UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Henrique Michel Persch

ABORDAGEM PARA GERAÇÃO DE PROCESSOS DE SOFTWARE CONSISTENTES

Santa Maria, RS, Brasil
2017

Henrique Michel Persch

ABORDAGEM PARA GERAÇÃO DE PROCESSOS DE SOFTWARE CONSISTENTES

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Orientadora: Profª Drª. Lisandra Manzoni Fontoura

Santa Maria, RS, Brasil
2017

Ficha catalográfica elaborada através do Programa de Geração Automática
da Biblioteca Central da UFSM, com os dados fornecidos pelo(a) autor(a).

Persch, Henrique Michel
ABORDAGEM PARA GERAÇÃO DE PROCESSOS DE SOFTWARE CONSISTENTES / Henrique Michel Persch.- 2017.
88 p.; 30 cm

Orientadora: Lisandra Manzoni Fontoura
Dissertação (mestrado) - Universidade Federal de Santa Maria, Centro de Tecnologia, Programa de Pós-Graduação em Informática, RS, 2017

1. Processo de Software 2. Feature Models 3. Requisitos 4. Riscos 5. Métodos de Priorização I. Fontoura, Lisandra Manzoni II. Título.

Henrique Michel Persch

ABORDAGEM PARA GERAÇÃO DE PROCESSOS DE SOFTWARE CONSISTENTES

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Aprovado em 12 de Dezembro de 2017:

Lisandra Manzoni Fontoura, Dra. (UFSM)
(Presidente/Orientador)

Giliane Bernardi (UFSM)

Ricardo M. Czekster (UNISC)

Santa Maria, RS
2017

DEDICATÓRIA

Primeiramente a Deus e a minha família, que sempre estiveram ao meu lado me apoiando e dando forças para alcançar meus objetivos, aos meus amigos e colegas de trabalho que sempre estiveram me ajudando e colaborando para essa grande conquista.

RESUMO

ABORDAGEM PARA GERAÇÃO DE PROCESSOS DE SOFTWARE CONSISTENTES

AUTOR: Henrique Michel Persch
ORIENTADORA: Prof.ª. Drª. Lisandra Manzoni Fontoura

Escolher os componentes mais apropriados para compor um processo específico para um projeto e garantir a consistência deste não são tarefas triviais, exigindo grande conhecimento e envolvimento de um engenheiro de processos. Este trabalho visa elaborar uma abordagem semiautomática para geração de processos de software adaptados e consistentes que visa prevenir riscos identificados e priorizados para um projeto. A partir de mecanismos de seleção, definidos por meio de critérios de adaptação, e classificação, usando métodos de priorização, são selecionados os componentes de processos de software mais adequados ao processo de desenvolvimento. Neste trabalho, são utilizados riscos relacionados à Gerência de Requisitos como critérios de adaptação, pois muitos projetos falham devido a problemas relacionados a essa área em projetos de software. Alguns riscos de requisitos comuns em projetos de software são: requisitos que não refletem as reais necessidades dos usuários, alterações nos requisitos durante o desenvolvimento, dificuldade em encontrar um acordo comum entre desenvolvedores e usuários, requisitos incompletos e/ou inconsistentes. Essas dificuldades acarretam retrabalho, atrasos no cronograma, elevados custos e, principalmente, a insatisfação dos clientes e usuários do software. A abordagem proposta gera processos de software que incorporaram atividades que visam prevenir riscos de requisitos priorizados para o projeto. Para isso, realizou-se o levantamento de atividades da área de requisitos, na qual foram elaborados modelos de representação, denominados *feature models,* que representam regras e conduzem a seleção de atividades mais adequadas ao processo. Como principais resultados deste trabalho citam-se: elaboração de *feature model* para atividades de requisitos, associação de riscos a atividades que visam preveni-las, desenvolvimento de regras de consistência para validação de processos e automatização na criação de um processo de software. Todos esses componentes foram inseridos, validados e implementadas em uma ferramenta de apoio para exemplificar o uso da abordagem. Essa ferramenta está sendo desenvolvida no grupo de pesquisa em Engenharia de Software (PEnSO) da Universidade Federal de Santa Maria. Ao final, a abordagem proposta foi validada por meio da construção de um cenário de teste para demonstração do funcionamento da abordagem, e os resultados expressos para um cenário de desenvolvimento de processos de software foram analisados. Contudo, a abordagem se demonstrou eficaz e consistente no resultado apresentado.

Palavras-chave: Processo de Software, *Feature Models*, Requisitos, Riscos, Métodos de Priorização.

ABSTRACT

APPROACH FOR GENERATING CONSISTENT SOFTWARE PROCESSES

AUTHOR: Henrique Michel Persch
ADVISOR: Drª.Lisandra Manzoni Fontoura

Choosing the most appropriate components to compose a specific process for a project and ensuring the consistency of this are not trivial tasks, requiring great knowledge and involvement of a process engineer. This work aims to elaborate a semiautomatic approach for generating adapted and consistent software processes that aims to prevent identified and prioritized risks for a project. From the selection mechanisms, defined through adaptation criteria, and classification, using prioritization methods, the software process components most appropriate to the development process are selected. In this work, risks related to Requirements Management are used as adaptation criteria, since many projects fail due to problems related to this area in software projects. Some common requirements risks in software projects are: requirements that do not reflect actual user needs, changes in requirements during development, difficulty in finding a common agreement between developers and users, incomplete and / or inconsistent requirements. These difficulties lead to rework, delays in the schedule, high costs and, mainly, the dissatisfaction of customers and users of the software. The proposed approach generates software processes that incorporate activities that aim to prevent risks of prioritized requirements for the project. For this, the activities of the requirements area were surveyed, in which models of representation, called feature models, were elaborated, which represent rules and lead to the selection of activities more appropriate to the process. The main results of this work are: elaboration of feature model for requirements activities, association of risks to activities that aim to prevent them, development of consistency rules for process validation and automation in the creation of a software process. All of these components were inserted, validated and implemented in a support tool to exemplify the use of the approach. This tool is being developed in the research group in Software Engineering (PEnSO) of the Federal University of Santa Maria. In the end, the proposed approach was validated through the construction of a test scenario to demonstrate the operation of the approach, and the results expressed for a scenario of software process development were analyzed. However, the approach has been shown to be effective and consistent in the presented result.

Keywords: Software Process, Feature Models, Requirements, Risks, Prioritization Methods.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

A adaptação de processos é uma abordagem para customizar processos de software de acordo com as necessidades do projeto e da organização. Sendo assim, pode ser entendida como o ato de ajustar as definições ou particularizar os termos de uma descrição de processo geral com o intuito de derivar um novo processo aplicável a um ambiente alternativo e, provavelmente, menos geral. Essa adaptação deve definir e estabelecer o escopo e executar as mudanças de uma maneira sistemática e consistente (GINSBERG e QUINN, 1995).

As vantagens de uma boa definição de processos são: melhoria no desempenho, previsibilidade, confiabilidade do processo de software, aumento da produtividade, entre outras. Além disso, facilita a comunicação entre os membros da equipe e ajuda a educar os novos membros da equipe do projeto (XU e RAMESH, 2002).

Zakaria e Mahrin (2015) destacam desafios e dificuldades associados a adaptação de processos. Esta requer experiência, intuição, conhecimento do contexto do projeto, suas finalidades, e por fim, o comprometimento dos membros da equipe, especialmente o gerente de projeto e do engenheiro de processo (HURTADO e BASTARRICA, 2009) (XU e RAMESH, 2007).

Existem diferentes processos de desenvolvimento propostos na literatura, tais como: Rational Unified Process (RUP, 2001), Extreme Programming - XP (BECK, 2004), SCRUM (SCHWABER e SUTHERLAND, 2013). Porém, cada processo possui fatores que são mais adequados em projetos com determinadas características. É tarefa do engenheiro de processos selecionar os processos de acordo com as características do projeto e da organização, pois cada projeto é único e possui especificidades próprias (CAMERON, 2002). Desta forma, podemos afirmar que não existe um *framework* padrão que pode ser usado para definir o processo de software em todos os ambientes de projeto (JENERS, *et al.*, 2013), visto que existem diversos fatores envolvidos no ambiente de desenvolvimento de projeto de software (CLARKE e CONNOR, 2012).

Normalmente, as organizações adotam abordagens *ad hoc*, ou seja, com pouca orientação e utilização de um método mais sistemático (RUI, *et al.*, 2009). Isso ocorre, principalmente, por causa das limitações de tempo (HE, *et al.*, 2009), ausência de diretrizes claras (XU e RAMESH, 2008) e falta de orientação, enfrentados pelos membros da equipe do projeto. A intuição e a perícia de um gerente de projeto experiente ou de um engenheiro de processos são necessárias nesta abordagem.

Outra dificuldade é de que os membros da equipe tendem a selecionar o processo antigo ou utilizar o processo que lhes é familiar sem considerar as características de contexto do projeto. E por fim, e talvez o mais importante, é de que os critérios e fatores de seleção em um processo de adaptação de software ainda é vaga e imprecisa. A seleção é sempre dada a uma pessoa experiente e que tem conhecimento na realização da adaptação do processo de software sem qualquer orientação ou estrutura adequada (ZAKARIA e MAHRIN, 2015).

Nos últimos anos vêm aumentado a pesquisa na área de adaptação de processos para resolução dos problemas citados. Akbar *et al.* (2011) realizaram uma revisão sistemática com o estado da arte nas áreas: processos ágeis, melhoria de processos de software e o domínio de adaptação de processos de software. Ruiz, *et al.* (2012) investigaram os requisitos e construtores na condução da adaptação de processos de software. Kalus e Kuhrmann (2013) produziram um catálogo que consiste nos critérios e impactos na adaptação de processos de software.

1.1. DEFINIÇÃO DO PROBLEMA

Atualmente, a grande maioria das organizações baseiam-se e apoiam suas decisões essencialmente em software. Com a importância crescente de software e o surgimento de novos paradigmas de desenvolvimento de software, o mercado impõe demandas e desafios nos processos de software.

Softwares são desenvolvidos por centenas ou milhares de pessoas divididas em equipes. Elas desempenham atividades diferentes, que são determinadas pelos processos de de software. Esses processos devem ser bem definidos, pois desta forma, as empresas de desenvolvimento conseguem alcançar os objetivos com maior qualidade e segurança. Para isso, existem diversos modelos de processo de software que representam as atividades do mundo real de um processo de software.

Assim, qualquer modelo de processo de software tem que modelar adequadamente o processo do mundo real e deve atender às exigências especificadas em cada fase do processo. Porém, a determinação do processo de software é tarefa árdua e requer experiência e precisão em sua elaboração. Seguindo essa linha de raciocínio, a motivação deste trabalho é facilitar a elaboração e adaptação de processos de software diminuindo a dependência de um especialista em processos e melhorando a qualidade dos processos gerados.

Segundo Ahn, et al., (2003) e Xu e Ramesh, (2002), como a adaptação de processos de software é uma atividade que requer uso intensivo de vários tipos de conhecimento, boa parte de sua realização depende de profissionais especializados (denominados de Engenheiros de Processos) e de atividades manuais, o que a torna consumidora de tempo, custosa e sujeita a erros. Desta forma, o aumento no nível de automatização nesta atividade traz consigo os seguintes benefícios:

• Redução de tempo e custo despendidos durante a definição de processos de software, o que tem impacto direto no prazo de entrega do produto de software final requerido;

• Simplificação do trabalho do Engenheiro de Processos, aumentando assim sua produtividade e a possibilidade de obtenção de processos de software mais adequados à situação corrente.

Para isto, este trabalho objetiva *desenvolver uma abordagem para adaptação de processos de software semiautomática com foco na melhoria da qualidade dos processos, de modo a garantir que os processos gerados sejam completos e consistentes, e ainda reduzir a necessidade de interferência do engenheiro de processos.*

Após a construção do objetivo geral, foi possível desmembra-lo nos seguintes objetivos específicos:

i) Construir um repositório de componentes de processos com diferentes fontes de pesquisa na literatura;

ii) Propor uma abordagem que classifique e priorize os componentes conforme suas finalidades e considere riscos inerentes ao projeto, para inseri-las em um processo de software;

iii) Formular uma sistemática para seleção dos componentes de processo, que considere a obrigatoriedade e as dependências entre os elementos do processo;

iv) Construir regras para garantia da consistência e completude do processo;

v) Implementar uma ferramenta de apoio que facilite a criação e a alteração do processo adaptado.

1.2. ESCOPO DA PESQUISA

A fim de satisfazer os objetivos gerais e específicos elencados, propôs-se o uso de diferentes recursos da literatura, os quais possibilitaram a construção da seguinte hipótese para o desenvolvimento deste trabalho: *É possível criar uma abordagem para semi-automatizar a seleção dos componentes de processo, considerando o contexto do projeto e os*

riscos inerentes como as informações de entrada, e com base em um mecanismo de análise multicritério e regras de consistência, gerar um processo de software adaptado e consistente a determinado projeto.

Neste trabalho, a área de pesquisa foi limitada a gerência de requisitos pelos motivos: i) uma boa definição e análise dos requisitos é a base para o desenvolvimento de software de sucesso; ii) os requisitos estão presentes em todas as fases de desenvolvimento de um projeto; iii) os requisitos são o primeiro passo para a concepção de um software e estão em constante consulta e transformação. Pressman (2011) menciona que "projetar um programa de computador elegante que resolva o problema errado não atende as necessidades de ninguém. Por isso, é importante entender o que o cliente quer antes de começar a projetar e construir".

Em relação aos critérios de adaptação, optou-se por utilizar a identificação dos principais riscos de um projeto de desenvolvimento de software. O motivo para utilização de riscos como critérios de adaptação se deve ao próprio PMBOK (PMI, 2013), que apresenta uma área de conhecimento voltada para descrever as melhores práticas no gerenciamento dos riscos. Segundo PMBOK, "O risco do projeto é um evento ou condição incerta que, se ocorrer, provocará um efeito positivo ou negativo em um ou mais objetivos do projeto tais como escopo, cronograma, custo e qualidade". Segundo Pinna e Carvalho (2008), riscos não gerenciados de forma adequada podem comprometer a qualidade do produto final, a expectativa do cliente pode não ser atendida e a equipe, que precisa conviver com ansiedades e conflitos durante a vida do projeto, pode ter sua produtividade reduzida.

Como contribuições deste trabalho, pode-se destacar:

- Elaboração de um repositório de componentes de processos de requisitos;
- Construção e definição de uma metodologia para caracterização de projetos;
- Implementação de uma abordagem com análise multicritério e construção de regras de consistência;
- Levantamento e análise dos principais riscos relativos a requisitos em projetos de software;
- Criação de uma sistemática para automatização na criação de processos de software consistentes e completos.

1.3. ESTRUTURA DA DISSERTAÇÃO

Este trabalho está estruturado da seguinte maneira: o Capítulo 2 apresenta o referencial teórico, abordando conceitos como processos de software, engenharia de requisitos, modelos de processo, riscos, modelos de variabilidades e métodos de decisão multicritério. No Capítulo 3 é apresentado o desenvolvimento da abordagem proposta para satisfazer os objetivos do trabalho. Já no Capítulo 4 é apresentada a ferramenta de apoio *Otimized Process Tool* a qual é utilizada para mapear a sistemática da abordagem proposta apresentada no desenvolvimento do trabalho, e por fim, nos Capítulos 5, 6 e 7 é descrita a validação da proposta a partir de um cenário de teste elaborado, os trabalhos relacionados, resultados e considerações finais, respectivamente.

2. REVISÃO DA LITERATURA

Neste capítulo são abordados conceitos que envolvem o desenvolvimento do trabalho, contribuindo para a compreensão e análise da abordagem proposta. Na Seção 2.1 são descritos processos de software, incluindo a diferenciação de métodos ágeis e planejados. Na Seção 2.2 são abordados conceitos de engenharia de requisitos descritos no RUP, CMMI, SWEBOK, ISO 12207 e na literatura técnica. Na Seção 2.3 são descritos conceitos relacionados a riscos em gerência de requisitos. Por fim, na Seção 2.4 são descritos modelos de variabilidade e na Seção 2.5 os métodos de decisão.

2.1 PROCESSOS DE SOFTWARE

Um processo de software é visto como um conjunto completo de atividades de engenharia de software necessárias para transformar os requisitos de um usuário em um produto de software considerando um contexto (ARMBRUST *et al.,* 2009). De acordo com Pressman (PRESSMAN e MAXIM, 2016), processo é um conjunto de atividades, ações e tarefas realizadas na criação de algum produto de trabalho. No contexto da engenharia de software, um processo não é uma prescrição rígida de como desenvolver um software. Ao contrário, é uma abordagem adaptável que possibilita as pessoas (equipe de software) realizar o trabalho de selecionar e escolher o conjunto apropriado de ações e tarefas.

Existem diversos modelos de desenvolvimento de software como: modelo cascata, modelo incremental, modelo de prototipação, modelo espiral (PRESSMAN e MAXIM, 2016) e RUP (RUP, 2001).

A questão é que projetos de desenvolvimento de software raramente adotam metodologias rígidas conforme os modelos de desenvolvimento tradicionais (CESARE, *et al.*, 2008). Além disso, devido às particularidades de cada projeto de desenvolvimento, os processos de software também precisam ser adaptados de acordo com as especificidades do projeto, para atender as necessidades da organização. A adaptação com base nas características do projeto é realizada por meio de alterações nos elementos que compõem os processos, chamados elementos de processo, e em suas relações, principalmente em tarefas, artefatos e papéis (MARTINEZ-RUIZ *et al.*, 2011).

A adaptação de processo de software refere-se à atividade de adequar um processo para atender às necessidades de um projeto específico (XU e RAMESH, 2007). A técnica de

adaptação de processos já é reconhecida em normas como ISO/IEC 15504 (ISO/IEC 15504, 2008), ISO/IEC 12207 (ISO/IEC 12207, 2008) e o CMMI (CMMI, 2013).

Um processo de adaptação consiste em excluir, modificar ou ainda, adicionar novos componentes ao processo padrão. A partir dele é criado um processo de desenvolvimento único e específico para o projeto a ser desenvolvido, ou seja, a adaptação de um processo está relacionada com a customização de um processo de desenvolvimento, voltada para atender às necessidades de uma organização e/ou um projeto.

Porém, as consequências da realização de uma má adaptação do processo pode levar a dois grandes problemas: em primeiro lugar, o cumprimento do orçamento do projeto e do tempo de desenvolvimento do projeto podem ficar comprometidos, além da qualidade do produto que depende diretamente da qualidade do processo de software usado no seu desenvolvimento. Em segundo lugar, processos inadequados podem envolver atividades desnecessárias que levam a um desperdício de tempo e dinheiro ou a omissão de atividades que são necessárias, o que pode afetar a qualidade do produto final (PEDREIRA, *et al.*, 2007).

Diversas abordagens foram desenvolvidas e implementadas para adaptação de processos. Dentre essas pode se citar: Linhas de Processos de Software (LPrS), V-Model (JOHANSSON, 1999), sistema baseado em regras (HENNINGER, 1998). LPrS surgiram a partir de Linhas de Produto e visam à elaboração de processos consistentes possibilitando a reutilização de componentes previamente definidos. (JAUFMAN e MÜNCH, 2005; ROMBACH, 2006). LPrS é uma forma de adaptação de processos que possui os seguintes objetivos: i) aumento da qualidade e adequação dos processos gerados; ii) representação de variabilidades e semelhanças entre os processos para potencializar a reutilização; e iii) diminuição dos riscos de uma adaptação inadequada do processo (JAUFMAN e MÜNCH, 2005).

O V-Model considera que inicialmente é necessário identificar os objetivos, partes interessadas e descrever os requisitos do projeto, para depois o engenheiro de software iniciar uma série de medidas de eficácia e desempenho, utilizados para avaliar os vários estágios do projeto (JOHANSSON, 1999). Essa abordagem virou um padrão da indústria de software depois de 1980 e, após o surgimento da Engenharia de Sistemas, tornou-se um conceito padrão em todos os domínios da indústria.

Henninger (1998) apresenta uma abordagem que utiliza um sistema baseado em regras, organizadas sob a forma de árvores de decisão, para adaptar processos de software para as necessidades específicas de projetos.

Quando se trata de conceitos de processos de software e adaptação de processos, inevitavelmente, vem à tona a discussão sobre métodos ágeis e processos planejados. As metodologias consideradas tradicionais, também chamadas de "pesadas" ou processos planejados, têm como característica marcante serem divididas em etapas e/ou fases. Essas fases são muito bem definidas e englobam atividades como Análise, Modelagem, Desenvolvimento e Testes. O foco principal das metodologias tradicionais é a previsibilidade dos requisitos do sistema, que traz a grande vantagem de tornar os projetos completamente planejados, facilitando a gerência do mesmo, mantendo sempre uma linha, caracterizando o processo como bastante rigoroso. Como exemplo de metodologia planejada cita-se o *RUP - Rational Unified Process.*

Os métodos ágeis surgem em forma de contraposto aos métodos planejados, visando resultados rápidos a fim de atender as necessidades dos clientes. "Os métodos ágeis são adaptativos e não preditivos, como é o caso das metodologias tradicionais" (COCKBURN, 2001). Os métodos ágeis contam com uma abordagem iterativa para especificação, desenvolvimento e entrega de software, e foram criados principalmente para apoiar o desenvolvimento de aplicações de negócios nas quais os requisitos de sistema mudam rapidamente durante o processo de desenvolvimento (SOMMERVILLE, 2011). Dentre os métodos ágeis, podemos destacar como os mais populares: eXtreme Programming (XP) (BECK, 2004) e Scrum (SCHWABER e BEEDLE, 2003).

2.1.1. Contextualização dos Processos

Existe um consenso de que os processos são adequados a contextos específicos, que podem ser determinados por características da organização ou projeto (BOEHM; TURNER, 2003; COCKBURN, 2004; KRUCHTEN, 2013).

Martins (*et. al*, 2011), em uma publicação no XXXI Encontro Nacional de Engenharia de Produção, propõe uma investigação do processo de desenvolvimento empregado em empresas da indústria de software. Na pesquisa foram investigadas oito empresas de desenvolvimento de software situadas na região metropolitana de Recife, que apresentam metodologias distintas de desenvolvimento de software como RUP, Scrum, XP, Cascata, entre outras. De acordo com a pesquisa, as empresas foram selecionadas por acessibilidade, mas de forma a garantir heterogeneidade quanto ao porte e ao modelo de negócio das empresas.

Quanto ao instrumento de coleta realizado pelos autores, tem-se a aplicação de um formulário com o intuito de caracterizar a metodologia e as práticas empregadas no processo de desenvolvimento de software. Também foi utilizada a entrevista e a observação para analisar, de forma detalhada, o processo de desenvolvimento de software.

A partir da análise e pesquisa realizada por Martins (*et. al*, 2011), foi possível extrair as características apresentadas na Tabela 1.

Tabela 1 - Características dos processos de Desenvolvimento de Software

Características dos processos de Desenvolvimento de Software
Análise de requisitos profunda e detalhada
Elaboração da descrição detalhada da arquitetura
Teste do sistema completo antes de mandar para o cliente
Definição dos processos e ferramentas
Projeto implementado conforme previamente definido
Mudanças de requisitos bem-vindas, até mesmo perto da entrega do produto final
Muitos documentos necessariamente produzidos
A comunicação registrada e formalmente documentada
Verificação contínua da qualidade
Contato diário com os clientes
Clara divisão do trabalho dentro da empresa
Nível de Formalismo Desejado

Fonte: Martins (*et. al*, 2011)

É possível identificar na literatura outros modelos de caracterização de processos muito utilizados, como *Octopus Model* proposto por Kruchten (2013), Boehm e Turner (2003) propõe cinco características para definição do contexto e, por fim, Alistair Cockburn (2004) também propõem que processos são definidos baseados em em: a) tamanho, b) criticidade, e c) habilidades. Neste trabalho optou-se por selecionar as características da Tabela 1, pois identificou-se que são fatores mais voltados para a caracterização do processo em si, e não do projeto conforme visualizado em outros modelos citados. Acredita-se que essa diferenciação na caracterização irá apresentar melhores resultados na adaptação de processos.

Posteriormente, será apresentado que as características dos processos destacados serão utilizadas para contribuir na análise das atividades concretas mais coerentes para o processo a ser elaborado. Essas características são informadas no início da elaboração do processo de desenvolvimento de software, em formato de contexto do projeto.

2.2 ENGENHARIA DE REQUISITOS

O amplo espectro de tarefas e técnicas que levam a um entendimento dos requisitos é chamado de engenharia de requisitos. A engenharia de requisitos é uma ação de engenharia de software importante que se inicia durante a atividade de comunicação e continua na modelagem. Ela deve ser adaptada às necessidades do processo, do projeto, do produto e das pessoas que estão realizando o trabalho (PRESSMAN, 2016). Os requisitos de um sistema são as descrições do que o sistema deve fazer, os serviços que oferecem e as restrições a seu funcionamento (SOMMERVILLE, 2011).

A engenharia de requisitos fornece os mecanismos para entender as necessidades do cliente, analisando suas expectativas, avaliando a viabilidade e negociando uma solução, validando as especificações e gerenciando as necessidades à medida que estas são transformadas em um sistema.

