5. O método apresentado adapta-se bem às regras de negócio?

M1 Proporção das respostas:

- Concordo Fortemente; Concordo; Discordo; Discordo Fortemente.

Q1.6. Na sua opinião este método contribui para facilitar a especificação de requisitos de qualidade e de suas características dinâmicas?

M1 Proporção das respostas:

- Concordo Fortemente; Concordo; Discordo; Discordo Fortemente.

Q1.7. Na sua opinião existe algo que deve ser modificado? Sugestões?

Q1.8. O método apresentado possui alguma dificuldade para ser seguido? especifique-a.

G2. Aferir se o método ajuda na especificação de requisitos não funcionais, quando comparado com outro método.

Q2.1. Qual o comparativo de qualidade dos requisitos nos dois métodos?

M1 Quantidade de requisitos elicitados e validados pelo cliente.

M2 Quantidade de requisitos elicitados, mas não validados pelo cliente.

G3. Aferir o esforço para especificar requisitos com o método proposto, quando comparado com outro método.

Q3.1. Qual o comparativo de tempo de trabalho dos dois engenheiros de requisitos?

M3.1.1 Tempo médio para execução do processo.

M3.1.2. Número de intervenções do cliente.

Q3.2. O método apresentado contribui no comprimento dos prazos?

M1 Proporção das respostas:

- Concordo Fortemente; Concordo; Discordo; Discordo Fortemente.

Q3.3. O método apresentado é efetivo?

M1 Proporção das respostas:

- Concordo Fortemente; Concordo; Discordo; Discordo Fortemente.

Q3.4. Ao comparar o método apresentado com outro para especificação de requisito, este é mais eficiente?

M1 Proporção das respostas:

- Concordo Fortemente; Concordo; Discordo; Discordo Fortemente.

Q3.5. Ao comparar o método apresentado com outro para especificação de requisito, este é mais rápido(horas)?

G4. Aferição da qualidade da especificação dos requisitos não funcionais.

Q4.1. Como foi a especificação do requisito de confiabilidade?

M1 Proporção das respostas:

- Elicitou e especificou detalhadamente; Elicitou e especificou de forma abstrata; Apenas elicitou; Não identificou o requisito.

Q4.2. Como foi a especificação do requisito de disponibilidade?

M1 Proporção das respostas:

- Elicitou e especificou detalhadamente; Elicitou e especificou de forma abstrata; Apenas elicitou; Não identificou o requisito.

5.1.3 Metodologia de Execução

O experimento foi realizado com uma turma de alunos do curso de ciência da computação, que cursam a disciplina de Engenharia de Requisitos, que na grade curricular do curso está no terceiro período do fluxo padrão.

Participaram da execução do experimento 39 alunos. Estes alunos foram divididos em 3 grupos, cada um deles com 13 integrantes. Onde cada um desses grupos recebeu um guia com um processo diferente de especificação de requisitos não funcionais: (1) O processo aqui proposto; (2) O processo construido por (RAMOS, 2014); e (3) Diretrizes gerais de *brainstorming*.

O experimento teve duração de aproximadamente quatro horas e todas as equipes receberam a mesma descrição sobre o sistema. Durante o tempo de execução as equipes participantes puderam consultar o cliente para esclarecer as dúvidas que surgiram durante a especificação dos requisitos não funcionais.

5.2 EXECUÇÃO DO EXPERIMENTO

5.2.1 Sistema Utilizado

O sistema utilizado para a execução do experimento e verificação das métricas do processo trata-se de um sistema de controle de piloto automático de aeronaves. Este sistema cuida de todo o funcionamento do avião em vôo de cruzeiro, além de, também ser responsável pelas funcionalidades de controlar pouso e decolagem.

O ambiente de execução desse tipo de sistema pode sofrer várias interferências, algumas delas podem ser o clima, temperatura, velocidade e direção do vento, neblina, detectar aves ou objetos que estejam voando em sua rota sem conhecimento da torre de comando e alguns outros fatores.

A partir das variações do ambiente o sistema precisa reagir de maneira rápida e eficaz a estes eventos ambientais. Dessa forma, fica claro que o sistema possui requisitos críticos de confiabilidade, disponibilidade e desempenho, de forma a evitar possíveis acidentes.

5.2.2 Análise dos Resultados

O experimento foi realizado com 39 alunos que utilizaram um dos 3 métodos sugeridos para a elicitação de requisitos não-funcionais; porém, apenas 20 participantes responderam o questionário de avaliação sobre a técnica utilizada. A Figura 8 apresenta a disposição desses 20 voluntários de acordo com o processo utilizado.

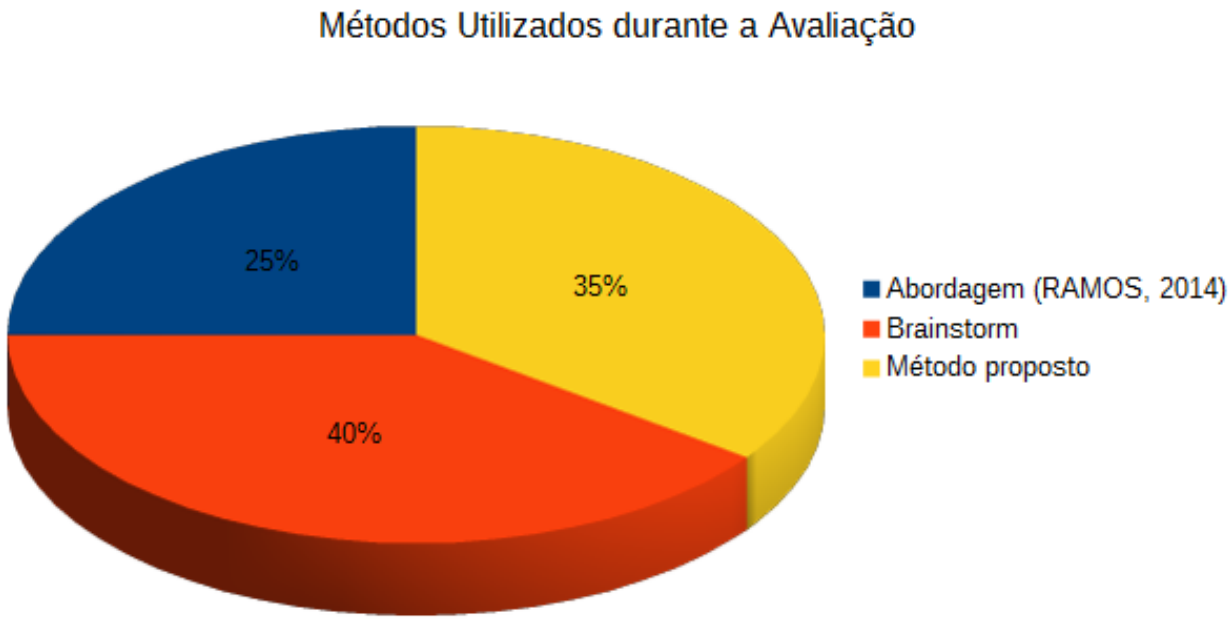


Figura 8 – Gráfico relativo a disposição dos participantes do experimento.

A primeira métrica abordada foi aferir a viabilidade e facilidade de execução do método proposto. Para isto, foram feitos alguns questionamentos que serão mostrados a seguir. Os resultados são relativos aos 20 alunos que responderam ao questionário.

A Figura 9 apresenta a proporção de respostas para o questionamento “O método pode ser seguido?” A partir dos gráficos apresentados na figura, pode-se afirmar que todos os participantes concordaram que os métodos utilizados podem ser seguidos de forma fácil.

