**UNIVERSIDADE FEDERAL DE SANTA MARIA**
**CENTRO DE TECNOLOGIA**
**PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA**

# UMA ESTRATÉGIA PARA VALIDAÇÃO DA COMPLETUDE E CONSISTÊNCIA EM PROCESSOS DE SOFTWARE

**DISSERTAÇÃO DE MESTRADO**

**Miguel Augusto Bauermann Brasil**

**Santa Maria, RS, Brasil**

**2014**

# UMA ESTRATÉGIA PARA VALIDAÇÃO DA COMPLETUDE E CONSISTÊNCIA EM PROCESSOS DE SOFTWARE

**Miguel Augusto Bauermann Brasil**

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

**Orientadora: Profª. Drª. Lisandra Manzoni Fontoura**

**Santa Maria, RS, Brasil**

**2014**

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

Brasil, Miguel Augusto Bauermann
Uma estratégia para validação da completude e consistência em processos de software / Miguel Augusto Bauermann Brasil.-2014.
143 p.; 30cm

Orientadora: Lisandra Manzoni Fontoura
Dissertação (mestrado) - Universidade Federal de Santa Maria, Centro de Tecnologia, Programa de Pós-Graduação em Informática, RS, 2014

1. Validação da consistência e completude em fragmentos de processo 2. Adaptação de processos 3. Melhoria da qualidade em processos de software I. Fontoura, Lisandra Manzoni II. Título.


E-mail: miguel.bauermann@gmail.com

**UFSM – UNIVERSIDADE FEDERAL DE SANTA MARIA**
**CENTRO DE TECNOLOGIA**
**PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA**

A Comissão Examinadora, abaixo assinada,
aprova a Dissertação de Mestrado

# UMA ESTRATÉGIA PARA VALIDAÇÃO DA COMPLETUDE E CONSISTÊNCIA EM PROCESSOS DE SOFTWARE

elaborada por
**Miguel Augusto Bauermann Brasil**

como requisito parcial de obtenção de grau de
**Mestre em Ciência da Computação**

**COMISSÃO EXAMINADORA:**

Lisandra Fontoura

**Lisandra Manzoni Fontoura, Dra.**
(Presidente/Orientadora)

**Giovani Rubert Librelotto, Dr. (UFSM)**

Rejane Frozza

**Rejane Frozza, Dr. (UNISC)**

**Santa Maria, 19 de Agosto de 2014.**

***Dedico este trabalho a minha esposa Caroline de Moraes Trevisan Brasil, ao meu amado filho Angelo Trevisan Brasil, aos meus pais Pedro Assis da Silva Brasil e Marilda Bauermann Brasil e a minha irmã Carla Cristina Bauermann Brasil, por todo amor, carinho, compreensão, ajuda e incentivo, sem os quais esta conquista seria muito árdua e penosa.***

# AGRADECIMENTOS

A realização deste trabalho foi possível graças:

A Deus, por ter me dado vida e saúde ao longo desta caminhada, guiando-me sempre para transpor os obstáculos do mundo de maneira integra e honrosa.

A minha orientadora, Prof. Dra. Lisandra Manzoni Fontoura, por ter acreditado em minha capacidade aceitando-me como orientando no Programa de Pós Graduação em Informática, pela presteza e dedicação ao me orientar neste trabalho e por todo o conhecimento compartilhado.

Aos meus pais, Pedro Assis da Silva Brasil e Marilda Bauermann Brasil, e a minha irmã, Carla Cristina Bauermann Brasil, pela paciência, ajuda, pelo total apoio e motivação e pelo contínuo incentivo ao longo de minha vida.

A minha esposa Caroline de Moraes Trevisan Brasil pelo total apoio, companheirismo, paciência e compreensão ao longo deste percurso e por ter me dado uma grande motivação durante o mestrado, nosso filho Angelo Trevisan Brasil.

Ao meu filho Angelo Trevisan Brasil, meu tesouro mais precioso, por mesmo sem perceber, doar o seu tempo de brincadeiras com o papai, para que este sonho se concretizasse. Filho, tuas idas engatinhando ao escritório enquanto escrevia esta dissertação, sem dúvida, me deram ânimo e coragem. Te amo.

Aos meus colegas de mestrado e grandes amigos, Guilherme Vaz Pereira, Wagner Gadêa Lorenz e Ruan Bonilha Pozzebon que me instruíram, apoiaram, orientaram e levantaram minha moral nos momentos de dificuldade. Aprendi muito com vocês.

Aos colegas e amigos do Instituto Federal Farroupilha, Campus São Vicente do Sul, Daniel Boemo, Eliana Zen, Henrique Tamiosso Machado, Rogério Cassanta Rosado, Maicon Amarante, Frederico Andres Bazana e tantos outros que de uma forma ou outra me apoiaram e incentivaram a alcançar este objetivo.

Aos novos colegas do Colégio Técnico Industrial (CTISM), por acreditarem na m capacidade me dando força e apoio. A todos, meu muito obrigado.

*No entorno em que vivo*
*E nas soluções dos meus problemas*
*Aprendo com eles*
*E com os dos outros...*
*Em qualquer tema*
*Encontro-me,*
*Desencontro-me...*
*E neste jogo,*
*Retorno e torno-me o homem que devo ser.*

*− Gilnei Lopes*

# RESUMO

Dissertação de Mestrado
Programa de Pós-Graduação em Informática
Universidade Federal de Santa Maria

## UMA ESTRATÉGIA PARA VALIDAÇÃO DA COMPLETUDE E CONSISTÊNCIA EM PROCESSOS DE SOFTWARE

AUTOR: Miguel Augusto Bauermann Brasil
ORIENTADORA: Profª. Drª. Lisandra Manzoni Fontoura
Local e data da defesa: Santa Maria, 19 de Agosto de 2014.

Não existe um modelo de processo de desenvolvimento único para ser adotado para todos os projetos de software. Normas e modelos de qualidade como a norma ISO/IEC 15504, MPS.BR, CMM e o CMMI preconizam que a adaptação de processos seja realizada para satisfazer às necessidades específicas dos projetos. Entretanto, a atividade de adaptar um processo de software é considerada uma tarefa complexa, exigindo conhecimento e experiência de quem a realiza. A seleção de elementos de processo incompletos, ou duplicados podem gerar ambiguidades que podem comprometer o andamento do projeto e gerar desconfiança para com o processo adaptado. Esta dissertação apresenta uma estratégia sistemática para validação da completude e consistência interna dos elementos formadores do processo adaptado, neste trabalho chamados fragmentos. O objetivo é contribuir para a melhoria da qualidade dos processos de desenvolvimento de software adaptados e auxiliar o engenheiro de processos na tarefa de adaptação de processos, fornecendo elementos de processo completos, consistentes e priorizados de acordo com as características do projeto. Para apoiar a proposta, foram desenvolvidas: i) uma ontologia para reconhecimento da similaridade em processos; ii) um metamodelo para adaptação de processos e iii) uma ferramenta web para definição de processos completos e consistentes. A estratégia proposta facilita o trabalho do engenheiro de processos informando para este quais elementos são adequados (completos e consistentes), e possibilita a eliminação de inconsistências levando a melhoria do processo.

**Palavras-chave:** Validação da consistência e completude em fragmentos de processo. Adaptação de processos. Melhoria da qualidade em processos de software.

# ABSTRACT

Master's Dissertation
Post-Graduation Program in Information Science
Federal University of Santa Maria

## A STRATEGY FOR VALIDATION OF COMPLETENESS AND CONSISTENCY IN SOFTWARE PROCESSES

AUTHOR: Miguel Augusto Bauermann Brasil
ADVISOR: Prof. Dra. Lisandra Manzoni Fontoura
Place and date: Santa Maria, 19 August, 2014.

There isn't a unique development process suitable for all software projects. Standards and quality models such as ISO/IEC 15504, MPS.BR, CMM and CMMI, recommend the process tailoring to satisfy specific project features. However process tailoring is a complex task because it requires knowledge and expertise of who performs. The incomplete or duplicate process elements selection can generate ambiguities which may disturb the project progress and generate distrust in relation to the tailored process. This dissertation presents a systematic strategy to completeness and internal consistency validation of the elements that are part of the tailored process, call fragments. The aim is to contribute to improving the software development process quality and help the process engineer on the process tailoring task, providing complete and consistent process elements which are prioritized according to the project features. To support the proposed strategy have benn developed: i) ontology to similarity recognition among process; ii) a metamodel for process tailoring; iii) a web toll for complete and consistent process definition. The proposed strategy facilitates the work of the engineer showing which elements are adequate (complete and consistent) to be part of the tailored process and enables the elimination of inconsistences lead to improving the process.

**Keywords:** Validate consistency and completeness in process fragments. Process Tailoring. Quality improvement in software process.

# LISTA DE FIGURAS

# LISTA DE TABELAS

# LISTA DE ABREVIATURAS E SIGLAS

| | | |
|---|---|---|
| AHP | - | *Analytic Hierarchy Process* |
| BPMN | - | *Business Process Modeling Notation* |
| CMM | - | *Capability Maturity Model* |
| CMMI | - | *Capability Maturity Model Integration* |
| CSS3 | - | *Cascading Style Sheets (Version 3)* |
| HTML | - | *HyperText Markup Language* |
| JPA | - | *Java Persistence API* |
| JSF | - | *Java Server Faces* |
| MfTP | - | *Metamodel for Tailoring Process* |
| MPS.BR | - | *Melhoria de Processos de Software Brasileiro* |
| OCTOPUS | - | *Octopus Model* |
| OntoSME | - | *Ontology for Situational Method Engineering* |
| OPF | - | *Open Process Framework* |
| OSPTA | - | *Octopus SME Process Tailoring Approach* |
| OWL | - | *Ontology Web Language* |
| PRiMA | - | *Project Risk Management Approach* |
| QaVaS | - | *Quality Validate Strategy* |
| RUP | - | *Rational Unified Process* |
| SAI | - | *Semantic Affinity of Intentions* |
| SAS | - | *Semantic Affinity of Sections* |
| SEI | - | *Software Engineering Institute* |
| SME | - | *Situational Method Engineering* |
| SPEM | - | *Software Process Engineering Metamodel* |
| SPO | - | *Standart Process Organization* |
| SPrL | - | *Software Process Line* |
| SQL | - | *Structured Query Language* |
| SQWRL | - | *Semantic Query-Enhanced Web Rule Language* |
| SSI | - | *Structural Similarity by Intensions* |
| SSS | - | *Structural Similarity by Sections* |
| STIC | - | *Setor de Tecnologia da Informação e Comunicação* |
| SWRL | - | *Semantic Web Rule Language* |
| XP | - | *Extreme Program* |

# SUMÁRIO

# INTRODUÇÃO

Definir um processo de software não é uma tarefa simples uma vez que exige conhecimento e experiência dos profissionais que a realizam (BARRETO; MURTA; ROCHA, 2011). Um processo de software definido possibilita a melhoria da qualidade dos produtos gerados, diminui o retrabalho, aumenta a produtividade da equipe e proporciona uma maior competividade da empresa (BRASIL; FONTOURA; ALVARO, 2013).

No entanto, independente do processo de software que a organização adote, não é adequado que este seja usado em tua totalidade sem antes sofrer alguma adaptação (KALUS; KUHRMANN, 2013). Normas e modelos de qualidade como a norma ISO/IEC 15504 (ISO/IEC 15504, 1998), MPS.BR (SOFTEX, 2012) , CMM (PAULK et al., 1993) e o CMMI (SEI, 2010) vão ao encontro desta afirmação preconizando que seja feita a atividade de adaptação de processos para atender às necessidades específicas de cada projeto.

Embora a adaptação de processos seja necessária e amplamente aceita, a maneira de fazer esta adaptação permanece obscura e muitas vezes são deixadas sob responsabilidade dos engenheiros de processo e do conhecimento que estes adquiriram ao longo do tempo (KALUS; KUHRMANN, 2013). Adaptar um processo de software envolve adicionar, excluir e/ou modificar os elementos que compõem o referido processo e seus relacionamentos com objetivo de torná-lo mais adequado às metas especificadas para o projeto (YOON; MIN; BAE, 2001).

Segundo Harmsen, Brinkkemper e Oei (1994), uma das causas do insucesso na execução dos projetos de software é devido a informações incompletas e/ou incoerentes nos elementos e/ou relacionamentos contidos no processo adotado para o projeto. Estes elementos incompletos e incoerentes, quando não detectados a tempo podem ocasionar insucesso do projeto.

Dada esta complexidade, observa-se que as abordagens de criação ou adaptação de processos por mais sistemáticas que sejam ainda deixam margem para seleção de elementos de processos incompletos ou inconsistentes podendo gerar possíveis ambiguidades. Tais ambiguidades por sua vez, podem motivar o descomprometimento com o processo adaptado e trazer consequências negativas para organização como desperdício de tempo e dinheiro (MAGDALENO, 2010).

Diante do exposto, o problema definido para nortear esta pesquisa é: Como possibilitar que processos definidos para satisfazer as características específicas de um projeto sejam formados por elementos de processo completos e consistentes?

Acredita-se que a definição de uma estratégia voltada à garantia da completude e da consistência dos elementos formadores do processo adaptado possa tornar estes elementos mais adequados às características do projeto e, assim, melhorar a qualidade do processo definido, uma vez que este processo será formado apenas de elementos de processo completos e consistentes.

Esta dissertação propõe uma estratégia para validação da completude e consistência em processos de software denominada Estratégia de Validação da Qualidade (*Quality Validate Strategy* – QaVaS). Por meio desta estratégia objetiva-se validar a consistência e completude dos elementos de processo que formam o processo adaptado e assim contribuir para a melhoria da qualidade em processos de software definidos para projetos específicos da organização.

Como principais contribuições têm-se: i) uma ontologia para identificação da similaridade semântica e estrutural de elementos de processo, nesta dissertação, chamados de fragmentos de processo; ii) um metamodelo para construção ou adaptação de processos voltado para garantia da consistência em elementos de processo; iii) métricas para avaliar o grau de completude interna de fragmentos de processo e uma estratégia para validação da completude; iv) regras para validação da consistência interna nos elementos que compõem os fragmentos; e v) regras de qualidade para construção de novos fragmentos.

## 1.1 *Objetivo geral*

Propor uma estratégia para validar a consistência e completude dos fragmentos que formam o processo adaptado e assim contribuir para a melhoria da qualidade em processos de software definidos para atender às expectativas dos projetos de uma organização.

## 1.2 *Objetivos específicos*

Para satisfazer o objetivo geral proposto, o mesmo pode ser desmembrado nos seguintes objetivos específicos:

i) Propor uma ontologia para identificação da similaridade semântica e estrutural dos fragmentos de processo, formalizando assim o conhecimento da área de processos de software adaptados a partir da abordagem *Situational Method Engineering*;

ii) Modelar um metamodelo para construção ou adaptação de processos de software completos e consistentes;

iii) Caracterizar regras de qualidade para construção de novos fragmentos de processo por meio da estratégia proposta;

iv) Atribuir itens para avaliar o grau de completude dos fragmentos de processo e uma estratégia para completa-los quando for o caso;

v) Formular uma estratégia para validação da consistência interna dos fragmentos, eliminando assim, duplicidades nos fragmentos de processo instanciados para atender às características específicas de um projeto;

vi) Implementar um protótipo de ferramenta que automatize a estratégia proposta.

## 1.3 *Estrutura da dissertação*

Este trabalho está organizado como segue:

O Capítulo 2 apresenta a base teórica da área de processos de software utilizada para embasar este trabalho.

O Capítulo 3 descreve a ontologia OntoSME, desenvolvida para proporcionar uma formalização dos elementos de processo modelados por meio da abordagem *Situational Method Engineering*.

No Capítulo 4 é apresentado o metamodelo *Metamodel for Tailoring Process* (MfTP) desenvolvido para permitir a instanciação de elementos de processo completos e consistentes e a utilização destes na definição de processos ágeis ou planejados de acordo com às caraterísticas específicas dos projetos.

No Capítulo 5 é descrita a estratégia proposta que tem como objetivo validar a completude e a consistência nos elementos de processos permitindo assim, uma melhoria

contínua não só dos elementos de processo armazenados na base de dados, mas dos processos construídos a partir destes elementos.

No Capítulo 6 é apresentada a ferramenta de apoio *Metamodel for Tailoring Process* Tool (MfTP), contendo o módulo principal e o módulo para validar a proposta deste trabalho.

No Capítulo 7 é apresentado o experimento real, realizado no Setor de Tecnologia de Informação e Comunicação do Instituto Federal Farroupilha, Campus São Vicente do Sul – Rio Grande do Sul.

No Capítulo 8 são apresentados os trabalhos relacionados e os seus comparativos com a estratégia proposta. Por fim, no Capítulo 9 são apresentadas as considerações finais e trabalhos futuros.

# 2 REFERENCIAL TEÓRICO

Este capítulo apresenta o referencial teórico necessário para o entendimento desta dissertação. Na Seção 2.1 são descritos os conceitos básicos sobre processos de software tradicionais e sobre adaptação de processos. Na Seção 2.2 são detalhados conceitos sobre *Situational Method Engineering*. Na Seção 2.3 é descrita a abordagem PRiMA utilizada para gerenciar riscos em projetos de software. Na Seção 2.4 é detalhada a abordagem OSPTA que propõem a adaptação de processos por meio da priorização de elementos de processo com base em características do projeto. Por sua vez, na Seção 2.5 são descritos conceitos referentes a ontologias.

## 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). Fuggetta (2000) define processos de software como um conjunto coerente de políticas, estruturas organizacionais, tecnologias, procedimentos e artefatos que são necessários para desenvolver, implantar e manter um produto de software.

No entanto, é de conhecimento que um processo necessita ser adaptado para que satisfaça às necessidades da organização ou projeto (HENDERSON-SELLERS; RALYTÉ, 2010). Esta adaptação é realizada para atender às diferentes situações (GUPTA; DWIVEDI, 2012). Segundo Aharoni e Reinhartz-Berger (2007), podem ser consideradas situações as características do projeto, como por exemplo, o cliente, a equipe de desenvolvimento, tipo de sistema ou tamanho da organização, entre outros fatores (TSUMAKI; TAMAI, 2006).

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). Como exemplos de alterações podem ser citados a adição, exclusão, remoção e/ou modificação de elementos de um processo a fim de atender às características situacionais do mesmo (GINSBERG; QUINN, 1995; XU, 2005; XU; RAMESH, 2008).

Desta forma, adaptar processos de software não é uma tarefa simples. Pequenas mudanças em um processo de software, mesmo que involuntariamente, podem afetar a sua consistência. De acordo com a norma ISO/IEC/IEEE 29148 (2011), um conjunto de elementos é considerado consistente quando estes não são contraditórios ou duplicados. Aharoni e Reinhartz-Berger (2007) complementa que um elemento de processo para ser considerado consistente deve poder ser reutilizado em diversas situações e permitir seu controle por meio de versões. Além disso, o elemento deve preservar em sua estrutura referências dos elementos que lhe deram origem.

Podem ser considerados exemplos que afetam a consistência na atividade de adaptação de processos a: i) seleção de dois elementos de processo cujas tarefas satisfazem o mesmo objetivo podem afetar a consistência deste processo fazendo com que, após adaptado, o processo leve seus executores a realizar tarefas similares ou duplicadas repetidas vezes, deixando assim, margem para ambiguidades (VAN DER AALST, 2013); e ii) seleção de elementos incompletos para compor o processo adaptado (BAJEC; VAVPOTIČ; KRISPER, 2007).

Diferentes abordagens para adaptação de processos de software são encontradas na literatura, dentre elas podemos citar: *Open Process Framework* (OPF) (ZOWGHI; FIRESMITH; HENDERSON-SELLERS, 2005), *Situational Method Engineering* (HENDERSON-SELLERS; RALYTÉ, 2010), Linhas de Processo de Software (JAUFMAN; MÜNCH, 2005; ROMBACH, 2006) e famílias de métodos (KORNYSHOVA; DENECKÈRE; ROLLAND, 2011).

O *Open Process Framework* (OPF) foi criado em 1996 e recentemente foi atualizado para estar em conformidade com a norma ISO/IEC 24744 (2007). OPF se destina principalmente à modelagem de processos de software específicos para projetos e para desenvolvimento de sistemas (GHOLAMI; SHARIFI; JAMSHIDI, 2014).

Para a adaptação de processos, o OPF recupera a partir de um repositório um conjunto de fragmentos com base em fatores situacionais, como: tamanho e habilidade da equipe, a cultura organizacional e a meta esperada para o projeto (GHOLAMI; SHARIFI; JAMSHIDI, 2014). Uma matriz bidimensional, denominada Matriz Deôntica, é utilizada para indicar possíveis relações entre cada par de fragmentos recuperados, sequenciando-os a fim de garantir a consistência em nível de relacionamento entre cada par de fragmentos (ZOWGHI; FIRESMITH; HENDERSON-SELLERS, 2005).

A abordagem *Situational Method Engineering* (SME) propõe a construção de métodos de desenvolvimento de software específicos para cada projeto de acordo com características

definidas para este. A adaptação de processos é feita por meio da recuperação e reuso de fragmentos de métodos previamente armazenados em um repositório (HENDERSON-SELLERS; RALYTÉ, 2010). A consistência dos elementos é realizada usando a estratégia de montagem (*assembly*) dos elementos que têm por objetivo eliminar fragmentos com objetivos similares ou com duplicidades (RALYTÉ; ROLLAND, 2001).

As linhas de processos de software (SPrL) visam à elaboração de processos consistentes no âmbito de sequenciamento, possibilitam a reutilização de componentes e a contextualização de processos. A consistência nas SPrL é tratada por meio do sequenciamento entre os componentes, garantindo desta forma que não seja elaborado um processo de forma ad-hoc. A variabilidade nas SPrL está empregada na utilização de componentes obrigatórios e opcionais (JAUFMAN; MÜNCH, 2005; ROMBACH, 2006).

Por sua vez, a abordagem denominada Família de Métodos (*Method Family*) consiste em agrupar e organizar vários componentes de método/processo em domínios específicos formando a estrutura de famílias. Por meio da integração de conceitos de SME com SPrL, os componentes de método são priorizados por meio do método matemático *Analytic Hierarchy Process* (AHP), agrupados em famílias de acordo com a intenção que satisfazem e posteriormente recuperados para formar uma linha de processos adaptada com base nas necessidades do projeto (KORNYSHOVA, 2013).

Na abordagem de família de métodos, a consistência dos elementos é garantida na relação que os componentes de método possuem entre si, de acordo com o domínio que satisfazem e por meio do sequenciamento destes componentes por meio da construção da linha de processo adaptada. Cabe ao engenheiro de processos a responsabilidade de identificar componentes que possuem o mesmo objetivo, agrupá-los em famílias, configurar a linha de processos e aplicar a linha de processos para adaptação (KORNYSHOVA; DENECKÈRE, 2012).

Em ambas as estratégias apresentadas, pode-se observar uma grande preocupação com a consistência dos elementos que compõem o processo adaptado em relação ao sequenciamento destes. Entretanto, a tarefa de eliminar as possíveis duplicidades internas destes elementos de processo e verificar se estes estão completos estruturalmente é ainda pouco explorada.

A estratégia apresentada nesta dissertação visa contribuir com a melhoria da qualidade nas atividades/elementos/componentes/fragmentos que compõem as referidas estratégias. A consistência destes elementos é assegurada a baixo nível por meio da eliminação de duplicidades e similaridades entre os elementos de processo, e a completude é garantida por

meio da validação da estrutura dos elementos, garantindo que os elementos de processo, na estratégia denominados fragmentos, sejam íntegros e completos.

## 2.2 *Situational Method Engineering*

*Situational Method Engineering* (SME) é uma abordagem que visa à construção de processos de desenvolvimento de software considerando as características específicas de cada projeto. Identificar as características que descrevem cada situação em um projeto e os fragmentos mais adequados de acordo com estas características são atividades cruciais em SME (HENDERSON-SELLERS; RALYTÉ, 2010).

Em SME, a situação do projeto pode ser considerada como as características que este possui. Segundo Bucher (2007), o termo "situação" se refere às contingências do projeto que devem ser levadas em conta na etapa de construção ou adaptação do processo. Um exemplo de contingência relevante é a distribuição da equipe, uma vez que trabalhar com equipes geograficamente distribuídas ou de forma local impacta diretamente no planejamento ou execução do projeto.

O principal elemento utilizado para construir processos adaptados em SME denomina-se *method fragment* (fragmentos de métodos). Um *method fragment* pode ser gerado a partir de um metamodelo por meio de um processo de instanciação. Um fragmento pode ser do tipo *process-fragment* (uma atividade, por exemplo) ou *product-fragment* (um artefato, por exemplo), sendo que esses tipos de fragmentos são definidos separadamente (HENDERSON-SELLERS; GONZALEZ-PEREZ; RALYTE, 2008).

O conceito de *method fragment* permite a descrição de relacionamentos muitos-para-muitos (n...n) entre as partes processo e produto. Essa característica tende a ser mais adequada para situações reais, em que tarefas podem ter muitos artefatos envolvidos, por exemplo. Tal como descrito por Henderson-Sellers, Gonzalez-Perez e Ralyte (2008), uma associação entre *process fragment* e *product fragments* deve ser usada para capturar dependências apropriadas entre eles, uma vez que cada fragmento de processo, quando instanciado, instancia um ou mais fragmentos de produto para construção do processo adaptado, ou seja, cada fragmento de processo pode se relacionar com um ou mais fragmentos de produto. De acordo com Pereira (2012), um fragmento de processo pode ser representado usando o modelo descrito na Tabela 1.

Tabela 1 - Modelo para definição de fragmentos de processo.

| Fragmento | Nome do fragmento |
|---|---|
| **Propósito** | Descrição do propósito do fragmento. |
| **Fonte** | Fonte a partir da qual o fragmento foi definido (RUP, Scrum e XP, por exemplo). |
| **Disciplina** | Disciplinas de engenharia de software. |
| **Fase(s)** | Fase de desenvolvimento na qual o fragmento é executado. |
| **Tarefas(s)** | Tarefas(s) que descrevem o trabalho que precisa ser realizado para que a finalidade do fragmento seja atingida. |
| **Papéis(s)** | Papéis envolvidos na execução das tarefas que compõem o fragmento. |
| **Técnicas de mapeamento** | Técnicas e orientações para executar as tarefas do fragmento. |
| **Ações(s)** | Relações entre fragmentos de processo e fragmentos de produto. Ex.: artefatos de entrada, artefatos de saída ou artefatos de entrada e saída de uma tarefa. |
| **Guia de adaptação** | **Informações de adaptação:** Descrição dos valores dos fatores propostos para definir o contexto do fragmento, como: "riscos associados ao projeto", "tamanho da equipe", entre outros.<br>**Critérios de adaptação:** Descrição de um ou mais critérios de adaptação. |

Fonte: Pereira (2012).

Uma vez identificados e documentados, os fragmentos de processo são armazenados em um repositório chamado de *method base* (base de métodos), para posteriormente serem reutilizados (HENDERSON-SELLERS; GONZALEZ-PEREZ; RALYTE, 2008). Neste repositório, fragmentos podem ser selecionados a partir das características do projeto, tais como: criticidade, taxa de mudança, custo do projeto e tamanho da equipe (SOMMERVILLE, 2010). De acordo com Ralyté e Rolland (2001), o processo adaptado em SME pode ser criado por meio de três abordagens: i) abordagem baseada em *assembly*; ii) abordagem baseada em extensão e; iii) abordagem baseada em paradigmas. A abordagem baseada em *assembly* consiste em recuperar da base de métodos os fragmentos candidatos, os quais satisfazem os requisitos do projeto e avaliar sua similaridade. Caso os fragmentos sejam similares a abordagem *assembly* delibera que seja criado um novo fragmento para eliminar possíveis inconsistências. Esta estratégia, segundo Ralyté e Rolland (2001), é a mais utilizada por analisar partes semelhantes dos fragmentos selecionados, e, posteriormente, combiná-los conforme a situação do projeto. A abordagem baseada em extensão e a abordagem baseada em paradigmas são detalhadas em (HENDERSON-SELLERS; RALYTÉ, 2010).

Portanto, a abordagem SME foi selecionada para ser usada neste trabalho, devido às seguintes características: i) propõe que “partes de processos”, denominadas fragmentos em SME, possam ser definidas e armazenadas em uma base de métodos, facilitando a criação de novos processos por meio do reuso; ii) possibilita a definição de fragmentos a partir de métodos ágeis ou planejados usando o *template* apresentado na Tabela 1; iii) considera as características do projeto e o processo de desenvolvimento de software adotado pela organização para realizar a adaptação de um processo; e iv) das abordagens estudadas foi a única que tenta por meio do uso de estratégias assegurar a completude e a consistência dos fragmentos de processo.

### 2.2.1 Técnica de modelagem baseada em *assembly*

A abordagem baseada em *assembly* serve para verificar a consistência e estabelecer *links* entre os fragmentos de métodos selecionados. Para realizar a função de conectar os fragmentos faz-se uso de duas estratégias, são elas: i) montagem por meio da estratégia de associação; e ii) montagem usando a estratégia de integração (RALYTÉ; ROLLAND, 2001).

A estratégia de associação ou processo de associação consiste em conectar fragmentos e formar um novo fragmento de tal modo que o primeiro fragmento produz um produto (artefatos de saída, por exemplo) que é a fonte do segundo fragmento selecionado (artefatos de entrada, por exemplo). A montagem por meio da estratégia de integração ou processo de integração consiste em identificar elementos comuns nos fragmentos, sejam eles fragmentos de processo ou fragmentos de produto, e posteriormente integrá-los eliminando as duplicidades (HENDERSON-SELLERS; RALYTÉ, 2010).

A Figura 1 mostra um mapa de processos descrevendo a abordagem baseada em assembly e como as diferentes estratégias podem ser empregadas.

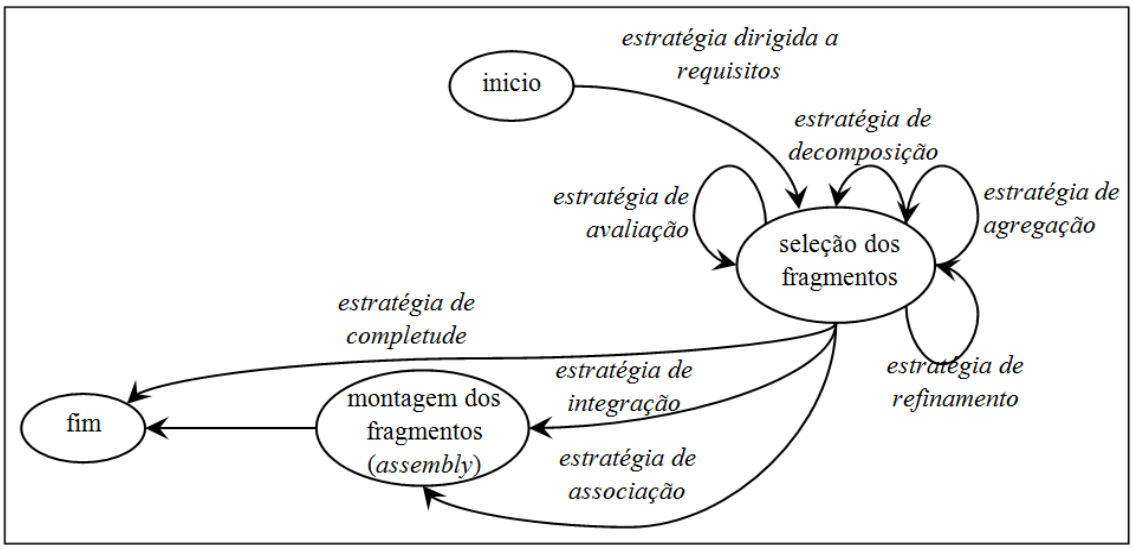



Figura 1 - Mapa de processos na abordagem baseada em assembly.

Fonte: Adaptado de Ralyté e Rolland (2001).

O mapa de processos apresenta diferentes maneiras de selecionar os fragmentos na base de métodos e duas formas distintas de montar o fragmento de método a partir da estratégia baseada em *assembly*. A intenção de montar o fragmento é satisfeita quando os fragmentos candidatos selecionados por intermédio das estratégias de integração ou associação são montados de forma coerente eliminando-se as partes repetidas ou similares (AZZOUZ; KRAIEM; GHEZALA, 2012).

Um exemplo da estratégia de associação e da estratégia de integração pode ser visualizado na Figura 2. A estratégia de associação é indicada quando os fragmentos selecionados não possuem elementos em comum. A estratégia de integração é recomendada quando os fragmentos recuperados apresentam objetivos semelhantes de engenharia, mas possuem diferentes formas de satisfazê-los.