A partir da análise de diferentes autores (SOMMERVILLE, 2011; PRESSMANN, 2016; BOURQUE, P. e FAIRLEY, R. E., 2014), é possível extrair as seguintes fases essenciais na engenharia de requisitos:

a) Elicitação de Requisitos: diz respeito à origem dos requisitos de software. É a primeira fase de construção de uma compreensão do problema. Um dos princípios fundamentais de um bom processo de elicitação de requisitos é o de uma comunicação eficaz entre as várias partes interessadas (BOURQUE, P. e FAIRLEY, R. E., 2014);
b) Análise de Requisitos: essa etapa visa verificar alguns aspectos importantes, como (i) detectar e resolver conflitos entre requisitos; (ii) descobrir os limites do software e como deve interagir com o seu ambiente operacional; (iii) elaborar requisitos de sistema para derivar os requisitos de software (BOURQUE, P. e FAIRLEY, R. E., 2014). Normalmente nessa etapa, ocorrem tarefas de classificação e organização de requisitos, além da priorização e a negociação destes requisitos (SOMMERVILLE, 2011);
c) Especificação de Requisitos: visa escrever os requisitos de usuário e de sistema em um documento de requisitos. Os requisitos devem ser claros, inequívocos, de fácil compreensão, completos e consistentes. Essa etapa permite uma avaliação rigorosa dos requisitos antes do projeto começar e reduz o redesenho posterior (BOURQUE, P. e FAIRLEY, R. E., 2014). A formalidade e o formato de uma especificação variam com o tamanho e a complexidade do software a ser construído (PRESSMAN, 2016);

d) Validação dos Requisitos: visa verificar se os requisitos definem o sistema que o cliente realmente quer (SOMMERVILLE, 2011). Ela se sobrepõe a análise, uma vez que está preocupada em encontrar problemas com os requisitos. A validação de requisitos examina a especificação para garantir que todos os requisitos de software tenham sido declarados de forma não ambígua; que as inconsistências, omissões e erros tenham sido detectados e corrigidos; e que os artefatos estejam de acordo com os padrões estabelecidos para o processo, projeto e produto (PRESSMAN, 2016);
e) Gerenciamento dos Requisitos: visa a compreensão e controle das mudanças nos requisitos do sistema (SOMMERVILLE, 2011). É um conjunto de atividades que ajuda a equipe de projeto a identificar, controlar e acompanhar as necessidades e suas mudanças à medida que o projeto prossegue (PRESSMANN, 2016).

Investir em uma boa análise e especificação de requisitos trará benefícios que garantirão o bom andamento de todas as fases posteriores do projeto, permitindo assim, que o cliente fique satisfeito e que seu projeto seja um sucesso. Porém, não existe um processo de definição de requisitos ideal para todos os projetos, é necessário que exista uma adaptação de acordo com as características do projeto que será desenvolvido.

Na literatura encontramos diversos processos de software, além de autores, que apresentam diversas atividades e tarefas que refletem a aplicação da engenharia de requisitos dentro de um processo de desenvolvimento de software. Neste trabalho, foram levantados alguns dos principais processos de engenharia de requisitos, juntamente com as diversas atividades propostas por cada um destes. A seguir, descrevemos de forma breve cada uma das fontes de pesquisa utilizadas neste trabalho para mapear todas as atividades que contemplam a engenharia de requisitos:

2.2.1. RUP – Rational Unified Process

O Processo Unificado da Rational, conhecido como RUP (*Rational Unified Process*) (RUP, 2001), é um processo de engenharia de software criado para apoiar o desenvolvimento orientado a objetos, fornecendo uma forma sistemática para se obter vantagens no uso da UML (*Unified Modeling Language*). O RUP oferece uma abordagem baseada em disciplinas para atribuir tarefas e responsabilidades dentro de uma organização. O principal objetivo do RUP é garantir que o software resultante tenha um alto nível de qualidade conforme a necessidade do usuário. Ele também garante que um software seja produzido dentro do tempo

e custo especificado. O ciclo de vida do RUP é dividido em 4 fases diferentes: Concepção, Elaboração, Construção e Transição (RUP, 2001).

As atividades relacionadas a requisitos são organizadas na disciplina de Requisitos. No RUP, a finalidade da disciplina é: i) estabelecer e manter concordância com os clientes e outros envolvidos sobre o que o sistema deve fazer; ii) oferecer aos desenvolvedores do sistema uma compreensão melhor dos requisitos do sistema; iii) definir as fronteiras do sistema (ou delimitar o sistema); iv) fornecer uma base para planejar o conteúdo técnico das iterações; v) fornecer uma base para estimar o custo e o tempo de desenvolvimento do sistema; e vi) definir uma interface de usuário para o sistema, focando nas necessidades e metas dos usuários. Em seu contexto, o RUP apresenta diversas atividades e tarefas relacionadas a essa disciplina, sendo minuciosamente descritas e de fácil compreensão.

2.2.2. Capability Maturity Model – Integration

O CMMI (*Capability Maturity Model Integration*) é um modelo de maturidade de melhoria de processos para o desenvolvimento de produtos e serviços. Trata-se de um modelo que está atualmente na versão 1.3 (CMMI, 2013). O objetivo do CMMI para desenvolvimento é auxiliar as organizações na melhoria de seus processos de desenvolvimento e manutenção de produtos e serviços. Consiste das melhores práticas relativas às atividades de desenvolvimento aplicadas a produtos e serviços. Ele abrange práticas que cobrem o ciclo de vida do produto desde a concepção até a entrega e manutenção, e se concentra no trabalho necessário para construção do produto em sua totalidade. O modelo é composto de 22 áreas de processo. Uma área de processo é dividida em objetivos e práticas. Os objetivos são componentes requeridos do modelo que descrevem as características que devem ser implementadas para satisfazer a área de processo.

As atividades de gerenciamento de requisitos podem ser identificadas principalmente na área de processos de nível de maturidade 2, Gestão de Requisitos. Além disso, pode-se relacionar a essa área outras como: Desenvolvimento de Requisitos, Planejamento de Projeto e Monitoramento e Controle de Projeto. Na área de processo Gestão de Requisitos é possível identificar uma relação de metas e práticas específicas que envolvem a engenharia de requisitos.

2.2.3. Guide to the Software Engineering Body of Knowledge

O SWEBOK (BOURQUE, P.e FAIRLEY, R. E., 2014) é um guia de uso e aplicação das melhores práticas de Engenharia de Software. Ele foi desenvolvido com conhecimentos recolhidos no período de quatro décadas e revisado por inúmeros profissionais de diversos países envolvidos com a Engenharia de Software. O SWEBOK versão 3 foi expandida para incluir 15 áreas de conhecimento, sendo que, cinco novos KAs (*knowledge áreas*) foram adicionadas a versão anterior (BOURQUE, P.e FAIRLEY, R. E., 2014). Os objetivos do SWEBOK englobam:

- ✓ Promover uma visão consistente da engenharia de software no mundo;
- ✓ Clarear e marcar as fronteiras entre a engenharia de software e as outras disciplinas relacionadas;
- ✓ Caracterizar o conteúdo da disciplina de engenharia de software;
- ✓ Classificar em tópicos a área de conhecimento da engenharia de software (programas);
- ✓ Prover uma fundação para o desenvolvimento do currículo, para certificação individual e para licenciamento de material.

No guia SWEBOK é encontrada uma área de conhecimento específica sobre requisitos, na qual apresenta uma preocupação com a elicitação, análise, especificação, e validação de requisitos de software e a gestão dos requisitos durante todo o ciclo de vida do produto de software. A partir desta área de conhecimento é possível extrair atividades e tarefas para cada fase do ciclo de vida do processo.

2.2.4. ISO/IEC 12207

A Norma ISO/IEC 12207 (ISO/IEC 12207, 2008), publicada em 1995, contempla os processos de ciclo de vida de software. Tem por objetivo auxiliar os envolvidos na produção de software a definir seus papeis, por meio de processos bem definidos proporcionando às organizações que a utilizam um melhor entendimento das atividades a serem executadas nas operações que envolvem, de alguma maneira, software (ROCHA *et al.*, 2001).

A estrutura da norma é flexível, modular e adaptável às necessidades de quem a utiliza. A norma é composta por um conjunto de processos, atividades e tarefas que podem ser adaptados de acordo com os projetos de software. Os processos são agrupados, por uma questão de organização, de acordo com a sua natureza, ou seja, o seu objetivo principal no ciclo de vida de software. Esse agrupamento resultou em quatro diferentes classes de

processos, que são: Processos fundamentais, Processo de apoio, Processos organizacionais e Processos de adaptação.

A ISO/IEC 12207 agrupa as atividades que podem ser realizadas durante o ciclo de vida de um software em sete grupos de processos. Cada um desses grupos é descrito em termos do seu propósito e resultados desejados, envolvendo a lista de atividades e tarefas que precisam ser realizadas para alcançar esses resultados. As atividades relacionadas a engenharia de requisitos estão relacionadas ao grupo Processos Técnicos (*Technical Processes*) e Processos de Implementação de Software (*Software Implementation Processes*).

2.2.5. Literatura Técnica

Como duas grandes referências da literatura na área de engenharia de software são os autores Ian. F. Sommerville e Roger. S. Pressman.

Ian F. Sommerville, é um professor britânico, autor do livro "Engenharia de Software" que está em sua 10ª edição (SOMMERVILLE, 2015), bem como uma série de outros livros e artigos. Em suas publicações, Sommerville apresenta uma série de atividades que devem ser realizadas para o levantamento e a especificação de requisitos dentro das organizações de desenvolvimento de software. Além disso, afirma que "o nível de detalhes que você deve incluir em um documento de requisitos depende do tipo de sistema em desenvolvimento e o processo usado". Destaca ainda, os diferentes processos relacionados a requisitos: especificação, elicitação e análise, validação e gerenciamento.

Roger S. Pressman é em engenheiro de software, escritor e consultor, norte-americano. Pressman é uma autoridade reconhecida internacionalmente nas tecnologias em melhoria de processos de software e engenharia de software (PRESSMAN e MAXIM, 2016). Em suas publicações, Roger S. Pressman defende que qualquer abordagem de engenharia (inclusive da engenharia de software) deve estar fundamentada em um comprometimento organizacional com a qualidade. Pressman destaca em diferentes capítulos conceitos sobre engenharia e modelagem de requisitos, nos quais aborda diversas atividades e métodos práticos para aplicação.

2.3. RISCOS EM REQUISITOS DE SOFTWARE

Riscos são eventos futuros e incertos que afetam os objetivos de um projeto ou uma organização, sendo necessária a gerência dos mesmos para evitar consequências

negativas no projeto. Gerência de riscos envolve atividades e medidas a serem tomadas para prevenir que riscos afetem negativamente os projetos e maximizem as consequências dos efeitos positivos. Atualmente, a gerência de riscos é vista como uma das atividades fundamentais em projetos de software, porém nem todas as empresas a utilizam, pois há a necessidade de uma maturidade nos processos de desenvolvimento de software (HALL, 2003).

Segundo Hall (2003), o Gerenciamento de riscos está subdivido em quatro atividades principais:

1. Identificação dos riscos: esta é uma das principais atividades do gerenciamento de riscos, pois visa identificar todas as ameaças do projeto e suas origens, para que estas possam ser tratadas futuramente;
2. Análise dos riscos: todos os riscos identificados na primeira etapa devem ser analisados visando classificá-los e ordená-los, assim, é possível dar prioridade ao tratamento dos riscos mais urgentes. Usualmente é usado o cálculo da exposição ao risco, que consiste em multiplicar a probabilidade do risco pelo impacto que este poderá causar ao projeto;
3. Planejamento de respostas aos riscos: para os riscos mais urgentes, deverão ser criados planos de contingência para diminuir suas probabilidades e impactos, contendo ações preventivas e corretivas;
4. Monitoramento e controle dos riscos: nesta etapa, os planos de resposta são aplicados e seus resultados são estudados. Também é feita uma análise periódica dos riscos identificados a fim de verificar a exposição ao risco. Novos riscos podem ser adicionados se identificados no decorrer do projeto e riscos existentes podem ser desconsiderados caso se tornem menos urgentes.

Sendo assim, a utilização de listas de riscos é importante no seu gerenciamento, pois facilitam a identificação de riscos que poderiam passar despercebidos aos olhos de um time sem muito conhecimento e experiência no assunto. Tendo uma lista para analisar e dizer se os riscos estão presentes ou não, facilita a aprendizagem, seu tratamento, e ao mesmo tempo em que aumenta a confiabilidade da identificação.

No desenvolvimento deste trabalho, foi levantado uma série de riscos que podem influenciar no desenvolvimento do processo de software do projeto. Para chegar a esta lista de riscos, foram seguidos os seguintes passos: a) Revisão bibliográfica sobre gerenciamento de riscos, com foco na identificação e no uso de *checklists*; b)Busca por artigos e trabalhos sobre

identificação e gerenciamento de riscos na biblioteca digital *IEEExplore*; c) Seleção dos artigos relevantes ao trabalho e que continham listas de riscos; d)Criação de uma lista única de riscos; e) Classificação dos riscos com base nas suas características e área de prevenção/minimização. Na Tabela 2 é apresentada uma listagem de doze principais riscos identificados na execução de projetos de software.

Tabela 2 – Checklist de Riscos relacionados a Requisitos

Riscos relacionados a Requisitos	
1º	Análise inadequada quando um novo requisito é adicionado
2º	Cliente e equipe têm visões diferentes de um mesmo requisito
3º	Escopo / objetivos pouco claros ou equivocados
4º	Falta de um acordo interativo entre cliente e os mantenedores relativos às especificações dos requisitos
5º	Grande Backlog
6º	Mudança constante nas funcionalidades do sistema
7º	Projeto de um tamanho/complexidade nunca feito antes
8º	Requisitos de baixa qualidade
9º	Requisitos incompletos do software
10º	Requisitos instáveis do software
11º	Requisitos mal entendidos (não refletem as expectativas do cliente)
12º	Requisitos não claros (ambíguos / imprecisos)

Fonte: Autor

Depois do processo de identificação dos possíveis riscos, há necessidade de analisar de forma quantitativa e qualitativa cada um deles. Na verdade, neste passo os gerentes de projeto priorizam a lista identificada, pois, na prática, não é possível fazer uma estratégia de controle para todos os riscos identificados. Nos processos de análise/avaliação, todos os riscos são analisados cuidadosamente pelos especialistas dando-se ênfase aos riscos que envolvem o projeto.

A partir disso, é possível elaborar um processo de software que gerencie os riscos priorizados de acordo com os planos de tratamento definidos no projeto. Segundo o PMBOK (2008), o plano de gerenciamento, engloba fatores importantes como a metodologia que será aplicada para o gerenciamento dos riscos, os papeis e responsabilidades de cada envolvido, além de orçamento, prazos, categoria, definições de probabilidade e impacto dos riscos, dentre outros. Segundo Vargas (2016) “o plano de riscos é um dos planos secundários do projeto”.

2.4. MODELO DE VARIABILIDADES

Desde sua primeira introdução em 1990, a modelagem de variabilidades ou *features models* tem sido a técnica mais comum para modelar a uniformidade e a variabilidade de linhas de produto (KANG e LEE, 2013).

Feature Models foram introduzidos pela primeira vez como parte do método *Feature Oriented Domain Analysis* (FODA), desenvolvido por Kang em 1990 (KANG e LEE, 2013). FODA é uma metodologia originada de um estudo de diferentes abordagens de análise de domínio. Ele dá uma visão dos requisitos e aspectos de arquitetura sobre os ativos a serem construídos.

O que é importante na modelagem da variabilidade é que: (i) existem diferentes segmentos de mercado ou comunidades de usuários que podem ter diferentes objetivos e/ou diferentes contextos de uso do produto; (ii) diferentes objetivos ou contextos de uso podem exigir atributos de qualidade e capacidades específicas; (iii) as mesmas capacidades podem ser implementadas de diferentes maneiras (decisões de projeto), que podem ter características de qualidade diferentes (KANG e LEE, 2013).

Feature Models é uma abordagem amplamente utilizada para a análise de domínio que captura semelhanças e variabilidades. Consiste em um diagrama de recursos e outras informações, como racionalidade, restrição e regra de dependência. Um diagrama de recursos é uma notação semelhante a uma árvore que mostra a estrutura hierárquica dos recursos. A raiz da árvore é referida como nó de conceito. Um ponto de variação é um local onde uma decisão pode ser tomada para determinar se nenhuma, uma ou mais características podem ser selecionadas para fazer parte do produto final.

Uma *feature model* representa a informação de todos os produtos possíveis de uma linha de produtos de software em termos de características e relações entre eles. Um modelo de recurso é representado como um conjunto hierarquicamente organizado composto por:

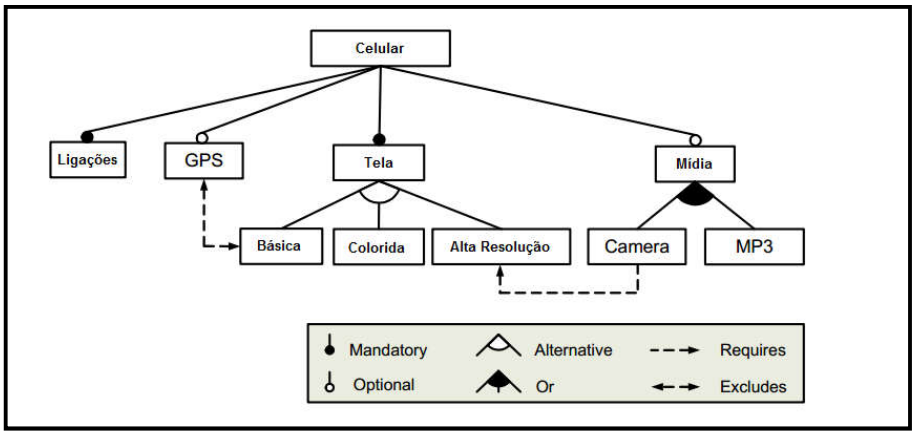
1. Relacionamentos entre um recurso pai (ou composto) e seus recursos filhos (ou subfunções);

2. Restrições entre árvores (ou hierarquias cruzadas) que normalmente são declarações de inclusão ou exclusão no formulário, que define se um recurso X, por exemplo, estiver incluído, os recursos A e B também devem ser incluídos (ou excluídos).

A Figura 1 ilustra um modelo simplificado da indústria telefônica. O modelo ilustra como os recursos são usados para especificar e criar software para celulares. O software

carregado no telefone é determinado pelos recursos que ele suporta. De acordo com a Figura 1, todos os telefones devem incluir suporte para chamadas, e exibir informações em uma tela com cores básicas, colorida ou de alta resolução. Além disso, o software pode incluir opcionalmente suporte para GPS (*Global Positioning System*) e dispositivos multimídia, como câmera, MP3, ou ambos. Caso exista o dispositivo multimídia de câmera, é requerido uma tela de alta resolução. Ainda, caso seja selecionada a opção de GPS (*Global Positioning System)*, é excluída a opção de tela básica, sendo somente possível a tela de definição colorida ou de alta resolução.

Figura 1 - Exemplo de *Feature Model*

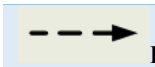
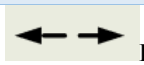


Fonte: BENAVIDES, *et al.*, 2010

A Figura 2 apresenta a formalização, em formato de lógica proposicional, das relações entre *feature models* apresentadas na Tabela 3. A lógica proposicional é a forma mais simples de lógica. Nela os fatos do mundo real são representados por sentenças sem argumentos, chamadas de proposições. Uma proposição é uma sentença, de qualquer natureza, que pode ser qualificada de verdadeiro ou falso. A lógica proposicional nos permite criar proposições que são representadas graficamente pelas imagens das árvores dos modelos de características.

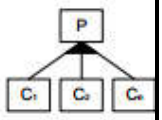
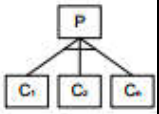
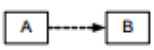
Na Tabela 3 são exibidas as relações para representação de *feature model*s.

Tabela 3 - Relações entre *Feature Models*

Relação	Descrição
 **Obrigatório**	Um recurso filho tem um relacionamento obrigatório com seu pai quando a criança é incluída em todos os produtos em que seu recurso pai aparece.
 **Opcional**	Um recurso filho tem um relacionamento opcional com seu pai quando o filho pode ser opcionalmente incluído em todos os produtos em que seu recurso pai aparece.
 **Alternativo**	Um conjunto de recursos filho tem um relacionamento alternativo com seu pai quando somente um recurso das crianças pode ser selecionado quando seu recurso pai é parte do produto.
 **Ou**	Um conjunto de recursos filho tem um ou outro relacionamento com seu pai, ou seja, quando um ou mais deles podem ser incluídos nos produtos em que seu recurso pai aparece.
 **Requerido**	Se uma característica A requer uma característica B, a inclusão de A em um produto implica a inclusão de B em tal produto.
 **Exclui**	Se um recurso A exclui um recurso B, ambos os recursos não podem fazer parte do mesmo produto.

Fonte: BENAVIDES, *et al.*, 2010.

Figura 2 - Exemplo de Mapeamento de uma *feature model* para lógica poposicional

Relação		Lógica Proposicional	Exemplo Celular
Obrigatório		$P \leftrightarrow C$	Celular ↔ Chamadas Celular ↔ Tela
Opcional		$C \rightarrow P$	GPS → Celular Multimídia → Celular
Ou		$P \leftrightarrow (C1 \quad C2 \quad ... \quad C_n)$	Media ↔ (Câmera MP3)
Alternativo		$C_1 \leftrightarrow (\neg C_2 \wedge ... \wedge \neg C_n \wedge P)) \wedge$ $C_2 \leftrightarrow (\neg C_1 \wedge ... \wedge \neg C_n \wedge P)) \wedge$ $C_n \leftrightarrow (\neg C_1 \wedge \neg C_2 \wedge ... \wedge \neg C_{n-1} \wedge P))$	Básica ↔ (¬Colorida ∧ ¬Alta Resolução ∧ Tela)) ∧ Colorida ↔ (¬Básica∧ ¬Alta Resolução ∧ Tela)) ∧ Alta Resolução ↔ (¬Básica ∧ ¬Colorida ∧ Tela))
Requer		$C \rightarrow P$	Câmera → Alta Resolução
Exclui		$\neg(A \wedge B)$	¬(GPS ∧ Básica)

Fonte: BENAVIDES, *et al.*, 2010

Feature Models são usados em diferentes cenários de produção de software que vão desde o desenvolvimento orientado por modelo, programação orientada a características, fábricas de software ou programação generativa, todos em torno do desenvolvimento de linha de produtos de software. Embora modelos de recurso são estudados em engenharia de linha de produto de software, esses modelos de informação podem ser usados em diferentes contextos como na definição de requisitos.

As empresas tendem a usar processos semelhantes para desenvolver diferentes tipos de projetos e/ou processos (por exemplo, desenvolvimento, manutenção, extensão, etc.). Dado um conjunto de processos semelhantes, podemos criar uma família de processos identificando os seus aspectos comuns destes processos, bem como a forma como estes variam de acordo com o tipo de projeto em desenvolvimento. Assim, uma linha de processo de software consiste em duas coisas: um modelo geral do processo, bem como uma especificação de quais elementos de processos podemos variar, e como. As definições de processo individuais são criadas por meio da remoção da variabilidade do modelo geral, permitindo a geração de definições de processos que podem ser especificamente adaptados a cada novo projeto (SIMMONDS, *et al.*, 2011).

2.5. MÉTODO DE DECISÃO MULTICRITÉRIO

O processo de tomada de decisão nas organizações está se tornando cada vez mais complexo. O objetivo é tomar as decisões corretas e consistentes e que estejam alinhadas com o planejamento estratégico. A partir do reconhecimento dessas necessidades geradas por problemas complexos, surgiram processos de decisão baseados em múltiplos critérios, nos quais se analisa critérios preferenciais, por exemplo: riscos, segurança, custo, tempo. A utilização destes é necessária para a compreensão da realidade do problema analisado e a escolha da alternativa que permitirá tomar a decisão mais acertada.

Nesta seção, é apresentado um método de apoio à tomada de decisões baseados em análise multicritérios. Em geral, esses métodos são utilizados para contribuir na decisão de escolha de alternativas e são amplamente utilizados em diversas áreas como: escolha de imóveis, determinação de portfólio de projetos, análises situacionais.

Problemas de tomada de decisão multicritério são geralmente caracterizados por um número finito de alternativas e por múltiplos critérios (atributos) muitas vezes conflitantes.

Technique for Order Preference by Similarity the Ideal Solution (TOPSIS) é uma técnica para avaliar o desempenho das alternativas verificando a similaridade de cada alternativa com a solução ideal. Essa técnica foi criada por Hwang e Yoon em 1981 (HWANG e YOON, 1981).

De acordo com essa técnica, a melhor alternativa é aquela que é a mais próxima da solução ideal composta de todos os melhores valores atingíveis dos critérios de benefício; já a solução ideal negativa consiste em todos os piores valores atingíveis dos critérios de sustentabilidade (PETRY e SILVA, *et. al*, 2014). A solução ideal positiva é uma solução que maximiza os critérios de benefício e minimiza os critérios de custo; já a solução ideal negativa maximiza os critérios de custo e minimiza os critérios de benefício. O seu conceito teórico, baseia-se no cálculo da distância euclidiana entre as alternativas mais benéficas e as alternativas de maior custo.

Os passos para a execução do método são (SILVA e NETTO, 2010):

Primeiro Passo: Montagem da matriz preliminar A = [a_{ij}], na qual há *j* alternativas (projetos), representadas por *a1, a2, a3, ... , a_j*. Para cada uma das alternativas, a medição do critério de ordem *i* é representada por a_{ij}, sendo *n* o número de critérios.

Segundo Passo: Realização do cálculo da matriz normalizada, que é realizada por meio da abordagem de cálculo de normalização por vetor. A equação 1, expressa sua aplicação.

$$r_{ij} = \frac{a_{ij}}{\sqrt{\sum_{i=1}^{j} a_{ij}^2}} \; i = 1, 2, 3 \ldots\ldots\ldots, n; \; j = 1 \ldots\ldots\ldots, j \qquad (1)$$

A matriz normalizada A_n corresponde à matriz A, com os valores de a_{ij} substituídos por r_{ij} ou seja, $A_n = [r_{ij}]$.

Terceiro Passo: O cálculo da matriz ponderada é realizado usando valores ponderados v_{ij}, obtidos na aplicação da equação 2, na qual, é multiplicado cada valor r_{ij} pelo peso (w_i) atribuído aos *n* critérios. O valor de w_i corresponde ao peso de cada critério.

$$v_{ij} = r_{ij} \times wi_i \quad i = 1, 2, 3 \ldots\ldots\ldots, n; \; j = 1 \ldots\ldots\ldots, j \qquad (2)$$

Quarto Passo: Nesta etapa ocorre a determinação das duas soluções: a solução ideal positiva (A^+) e solução ideal negativa (A), a primeira representada pelo valor máximo ponderado de cada critério e a segunda o valor mínimo, calculados por (3) e (4):

$$A^+ = \{v_i^+ \ldots v_i^+\} = \{(\max v_{ij} |\; i \in I'), (\min v_{ij} |\; i \in I'')\} \qquad (3)$$

$$A^- = \{v_i^- \ldots v_i^-\} = \{(\min v_{ij} |\; i \in I'), (\max v_{ij} |\; i \in I^{''})\} \qquad (4)$$

sendo *I'* associado ao critério de benefício e *I"* ao critério de custo.