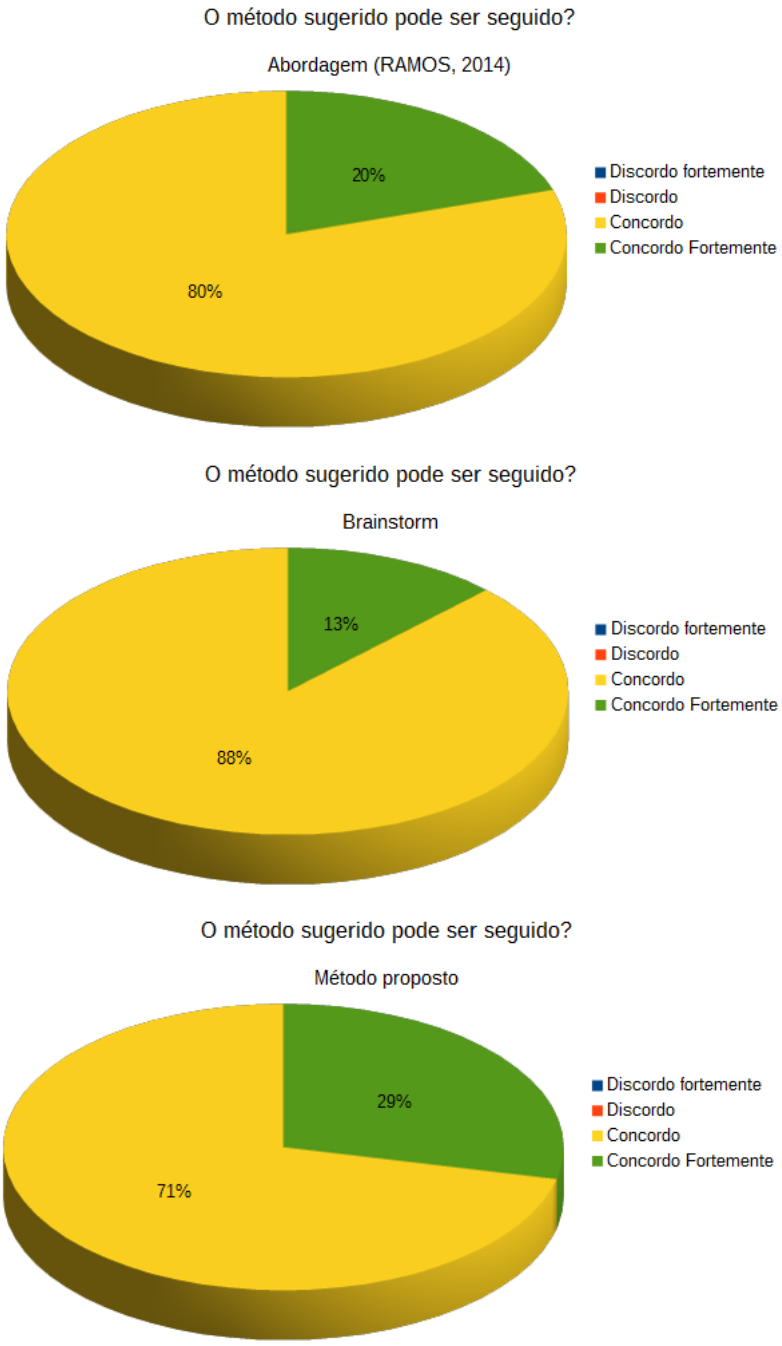


Figura 9 – Gráfico relativo à questão: "O método pode ser seguido?"

A Figura 10 mostra que a maior parte dos alunos concordaram que os processos utilizados são intuitivos. Porém alguns voluntários que não acharam o método intuitivo relataram que não compreenderam ou não utilizaram algumas das técnicas recomendadas pelo processo utilizado.

Em relação à facilidade proporcionada na descoberta dos requisitos de qualidade (Figura 11), os processos utilizados foram bem avaliados pelos participantes. Porém, alguns voluntários discordaram desse quesito em relação ao método proposto nesta dissertação. A

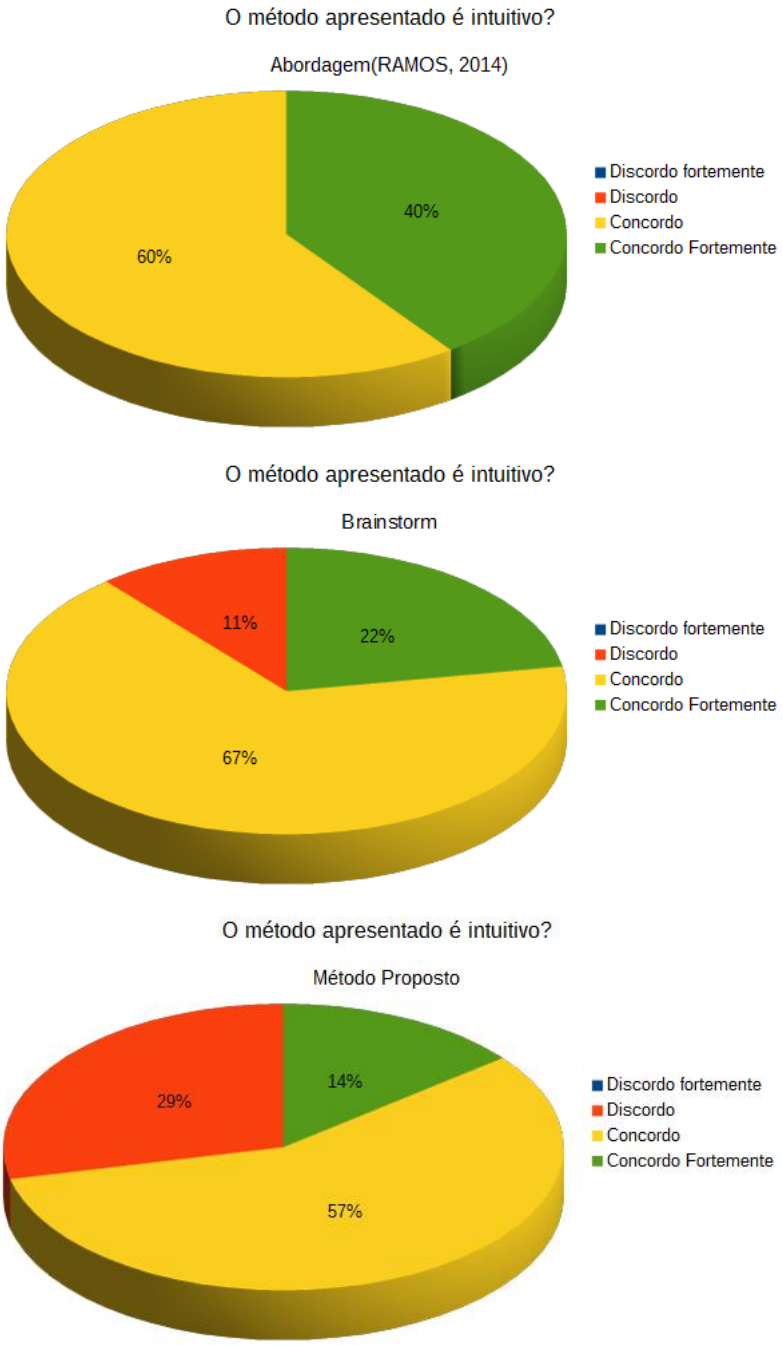
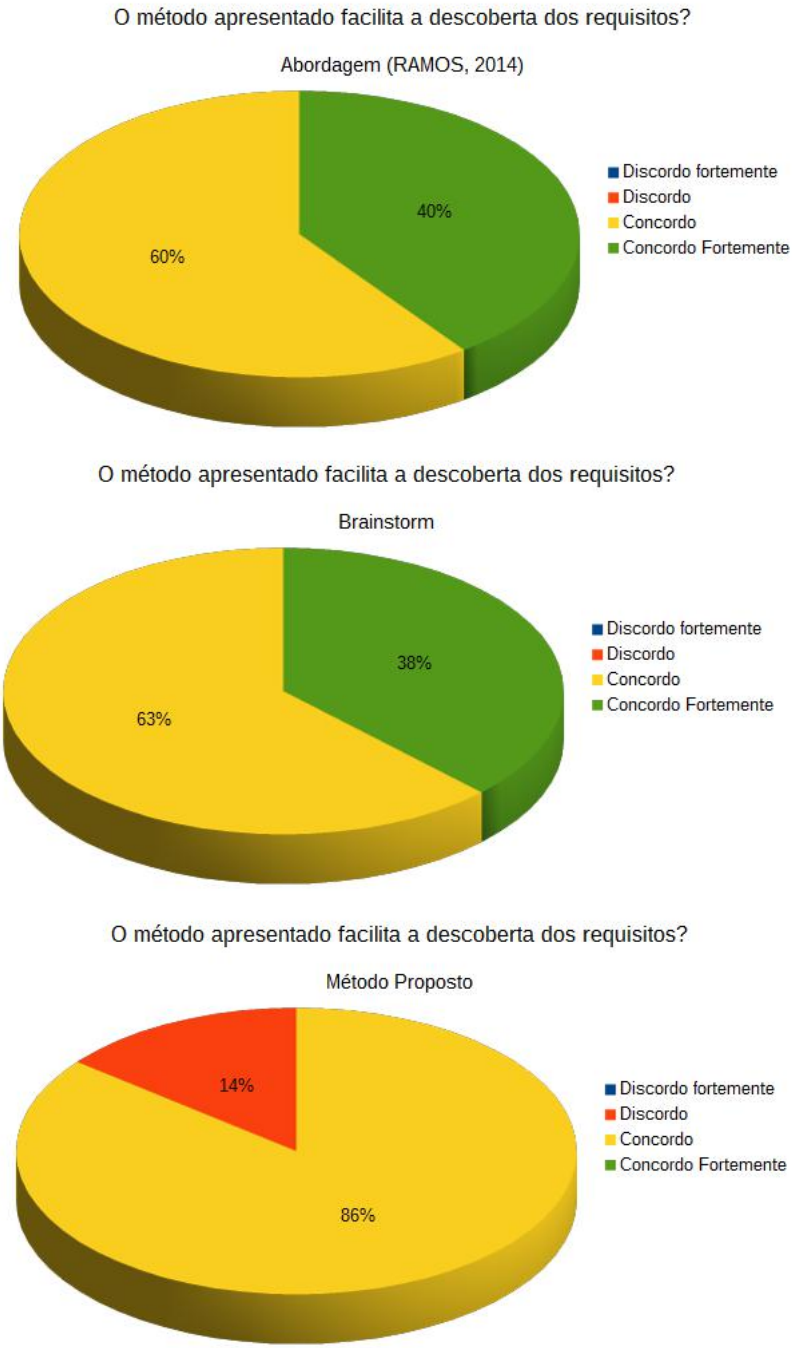