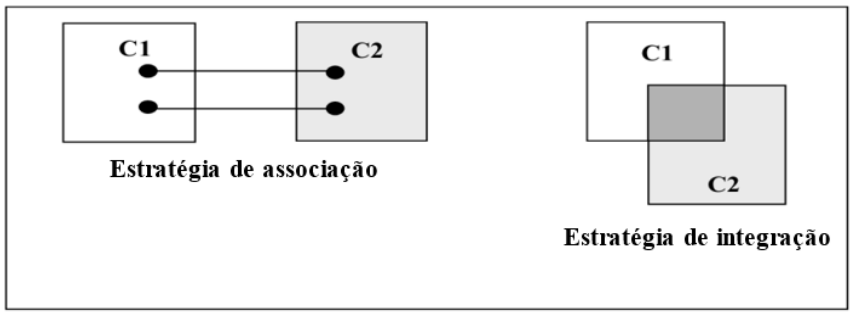



Figura 2 - Estratégias para montar fragmentos de métodos.

Fonte: Adaptado de Henderson-Sellers e Ralyté (2010).

Em ambas as estratégias, recomenda-se o uso de medidas de similaridade para identificar semelhanças semânticas e estruturais dos fragmentos envolvidos no processo de *assembly*. Segundo Ralyté e Rolland (2001), estas medidas podem ser de dois tipos: i) métricas de similaridade para produtos; e ii) métricas de similaridade para processos. O uso de métricas de similaridade tem como objetivo eliminar duplicidade nos fragmentos candidatos selecionados para compor o processo adaptado.

No entanto, possíveis defeitos podem ocorrer quando a abordagem em *assembly* é utilizada de forma isolada. Os principais defeitos são: i) incompletude interna; ii) inconsistência entre fragmentos; e iii) inaplicabilidade dos fragmentos em relação às características do projeto (BRINKKEMPER; SAEKI; HARMSEN, 1998).

A incompletude interna ocorre quando uma parte do fragmento selecionado necessita de outra parte do fragmento para compor o processo situacional. Já a inconsistência ocorre quando um fragmento selecionado contradiz ou é similar a outro fragmento selecionado para compor o processo situacional. Por outro lado, a inaplicabilidade ocorre quando o fragmento selecionado não pode ser aplicado no contexto em que foi selecionado.

De acordo com Hoef (1995 **apud** BRINKKEMPER; SAEKI; HARMSEN, 1998) todos estes defeitos se relacionam diretamente com a qualidade interna do fragmento e independem da situação do método para acontecer. Segundo Brinkkemper; Saeki e Harmsen (1998), os critérios mais importantes a serem levados em consideração são a completude (evitando a incompletude interna) e a consistência (evitando a inconsistência).

A estratégia apresentada nesta dissertação vem a contribuir com a estratégia assembly, pois a mesma permite: i) identificar fragmentos similares em tempo de execução; ii) priorizar o fragmento para determinar qual tarefa, artefato e papéis permanecem ao criar um novo fragmento e quais vão ser eliminados; e iii) recuperar ou descartar os fragmentos incompletos por meio de regras previamente estabelecidas.

### 2.2.2 Métricas de similaridade para processos

A similaridade é uma propriedade importante para a cognição, uma vez que, desempenha um papel fundamental na resolução de problemas, lembrando, prevenindo e classificando elementos e eventos. Na verdade, se não houvesse o conceito de similaridade

para cada situação, o indivíduo perceberia objetos e eventos como algo novo e teria que aprender como tratar cada situação (JENERS; LICHTER; PYATKOVA, 2012).

O conceito de similaridade está amplamente relacionado ao domínio em que a similaridade será inserida (WANGENHEIM; WANGENHEIM, 2003). Neste trabalho, o domínio escolhido é o da adaptação de processos por meio da abordagem SME e as métricas de similaridade a serem utilizadas denominam-se medida de similaridade semântica e a medida de similaridade estrutural dos fragmentos. Ambas as métricas foram utilizadas inicialmente por Ralyté e Rolland (2001) com base no conceito de similaridade proposto por Castano e Antonellis (1993).

A afinidade semântica dos fragmentos pode ser calculada usando duas medidas, são elas:

a) Afinidade semântica de intenções (SAI – *Semantic Affinity of Intentions*): utilizada para medir a proximidade de duas intenções. Uma intenção é formada por verbo e objetivo. Duas intenções são consideradas similares quando seus verbos e objetivos são sinônimos (RALYTÉ; ROLLAND, 2001). Por sua vez, a sinonímia dos termos que compõem a intenção pode ser definida pelo uso de *thesaurus* ou ontologias (ATKINSON; GUTHEIL; KIKO, 2006).

b) Afinidade semântica de seções (SAS – *Semantic Affinity of Sections*): utilizada para avaliar a proximidade de suas seções. Uma seção é formada por três parâmetros, são eles: i) intenção; ii) objetivo da intenção; e iii) estratégia para atingir o objetivo.

Na Figura 3 são exibidas as fórmulas que definem as duas medidas de similaridade por afinidade extraídas de Ralyté e Rolland (2001).

$$\mathrm{SAI}(i_i, i_j) = \begin{cases} 1 \text{ if}((\mathrm{i_i}.\mathrm{verb}\ \mathrm{SYN}\ \mathrm{i_j}.\mathrm{verb})\text{^}(\mathrm{i_i}\ .\mathrm{target}\ \mathrm{SYN}\ \mathrm{i_j}.\mathrm{target})) \\ 0 \text{ else} \end{cases}$$

$$\mathrm{SAS}(< \mathrm{i_i}, \mathrm{i_j}, \mathrm{s_{ij}} >, < \mathrm{i_k}, \mathrm{i_l}, \mathrm{s_{kl}} >) = \begin{cases} 1 \text{ if SAI } (\mathrm{i_i}, \mathrm{i_k}) = 1\text{^}\mathrm{SAI}(\mathrm{i_j}, \mathrm{i_l}) = 1\text{^}\mathrm{s_{ij}}\mathrm{SYNs_{kl}} \\ 0 \text{ else} \end{cases}$$

Figura 3 - Fórmulas da medida de similaridade semântica.
Fonte: Ralyté e Rolland (2001).

De acordo com a fórmula, dois elementos candidatos são semanticamente afins se o verbo do primeiro elemento for sinônimo do verbo do segundo elemento e o objetivo do

primeiro elemento for sinônimo do objetivo do segundo elemento. As seções de dois elementos são semanticamente afins se o primeiro e segundo elemento possuírem a mesma intenção e o terceiro e o quarto elemento possuírem a mesma intenção e as seções que compõem cada um dos elementos forem sinônimas.

Entretanto, a medida de afinidade semântica não é o suficiente para avaliar se dois elementos são similares, torna-se necessário comparar suas estruturas. Para esta tarefa, utiliza-se a afinidade estrutural dos fragmentos que faz uso de duas métricas: i) similaridade estrutural por intenções (SSI); e ii) similaridade estrutural por seções (SSS) (RALYTÉ; ROLLAND, 2001).

a) Similaridade estrutural por intenções (SSI - *Structural Similarity by Intentions*): é utilizada para medir a intenção de similaridade de dois elementos estruturados. Esta fórmula faz uso da medida de afinidade de intenções para realizar o cálculo de similaridade estrutural (RALYTÉ; ROLLAND, 2001).

b) Similaridade estrutural de seções (SSS - *Structural Similarity by Sections*): permite medir a proporção de similaridade das seções nos elementos estruturados. A Figura 4 apresenta as fórmulas que definem as duas medidas de similaridade estrutural extraídas de Ralyté e Rolland (2001). Pode-se visualizar por meio da Figura 4 as fórmulas das similaridades SSI e SSS.

$$\mathrm{SSI}(m_1, m_2) = \frac{2 * Número\ de\ intenções\ similares\ em\ m_1\ e\ m_2}{\sum_{i=1}^{2} Número\ de\ intenções\ em\ m_i}$$

$$\mathrm{SSS}(m_1, m_2) = \frac{2 * Número\ de\ seções\ similares\ em\ m_1\ e\ m_2}{\sum_{i=1}^{2} Número\ de\ seções\ em\ m_i}$$

Figura 4 - Fórmulas da medida de similaridade estrutural.
Fonte: Ralyté e Rolland (2001).

Neste contexto, as fórmulas SAI, SAS, SSI e SSS servem para medir a similaridade dos fragmentos de processo selecionados, analisando nestes, as intenções, os objetivos e as estratégias utilizadas pelos fragmentos para satisfazer estes objetivos. Neste trabalho, as fórmulas descritas serviram como motivação para modelagem da ontologia OntoSME, no que diz respeito a criação das classes, a criação dos relacionamentos entre as classes, a inclusão do

conhecimento e a criação das duas consultas em *Semantic Query-Enhanced Web Rule Language* (SQWRL) para o reconhecimento da similaridade semântica e estrutural dos fragmentos.

## 2.3 *Abordagem para gerenciamento de riscos em projetos de software*

A abordagem *Project Risk Management Approach* (PRiMA), proposta por Fontoura (2006) é uma abordagem sistemática para gerenciar riscos em projetos de software, por meio da adaptação de processos. O objetivo da abordagem é permitir a elaboração de um processo específico para um dado projeto, visando minimizar a exposição do projeto aos riscos identificados de acordo com o contexto do projeto. PRiMA propõe um metamodelo, chamado PRiMA-M, elaborado a partir da integração dos metamodelos RUP (IBM, 2006) e XP (BECK, 2004), que permite representar elementos de processo, necessários para a modelagem de métodos ágeis, planejados e híbridos.

O metamodelo PRiMA-M representa um conjunto de conceitos que são usados para a modelagem de processos de software. Com a utilização do metamodelo, pode-se instanciar elementos de processos, que são usados na definição de processos planejados, ágeis ou híbridos; ou para representar subprocessos específicos para padrões de tratamento de riscos, bem como customizações desses processos para uso em projetos específicos. As instâncias de elementos de processo são armazenadas em uma base e combinadas para elaborar diferentes modelos de processos de software (FONTOURA, 2006).

Visando facilitar a tarefa de adaptação de processos, Fontoura (2006) propôs um framework, chamado de PRiMA-F, com o objetivo de organizar as atividades instanciadas a partir do metamodelo PRiMA-M e definir o fluxo de execução das mesmas. O fluxo de execução foi modelado por meio de diagramas de atividades, elaborados por disciplina. Atividades com mesmo objetivo e artefatos de entrada e saída equivalentes são instanciadas no mesma atividade do fluxo e recebem o nome de pontos de flexibilização.

Tabelas descrevendo atividades equivalentes foram propostas por Fontoura (2006) e usadas neste trabalho para definição das regras que permitem o reconhecimento da similaridade estrutural dos fragmentos de processo. Um exemplo de atividades equivalentes pode ser visualizado na Tabela 2, onde o ponto de flexibilização PF3 apresenta as tarefas

"*Elicit Stakeholders Requests*" (RUP) e "*Write User Story*" (XP), indicando que ambas são comuns e podem ser utilizadas de forma alternada na mesma etapa do processo.

Tabela 2 - Atividades propostas para atender a disciplina de requisitos.

| PONTOS DE FLEXIBILIZAÇÃO | ATIVIDADES |
|---|---|
| PF1 | *Structure the Use-case Model (RUP)* |
| PF2 | *Manage Dependencies (RUP)* |
| PF3 | *Elicit Stakeholder Requests (RUP)*<br>*Write User Story (XP)* |
| PF4 | *Develop Requirements Management Plan (RUP)* |
| PF5 | *Develop a System prototype* |
| PF6 | *Validate a system prototype* |
| PF7 | *Develop Vision (RUP)* |
| PF8 | *Capture a Common Vocabulary (RUP)* |
| PF9 | *Find Actors and Use Case (RUP)*<br>*Write User Story (XP)*<br>*Name chunks of functionality* |
| PF10 | *Manage Dependencies (RUP)* |
| PF11 | *Divide User Story (XP)* |
| PF12 | *Priorize Use Cases (RUP)*<br>*Priorize User Story (XP)*<br>*Priorize Product Backlog*<br>*Priorize the Software Requirements* |
| PF13 | *Detail a Use Case (RUP)* |
| PF14 | *Detail the Software Requirements (RUP)* |
| PF15 | *Review Requirements (RUP)*<br>*Review the Requirements with Customer* |

Fonte: Fontoura (2006).

## 2.4 *Abordagem Octopus SME Process Tailoring Approach*

*Octopus SME Process Tailoring Approach* (OSPTA), proposta por Pereira (2012), propõe a adaptação de processos a partir de fragmentos, que selecionados a partir de um critério de adaptação são priorizados pela técnica multicritério *Analytic Hierarchy Process* (AHP) (VIDAL; MARLE; BOCQUET, 2011). Com base na contextualização do projeto, descrita por meio do *Octopus Model* (KRUCHTEN, 2010), e em critérios de adaptação, a abordagem OSPTA tem como objetivo guiar o engenheiro de processos na escolha dos melhores fragmentos para serem incluídos no processo padrão da organização para formar o processo adaptado (PEREIRA, 2012).

A escolha da AHP por Pereira (2012) vem a resolver um dos fatores críticos em abordagens baseadas em SME que é o da priorização dos fragmentos candidatos recuperados da base de métodos a fim de formar o processo adaptado com fragmentos adequados. Associada a AHP, Pereira (2012) utilizou a escala proposta por Vidal (VIDAL; MARLE; BOCQUET, 2011). A referida escala é apresentada na Tabela 3.

Tabela 3 - Escala de priorização de fragmentos utilizada na abordagem OSPTA.

| Escala | Avaliação numérica | Recíproco |
|---|---|---|
| **Extremamente preferido** | 9 | 1/9 |
| **Muito fortemente preferido** | 7 | 1/7 |
| **Fortemente preferido** | 5 | 1/5 |
| **Moderadamente preferido** | 3 | 1/3 |
| **Igualmente preferido** | 1 | 1 |

Fonte: Vidal; Marle; Bocquet (2011).

Pereira (2012) utiliza o modelo *Octopus Model* (KRUCHTEN, 2010), o qual propõe oito fatores que afetam significativamente o desenvolvimento de software, sendo utilizado para caracterizar o projeto e os fragmentos armazenados na base de métodos para priorizá-los no momento da adaptação. Assim, serão selecionados os fragmentos que satisfazem requisitos de adaptação e adequados às características do projeto. A Tabela 4 apresenta os fatores que compõem o modelo e seus possíveis valores (PEREIRA, 2012).

Tabela 4 - Valores para definição de contextos com *Octopus Model.*

| **FATORES** | **POSSIVEIS VALORES** | | |
|---|---|---|---|
| | **Características ágeis para planejadas →** | | |
| **Tamanho** | Pequeno | Médio | Grande |
| **Arquitetura** | Estável | Modificada | Móvel |
| **Modelo de Negócios** | *In house* ou sob medida para um cliente | Comercial | Componente de um grande sistema |
| **Distribuição da Equipe** | Mesmo local | Equipes diferentes | Distribuída geograficamente |
| **Taxa de Mudança (%)** | Mais que 30 | Entre 10 e 30 | Menos que 10 |
| **Idade do Sistema** | Novo desenvolvimento | Manutenção | Evolução de sistema legado |
| **Criticidade** | Perda de conforto | Perda de dinheiro | Mortes |
| **Controle** | Dinâmico / flexível | Regras simples | Mecânico / formal |

Fonte: Pereira (2012).

A partir dos valores do *Octopus Model* é possível definir contextos ágeis ou planejados para cada projeto. Por sua vez, a caracterização dos fragmentos por meio deste contexto é responsável por informar ao engenheiro de processos, quais fragmentos são os mais adequados em relação ao contexto definido para o projeto (PEREIRA; FONTOURA, 2012).

## 2.5 *Ontologias*

Uma ontologia pode ser definida como "uma especificação formal e explícita de uma conceitualização compartilhada". De forma geral, uma ontologia é um modelo de dados que representa um conjunto de conceitos de um determinado domínio de informação, bem como os relacionamentos entre estes conceitos (GRUBER, 1995).

Pode-se afirmar que cada ontologia é um sistema de conceitos e relacionamentos que juntos fornecem a estrutura básica para a construção de uma base de conhecimento acerca do domínio que esta representa. Este sistema define um vocabulário do domínio e um conjunto de restrições sobre a forma como estes termos podem ser combinados para modelar o domínio (DEVEDZIĆ, 2002).

Embora as ontologias não apresentam sempre a mesma estrutura, existem características e componentes básicos comuns presentes em grande parte delas. Dentre esses componentes, é possível citar três elementos base para a construção de uma ontologia, são eles: i) classes (descrevem conceitos organizados em uma taxonomia); ii) indivíduos (instâncias de classes); e iii) propriedades (que podem ser simples atributos como nome e idade ou relações que definem o tipo de iteração entre indivíduos de classes distintas) (GRUBER, 1995).

Em relação à forma de criação, as ontologias podem ser criadas de diferentes maneiras, por exemplo, em formato texto, RDF (HAYES, 2004), RDF *Schema* (BRICKLEY; GUHA, 2004) ou OWL (MCGUINNESS; VAN HARMELEN, 2004). A linguagem *Ontology Web Language* (OWL) é considerada pela W3C uma linguagem para representação de ontologias, entretanto, devido ao crescimento considerável de usuários surgiram algumas limitações na representação desta linguagem, como por exemplo, a limitação na representação de subclasses e propriedades. Dessa forma, em 2007 foi criado o *W3C OWL Working Group*

para resolver estas questões e trabalhar em uma nova versão da linguagem denominada OWL2 (CONSORTIUM, 2012).

As consultas em uma ontologia são realizadas por meio da linguagem *Semantic Web Rule Language* (SWRL) que se caracteriza por permitir a escrita de predicados unitários (pertencer a uma classe) e binários (relações). No entanto, é limitada por não conseguir representar o conhecimento de um domínio apenas por seus construtores (HORROCKS et al., 2004).

A linguagem SWRL também possui uma linguagem para consulta chamada *Semantic Query-Enhanced Web Rule Language* (SQWRL) que permite a criação de consultas na ontologia similar ao SQL. Desta forma, a adoção da linguagem SQWRL nas ontologias construídas é considerada uma poderosa ferramenta, pois possibilita realizar buscas nas ontologias desenvolvidas em OWL com o máximo de expressividade semântica (O'CONNOR; DAS, 2009).

Na abordagem SME, as ontologias podem ser utilizadas para descrever aspectos semânticos entre os fragmentos, definindo termos e relacionamentos entre os fragmentos armazenados na base de métodos. Esta associação semântica pode ser utilizada pelo engenheiro de processos para analisar a similaridade entre fragmentos e assim auxiliar na construção do processo adaptado (HENDERSON-SELLERS, 2011).

Por sua vez, segundo Gruninger (2007), o uso de ontologias é indicado quando existe a necessidade de confiabilidade entre os conceitos do vocabulário ou linguagem utilizada em relação ao domínio compreendido por estes conceitos. Esta confiabilidade é devida à representação formal que se adquire com o uso de ontologias, podendo até tornar possível a automação.

# 3 MODELAGEM DE UMA ONTOLOGIA PARA RECONHECIMENTO DA SIMILARIDADE EM PROCESSOS

Este capítulo apresenta a modelagem de uma ontologia denominada OntoSME utilizada para representar processos de software modelados segundo a abordagem *Situational Method Engineering*, detalhada na Seção 2.2 deste trabalho. Como principais objetivos da ontologia proposta busca-se representar: i) o conhecimento necessário para o reconhecimento da similaridade semântica e estrutural em fragmentos de processos; e ii) regras de relacionamento entre a classe fragmentos e suas subclasses.

A construção da ontologia foi baseada na sistemática proposta por Noy e McGuiness (2001), a qual propõe as seguintes atividades: i) definir o domínio e escopo que a ontologia irá cobrir; ii) criar e organizar hierarquicamente as classes presentes na ontologia; iii) determinar as propriedades de dados, as quais tem como objetivo caracterizar suas entidades, bem como as propriedades de objetos, cujo papel é estabelecer relações entre as classes; iv) estabelecer as restrições destas propriedades; e v) criar as instâncias das classes. A seguir, tais etapas serão detalhadas.

Para a definição do domínio e escopo da ontologia foi realizado um mapeamento do domínio de processos de software modelados por meio da abordagem SME. Buscou-se mapear o conhecimento acerca das instâncias, das estruturas, dos relacionamentos e dos verbos e substantivos que fazem parte dos fragmentos de processo, fragmentos de produto e das demais classes que compõem a referida abordagem.

Com base nos conceitos mapeados, buscou-se reutilizar conceitos e informações já definidos por outras ontologias e por trabalhos já publicados na área, a fim de criar e organizar hierarquicamente as classes da ontologia. Foram criados dois tipos de classes, são elas: i) classes abstratas; e ii) classes concretas. Para a criação das classes levou-se em consideração a nomenclatura utilizada para adaptação de processos de software, a hierarquia presente no modelo de processo RUP (IBM, 2006) e se a referida classe possui indivíduos instanciados (classe concreta) ou não (classe abstrata).

Na terceira fase da metodologia, buscou-se determinar as propriedades de dados e de objetos para cada classe concreta criada, às quais tem o objetivo de definir características das classes, e assim criar um elo entre as mesmas. Estas classes e propriedades representam as entidades e relações existentes dentro do domínio de adaptação de processos por meio da

abordagem SME. A execução da quarta fase da metodologia não se fez necessária em virtude das propriedades criadas não possuírem restrições no âmbito deste trabalho.

Por fim, para compor a base de conhecimento, buscou-se realizar três atividades, são elas:

i) Extrair dos modelos de processo RUP (IBM, 2006) e XP (BECK, 2004) as tarefas, artefatos, fases, propósitos, papéis e disciplinas a fim de criar os indivíduos da ontologia e mapear o correto relacionamento destes nas classes já criadas; e b) mapear todos os verbos e substantivos utilizados nas tarefas dos referidos modelos para compor o conhecimento referente à sinonímia das palavras.

ii) Associar os verbos e substantivos utilizados nas tarefas do RUP e do XP a seus respectivos sinônimos definidos a partir do site Thesaurus.com (YAGUCHI et al., 2006). O objetivo desta atividade é ampliar as chances da ontologia na identificação das palavras similares semanticamente. Como por exemplo, a tarefa "iniciar projeto" pode ser similar à atividade "começar projeto".

iii) Mapear do trabalho de Fontoura (2006), o conjunto de tarefas similares extraídas do RUP, do XP e de outros processos a fim de conceitualmente definir na ontologia que a tarefa X é similar à tarefa Y, uma vez que X possui a mesma finalidade e artefatos de entrada e saída que Y. Possuir este conhecimento é de grande importância, pois tarefas equivalentes não devem ser instanciadas em um mesmo processo.

Para realizar tais atividades foi utilizada a linguagem OWL, versão 1.0, por meio do software Protégé 3.4.4 (HORRIDGE, 2011). A Ferramenta Protégé foi escolhida por contar com uma grande quantidade de usuários, além de possuir uma excelente documentação e ser *open source*.

## 3.1 *Classes*

A representação do conhecimento por meio de ontologias é realizada a partir da modelagem de classes e propriedades de um domínio específico. Para representar o domínio proposto pela ontologia OntoSME foi criado um conjunto de classes a fim de auxiliar na identificação da similaridade semântica e estrutural dos fragmentos de processo. A Figura 5 exibe a hierarquia de classes da ontologia proposta.

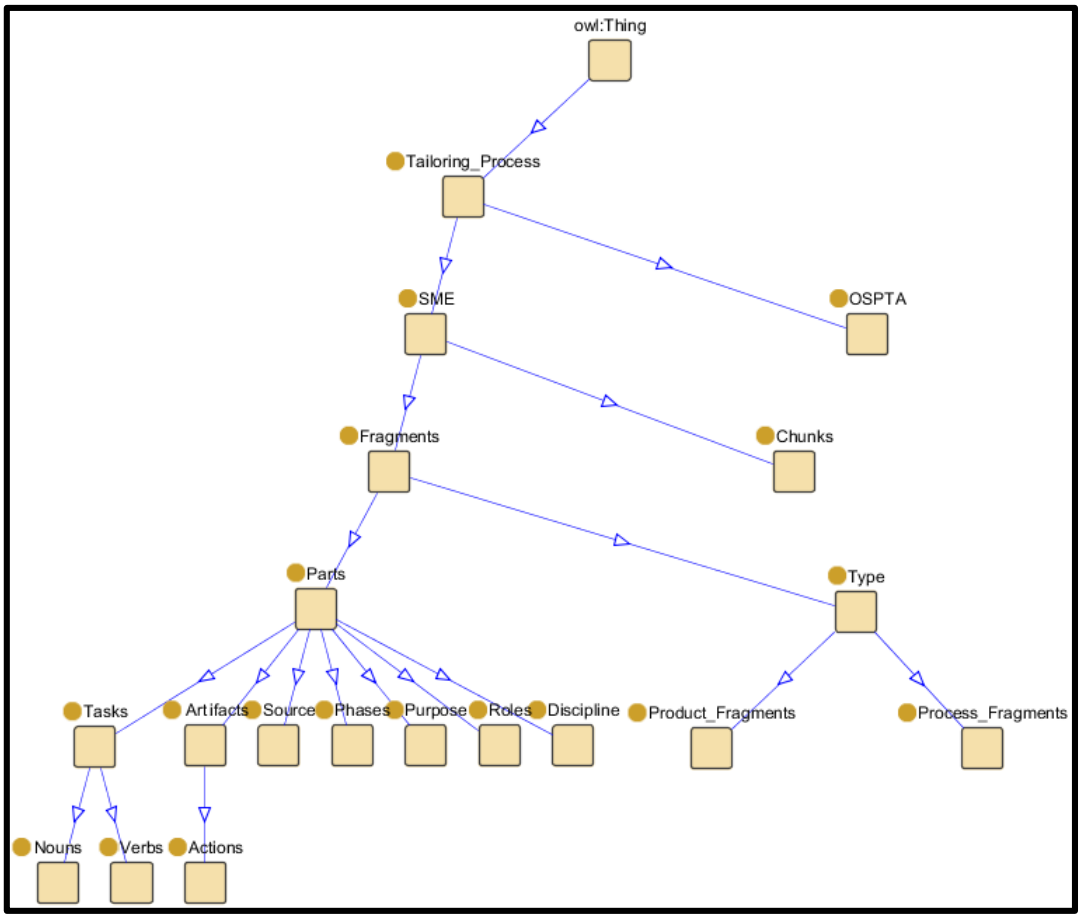



Figura 5 - Estrutura da ontologia OntoSME.

Fonte: Autor.

Sobre as classes criadas é importante ressaltar que nem todas possuem instâncias, uma vez que foram definidas como classes abstratas e apresentam a função somente de definir a hierarquia da ontologia. Exemplos deste tipo de classes são as classes "*Product_Fragments*" e "*Process_Fragments*" que são subclasses da classe "*Type*" e que representam o tipo de fragmento a ser instanciado. Por sua vez, as classes "*Fragments*" e "*Parts*" também são classes abstratas, uma vez que para existirem é necessário que indivíduos sejam instanciados nas classes concretas "*Tasks*", "*Artifacts*", "*Source*", "*Phases*", "*Purpose*", "*Roles*" e "*Discipline*".

Ao contrário das classes abstratas, as classes concretas são as que possuem um ou mais indivíduos instanciados. A classe "*Tasks*" representa as tarefas que um fragmento pode possuir e apresenta a função de fornecer o conhecimento necessário para identificação da similaridade das tarefas. Tal identificação realiza-se por meio dos indivíduos instanciados da classe "*Tasks*", sendo possível descobrir: i) tarefas similares conceitualmente por meio dos

objetivos que satisfazem; ou ii) tarefas similares semanticamente, analisando-se a sinonímia dos verbos e substantivos.

Outras classes relevantes são as classes concretas "*Nouns*" e "*Verbs*" tratadas como subclasses da classe "*Tasks*". Os indivíduos instanciados a partir da classe "*Verbs*" fornecem informações relevantes quanto à similaridade semântica dos verbos, informando, por exemplo, qual verbo é sinônimo de outro verbo.

A classe "*Verbs*" descreve outras informações referentes aos verbos, como por exemplo, com que substantivos o verbo está relacionado, em que disciplinas do processo o verbo aparece ou de qual modelo de processo o verbo foi retirado. Por sua vez, a classe "*Nouns*" contribui com informações referentes à similaridade semântica dos substantivos, explicitando quando o substantivo é usado (atividade ou tarefa), quais verbos o substantivo pode utilizar e em quais disciplinas o substantivo pode ser usado.

Na classe "*Artifacts*" foram instanciados 175 indivíduos, e sua função é armazenar os nomes dos artefatos produzidos pelas tarefas vinculadas aos fragmentos. Por sua vez, na classe "*Discipline*" foram criadas 9 instâncias, e sua função é armazenar a disciplina em que o fragmento pode fazer parte.

Na classe "*Phases*" foram instanciados 4 indivíduos representando as fases do ciclo de vida que o fragmento de método pode ser empregado. A classe "*Roles*" representa os papéis que executam as tarefas descritas no fragmento e relaciona-se diretamente com seus produtos de trabalho (classe *Artifacts*). Do mesmo modo, a classe "*Purpose*" tem a função de descrever a finalidade dos fragmentos. Optou-se por definir "*Purpose"* como classe, pois se pretende, em trabalhos futuros, identificar propósitos similares na ontologia OntoSME. Por fim, na classe "*Source*", foram instanciados nove indivíduos que representam a origem dos fragmentos.

## 3.2 *Propriedades de dados e propriedades de objetos da ontologia OntoSME*

Após criar a estrutura de classes da ontologia OntoSME e especificar os relacionamentos entre elas, deu-se início à definição das propriedades de dados e de objetos para cada classe da ontologia. Considera-se propriedade de dados todo relacionamento cujo tipo pode ser representado pelos tipos primitivos de variáveis.

Por sua vez, considera-se propriedade de objetos todo relacionamento cujo tipo é proveniente de uma classe da ontologia. Pode-se visualizar no Apêndice A, a Tabela 13 que lista as propriedades de dados e de objetos da ontologia OntoSME e suas respectivas funções.

De acordo com a Tabela 13, as propriedades de dados são visualizadas como as características utilizadas para descrever a classe em questão. A classe "*Nouns*", por exemplo, possui as propriedades de dados "*name*" e "*used_in*". A propriedade "*name*" visa representar o nome de cada indivíduo que pertence à classe "*Nouns*", por exemplo, os substantivos "*code*" e "*components*". A propriedade "*used_in*" representa se o substantivo pertence a uma atividade ou a uma tarefa.

Da mesma forma, as propriedades de objeto de uma ontologia têm o papel de estabelecer as relações entre os indivíduos instanciados a partir das classes. Um exemplo deste relacionamento pode ser observado na Figura 6, no qual o indivíduo "*Create Spike Solutions (XP)*" é similar conceitualmente aos indivíduos "*Construct Architectural Proof-of-concept (RUP)*" e "*Implement Innovative Idea*". Por sua vez, a propriedade de objeto "*use_verb*" explicita que a instância da tarefa "*Create Spike Solutions (XP)*" faz uso do verbo "*Create*", que a propriedade "*use_noun*" utiliza o substantivo "*Spike Solutions*" e que o atributo "*has_disciplines*" faz referência à disciplina "*Analysis and Design*".

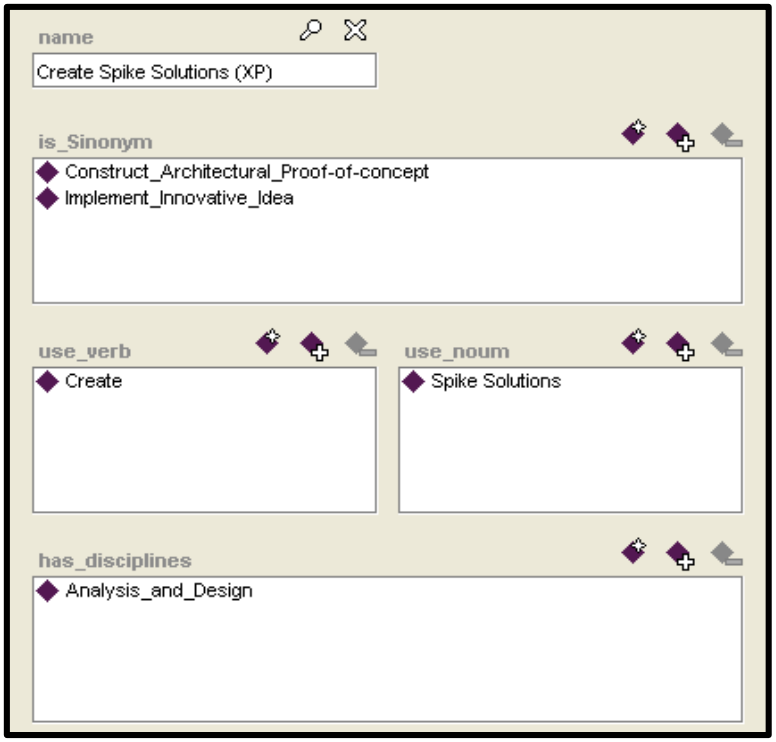



Figura 6 - Instância da classe "*Tasks*".