Quinto Passo: neste passo é realizado o cálculo dos desvios pela distância euclidiana entre os valores ponderados para cada alternativa *j*, expresso nas equações 5 e 6, respectivamente.

$$_{j}^{+} = \sqrt{\sum_{i=1}^{n} (v_{ij} \quad v_i^{+})^2}, j = 1,2,3 \ldots \qquad (5)$$

$$_{j}^{-} = \sqrt{\sum_{i=1}^{n} (v_{ij} \quad v_i^{-})^2}, j = 1,2,3 \ldots \qquad (6)$$

Sexto Passo: Realização do cálculo para determinação do resultado da aproximação relativa de cada alternativa *j* às situações ideais positivas e negativas, representadas por φ. Finalmente, chega-se ao resultado da aproximação às situações positivas e negativas, com o emprego da equação 7:

$$\varphi = \frac{_{j}^{-}}{(_{j}^{+} + \quad _{j}^{-})} \quad j = 1,2,3 \ldots \qquad (7)$$

O maior valor de φ corresponderá à melhor alternativa dentro dos princípios de avaliação do TOPSIS. A escolha da melhor alternativa corresponderá ao valor de φ com maior coeficiente de prioridade.

Além dessa técnica existem outras técnicas de priorização baseadas em análises multicritérios como: AHP (SAATY, T. L,1980), TODIM (GOMES e MARANHÃO, 2008), Promethee (PROMETHEE, 2013), SMART (Simple Multi-Atribute Rating Technique) (STARFIELD, 2005), ELECTRE (Elimination and Choice Translating Reality) (BOTTI e PEYPOCH, 2013). Cada uma dessas técnicas possui teorias e critérios de avaliação das alternativas distintas, buscando obter o melhor resultado.

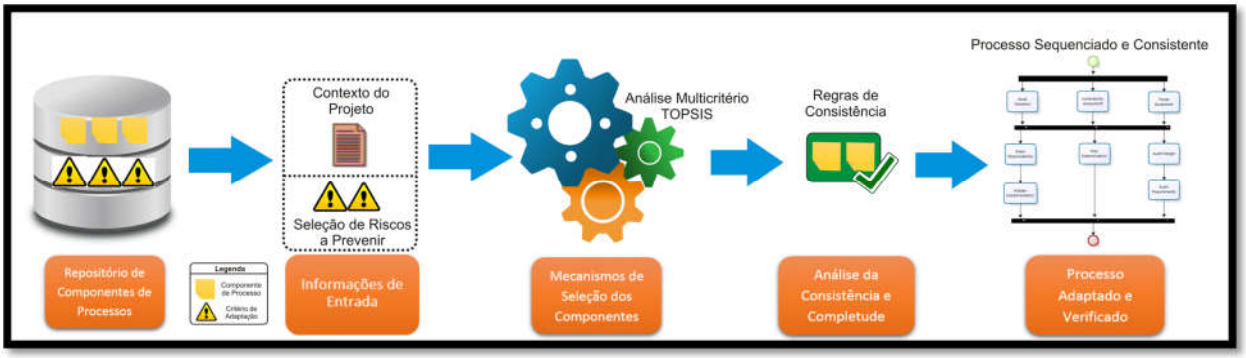
Neste trabalho, optou-se pelo uso da técnica de análise multicritério em relação as demais, pelos seguintes motivos: i) permite a adoção de uma quantidade não limitada de critérios para avaliar uma quantidade não limitada de alternativas; ii) a simplicidade dos procedimentos matemáticos contribuem para a facilidade de implementação e aplicação; iii) algoritmo permite não exigir uma sintonia de quaisquer parâmetros, iv) permite tratar de forma objetiva um problema de tomada de decisão, e v) consegue demonstrar qual a melhor e a pior alternativa em um determinado cenário.

3. DESENVOLVIMENTO

Este trabalho propõe uma abordagem para construção semiautomática de processos de software adaptados com foco na prevenção de riscos, que visa garantir que os processos gerados sejam completos e consistentes, influenciando diretamente na necessidade da redução da interferência de um engenheiro de processos. Restringiu-se a utilização dos elementos relacionados ao gerenciamento de requisitos, visto que, é identificada por diversos autores, uma grande importância na determinação de riscos relacionados a requisitos, devido a sua grande influência no destino de um projeto de software, vários casos de fracasso em projetos de software são associados a problemas relacionados a requisitos.

A abordagem proposta é composta por cinco passos, conforme apresentado na Figura 3, que são: i) construção e elaboração do repositório de componentes de processos; ii) identificação das informações de entrada; iii) desenvolvimento dos mecanismos de seleção de atividades; iv) aplicação e análise das regras de consistência e completude, e por fim, v) o processo adaptado e verificado.

Figura 3 - Abordagem para Adaptação de Processos



Fonte: Autor.

A primeira etapa na elaboração da abordagem para automação de processos de software consistentes foi a construção de um repositório de componentes de processos e critérios para adaptação de processos. Os critérios de adaptação foram baseados em riscos no desenvolvimento de processos de software. No segundo passo, tem-se a determinação das informações de entrada necessárias a adaptação de um processo de software, que são: o contexto que o projeto está inserido e os riscos a serem prevenidos. Na terceira etapa, foi

implementada uma análise multicritério, usando a técnica TOPSIS, na qual os critérios utilizados são: características do projeto e dos componentes, além da determinação de modelos de variabilidade para a seleção dos componentes que melhor se adaptam ao projeto. No quarto passo é realizada uma análise da consistência e completude do processo elaborado com a utilização de regras, e por fim, como resultado é montado o processo de software adaptado e verificado ao projeto. Cada uma das etapas, são descritas na seções que seguem deste capítulo.

3.1. ELABORAÇÃO DO REPOSITÓRIO

O primeiro passo para a construção da abordagem proposta, pode ser resumida no levantamento de informações para compor o repositório para a construção do processo de desenvolvimento de software. Sendo assim, inicialmente, realizou-se um levantamento de atividades relacionadas à engenharia de requisitos para os processos descritos na Seção 2.2.

A escolha dos autores e metodologias destacadas, se deve a grande utilização e procura dessas referências nos processos de desenvolvimento de software. Um modelo de processos como RUP (RUP, 2001) é comumente utilizado por empresas de desenvolvimento, devido ao seu alto grau de especificação das atividades e tarefas que devem ser realizadas para execução do processo. Já o CMMI (CMMI, 2013) é um modelo maturidade que sugere o que deve aplicado em processos de software, para guiar as organizações a conhecerem e melhorarem seus processos de software. Ao mesmo tempo, é quase impossível falar de engenharia de software e processos de software sem mencionar os autores Sommerville (SOMMERVILLE, 2007) e Pressmann (PRESSMAN e MAXIM, 2016), visto que a literatura desenvolvida e apresentada por ambos é frequentemente mencionada nos trabalhos que abordam os temas. E por fim, temos as ISO 12207 (ISO/IEC 12207, 2008) e o guia SWEBOK (BOURQUE, PIERRE; FAIRLEY, RICHARD E., 2014), que são referências regulamentadas internacionalmente para melhoria da qualidade do processo de software, sendo regulamentada pela ISO (*International Organization for Standardization*), IEC (*International Electrotechnical Commission*) e o SWEBOK, ambas desenvolvidas e documentadas pela IEEE *Computer Society.*

Além desses, o presente trabalho tem foco em atividades de engenharia de requisitos, visando suprir o desenvolvimento de tarefas relacionadas a requisitos em um processo de desenvolvimento de software.

O levantamento das atividades foi realizado a partir da análise de cada fonte da literatura descrita na Seção 2.2. Esse processo ocorreu da seguinte maneira: primeiramente foi realizada uma análise geral do conteúdo da fonte, objetivando verificar quais as informações disponíveis. Em seguida, iniciou-se o processo de extração das informações utilizando uma planilha para registro e posterior inserção no repositório. As informações extraídas são descritas no metamodelo utilizado para desenvolvimento da ferramenta de apoio deste trabalho. O metamodelo por ser visualizado na Figura 4 e foi elaborado a partir dos conceitos propostos por *Software Process Engineering Metamodel* (SPEM) 2.0 (OMG, 2008) e pelo RUP *version* 7.2 (RUP, 2001).

Segundo o metamodelo proposto, o ciclo de vida é composto de várias disciplinas (*Discipline*). Disciplinas são formadas por um conjunto de atividades (*Activity*) e associadas a um conjunto de papéis (*Role*). As atividades são formadas por tarefas (*Tasks*) que os papéis executam.

Artefatos (*Artifact*) são produtos de trabalho gerados durante a execução de tarefas associadas com o desenvolvimento do software. Para cada tarefa, ações são definidas (*Action*) representando o tipo de interação entre artefato e tarefa. As ações podem ser: *in, out* e *in/out*. Um tipo de ação "*in*" indica a leitura de um artefato, enquanto que uma ação do tipo "*out*" significa a criação de um artefato. Um tipo de ação "*in/out*" indica que ocorreram modificações no artefato. Neste trabalho definiu-se a utilização da denominação de Componente de Processo como sendo uma atividade englobando as informações de todos os elementos como tarefas, papeis, e artefatos de entrada e de saída.

A adaptação de processos considera a caracterização das atividades e requisitos de adaptação. Pode-se definir um ou mais contextos (*Context*) para cada atividade. Em tipo de contexto (*ContextType*) são definidos várias características associadas a uma atividade e a valores de cada uma (*ContextValue*).

Requisitos de adaptação (*TailoringCriteria*) definem os requisitos que podem ser satisfeitos pelo processo da instanciação de uma determinada atividade. Pode-se definir um ou mais tipos de requisitos de adaptação (*TailoringCriteriaType*) para cada atividade.

Para cada projeto (*Project*) existe uma organização associada (*Organization*) e um ou mais processos (*Process*). Cada projeto possui seu próprio contexto situacional associado.

Arquiteturas (*Architecture*) são formadas por um conjunto de atividades. Cada atividade é obrigatória ou opcional e concreta ou abstrata, sendo que cada atividade pertence a uma ou mais arquiteturas (*ArchitectureActivity*). A arquitetura de software de um programa ou

sistema computacional é a estrutura do sistema, que é composta de componentes de software, as propriedades visíveis externamente dos mesmos e a relação entre eles. Neste trabalho as arquiteturas foram utilizadas para inserção no repositório dos modelos de variabilidades que foram construídos e são apresentados na seção 3.3.

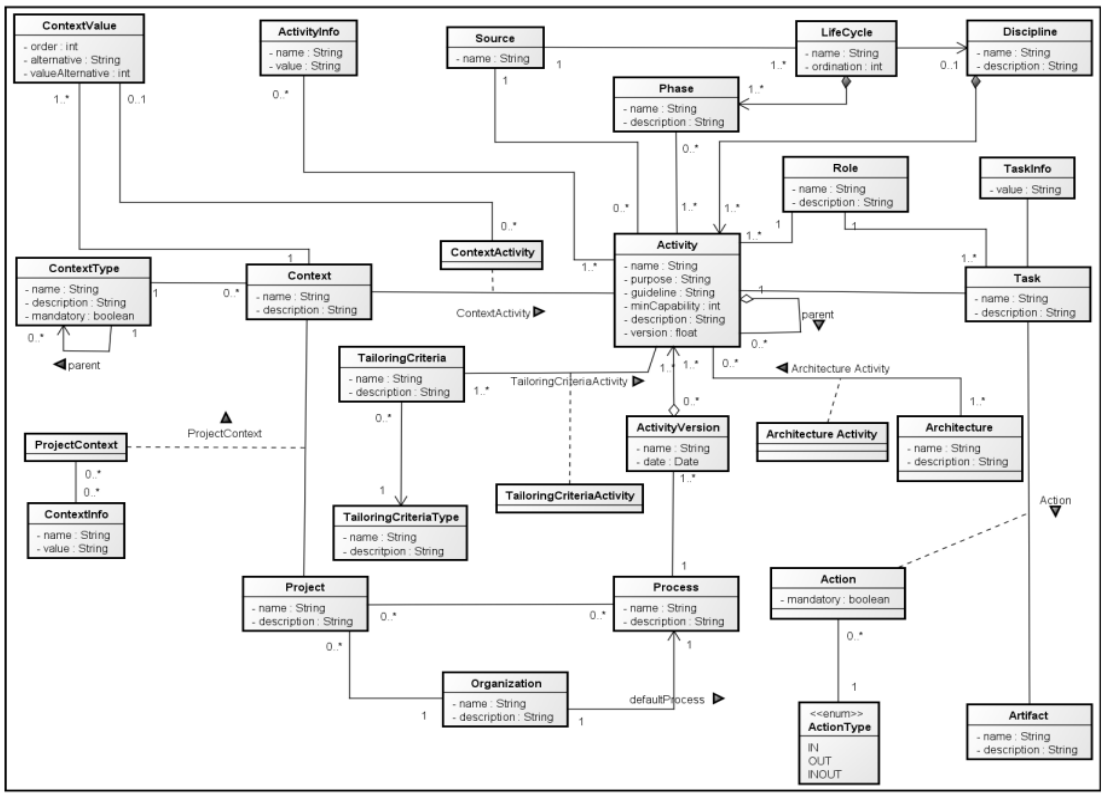
Trabalhou-se com 5 fases/etapas do processo de desenvolvimento: i) Elicitação e Levantamento de Requisitos, ii) Análise e Elaboração de Requisitos, iii) Especificação, iv) Validação e v) Gerenciamento. A utilização dessas etapas/fases, teve como referência as mesmas etapas/fases utilizadas pelo guia SWEBOK (BOURQUE, PIERRE; FAIRLEY, RICHARD E., 2014).

Os processos foram analisados visando identificar a similaridade dos elementos de processos, objetivando a inserção dos mesmos no repositório de componentes.

Na Tabela 4 é possível verificar um exemplo de levantamento de atividades da etapa de Elicitação e Levantamento de Requisitos, sendo que, na primeira coluna se destaca a etapa/ fase na qual a atividade está inserida, na segunda coluna o nome da atividade, na terceira e quarta, o responsável pela sua execução e a fonte de pesquisa na qual foi coletada, respectivamente. O restante das atividades coletadas e suas respectivas informações (fases, descrição, responsável e fonte) são apresentadas no Apêndice A deste trabalho.

Cabe salientar que, ao identificar atividades equivalentes, ou seja, atividades que possuem a mesma finalidade ou com o objetivo muito semelhante, porém, com técnicas ou fontes literárias distintas, ambas foram catalogadas e inseridas no repositório, pois o objetivo é realizar a construção de uma abordagem que considere diferentes características ou circunstâncias no qual o processo pode ser utilizado, o que pode levar a seleção de uma atividade de determinada metodologia ao invés de outra.

Figura 4 - Metamodelo ferramenta



Fonte: Autor.

Tabela 4 - Levantamento de atividade de Requisitos

Etapa/Fase	Atividades	Fonte
Elicitação e Levantamento de Requisitos	Levantar Necessidades	CMMI
	Descoberta de Requisitos	SOMMERVILLE
	Identificação de envolvidos	PRESSMAN
	Reconhecimento de diversos pontos de vista	
	Coleta colaborativa de requisitos	
	Cenários de uso	
	Artefatos do levantamento de requisitos	
	Levantamento de requisitos ágil	
	Desenvolvimento de casos de uso	
	Elicitar os Requisitos dos *Stakeholders*	RUP
	Encontrar atores e casos de uso.	
	Capturar um vocabulário comum	
	Identificar *Stakeholders*	ISO 12207-2008
	Identificar Requisitos	
	Registrar Requisitos	
	Elicitar Requisitos	SWEBOK

Fonte: Autor.

3.2. CARACTERÍSTICAS DE ENTRADA DO PROCESSO

Além da etapa de levantamento dos componentes de processo, foi realizada uma análise das características relevantes do projeto que irão contribuir para a seleção das atividades para o processo adaptado a ser desenvolvido. Essa etapa é de fundamental importância, pois irá permitir extrair a informação do cenário de uso do processo, ou seja, as características do projeto que se referem ao contexto no qual ele será implementado e/ou está inserido.

Para determinação das características do projeto, utilizou-se de perguntas-chave que posicionam e consolidam o contexto que o projeto está inserido. As perguntas são baseadas conforme apresentado na Subseção 2.1.1 destacadas na Tabela 1, que trata da contextualização dos processos, e foram baseadas em uma pesquisa com empresas da área de tecnologia de informação. Da mesma forma que o projeto possui um contexto, os componentes de processo também possuem uma caracterização baseada nas mesmas perguntas. As respostas dessas perguntas são valoradas e, posteriormente, é utilizado um mecanismo de análise multicritério para seleção dos componentes mais apropriados.

Além das características que englobam o contexto que o projeto está inserido, este trabalho acrescenta o conceito relativo a critérios de adaptação. Esses critérios de adaptação se referem aos riscos que podem vir a existir no desenvolvimento do processo de software a ser elaborado e precisam ser tratados para que sejam prevenidos ou minimizados.

Como critérios de adaptação, este trabalho utilizou o conceito de riscos da seguinte maneira: foi realizada uma pesquisa na literatura, extraindo e destacando doze principais riscos no gerenciamento de requisitos em processos de desenvolvimento de software. Os riscos são apresentados na Tabela 2, na Seção 2.3. Para prevenir e/ou minimizar a ação de riscos na elaboração do processo de software, foi realizada uma análise dos objetivos dos componentes de processo que pudessem prevenir ou mitigar a ação desses riscos no projeto, e a partir disso, foi possível realizar uma ligação entre o risco identificado para o projeto com o componente de processo que contribui para que o risco seja prevenido. A opção para seleção desta disciplina, justifica-se pelo grande número de projetos de desenvolvimento de software que falham por não analisar e prever tratamento adequado aos riscos inerentes a eles, e além disso, em toda a vida do projeto poderão surgir novos riscos, os já identificados podem se modificar e outros poderão deixar de existir.

Na Tabela 5 é exemplificada uma associação de um risco (Cliente e equipe têm visões diferentes de um mesmo requisito) com componentes de processo (Identificar os envolvidos, Reconhecer os diversos pontos de Vista, Desenvolver Visão, entre outras) que contribuem e/ou objetivam prevenir ou minimizar a ação desse risco em um processo de software.

Sendo assim, com base nos riscos destacados, a abordagem objetiva selecionar componentes de processo que possam satisfazer as necessidades específicas dos riscos a serem prevenidos/minimizados, e inseri-los dentro do processo adaptado, não deixando de considerar o contexto que o processo será aplicado. O método de seleção considerando critérios de adaptação aliado ao contexto do projeto e do componente de processo é apresentado na Seção 3.3.

Tabela 5 - Associação Risco e Componente de Processo

Risco do Projeto	**Componente de Processo**
Cliente e equipe têm visões diferentes de um mesmo requisito	- Identificar os envolvidos
	- Reconhecer os diversos pontos de Vista
	- Desenvolver Visão
	- Especificação dos Requisitos de Software
	- Especificação dos Requisitos de Software
	- Identificar Atores e Casos de Uso
	- Estruturar Modelos de Casos de Uso
	- Avaliação dos Requisitos

Fonte: Autor.

3.3. PRIORIZAÇÃO DOS COMPONENTES DE PROCESSOS

Nesta etapa é definido o método para seleção dos componentes de processo mais adequados, ou seja, é determinado o mecanismo que irá analisar as informações prévias do contexto do projeto e dos componentes que satisfazem os riscos a serem prevenidos ou mitigados, com o objetivo final de contribuir para a seleção do melhores componentes que irão compor o processo adaptado.

A priorização dos componentes se dá da seguinte maneira: após a seleção dos critérios de adaptação são resgatados os componentes de processo que previnem estes critérios de adaptação. Com base nestes componentes é realizada uma análise multicritério do contexto, comparando-o com o contexto em que o projeto se insere. A técnica de análise multicritério utilizada denomina-se TOPSIS - *Technique for Order Preference by Similarity the Ideal*

Solution. Essa técnica utiliza a teoria da distância euclidiana entre a melhor solução positiva e a pior solução negativa, retornando um resultado que determina que quanto mais próximo do valor 1, melhor o componente se encaixa ao critério de risco adotado. Os valores extraídos para os cálculos são baseados nas perguntas do contexto do projeto e do componente que o dois estão inseridos, ou seja, as respostas das perguntas possuem valores de um a cinco. No momento da análise multicritério, a ferramenta busca esses valores e repassa ao algoritmo de análise multicritério que realiza os cálculos conforme a teoria apresentada.

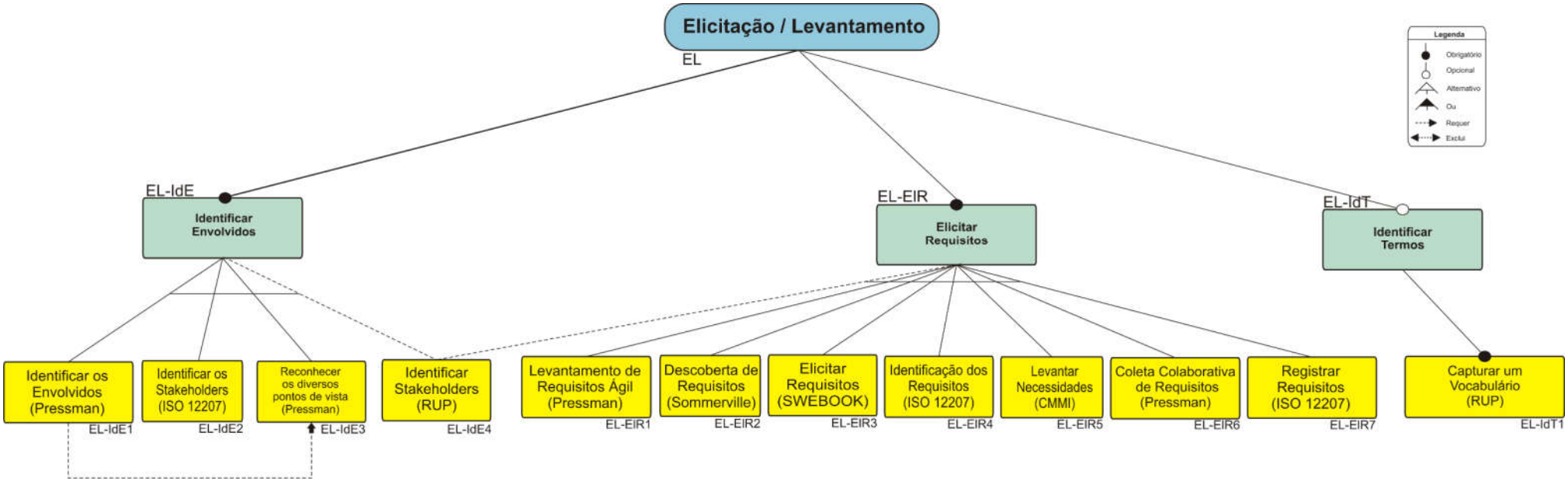
Porém, somente o uso dessa técnica não considera alguns fatores importantes, como obrigatoriedade de atividades, atividades semelhantes, dependências entre atividades, entre outras. Sendo assim, para resolver esse problema, foi implementado um mecanismo complementar, denominado modelos de variabilidades, baseado nos conceitos descritos na Seção 2.4. No metamodelo, os modelos de variabilidades foram introduzidos com o conceito de arquiteturas de processo.

Para o desenvolvimento dos modelos de variabilidades, inicialmente foi realizado o trabalho de agrupamento das atividades conforme seus objetivos. Neste agrupamento, foram criadas atividades abstratas e concretas, que foram sendo distribuídas em um formato de árvore. Para que fosse possível uma melhor visualização, utilizou-se de conceitos de modelos de variabilidades (*feature models)* conforme apresentados na Seção 2.4. Optou-se por essa abordagem devido a possibilidade de se definir as características e os relacionamentos entre os componentes, além de permitir a inclusão de regras de composição. As regras de composição referem-se a restrições adicionais para restringir combinações de componentes, além de permitir a criação de um formalismo para verificação e validação do modelo.

A Figura 5 apresenta um modelo de variabilidade (*feature model*) elaborado para a fase de "Elicitação e Levantamento de Requisitos". É possível verificar que o nó central da árvore contém a informação da etapa e/ou fase que esta se refere. Conforme o exemplo, a fase expressa se refere a "Elicitação/Levantamento". Logo abaixo, em um nível inferior (destacado em verde), são apresentados os componentes abstratos que foram criados visando agrupar os componentes de processo concretos, apresentadas na base da árvore (destacadas em amarelo), ou seja, com base na análise da descrição e dos objetivos dos componentes, estes foram agrupados e classificadas em um modelo de árvore. O uso do modelo de variabilidades permite criar algumas condições que devem ser consideradas na elaboração de um processo de software adaptado, por exemplo, a obrigatoriedade ou opcionalidade de componentes,

dependências e exclusão do componentes. Essas condições contribuem na garantia da consistência e completude do processo.

Figura 5 - Feature Model Etapa Elicitação / Levantamento de Requisitos



Para facilitar a compreensão, cada fase (destacada em azul), atividade abstrata (destacada em verde) e atividades concretas (destacadas em amarelo), foram nomeadas por abreviaturas usando o código que está descrito junto ao nome da fase e atividade. Esse critério foi utilizado para facilitar o desenvolvimento da regra do modelo de variabilidades.

3.4. ANÁLISE DA CONSISTÊNCIA E COMPLETUDE

A etapa quatro, leva em consideração a análise da consistência e completude do processo. Para realizar essa análise foram construídas as regras de composição baseados nos modelos de variabilidades. Sendo assim, para a formalização do modelo de variabilidade (*feature model*) apresentada na Figura 5, foram elaboradas regras de composição dos componentes para cada fase do modelo.