Figura 10 – Gráfico relativo a questão: "O método é intuitivo?"

causa apontada para esse descontentamento de alguns foi o fato do método proposto focar em alguns requisitos de qualidade específicos, não considerando outros requisitos da ISO 9126 (ISO/IEC, 2001). Além de, ter sido atribuído esse resultado ao processo ser um pouco mais longo que os outros processos.

No quesito de facilidade da compreensão e detalhamento (especificação) dos requisitos de qualidade (Figura 12), todos os voluntários que utilizaram *brainstorm* concordaram que é um método que facilita a descrição dos requisitos não-funcionais, pois em sua opinião



[]

Figura 11 – Gráfico relativo a questão: "O método apresentado facilita a descoberta (elicitação) de requisitos de qualidade?"

descrever rápido e em maior numero é o importante. Entretanto os participantes que utilizaram os dois métodos mais elaborados obtiveram resultados ligeiramente diferentes. Enquanto no processo de Pires (RAMOS, 2014) a maioria dos voluntários concordaram com a questão, o restante se dividiu entre concordar fortemente e discordar. Já no método proposto nesta dissertação, a maioria dos voluntários concordaram fortemente com a afirmação, enquanto o restante se dividiu entre a concordância e a discordância. Alguns voluntários discordaram por terem elicitado menos requisitos que o método os outros dois métodos.

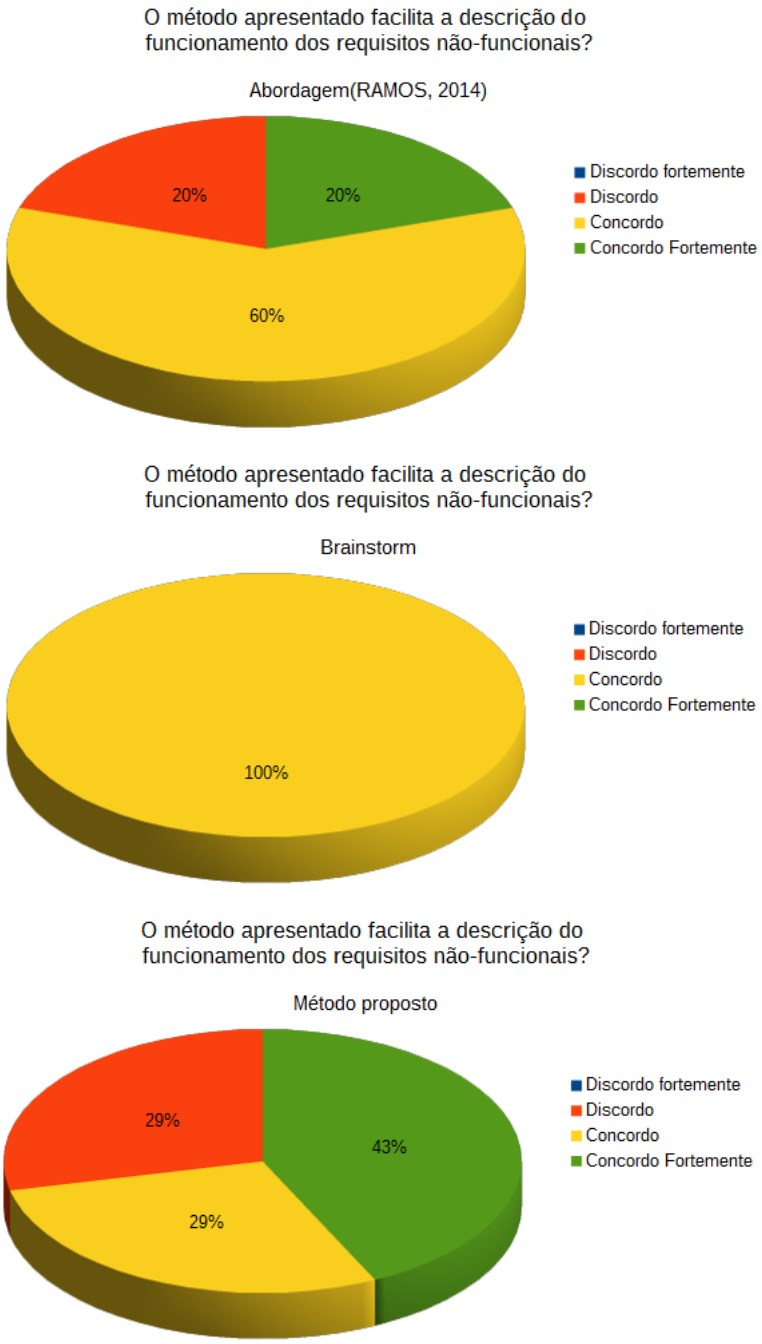


Figura 12 – Gráfico relativo a questão: “O método apresentado facilita a compreensão e detalhamento dos requisitos de qualidade?”

Em relação ao questionamento: "O método adapta-se bem as regras de negócios?", os gráficos apresentados na Figura 13 mostram que segundo os participantes tanto o método de *brainstorm*, quanto o método proposto nesta dissertação foram bem aceitos por todos os voluntários. Porém uma pequena parcela de voluntários acharam que o processo de Pires (RAMOS, 2014) é de difícil adaptação para aplicação às regras de negócio do sistema alvo.