Fonte: Autor.

Do mesmo modo, ao se expandir a instância do verbo "*Create*" e do substantivo "*Spike Solutions*" observa-se que os mesmos também se relacionam com outras propriedades de objeto. Dentre estas propriedades, destaca-se a propriedade "*is_synonym*" que especifica que o verbo "*Implement*" é similar semanticamente ou sinônimo do verbo "*Create*" e que os substantivos "*Architectural Proof-of-concept*" e "*Innovative Idea*" são similares semanticamente ao substantivo "*Spike Solutions*". A Figura 7 apresenta a relação desta propriedade com as demais propriedades das classes "*Verb*" e "*Noun*".

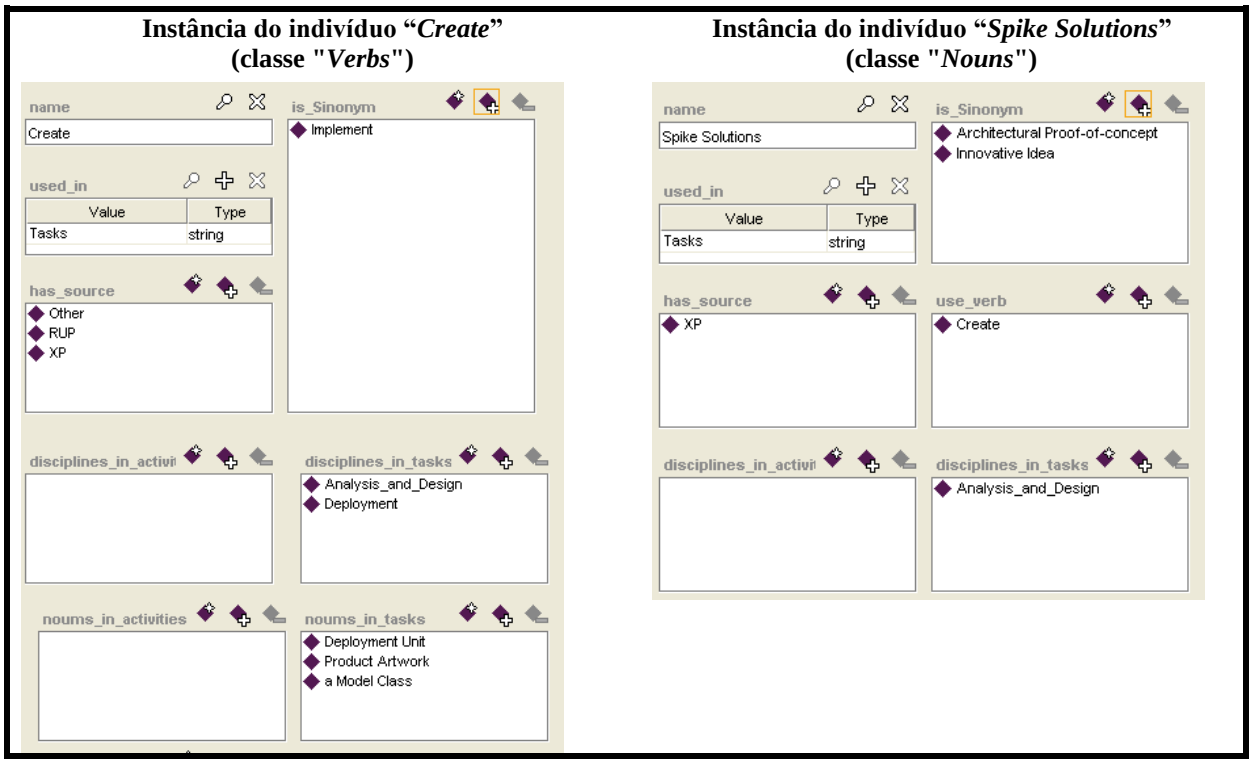



Figura 7 - Instância de indivíduos da classe "*Verbs*" e da classe "*Nouns*".

Fonte: Autor.

## 3.3 *Instanciação da ontologia e inclusão do conhecimento*

Após criar as classes e as propriedades de dados e de objetos da ontologia OntoSME, tornou-se necessário instanciar os objetos de cada classe concreta e criar os relacionamentos entre estes indivíduos a fim de proporcionar informações relevantes sobre o domínio em

questão. Na ontologia proposta, os relacionamentos foram construídos por meio das propriedades de dados que ligam os indivíduos de duas classes distintas.

Figura 8 mostra por meio de um diagrama de classes a relação entre as classes dos indivíduos instanciados com demais classes da ontologia OntoSME.

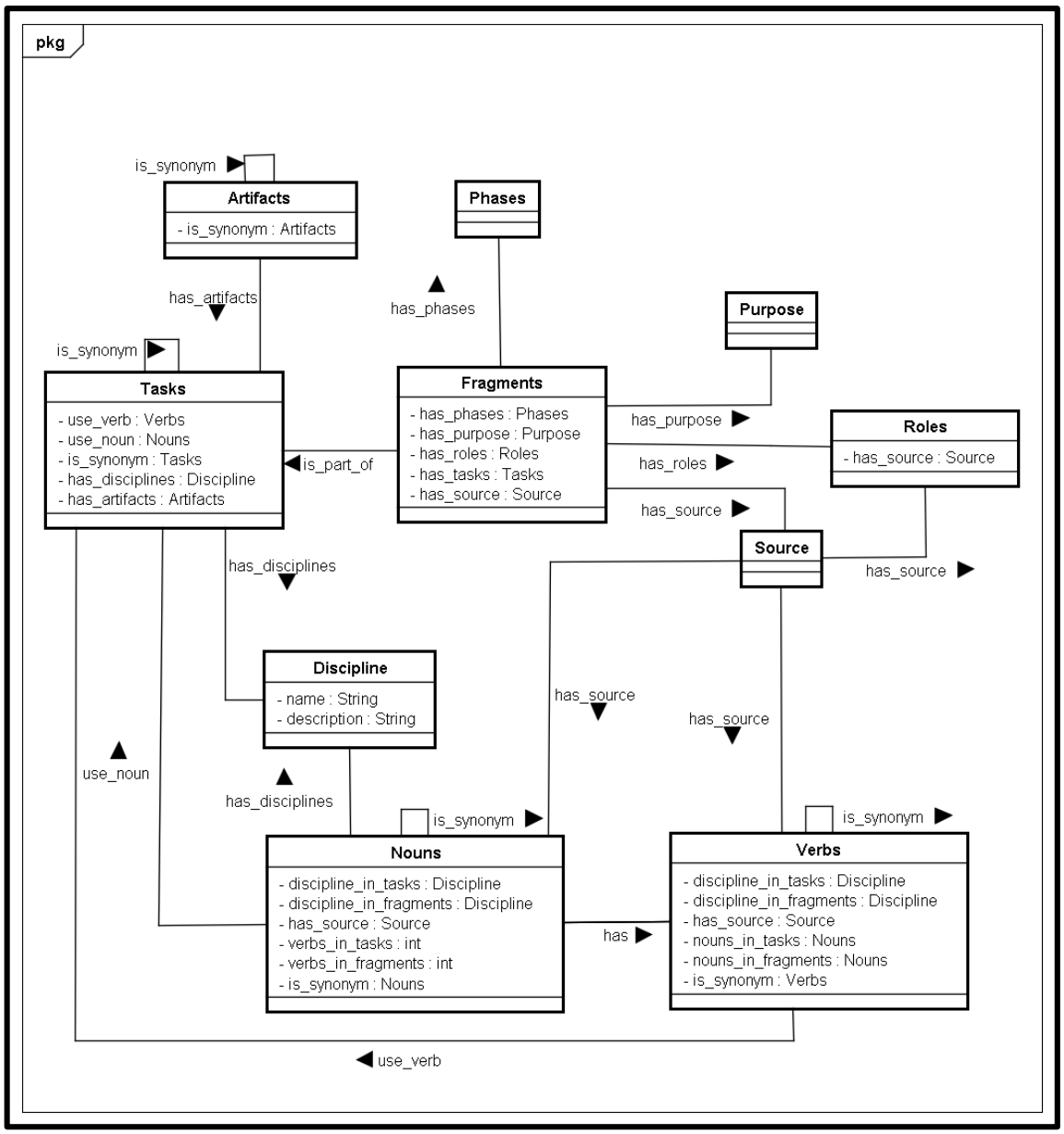



Figura 8 - Relação entre classes e indivíduos instanciados.

Fonte: Autor.

De acordo com o diagrama de classes, os indivíduos representados pelas tarefas do processo são armazenados na classe "*Tasks*" e se relacionam diretamente com as classes: i)

"*Verb*"; ii) "*Noun*"; iii) "*Discipline*"; iv) "*Artifacts*"; v) e com a própria classe "*Tasks*". Esta relação apresenta-se por meio das seguintes propriedades:

i) "*use_verb*": por meio deste relacionamento é possível especificar que os indivíduos instanciados por meio da classe "*Tasks*" usam verbos instanciados da classe "*Verb*". Na classe "*Verb*", cada instância apresenta um conjunto de propriedades que a caracterizam e permitem afirmar que o verbo utilizado pela tarefa é proveniente de determinado modelo de processos (propriedade "*has_source*" da classe "*Verb*") ou é sinônimo de outro verbo (propriedade "*is_synonym*" da classe "*Verb*").

ii) "*use_noun*": este relacionamento mostra que as tarefas instanciadas por meio da classe "*Tasks*" possuem substantivos já instanciados na classe "*Nouns*".

iii) "*is_synonym*": este relacionamento mostra que as tarefas instanciadas na classe "*Tasks*" são similares conceitualmente a outros indivíduos da classe "*Tasks*". Este conhecimento foi extraído do trabalho de Fontoura (2006) e é validado por meio da ontologia OntoSME.

iv) "*has_disciplines*": esta relação indica as disciplinas do processo em que determinada tarefa pode aparecer.

v) "*has_artifacts*": esta relação indica os artefatos que determinada tarefa pode produzir. Os indivíduos que se relacionam por meio desta propriedade são provenientes da classe "*Artifacts*" que, por sua vez, possui a propriedade "*is_synonym*", cuja função é representar os sinônimos do artefato.

A classe "*Tasks*" apresenta ainda uma relação com a classe abstrata "*Fragments*" explicitando que "*Tasks*" é parte de "*Fragments*". A classe "*Fragments*" se relaciona com as classes "*Phases*", "*Purpose*", "*Roles*" e "*Source*" por meio das propriedades "*has_phases*" (fases do processo de software), "*has_purpose*" (propósito do fragmento ou atividade), "*has_roles*" (papéis dos envolvidos) e "*has_source*" (origem de onde foi extraído a atividade ou fragmento).

A classe "*Nouns*" se relaciona com: i) a classe "*Tasks*" por meio da propriedade "*discipline_in_tasks*" (disciplina quando o substantivo aparece nas tarefas) e da propriedade "*verbs_in_tasks*" (verbos com que o substantivo se relaciona em nível de tarefas); ii) a classe "*Fragments*" por meio da propriedade "*discipline_in_fragments*" (disciplinas em nível de fragmentos ou atividades) e "*verbs_in_fragments*" (verbos em nível de fragmentos ou atividades); e iii) com a classe "*Source*" por meio da propriedade *"has_source*" (origem do substantivo, por exemplo RUP). Por sua vez, a propriedade "*is_synonym*", encontrada na

classe "*Nouns*" faz referência a outros substantivos sinônimos ou considerados similares conceitualmente em relação ao elemento avaliado.

Uma relação semelhante é encontrada na classe "*Verbs*". De acordo com a Figura 7 um indivíduo da classe "*Verbs*" pode estar relacionado a uma ou várias: i) disciplinas em nível de tarefas (relacionamento "*discipline_in_tasks*") indicando que determinado verbo faz parte de determinada tarefa usada em determinada disciplina; ii) disciplinas em nível de fragmentos (relacionamento "*discipline_in_fragments*") indicando que determinado verbo faz parte de um fragmento e que este é usado em determinada disciplina; iii) modelos de processo (relacionamento "*has_source*"); iv) substantivos em nível de tarefas (relacionamento "*nouns_in_tasks*"); e v) substantivos em nível de fragmentos (relacionamento "*nouns_in_fragments*"). Assim como os substantivos, na ontologia OntoSME, os verbos podem a vir a possuir nenhum ou vários substantivos ou verbos similares quanto ao conceito por meio do relacionamento "*is_synonym*".

Por fim, a classe "*Roles*" relaciona-se com a classe "*Source*" indicando que cada indivíduo instanciado a partir desta classe representa um papel e é originado de algum modelo de processo, ou melhor, prática. As classes "*Source*", "*Phases*", "*Purpose*" e "*Discipline*" possuem apenas propriedades simples (*"name" e "description*") representando na ontologia os nomes de modelos de processo, as fases de processos de software, os propósitos para as atividades ou fragmentos e as disciplinas de processos de software, respectivamente.

## 3.4 *Exemplos de consultas OWL na base de conhecimento*

Conforme descrito anteriormente o objetivo principal da ontologia OntoSME é fornecer o conhecimento necessário para o reconhecimento da similaridade semântica e estrutural dos fragmentos de processos e servir como ferramenta de apoio para as regras de relacionamento entre a classe fragmentos e suas subclasses. Deste modo, com base nas fórmulas SAI e SAS apresentadas na Seção 2.2.2, foram criadas as fórmulas apresentadas na Figura 9 e Figura 11.

O objetivo destas fórmulas é recuperar da base de conhecimento apenas tarefas similares semanticamente (verbos e substantivos sinônimos) ou tarefas similares quanto ao seu objetivo (similares conceitualmente). Vale ressaltar que para uma tarefa ser similar semanticamente a um conjunto de tarefas, seu verbo e substantivo devem ser sinônimos. Por

sua vez, uma tarefa é similar conceitualmente quando esta possuir o mesmo objetivo e produzir artefatos semelhantes dentro de determinado contexto.

A sinonímia dos verbos e substantivos é verificada pela fórmula apresentada na Figura 9 e escrita na linguagem SWRL. De acordo com a fórmula é possível recuperar na ontologia todos os verbos (classe "*Verbs*") e substantivos (classe "*Nouns*") em que o verbo e substantivo do fragmento ou tarefa ("$Verb\ of fragment_{i,j}$" e "$Noun\ of fragment_{i,j}$") seja sinônimo do verbo e do substantivo informado (relacionamento i*s_synonym*)**.**

$$\mathrm{SYN}(i.verb_{i,j}, i.noun_{i,j}) = Verbs(?verbClass) \wedge Nouns(?nounClass, ?prop) \wedge is\_synonym(?verbClass, "Verb\ of\ fragment_{i,j}") \wedge is\_synonym(?nounClass, "Noun\ of\ fragment_{i,j}") \wedge sqwrl:makeSet(?set, ?verbClass) \wedge sqwrl:groupBy(?set, ?verbClass) \rightarrow sqwrl:select(?verbClass, ?nounClass)$$

Figura 9 - Fórmula para medir a sinonímia dos verbos e substantivos.

Fonte: Autor.

Uma situação do uso desta fórmula pode ser representada quando é pesquisado na base de conhecimento da ontologia o verbo "*Elicit*" e o substantivo "*Stakeholder requests*". Como o domínio em questão é o domínio da adaptação de processos ao invés de retornar verbos similares a "*Elicit*" ou substantivos similares a "*Stakeholder requests*", porém fora do contexto deste trabalho, a consulta retorna o verbo "Write" e o substantivo "*User Story*" indicando que semanticamente a tarefa "*Elicit Stakeholder requests*" proveniente do RUP pode ser similar à tarefa "*Write User Story*" descrita no XP. A Figura 10 mostra o resultado da consulta utilizando a fórmula da Figura 9.

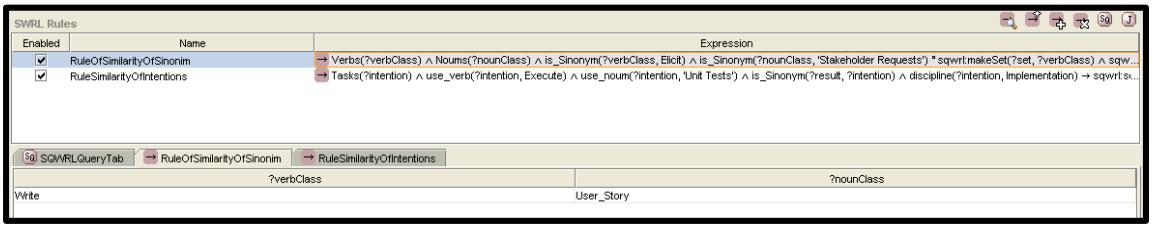



Figura 10 - Resultado da sinonímia entre o verbo "*Elicit*" e o substantivo "*Stakeholder requests*".

Fonte: Autor.

Por sua vez, a similaridade de intenções é executada em nível de tarefas, utilizando como parâmetros verbos, substantivos e a fórmula mostrada na Figura 11. De acordo com a fórmula, a consulta deve recuperar os indivíduos instanciados na classe "*Tasks*" que usam determinado verbo (relacionamento "*use_verb*") e substantivo (relacionamento " *use_noum*") e que apresentem intenções (parâmetro "*?intention*") consideradas sinônimas ou similares (relacionamento "*is_synonym(?result, ?intention)*"). Os verbos e substantivos são informados nos parâmetros "*Verb of fragment*$_{i,j}$" e "*Noun of fragment*$_{i,j}$" respectivamente.

$$
\begin{aligned}
&\text{SYN}\left(i.verb_{i,j}, i.noun_{i,j}\right) \\
&= Tasks(?\,intention) \\
&\wedge\ use\_verb(?\,intention, "Verb\ of\ fragment_{i,j}")\\
&\wedge\ \ use\_noum(?\,intention, "Noun\ of\ fragment_{i,j}") \\
&\wedge is\_synonym(?\,result, ?\,intention) \\
&\wedge\ \ discipline(?\,intention, "Discipline\ of\ fragment_{i,j}") \\
&\rightarrow\ \ sqwrl{:}\,select(?\,result)
\end{aligned}
$$

Figura 11 - Fórmula SWRL para medir a intenção dos verbos.

Fonte: Autor.

Um exemplo da utilização da fórmula da similaridade de intenções é exibido na Figura 12. Deseja-se recuperar as tarefas cujas intenções sejam comuns à tarefa "*Execute Unit Tests*" proveniente do XP. Dessa forma, é informado na consulta o verbo *"Execute"* e o substantivo "*Unit Tests*" retornando como resultado a tarefa "*Execute developer test*", originária do XP.

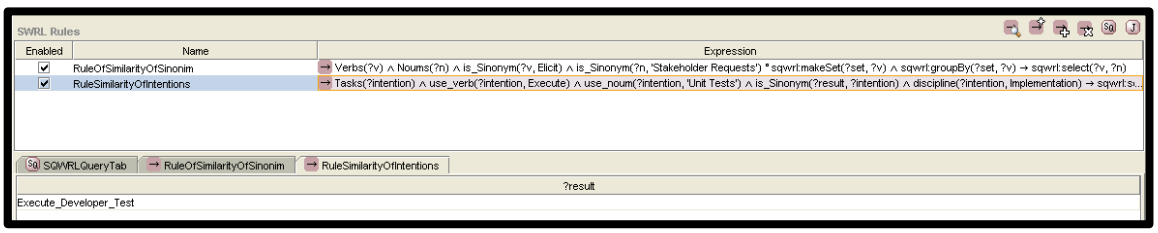



Figura 12 - Resultado da consulta utilizando a fórmula da similaridade de intenções.

Fonte: Autor.

# 4 METAMODELO PARA ADAPTAÇÃO DE PROCESSOS - MFTP

O metamodelo *"Metamodel for Tailoring Process"* (MfTP) é uma evolução do metamodelo M2F proposto por Pereira (2012) para a abordagem OSPTA, descrita na Seção 2.4. A necessidade de evoluir o metamodelo M2F surgiu da carência de possuir um metamodelo que forneça apoio às estratégias de validação da completude e consistência dos fragmentos de processo, e possibilite a adequação de elementos de processo provenientes de outras abordagens voltadas à construção ou adaptação de processos, por exemplo, OpenUp (ZOWGHI; FIRESMITH; HENDERSON-SELLERS, 2005) ou linhas de processo de software (JAUFMAN; MÜNCH, 2005; ROMBACH, 2006).

Modelado a partir do SPEM 2.0 (SEI, 2010) e do RUP versão 7.2 (IBM, 2006), o metamodelo MfTP permite a instanciação de elementos de processo que podem ser usados na definição de processos ágeis ou planejados, bem como a customização destes processos levando em conta as caraterísticas específicas dos projetos. O SPEM foi escolhido por: i) ser um metamodelo proposto pela OMG para descrever processos de software; ii) representar uma proposta de unificação entre diferentes modelos de software; e iii) propiciar a acomodação de uma grande variedade de processos de software. Por sua vez, tanto SPEM quanto o RUP usam a UML como notação e são orientados a objetos, sendo este, um fator este decisivo para sua escolha.

Deste modo, os elementos de processo instanciados a partir do metamodelo MfTP são armazenados em um repositório e posteriormente selecionados e combinados para formar os mais diferentes processos de software completos e consistentes. Pode-se visualizar na Figura 13, o diagrama de classes do metamodelo MfTP.

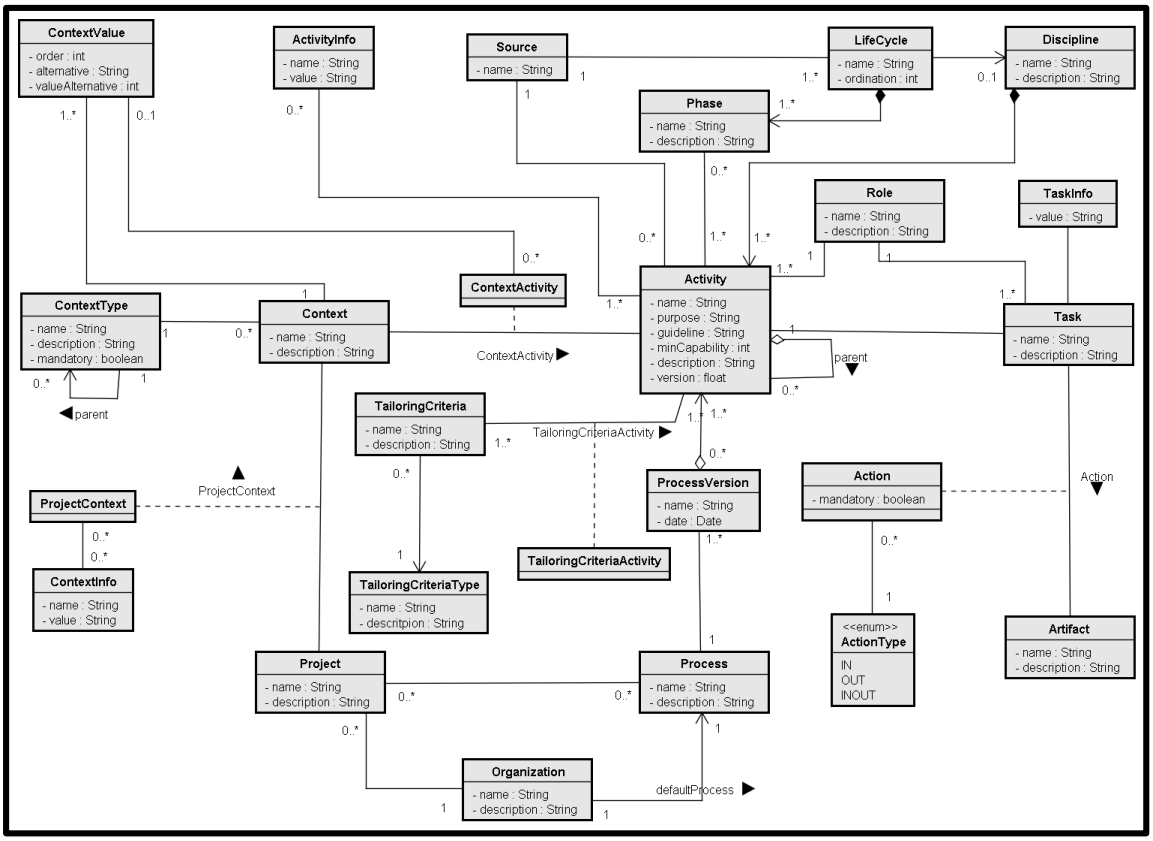



Figura 13 - Diagrama de classes do Metamodelo MfTP.

Fonte: Autor.

De acordo com o metamodelo, do ponto de vista gerencial o ciclo de vida (classe *LifeCycle*) é composto por uma ou mais instâncias de fases (classe *Phase*) e do ponto de vista técnico o ciclo de vida é composto por uma ou mais instâncias de disciplinas (classe *Discipline*).

Cada atividade (classe *Activity*) relaciona-se com nenhuma ou uma disciplina. A propriedade *version* na classe *Activity* tem a função de armazenar a versão das atividades armazenadas no repositório. Caso a atividade venha a sofrer alguma alteração e esta já estiver sido instanciada em algum processo, uma nova versão é gerada. Neste trabalho, convencionou-se chamar todos os indivíduos instanciados a partir da classe "*Activity*" de fragmentos uma vez que os conceitos utilizados para adaptação do processo de software são provenientes da abordagem OSPTA e consequentemente da abordagem SME.

Por sua vez, as atividades são constituídas de uma origem (classe *Source*) a partir da qual o fragmento foi definido (processo, modelo ou melhor prática) e de tarefas (classe *Task*) que são instanciadas e vinculadas a um ou mais papéis (classe *Role*). Vale ressaltar que para

uma atividade existir é necessário no mínimo um relacionamento com uma instância da classe tarefas e esta deve pertencer a no mínimo uma disciplina (classe *Discipline*).

A classe *Artifact* é responsável por instanciar os indivíduos que representam os produtos de trabalho mediante a execução das tarefas. Para cada relacionamento entre tarefas (classe *Tasks*) e artefatos (classe *Artifact*) uma ação (classe *Action*) é gerada, indicando se o produto de trabalho quando instanciado é um artefato: i) de entrada (leitura do artefato – *IN*); ii) criado (escrita do artefato – *OUT*); ou iii) modificado (leitura e escrita do artefato – *IN/OUT*). São considerados exemplos de artefatos os protótipos de interface do usuário, modelos de casos de uso, contratos, defeitos de dados associados, entre outros.

A classe *Project* é responsável por representar os projetos definidos para as organizações (classe *Organization*). Desta forma, cada organização pode possuir um ou mais projetos e cada projeto pode ter um ou mais processos versionados (classe *Process*), permitindo assim, que se um processo após sua criação não tenha atendido às expectativas da organização, por exemplo, esse seja evoluído criando uma nova versão a partir do processo atual. A classe responsável por este versionamento do processo é a classe *ProcessVersion*.

No momento da criação de um projeto é possível definir um contexto (classe *Context*) para o projeto por meio da classe *ProjectContext* e pode ser de um tipo (classe *ContextType*). Cada contexto vinculado ao projeto possui um valor (classe *ContextValue*) previamente definido. O contexto também é vinculado às atividades pela classe *ContextActivity* tornando-se fundamental na priorização das atividades no momento da adaptação do processo.

Como exemplos de contextos que podem ser instanciados por meio da classe *ContextType*, cita-se o *Octopus Model*, proposto por Kruchten (2010), o qual apresenta oito fatores que podem contextualizar um projeto, são eles: tamanho da equipe, criticidade, modelo de negócio, arquitetura estável, distribuição da equipe, controle, taxa de mudanças e idade do sistema. Estes oito fatores, uma vez instanciados, formam os objetos da classe *Context*.

Cada atividade possui um ou mais critérios de adaptação (classe *TailoringCriteria*) e cada critério de adaptação deve ser de um tipo (classe *TailoringCriteriaType*). O relacionamento entre os critérios de adaptação e as atividades é dado pela classe *TailoringCriteriaActivity*, que tem a função de associar os critérios de adaptação vinculados a cada atividade de processo e de recuperar da base de métodos as atividades que satisfaçam os critérios definidos pelo engenheiro de processos ou pelo responsável em adaptar ou definir o processo da organização. São considerados exemplos de critérios de adaptação: riscos associados a um projeto (FONTOURA; PRICE, 2008), critérios de qualidade esperados para

o projeto (BRASIL; PEREIRA; FONTOURA, 2012), metas de desenvolvimento interno e externo (KALUS; KUHRMANN, 2013), entre outros.

No entanto, o grande diferencial do modelo MfTP em relação ao modelo proposto por Pereira e Fontoura (2012) é o relacionamento entre as classes "*Task*", "*Role*", "*Artifact*" e "*Action*". No modelo M2F tais classes eram vinculadas diretamente na classe "*Activity*". Na abordagem proposta, este grupo de classes está vinculado à classe "*Task*" fazendo com que, para que um fragmento exista, este esteja vinculado a pelo menos uma tarefa e nesta tarefa exista um artefato de entrada, um artefato de saída e um papel previamente definido. Por sua vez, o modelo MfTP também propicia a criação de atividades agregadas a partir de elementos e características de atividades já criadas e validadas, aplicando assim, conceitos de reuso inclusive na etapa de criação de novas atividades.

# 5 ESTRATÉGIA QUALITY VALIDATE STRATEGY

Este capítulo descreve a estratégia denominada *Quality Validate Strategy* (QaVaS), que tem como objetivo validar a completude e a consistência em fragmentos de processos de software instanciados por meio da abordagem OSPTA. Além disso, a estratégia visa à melhoria da qualidade dos fragmentos de processo, a redução da complexidade na seleção dos fragmentos e auxiliar o engenheiro de processos impedindo que o mesmo selecione fragmentos incompletos, com duplicidades ou com tarefas similarmente iguais.

Na Seção 5.1 é apresentada uma descrição geral da abordagem. Na Seção 5.2 descreve-se como são definidos e instanciados os fragmentos de métodos a partir do metamodelo MfTP. Na Seção 5.3 é detalhada a sistemática de recuperação e seleção dos fragmentos candidatos à validação. Na Seção 5.4 são apresentadas as regras de qualidade definidas para validação de novos fragmentos de processo. Por sua vez, na Seção 5.5 e na Seção 5.6 são detalhados os processos de validação da completude e consistência respectivamente.

## 5.1 *Descrição Geral*

A estratégia denominada *Quality Validate Strategy* (QaVaS) foi desenvolvida com base em conceitos de SME e tem como objetivo garantir a consistência e a completude dos fragmentos de processo instanciados a partir do metamodelo MfTP. Por meio da estratégia proposta, novos fragmentos completos e consistentes podem ser criados e fragmentos incompletos podem ser recuperados quando for o caso, permitindo assim, uma melhoria contínua não só dos fragmentos armazenados na base de dados, mas dos processos construídos a partir destes fragmentos.

Dentre os diferenciais da estratégia QaVaS destacam-se a possibilidade de:

i) Recuperar em tempo de execução fragmentos incompletos que tenham sido armazenados de maneira incompleta na base de métodos e recuperá-los com base em regras pré-estabelecidas;

ii) Armazenar os fragmentos corrigidos pelo gerente de processos após recuperação;

iii) Criar novos fragmentos a partir de fragmentos já existentes;

iv) Eliminar inconsistências entre os fragmentos recuperados a partir dos critérios de adaptação definidos para o projeto a fim de evitar ambiguidades no processo adaptado;

v) Alterar o processo definido para um mesmo projeto e versionar o referido processo de acordo com as diferentes versões elaboradas, selecionando inclusive fragmentos já tratados pelas estratégias.

A estratégia QaVaS é dita multicritérios por considerar o critério de adaptação como meio de seleção dos fragmentos candidatos na base de métodos e os oito fatores do *Octopus Model* (KRUCHTEN, 2010) como critérios de comparação para priorizar os fragmentos de acordo com o contexto do projeto. Ressalta-se que a estratégia proposta está preparada para dar suporte a outros critérios de adaptação como os quarenta e nove critérios de adaptação estudados por Kalus e Kuhrmann (2013) com base em uma revisão sistemática da literatura de critérios para adaptação de processos.