Para o exemplo apresentado na Figura 5, desenvolveu-se a seguinte regra:

Elicitação e Levantamento ↔ ((EL-IdE1 → EL-IdE3 ↔ (¬ EL-IdE2 ∧ ¬ EL-IdE4 ∧ EL-IdE) ∧
((EL-IdE2 ↔ (¬EL-IdE1 ∧ ¬ EL-IdE3 ∧ ¬ EL-IdE4 ∧ EL-IdE) ∧
((EL-IdE3 ↔ (¬ EL-IdE2 ∧ ¬ EL-IdE1 ∧ ¬ EL-IdE4 ∧ EL-IdE) ∧
((EL-IdE4 ↔ (¬ EL-IdE2 ∧ ¬ EL-IdE3 ∧ ¬ EL-IdE1 ∧ EL-IdE))
Elicitação e Levantamento ↔ ((EL-ElR1 ↔ (¬EL-ElR2 ∧ ¬ EL-ElR3 ∧ ¬ EL-ElR4 ∧ ¬ EL-ElR5 ∧ ¬ EL-ElR6 ∧ ¬ EL-ElR7 ∧ EL-ElR)) ∧
((EL-ElR2 ↔ (¬EL-ElR1 ∧ ¬ EL-ElR3 ∧ ¬ EL-ElR4 ∧ ¬ EL-ElR5 ∧ ¬ EL-ElR6 ∧ ¬ EL-ElR7 ∧ EL-ElR)) ∧
((EL-ElR3 ↔ (¬ EL-ElR2 ∧ ¬ EL-ElR1 ∧ ¬ EL-ElR4 ∧ ¬ EL-ElR5 ∧ ¬ EL-ElR6 ∧ ¬ EL-ElR7 ∧ EL-ElR)) ∧
((EL-ElR4 ↔ (¬ EL-ElR2 ∧ ¬ EL-ElR3 ∧ ¬ EL-ElR1 ∧ ¬ EL-ElR5 ∧ ¬ EL-ElR6 ∧ ¬ EL-ElR7 ∧ EL-ElR)) ∧
((EL-ElR5 ↔ (¬ EL-ElR2 ∧ ¬ EL-ElR3 ∧ ¬ EL-ElR4 ∧ ¬ EL-ElR1 ∧ ¬ EL-ElR6 ∧ ¬ EL-ElR7 ∧ EL-ElR)) ∧
((EL-ElR6 ↔ (¬ EL-ElR2 ∧ ¬ EL-ElR3 ∧ ¬ EL-ElR4 ∧ ¬ EL-ElR5 ∧ ¬ EL-ElR1 ∧ ¬ EL-ElR7 ∧ EL-ElR)) ∧
((EL-ElR7 ↔ (¬ EL-ElR2 ∧ ¬ EL-ElR3 ∧ ¬ EL-ElR4 ∧ ¬ EL-ElR5 ∧ ¬ EL-ElR6 ∧ ¬ EL-ElR1 ∧ EL-ElR)))
Elicitação e Levantamento → EL-IdT → EL-IdT1

A compreensão das regras deve ser analisada da seguinte maneira: na primeira "ramificação" da árvore a atividade abstrata "Identificar Envolvidos", é uma atividade

obrigatória do processo (pois é destacada com um círculo preenchido sobre ela), na qual os componentes concretos “Identificar Envolvidos (Pressman)”, “Identificar Stakeholders (ISO 12207)”, “Reconhecer os diversos Pontos de Vista (Pressman)” e “Identificar Stakeholders (RUP)” cumprem os objetivos da mesma, ou seja, qualquer um dos componentes concretos destacados abaixo da mesma podem ser selecionados, e todos irão satisfazer o objetivo do componente abstrato. Porém, é destacado ainda que a regra que seleciona o componente “Identificar Envolvidos (Pressman)” requer-se que seja incluída no processo o componente “Reconhecer os diversos Pontos de Vista (Pressman)”. Essa é a interpretação que deve ser realizada na apresentação.

Para cada etapa de desenvolvimento de software foram desenvolvidos modelos de variabilidades que implementam as condições para que os componentes abstratos sejam satisfeitos. Os demais modelos correspondentes a outras fases e suas respectivas regras são apresentados no Apêndice C deste trabalho.

Com base nas regras criadas a partir dos modelos de variabilidades, é possível analisar se as atividades selecionadas garantem a consistência e a completude do processo. As regras foram criadas com base nos conhecimentos adquiridos, levando em consideração os objetivos das atividades e o conhecimento de especialistas das áreas de adaptação de processos.

É possível observar que é comum entre os autores da área de pesquisa sobre definição de processos, o uso de regras para checagem da consistência de um processo de software. (PEREIRA, 2011). Alguns autores enfatizam também uma etapa específica, durante a definição dos processos de software, para checagem da consistência desses processos (ATKINSON & WEEKSJOHNNOLL, 2007). Atkinson e & WeeksJohnNoll (2007), por exemplo, destacam a importância da checagem de um processo de software antes de sua execução devido à habilidade que esse mecanismo fornece para detectar e eliminar inconsistências em um processo antes que elas se tornem erros e comprometam o sucesso de um projeto de software.

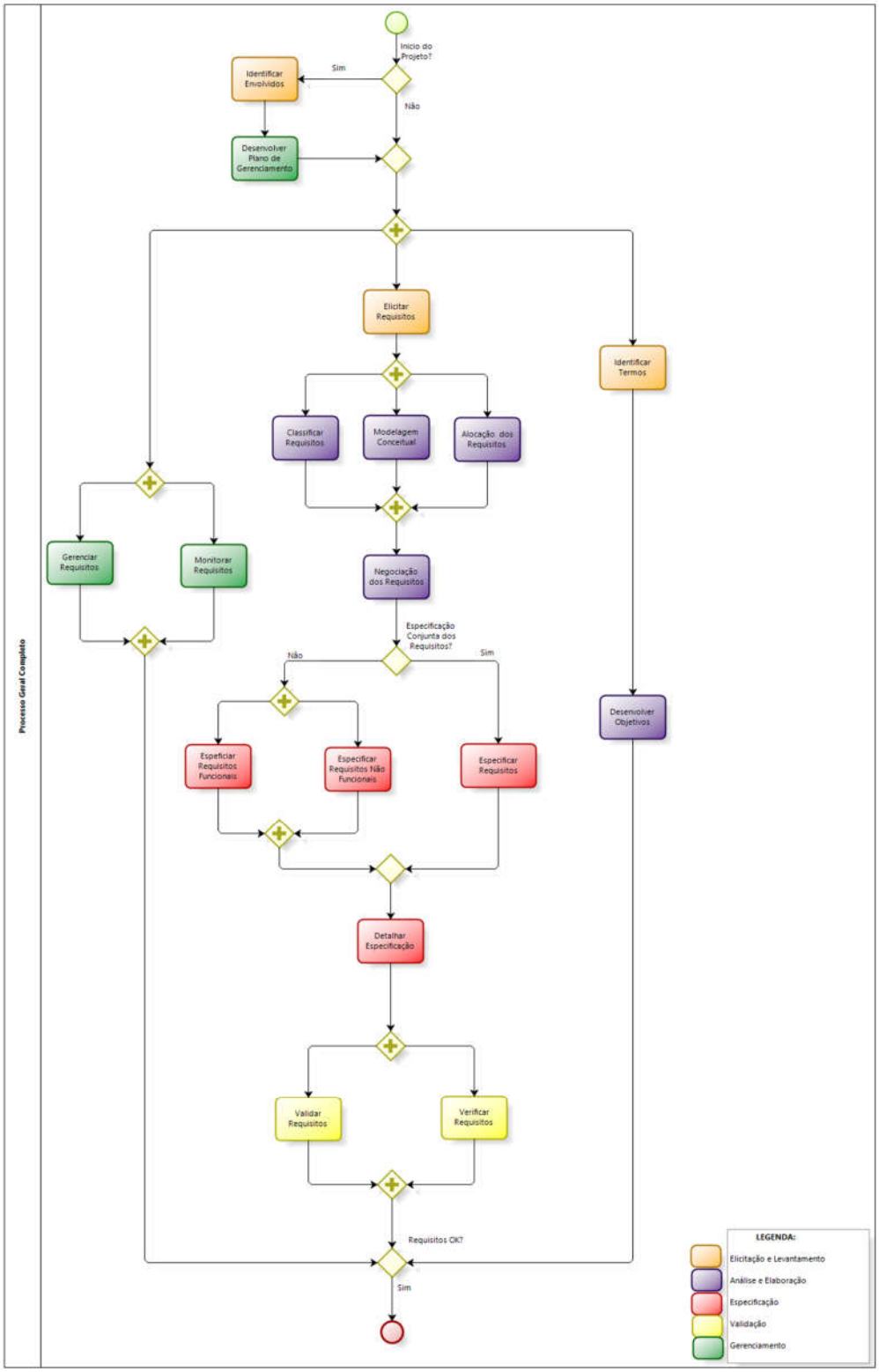
3.5. RESULTADO DO PROCESSO DE ADAPTAÇÃO

A quinta etapa do desenvolvimento deste trabalho, foi a elaboração do processo sequenciando de cada uma das etapas de desenvolvimento de software e seus componentes abstratos. A partir do sequenciamento dos componentes é possível verificar o processo como um todo. Na Figura 6 é apresentada a linha de processo contendo todas as fases e os

componentes de um processo de desenvolvimento de software referente à disciplina de requisitos.

O sequenciamento foi elaborado conforme análise realizada dos objetivos de cada componente. É possível visualizar na Figura 6, as cinco fases que englobam o processo de software (Elicitação e Levantamento, Análise e Elaboração, Especificação, Validação e Gerenciamento) e o fluxo de execução das atividades abstratas relacionadas ao gerenciamento de requisitos. A notação utilizada para a construção dos processos de software é a BPMN - *Business Process Model and Notation.* Essa notação especifica o processo de negócio em um diagrama que é fácil de ler tanto para os usuários técnicos quanto para os usuários de negócios, permitindo a representação de detalhes complexos do processo.

Figura 6 – Linha de Processos da Disciplina de Requisitos



Fonte: Autor.

4. FERRAMENTA DE APOIO *OPTIMIZED PROCESS TOOL*

Neste capítulo é apresentada a ferramenta de apoio *Optimized Process Tool* utilizada no desenvolvimento deste trabalho, destacando as funcionalidades que implementam os conceitos na abordagem proposta.

A ferramenta *Optimized* foi elaborada considerando o metamodelo descrito na Seção 3.1. O metamodelo permite a instanciação de componentes do processo para definição de processos planejados, bem como na customização de processos para o uso em projetos específicos. Os componentes de processos instanciados são armazenados em um repositório e combinados para produzir diferentes modelos de processos de software.

A ferramenta está estruturada em módulos, que são: o módulo principal e o módulo para adaptação de processos de software. No módulo de adaptação de processos de software é apresentado a abordagem desenvolvida no presente trabalho denominada *Automation Process* (*AutProcess*).

O módulo principal permite o acesso às funcionalidades necessárias para instanciação e inserção dos elementos utilizados nos processos de adaptação de processos de software. Podem-se registrar artefatos, tarefas, papéis, atividades, requisitos de adaptação, tipos de contextualização e seus valores, contextualização das atividades, contextualização do projeto e a definição de arquiteturas para processos. Na Figura 7 é possível visualizar o módulo principal da ferramenta, contendo todas as opções de ação (lado esquerdo da figura). As funcionalidades da ferramenta são subdivididas em categorias, que são: *Getting Started, Organization, Knowledge Base, Contextualization, Tailoring Criteria, Architecture e Settings*. A seguir uma breve descrição de cada uma delas:

Getting Started: nesta opção o usuário pode selecionar uma das abordagens para adaptação de processo de software contidas na ferramenta. Neste trabalho utilizou-se a abordagem *AutProcess* para adaptação de processos.

Organization: descreve as funcionalidades relacionadas à organização, tais como: cadastrar nova organização, criar um processo e cadastrar projeto, e gerenciar versionamento dos processos cadastrados. Na opção *Organization*, o usuário pode realizar o cadastro da organização com seu nome e descrição. Em *Project*, o usuário pode fazer o cadastramento do projeto, atribuindo uma organização para o mesmo e efetuando a caracterização do projeto. Em *Process*, pode-se visualizar os processos que foram criados e estão relacionados com uma determinada organização e projeto.

Figura 7 – Módulo Principal *Optimized Process Tool*

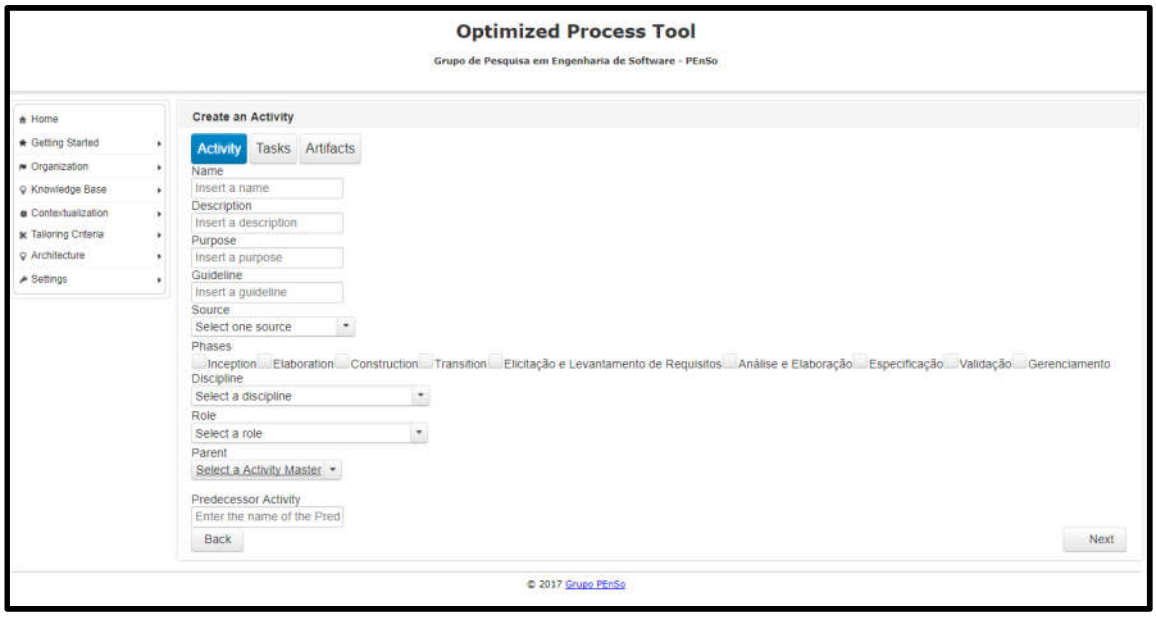
Fonte: Autor.

Knowledge Base: armazena o conhecimento organizacional utilizado na adaptação dos processos. Descreve as funcionalidades envolvendo a definição das *Action, Activity, Artifact, Discipline, Life Cycle, Phase, Role, Source e Task.*

Na opção ***Activity***[1], o usuário pode cadastrar as atividades e seus atributos como: nome, descrição, propósito, orientações, origem, fases, disciplina e papel (Figura 8). A opção "*Parent*" é utilizada para definir se a atividade a qual está sendo cadastrada é proveniente de outra atividade cadastrada.

[1]O conceito de componente de processo, conforme descrito neste trabalho, é chamado de atividade na ferramenta. Isto ocorre porque a ferramenta é genérica para diferentes abordagens de adaptação e usa o conceito de atividade definido pelo RUP.

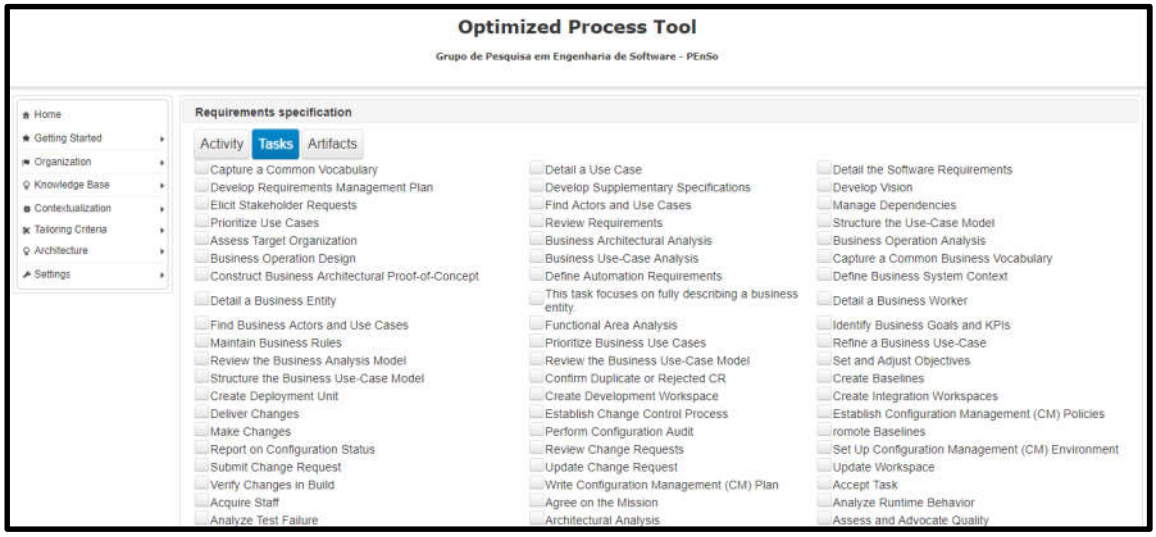
Figura 8 – Passos para criar uma atividade



Fonte: Autor.

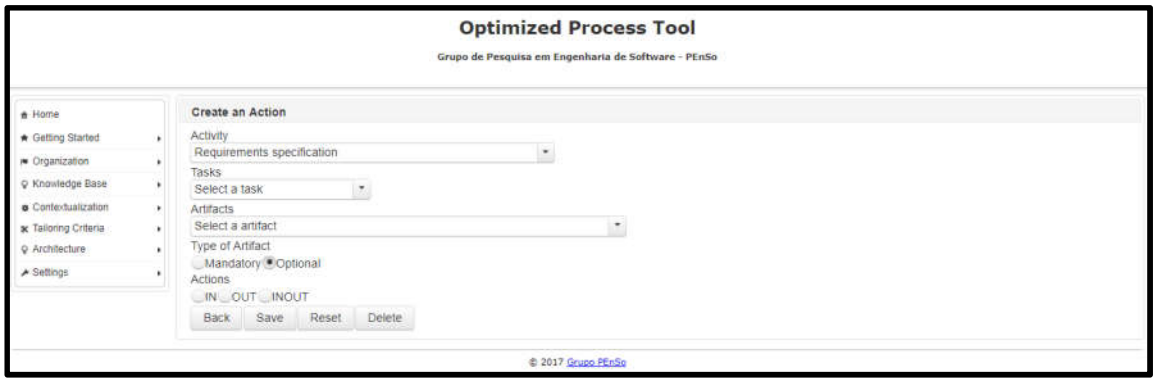
O segundo passo para o cadastramento de uma atividade é a escolha das tarefas (*Tasks*) que compõem essa atividade (Figura 9). O terceiro passo é a escolha dos artefatos que compreendem as atividades selecionadas. Na tela da Figura 9 é ilustrada a seleção das tarefas do processo, e na Figura 10 é ilustrada a opção para associar os artefatos às tarefas.

Figura 9 – Segundo passo: selecionar as tarefas



Fonte: Autor.

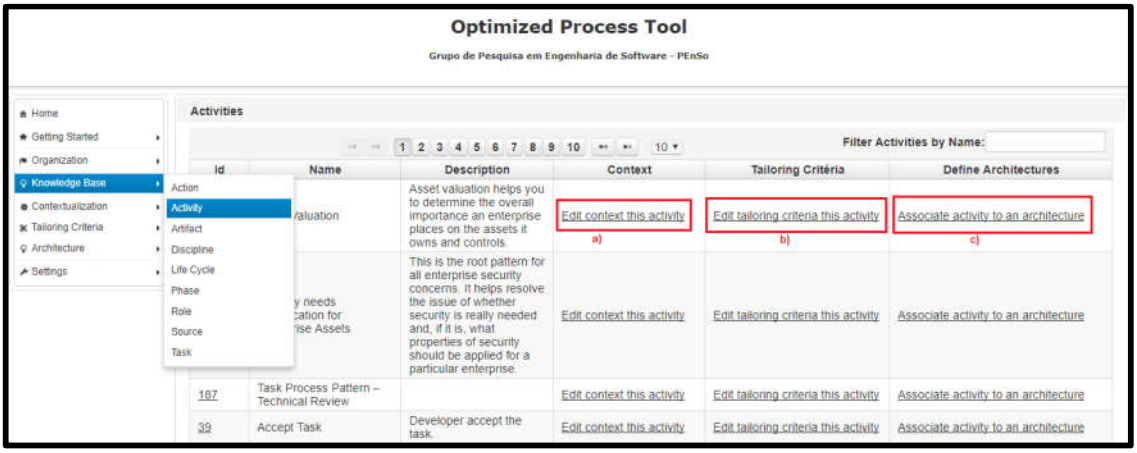
Figura 10 – Terceiro Passo: Determinação dos artefatos de Entrada e Saída



Fonte: Autor.

Finalizadas as etapas descritas, associa-se as atividades cadastradas a um ou mais contextos e a um ou mais requisitos de adaptação, e definindo qual arquitetura a atividade pertence, sendo que a atividade pode pertencer a mais de uma arquitetura. A Figura 11 ilustra cada uma das opções descritas ("*Edit context this activity*", "*Edit tailoring criteria this activity*" e "*Associate activity to an architecture*").

Figura 11 – Determinação das características da atividade para adaptação



Fonte: Autor.

Na opção "*Edit context this activity*" (destacado por 'a'), define-se um ou mais contextos para cada atividade, por exemplo, seleciona-se a contextualização utilizando um modelo pré-definido. Neste trabalho o modelo utilizado é o *Contextualization*, definindo valores para cada um dos seus doze fatores para a atividade em questão.

No momento do cadastro de um novo componente de processo, o mesmo já pode ser associado a um ou mais critérios de adaptação que irá satisfazer. Na opção "*Edit tailoring criteria this activity*" (destacado por 'b') define-se um ou mais requisitos de adaptação que a atividade visa satisfazer, por exemplo, a atividade "*Identify those involved*" visa garantir a satisfação do critério destacado por "Cliente e equipe têm visões diferentes de um mesmo requisito" (*Client and team have different views of the same requirement*). Na opção "*Associate activity to an architecture"* (destacado por 'c') é associado a atividade a uma arquitetura. Nessa opção são também definidos os atributos da atividade considerando a arquitetura na qual ela está associada. Esses atributos indicam se a atividade é obrigatória ou opcional e concreta ou abstrata.

Contextualization: descreve as funcionalidades relacionadas a caracterização do projeto, como tipos de caracterização e seus valores relacionados. O submenu *Organization* contém as opções *Context, Context Value e Context Type*. Em *Context Type*, o usuário pode realizar o cadastro dos tipos de contexto, por exemplo, *Octopus Model* ou Boehm, Turner ou uma caracterização conforme definido por um especialista.

Tailoring Criteria: descreve as funcionalidades relacionadas aos tipos de requisitos de adaptação. O submenu *Tailoring Criteria* contém as opções "Critério (*Criteria)"* e "Tipo de Critério (*Criteria Type)"*.

Na opção Tipo de Critério, podem-se cadastrar diferentes tipos de requisitos de adaptação, por exemplo, riscos, requisitos de segurança, qualidade, entre outros. Critério define as diferentes ocorrências para cada tipo de requisitos de adaptação, por exemplo, para riscos tem-se "Escopo / objetivos pouco claros ou equivocados", "Análise inadequada quando um novo requisito é adicionado", entre outros.

Architecture: neste trabalho descrevem-se os diferentes modelos de variabilidades definidos com seus componentes obrigatórios ou opcionais e concretos ou abstratos. Na opção *Architecture*, o usuário tem a listagem de todas as arquiteturas cadastradas, e a opção de cadastramento de uma nova arquitetura. Na opção "Definir (*Define)"*, tem-se a definição das atividades que irão compor a arquitetura selecionada.