A outra questão abordada de fundamental importância para a avaliação do método

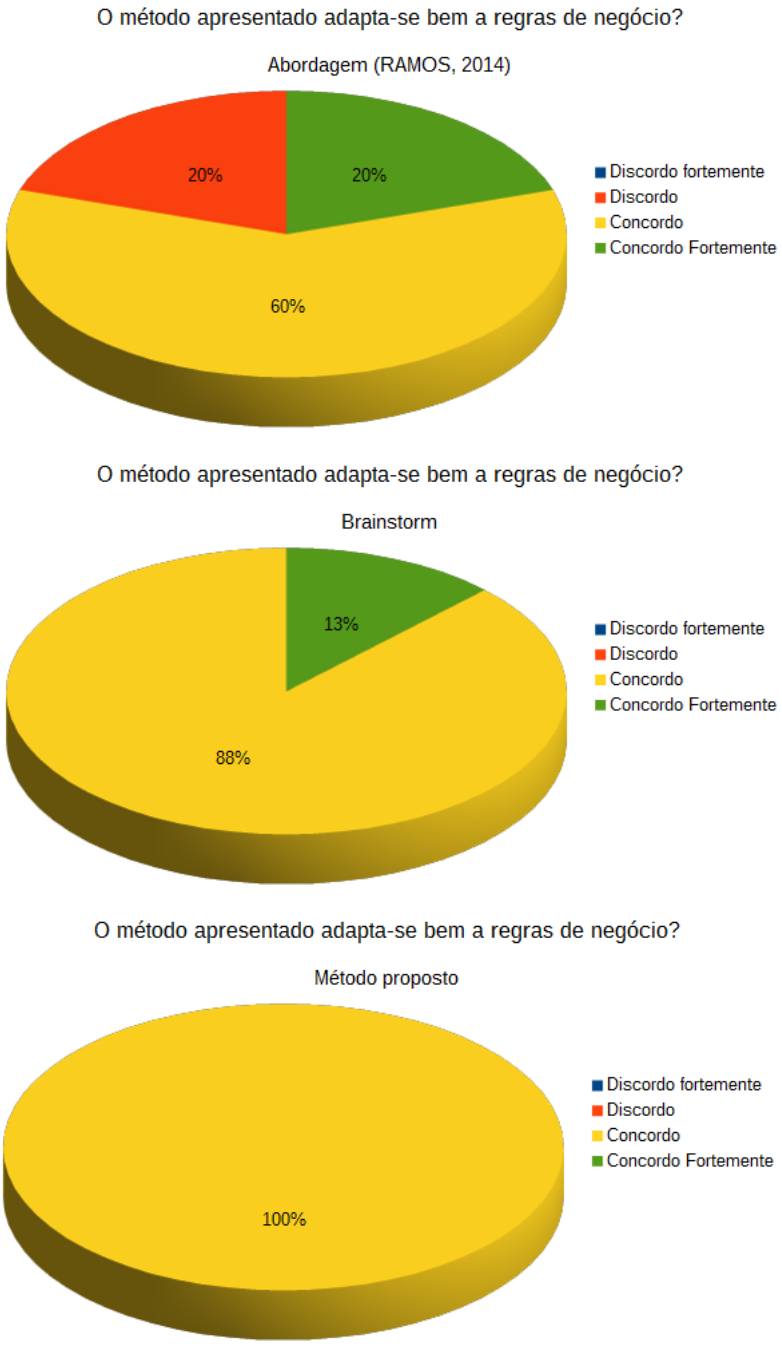


Figura 13 – Gráfico relativo a questão: "O método apresentado adapta-se bem as regras de negócios?"

proposto foi a opinião dos voluntários sobre como o método facilita o detalhamento dos requisitos de qualidade e das respectivas características dinâmicas no funcionamento do software. Dentre as questões qualitativas, esta é considerada a mais importante, pois avalia a principal característica desejada para o processo.

A Figura 14 apresenta uma visão geral dos resultados obtidos. Como pode ser visto, a grande maioria dos voluntários (57%) concorda fortemente que o método proposto facilita o detalhamento dos requisitos e a especificação do comportamento dinâmico, enquanto

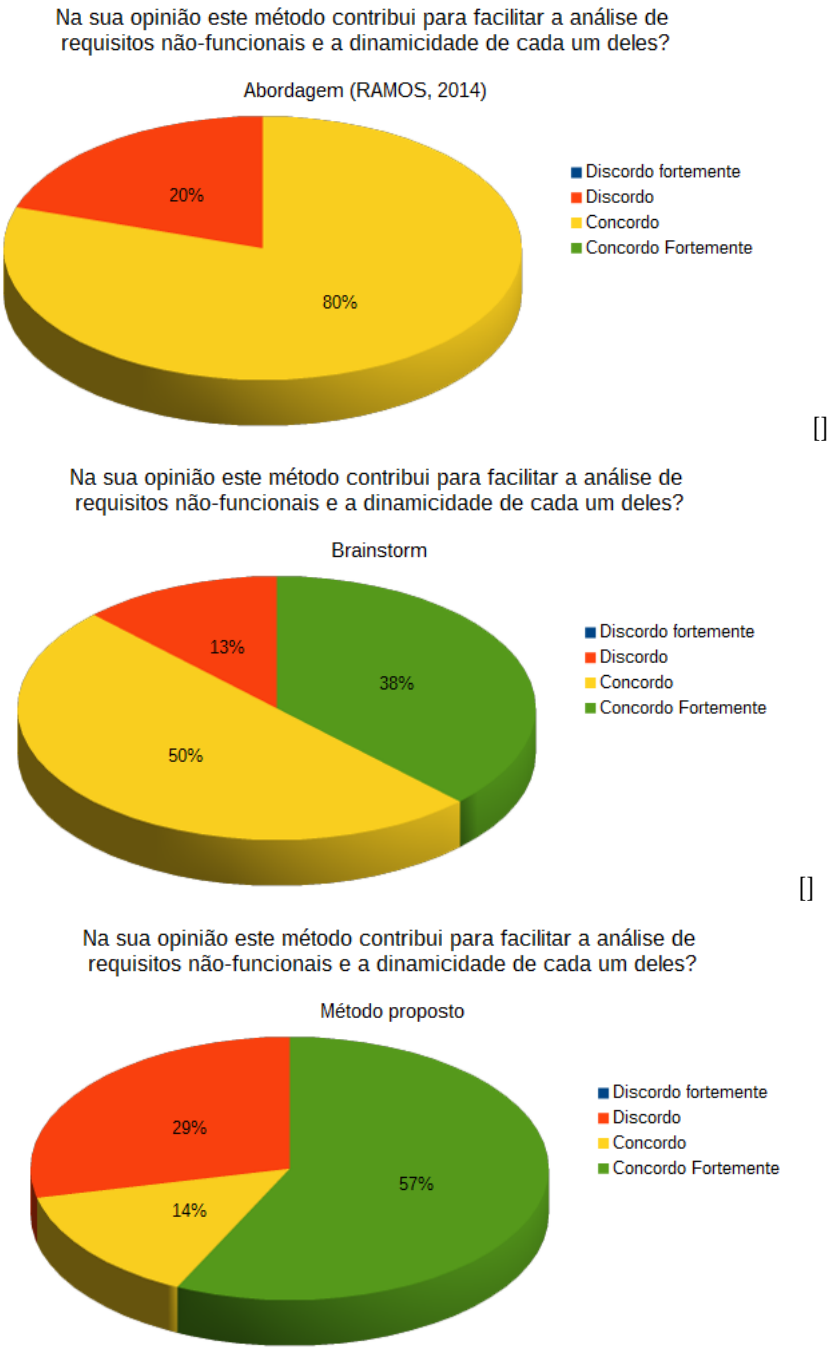


Figura 14 – Gráfico relativo a questão: "O método contribui para facilitar o detalhamento (especificação) de requisitos de qualidade e de suas características dinâmicas?"

outros (14%) apenas concordam. só (29%) dos voluntários discordam. Vale salientar que os voluntários que discordaram também relataram que não conseguiram acompanhar bem as atividades do processo, o que recai no problema da curva de aprendizagem, já que o processo foi apresentado e executado no momento do experimento, sem que houvesse preparação prévia por parte dos voluntários.

Durante a execução do experimento foram observados alguns outros fatores, dentre

estes está o tempo em que foi realizado o processo. Os voluntários tiveram um tempo de 4 horas para a execução, porém a equipe que estava utilizando *brainstorm* demorou em torno de 1h 30 min para executar e documentar os requisitos. Enquanto isso, o grupo que utilizou o processo de Pires (RAMOS, 2014) realizou o trabalho em 2h . Por fim, o grupo que utilizou o método proposto nesta dissertação finalizou sua execução em aproximadamente 3h e 20 min. Porém, acredita-se que esse tempo seria compensado em uma segunda execução do processo, uma vez que a maior parte deste tempo foi dedicada à compreensão do processo em si (curva de aprendizado).

Outro ponto que deve ser destacado é a qualidade da especificação dos requisitos de qualidade que foram identificados. A partir da documentação gerada por cada equipe (ver Anexo A), é possível ver que os requisitos obtidos com o método proposto nesta dissertação se mostraram mais completos, sendo a única equipe que os especificou com detalhes quantificáveis, passíveis de validação e teste. Porém, a quantidade de requisitos identificados foi menor, quando comparado aos outros. Isso se deve ao fato do método proposto focar em apenas cinco requisitos de qualidade. Como apresentado no Capítulo 6, em trabalhos futuros outros refinamentos podem ser adicionados.