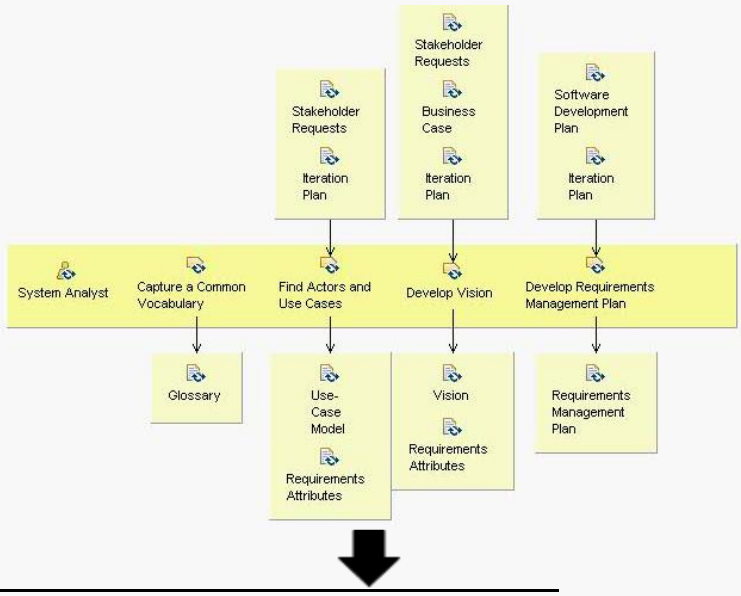
O processo da estratégia QaVaS é composto por cinco etapas, são elas: i) definição dos fragmentos de método; ii) recuperação dos fragmentos de acordo com os critérios de adaptação definidos para o projeto; iii) priorização dos fragmentos de acordo com o contexto definido para o projeto; iv) validação da completude dos fragmentos; e v) validação da consistência dos fragmentos. Nas seções a seguir são detalhadas as etapas descritas a fim de elucidar o processo.

## 5.2 *Definição dos fragmentos de métodos*

Para definição dos fragmentos de métodos, utilizou-se o modelo de representação de fragmentos proposto por Pereira (2012), descrito na Seção 2.4, com algumas adequações. O autor definiu um metamodelo para representar os conceitos de elementos de processos a partir do metamodelo do *Rational Unified Process* (RUP) (IBM, 2006) e da ISO/IEC 24744 (ISO/IEC 24744, 2007). Este metamodelo foi adaptado e está incorporado ao metamodelo MfTP, detalhado no Capítulo 4.

Optou-se neste trabalho por utilizar a estrutura de fragmentos proveniente da abordagem OSPTA, visto que a base de métodos existentes já possui um conjunto de fragmentos definidos sendo necessário somente adaptá-los ao metamodelo MfTP. A Figura 14

mostra um exemplo de fragmento definido a partir da atividade “Analisar o problema” do RUP (IBM, 2006).




| | |
|---|---|
| **Fragmento** | Analisar o problema. |
| **Descrição** | O primeiro passo em qualquer análise e problema é ter certeza de que todas as partes envolvidas concordam sobre qual é o problema que precisa ser resolvido. Após compreender plenamente o problema é necessário saber quais são as partes interessadas para criar uma visão conceitual do projeto |
| **Propósito** | O propósito deste fragmento é estabelecer um acordo sobre o problema a ser resolvido; identificar os envolvidos e identificar as fronteiras do sistema. |
| **Fonte** | RUP |
| **Disciplina** | Requisitos. |
| **Fase(s)** | Concepção |
| **Tarefas(s)** | ▪ Capturar vocabulário comum;<br>▪ Desenvolver plano de gerenciamento de requisitos;<br>▪ Encontrar atores e casos de uso;<br>▪ Desenvolver visão. |
| **Artefatos** | ▪ Glossário;<br>▪ Plano de gerenciamento de requisitos;<br>▪ Modelos de caso de uso;<br>▪ Visão |
| **Papéis(s)** | ▪ Analista de sistemas;<br>▪ *Stakeholders.* |
| **Guia de adaptação** | **Contexto do fragmento:**<br>▪ ***Tamanho:*** *Largo;*<br>▪ ***Arquitetura estável:*** *Novo*<br>▪ ***Modelo de negócio:*** *Grande componente no sistema*<br>▪ ***Distribuição do time:*** *Geograficamente distribuido*<br>▪ ***Taxa de mudanças:*** *Menor que 10*<br>▪ ***Idade do sistema:*** *Evolução legado*<br>▪ ***Criticidade:*** *Perda de dinheiro*<br>▪ ***Governança:*** *Mecânico/Formal*<br>**Critérios de adaptação:**<br>▪ ***Riscos:*** *Falta de entendimento dos requisitos* |
| **Versão** | 1.0 |
| **Fragmentos de origem** | Nenhum. |

Figura 14 - Fragmento “Analisar o problema”.

Fonte: Autor.

Neste contexto, um fragmento de método é descrito por meio de um conjunto de atributos brevemente descritos a seguir. O atributo “Fragmento” descreve o nome do fragmento. No caso do fragmento ter sido definido a partir do RUP, o nome da atividade do RUP é usado como nome do fragmento. O atributo “Descrição” traz informações pertinentes sobre o fragmento. O objetivo do fragmento é dado pelo item “Propósito”. No atributo “Fonte” é descrita a origem do fragmento, ou seja, de qual modelo de processo este se refere. A propriedade “Disciplina” descreve a qual área de concentração a atividade está relacionada.

Do mesmo modo, o atributo “Fase” apresenta a fase do ciclo de vida do processo no qual as tarefas descritas pelo fragmento são executadas. No atributo “Tarefas” são descritas as tarefas que precisam ser realizadas para que o propósito do fragmento seja atingido. No atributo “Artefatos” são descritos os artefatos que são produzidos pelas tarefas, se estas forem executadas. Na propriedade “Papéis” são detalhados as pessoas, os times e os papéis que executam as referidas tarefas.

As propriedades “Contexto do fragmento” e “Critérios de adaptação” detalham como e onde os fragmentos de método podem ser executados, ou seja, qual a situação que o fragmento de método satisfaz. Na primeira propriedade é informado o contexto do projeto, atualmente os valores provenientes do *Octopus Model*. Na segunda propriedade são descritos os critérios de adaptação responsáveis por caracterizar o fragmento informando o que este fragmento satisfaz, por exemplo, riscos associados ao projeto.

Na propriedade “Versão” é descrita a versão do fragmento. Por padrão quando um fragmento é criado, este recebe a versão 1.0. Cada alteração do fragmento, se este já estiver associado a um processo, uma versão é criada.

No atributo “Fragmentos de origem” são descritos os fragmentos que originaram o fragmento em questão. Por padrão, esta propriedade vem com valor nulo e só é utilizada quando a estratégia QaVaS é executada e cria um novo fragmento completo e consistente.

Ressalta-se que a escolha de utilizar uma base de métodos já pronta é de fundamental importância para a validação da estratégia QaVaS, uma vez que a base de dados importada até então não havia passado por nenhuma validação de consistência e completude em seus fragmentos. Cabe à estratégia proposta recuperar ou excluir do processo de seleção da base de métodos tais fragmentos candidatos caso estes estejam incompletos ou inconsistentes.

## 5.3 *Recuperação e seleção dos fragmentos na base de métodos*

Na estratégia proposta por este trabalho, têm-se as etapas de validação da consistência e completude após a recuperação dos fragmentos da base de métodos. Acredita-se que estas validações são necessárias para diminuir a complexidade referente à seleção de fragmentos e impedir que o engenheiro de processos selecione fragmentos incompletos ou inconsistentes, comprometendo assim a qualidade do processo construído.

Pode-se visualizar na Figura 15, modelada a partir de um diagrama em *Business Process Modeling Notation* (BPMN) (FETTKE, 2008), as atividades necessárias para construir o processo adaptado a partir dos fragmentos completos e consistentes validados pela estratégia QaVaS. A imagem de uma "pessoa" representa as atividades que são realizadas pelo engenheiro de processos responsável por: "Informar projeto e contexto do projeto", "Informar critérios de adaptação", "Selecionar fragmentos completos" e "Selecionar fragmentos". Definindo, desta forma, o processo da organização.

As atividades contendo a imagem de um "*script*", representam algoritmos de execução da estratégia QaVas responsáveis por validar a completude e consistência e construir novos fragmentos se necessário para formar o processo adaptado. Como exemplo, "Recuperar fragmentos", "Descartar fragmentos incompletos" e "Criar versão do processo adaptado".

Por sua vez, as atividades representadas pela imagem de um "*Loop*" e de um "Sinal de adição" representam os subprocessos de validação da completude e consistência respectivamente. Os artefatos produzidos quando uma atividade ou subprocesso é executado são simbolizados por meio da imagem de um "Papel". Para melhor visualização, ambas as imagens descritas nos parágrafos anteriores se encontram detalhadas na legenda disponível na Figura 15.

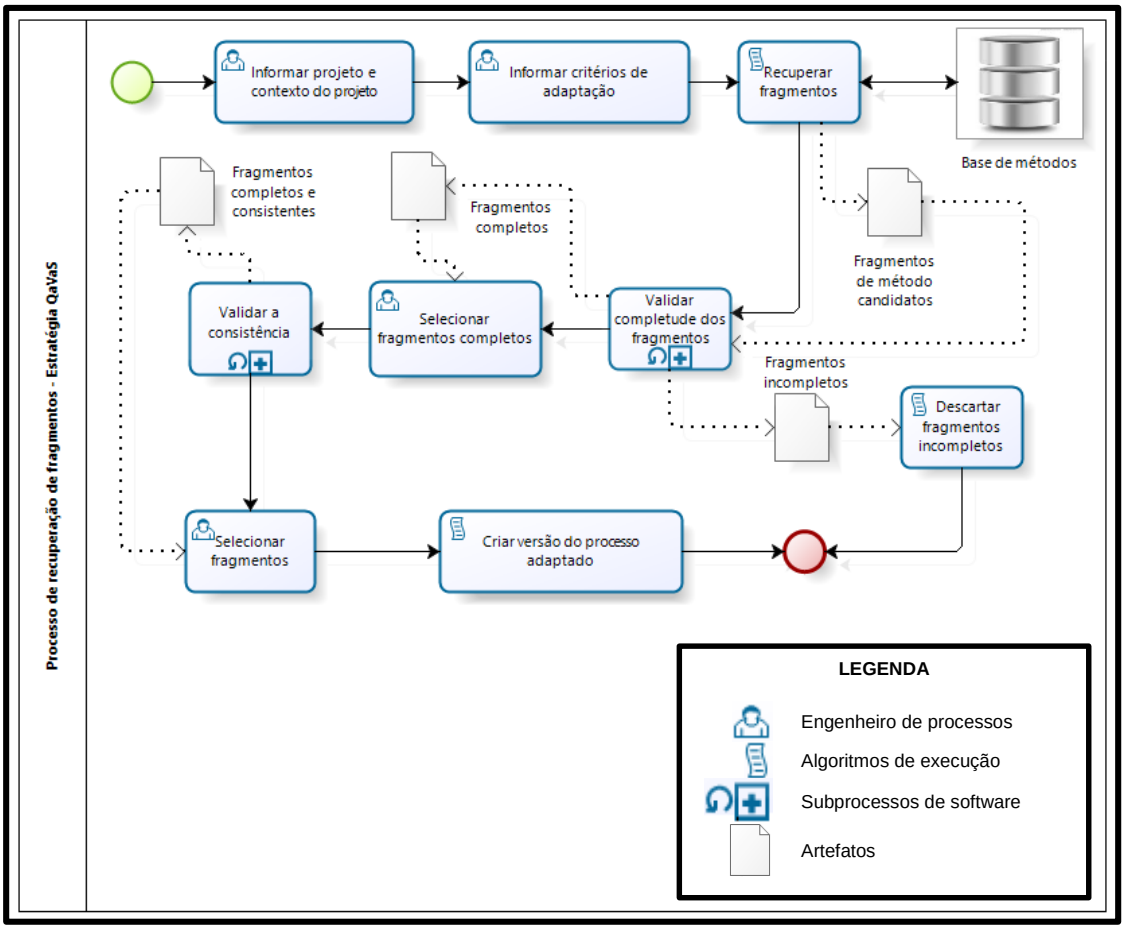



Figura 15 - Processo de recuperação e seleção dos fragmentos.

Fonte: Autor.

O processo de recuperação dos fragmentos inicia-se pelas atividades "Informar projeto e contexto do projeto" e "Informar critérios de adaptação", realizadas pelo engenheiro de processos. Considera-se como critérios de adaptação os requisitos não-funcionais do projeto, por exemplo, requisitos de segurança, critérios de qualidade, entre outros.

Nesta dissertação com o intuito de validar a estratégia proposta, utilizou-se como critérios de adaptação, riscos definidos para projetos. Para cada risco existe um conjunto de fragmentos associados, por exemplo, para o risco falta de entendimento dos requisitos, têm-se os fragmentos: "*Scenarios Define Problem*", "*Implied Requirement*", "*Build Prototype*", "*Constant Refactoring*" e "*Planning Game*".

Outros exemplos de riscos que são adicionados ao processo são: falta de envolvimento dos usuários, instabilidade de requisitos, falta de conhecimento/habilidade da equipe, entre outros. O relacionamento entre riscos e atividades é uma proposta de Fontoura (2006) e visa associar riscos de projeto às atividades que atendem estes riscos. Do mesmo modo, neste

trabalho, os fragmentos armazenados na base de métodos foram associados a riscos para serem posteriormente recuperados de acordo com a necessidade do projeto.

Desta forma, a partir da execução da etapa “Informar critérios de adaptação” são recuperados um conjunto de fragmentos que satisfazem os riscos informados pelo engenheiro de processos. De posse deste conjunto de fragmentos candidatos, inicia-se a estratégia QaVaS por meio da execução da atividade “Validar a completude dos fragmentos”.

A atividade “Validar a completude dos fragmentos” é entendida pela estratégia como um subprocesso que executa a validação da completude (Seção 5.6). Sua finalidade é avaliar os fragmentos candidatos e separá-los em dois grupos, que são: i) o grupo de fragmentos completos; e ii) o grupo de fragmentos incompletos.

O grupo de fragmentos completos é composto pelos fragmentos candidatos aprovado pela estratégia da completude e que podem ser selecionados pelo engenheiro de processos para dar início à estratégia da consistência. Já o grupo de fragmentos incompletos, é composto pelos fragmentos candidatos que obtiveram uma média de completude inferior a 60%, estes fragmentos são descartados pela estratégia QaVaS para impedir que o engenheiro de processos selecione-os erroneamente.

Após a validação da completude dos fragmentos, o engenheiro de processos seleciona os fragmentos completos executando a atividade “Selecionar fragmentos completos”. A próxima atividade é “Validar a consistência”, aplicando os conceitos definidos na Seção 5.7. Para fragmentos similares definiu-se que é necessária a criação de um novo fragmento versionado a partir das regras apresentadas na Seção 5.5, que aborda as regras para criação de fragmentos completos e consistentes.

A criação de um novo fragmento versionado permite que os fragmentos criados por meio da estratégia QaVaS sejam monitorados por sua versão na base de métodos e possibilita que o engenheiro de processos selecione, para compor o processo adaptado, somente fragmentos completos e consistentes que já foram validados pela estratégia.

Por sua vez, de posse dos fragmentos completos e consistentes, tem-se a atividade “Selecionar os fragmentos”. Nesta atividade, o engenheiro de processos seleciona os fragmentos priorizados que melhor se adequam à caracterização do projeto. A priorização dos fragmentos será abordada na Seção 5.4.

Como última atividade, tem-se a criação de uma versão para o processo adaptado e o seu armazenamento. Cabe ressaltar que uma vez o processo criado, ele pode ser modificado e a estratégia criará uma nova versão para o processo.

## 5.4 *Priorização dos fragmentos de método*

A priorização dos fragmentos, empregada na atividade “Selecionar fragmentos completos” na Figura 15 da Seção 5.3, utiliza um método matemático chamado *Analytic Hierarchy Process* (AHP). O método AHP (VARGAS, 2010) é um método multicritério amplamente utilizado e conhecido para apoio à tomada de decisão em ambientes complexos.

Com o método AHP, muitas variáveis ou critérios podem ser levados em conta para a seleção e priorização de alternativas. Segundo Vargas (2010), o AHP trabalha com alternativas e com uma meta global, ou seja, a probabilidade numérica de cada alternativa é calculada e quanto maior for esta probabilidade, maior é a chance de uma alternativa satisfazer a meta global.

Neste trabalho, as alternativas esperadas para compor o método AHP, correspondem aos fragmentos de métodos armazenados na base de métodos. A meta global é definida pela atividade de priorizar os fragmentos mais adequados para satisfazer os requisitos de adaptação definidos na segunda atividade da abordagem.

O método de priorização permite ainda que sejam atribuídos pesos relativos para cada um dos fatores usados para definição do contexto, ou seja, neste trabalho para cada fator do *Octopus Model*, possibilitando que seja atribuída uma importância relativa dos fatores em relação à meta global.

A partir de uma matriz comparativa relacionada a cada critério de comparação, são calculadas as prioridades relativas de cada fragmento. Desta forma, cada fragmento de método terá sua probabilidade relativa para cada fator do *Octopus Model*. A probabilidade relativa significa a chance que o fragmento tem de atender com sucesso a meta global em relação a cada critério. Após isso, é calculada a probabilidade final de cada fragmento em relação à meta global, definindo assim, uma listagem de fragmentos priorizados.

Os resultados obtidos com o método AHP servem para: i) guiar o engenheiro de processos na escolha dos melhores elementos para serem inclusos no processo adaptado; e ii) auxiliar a estratégia QaVaS a definir quais tarefas permanecem e quais tarefas são excluídas no momento que a estratégia de validação da consistência for empregada e a construção de um novo fragmento for necessária. Ressalta-se que a priorização por meio do método AHP é realizada nos fragmentos de processo e que a estratégia QaVaS apenas usa esta priorização como critério de decisão na escolha das tarefas similares entre os fragmentos priorizados.

Um exemplo de priorização de fragmentos de processos selecionados a partir de riscos definidos para o projeto e priorizados de acordo com o método AHP pode ser visualizado na Tabela 5. Os fragmentos de processo foram recuperados com base nos riscos "*Misunderstanding the Requirements*" e "*Requirements Instability*" e priorizados de acordo com o contexto do projeto, definido por meio do *Octopus Model*, são eles: i) "*Size=Small"*; ii) *Stable Architecture = New*; iii) "*Business Model = In House*"; iv) "*Team Distribution = Collocated*"; v) "*Rate of Change = More than 50*"; vi) "*Age of System = New development*"; vii) "*Criticity = Comfort loss*"; e viii) "*Governance = Simple rules*".

Tabela 5 - Fragmentos priorizados segundo o método AHP.

| ***Tailoring Requirements*** | ***Fragments to prevent risks*** | ***Probability*** |
|---|---|---|
| *Misunderstanding the Requirements* | *On Site Customer* (BECK, 2004) | 19.851% |
| | *Early and Regular Deliver XP* (BECK, 2004) | 19.851% |
| | *Write User Story* (BECK, 2004) | 19.851% |
| | *Divide User Story* (BECK, 2004) | 19.851% |
| | *Build Prototype* (COPLIEN; ALEXANDER, 1996) | 11.967% |
| | *Scenarios Define Problem* (COPLIEN; ALEXANDER, 1996) | 2.877% |
| | *Implied Requirement* (COPLIEN; ALEXANDER, 1996) | 2.877% |
| | *Early and Regular Deliver RUP* (PROCESS, 2007) | 2.877% |
| *Requirements Instability* | *Early and Regular Deliver XP* (COPLIEN; ALEXANDER, 1996) | 33.293% |
| | *On Site Customer* (BECK, 2004) | 33.293% |
| | *Build Prototype* (COPLIEN; ALEXANDER, 1996) | 19.712% |
| | *Implied Requirement* (COPLIEN; ALEXANDER, 1996) | 4.567% |
| | *Scenarios Define Problem* (COPLIEN; ALEXANDER, 1996) | 4.567% |
| | *Early and Regular Deliver RUP* (PROCESS, 2007) | 4.567% |

Fonte: Autor.

## 5.5 *Regras de qualidade para construção de novos fragmentos*

A complexidade é um fator evidente quando se trata em construir um processo adaptado. Isso se deve uma vez que cada processo é formado por um conjunto de fragmentos compostos de várias partes correlacionadas a fim de satisfazer determinado objetivo.

Desta forma, entende-se que a noção de qualidade na etapa de seleção e priorização dos fragmentos candidatos que podem vir a compor o processo adaptado é diretamente relacionada à garantia de consistência e completude destes fragmentos. Entende-se também, que esta avaliação pode ser feita de forma automatizada mediante a construção de um algoritmo que obedeça a regras pré-estabelecidas.

Neste trabalho, foram definidas regras de qualidade para construção de novos fragmentos. A Tabela 6 apresenta as regras de qualidade propostas.

Tabela 6 - Regras de completude e consistência para construção de novos fragmentos.

| | |
|---|---|
| 1. Completude | 1.1. Após criar o fragmento final, não deve haver elementos incompletos neste. Por exemplo, cada tarefa do novo fragmento deve estar relacionada a um papel e ao menos a um produto de trabalho.<br>1.2. Quando um fragmento é criado a partir de dois ou mais fragmentos, é necessário que um artefato de saída de um fragmento candidato seja usado como artefato de entrada no novo fragmento;<br>1.3. Cada fragmento de processo deve gerar ou alterar pelo menos um produto de trabalho;<br>1.4. Cada novo fragmento de processo deve possuir um ou mais critérios de adaptação e estar associado a um contexto para posterior priorização. |
| 2. Consistência | 2.1. Para que um fragmento seja criado com base em dois ou mais fragmentos candidatos, estes devem: i) satisfazer a mesma intenção, ou seja, devem possuir tarefas e artefatos de saída similares e pertencer a mesma disciplina do ciclo de vida; ou ii) possuírem o mesmo critério de adaptação, o mesmo contexto e a mesma disciplina do ciclo de vida;<br>2.2. Para criação de um novo fragmento, os fragmentos candidatos devem possuir o mesmo critério de adaptação;<br>2.3. Não deve haver duplicação de nomes para fragmentos de método diferentes, ou seja, não devem existir fragmentos com mesmo nome que possuam tarefas distintas em disciplinas;<br>2.4. Todo fragmento de processo criado deve ser versionado e referenciar os fragmentos de processo candidatos que lhe deram origem. |

Fonte: Autor.

Tais regras foram implementadas em um algoritmo escrito na linguagem Java. Por sua vez, são utilizadas no processo de validação da consistência por meio das atividades “Aplicar estratégia de associação” e “Aplicar estratégia de integração”.

Torna-se importante salientar, que o conceito empregado nas regras propostas para construção de novos fragmentos foram inspiradas no conceito empregado por Brinkkemper; Saeki e Harmsen (1998). Entretanto, tais regras precisaram ser adaptadas para satisfazer o contexto da estratégia QaVaS, uma vez que as regras propostas por Brinkkemper; Saeki e Harmsen (1998) não satisfazem a necessidade de priorizar os fragmentos de acordo com o contexto do projeto, nem versionam os referidos fragmentos.

## 5.6 *Validação da completude*

Por definição, um fragmento é considerado completo quando todas as informações referentes a sua estrutura podem ser instanciadas e quando o relacionamento entre suas partes pode ser criado. Como proposta para validação da completude foi criada a Tabela 7 para auxiliar na avaliação da completude de um fragmento.

Tabela 7 - Tabela para auxiliar na avaliação da completude de um fragmento.

| **Avaliação do item** | **Tipo** | **Índice de impacto** |
|---|---|---|
| Possuir um título | Eliminatório | 3 |
| Possuir uma fonte de origem | Classificatório | 2 |
| Estar associado a uma disciplina da engenharia de software | Eliminatório | 3 |
| Possuir fases no ciclo de vida | Classificatório | 1 |
| Possuir tarefas associadas | Eliminatório | 3 |
| Possuir ao menos um artefato | Classificatório | 3 |
| Possuir como artefato de entrada artefatos de saída provenientes de outros fragmentos | Classificatório | 2 |
| Possuir papéis associados às tarefas | Classificatório | 3 |
| Possuir uma descrição | Classificatório | 1 |
| Possuir um propósito | Classificatório | 1 |
| Possuir critérios de adaptação e um contexto associado | Eliminatório | 3 |
| | **Total de pontos possíveis** | 25 |

Fonte: Autor.

De acordo com a Tabela 7, são propostos 11 itens para avaliação da completude em fragmentos. Tais itens foram divididos em dois grupos (itens do tipo eliminatório e itens do tipo classificatório) e pontuados diretamente usando uma escala de 1 a 3 (um a três), sendo 3 (três) o peso dado ao item de maior importância. Os itens do tipo eliminatório, além de pontuar no cálculo da completude, podem eliminar o fragmento candidato da seleção. Por sua vez, os itens classificatórios ajudam o fragmento a ficar melhor classificado na escala da completude.

O cálculo para definir a média de completude de um fragmento é determinado a partir do total de pontos obtidos nos itens em conformidade, dividido pelo total de pontos possíveis para os itens avaliados, que juntos somam 25 (vinte e cinco) pontos. O quociente desta divisão multiplicado por 100 (cem) fornecerá o grau de completude do fragmento responsável por definir se o fragmento está habilitado para seleção ou não. Na Figura 16, pode-se visualizar a fórmula para determinar o grau de completude do fragmento.

$$Grau\ de\ completude = \frac{\sum dos\ pontos\ obtidos}{\sum dos\ pontos\ possíveis}\ x\ 100$$

Figura 16 - Fórmula do grau de completude de um fragmento.
Fonte: Autor.

O processo para avaliar se um fragmento pode ser selecionado é mostrado na Figura 17. A primeira atividade denominada “Priorizar fragmentos” consiste em priorizar os fragmentos candidatos de acordo com o contexto definido para o projeto utilizando a técnica AHP. A priorização no processo da completude tem o objetivo de auxiliar o engenheiro de processos a definir: i) quais fragmentos completos satisfazem o contexto definido para o projeto; e ii) quais fragmentos necessitam ser recuperados, caso estes sejam fragmentos incompletos passíveis de recuperação.

A próxima atividade denomina-se “Avaliar a completude dos fragmentos com base nos pesos definidos na tabela da completude”. A função desta atividade é avaliar a completude dos fragmentos com base na Tabela 7. Os fragmentos avaliados por esta tabela que atingirem o grau de completude igual a 100% (cem por cento) são considerados pelo processo de avaliação da completude fragmentos completos e habilitados para seleção pelo engenheiro de processos ou pelo responsável de definir o processo da organização.

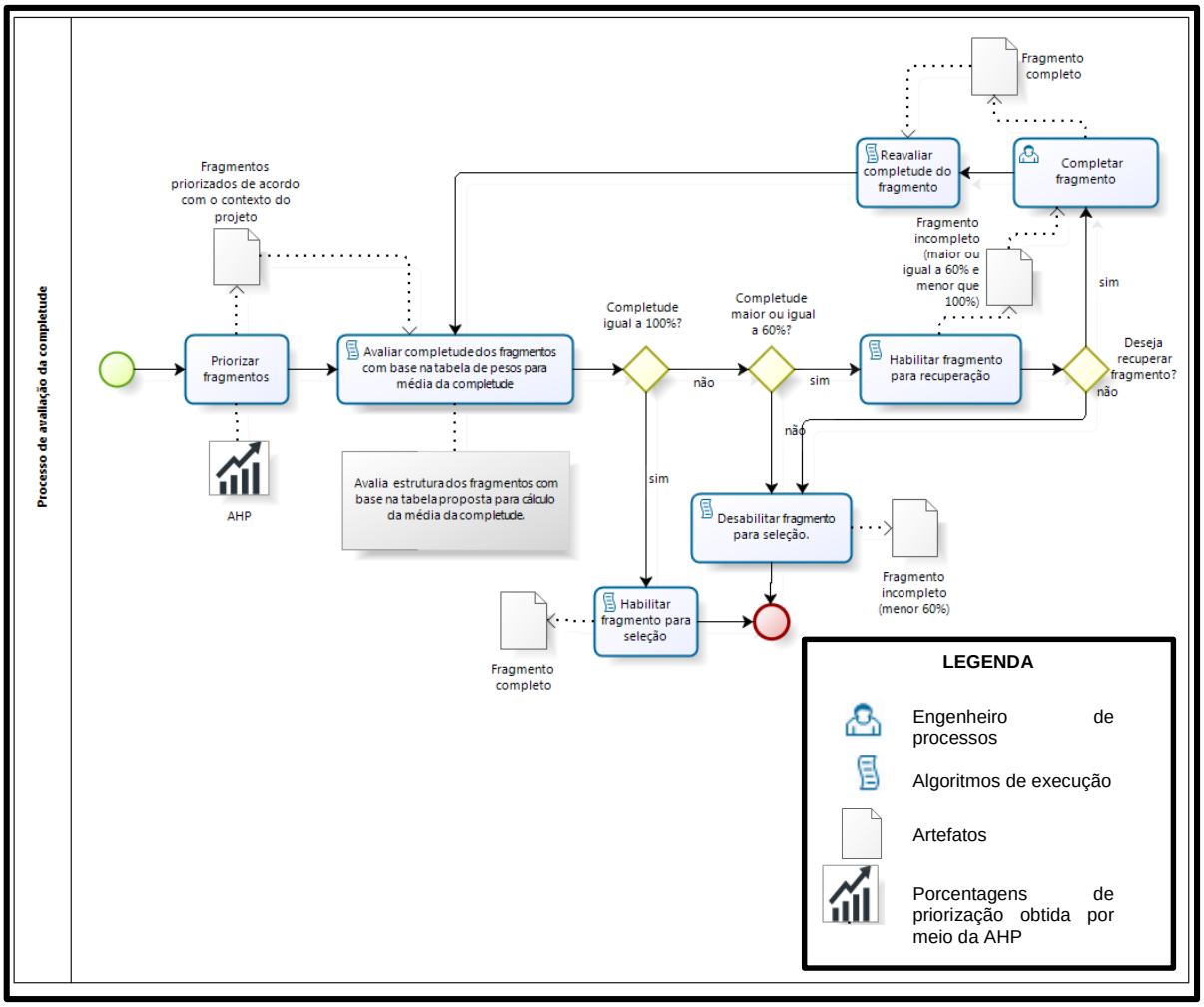



Figura 17 - Processo de avaliação da completude de um fragmento.

Fonte: Autor.

Já os fragmentos que obtiverem grau de completude maior ou igual a 60% (sessenta por cento) e que não foram suprimidos do processo por não respeitar os itens eliminatórios são habilitados para recuperação. Por sua vez, os fragmentos que obtiveram média inferior a 60% ou que não atenderam aos itens eliminatórios são desabilitados, tornando-se impossível sua seleção.

A recuperação dos fragmentos com média de completude igual ou superior a 60% depende do engenheiro de processos completar o referido fragmento por meio da atividade "Completar fragmento". Entretanto, essa atividade não garante que o fragmento esteja completo, necessita-se reavaliar a completude do fragmento por meio de uma nova avaliação dando início novamente ao processo.

Ao final do processo de validação da completude, os fragmentos classificam-se por meio de dois artefatos, são eles: o artefato "Fragmento completo" e o artefato "Fragmento

incompleto". O primeiro, representa os fragmentos completos, habilitados para seleção e para a validação da consistência e o segundo, representa os fragmentos incompletos que devem ser descartados pela estratégia QaVaS.

A fim de exemplificar o processo de validação da completude, criou-se um projeto fictício. Pretende-se com o referido projeto criar um processo adaptado para o mesmo recuperando da base de métodos fragmentos candidatos que satisfaçam os riscos "*Misunderstanding the Requirements*" e "*Requirements Instability*" que possam vir a ser priorizados a partir da seguinte configuração do contexto definido para o projeto:

i) "*Size=Large"*;
ii) *Stable Architecture = Stable*;
iii) "*Business Model = Large System Component*";
iv) "*Team Distribution = Geographic*";
v) "*Rate of Change = Less than 10*";
vi) "*Age of System = Legacy Evolution*";
vii) "*Criticity = Deaths*"; e
viii) "*Governance = Mechanic/formal*".

Desta forma, por meio dos riscos "*Misunderstanding the Requirements*" e "*Requirements Instability*" foram recuperados da base de métodos 5 (cinco) fragmentos candidatos para serem avaliados pelo processo de validação da completude, são eles: i) "*Analyze the Problem*" (completo); ii) "*Define a Cantidate Architecture"* (incompleto); iii) "*Name chunks of functionality*" (incompleto); iv) "*Define a Cantidate Architecture"* (completo; com uma tarefa); e v) "*Write User Story*" (incompleto). Os referidos fragmentos podem ser visualizados por meio da Figura 18 e da Figura 19. Nestas figuras, os elementos inexistentes que deveriam completar os fragmentos incompletos são representados por meio de imagens em escala cinza com transparência de 50%.