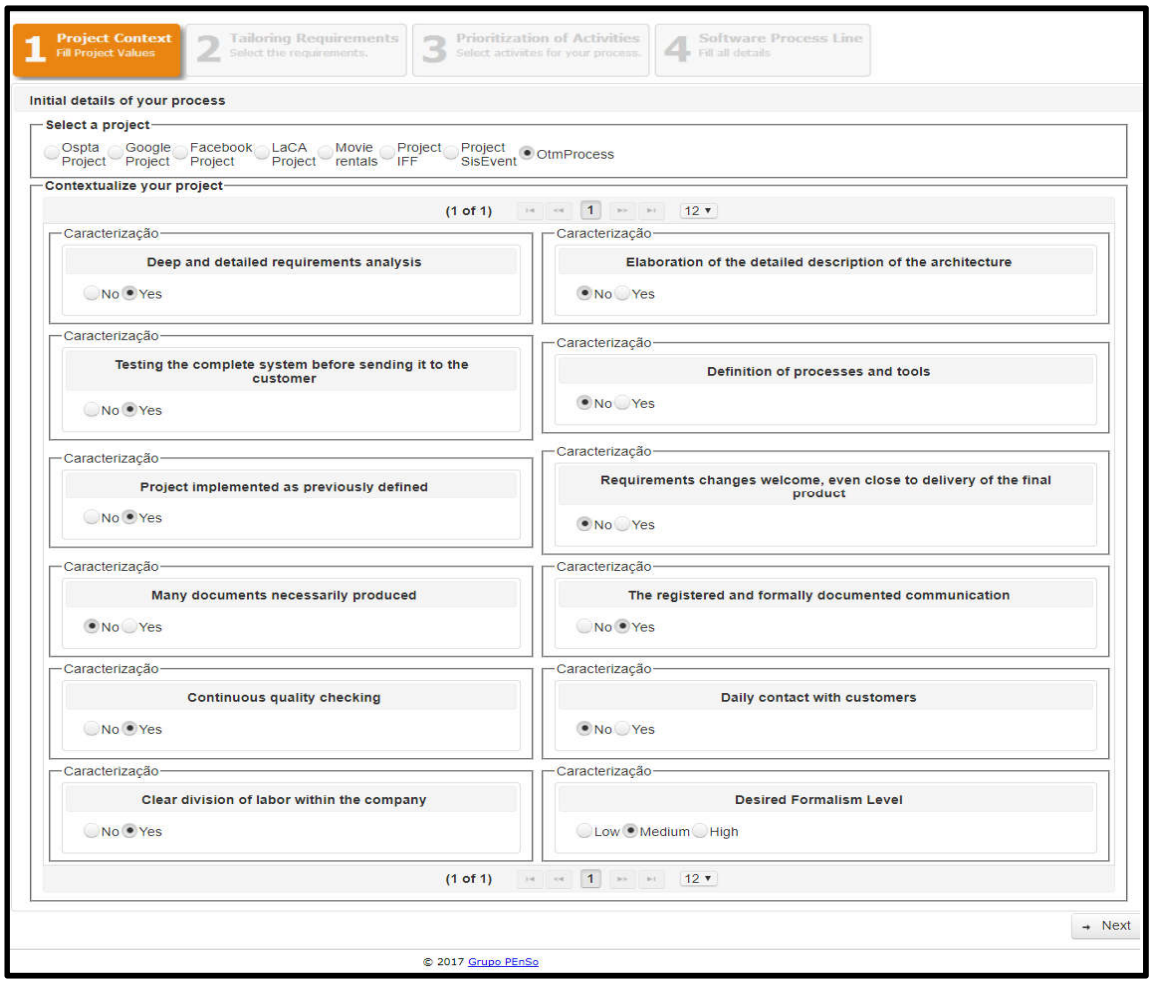
Settings: descreve as funcionalidades relacionadas ao cadastramento dos usuários e suas autorizações.

Como é possível verificar, a ferramenta de apoio apresenta um conjunto de elementos e características que possibilitam a elaboração de um processo de software. Após a demonstração dos recursos e funcionalidades gerais da ferramenta, é apresentado o módulo que apoia o desenvolvimento da abordagem apresentada nos objetivos deste trabalho.

O módulo para adaptação de processos (*AutProcess*) é organizado de acordo com o conjunto de etapas, que são: i) definição das características do projeto; ii) seleção dos requisitos de adaptação e da arquitetura de processos; iii) priorização das atividades e análise da consistência e completude; e v) criação da linha de processos de software adaptada. A seguir, uma breve descrição das etapas para elaboração do processo de software:

- **Definição das características do projeto (*Definitions of project context*):** a primeira etapa da abordagem de adaptação, se refere a contextualização do projeto, ou seja, o levantamento das características que o processo se insere. Nesta etapa, são destacadas os doze requisitos que caracterizam o projeto. Os requisitos são valorados e formam a contextualização do projeto, que posteriormente é utilizado na análise multicritério para destacar a melhor atividade para o contexto do projeto. A Figura 12 apresenta o formulário usado na ferramenta para a determinação das características de um projeto.

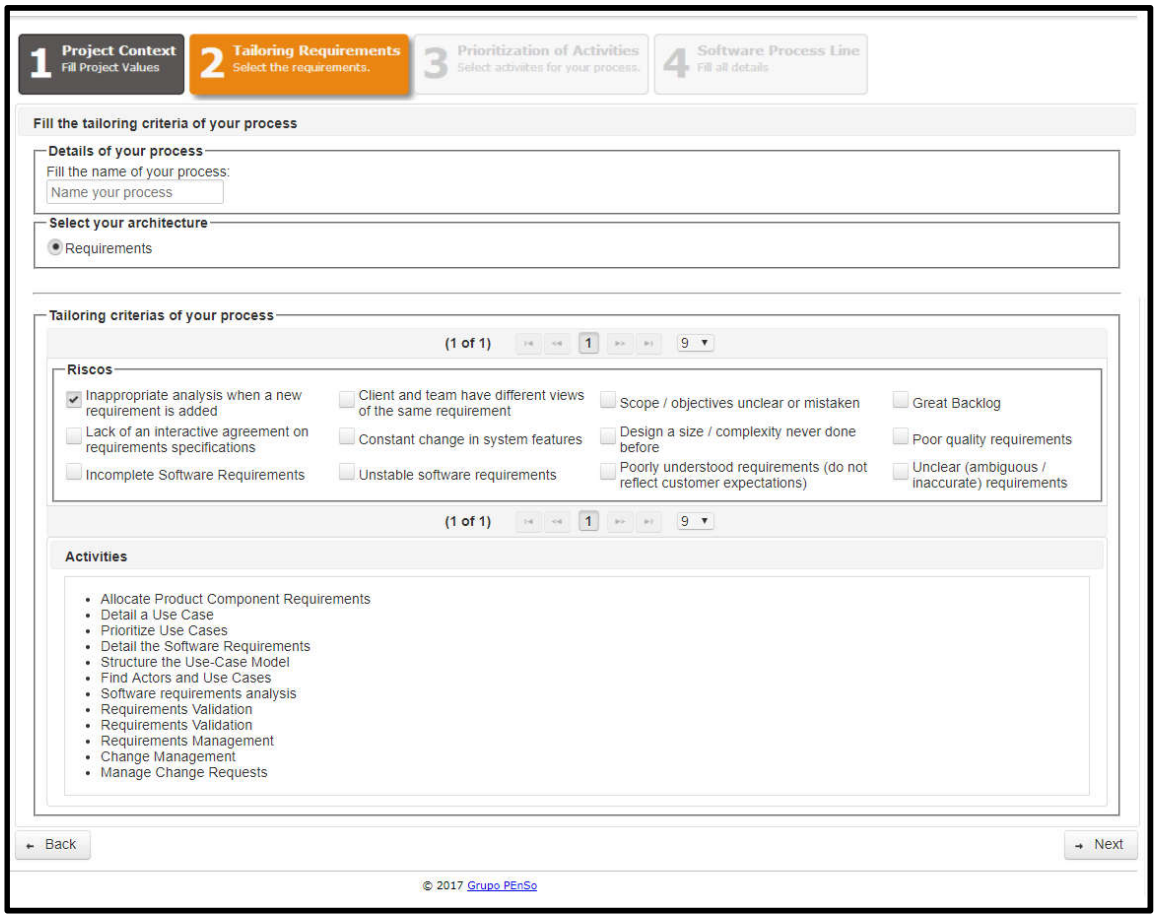
Figura 12 - Caracterização do Projeto



Fonte: Autor.

- **Seleção dos requisitos de adaptação (*Selection of the tailoring requirements*):** a segunda etapa, se refere a seleção dos riscos que devem ser prevenidos e/ou evitados no decorrer da elaboração do processo. Os riscos destacados foram apresentados na Seção 2.3 deste trabalho. Ao selecionar determinado risco, a ferramenta apresenta os componentes que visam satisfazer este risco logo abaixo. Podem ser selecionados diversos riscos para tratamento dentro do processo adaptado a ser desenvolvido. A Figura 13 apresenta os recursos da etapa dois.

Figura 13 - Seleção dos Critérios de Adaptação

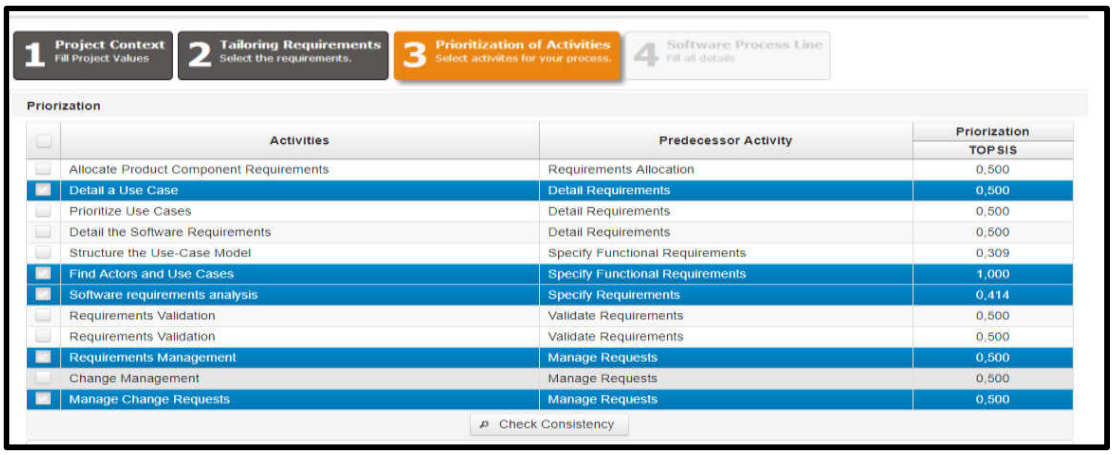


Fonte: Autor.

- **Seleção e priorização das atividades selecionadas (*Priorization of Activities*):** a terceira etapa, destaca as melhores atividades de acordo com o contexto definido para o projeto. Ou seja, ao realizar essa etapa a abordagem desenvolvida na ferramenta apresenta as melhores atividades de acordo com o cálculo de análise multicritério denominada TOPSIS, apresentada na Seção 2.5. Os componentes são selecionados automaticamente com base nos seus contextos e os critérios de adaptação que devem ser satisfeitos levando em consideração os valores retornados pela análise multicritério. Para a realização dos cálculos do algoritmo, os valores são extraídos das perguntas-chave do contexto do projeto e do componente, na qual os valores são determinados em uma escala de um a cinco. Posteriormente esses valores são repassados ao algoritmo de análise multicritério. Nesta etapa, o engenheiro de

processo pode incluir ou remover atividades do processo a ser elaborado. A Figura 14 apresenta o terceiro passo para a seleção dos componentes de software mais apropriados ao processo.

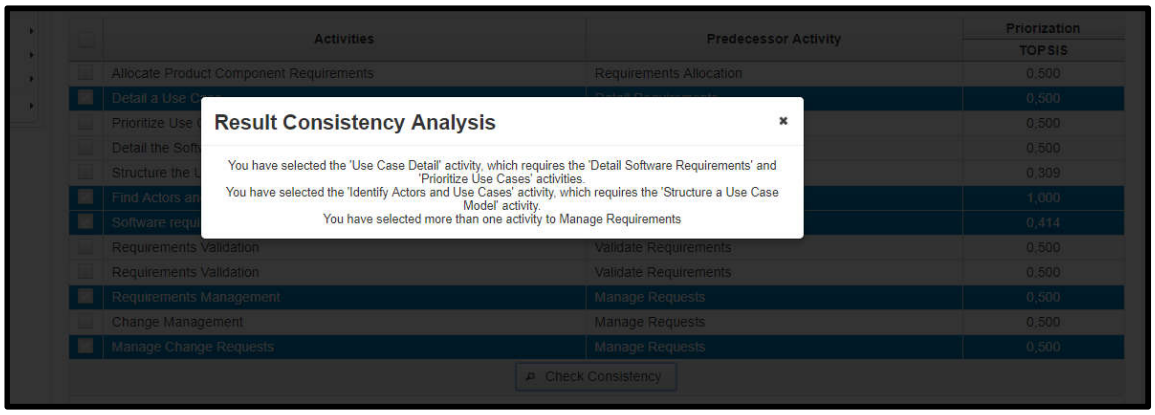
Figura 14 - Apresentação das atividades retornadas



Fonte: Autor.

- **Análise da Consistência e Completude do Processo:** Caso ocorra alguma alteração no processo, sendo incluído ou removido algum componente, a ferramenta possibilita uma verificação da consistência do processo, com base nas regras elaboradas. A ferramenta possibilita a inserção de regras de consistência de acordo com a necessidade do processo. Para a análise da consistência, basta selecionar a opção "*Check Consistency*". Acionando essa opção, a ferramenta realiza uma análise das regras com base nos componentes selecionados. Após a análise, caso ocorra alguma inconsistência na seleção das atividades, a ferramenta irá alertar o especialista que poderá realizar as devidas correções, ou até mesmo, se não existirem inconsistências o sistema irá apresentar uma mensagem de que o processo está consistente e apto para desenvolvimento. A Figura 15 exemplifica o processo de análise da consistência do processo.

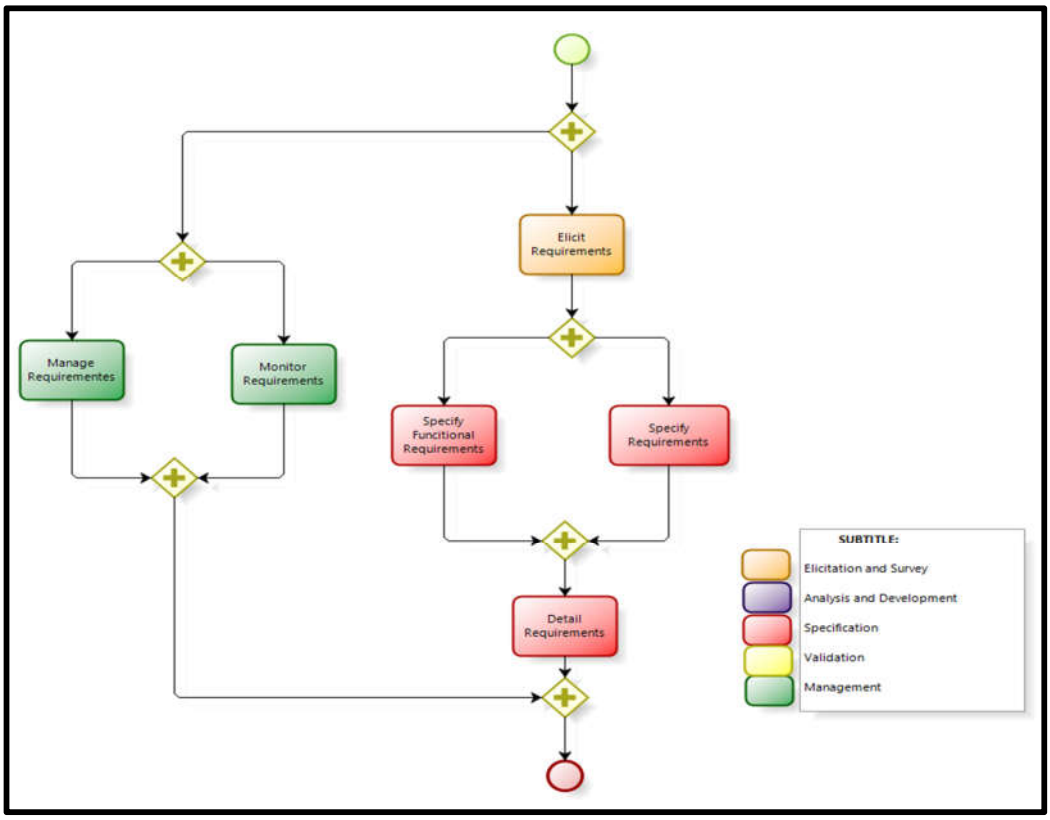
Figura 15 - Análise da Verificação de Consistência do Processo



Fonte: Autor.

- **Apresentação do processo de software adaptado** (***Creation of the tailored software processes lines***): o último passo apresenta o processo adaptado construído, ou seja, apresenta o objetivo final da abordagem. A sequência dos componentes é finalmente apresentada e possibilita ser incorporada e implementada ao projeto a qual foi destinada. A Figura 16 apresenta um exemplo de linha de processos adaptada conforme as etapas de adaptação realizadas com o apoio da ferramenta *Optimized Process Tool.*

Figura 16 – Exemplo de Linha de Processos Elaborada



Fonte: Autor.

5. VALIDAÇÃO DA PROPOSTA

Para validar a proposta, foram construídos cenários de teste para demonstração do funcionamento e efetividade da abordagem desenvolvida. O objetivo geral deste estudo foi ilustrar o funcionamento da abordagem e o uso da ferramenta de apoio desenvolvida para apoiar a abordagem semi-automatizada no desenvolvimento de processos adaptados consistentes.

5.1. DESCRIÇÃO DAS QUESTÕES DE INVESTIGAÇÃO

As questões de investigação visam avaliar as seguintes premissas, que nortearam a elaboração da abordagem. São elas:

I) A abordagem de adaptação de processos proposta no trabalho contribui significativamente para a construção de processos adaptados e consistentes?

II) A ferramenta de apoio explora e executa os conceitos apresentados na proposta de adaptação?

III) O formato na elaboração de processos de software está condizente com a realidade no desenvolvimento de projetos?

IV) A abordagem apresenta benefícios na construção de processos adaptados garantindo uma maior consistência do processo ao seu final?

5.2.DEFINIÇÃO DO CENÁRIO

A questão de pesquisa endereçada neste estudo de caso diz respeito à abordagem de semi-automação na criação de processos de software adaptados e consistentes, que visa diminuir a necessidade de um engenheiro de processos e garantir uma análise mais profunda em relação à consistência do processo elaborado. Essa questão é importante de ser explorada devido a abordagem proposta na qual envolve diversos conceitos e análises para a construção de processos de software e, além disso, devido a comprovação da sua validade antes de ser aplicada em um ambiente organizacional real.

A partir disso, podemos definir como objetivo: estabelecer uma análise sobre a abordagem de automatização na elaboração de processos de software adaptados, verificando se a mesma contribui para a construção de processos consistentes.

Foi considerado a seguinte descrição de cenário para a validação: A empresa OtimizaEvent é uma empresa de gerenciamento dos mais variados tipos de eventos e concursos. Porém, devido à competitividade nesse ramo, a empresa precisa reduzir custos. Para tanto, surgiu à ideia da empresa possuir um sistema próprio de gerenciamento de eventos que possibilite administrar todos os mecanismos de inscrições, pagamentos, emissão de comprovantes, resultados, entre outros. Até então esse processo era terceirizado pela empresa a custos elevados. Para isso, a empresa realizou algumas pesquisas de preços e selecionou uma empresa de TI que considerou apta a realizar tal desenvolvimento. Entretanto, a OtimizaEvent, exigiu que o software utilizasse mecanismos que garantissem a segurança do processo, devido a trabalhar com informações sigilosas, como informações de candidatos, notas de provas, número de cartões de crédito, entre outros. Porém, a empresa de desenvolvimento é relativamente nova no mercado de TI e teve poucos projetos em que exigiu um nível de segurança alto. Sendo assim, a empresa de TI, juntamente com a OtimizaEvent, definiu para o desenvolvimento desse software determinadas políticas e condições em relação aos riscos que devem ser prevenidos e/ou minimizados no processo de desenvolvimento, destacadas a seguir:

- R1: Evitar que Escopo / objetivos sejam pouco claros ou equivocados;
- R2: Prevenir que requisitos incompletos do software sejam elaborados;

Além da definição de riscos, no início do desenvolvimento a equipe se reuniu com a empresa e realizaram um levantamento das características que devem ser levadas em consideração, definidas conforme a Tabela 6.

Tabela 6 - Caracaterísticas do Cenário de Teste

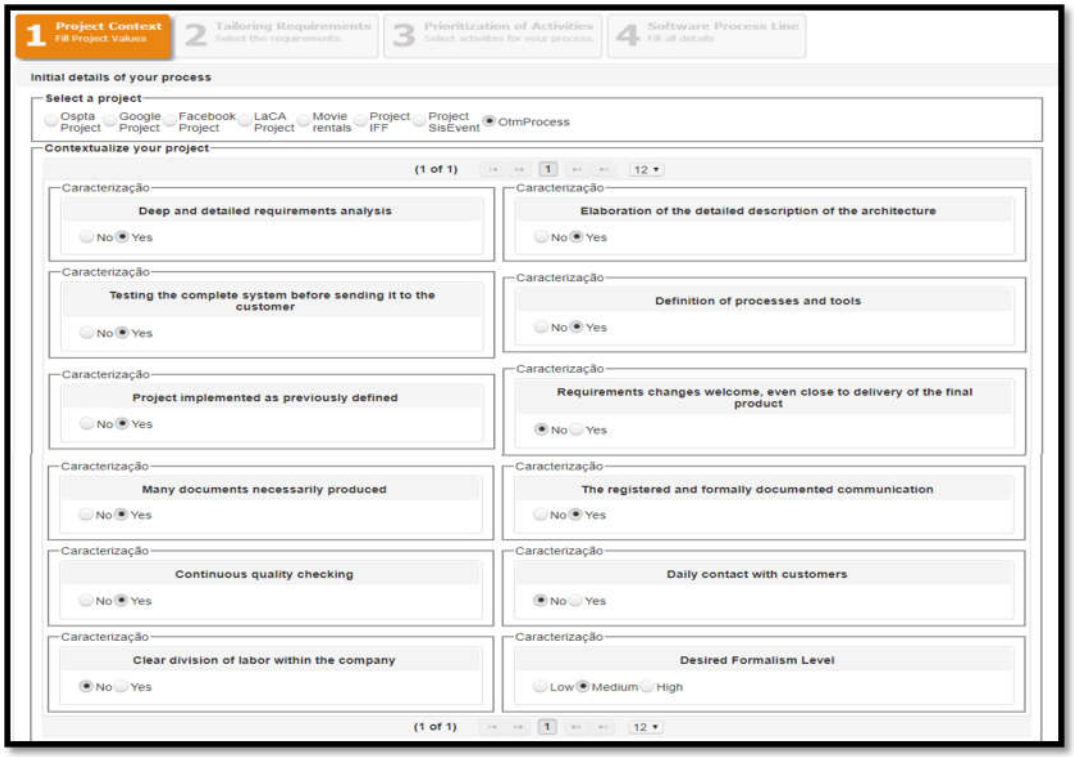
Características/Fatores	**Presente**
Análise de requisitos profunda e detalhada	Sim
Elaboração da descrição detalhada da arquitetura	Sim
Teste do sistema completo antes de mandar para o cliente	Sim
Definição dos processos e ferramentas	Sim
Projeto implementado conforme previamente definido	Sim
Mudanças de requisitos bem vindas, até mesmo perto da entrega do produto final	Não
Muitos documentos necessariamente produzidos	Sim
A comunicação registrada e formalmente documentada	Sim
Verificação contínua da qualidade	Sim
Contato diário com os clientes	Não
Clara divisão do trabalho dentro da empresa	Não
Nível de Formalismo Desejado	Médio

Fonte: Autor.

Após a definição das características e dos riscos inerentes ao projeto, será demonstrado o funcionamento da abordagem com a utilização da ferramenta de apoio. Este cenário considerou que as atividades descritas nos modelos de variabilidade já estão previamente cadastradas no repositório.

- **Inserção das informações de entrada:** a primeira etapa do cenário de teste aplicado é a inserção das informações do contexto do projeto. A Figura 17 destaca a caracterização do projeto, conforme exposto na Tabela 6.

Figura 17 - Caracterização do Cenário de Teste

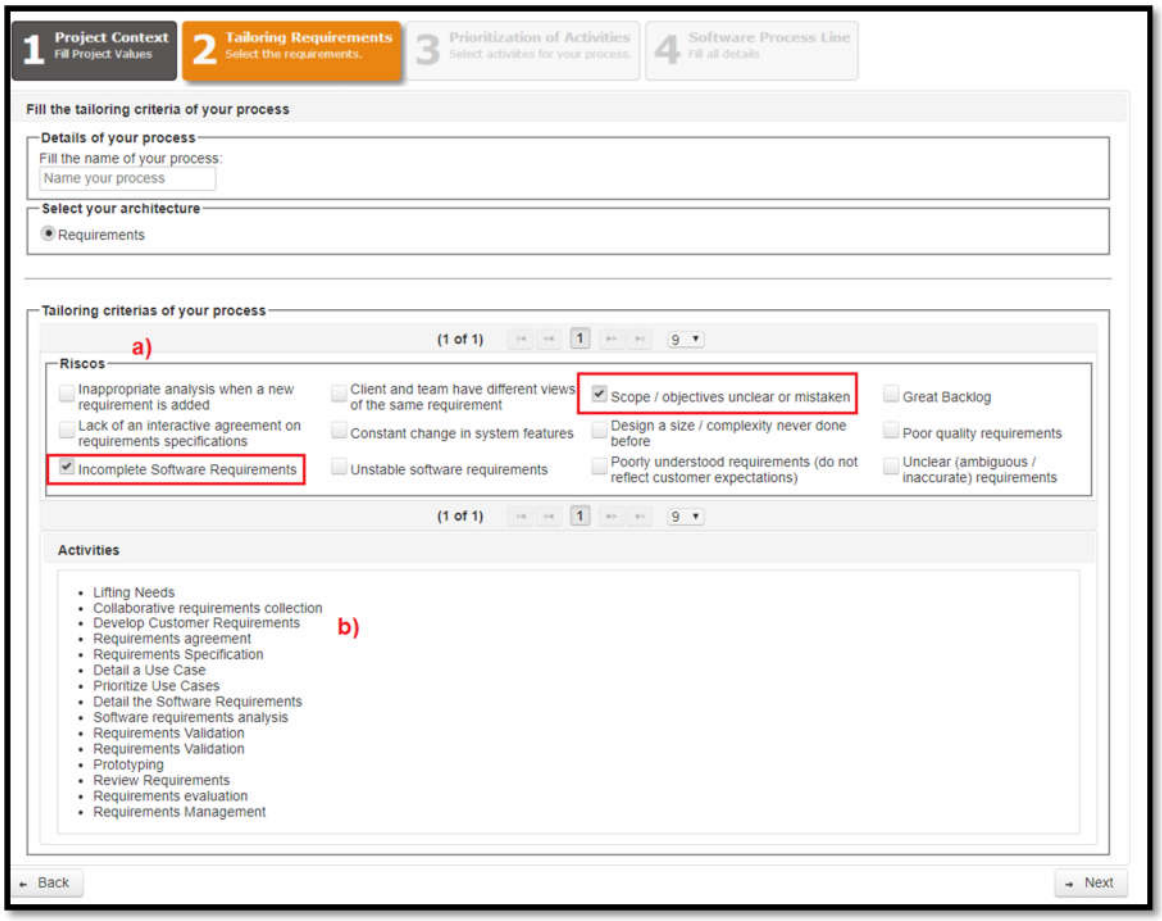


Fonte: Autor.