No decurso da prática foi observada uma maior dificuldade em acompanhar o processo Pires (RAMOS, 2014), mostrando-se o método menos intuitivo dos três utilizados, pelo menos do ponto de vista de usuários pouco experientes com atividades da engenharia de requisitos. Ao longo do experimentou observou-se ainda que existiram muitas dúvidas sobre os processos, geradas pela falta de conhecimentos em disciplinas que ainda não foram ministradas para eles, que cursavam o terceiro período do curso.

Os requisitos não-funcionais de confiabilidade e disponibilidade que foram elicitados e especificados de forma enfática no método proposto por este trabalho. Trouxeram em sua especificações o detalhamento de requisitos como o tempo para analisar falhas, o tempo de disponibilidade, período do dia que deve ser monitorado com mais intensidade, se existir algum outro requisito não-funcional que seja conflitante ele mostra e o usuário deverá como deve agir. Apenas foi citado de forma breve ou apenas identificados (elicitados) nos outros métodos usados, sem apresentar sequer uma descrição ou detalhamento do impacto do requisito nas regras de negócio do sistema.

A partir dos dados coletados durante a realização do experimento foi possível observa a contribuição do método para apoiar a engenharia de requisitos de qualidade que contemple a especificação de reconfigurações dinâmicas do sistema,pois os participantes que utilizaram o processo conseguiram elicitar os requisitos com maior detalhamento. Por exemplo a equipe que utilizou o *Brainstorm* apenas identificou o requisito de qualidade e não trouxe as características fundamentais para o desenvolvimento. Enquanto participantes que utilizaram o método proposto trouxeram não apenas a descrição como também os limites para variabilidade de disponibilidade do sistema, informações sobre a manutenabilidades, o

nível de segurança necessário e mais algumas outras informações que podem ser conferidas na seção de anexos A.

5.2.2.1 Ameaças à Validade do Experimento

Durante a execução do experimento para a avaliação do método proposto, surgiram alguns fatores que podem representar ameaças à validade do experimento. Uma destas ameaças foi a avaliação qualitativa que tornou o resultado relativamente subjetivo. A amostra de perfis para voluntários do experimento deveria ser diversificado, porém, o experimento foi executado em uma turma de engenharia de requisitos do terceiro período do fluxo padrão. Com isso, foi reduzida a diversidade de conhecimento dos participantes. Dessa forma, a interferência de fatores como experiência, conhecimentos correlatos, como projeto de software e de programação não foi avaliada.

Outra ameaça ao experimento foi a falta de treinamento das equipes para trabalhar com os métodos selecionados, isso ocorreu por existir grande dificuldades em reunir as equipes que iriam participar da execução e a falta de engajamento dos participantes para a realização do experimento.

6 CONCLUSÃO

A evolução atual das plataformas complexas e de serviços críticos de alta demanda evidencia a tendência de se ter softwares cada vez mais complexos e dinâmicos. Além do mais, a crescente dependência do software evidencia ainda mais a necessidade de preservar os requisitos de qualidade, tais como escalabilidade e disponibilidade, mesmo que para isso sejam necessárias adaptações dinâmicas no comportamento do software. Porém, as técnicas de engenharia de requisitos, que atualmente são aplicadas a sistemas estáticos, não se adapta a esse tipo de software. Durante a revisão bibliográfica sobre as soluções existentes para implementar aplicações em ambientes dinâmicos, viu-se a dificuldade existe para elicitar e especificar requisitos de software que envolvem adaptações dinâmicas.

Com isto, este trabalho apresentou um método para apoiar a engenharia de requisitos de qualidade que envolvem ajustes dinâmicos do software. O método proposto enfatiza as atividades de elicitação e especificação dos requisitos, incluindo a documentação das variações dinâmicas no software decorrentes da satisfação dos requisitos de qualidade.

O processo foi proposto para contribuir com o engenheiro de requisitos, seja ele experiente ou não, de forma a facilitar a descoberta e detalhamento dos requisitos de qualidade e suas possíveis adaptações dinâmicas.

O método proposto foi avaliado através de um experimento quanti-qualitativo no contexto de um ambiente dinâmico e com estudantes da disciplina de engenharia de requisitos do terceiro período do curso de ciência da computação.

Através da análise dos dados do experimento pode-se observa que o novo método se adaptou bem às regras de negócio da aplicação e tornou mais completa a especificação dos requisitos identificados. O processo induz o responsável pela engenharia de requisitos a compreender e detalhar as características principais de cada requisito de qualidade, incluindo aspectos quantificáveis que viabilizam validação e especificação de casos de teste. Além do mais, os requisitos especificados também consideram as possíveis formas que o sistema pode assumir quando o ambiente sofrer modificações, de forma a preservar o máximo possível os requisitos de qualidade.

Apesar de sua aplicação levar um pouco mais de tempo do que outras abordagens utilizadas para comparação, seu resultado é satisfatório e traz mais detalhes para que o desenvolvedor possa entender os requisitos de qualidade do sistema e seu funcionamento.

Uma das limitações encontradas foi a ausência de ferramentas que contribuíssem com a automatização de algumas fases do método. E este é um dos trabalhos futuros pretendidos. Outra limitação do método proposto é o seu foco de atuação em cinco requisitos de qualidade. Pretende-se evoluí-lo de forma a considerar outros requisitos de qualidade, além de detalhar

outros cenários de variação dinâmica dos requisitos já considerados.

REFERÊNCIAS

ÁGIL, M. Manifesto para o desenvolvimento ágil de software. **Disponıvel em: http://manifestoagil. com. br/. Acessado em**, v. 17, 2011.

APEL, S. et al. **Feature-Oriented Software Product Lines**. [S.l.]: Springer, 2013.

ASADI, M. et al. Requirements engineering in feature oriented software product lines: an initial analytical study. In: ACM. **Proceedings of the 16th International Software Product Line Conference-Volume 2**. [S.l.], 2012. p. 36–44.

ASIKAINEN, T.; MÄNNISTÖ, T.; SOININEN, T. Kumbang: A domain ontology for modelling variability in software product families. **Advanced Engineering Informatics**, Elsevier, v. 21, n. 1, p. 23–40, 2007.

BASS, L.; CLEMENTS, P.; KAZMAN, R. **Software Architecture in Practice**. [S.l.]: Addison-Wesley Professional, 2012.

BENCOMO, N.; LEE, J.; HALLSTEINSEN, S. How dynamic is your dynamic software product line? Lancaster University, 2010.

BOEHM, B. W.; BROWN, J. R.; KASPAR, H. Characteristics of software quality. North-Holland, 1978.

BRASIL, A. Bpm cbok v3. 0: Guia para o gerenciamento de processos de negócio-corpo comum de conhecimento. **2ª edição**, 2013.

CAPILLA, R. et al. An overview of dynamic software product line architectures and techniques: Observations from research and industry. **Journal of Systems and Software**, Elsevier, v. 91, p. 3–23, 2014.

CAPOTE, G. Bpm para todos: Uma visão geral, abrangente, objetiva e esclarecedora sobre gerenciamento de processos de negócio–bpm. **Gart Capote**, 2012.

CARVALHO, S. T. **Modelagem de Linha de Produto de Software Dinâmica para Aplicações Dinâmicas**. Tese (Doutorado) — Instituto de Computação, Universidade Federal Fluminense, Niterói, Brasil, 2013.

COPETTI, A. **Modelagem de Linha de Produto de Software Dinâmica para Aplicações Ubíquas** . Tese (Doutorado) — Instituto de Computação, Universidade Federal Fluminense, Niterói, RJ, Brasil, 2010.

EBERLEIN, C. K. A. Requirements engineering for software product lines. **The University of Calgary**, Citeseer, 2001.

FERNANDES, P.; WERNER, C.; MURTA, L. G. P. Feature modeling for context-aware software product lines. In: **SEKE**. [S.l.: s.n.], 2008. p. 758–763.

ISO/IEC. **ISO/IEC 9126. Software engineering – Product quality**. [S.l.]: ISO/IEC, 2001.

OLIVEIRA, R. F. de. **Formalização e Verificação de Consistência na Representação de Variabilidades**. Tese (Doutorado) — UNIVERSIDADE FEDERAL DO RIO DE JANEIRO, 2006.