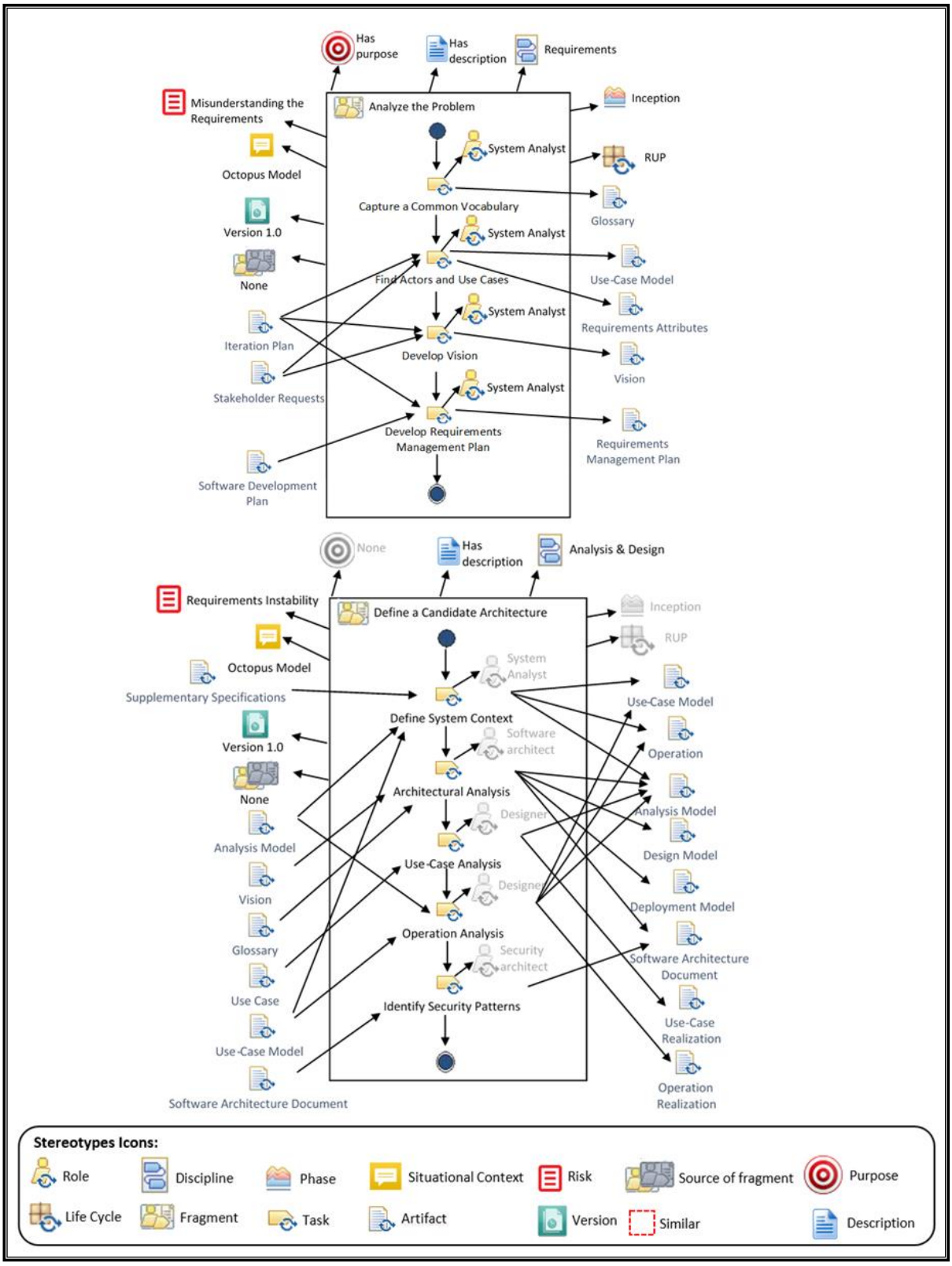



Figura 18 - Fragmentos recuperados por meio dos riscos "*Misunderstanding the Requirements*" e "*Requirements Instability*".

Fonte: Autor.

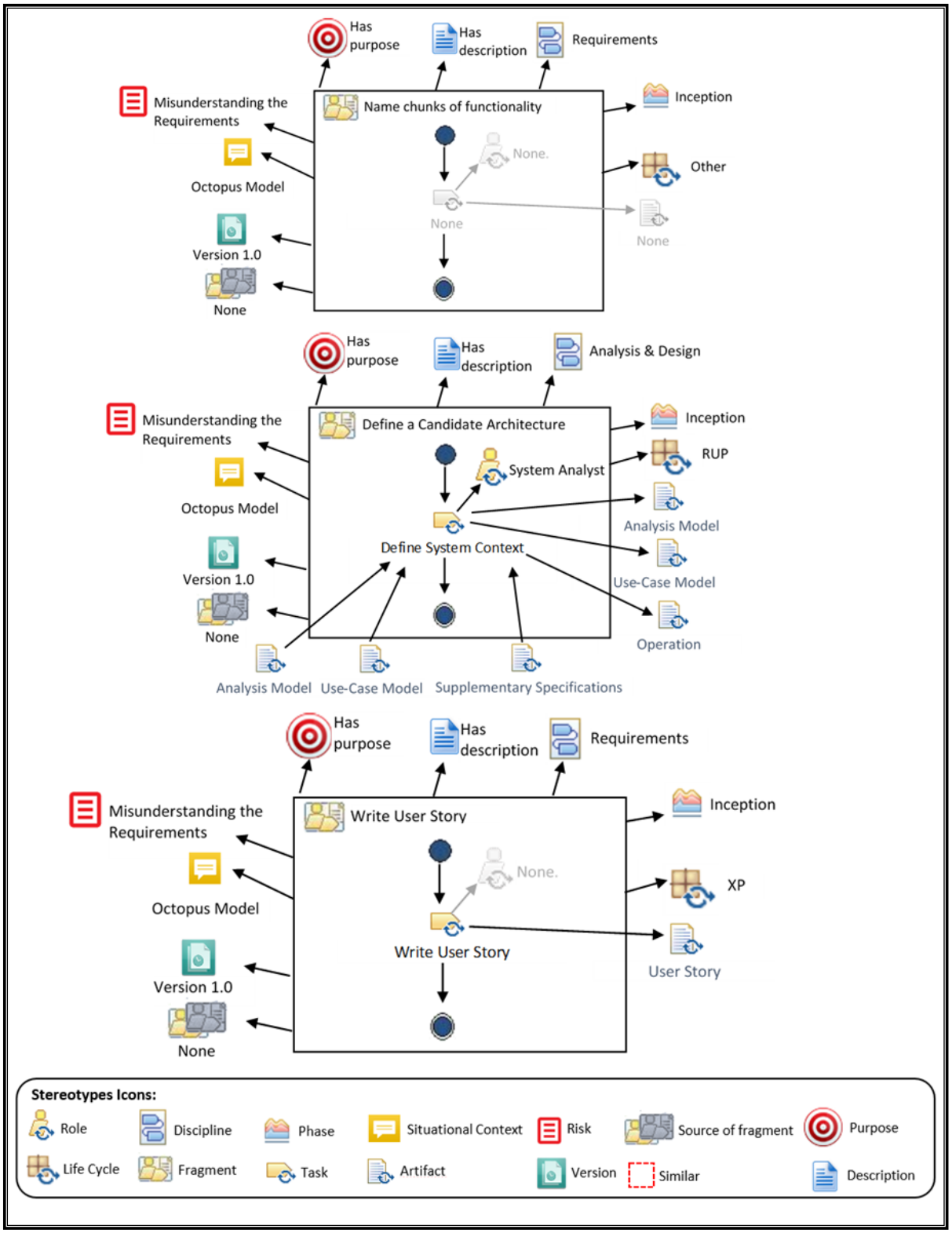



Figura 19 - Fragmentos recuperados por meio do risco "*Misunderstanding the Requirements*".

Fonte: Autor.

Desta forma, ao aplicar o processo de validação da completude nos fragmentos candidatos obteve-se os seguintes resultados:

i) O fragmento "*Analyze the Problem*" obteve grau de completude de 100% por satisfazer todos os itens detalhados na Tabela 7 e priorização de 38,886% em relação ao contexto definido para o projeto.

ii) O fragmento "*Define a Canditate Architecture*" obteve grau de completude de 72%, informando ao engenheiro de processos que o fragmento é 8% incompleto. Segundo as regras para avaliação da completude o fragmento foi cadastrado sem conter: i) uma origem; ii) fases no ciclo de vida; iii) papéis associados às tarefas cadastradas; e iv) um propósito. De acordo com as regras de validação da completude este fragmento pode ser recuperado, pois possui todos os itens eliminatórios e obteve grau de completude superior a 60%. A priorização deste fragmento segundo o contexto definido para o projeto foi de 38,884%.

iii) O fragmento "*Name chunks of funcionality*" obteve média de completude igual a 36% sendo eliminado do processo de avaliação da completude por não possuir tarefas associadas, item este eliminatório. O referido fragmento obteve grau de priorização de 1,230%.

iv) O fragmento "*Define a Canditate Architecture*", com apenas uma tarefa, obteve grau de completude igual a 100% e priorização de 19,553%. Embora este fragmento possua nome similar a outro fragmento recuperado, a validação da completude não exclui o referido fragmento da estratégia QaVaS, cabe ao processo da consistência avaliar qual dos dois fragmentos pode ser eliminado.

v) O fragmento "*Write User Story*" obteve média da completude igual a 88% devido ao não relacionamento da tarefa "*Write User Story*" com o papel do responsável por executá-la. Dessa forma, cabe ao engenheiro de processos recuperar o fragmento apenas associando um papel à referida tarefa. O fragmento após priorizado obteve grau de priorização de 1,447%.

Os fragmentos incompletos "*Define a Canditate Architecture*" (72%) e "*Write User Story*" (88%) que obtiveram média de completude superior a 60% foram recuperados pelo engenheiro de processos e novamente avaliados pela estratégia, desta vez, obtendo média de completude de 100%, habilitados pela estratégia para seleção para dar início ao processo de validação da consistência. Por sua vez, fragmento "*Name chunks of funcionality*" foi descartado por não possuir tarefas, item este, considerado eliminatório pelo processo de validação da completude. A Figura 20 e a Figura 21 mostram os fragmentos após recuperação.

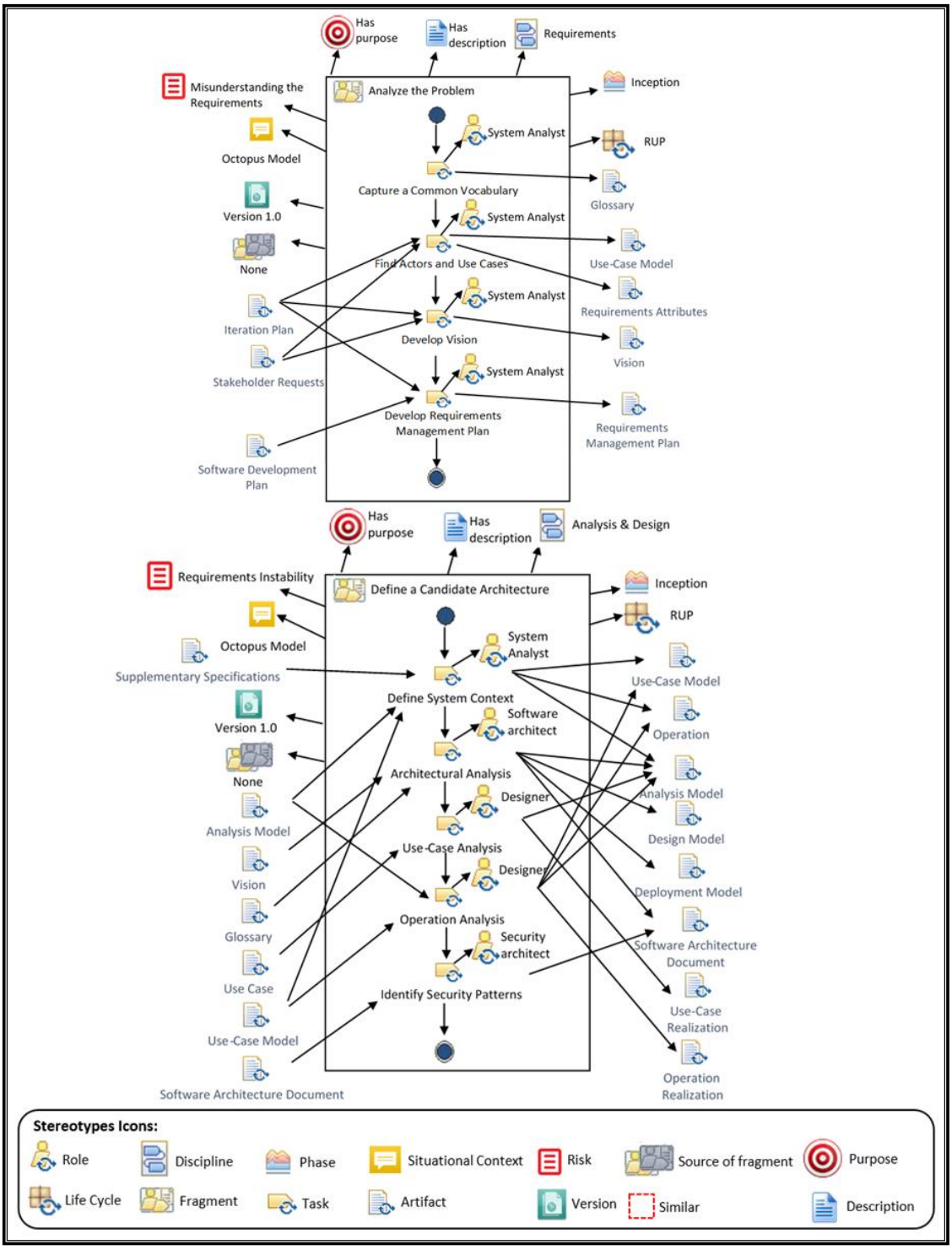



Figura 20 - Fragmentos recuperados por meio do processo de validação da completude.

Fonte: Autor.

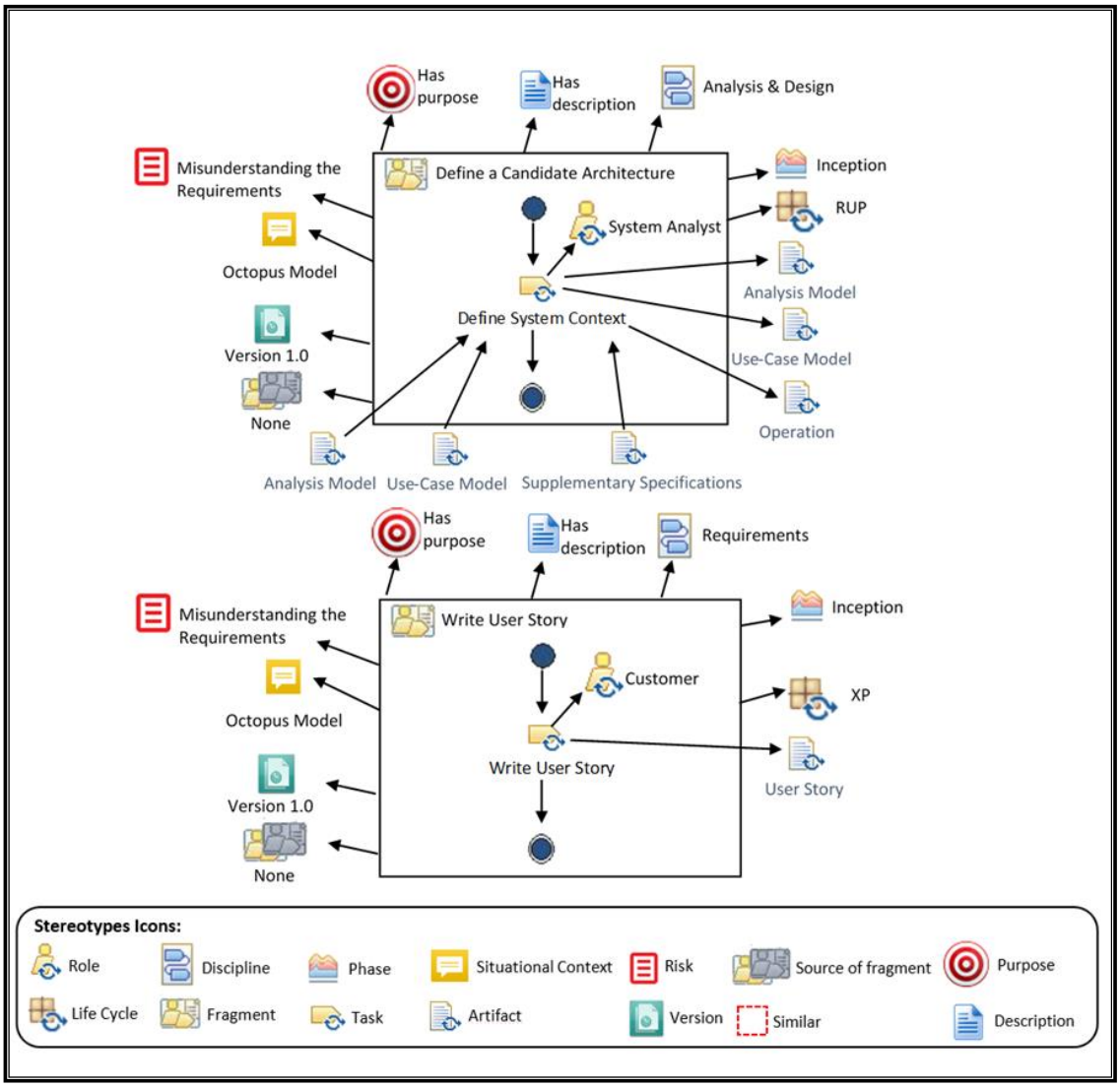



Figura 21 - Fragmentos recuperados por meio do processo de validação da completude.

Fonte: Autor.

Ressalta-se que embora o fragmento "*Write User Story*" tenha uma média de priorização baixa (1,447%) em relação ao contexto definido para o projeto, a estratégia não impediu que o engenheiro de processos recuperasse o referido fragmento uma vez que cabe a este profissional o papel de decidir se o fragmento priorizado pode ou não fazer parte do processo adaptado. Por sua vez, os fragmentos completos priorizados e selecionados pelo engenheiro de processos são encaminhados para o processo de validação da consistência, no qual são submetidos a outras avaliações a fim de melhorar a qualidade do processo.

## 5.7 *Validação da consistência*

Neste trabalho considera-se que um fragmento é consistente quando o mesmo pode ser reutilizado várias vezes sem que se anule ou possua em sua estrutura tarefas similares a outros fragmentos durante a construção do processo adaptado. Este conceito vai ao encontro ao conceito apresentado por Aharoni e Reinhartz-Berger (2007), que define consistência como a capacidade que um fragmento tem de ser reutilizado em diferentes contextos sem se repetir ou se modificar após instanciado.

Desta forma, para atingir o objetivo de garantir a consistência nos fragmentos selecionados pelo engenheiro de processos foram definidas quatro premissas. São elas:

i) Somente poderão sofrer avaliações os fragmentos candidatos que tiverem sido aprovados na estratégia da completude. Analisar fragmentos já completos impede a geração de inconsistências na relação dos elementos que compõem a estrutura de um fragmento.

ii) Caso o fragmento venha a ser alterado, uma nova versão para o fragmento deve ser criada, impedindo assim, que processos já criados sem a estratégia QaVaS sofram alteração em sua estrutura.

iii) Se fragmentos forem similares semanticamente ou estruturalmente, um novo fragmento deve ser criado a partir destes fragmentos utilizando a estratégia da integração, da associação e as regras para garantia da consistência e completude em fragmentos.

iv) Caso os fragmentos possuam o mesmo nome, mas forem diferentes de forma semântica e estrutural, estes devem ser renomeados e versionados, eliminando assim possíveis inconsistências no banco de dados.

O processo para verificação da consistência nos fragmentos candidatos tem início após a seleção dos fragmentos completos pelo engenheiro de processos. Pode-se visualizar na Figura 17 o processo de validação da consistência modelado por meio de um diagrama BPMN (OMG, 2011).

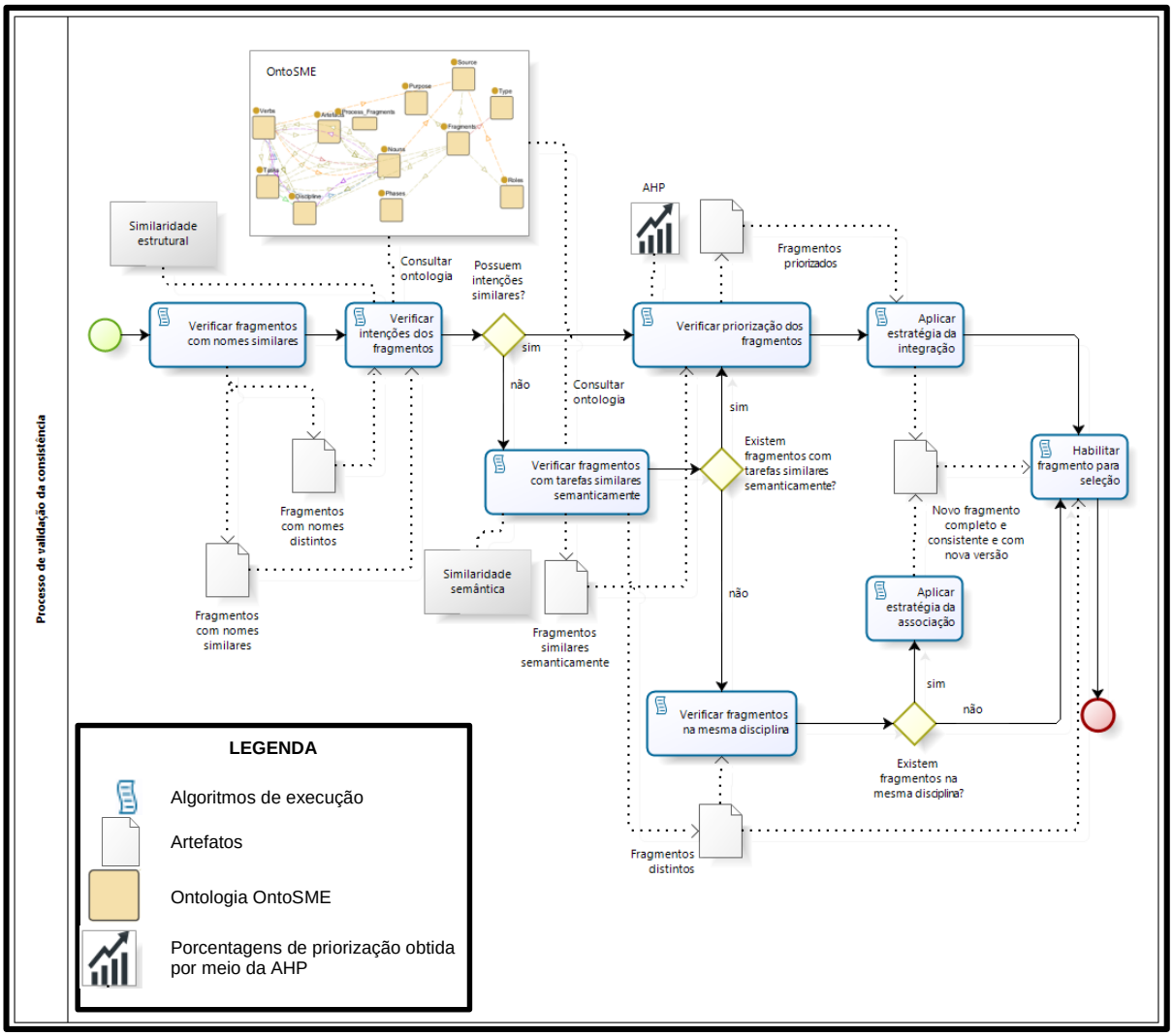



Figura 22 - Processo de validação da consistência e criação do novo fragmento.

Fonte: Autor.

Ao iniciar a execução do processo, por meio da atividade “Verificar fragmentos com nomes similares”, uma verificação para separar fragmentos com mesmo nome ou com nomes similares é efetuada formando dois conjuntos de fragmentos, o conjunto de fragmentos com nomes similares (artefato “Fragmentos com nomes similares”) e o conjunto de fragmentos com nomes distintos (artefato “Fragmentos com nomes distintos”).

Por sua vez, de posse do conjunto de fragmentos com nomes similares, executa-se a atividade “Verificar intenções dos fragmentos”, verificando se os fragmentos candidatos possuem ou não intenções similares. A identificação da intenção, neste trabalho, recebe o nome de Similaridade Estrutural, para afirmar se um fragmento é similar ao outro a estratégia deve:

i) Percorrer a estrutura do fragmento em avaliação;

ii) Extrair desta estrutura: a origem, a disciplina e as tarefas instanciadas;

iii) Percorrer cada tarefa do fragmento em avaliação;

iv) Consultar a ontologia OntoSME, descrita no Capítulo 3 desta dissertação, e executar a fórmula apresentada na Figura 11, Seção 3.4, passando como parâmetros: a) a disciplina; b) o verbo da tarefa em questão; e c) o substantivo que compõe a tarefa do fragmento avaliado.

v) Receber o resultado da consulta a fim de descobrir as intenções similares de cada tarefa do fragmento;

vi) Percorrer a estrutura dos demais fragmentos candidatos verificando se existem tarefas similares às tarefas do fragmento em avaliação;

vii) Agrupar os fragmentos com tarefas similares.

Caso no processo de reconhecimento de similaridade estrutural dos fragmentos existam fragmentos com intenções similares, cabe à estratégia construir um novo fragmento para compor o processo de software eliminando assim, as inconsistências. A construção deste novo fragmento por meio da identificação de intenções similares ocorre por meio da execução das atividades "Verificar priorização dos fragmentos" e "Aplicar estratégia da integração".

A atividade "Verificar priorização dos fragmentos" recupera os valores atribuídos no processo de validação da completude por meio da técnica AHP (VIDAL; MARLE; BOCQUET, 2011) e traz consigo uma lista de fragmentos priorizados. Por sua vez, a atividade "Aplicar estratégia de integração" faz uso da estratégia da integração (Seção 2.2.1) e das regras de qualidade (Seção 5.5) e cria o novo fragmento completo e consistente.

A identificação da similaridade estrutural associada à priorização de fragmentos é muito importante na etapa de criação do novo fragmento completo e consistente, uma vez que define quais tarefas devem permanecer no novo fragmento e quais devem ser excluídas. Outros elementos que são associados ao novo fragmento são os critérios de adaptação que deram origem ao novo fragmento e o contexto do projeto no qual foi criado. Após a criação do novo fragmento, este é versionado, salvo na base de métodos e habilitado para seleção por meio da atividade "Habilitar fragmento para seleção".

No entanto, se os fragmentos que compõem o conjunto de fragmentos com nomes similares não forem similares estruturalmente (intenção), realiza-se a verificação dos fragmentos com tarefas similares semanticamente (verbos e substantivos similares) por meio da execução da atividade "Verificar fragmentos com tarefas similares semanticamente". Esta

verificação consulta novamente a ontologia OntoSME (Capítulo 3) e faz uso da fórmula para medir a sinonímia dos verbos e substantivos (Figura 9 da Seção 3.4) que compõem as tarefas dos fragmentos. Caso os fragmentos sejam similares semanticamente, as atividades "Priorizar tarefas dos fragmentos com intenções similares" e "Aplicar estratégia da integração" são executadas e um novo fragmento completo e consistente é gerado e habilitado para seleção.

Caso os fragmentos candidatos não sejam similares semanticamente, é realizada uma última verificação usando a atividade "Verificar fragmentos com mesmo propósito". Nesta atividade, os fragmentos que possuem a mesma disciplina e o mesmo critério de adaptação são associados por meio da estratégia da associação (Seção 2.2.1) para criação de um novo fragmento que por sua vez é versionado, armazenado na base de dados e habilitado para seleção. A necessidade de associar estes fragmentos surgiu a fim de reduzir a complexidade na seleção dos fragmentos por parte do engenheiro de processos. Por sua vez, os fragmentos que concluíram todo o processo sem gerar um novo fragmento, são considerados completos e consistentes e habilitados para seleção.

A fim de exemplificar as atividades que compõem o processo de validação da consistência, considerou-se o exemplo definido na Seção 5.6 em que um projeto fictício foi definido e contextualizado. Desta forma, considerando que o engenheiro de processos selecionou os fragmentos mostrados por meio da Figura 20, o processo de validação da consistência apresentou os seguintes resultados:

i) Para os fragmentos de nome "*Define a Candidate Architecture*" foram identificados que ambos os fragmentos além de possuir nomes similares, possuem intenções similares. Neste caso, cabe à estratégia construir um novo fragmento integrando às características do fragmento melhor priorizado (primeiro fragmento com grau de priorização de 38,884%) com as características do segundo fragmento (grau de priorização igual a 19,553%). Pode-se visualizar por meio da Figura 23 os referidos fragmentos e na Figura 24 a integração realizada para construção do novo fragmento a partir da seleção de fragmentos com nomes e intenções similares.

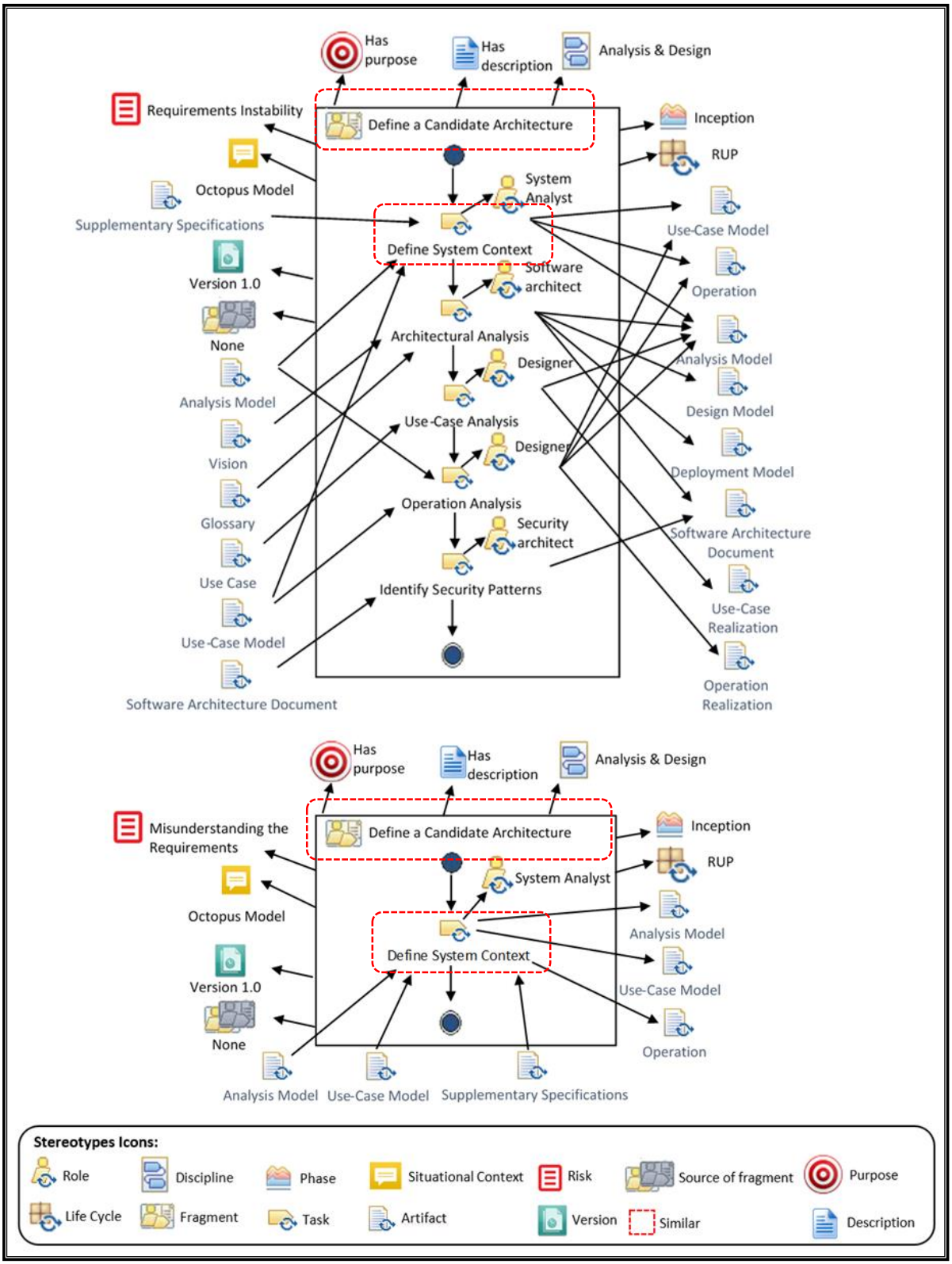



Figura 23 - Construção de um novo fragmento a partir de fragmentos de mesmo nome.

Fonte: Autor.