- **Seleção dos critérios de adaptação:** na segunda etapa da execução da abordagem temos a seleção dos riscos que se deseja prevenir. De igual forma, os riscos foram previamente detalhados. Na Figura 18 a) é realizada a seleção dos critérios de adaptação, os quais se referem aos riscos que devem ser prevenidos e/ou minimizados no projeto. Ao mesmo tempo em que são selecionados os critérios de adaptação em b) são apresentados os componentes de processo que contribuem para a prevenção ou mitigação do risco

destacado. No nosso exemplo marcamos as opções "*Scope/objectives nuclear or mistaken*" e "*Incomplete software requirements*".

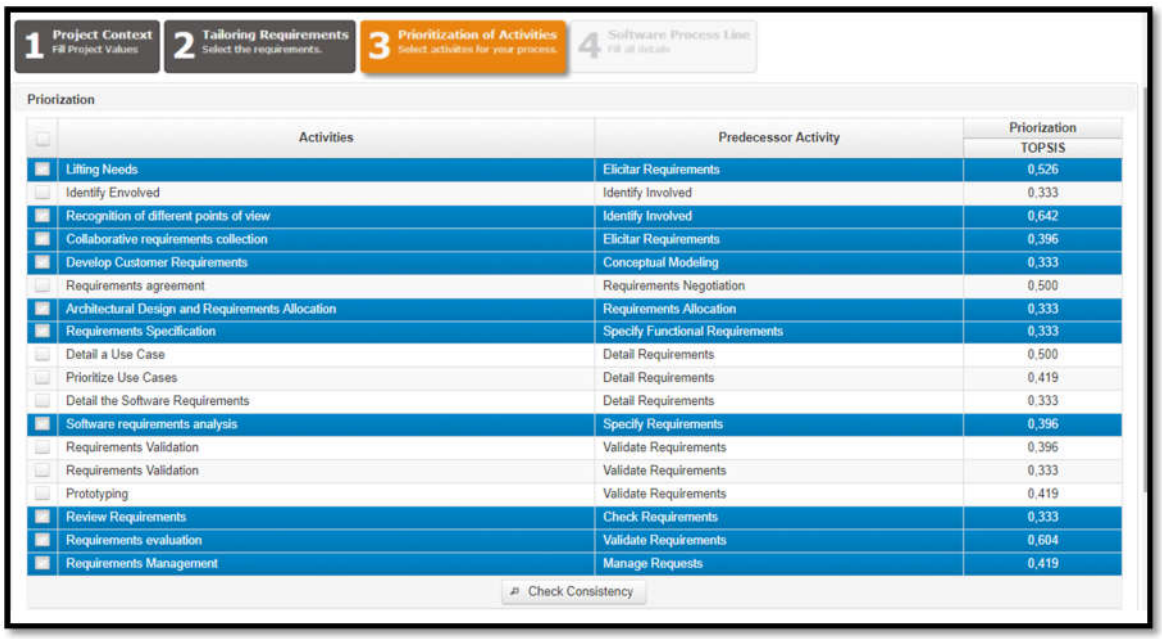
Figura 18 - Determinação dos Critérios de Adaptação



Fonte: Autor.

- **Determinação da seleção dos componentes:** Ao passar para a terceira etapa da abordagem, de acordo com análise multicritério é apresentada automaticamente uma sugestão dos componentes que melhor se adaptam ao contexto que o projeto está inserido. A Figura 19 ilustra a apresentação semiautomática dos componentes que podem ser selecionadas. Pode-se verificar na que a abordagem selecionou os componentes de processo que mais se assemelham com as características do processo, ou seja, as que possuem maior valor de priorização conforme a análise multicritério da técnica TOPSIS.

Figura 19 - Apresentação da xautomação do processo

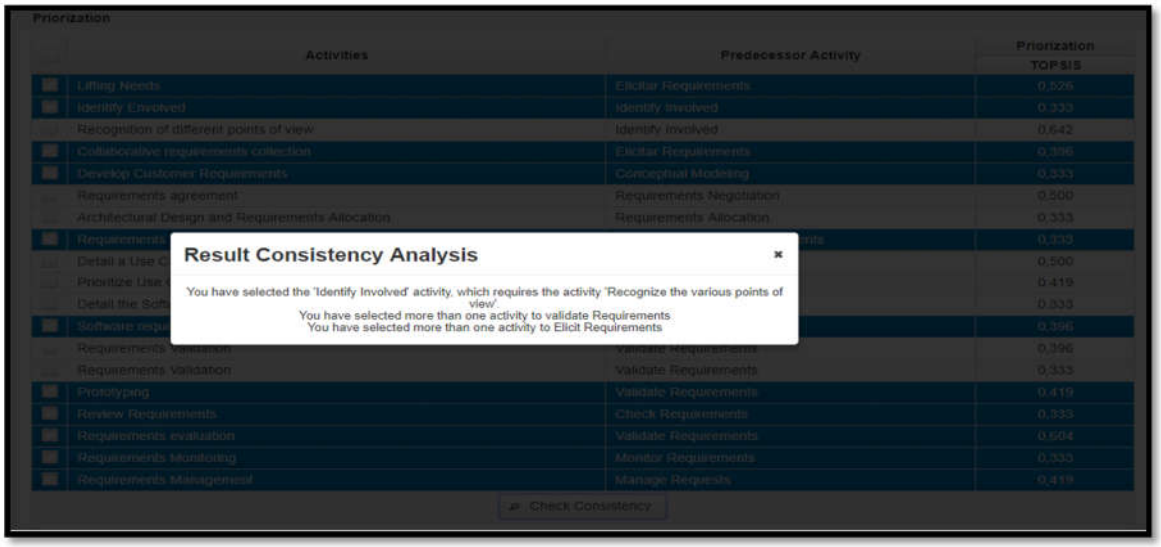


Fonte: Autor.

- **Análise da Consistência:** neste momento o engenheiro de processos pode decidir modificar algum componente, e/ou deseja analisar se o processo está consistente com base nas regras criadas a partir dos modelos de variabilidades. Para isso, a abordagem apresenta uma opção para verificar a consistência do processo. Caso exista alguma incoerência, o sistema irá acusar. Para exemplificar, foram alteradas algumas seleções para demonstrar o funcionamento da consistência do processo, e ao checar a consistência, são apresentados os erros conforme a Figura 20. É possível verificar que a abordagem apresentou três incoerências, sendo duas incoerências referente a seleção de mais de um componente para execução, e uma incoerência referente a uma dependência entre componentes de processos. A opção por correção da incoerência é de decisão do engenheiro de processos.

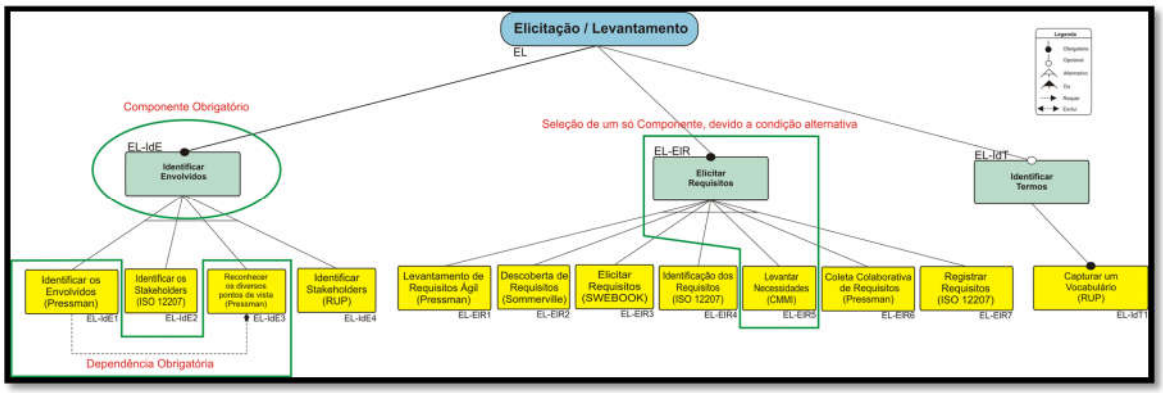
Para exemplificar melhor a aplicação das regras dos modelos de variabilidades, é apresentada a Figura 21, na qual é possível verificar que a abordagem validou três regras que foram construídas para garantir a consistência do processo. O mesmo ocorre para os demais modelos de variabilidades que estão expressos no Apêndice C deste trabalho.

Figura 20 - Análise da Consistência do Processo



Fonte: Autor.

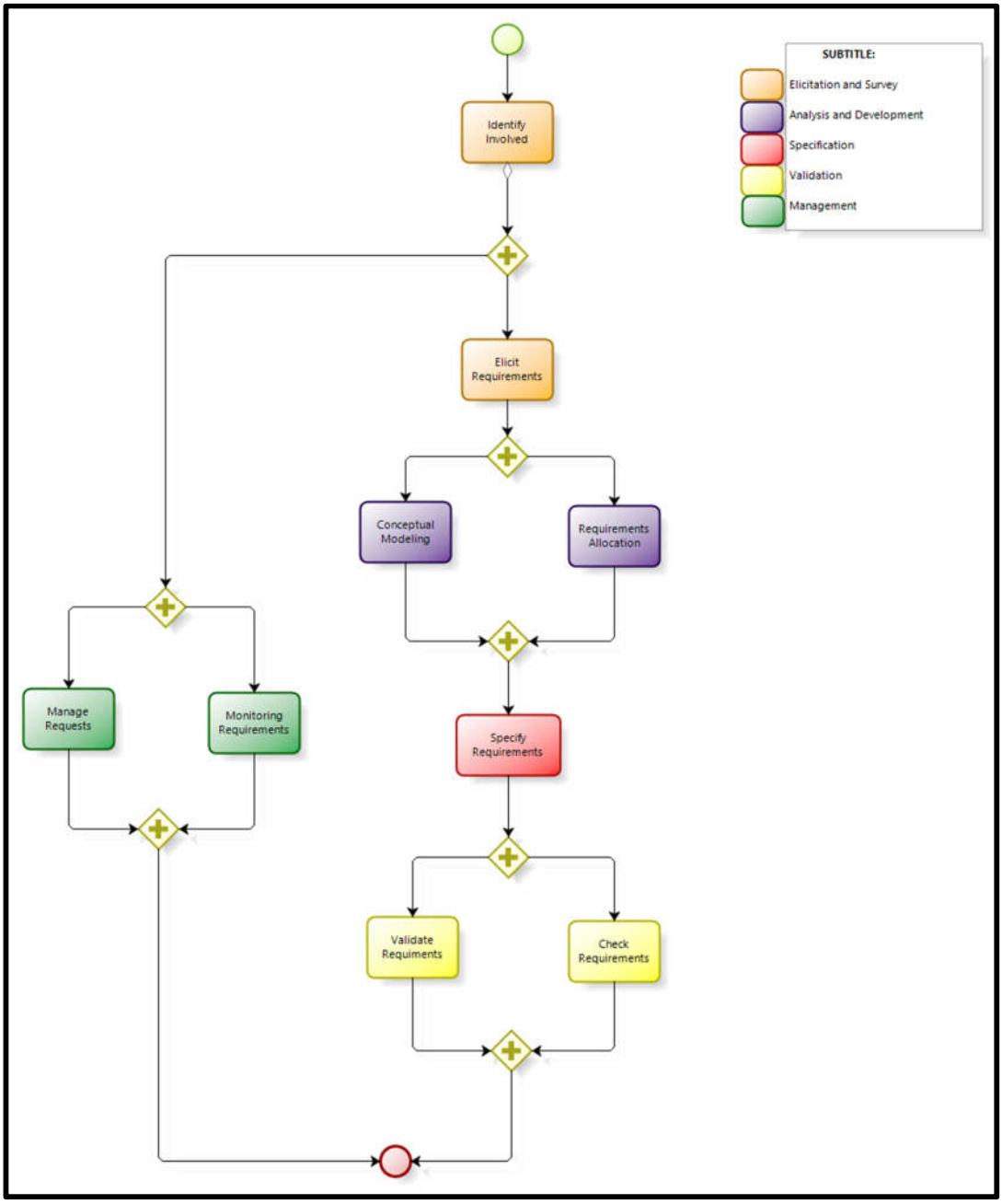
Figura 21 - Exemplo de regras aplicadas no processo



Fonte: Autor.

- **Processo de software adaptado:** e como última etapa e produto de trabalho, é apresentado o processo de software adaptado com os componentes abstratos destacados, conforme demostrado na Figura 22.

Figura 22 - Processo de Software Adaptado e Consistente



Fonte: Autor.

Pela apresentação do cenário, tentou-se demonstrar o funcionamento e a efetividade da abordagem e da ferramenta de apoio desenvolvidas. Como pode-se visualizar, os resultados da aplicação são coerentes. Sendo assim, é possível afirmarmos que houve uma redução na necessidade de um engenheiro de processos com conhecimento avançados na elaboração de

processos de software. Como validações futuras, pretende-se aplicar essa técnica em empresas de desenvolvimento, buscando a vivência de um cenário real para otimização das regras e da ferramenta desenvolvida para apoiar a abordagem.

Após a definição e execução do cenário de teste descrito, é necessário retomarmos inicialmente as questões de investigação buscando respondê-las. As questões de investigação e suas respostas são descritas abaixo:

I) A abordagem de adaptação de processos proposta no trabalho contribui significativamente para a construção de processos adaptados e consistentes? R.: De acordo com o desenvolvimento do cenário de teste, foi possível realizar a validação da abordagem proposta, onde é possível verificar que a abordagem permite a construção de um processo de software adaptado, realizando a seleção dos componentes de processo mais adequados e, além disso, levando em consideração critérios de adaptação e o contexto em que o projeto está inserido. As regras complementam e contribuem na garantia da construção de um processo de software mais consistente. Portanto, podemos sim, afirmar que a abordagem contribui significativamente para a construção de processos de software adaptados e mais consistentes.

II) A ferramenta de apoio explora e executa os conceitos apresentados na proposta de adaptação? R.: Sim, foi possível demonstrar que as quatro etapas que seguem o módulo de adaptação, expressam fielmente a abordagem esboçada no desenvolvimento deste trabalho. Além disso, a ferramenta de apoio se mostrou uma facilitadora na construção do processo de software, possibilitando de forma dinâmica, a ligação entre os componentes e apresentando resultados satisfatórios na adaptação de processos de software.

III) O formato na elaboração de processos de software está condizente com a realidade no desenvolvimento de projetos? R.: De acordo com a análise realizada através dos trabalhos relacionados, o processo de adaptação proposto nesta abordagem se assemelha bastante com outros já utilizados, ou seja, a abordagem apresentou uma sequência de etapas que realizaram uma análise do contexto, critérios de adaptação e de regras de consistência, que culminaram em um processo de software adaptado. Sendo assim, podemos afirmar com certeza que o formato de elaboração de processos de software condiz com a realidade no desenvolvimento de projetos.

IV) A abordagem apresenta benefícios na construção de processos adaptados garantindo uma maior consistência do processo ao seu final? R.: De acordo com a análise realizada na demonstração do cenário de teste, a abordagem se demonstrou benéfica na construção de

processos de software adaptados, considerando diferentes fatores e realizando uma análise a partir de diferentes conceitos. O uso de análise multicritério para a seleção de componentes de processos, aliadas a atribuição de regras de consistência mostrou bons resultados e garantiu a construção de um processo de software condizente com o selecionado e com o projeto. Existe a possibilidade da ferramenta conter inconsistências, devido a isso é necessário análises mais elaboradas para garantir melhores resultados, porém esses podem ser buscados no decorrer de sua utilização e na inserção de componentes em contextos de projetos reais.

Portanto, após a apresentação do cenário de teste e das respostas as perguntas de investigação, podemos afirmar que a abordagem apresenta grandes benefícios na adaptação de processos, considerando informações relevantes como dados de entrada e por fim, realizando um processamento coerente com a construção de processos adaptados. Em cenários organizacionais será possível realizar o levantamento de novas necessidades e destacar melhorias, mas o propósito inicial na apresentação dessa validação foi atingida e concluída com êxito, apresentando bons resultados.

6. TRABALHOS RELACIONADOS

Neste capítulos são descritos os principais trabalhos relacionados que contribuiram para a realização da pesquisa científica descrita no decorrer deste trabalho. Diversas abordagens visam adaptar o processo com base nas características do projeto ou organização, entretanto, poucas são pesquisas buscam validar.a consistência e a completude do processo.

A abordagem de Pereira (2011) apresenta uma infraestrutura que suporta a definição e adaptação de processos de software consistentes baseados no metamodelo SPEM 2.0. Abrange os principais elementos e relacionamentos de um processo de software e aborda aspectos de sua consistência tanto para as atividades de definição quanto para as atividades de adaptação. As regras de boa formação focam na garantia de que as dependências de um processo de software sejam respeitadas durante uma operação de adaptação, evitando inconsistências no processo resultante. Os autores utilizaram o *Eclipse Modeling Framework* para desenvolver uma ferramenta de apoio. Para validação da proposta, foram utilizados cenários de teste, com simulações de ocorrências para demonstrar o funcionamento das regras e dos aspectos referentes a abordagem.

Alegría *et al.* (2013) apresentaram uma proposta de adaptação de processos na qual propõem uma abordagem para adaptar automaticamente processos organizacionais para contextos de projeto com base em técnicas de engenharia orientadas a modelos (MDE) para que os processos apropriados sejam criados de forma rápida e com pouco esforço. Engenharia orientada a modelo (MDE) (SCHMIDT, 2006) é uma abordagem de desenvolvimento de software na qual modelos abstratos são definidos e, sistematicamente transformados em modelos concretos e, eventualmente, em código fonte. Esta abordagem promove a reutilização por meio de uma estratégia generativa.

A adaptação é implementada por meio de uma série de transformações cujas entradas são o processo organizacional incluindo variabilidades e um modelo do contexto do projeto, e cuja saída é o processo adaptado ao contexto. Os autores destacam que o processo de adaptação é automático, que segue regras de transformação definidas em ATL (*Atlas Transformation Language*) e já se aplica a transformações validadas, ou seja, definidas por engenheiros de processos. Como validação, o trabalho apresentou um estudo de caso no qual mostrou que é possível aplicar transformações para adaptar um processo de engenharia de requisitos a diferentes contextos de projeto de forma planejada (ALEGRÍA, *et al.*, 2013).

Como complemento ao trabalho de Alegría *et al.* (2013), Silvestre *et al*. (2014) apresentam uma abordagem que requer dois modelos de entrada: um modelo de processo de

software organizacional e um modelo de contexto do projeto. O modelo de contexto organizacional indica os atributos de projeto que podem influenciar a adaptação do processo junto com seus potenciais valores. As regras de transformação da adaptação da proposta são escritas em ATL. Para cada elemento variável identificado como parte do processo organizacional, há uma regra incluída na transformação. Para os elementos de processo opcionais, a regra decide de acordo com os valores dos atributos do modelo do contexto do projeto, se ele deve ser incluído ou não no processo adaptado.

Embora esta proposta de adaptação tenha-se mostrado tecnicamente viável em cenários reais, tem clara limitações importantes relatados por engenheiros de processo. Por essa razão, o autor desenvolveu uma Arquitetura de Regras de Adaptação, que corresponde a uma ferramenta que permite aos engenheiros de processo definir interativamente as regras do processo de adaptação usando uma interface gráfica de usuário. A saída da ferramenta é a transformação adaptada que pode ser utilizada para adaptar o processo de software organizacional (SILVESTRE, *et al.*, 2014).

Em relação a riscos inerentes a projetos de software, é possível destacar o trabalho de Maia (*et al*., 2004), que propõe um modelo para auxiliar o Engenheiro de Processos durante a adaptação do processo padrão de uma dada organização de software para um projeto específico da mesma, baseado nas características do projeto, nas diretrizes de adaptação do processo padrão e nas informações acerca de adaptações realizadas anteriormente. Em seu trabalho, Maia (*et. al.,* 2004) utiliza conceitos de regras de adaptação e casos que direcionam a área de gerenciamento de riscos. Não fica claro se os autores utilizaram ferramenta de apoio e como a proposta foi validada.

A partir da leitura e análise dos trabalhos relacionados foi possível verificar que a área de estudo que envolve a adaptação de processos possui várias soluções que objetivam melhorar a construção de processos de softwares para as organizações. Porém, em sua grande maioria, as soluções propostas pouco comprovam a consistência e a completude dos processos gerados, exigindo uma grande interferência de engenheiros de processos experientes. Desta forma, o presente trabalho se torna relevante pois busca resolver esses dois problemas pouco explorados na literatura, que são a análise e verificação da consistência e da completude dos processos gerados, e a construção de mecanismos que visam diminuir a interferência de um engenheiro de processos experiente na tomada de decisões referente ao projeto.

7. RESULTADOS E CONSIDERAÇÕES FINAIS

Este trabalho apresenta uma solução para o desenvolvimento e geração de Processos de software para projetos específicos. Através da abordagem, é possível construir uma linha de processos levando em consideração as características do projeto, visando o cumprimento de critérios de adaptação e, por fim, resultando em um processo de software adaptado e consistente.

O trabalho foca na aplicação da abordagem para o gerenciamento de requisitos, considerando a definição de diferentes componentes de processos a partir de distintas metodologias de desenvolvimento como RUP, CMMI, ISO 12207, e autores como Sommerville e Pressman. Os componentes selecionados foram classificados, agrupados e caracterizados de acordo com seus objetivos, fontes de extração e fases de aplicação. A partir desse levantamento, foi possível reutilizar esses componentes em diferentes processos.

A caracterização do projeto ocorre com a aplicação de um questionário que engloba doze perguntas chave, respondidas no início da elaboração do processo. Essas características são avaliadas e comparadas as características das atividades com base em uma análise multicritério denominada TOPSIS. O conceito considera que a melhor alternativa é aquela que é a mais próxima da solução ideal composta de todos os melhores valores atingíveis dos critérios de benefício; já a solução ideal negativa consiste em todos os piores valores atingíveis dos critérios de sustentabilidade.

Visando exemplificar e aplicar a abordagem proposta, desenvolveu-se uma ferramenta de apoio denominada "*Otimized Process Tool*", que apresenta módulos e recursos para facilitar o desenvolvimento de um processo adaptado levando em consideração os critérios e soluções deste trabalho.

Este trabalho se difere dos trabalhos relacionados, pois apresenta uma abordagem que semi-automatiza a tomada de decisões do especialista, apresentando uma semi-automação na criação da adaptação do processo. Além disso, regras de consistência são aplicadas ao processo adaptado, garantindo uma maior consistência e completude do mesmo. Essas regras foram elaboradas utilizando modelos de variabilidades, que facilitam a compreensão e a formalização dos elementos criados. As regras foram criadas com base no conhecimento de especialistas e introduzidas na ferramenta de apoio da abordagem proposta.

A validação deste trabalho ocorreu a partir da aplicação de um cenário de teste que buscou expressar uma situação de desenvolvimento, na qual se busca a elaboração de um processo de software. De acordo com a análise realizada, foi possível demonstrar a

possibilidade de redução na necessidade de um engenheiro de processos, e a comprovação de que as técnicas de análise multicritério, associada a regras de consistência possibilitam validar com uma maior coerência o processo de software construído. Uma aplicação em um cenário real pode contribuir ainda mais no aprimoramento da abordagem e da ferramenta desenvolvida. É importante destacar que o foco desse trabalho não é o desenvolvimento da ferramenta, mas sim a demonstração da abordagem proposta, sendo que a ferramenta funciona como recurso de apoio para demonstrar o funcionamento da abordagem.

Portanto, realizando uma análise geral do trabalho desenvolvido, é possível concluir que a abordagem desenvolvida possui um viés científico válido, pois apresenta uma técnica nova para adaptação de processos com a associação de diferentes conceitos, apresentando uma solução consistente e facilitadora na adaptação de processos.

Como trabalhos futuros, sugere-se a inclusão ou estudo de um módulo de avaliação de processos adaptados, criação dinâmica e interativa de um fluxo da linha de processos, podendo inserir e remover elementos de software de acordo com suas caraterísticas situacionais e critérios de adaptação. Além disso a expansão de novas funcionalidades como, por exemplo, a criação de uma página web interativa com o processo criado, apresentando todas as informações de atividades, tarefas e papéis que estão contidos no processo, criação de um recurso que avalia a inserção de novas regras de boa formação de processos, para que uma regra não exclua ou inative o uso de uma já cadastrada.

REFERÊNCIAS BIBLIOGRÁFICAS

AHN, Y. W.; AHN, H. J.; PARK, S. J. **Knowledge and Case-Based Reasoning for customization of software processes**. International Journal of Software Engineering and Knowledge Engineering, 2003.

AKBAR, R.; HASSAN, M. F.; ABDULLAH, A. **A Review of Prominent Work on Agile Processes Software Process Improvement and Process Tailoring Practices**. Springer Berlin Heidelberg, 2011.

ALEGRÍA, J. A. H.; BASTARRICA, M. C.; QUISPE, A. e OCHOA, S. F. **MDE-based process tailoring strategy**. Chile: Wiley Online Library, 2013.

ARMBRUST, O. et al. **Scoping software process lines**. New York: Software Process: Improvement and Practice - Examining Process Design and Change, 2009.

ATKINSON, D. C.; WEEKSJOHNNOLL, D. C. **Tool support for iterative software process modeling**. Information and Software Technology, 2009.

BECK, K. **Programação Extrema (XP) Explicada**: Acolha as Mudanças. Brasil: Bookman, 2004.

BENAVIDES, D.; SEGURA, S.; CORTES, A. R. **Automated Analysis of Feature Models 20 Years Later**: A Literature Review, 2010.

BOEHM, B.; TURNER, R. **Using Risk to Balance Agile and Plan-Driven Methods**. IEEE Computer Society, 2003.

BOURQUE, PIERRE; FAIRLEY, RICHARD E. **SWEBOK**: Guide to the Software Engineering Body of Knowledge, 2014.