PARRA, C. **Towards dynamic software product lines: Unifying design and runtime adaptations**. Tese (Doutorado) — Université des Sciences et Technologie de Lille-Lille I, 2011.

PEREIRA, P.; TORREÃO, P.; MARÇAL, A. S. Entendendo scrum para gerenciar projetos de forma ágil. **Mundo PM**, v. 1, p. 3–11, 2007.

PESSOA, L. M. Flexibilidade em linhas de produtos dinâmicas cientes de qualidade: uma abordagem baseada em linguagens específicas de domínio. 2014.

PRESSMAN, R. S. **Engenharia de software**. [S.l.]: McGraw Hill Brasil, 2011.

PROJECT, T. **TWE, Together Workflow Editor**. 1996. Disponível em: <http://www.together.at/index>.

QUEIROZ, P. G. G. **Uma abordagem de desenvolvimento de linha de produtos com uma arquitetura orientada a serviços**. Tese (Doutorado) — Universidade de São Paulo, 2009.

RAMOS, L. P. **Uma abordagem de gestão e desenvolvimento de automatização de processos de negócios com apoio de BPMS**. Tese (Doutorado) — Universidade de Fortaleza, 2014.

ROCHA, R. dos S. Linha de produto para gestão de processos de negócio incluindo aspectos dinâmicos. 2012.

SCHWABER, K. **Agile project management with Scrum**. [S.l.]: Microsoft press, 2004.

SCRUMSTUDY. Um guia para o conhecimento em scrum (guia sbok™). SCRUMstudy™, 2016.

SOARES, M. dos S. Metodologias ágeis extreme programming e scrum para o desenvolvimento de software. **Revista Eletrônica de Sistemas de Informação ISSN 1677-3071 doi: 10.5329/RESI**, v. 3, n. 1, 2004.

SOLINGEN, R. V. et al. Goal question metric (gqm) approach. **Encyclopedia of software engineering**, Wiley Online Library, 2002.

SOMMERVILLE, I. **Engenharia de Software**. [S.l.]: Addison-Wesley, Reading, 8ª Edição, MA, 2007.

TALIB, M. A. et al. Requirements for evolvable dynamic software product lines. In: **SPLC Workshops**. [S.l.: s.n.], 2010. p. 43–46.

WANGENHEIM, C. G. von; RUHE, G. Análise de custo e beneficio de mensuração baseada em gqm-um estudo de caso replicado.

ANEXO A – RESULTADO DA ELICITAÇÃO DE REQUISITOS CONSTRUIDA DURANTE O EXPERIMENTO

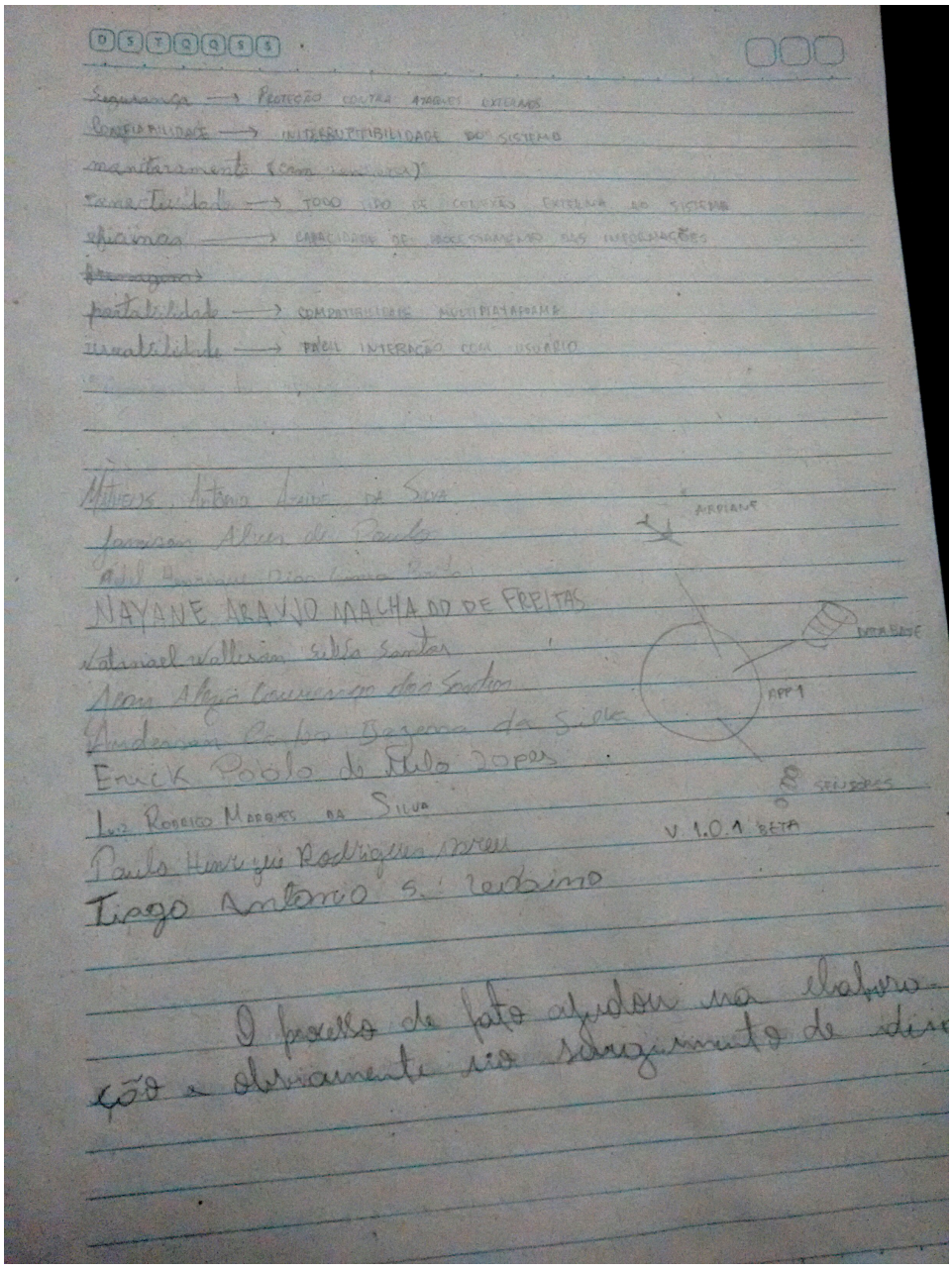
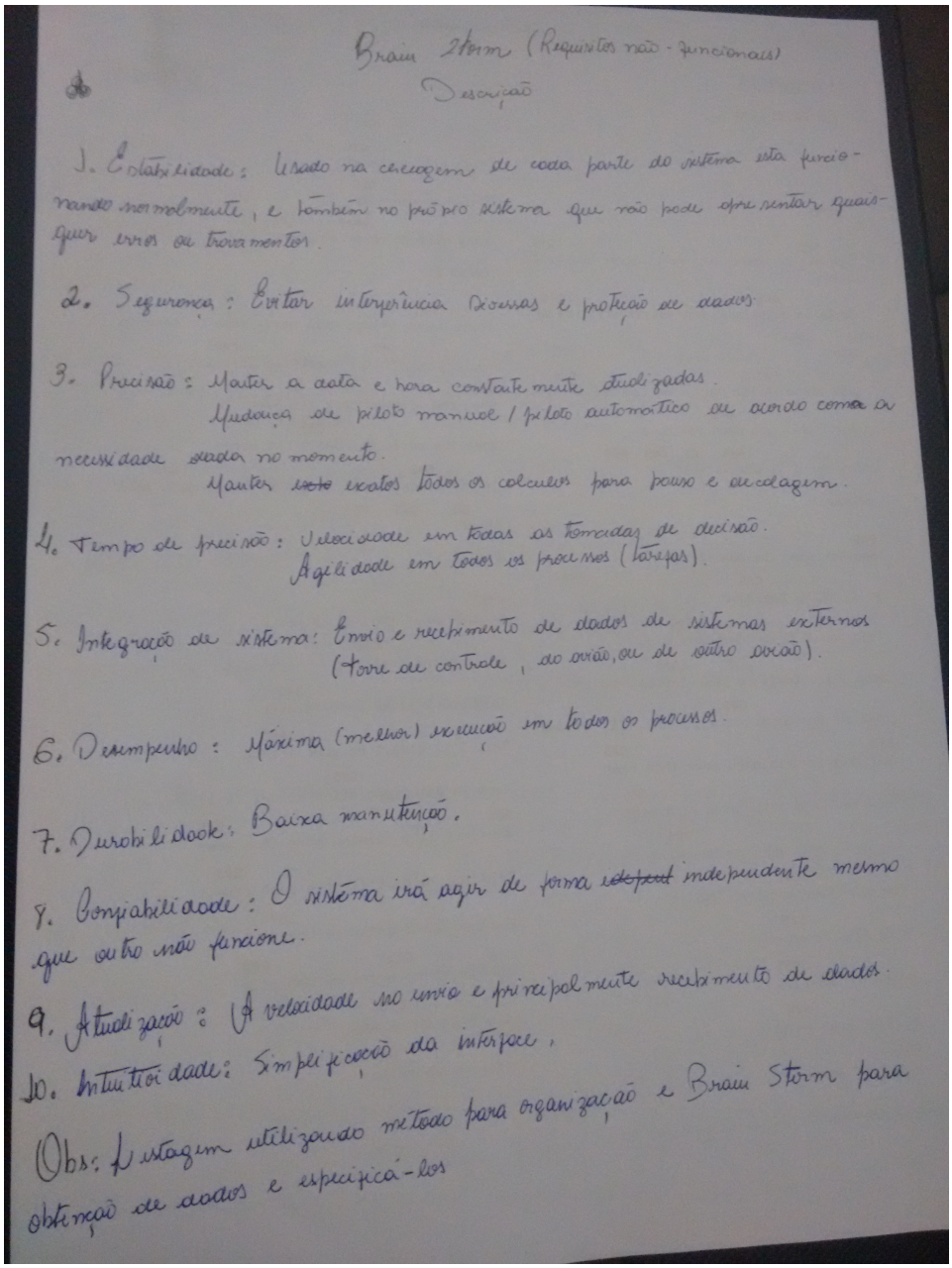