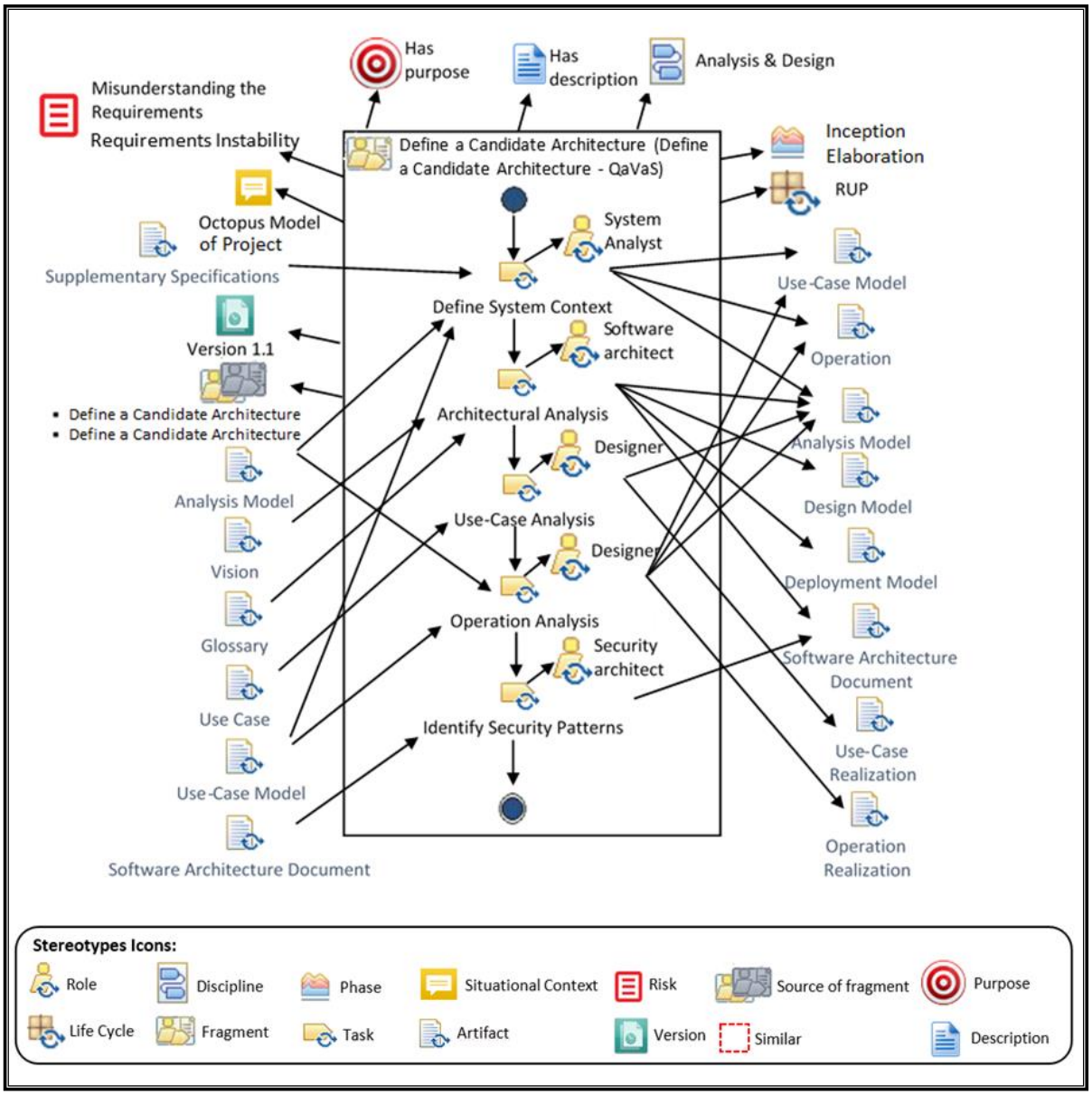



Figura 24 - Novo fragmento construído por meio do processo de validação da consistência.

Fonte: Autor.

ii) Para os fragmentos com nomes distintos "*Analyze the Problem*" (38,886% de priorização) e "*Write User Story*" (1,447% de priorização) foi identificado mediante o processo de validação da consistência que os mesmos possuem tarefas similares em suas estruturas. Conforme pode-se visualizar por meio da Figura 25, a ontologia OntoSME identificou que as tarefas "*Capture a Common Vocabulary*" e "*Write User Story*" satisfazem a mesma intenção, portanto, caso os dois fragmentos permaneçam no processo, o mesmo possuirá inconsistência. Neste caso, coube ao processo de validação da consistência construir um novo fragmento, como mostra a Figura 26.

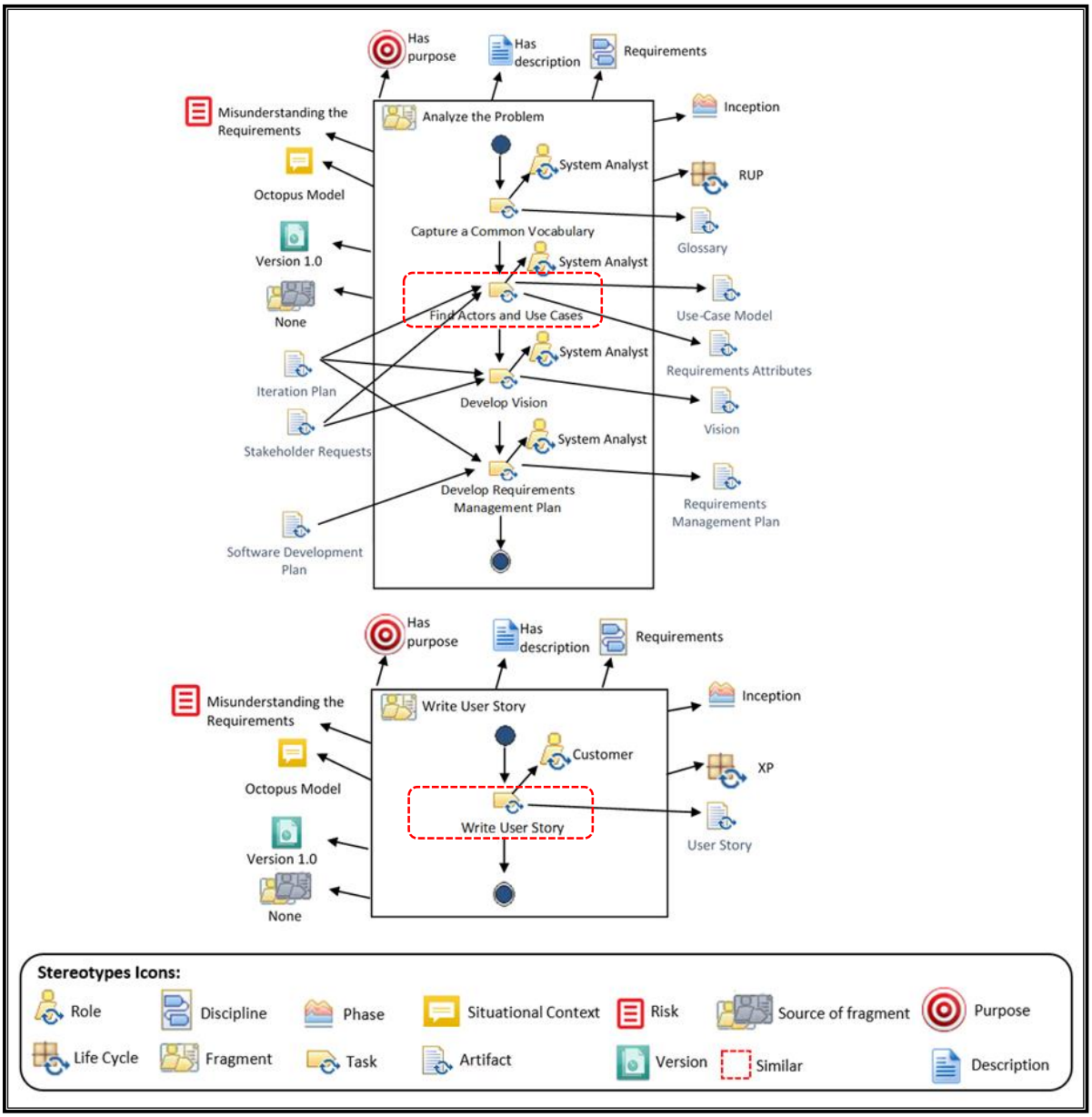



Figura 25 - Fragmentos de tarefas similares e nomes distintos.

Fonte: Autor.

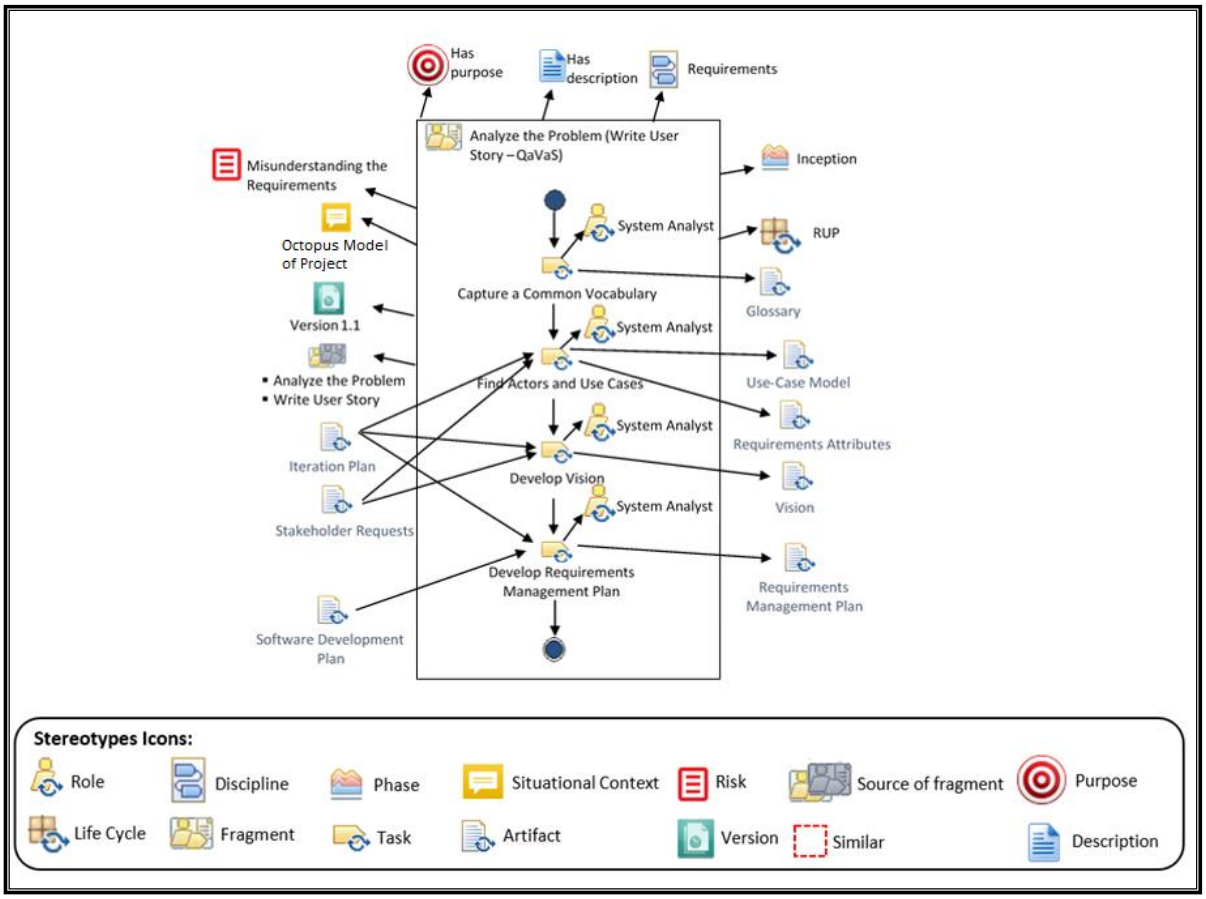



Figura 26 - Construção de um novo fragmento a partir de fragmentos de nomes distintos.

Fonte: Autor.

iii) A fim de exemplificar a similaridade semântica, pode-se citar a tarefa "*Develop Vision*" pertencente ao fragmento modelado em SPEM na Figura 26. Neste caso, se o processo de validação da consistência não encontrar tarefas similares estruturalmente, ou seja, de mesmo conceito, e encontrar em outro fragmento candidato a tarefa "*Envolve Vision*" cadastrada aleatoriamente, cabe ao processo da consistência eliminar esta duplicidade por meio da estratégia de integração.

iv) Por fim, para exemplificar a criação de um novo fragmento usando a atividade "Verificar fragmentos de mesma disciplina", considera-se a inclusão na lista de fragmentos completos candidatos do fragmento "*Design Database*" (Figura 27). Neste caso, o processo de validação da consistência não identificando similaridades nos fragmentos candidatos, verifica se os fragmentos avaliados estão na mesma disciplina.

v) Em caso positivo associa os referidos fragmentos a fim de reduzir a complexidade na etapa de adaptação de processos. Pode-se visualizar na Figura 28 os fragmentos

candidatos que possuem a mesma disciplina e na Figura 29, um exemplo de fragmento criado por meio desta atividade.

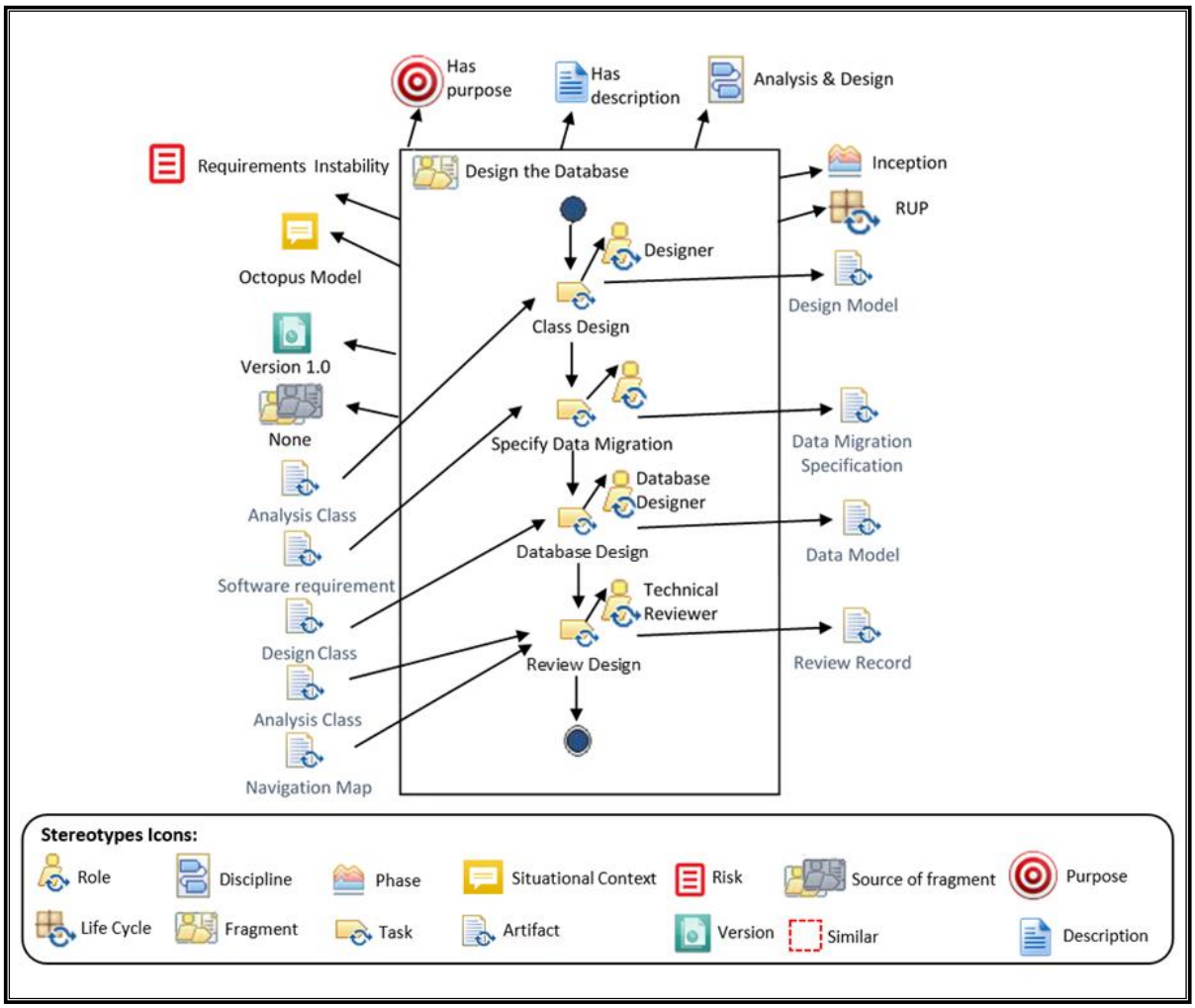



Figura 27 - Fragmento "*Design Database*".

Fonte: Autor.

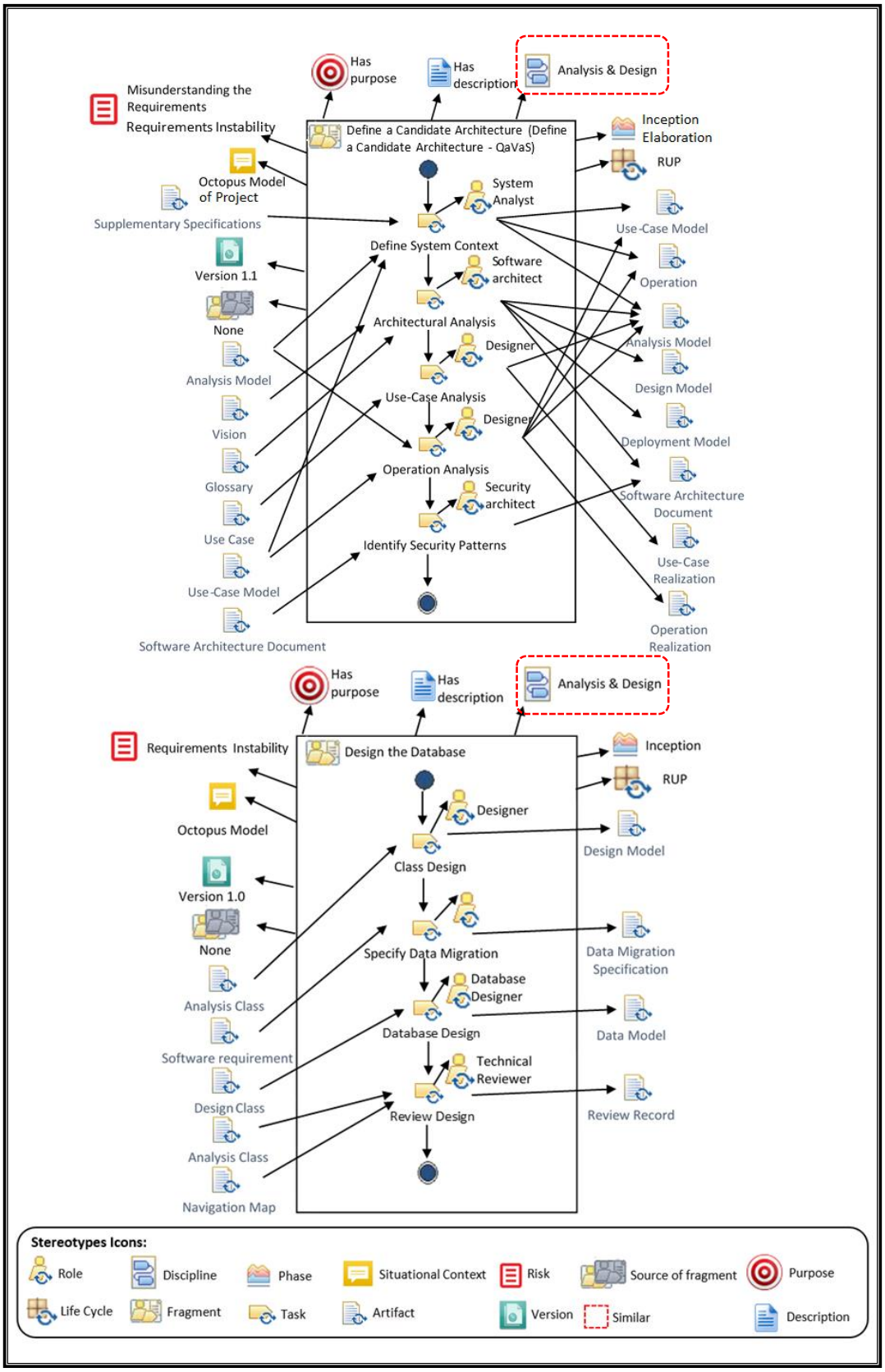



Figura 28 - Fragmentos candidatos de mesma disciplina.

Fonte: Autor.

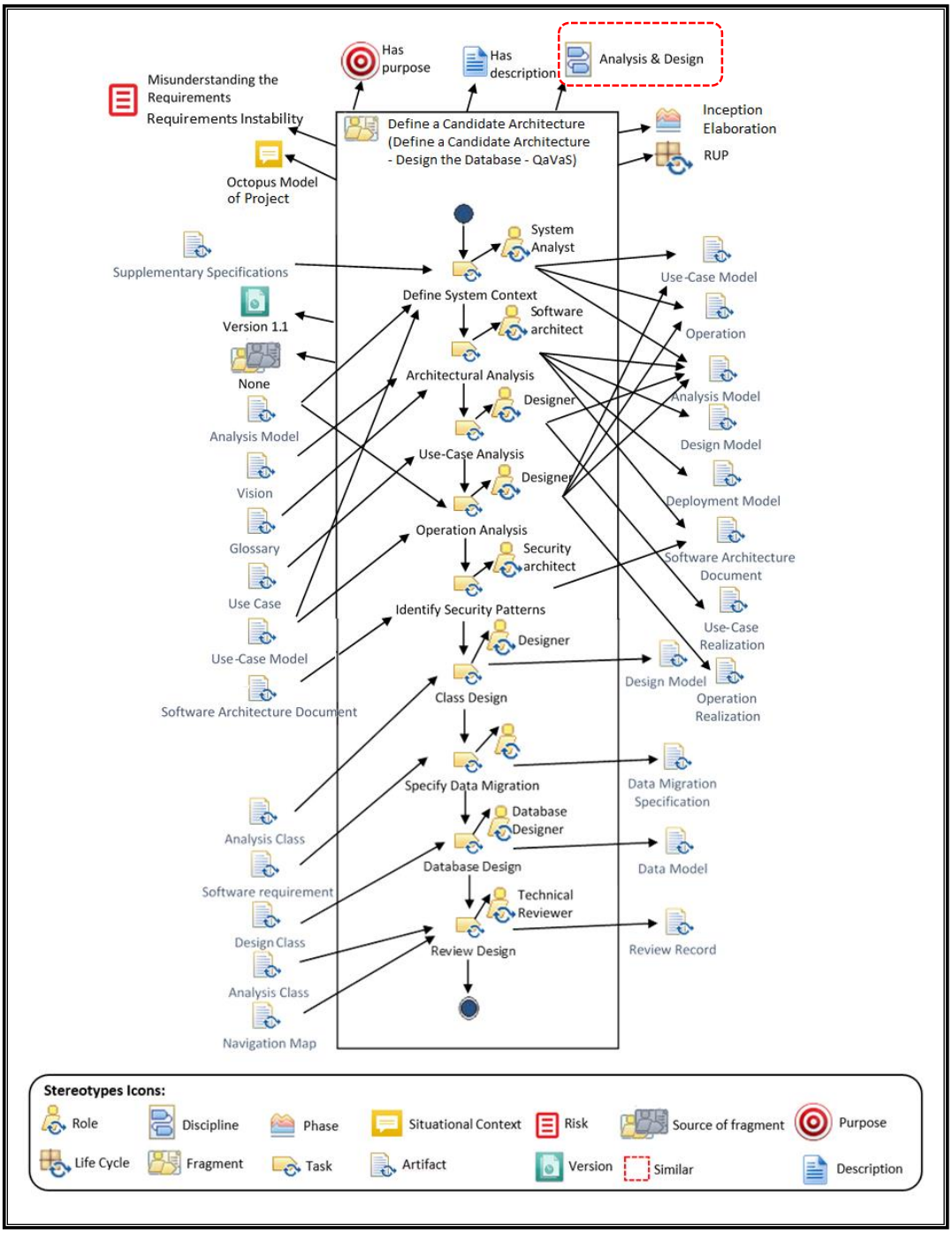



Figura 29 - Novo fragmento construído a partir de fragmentos de mesma disciplina.

Fonte: Autor.

Salienta-se ainda que todas as atividades que compõem a sistemática da validação da consistência não devem ser executadas de forma *ad-hoc* e que o apoio ferramental se faz necessário para avaliar a estratégia QaVaS. Dessa forma, uma ferramenta denominada MfTP Tool foi criada para apoiar e validar a abordagem QaVaS.

# 6 FERRAMENTA *METAMODEL FOR TAILORING PROCESS*

A fim de validar e automatizar a estratégia proposta, um sistema modularizado foi construído. O referido sistema apresenta dois módulos, que são: o módulo principal e o módulo de validação da estratégia QaVaS.

O módulo principal possibilita o cadastro dos elementos usados para elaboração de processos segundo o metamodelo MfTP. Por sua vez, o módulo da estratégia proposta tem a função de guiar o engenheiro de processos na adaptação de processos a partir da construção e reutilização de fragmentos completos e consistentes.

Cabe ainda descrever que ambos os módulos foram desenvolvidos utilizando a linguagem Java para especificação das regras de negócio, *Java Persistence API* (JPA) e *Hibernate* para persistência e acesso a dados; Java Server Faces (JSF), HTML, CSS3 e *JQuery* para compor a camada de apresentação e o banco de dados MySQL para persistir a base de dados da ferramenta. A integração da ferramenta com a ontologia OntoSME, detalhada no Capítulo 3, deu-se por meio do *framework* Jena (JENA, 2007).

## 6.1 *Módulo principal*

O módulo principal contém a sistemática proposta neste trabalho para adaptação de processos a partir da construção e reutilização de fragmentos que sejam completos e consistentes. Por meio deste módulo é possível definir as características funcionais do projeto, por exemplo, critérios de adaptação e contexto do projeto, bem como manter todo conhecimento necessário para adaptar ou criar um processo de desenvolvimento de software.

Pode-se visualizar na Figura 30, a tela principal da ferramenta MfTP Tool. No menu da esquerda são exibidas as principais funcionalidades do sistema que são divididas em cinco grupos: i) *Organization*; ii) *Knowledge Base*; iii) *Contextualization*; iv) *Tailoring Criteria*; e v) *Settings*.

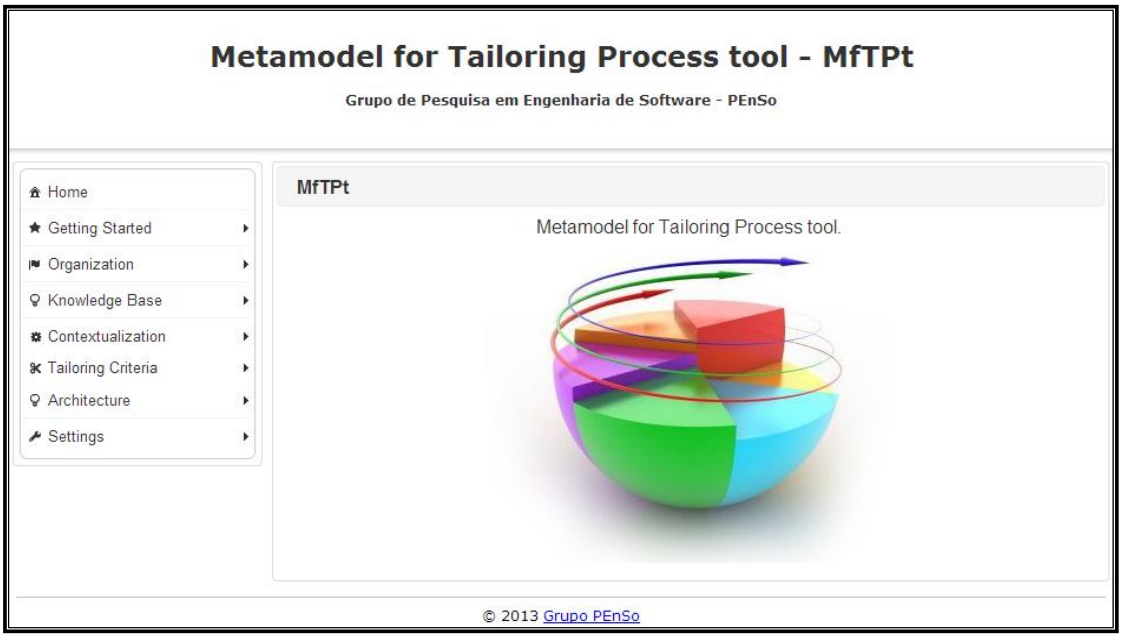



Figura 30 - Módulo principal do sistema (tela inicial).

Fonte: Autor.

No grupo "*Organization*" pode-se cadastrar uma nova organização (link *Organization*), criar um novo projeto e caracterizá-lo (link *Project*) e criar novas versões para um processo já definido para a organização (link *Process*). Pode-se visualizar na Figura 31, o grupo "*Organization*" com suas respectivas opções.

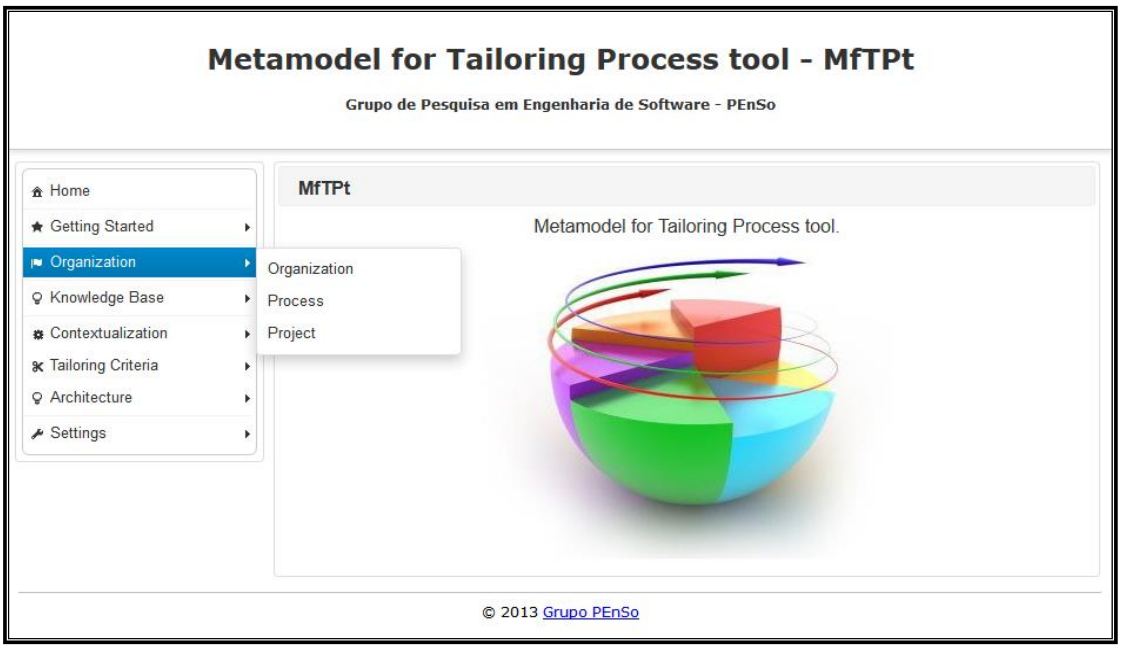



Figura 31 - Módulo principal do sistema, grupo "*Organization*".

Fonte: Autor.

No grupo "*Knowledge Base*" pode-se cadastrar uma nova atividade (*link Activity*), neste trabalho denominado fragmento, persistir novos artefatos (*link Artifact*), disciplinas (*link Discipline*), ciclo de vida (*link Life Cycle*), fases do processo (*link Phase*), papéis (*link Role*), origem (*link Source*), tarefas (*link Task*) e ações que descrevem o relacionamento entre os artefatos e as tarefas (*link Action*). Pode-se visualizar na Figura 32, o grupo *Knowledge Base* com suas respectivas opções.