BOTTI, LAURENT, PEYPOCH, NICOLAS. **Multi-criteria ELECTRE method and destination competitiveness**. Artigo publicado na Revista Elsevier, 2013.

CAMERON, J. **Configurable Delepment Process**. COMMUNICATIONS OF THE ACM, 2002.

CESARE, S. D. et al. **Tailoring Software Development Methodologies in Practice**: A Case Study. Journal of Computing and Information Technology, 2008.

CLARKE, P.; CONNOR, R. V. O. **The situational factors that affect the software development process:** Towards a comprehensive reference framework. Journal of Information Software and Technology, 2012.

CMMI. **CMMI for Development**. 3ª ed. s.l.:Addison-Wesley, 2013.

COCKBURN, A. **Crystal clear:** a human-powered methodology for small teams. Pearson Education, 2004.

GINSBERG, M. P.; QUINN, L. H. **Process Tailoring and the Software Capability Maturity Model**. SEI Joint Program Office, 1995.

GOMES, Luiz F. A. M.; MARANHÃO, Francisco J. C. **A Exploração de Gás Natural em Mexilhão: Análise Multicritério pelo método TODIM**. Revista Pesquisa Operacional, v.28, Rio de Janeiro, 2008. Disponível em: <http://www.scielo.br/pdf/pope/v28n3/v28n3a06.pdf>. Acesso em: 12 de Agosto 2016.

HALL, E. **Managing Risk**: Methods for Software Systems Development. Addison-Wesley, 2003.

HENNINGER, S. R. **An Environment for Reusing Software Processes**. ICSR '98 Proceedings of the 5th International Conference on Software Reuse, 1998.

HE, R., WANG, H. e LIN, Z. **A Software Process Tailoring Approach Using a Unified Lifecycle Template**. Computational Intelligence and Software Engineering, 2009.

HURTADO, J. A. e BASTARRICA, M. C. **Process Model Tailoring as a Means for Process Knowledge Reuse**. 2nd Workshop on Knowledge Reuse, 2009.

HWANG, C.-L.; YOON, K.. **Multiple Attribute Decision Making**: Methods and Applications A State-of-the-Art Survey, 1981.

ISO/IEC 12207. **Systems and software engineering** - Software life cycle processes. IEEE, 2008.

ISO/IEC 15504.**Tecnologia da informação - Avaliação de processo**, 2008.

JAUFMAN, O.; MÜNCH, J. **Acquisition of a project-specific process.** Springer, 2005.

JENERS, S. et al. **Harmonizing Software Development Processes with Software Development Settings** – A Systematic Approach, 2013.

JOHANSSON, C. T**he V-Model. Quality Managment**, 1999.

KANG, K. C.; LEE, H. **Variability Modeling**. Systems and Software Variability Management, 2013.

KALUS, G.; KUHRMANN, M. **Criteria for Software Process Tailoring:** A Systematic Review. Proceedings of the 2013 International Conference on Software and System Process, 2013.

KRUCHTEN, P. **Contextualizing agile software development**. Journal of Software: Evolution and Process, v. 25, n. 4, p. 351–361, 2013.

MAIA, A. B., et al. **Um Modelo para Auxiliar a Adaptação de Processos de Software**. IV Congresso Brasileiro de Computação, 2004.

MARTINEZ-RUIZ, T. et al. **Applying AOSE Concepts to Model Crosscutting Variability in Variant-Rich Processes**. Publicado em: *Software Engineering and Advanced Applications*, 2011.

MARTINS, D. A.; et al. **Caracterização do processo de Desenvolvimento de Software**: Um Estudo de múltiplos Casos da Região metropolitana de Recife. Belo Horizonte, 2011.

OMG. **Software & Systems Process Engineering Meta-Model Specification**, 2008.

PEDREIRA, O. et al. **A Systematic Review of Software Process Tailoring**. ACM, 2007.

PEREIRA, E. B., et al. **A Set of Well-Formedness Rules to Checking the Consistency of the Software Processes based on SPEM 2.0**. Rio de Janeiro: Springer, 2011.

PEREIRA, E. B. **Uma infraestrutura para consistência dos processos de software baseados no metamodelo SPEM 2.0.** Porto Alegre, 2011.

PETRY, J. F., et.al. **Análise Decisória Multicritério na avaliação da Sustentabilidade dos municípios de Santa Catarina.** Revista de Gestão do Unilasalle, ISSN 2316-5537, 2014.

PINNA, C. C. d. A.; CARVALHO, M. M. **Gestão de Escopo em projetos de aplicações Web**. Revista Produção, 2008.

PMI**. Um Guia do Conhecimento em Gerenciamento de Projetos**. 5ª ed. Newtown Square, Pennsylvania, EUA, 2013.

PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de Software** - Uma Abordagem Profissional. 8ª ed. Porto Alegre: Bookman, 2016.

PROMETHEE MÉTHODS. **Visual Promethee 1.4 Manual**, 2013. Disponível em: <http://www.promethee-gaia.net/files/VPManual.pdf>. Acesso em Outubro de 2015.

ROMBACH, D. **Integrated software process and product lines.** Verlag Berlin, Heidelberg: Springer - Unifying the Software Process Spectrum, 2006.

RUI, H., HAO, W.; ZHIQING, L. **A Software Process Tailoring Approach Using a Unified Lifecycle Template**. IEEE, 2009.

RUIZ, T. M.; MÜNCH, J., García, F. e PIATTINI, M. **Requirements and constructors for tailoring software processes:** a systematic literature review. Software Quality Journal, 2012.

RUP. **Rational Unified Process**, 2001.

SAATY, T. L ***The Analytic Hierarchy Process***. N. York, USA: McGraw-Hill, 1980.

SCHMIDT, D. C. **Model-Driven Engineering**. IEEE Computer Society, 2006.

SCHWABER, K.; SUTHERLAND, J. **Guia do Scrum**. Scrum Org. and Scrum Inc., 2013.

SCHWABER, K.; BEEDLE, M. **Agile Software Development with Scrum**. Prentice Hall, 2003.

SILVA, Renaud B.; NETTO, Maria A. C. **Uma estrutura de apoio à decisão para orientar a escolha de projetos prioritário para a infraestrutura de transporte do Brasil.** 2010. Disponível em: <http://www.academia.edu/2117266/...>. Acessado em: 07 de Janeiro de 2017.

SILVESTRE, L.; BASTARRICA, M. C.; OCHOA, S. F. **A Model-based Tool for Generating Software Process Model Tailoring Transformations**. IEEE, 2014.

SIMMONDS, J.; BASTARRICA, M. C.; SILVESTRE, L.; QUISPE, A. **Analyzing Methodologies and Tools for Specifying Variability in Software Processes**, 2011.

SOMMERVILLE, Ian. **Engenharia de Software** – 9ª Edição. Editora Pearson, 2011.

STARFIELD, Tony. **Simple Multi-Attribute Ranking Technique** - SMART, 2005.

TELES, V. M. **Extreme Programming**. 2ª ed. São Paulo - SP, Brasil: Novatec, 2014.

VARGAS, R. V. **Gerenciamento de Projetos:** Estabelecendo diferenciais competitivos, 2016.

XU, P. e RAMESH, B. **A Tool for the capture and use of Process knowledge in process tailoring**. IEEE - Proceedings of the 36th Hawaii International Conference on System Sciences, 2002.

XU, P.; RAMESH, B. **Software Process Tailoring: An Empirical Investigation.** Journal of Management Information Systems, 2007.

XU, P.; RAMESH, B. **Using Process Tailoring to Manage Software Development Challenges**, 2008.

ZAKARIA, N. A.; MAHRIN, S. I. M. N. **The State of the Art and Issues in Software Process Tailoring**. IEEE - International Conference on Software Engineering and Computer Systems, 2015.

APÊNDICES

APÊNDICE A – LEVANTAMENTO DAS ATIVIDADES.

Etapa/Fase	Atividades	Responsável	Fonte	Descrição
Elicitação e Levantamento de Requisitos	Levantar Necessidades	Analisa do Sistema	CMMI	O levantamento de necessidades vai além da coleta de requisitos, envolvendo a identificação proativa de requisitos adicionais não fornecidos explicitamente pelos clientes. Recomenda-se que os requisitos adicionais levem em consideração as várias atividades do ciclo de vida do produto e seus impactos sobre o produto.
	Descoberta de Requisitos	Analisa do Sistema	SOMMERVILLE	Essa é a atividade de interação com os stakeholders do sistema para descobrir seus requisitos. É o processo de reunir informações sobre o sistema requerido e os sistemas existentes e separar dessas informações os requisitos de usuário e de sistema.
	Identificação de envolvidos	Analisa do Sistema	PRESSMAN	Identificar qualquer pessoa que se beneficie de forma direta ou indireta do sistema que está sendo desenvolvido.
	Reconhecimento de diversos pontos de vista	Analisa do Sistema	PRESSMAN	Como há muitos envolvidos diferentes, os requisitos do sistema deverão ser explorados sob vários pontos de vista.
	Coleta colaborativa de requisitos	Analisa do Sistema	PRESSMAN	O objetivo é identificar o problema, propor elementos da solução, negociar diferentes abordagens e especificar um conjunto preliminar de requisitos da solução em uma atmosfera que seja propicia para o cumprimento da meta.
	Levantamento de requisitos ágil	Analisa do Sistema	PRESSMAN	O levantamento de requisitos é feito pedindo-se aos envolvidos para que criem jornadas de usuário. Cada jornada de usuário descreve um requisito simples do sistema, redigido do ponto de vista do usuário.
	Elicitar as Solicitações dos Stakeholders	Analisa do Sistema	RUP	Esta tarefa descreve como obter pedidos das partes interessadas sobre o que eles gostariam que o sistema fornecesse.
	Levantar um Vocabulário Comum	Analisa do Sistema	RUP	Esta tarefa descreve como definir o conjunto comum de termos que precisam ser usados consistentemente no projeto.
	Identificação dos Stakeholders	Analisa do Sistema	ISO 12207-2008	O projeto deve identificar as partes interessadas individuais ou classes de partes interessadas que tenham um interesse legítimo no sistema ao longo do seu ciclo de vida.
	Identificação dos Requisitos	Analisa do Sistema	ISO 12207-2008	O projeto deve suscitar os requisitos das partes interessadas. O projeto deve definir as limitações de uma solução de sistema
	Registrar Requisitos	Analisa do Sistema	ISO 12207-2008	O projeto deve registar os requisitos das partes interessadas numa forma adequada para a gestão dos requisitos durante o ciclo de vida e para além.
	Elicitar Requisitos	Analisa do Sistema	SWEBOK	A elicitação de requisitos está relacionada com as origens dos requisitos de software e como o engenheiro de software pode coletá-los. É o primeiro estágio na construção de uma compreensão do problema que o software é necessário para resolver. É fundamentalmente uma atividade humana e é onde os stakeholders são identificados e os relacionamentos estabelecidos entre a equipe de

				desenvolvimento e o cliente.
Análise e Elaboração	Desenvolver Requisitos de Cliente	Especificador de Requisitos	CMMI	As necessidades das partes interessadas, suas expectativas, restrições e interfaces, são conflitantes ou identificadas de forma insuficiente. Para que as necessidades das partes interessadas, suas expectativas, restrições e limitações sejam claramente identificadas e compreendidas, utiliza-se um processo iterativo ao longo da vida do projeto para que esse objetivo seja alcançado. Para facilitar o nível esperado de interação, um substituto do usuário final ou do cliente é frequentemente envolvido para representar suas necessidades e auxiliar na resolução de conflitos. Podem ser utilizados como substituto as unidades de marketing ou de relacionamento com o cliente da organização, assim como os membros da equipe de desenvolvimento de disciplinas tais como desenvolvimento ou suporte de interface humano-computador. Recomenda-se que restrições ambientais e legais, entre outras, sejam consideradas na criação e determinação do conjunto de requisitos do cliente.
	Analisar Requisitos Visando ao Balanceamento	Analisa do Sistema	CMMI	Analisar os requisitos para balancear as necessidades e as restrições das partes interessadas. As necessidades e restrições das partes interessadas podem tratar custo, prazo, desempenho, funcionalidade, componentes reusáveis, manutenabilidade ou risco.
	Classificação e organização de requisitos	Analisa do Sistema	SOMMERVILLE	Essa atividade toma a coleção de requisitos não estruturados, agrupa requisitos relacionados e os organiza em grupos coerentes.
	Priorização e negociação de requisitos	Analisa do Sistema	SOMMERVILLE	Essa atividade está relacionada com a priorização de requisitos e em encontrar e resolver conflitos por meio da negociação de requisitos.
	Construção do modelo de análise	Analisa do Sistema	PRESSMAN	O objetivo do modelo de análise é fornecer uma descrição dos domínios informacional, funcional e comportamental necessários para um sistema baseado em computador. O modelo é modificado dinamicamente à medida que você aprende mais sobre o sistema a ser construído e outros envolvidos adquirem um melhor entendimento sobre aquilo que realmente querem. Por essa razão, o modelo de análise é uma reprodução dos requisitos em determinado momento. É esperado que ele mude.
	Negociação de requisitos	Analisa do Sistema	PRESSMAN	O intuito da negociação é desenvolver um plano de projeto que atenda às necessidades dos envolvidos e, ao mesmo tempo, reflita as restrições do mundo real (por exemplo, tempo, pessoal, orçamento) impostas à equipe de software.
	Desenvolver Visão	Analisa do Sistema	RUP	Esta tarefa descreve como desenvolver a visão geral do sistema, incluindo o problema a ser resolvido, as principais partes interessadas, o escopo / limite do sistema, as principais características do sistema e quaisquer restrições
	Aceitação dos Requisitos	Stakeholders	ISO 12207-2008	O projeto deve alimentar os requisitos analisados para as partes interessadas pertinentes para assegurar que as necessidades e expectativas foram adequadamente capturadas e expressas.

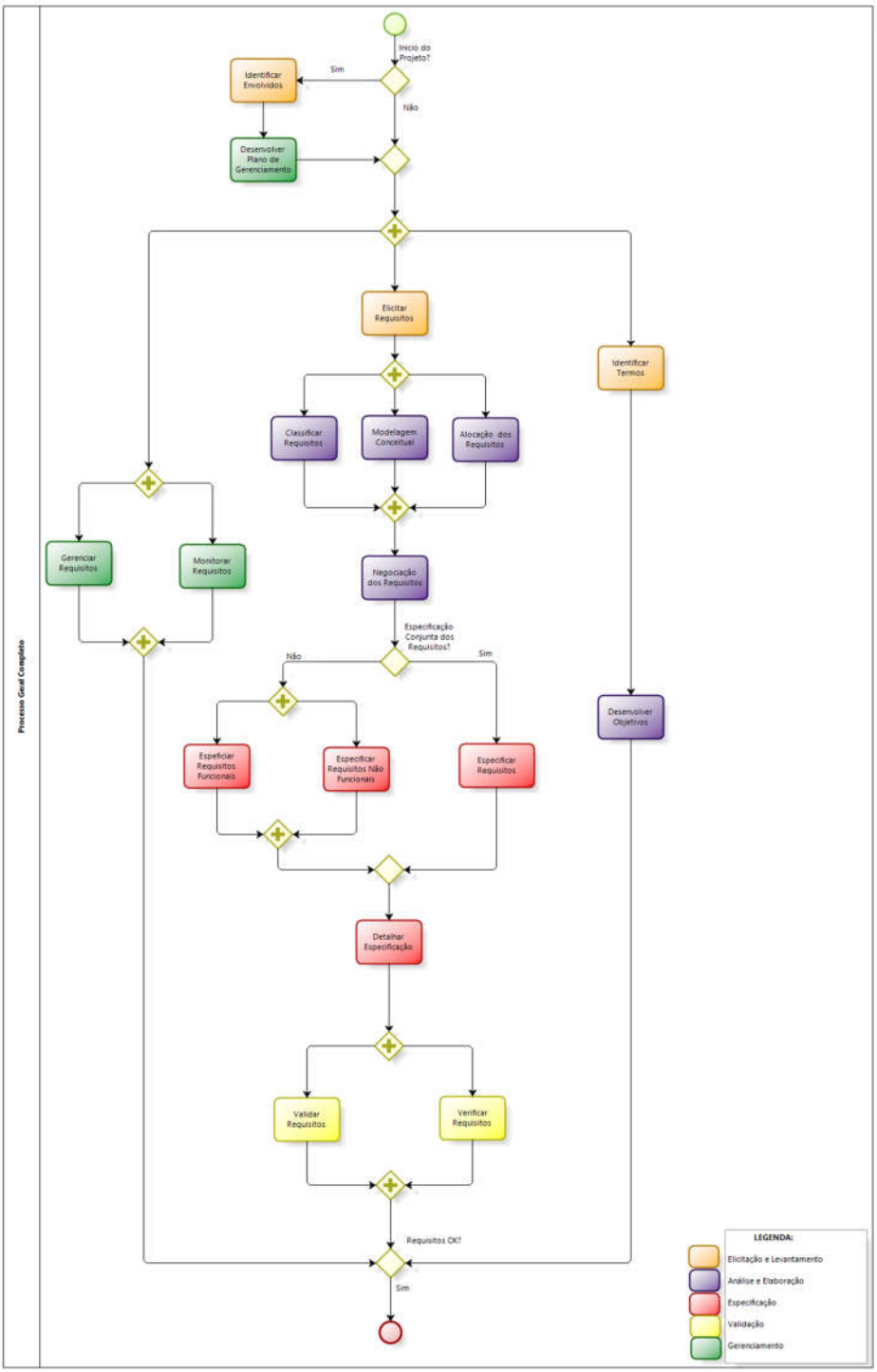
	Classificação dos Requisitos	Analisa do Sistema	SWEBOK	Os requisitos podem ser classificados em várias dimensões. Exemplos incluem o seguinte: se o requisito é funcional ou não funcional, se o requisito é derivado de um ou mais requisitos de alto nível ou uma propriedade emergente ou está sendo imposta diretamente no software por um stakeholder ou alguma outra fonte, se o requisito é Sobre o produto ou o processo.
	Modelagem Conceitual	Analisa do Sistema	SWEBOK	O desenvolvimento de modelos de um problema do mundo real é a chave para a análise de requisitos de software. Seu objetivo é ajudar a compreender a situação em que o problema ocorre, bem como descrever uma solução. Assim, modelos conceituais compreendem modelos de entidades do domínio do problema, confundidos para refletir suas relações e dependências do mundo real.
	Design da Arquitetura e Alocação dos Requisitos	Analisa do Sistema	SWEBOK	Em algum momento, a arquitetura da solução deve ser derivada. O projeto arquitetônico é o ponto no qual o processo de requisitos se sobrepõe com o design de software ou sistemas e ilustra como é impossível separar de forma limpa as duas tarefas. Em muitos casos, o engenheiro de software atua como arquiteto de software porque o processo de análise e elaboração dos requisitos exige que sejam identificados os componentes de arquitetura / design que serão responsáveis pela satisfação dos requisitos. Essa é a alocação de requisitos - a atribuição aos componentes da arquitetura responsáveis pela satisfação dos requisitos.
	Alocar Requisitos de Componente de Produto	Analisa do Sistema	CMMI	Os requisitos para componentes de produto da solução definida incluem: alocação de desempenho de produto; restrições de design; adequação, forma e função para satisfazer aos requisitos e facilitar a produção. Nos casos em que requisitos de alto nível especificam execução cuja responsabilidade deva estar distribuída em dois ou mais componentes de produto, essa execução deve ser particionada em uma alocação para cada componente de produto na forma de um requisito derivado.
Especificação	Estabelecer Conceitos Operacionais e Cenários	Especificador de Requisitos	CMMI	Os conceitos operacionais e os cenários são aprimorados para facilitar a seleção das soluções de componentes de produto que, quando implementadas, irão satisfazer ao uso pretendido do produto. Os conceitos operacionais e os cenários documentam a interação dos componentes de produto com ambiente, com usuários e com outros componentes de produto, independentemente da disciplina de Engenharia. Recomenda-se que eles sejam documentados para todos os modos e estados das operações, desde a implantação do produto até sua descontinuação, passando pela entrega, suporte (incluindo manutenção e sustentação) e treinamento.
	Especificação dos Requisitos	Especificador de Requisitos	SOMMERVILLE	Descrever o requisitos do usuário e de sistema em um documento de requisitos.
	Detalhar Casos de Uso	Analisa do Sistema	RUP	Esta tarefa é onde os detalhes são adicionados a um caso de uso específico.
	Priorizar Casos de Uso	Arquiteto de Software	RUP	Esta tarefa é onde os casos de uso são priorizados, de modo que sua ordem de desenvolvimento pode ser decidida. Essa tarefa é onde os casos de uso com

				significância arquitetural são identificados e priorizados.
	Desenvolver Especificações Suplementares.	Arquiteto de Software	RUP	Esta tarefa captura os requisitos que não se aplicam a casos de uso específicos.
	Detalhar os Requisitos de Software	Analisa do Sistema	RUP	Esta tarefa descreve como detalhar os requisitos de software do sistema. O Plano de Gerenciamento de Requisitos define a localização e organização dos requisitos, portanto, afeta diretamente como e onde os requisitos são detalhados.
	Especificação dos Requisitos.	Especificador de Requisitos	ISO 12207-2008	O uso específico do sistema a ser desenvolvido deve ser analisado para especificar os requisitos do sistema. A especificação de requisitos do sistema deve descrever: funções e capacidades do sistema; Necessidades de negócios, organizacionais e de usuários; Segurança, engenharia de fatores humanos (ergonomia), interface, operações e requisitos de manutenção; Restrições de design e requisitos de qualificação. A especificação dos requisitos do sistema deve ser documentada
	Estrutura o Modelo de Casos de Uso	Analisa do Sistema	RUP	Esta tarefa é onde o modelo de caso de uso é estruturado para tornar os requisitos mais fáceis de entender e manter. Isso inclui alavancar a uniformidade entre casos de uso e atores, e identificar comportamentos opcionais e excepcionais.
	Encontrar Atores e Casos de Uso	Analisa do Sistema	RUP	Esta tarefa é onde os atores e casos de uso são identificados para apoiar os requisitos que estão sendo implementados. Identificar os atores e casos de uso define explicitamente o escopo do sistema.
	Especificação dos Requisitos de Software	Especificador de Requisitos	SWEBOK	Especificações de requisitos de software estabelece a base para o acordo entre clientes e fornecedores ou fornecedores (em projetos orientados pelo mercado, esses papéis podem ser desempenhados pelas divisões de marketing e desenvolvimento) sobre o que o produto de software deve fazer, bem como o que não é esperado Faz. A especificação de requisitos de software permite uma avaliação rigorosa dos requisitos antes do projeto começar e reduz o redesenho posterior. Deve também fornecer uma base realista para estimar os custos, riscos e cronogramas do produto.
	Análise dos Requisitos de Software	Especificador de Requisitos	ISO 12207-2008	O implementador deve estabelecer e documentar os requisitos de software (incluindo as especificações de características de qualidade), os resultados das avaliações devem ser documentados.
Validação	Validar Requisitos	Revisor Técnico	CMMI	Os requisitos são validados para aumentar a probabilidade de que o produto resultante funcione conforme pretendido no ambiente de uso.
	Validação de requisitos	Analista do Sistema	PRESSMAN	Verificação para que o modelo de requisitos reflita de maneira precisa as necessidades do envolvido e forneça uma base sólida para o projeto.
	Validação de Requisitos	Analista do Sistema	SOMMERVILLE	Essa atividade verifica se os requisitos definem o sistema que o cliente realmente quer. Desta forma, está preocupada em encontrar problemas com os requisitos.
	Revisão dos Requisitos	Analista do Sistema	SWEBOK	Talvez o meio mais comum de verificação seja a inspeção ou revisão do(s) documento(s) de requisitos. A um grupo de revisores é atribuído para procurar erros, suposições erradas, falta de clareza e desvio da prática padrão. A

				composição do grupo que conduz a revisão é importante (pelo menos um representante do cliente deve ser incluído para um projeto orientado pelo cliente, por exemplo), e pode ajudar a fornecer orientação sobre o que procurar na forma de listas de verificação.
	Prototipação	Gestor de Testes	SWEBOK	A prototipagem é geralmente um meio para validar a interpretação do engenheiro de software dos requisitos de software, bem como para obter novos requisitos. Tal como acontece com a elicitação, há uma gama de técnicas de prototipagem e um número de pontos no processo onde a validação do protótipo pode ser apropriado.
	Modelo de Validação	Analisa do Sistema	SWEBOK	Normalmente é necessário validar a qualidade dos modelos desenvolvidos durante a análise. Por exemplo, em modelos de objetos, é útil realizar uma análise estática para verificar se existem caminhos de comunicação entre objetos que, no domínio dos stakeholders, trocam dados. Se forem usadas noções de análise formal, é possível usar o raciocínio formal para provar propriedades de especificação
	Testes de Aceitação	Gestor de Testes	SWEBOK	Uma propriedade essencial de um requisito de software é que deve ser possível validar que o produto acabado satisfaz. Requisitos que não podem ser validados são realmente apenas "desejos". Uma tarefa importante é, portanto, planejamento de como verificar cada requisito. Na maioria dos casos, a concepção de testes de aceitação faz isso para como os usuários finais costumam fazer negócios usando o sistema
	Revisão dos Requisitos	Revisor Técnico	RUP	Esta tarefa descreve como rever os requisitos de produtos de trabalho.
	Avaliação dos Requisitos	Revisor Técnico	ISO 12207-2008	O projeto analisará o conjunto completo de requisitos elicitados. Os resultados das avaliações devem ser documentados.
Gerenciamento	Gerenciar Requisitos	Analisa do Sistema	CMMI	Os requisitos são gerenciados e as inconsistências são identificadas em relação aos planos de projeto e produtos de trabalho.
	Monitoramento de requisitos	Revisor de Gerenciamento	PRESSMAN	O monitoramento de requisitos dá suporte à validação continua por analisar modelos de meta do usuário em relação ao sistema em uso.
	Gerenciamento de Requisitos	Analisa do Sistema	SOMMERVILLE	Essa atividade realiza o processo de compreensão e controle das mudanças nos requisitos do sistema.
	Gerenciamento de Mudanças	Management Reviewer	SWEBOK	O gerenciamento de mudanças é central para o gerenciamento de requisitos. Este tópico descreve o papel do gerenciamento de mudanças, os procedimentos que precisam ser implementados e a análise que deve ser aplicada às alterações propostas.
	Atributos dos Requisitos	Analisa do Sistema	SWEBOK	Os requisitos devem consistir não apenas em uma especificação do que é necessário, mas também em informações auxiliares, o que ajuda a gerenciar e interpretar os requisitos. Os atributos de requisitos devem ser definidos, registrados e atualizados à medida que o software em desenvolvimento ou manutenção evolui. Isto deve incluir as várias dimensões de classificação do requisito e o método de verificação ou a seção de plano de teste de aceitação

				relevante. Ele também pode incluir informações adicionais, como um resumo de razão para cada requisito, a fonte de cada requisito e um histórico de alterações. O atributo de requisitos mais importante, no entanto, é um identificador que permite que os requisitos sejam identificados de forma única e inequívoca.
	Rastrear Requisitos	Analisa do Sistema	SWEBOK	O rastreamento de requisitos tem a ver com a recuperação da fonte dos requisitos e a previsão dos efeitos dos requisitos. O rastreamento é fundamental para a realização da análise de impacto quando os requisitos mudam. Um requisito deve ser rastreável para trás para os requisitos e partes interessadas que o motivou (de um requisito de software de volta para o sistema requisitos que ajuda a satisfazer, por exemplo). Por outro lado, um requisito deve ser rastreável para a frente nos requisitos e entidades de design que o satisfaçam (por exemplo, a partir de um requisito do sistema para os requisitos de software que foram elaborados a partir dele, e sobre os módulos de código que implementá-lo, ou os casos de teste
	Medir Requisitos	Revisor de Gerenciamento	SWEBOK	Como uma questão prática, é tipicamente útil ter algum conceito do "volume" dos requisitos para um determinado produto de software. Esse número é útil na avaliação do "tamanho" de uma mudança de requisitos, na estimativa do custo de uma tarefa de desenvolvimento ou manutenção ou simplesmente para uso como denominador em outras medições. A medição do tamanho funcional (FSM) é uma técnica para avaliar o tamanho de um corpo de requisitos funcionais.
	Gerenciar Dependências	Analisa do Sistema	RUP	Esta tarefa descreve como usar as dependências entre requisitos para gerenciar o escopo do projeto, bem como as mudanças de requisitos em si.
	Desenvolver Plano de Gerenciamento de Requisitos	Analisa do Sistema	RUP	Esta tarefa descreve como desenvolver um plano para documentar requisitos, seus atributos e diretrizes para rastreabilidade e gerenciamento de requisitos do produto.
	Gerenciar Solicitações de Mudanças	Analisa do Sistema	RUP	Esta atividade garante que a devida consideração é dada ao impacto da mudança no projeto e que as alterações aprovadas são feitas dentro de um projeto de forma consistente.