Figura 15 – Resultado da abordagem de (RAMOS, 2014)



Brain Storm (Requisitos não - funcionais)

Descrição

1. Estabilidade: Usado na checagem de cada parte do sistema esta funcionando normalmente, e também no próprio sistema que não pode apresentar quaisquer erros ou travamentos.

2. Segurança: Evitar interferências externas e proteção de dados.

3. Precisão: Manter a data e hora constantemente atualizadas.
Mudança de piloto manual / piloto automático de acordo com a necessidade exigida no momento.
Manter ~~exato~~ exatos todos os calculos para pouso e decolagem.

4. Tempo de precisão: Velocidade em todas as tomadas de decisão.
Agilidade em todos os processos (tarefas).

5. Integração de sistema: Envio e recebimento de dados de sistemas externos (torre de controle, do avião, ou de outro avião).

6. Desempenho: Máxima (melhor) execução em todos os processos.

7. Durabilidade: Baixa manutenção.

8. Confiabilidade: O sistema irá agir de forma ~~adaptat~~ independente mesmo que outro não funcione.

9. Atualização: A velocidade no envio e principalmente recebimento de dados.

10. Intuitividade: Simplificação da interface.

Obs: Usamos utilizando método para organização e Brain Storm para obtenção de dados e especificá-los

Figura 16 – Resultado do método Brainstorm

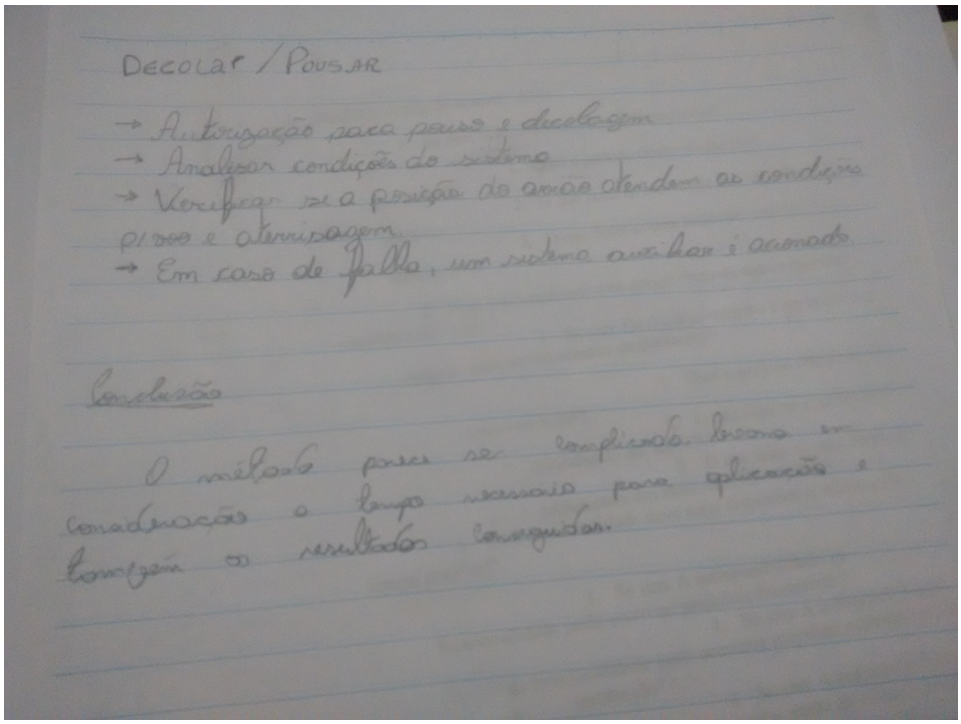
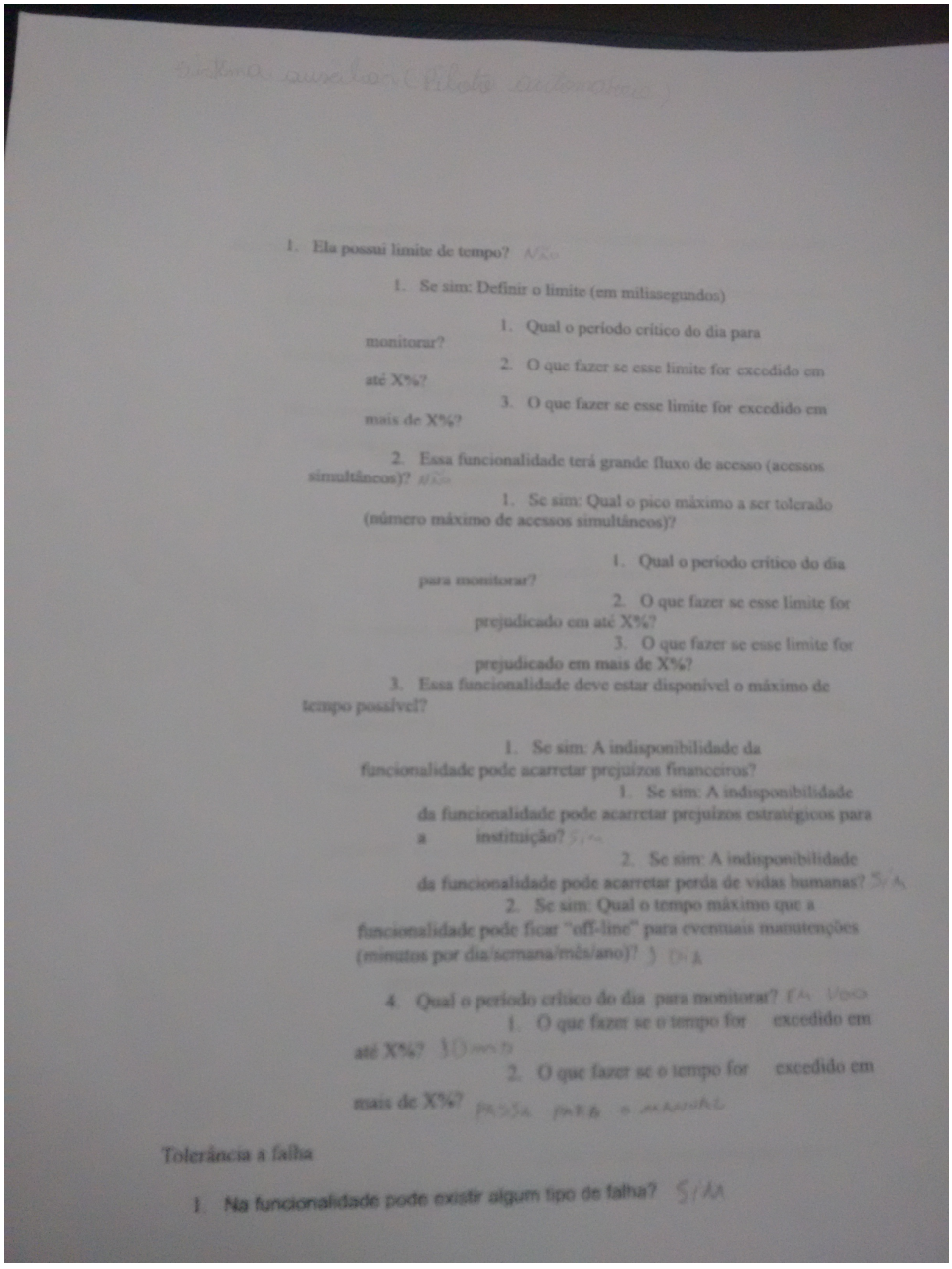