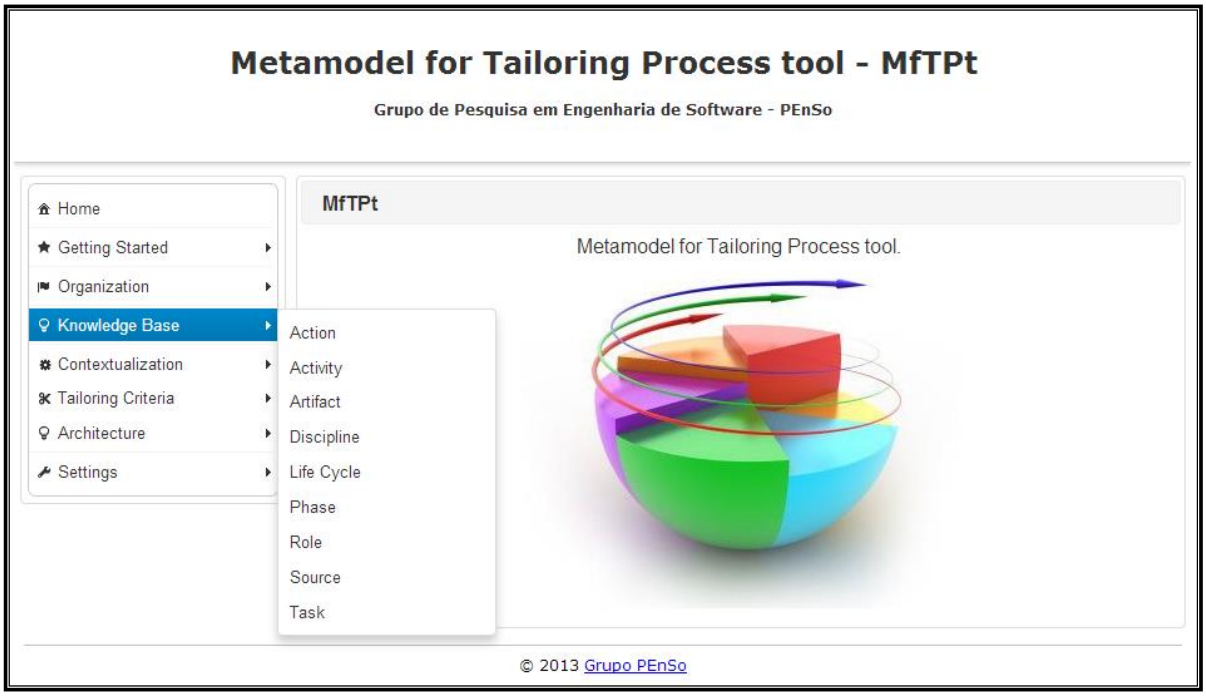



Figura 32 - Módulo principal do sistema – grupo "*Knowledge Base*".

Fonte: Autor.

O grupo "*Contextualization*" descreve as funcionalidades referentes à caracterização do projeto. Na opção tipo do contexto (*link Context Type*) são definidos tipos de contexto, por exemplo, *Octopus Model*. O Contexto (*link Context*) define os fatores para cada tipo de contextualização, por exemplo, para o Octopus Model "*Size*", "*Criticity*", "*Business Model*", entre outros. O valor do contexto (*link Context Value*) define os valores para cada fator do contexto, por exemplo, para "*Business Model*" tem-se os valores "*In house*", "*Commercial*" e "*Large System Component*" (sob medida para um cliente, comercial e componente de um grande sistema). Pode-se observar na Figura 33, o grupo "*Contextualization*"" com suas respectivas opções.

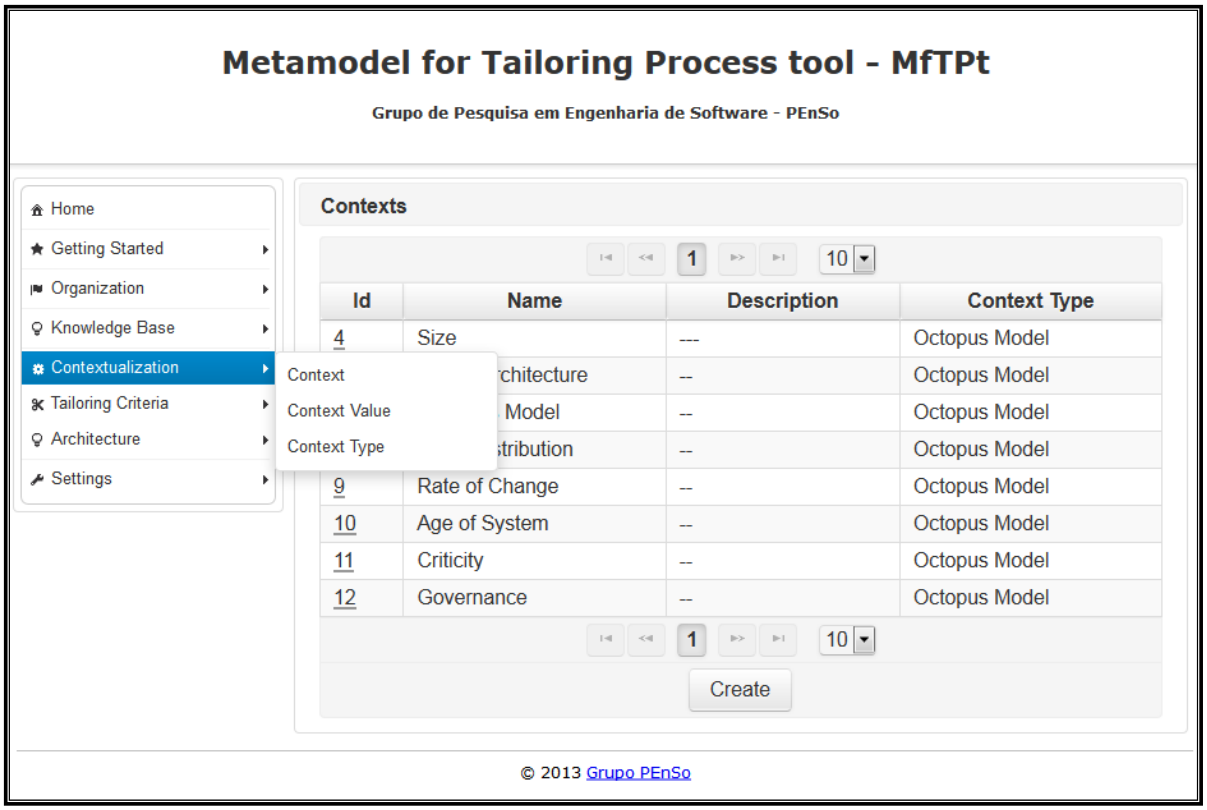



Figura 33 - Módulo principal do sistema – grupo "*Contextualization*".

Fonte: Autor.

A funcionalidade responsável por persistir os critérios de adaptação utilizados para recuperar os fragmentos de processo é dada pelo grupo "*Tailoring Criteria*". Neste grupo é possível persistir informações como os tipos de critérios de adaptação (link *Criteria Type*) e os critérios de adaptação (link *Criteria*).

Em "*Criteria Type*", é efetuado o cadastramento dos diferentes tipos de requisitos de adaptação, por exemplo, riscos, CMMI, MPS.BR, entre outros. Em "*Criteria*", define-se os diferentes requisitos para cada tipo de requisitos de adaptação, por exemplo, para MPS.BR tem-se os seus respectivos níveis. Pode-se observar na Figura 34 o grupo "*Tailoring Criteria*" com suas respectivas opções.

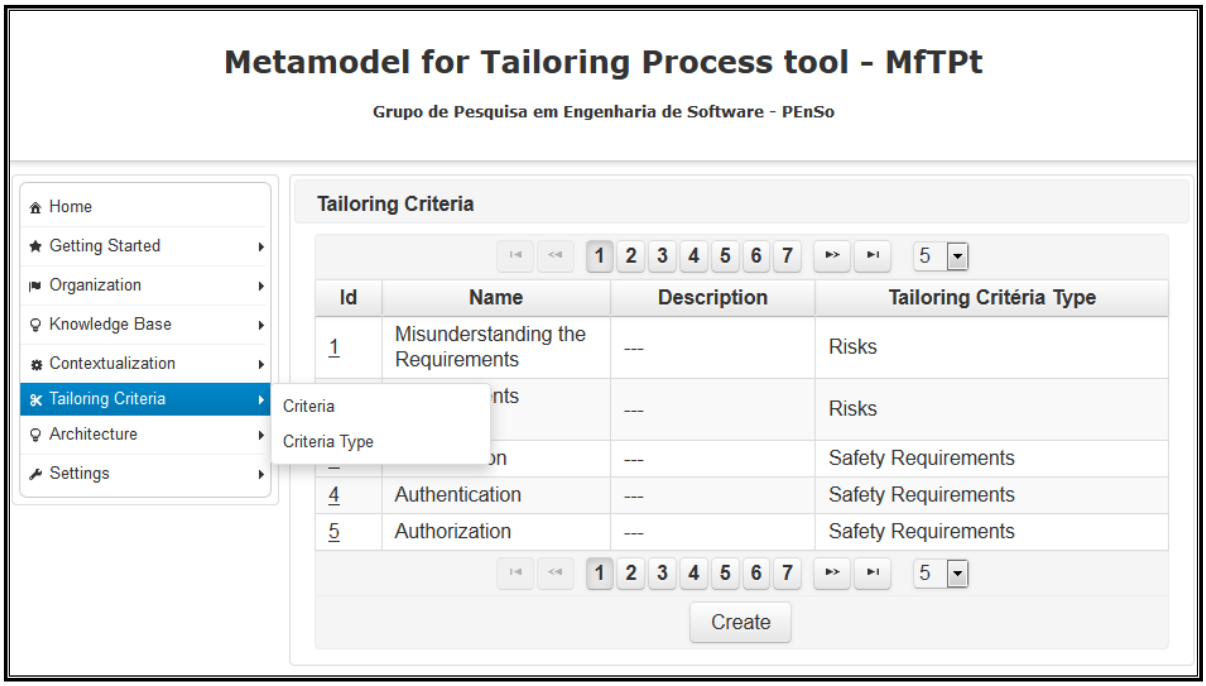



Figura 34 - Módulo principal do sistema – grupo "*Tailoring Critéria*".

Fonte: Autor.

Por fim, o grupo "*Settings*" descreve as funcionalidades relacionadas aos usuários e controle de acesso. Neste grupo é possível inserir novos usuários para utilizar a ferramenta (*link User*), bem como controlar o nível de acesso dos usuários criados (*link Autority*). Pode-se observar por meio da Figura 35 o grupo "*Settings*" e suas respectivas funcionalidades.

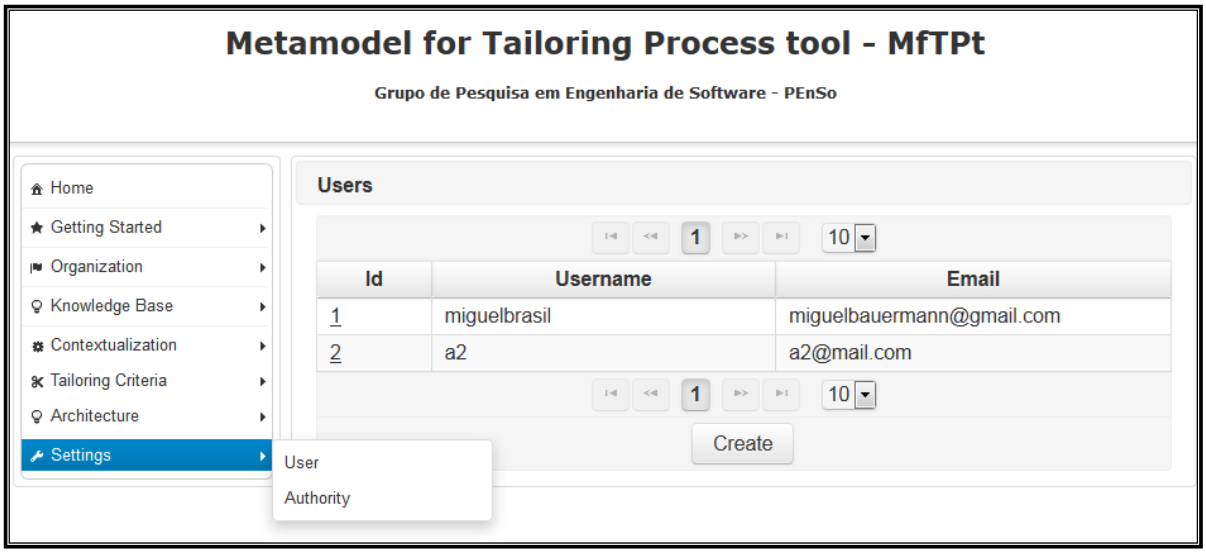



Figura 35 - Módulo principal do sistema – grupo "*Settings*".

Fonte: Autor.

No entanto, para que a ferramenta seja utilizada de acordo com a abordagem proposta, torna-se necessário primeiro realizar algumas configurações. O primeiro passo é definir os elementos de processo que serão utilizados por meio do grupo "*Knowledge Base*". Pode-se visualizar na Figura 36, o início da criação de uma atividade, considerada neste trabalho um fragmento de processo de software.

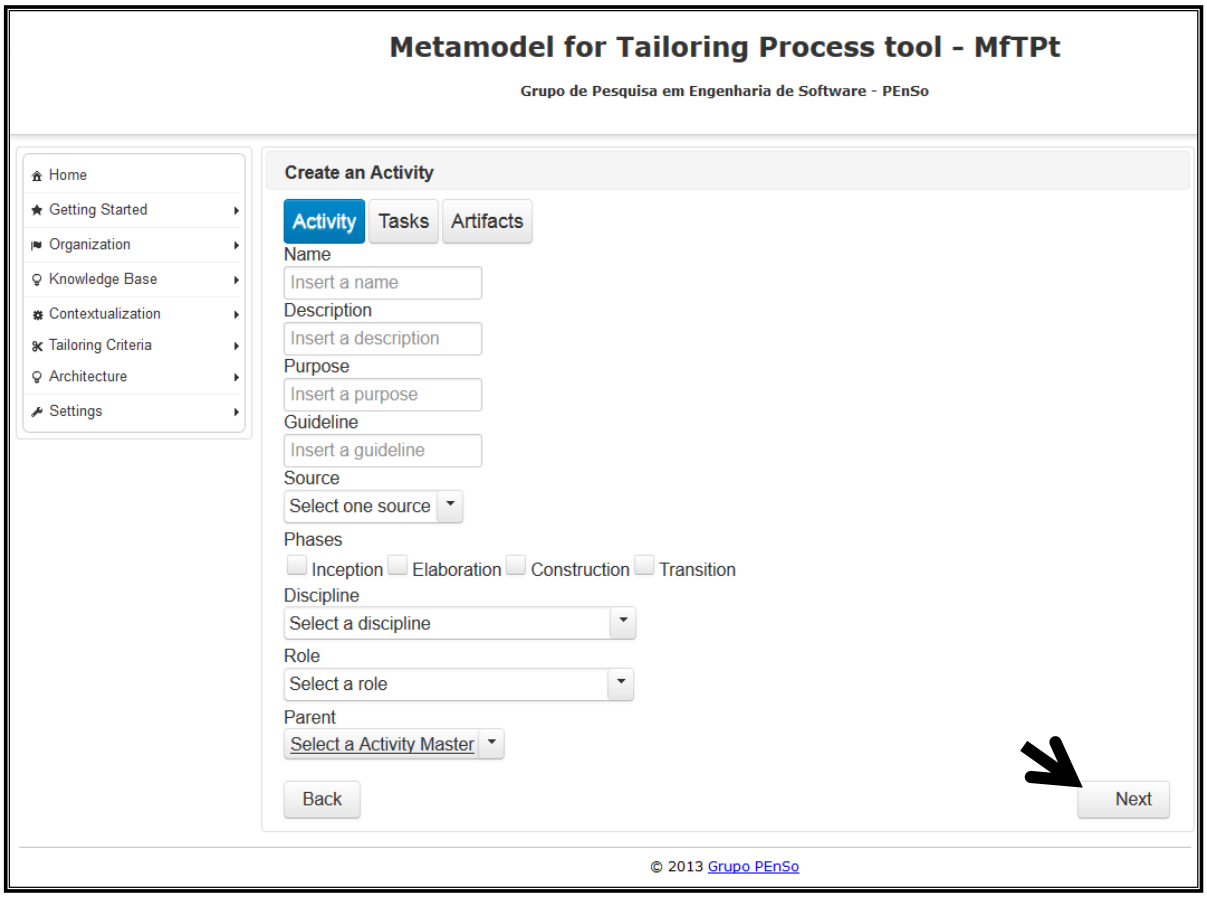



Figura 36 - Primeiro passo para criação de uma atividade (fragmento de processo).

Fonte: Autor.

A segunda etapa, como mostra a Figura 37, é selecionar as tarefas que compõem o fragmento de processo. É importante salientar que tais tarefas já possuem em suas instâncias o relacionamento com os papéis envolvidos para realizá-las, basta somente vinculá-las ao fragmento e informar no terceiro passo que artefatos são produzidos por meio deste relacionamento.

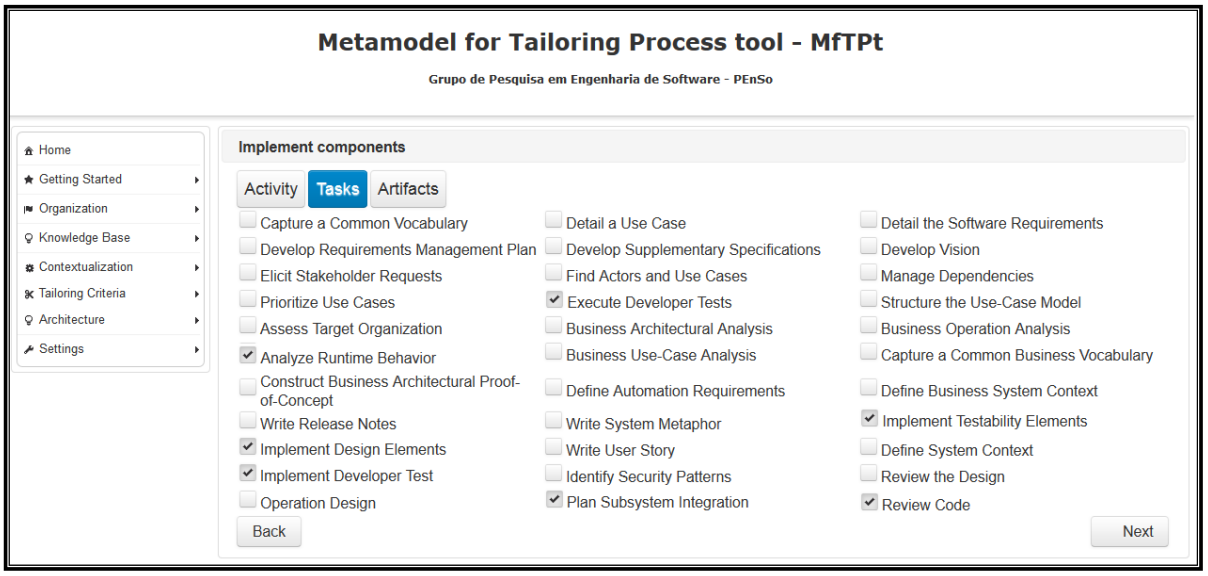



Figura 37 - Seleção das tarefas vinculadas ao fragmento.

Fonte: Autor.

O vínculo das atividades (fragmentos), das tarefas e dos artefatos produzidos por cada tarefa é realizado no terceiro passo. Nesta etapa, são exibidas ao engenheiro de processos as tarefas selecionadas para a atividade em questão e um link no qual a associação entre atividades, tarefas e artefatos é realizada. Na Figura 38, pode-se visualizar a tela com as tarefas selecionadas para a atividade "*Implement Components*".

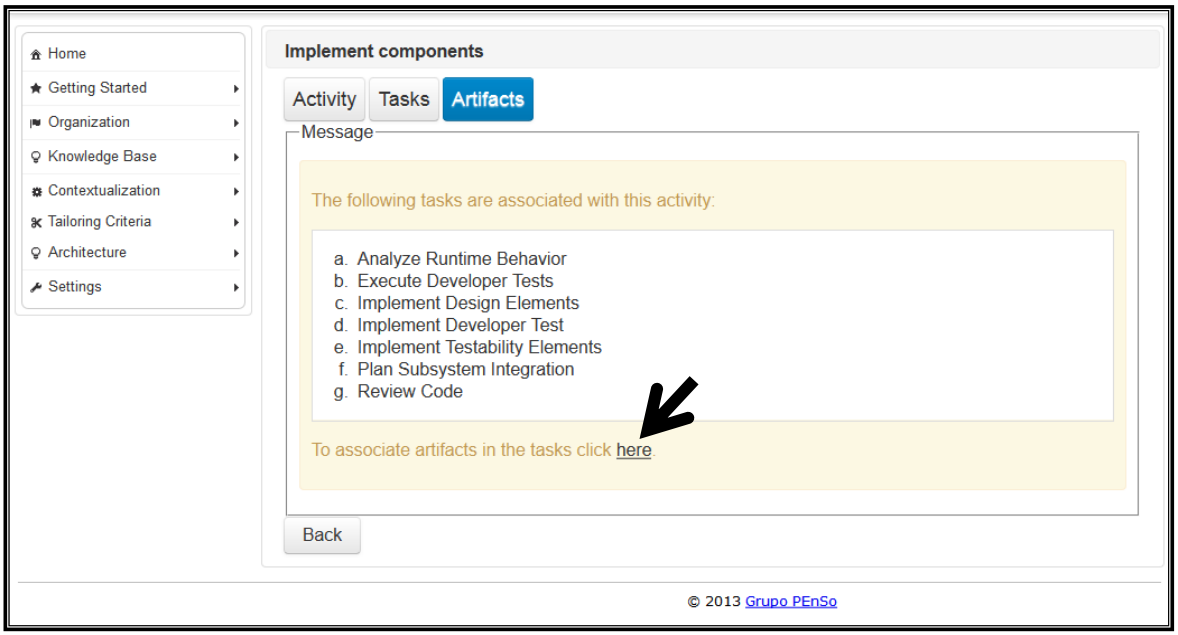



Figura 38 - Tela com tarefas selecionadas para atividade "*Implement Components*".

Fonte: Autor.

Ao clicar no link marcado com uma seta na Figura 38, o engenheiro de processos é direcionado para a tela onde a associação entre as atividades, fragmentos e artefatos é realizada. Para cada atividade é vinculada as suas respectivas tarefas com seus respectivos artefatos. Pode-se visualizar por meio da Figura 39 a tela de ligação entre atividades, tarefas e artefatos.


Figura 39 - Associação entre atividades, tarefas e artefatos.

Fonte: Autor.

O vínculo das atividades, das tarefas e dos artefatos produzidos por cada tarefa é realizado no terceiro passo. Nesta etapa, para cada tarefa selecionada pelo engenheiro de processos um ou mais artefatos de entrada, saída ou entrada e saída podem ser ligados diretamente com a tarefa.

A definição do contexto do uso do fragmento, bem como os critérios de adaptação que o fragmento satisfaz é realizada por meio dos *links* "*Edit context this activity*" e "*Edit taliloring critéria this activity*", respectivamente. Os fragmentos provenientes do processo padrão da organização podem ser inseridos na tela de critérios de adaptação permitindo que o engenheiro de processos recupere o processo padrão e por sua vez possa adaptá-lo às características do projeto ou escolha desconsiderar o processo padrão criando um novo processo. Pode-se visualizar na Figura 40, a tela de definição do contexto do uso de fragmentos.

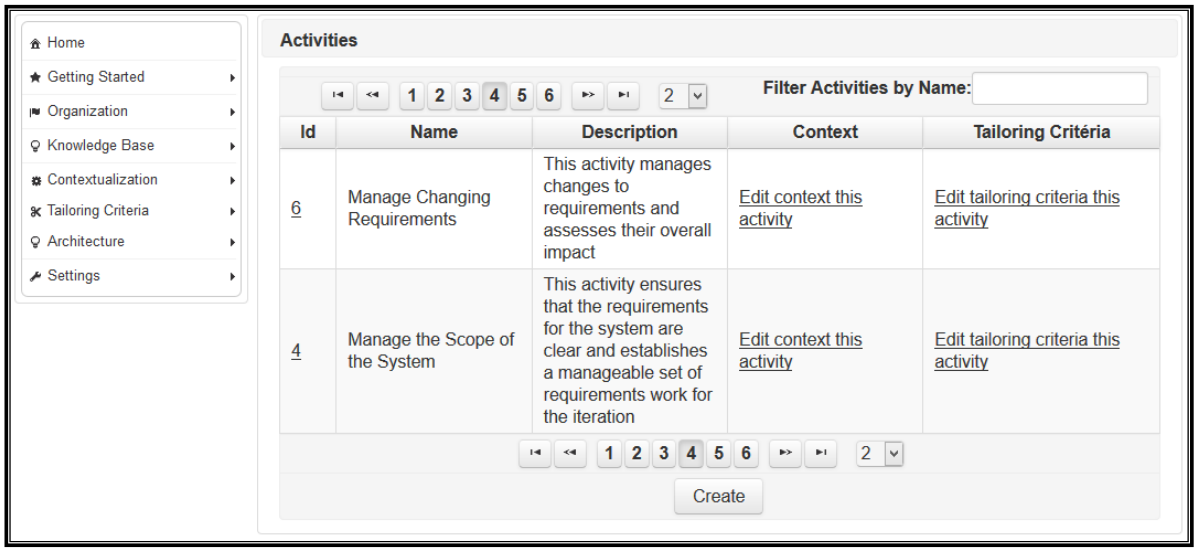



Figura 40 - Definição do contexto e critérios de adaptação para uma atividade.

Fonte: Autor.

No entanto, o módulo base por si só não é suficiente para a validação da estratégia proposta uma vez que foi desenvolvido para inserir e manter os elementos de processo e dar suporte a abordagens de adaptação de validação de processos de software. Tornou-se necessário a construção de outro módulo para validação da completude e consistência em elementos de processo.

## 6.2 *Módulo para definição de processos completos e consistentes*

A fim de validar a proposta deste trabalho, foi desenvolvido um módulo para definição de processos completos e consistentes. O objetivo deste módulo, é validar a estratégia QaVaS por meio da adaptação ou construção de um processo voltado a atender às características específicas de um projeto.

Para atender a necessidade de adaptar ou construir um processo específico e assim validar a estratégia QaVaS, optou-se por utilizar a abordagem OSPTA. Um processo será adaptado na ferramenta quando, a partir do processo padrão da organização, novos fragmentos de processo forem agregados ao processo de acordo com o contexto e os critérios de adaptação selecionados para o projeto. Por sua vez, um processo será construído no sistema, quando o processo padrão da organização for desconsiderado pelo engenheiro de processos e novos fragmentos candidatos forem recuperados e selecionados.

Desta forma, conforme apresentado na Seção 5.1 deste trabalho, o processo da validação da estratégia é executado a partir da seleção de fragmentos na base de dados, ou seja, após o engenheiro de processo ter informado o projeto e as suas caraterísticas.

Pode-se visualizar na Figura 41, os passos utilizados para a validação da estratégia QaVaS representados na forma de um *Wizard* de configuração. Optou-se por adotar o *layout* de *Wizard* por propiciar uma forma intuitiva para guiar o engenheiro de processos na tarefa de criar ou adaptar um processo conforme mostra os trabalhos de Brasil, Fontoura e Alvaro (2013) e Brasil, Pereira e Fontoura (2012) com foco na adaptação e melhoria de processos de software.

A primeira etapa, conforme se pode visualizar na Figura 41, consiste na atividade de selecionar o projeto e recuperar da base de dados o contexto definido no momento da criação do projeto.

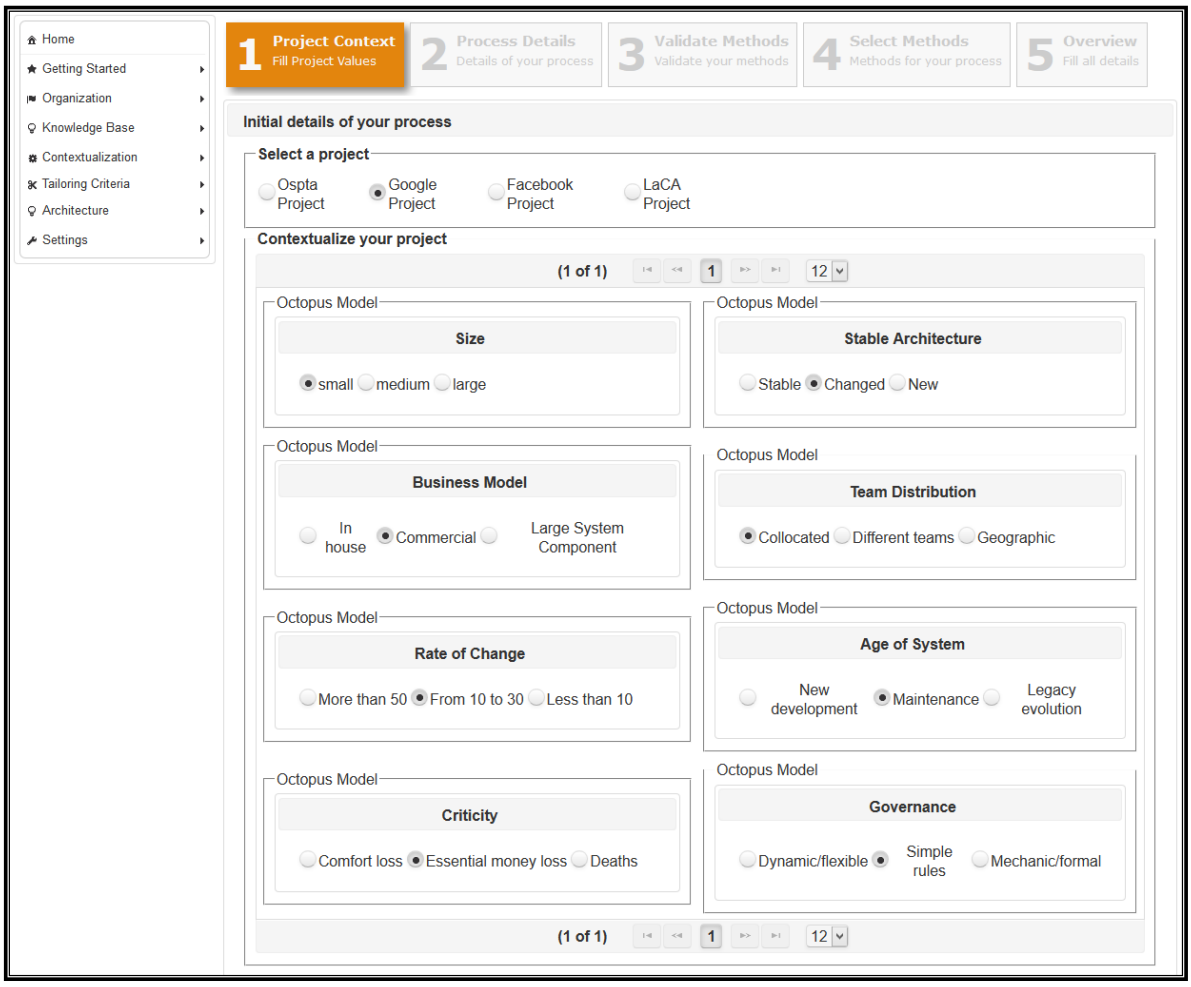



Figura 41 - Módulo para validação da estratégia QaVaS.

Fonte: Autor.

A etapa de número dois, conforme se pode observar na Figura 42, consiste em nomear o processo por meio do campo "*Name your process*" e selecionar os critérios de adaptação para o projeto e o processo padrão da organização na região denominada "*Tailoring criterias of your process*". Caso não seja informado um nome para o processo, por definição, o nome do processo será denominado por meio da concatenação da palavra "*Process*" acrescido da data e hora atual.