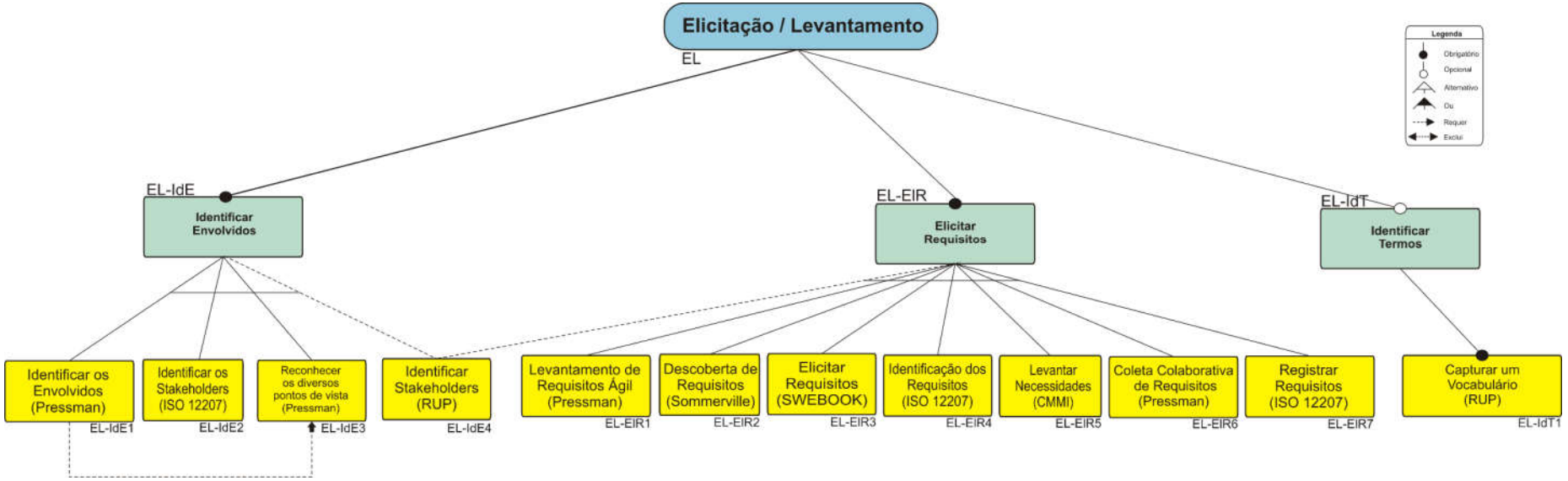
APÊNDICE B – SEQUENCIAMENTO DAS ATIVIDADES ABSTRATAS DOS PROCESSO.



Fonte: Autor.

APÊNDICE C – MODELOS DE VARIABILIDADES E SUAS RESPECTIVAS REGRAS

1) Etapa: Elicitação e Levantamento de Requisitos e Regras de Features



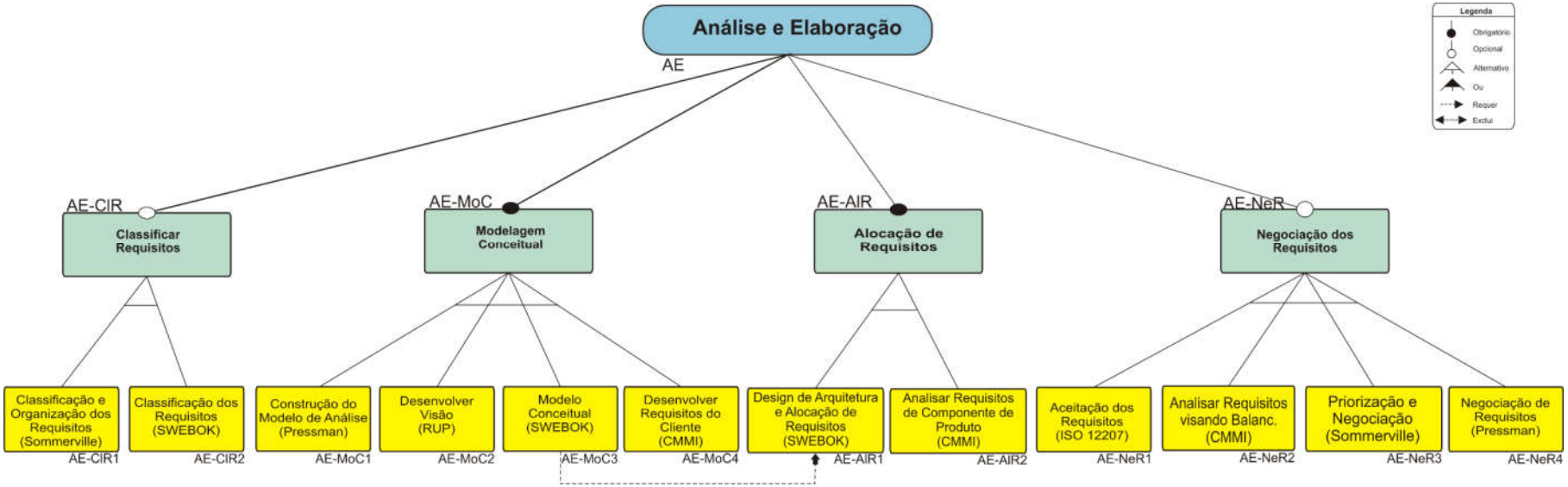
Elicitação e Levantamento ↔ ((EL-IdE1 → EL-IdE3 ↔ (¬ EL-IdE2 ∧ ¬ EL-IdE4 ∧ EL-IdE) ∧ ((Identificação dos Stakeholders (ISO ↔ (¬EL-IdE1 ∧ ¬ EL-IdE3 ∧ ¬ EL-IdE4 ∧ EL-IdE) ∧ ((EL-IdE3 ↔ (¬ EL-IdE2 ∧ ¬ EL-IdE1 ∧ ¬ EL-IdE4 ∧ EL-IdE) ∧ ((EL-IdE4 ↔ (¬ EL-IdE2 ∧ ¬ EL-IdE3 ∧ ¬ EL-IdE1 ∧ EL-IdE))

Elicitação e Levantamento ↔ ((EL-EIR1 ↔ (¬EL-EIR2 ∧ ¬ EL-EIR3 ∧ ¬ EL-EIR4 ∧ ¬ EL-EIR5 ∧ ¬ EL-EIR6 ∧ ¬ EL-EIR7 ∧ EL-EIR)) ∧ ((EL-EIR2 ↔ (¬EL-EIR1 ∧ ¬ EL-EIR3 ∧ ¬ EL-EIR4 ∧ ¬ EL-EIR5 ∧ ¬ EL-EIR6 ∧ ¬ EL-EIR7 ∧ EL-EIR)) ∧ ((EL-EIR3 ↔ (¬ EL-EIR2 ∧ ¬ EL-EIR1 ∧ ¬ EL-EIR4 ∧ ¬ EL-EIR5 ∧ ¬ EL-EIR6 ∧ ¬ EL-EIR7 ∧ EL-EIR)) ∧ ((EL-EIR4 ↔ (¬ EL-EIR2 ∧ ¬ EL-EIR3 ∧ ¬ EL-EIR1 ∧ ¬ EL-EIR5 ∧ ¬ EL-EIR6 ∧ ¬ EL-EIR7 ∧ EL-EIR)) ∧ ((EL-EIR5 ↔ (¬ EL-EIR2 ∧ ¬ EL-EIR3 ∧ ¬ EL-EIR4 ∧ ¬ EL-EIR1 ∧ ¬ EL-EIR6 ∧ ¬ EL-EIR7 ∧ EL-EIR)) ∧ ((EL-EIR6 ↔ (¬ EL-EIR2 ∧ ¬ EL-EIR3 ∧ ¬ EL-

ElR4 ∧ ¬ EL-ElR5 ∧ ¬ EL-ElR1 ∧ ¬ EL-ElR7 ∧ EL-ElR)) ∧ ((EL-ElR7 ↔ (¬ EL-ElR2 ∧ ¬ EL-ElR3 ∧ ¬ EL-ElR4 ∧ ¬ EL-ElR5 ∧ ¬ EL-ElR6 ∧ ¬ EL-ElR1 ∧ EL-ElR)))

Elicitação e Levantamento → EL-IdT → EL-IdT1

2) Etapa: Análise e Elaboração de Requisitos



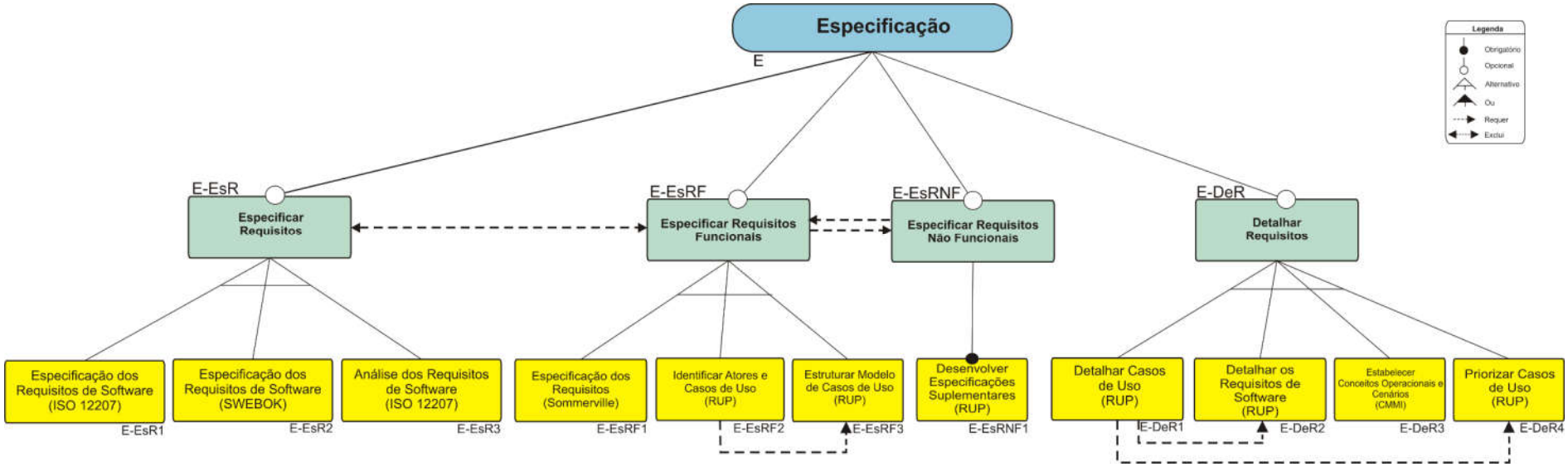
Análise e Elaboração → ((AE-ClR1 ↔ (¬AE-ClR2 ∧ AE-ClR))∧ ((AE-ClR2 ↔ (¬AE-ClR1 ∧ AE-ClR)))

Análise e Elaboração ↔ ((AE-MoC1 ↔ (¬AE-MoC2 ∧ ¬(AE-MoC3 →(AE-AlR1) ∧ ¬AE-MoC4 ∧ AE-MoC)) ∧ ((AE-MoC2 ↔ (¬AE-MoC1∧ ¬(AE-MoC3 →(AE-AlR1) ∧ ¬AE-MoC4 ∧ AE-MoC)) ∧ ((AE-MoC3 →(AE-AlR1) ↔ (¬AE-MoC2 ∧ ¬ AE-MoC1 ∧ ¬AE-MoC4 ∧ AE-MoC)) ∧ ((AE-MoC4 ↔ (¬AE-MoC2 ∧ ¬(AE-MoC3 →(AE-AlR1) ∧ ¬ AE-MoC1 ∧ AE-MoC)))

Análise e Elaboração ↔ ((AE-AlR1 ↔ (¬AE-AlR2 ∧ AE-AlR)) ∧ ((AE-AlR2 ↔ (¬AE-AlR1 ∧ AE-AlR)))

Análise e Elaboração ↔ AE-NeR → ((AE-NeR1 ↔ (¬AE-NeR2 ∧ ¬AE-NeR3 ∧ ¬AE-NeR4 ∧ AE-NeR) ∧ ((AE-NeR2 ↔ (¬AE-NeR1 ∧ ¬AE-NeR3 ∧ ¬AE-NeR4 ∧ AE-NeR) ∧ ((AE-NeR3 ↔ (¬AE-NeR2 ∧ ¬ AE-NeR1 ∧ ¬AE-NeR4 ∧ AE-NeR) ∧ ((AE-NeR4↔ (¬AE-NeR2 ∧ ¬AE-NeR3 ∧ ¬ AE-NeR1 ∧ AE-NeR))

3) Etapa: Especificação



Especificação → ¬(E-EsR ∧ (E-EsRF → E-EsRNF))

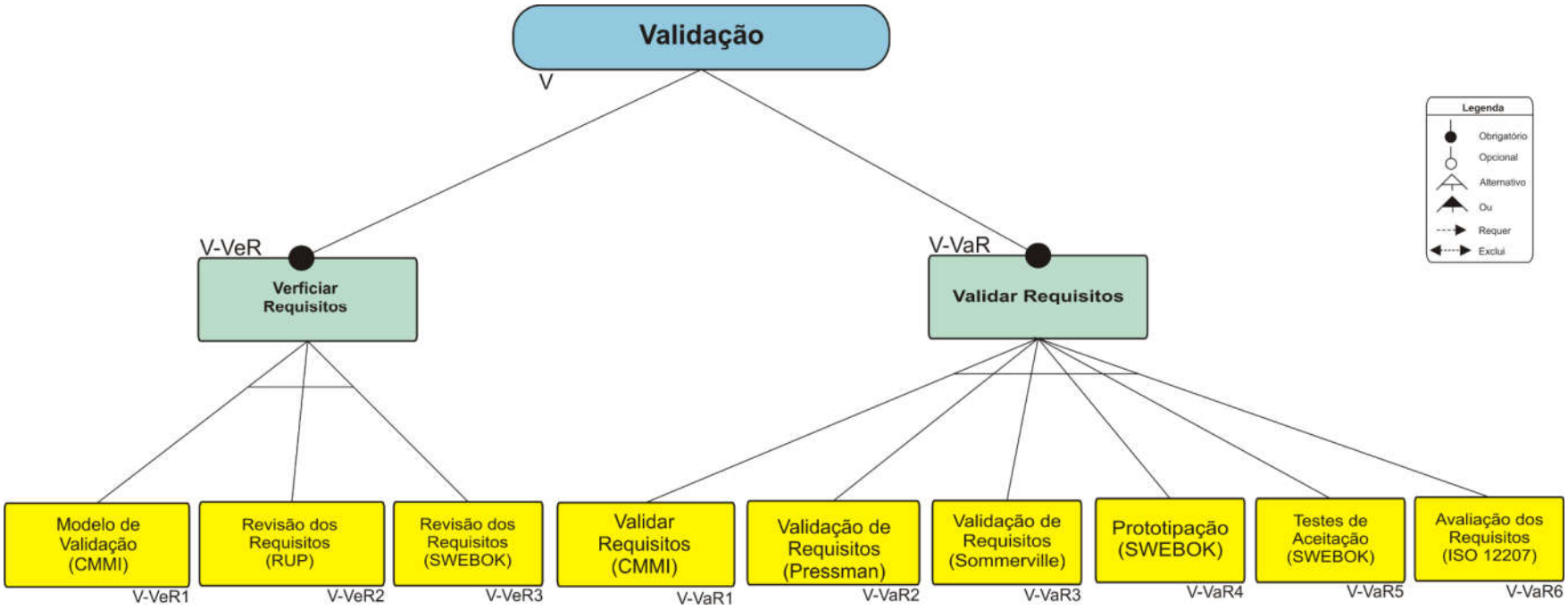
Especificação → (E-EsR1 ↔(¬E-EsR2 ∧¬ E-EsR3 ∧ E-EsR)) ∧ (E-EsR2↔(¬ E-EsR1 ∧¬ E-EsR3 ∧ E-EsR)) ∧ (E-EsR3 ↔(¬E-EsR2 ∧¬ E-EsR1 ∧ E-EsR)))

Especificação → (E-EsRF1 ↔((¬E-EsRF2 → E-EsRF3) ∧¬ E-EsRF3 ∧ E-EsRF)) ∧ (E-EsRF2 → E-EsRF3 ↔(E-EsRF1 ∧ E-EsRF)) ∧ (E-EsRF3 ↔(¬E-EsRF2 ∧¬ E-EsRF1 ∧ E-EsRF)))

Especificação → E-EsRNF ↔ E-EsRNF1

Especificação → (E-DeR1 → E-DeR2 → E-DeR4 ↔((¬E-DeR3 ∧ E-DeR) ∧ (E-DeR2 ↔((¬E-DeR3 ∧¬E-DeR1 ∧¬ E-DeR4 ∧ E-DeR) ∧ (E-DeR3↔((¬ E-DeR2 ∧¬E-DeR1 ∧¬ E-DeR4 ∧ E-DeR) ∧ (E-DeR4 ↔((¬ E-DeR2 ∧¬E-DeR1 ∧¬ E-DeR3 ∧ E-DeR)))

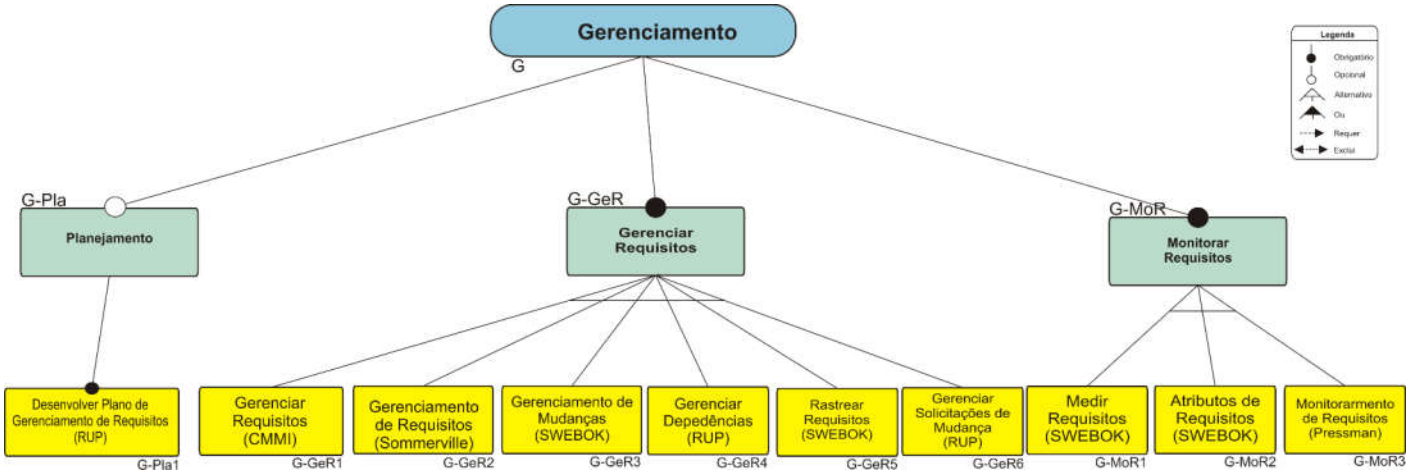
4) Etapa: Validação



Validação ↔ (V-VeR1(CMMI) ↔ (¬V-VeR2 (RUP) ∧ ¬V-Ver3(SW) ∧ V-VeR)) ∧ (V-VeR2(RUP) ↔ (¬V-VeR1 (CMMI) ∧ ¬ V-VeR3(SW) ∧ ∧ V-VeR)) ∧ (V-VeR3 (SW) ↔ (¬V-VeR1(CMMI) ∧ ¬ V-VeR2(RUP) ∧ V-VeR))

Validação ↔ (V-ValR1(CMMI) ↔ (¬V-ValR2(Pr) ∧ ¬ V-ValR3(Som) ∧ ¬ V-ValR4(SW) ∧ ¬ V-ValR5(SW) ∧ ¬ V-ValR6(ISO) ∧ V-ValR)) ∧ (V-ValR2(Pr) ↔ (¬V-ValR1(CMMI) ∧ ¬ V-ValR3(Som) ∧ ¬ V-ValR4(SW) ∧ ¬ V-ValR5(SW) ∧ ¬ V-ValR6(ISO) ∧ V-ValR)) ∧ (V-ValR3(Som) ↔ (¬V-ValR2(Pr) ∧ ¬ V-ValR1(CMMI) ∧ ¬ V-ValR4(SW) ∧ ¬ V-ValR5(SW) ∧ ¬ V-ValR6(ISO) ∧ V-ValR)) ∧ (V-ValR4(SW) ↔ (¬V-ValR2(Pr) ∧ ¬ V-ValR3(Som) ∧ ¬ V-ValR1(CMMI) ∧ ¬ V-ValR5(SW) ∧ ¬ V-ValR6(ISO) ∧ V-ValR)) ∧ (V-ValR5(SW) ↔ (¬V-ValR2(Pr) ∧ ¬ V-ValR3(Som) ∧ ¬ V-ValR4(SW) ∧ ¬ V-ValR1(CMMI) ∧ ¬ V-ValR6(ISO) ∧ V-ValR)) ∧ (V-ValR6(ISO)) ↔ (¬V-ValR2(Pr) ∧ ¬ V-ValR3(Som) ∧ ¬ V-ValR4(SWI) ∧ ¬ V-ValR5(SW) ∧ ¬ V-ValR1(CMMI) ∧ V-ValR))

5) Etapa: Gerenciamento



Gerenciamento → G-Pla → G-Pla1

Gerenciamento ↔ (G-GeR1 ↔ (¬G-GeR2 ∧¬ G-GeR3 ∧ ¬G-GeR4 ∧ ¬G-GeR5 ∧ ¬G-GeR6 ∧ G-GeR)) ∧ (G-GeR2 ↔ (¬G-GeR1 ∧ ¬G-GeR3 ∧ ¬G-GeR4 ∧¬ G-GeR5 ∧ ¬G-GeR6 ∧ G-GeR)) ∧ (G-GeR3↔ (¬G-GeR2 ∧ ¬G-GeR1 ∧ ¬G-GeR4 ∧ ¬G-GeR5 ∧ ¬ G-GeR6 ∧ G-GeR)) ∧ (∧ G-GeR4 ↔ (¬G-GeR2 ∧ ¬G-GeR3 ∧ ¬G-GeR1 ∧¬ G-GeR5 ∧ ¬G-GeR6 ∧ G-GeR)) ∧ (G-GeR5 ↔ (¬G-GeR2 ∧ ¬G-GeR3 ∧¬ G-GeR4 ∧ ¬G-GeR1 ∧ ¬G-GeR6 ∧ G-GeR)) ∧ (G-GeR6 ↔ (¬G-GeR2 ∧ ¬G-GeR3 ∧ ¬G-GeR4 ∧ ¬G-GeR5 ∧ ¬G-GeR1 ∧ G-GeR))

Gerenciamento ↔ (G-MoR1 ↔ (¬G-MoR2 ∧ ¬G-MoR3 ∧ G-MoR)) ∧ (G-MoR2 ↔ (¬G-MoR1 ∧ ¬G-MoR3 ∧ G-MoR)) ∧ G-MoR3 ↔ (¬G-MoR2 ∧ ¬G-MoR1 ∧ G-MoR