Figura 17 – Resultado do método proposto



1. Ela possui limite de tempo? NÃO

    1. Se sim: Definir o limite (em milissegundos)

        1. Qual o período crítico do dia para monitorar?
        2. O que fazer se esse limite for excedido em até X%?
        3. O que fazer se esse limite for excedido em mais de X%?

    2. Essa funcionalidade terá grande fluxo de acesso (acessos simultâneos)? NÃO

        1. Se sim: Qual o pico máximo a ser tolerado (número máximo de acessos simultâneos)?

            1. Qual o período crítico do dia para monitorar?
            2. O que fazer se esse limite for prejudicado em até X%?
            3. O que fazer se esse limite for prejudicado em mais de X%?

    3. Essa funcionalidade deve estar disponível o máximo de tempo possível?

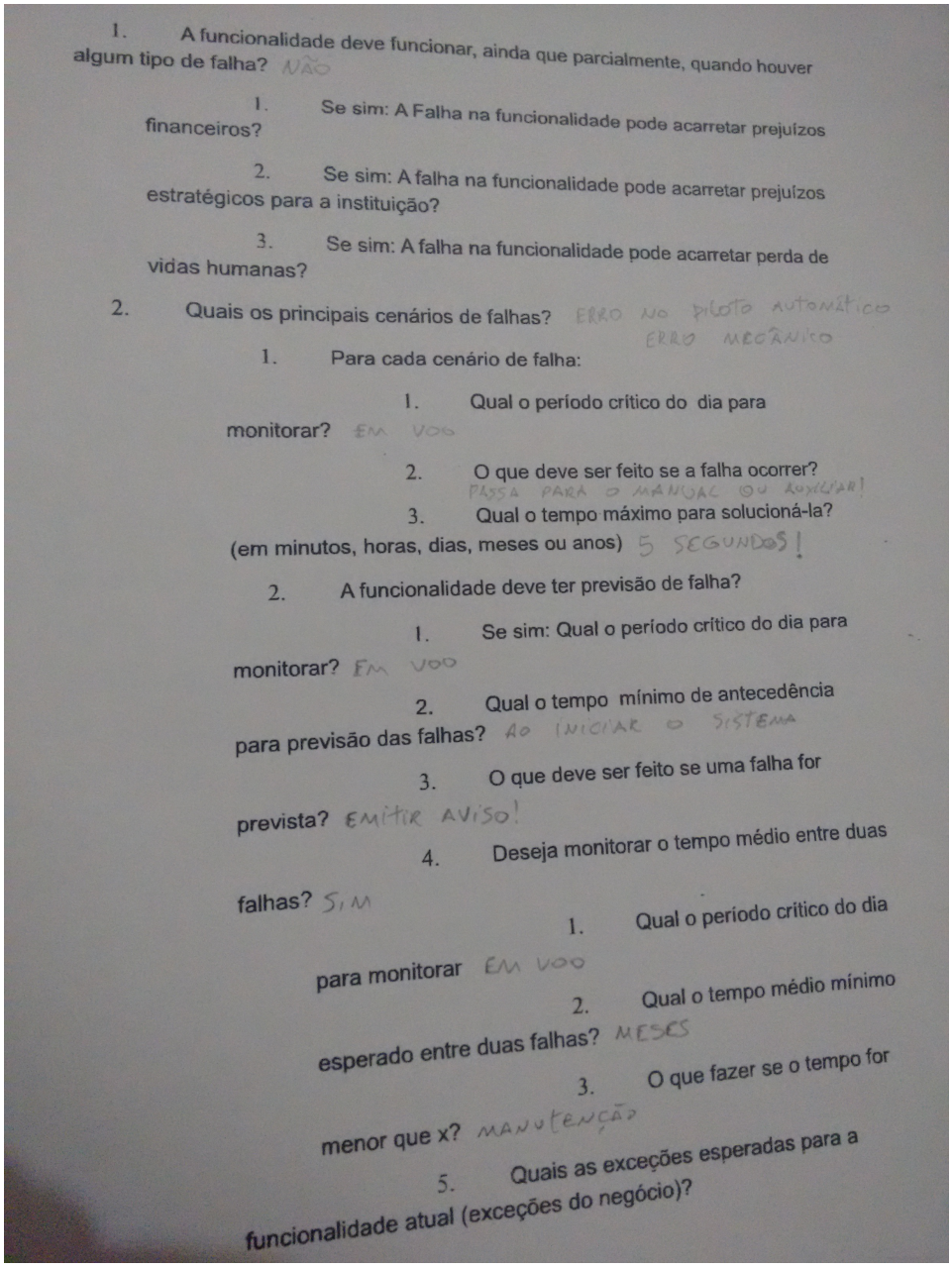
        1. Se sim: A indisponibilidade da funcionalidade pode acarretar prejuízos financeiros?
            1. Se sim: A indisponibilidade da funcionalidade pode acarretar prejuízos estratégicos para a instituição? SIM
            2. Se sim: A indisponibilidade da funcionalidade pode acarretar perda de vidas humanas? SIM
        2. Se sim: Qual o tempo máximo que a funcionalidade pode ficar "off-line" para eventuais manutenções (minutos por dia/semana/mês/ano)? 3 DIA

    4. Qual o período crítico do dia para monitorar? DIA TODO
        1. O que fazer se o tempo for excedido em até X%? 30 MIN
        2. O que fazer se o tempo for excedido em mais de X%? PASSA PARA O SEGUNDO

Tolerância a falha

1. Na funcionalidade pode existir algum tipo de falha? SIM

Figura 18 – continuação do Resultado do método proposto



1. A funcionalidade deve funcionar, ainda que parcialmente, quando houver algum tipo de falha? NÃO
    1. Se sim: A Falha na funcionalidade pode acarretar prejuízos financeiros?
    2. Se sim: A falha na funcionalidade pode acarretar prejuízos estratégicos para a instituição?
    3. Se sim: A falha na funcionalidade pode acarretar perda de vidas humanas?
2. Quais os principais cenários de falhas? ERRO NO PILOTO AUTOMÁTICO ERRO MECÂNICO
    1. Para cada cenário de falha:
        1. Qual o período crítico do dia para monitorar? EM VOO
        2. O que deve ser feito se a falha ocorrer? PASSA PARA O MANUAL OU AUXILIAR!
        3. Qual o tempo máximo para solucioná-la? (em minutos, horas, dias, meses ou anos) 5 SEGUNDOS!
    2. A funcionalidade deve ter previsão de falha?
        1. Se sim: Qual o período crítico do dia para monitorar? EM VOO
        2. Qual o tempo mínimo de antecedência para previsão das falhas? AO INICIAR O SISTEMA
        3. O que deve ser feito se uma falha for prevista? EMITIR AVISO!
        4. Deseja monitorar o tempo médio entre duas falhas? SIM
            1. Qual o período crítico do dia para monitorar EM VOO
            2. Qual o tempo médio mínimo esperado entre duas falhas? MESES
            3. O que fazer se o tempo for menor que x? MANUTENÇÃO
    5. Quais as exceções esperadas para a funcionalidade atual (exceções do negócio)?

Figura 19 – continuação do Resultado do método proposto