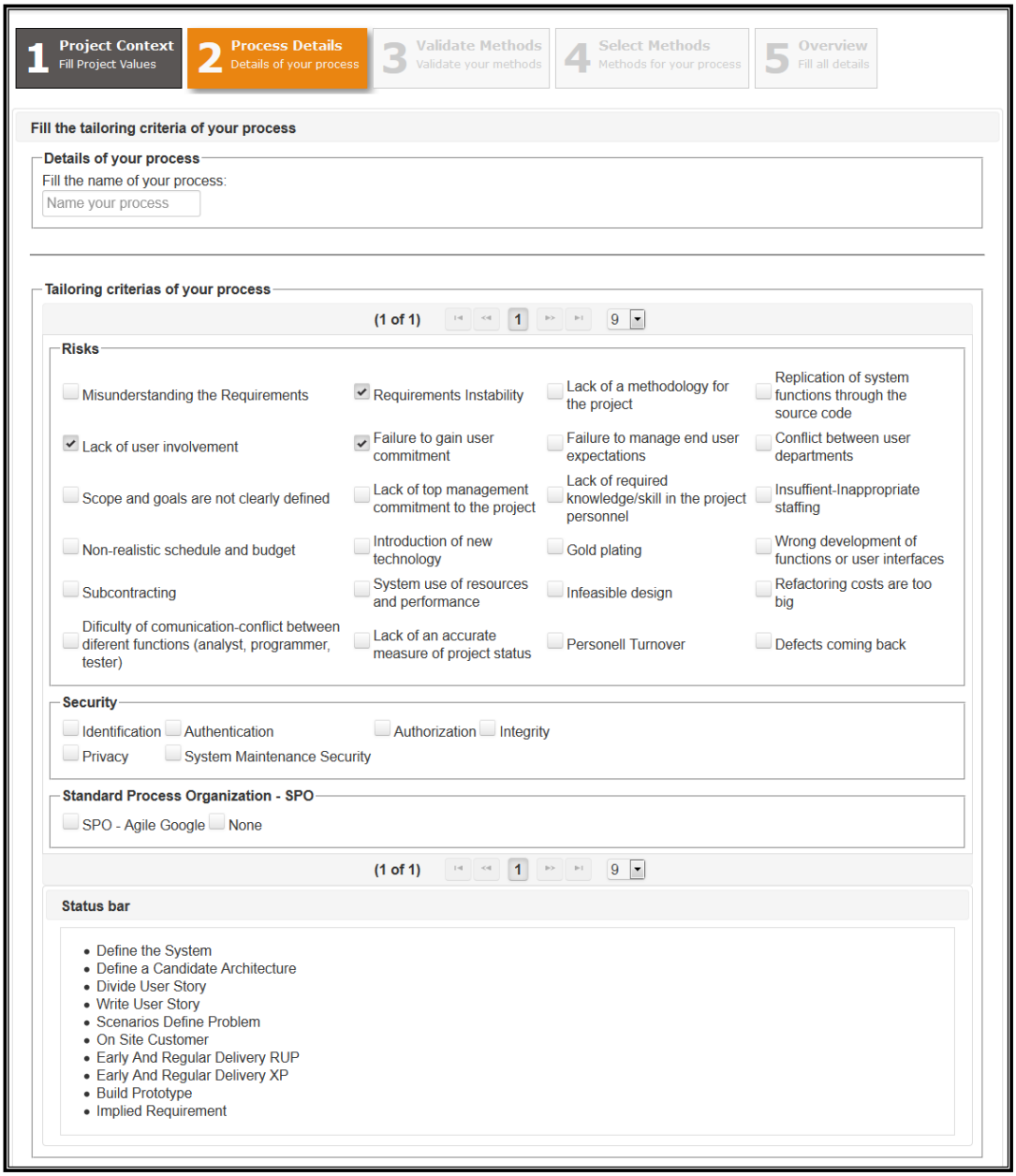



Figura 42 - Segunda etapa do módulo de validação da estratégia QaVaS.

Fonte: Autor.

A tela da segunda etapa (Figura 42) conta também com uma barra de status denominada "*Status bar*", cuja função é fornecer ao engenheiro de processos uma visão prévia dos fragmentos recuperados da base de métodos.

Na terceira etapa a validação da estratégia QaVaS tem início. Com base nas proposições definidas na Seção 5.6 deste trabalho, para cada fragmento recuperado da base de métodos, a validação da completude é realizada. Na tela exibida por meio da Figura 43, os fragmentos recuperados são expostos em uma tabela contendo quatro colunas.

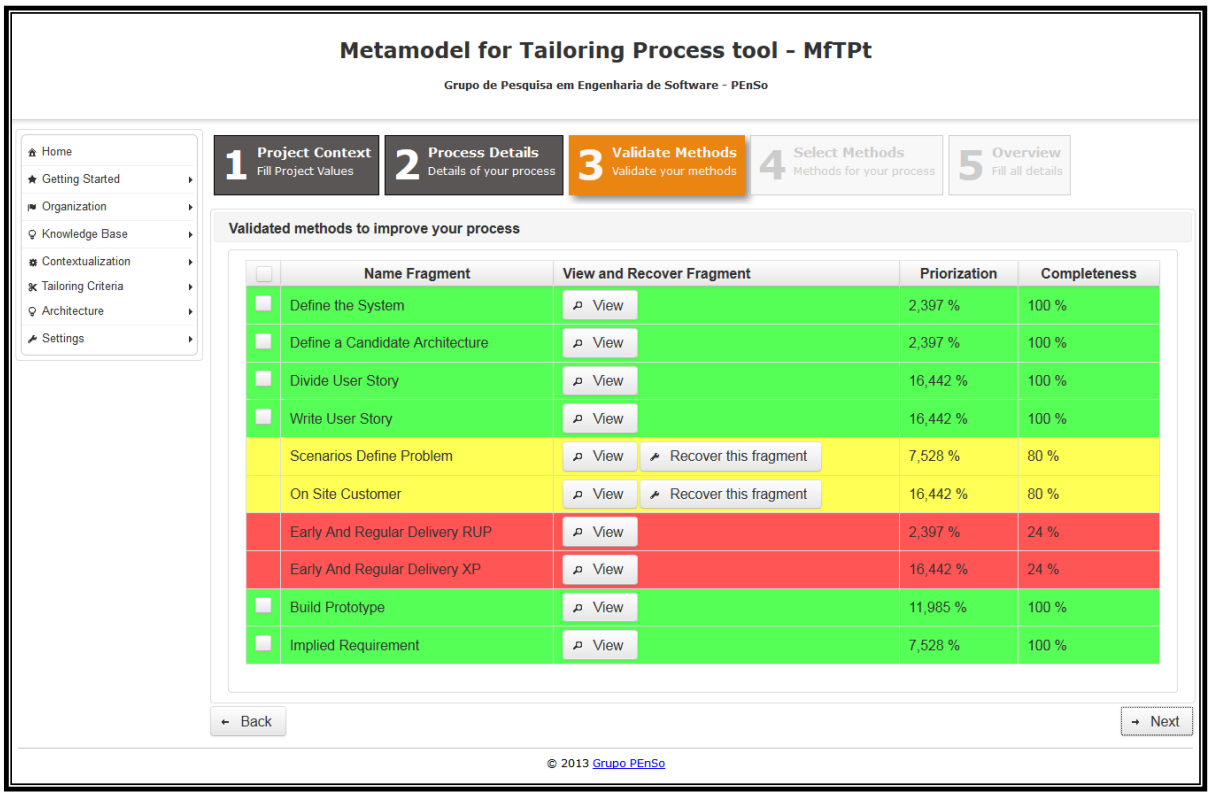



Figura 43 - Execução da estratégia QaVaS.

Fonte: Autor.

A primeira coluna possui a funcionalidade de seleção dos fragmentos mediante uma "caixa de seleção". A segunda coluna, denominada "*Name fragment*", apresenta o nome do fragmento selecionado. Por sua vez, a coluna denominada "*View and recover fragment*" possui as funcionalidades de visualizar o fragmento em questão e recuperá-lo quando for o caso.

A coluna denominada "*Completeness*" informa o grau de completude dos fragmentos avaliados pela estratégia QaVaS. Conforme descrito na Seção 5.6, fragmentos com

completude inferior a 60% ou que não atenderam aos itens eliminatórios da estratégia da completude devem ser descartados. Tais fragmentos na ferramenta são hachurados em vermelho (fundo vermelho) e desabilitados, excluindo-os do processo de seleção. Cabe a estes fragmentos descartados apenas serem visualizados na ferramenta por meio do botão "*View*".

Os Fragmentos incompletos com média igual ou superior a 60% recebem marcação amarela e, para estes, é exibida a funcionalidade de recuperação através do botão "*Recover this fragment*". Por fim, fragmentos completos recebem destaque em verde e são habilitados para seleção por meio de uma caixa de seleção situada ao lado esquerdo do nome. Por meio desta caixa de seleção, o engenheiro de processo pode selecionar os referidos fragmentos e dar início à etapa de validação da consistência.

A Figura 44 e a Figura 45 mostram os detalhes de um fragmento (botão "*View*") e o processo de recuperação de um fragmento (botão "*Recover this fragment*"), respectivamente.

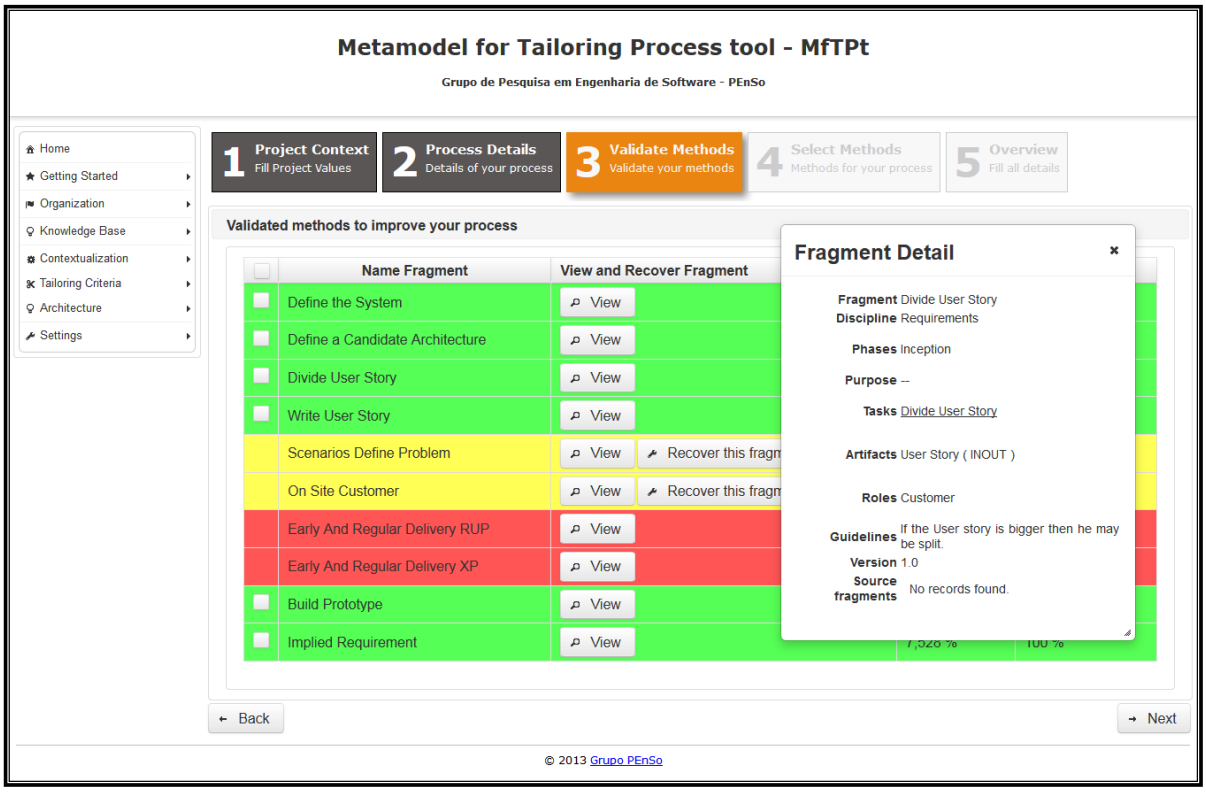



Figura 44 - Detalhes do fragmento usando o botão "*View*".

Fonte: Autor.

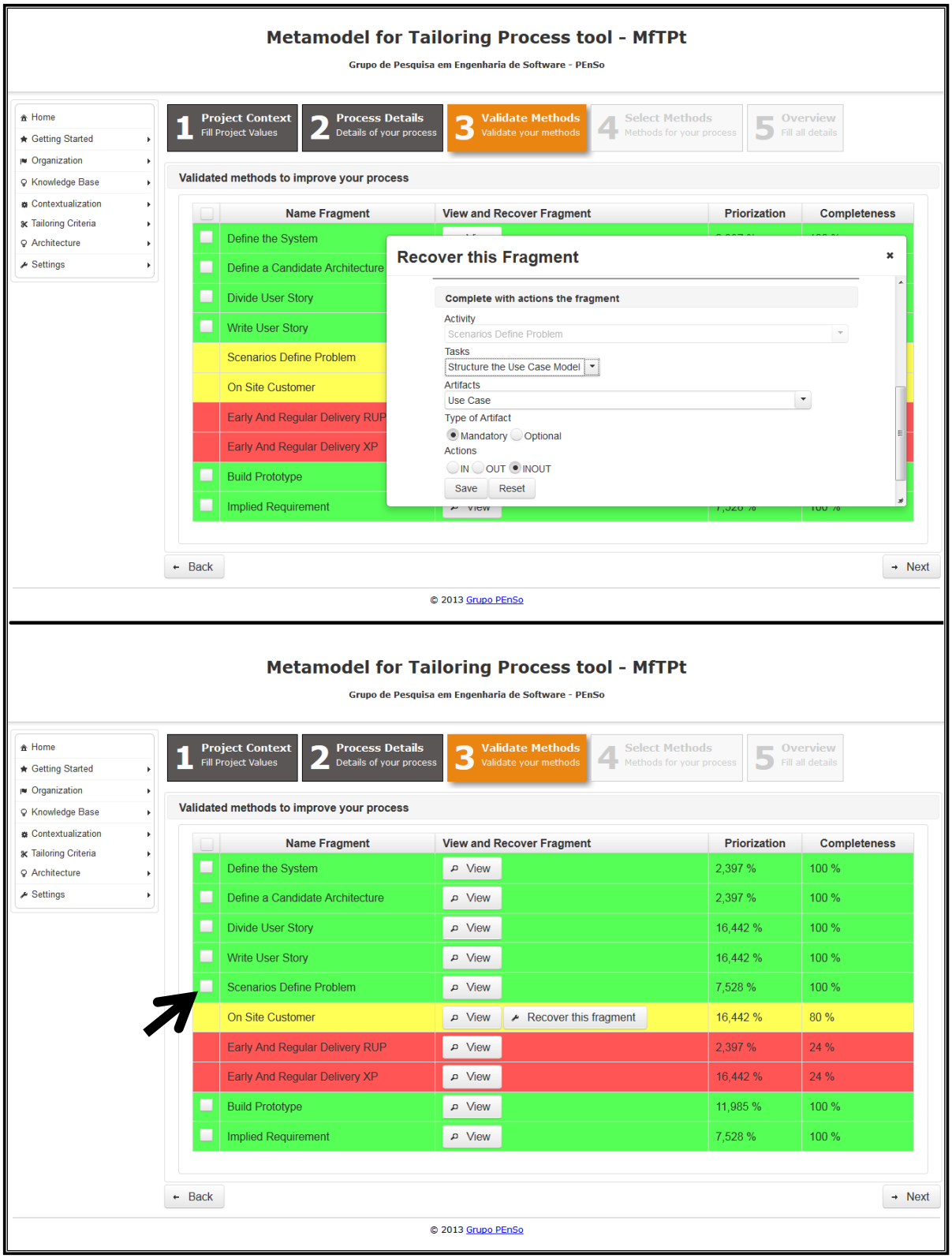



Figura 45 - Processo de recuperação de um fragmento incompleto.

Fonte: Autor.

A restauração de um fragmento é realizada ainda na terceira etapa, preenchendo-se os elementos que faltam no referido fragmento usando o botão "*Recover fragment*". Por meio da funcionalidade de visualizar o fragmento é possível verificar as partes que faltam do fragmento, estas partes faltantes são representadas por meio do ícone de uma ferramenta (*modal popup* na Figura 46). Após restaurar os fragmentos passíveis de recuperação, os fragmentos antes incompletos com destaque em amarelo, passam novamente pela estratégia de validação da consistência recebendo realce em verde se forem considerados completos.

Os fragmentos completos são disponibilizados para seleção pela ferramenta. No momento que o engenheiro de processos selecionar o fragmento, este recebe realce azul e está pronto para receber a validação da consistência no momento que o botão "*Next*" for pressionado. A Figura 46 exibe a seleção dos fragmentos aprovados pela validação da completude e melhor priorizados de acordo com o contexto do projeto.

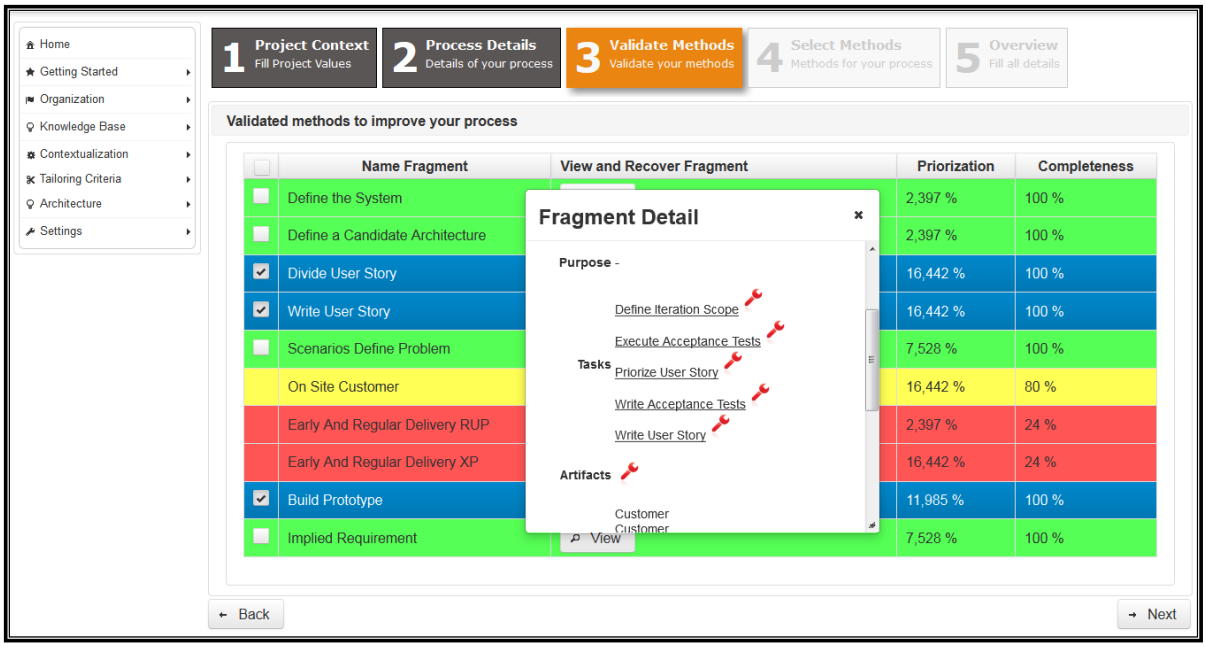



Figura 46 - Seleção dos fragmentos aprovados pela validação da completude.

Fonte: Autor.

A validação da consistência entra em ação ainda entre a etapa 3 e a etapa 4 do *Wizard*. Neste momento, as regras para validação da consistência apresentadas na Seção 5.7 são executadas e utilizam a ontologia descrita no Capítulo 3 deste trabalho para apoiar no processo de reconhecimento da similaridade dos fragmentos. Para todos os fragmentos

similares, um novo fragmento é criado, eliminando assim, duplicidades no processo e nos fragmentos que compõem o mesmo.

Os fragmentos criados pela ferramenta são considerados completos e consistentes uma vez que foram validados pela estratégia QaVaS e por sua vez são disponibilizados na quarta etapa. Pode-se visualizar na Figura 47, a tela da quarta etapa, na qual o engenheiro de processos pode fazer uso da sua experiência e escolher os fragmentos de métodos considerados completos e consistentes pela estratégia QaVaS.

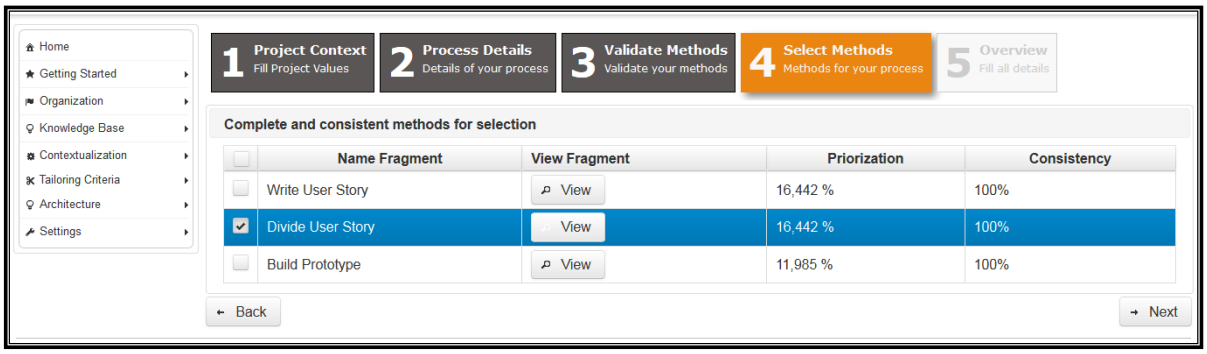



Figura 47 - Tela para seleção dos fragmentos completos e consistentes.

Fonte: Autor.

A tela para seleção de fragmentos completos e consistentes traz uma tabela dividida em quatro colunas. A primeira coluna permite que o engenheiro de processos selecione o fragmento candidato melhor priorizado e assim, inclua este fragmento no processo específico. A segunda coluna, denominada "*Name fragment*", traz o nome do fragmento. Na coluna denominada "*View fragment*" o engenheiro de processos pode visualizar o fragmento antes e depois de selecionar o mesmo. Na coluna "*Priorization*", o engenheiro de processos pode analisar o grau de priorização dos fragmentos em relação ao contexto do projeto. Por fim, a coluna "*Consistency*" traz a informação que o fragmento que ali está é 100% consistente.

O nome dos fragmentos construídos a partir da estratégia QaVaS recebem como sufixo o nome dos fragmentos que o originaram acrescido do texto "- QaVaS". Optou-se por esta identificação para visualizar o real impacto que a estratégia da consistência tem nas partes que podem compor o processo específico.

Cabe ainda relatar que para os novos fragmentos construídos, a ferramenta gera uma nova versão (propriedade *version* da tabela *activity* pertencente ao metamodelo MfTP) e mapeia os fragmentos que originaram os fragmentos criados por meio da estratégia QaVaS

(tabela *activity_parent* pertencente ao metamodelo MfTP). No entanto, estas informações foram omitidas da visualização do fragmento para serem posteriormente utilizadas na evolução da estratégia e do sistema.

A quinta e última etapa, consiste em criar o processo adaptado por meio da seleção dos fragmentos completos e consistentes validados mediante a aplicação da estratégia QaVaS. O processo criado recebe: i) um nome de forma automática, caso o engenheiro de processos não tenha informado na segunda etapa do *Wizard*; ii) uma descrição automática contendo o nome do projeto ao qual o processo se refere; e iii) uma versão para garantia da consistência dos fragmentos e do processo.

A Figura 48 mostra a tela que exibe o processo específico criado ou adaptado.

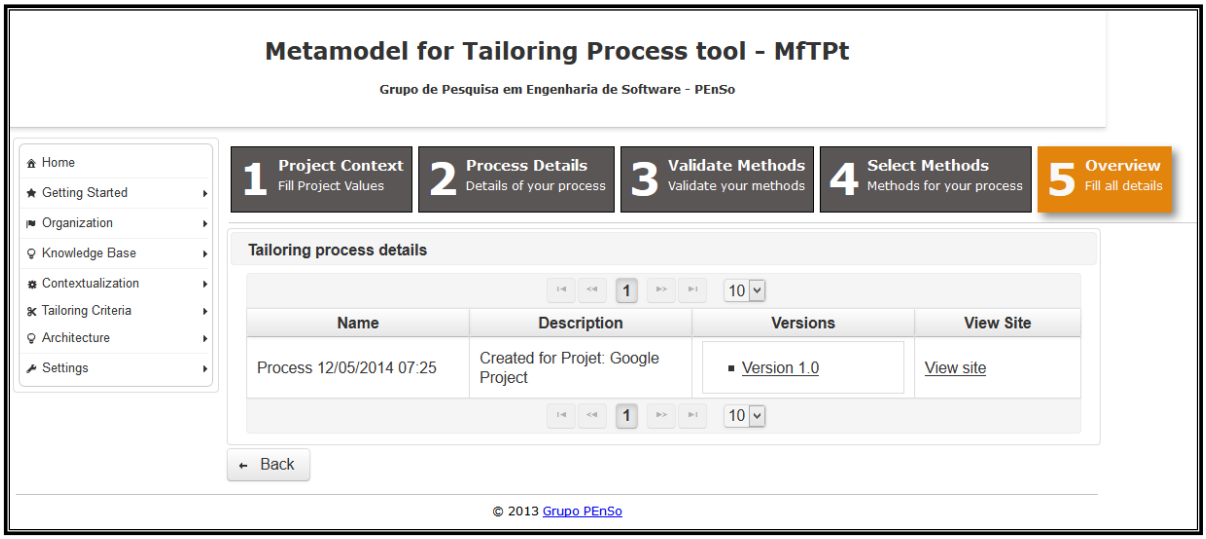



Figura 48 - Tela que exibe o processo específico criado ou adaptado.

Fonte: Autor.

Ressalta-se que embora a adaptação ou criação dos processos fique a cargo do nível de experiência do engenheiro de processos, a estratégia QaVaS inclusa no módulo para construção de processos completos e consistentes, tem a função de fornecer ao engenheiro de processos apoio para que este somente selecione fragmentos de processo completos e consistentes, gerando processos de qualidade. Esta ajuda proposta pela estratégia QaVaS contribui tanto para a melhoria do processo, quanto para diminuir a complexidade existente na tarefa realizada por este profissional, evitando que o mesmo selecione elementos de processo incompletos ou inconsistentes.

# 7 VALIDAÇÃO DA ESTRATÉGIA

Neste capítulo é apresentada a validação da estratégia proposta. A estratégia QaVaS foi validada por meio de um experimento realizado no Setor de Tecnologia da Informação e Comunicação (STIC), do Instituto Federal Farroupilha, no Campus São Vicente do Sul.

O experimento realizado foi do tipo controlado e levou em consideração a experiência das pessoas envolvidas no teste, as tarefas envolvidas no processo padrão da organização, o contexto real dos projetos e os riscos enfrentados pela equipe, conforme especifica Münch et al. (2012).

## 7.1 *Descrição do contexto*

O grupo de colaboradores do Setor de Tecnologia da Informação e Comunicação (STIC), do Instituto Federal Farroupilha, Campus São Vicente do Sul, foi criado com o propósito de desenvolver e gerir a área de informação e comunicação no Campus de São Vicente do Sul. Dividido em várias seções de acordo com sua atribuição, possui a função de prestar apoio técnico, administrativo e operacional, fornecendo suporte de hardware, software e serviços de informação e comunicação aos demais servidores do Campus e à reitoria do Instituto Federal Farroupilha.

A equipe de desenvolvimento de software do STIC que participou da validação da estratégia QaVaS é composta por cinco pessoas, sendo o Gerente de Projetos classificado como muito experiente por possuir mais de cinco anos de experiência. Os demais integrantes da equipe foram classificados como pouco experientes por possuir menos de dois anos de experiência.

O processo de desenvolvimento de software do setor foi definido a quatro anos, de maneira *ad-hoc*, baseado na experiência do Gerente de Projetos e demais integrantes, e é formado por um pequeno conjunto de atividades do modelo RUP. De acordo com o Gerente de Projetos, o modelo RUP foi utilizado como referência pela familiaridade dos integrantes da equipe e pela caraterística de ser planejado, visto que, a equipe envolvida no processo constantemente era redistribuída para outros setores do Instituto.

Pode-se visualizar por meio da Figura 49, o processo padrão de desenvolvimento definido para o STIC. O referido processo foi mapeado por meio de um diagrama BPMN

onde para cada atividade é atribuído o papel do responsável por executá-la. Salienta-se que como a equipe é pequena, para cada integrante pode ser atribuído mais de um papel.

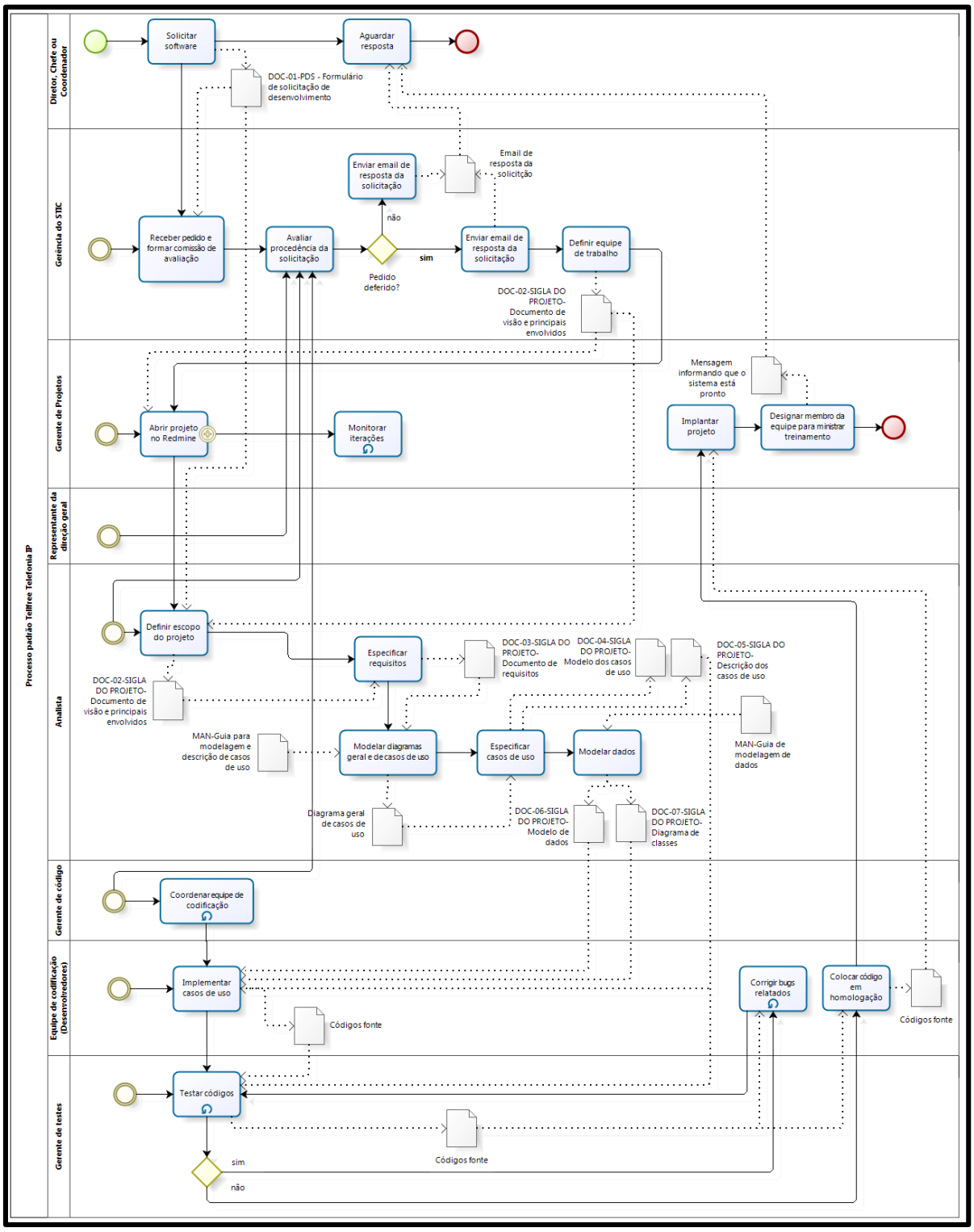



Figura 49 - Processo de Software Padrão do STIC.

Fonte: Adaptado do processo de desenvolvimento de software do STIC.

Por sua vez, foi realizado um mapeamento das atividades de acordo com a documentação que o setor de informação e comunicação utiliza a fim de identificar as disciplinas e as tarefas pertencentes a cada atividade. Na Tabela 8 são detalhadas todas as atividades de acordo com cada disciplina.

Tabela 8 - Atividades do processo de desenvolvimento padrão de acordo com suas disciplinas.

| DISCIPLINA | ATIVIDADES |
|---|---|
| **Abertura de projeto** | Solicitar software<br>Aguardar resposta<br>Receber pedido e formar comissão de avaliação<br>Avaliar procedência da solicitação<br>Enviar e-mail de resposta da solicitação<br>Definir equipe de trabalho<br>Abrir projeto no *Readmine*<br>Monitorar iterações |
| **Requisitos** | Monitorar iterações<br>Especificar requisitos<br>Monitorar iterações |
| **Análise e Projeto** | Modelar diagrama geral e de casos de uso<br>Especificar casos de uso<br>Modelar dados |
| **Implementação** | Monitorar iterações<br>Coordenar equipe de codificação<br>Implementar casos de uso |
| **Testes** | Monitorar iterações<br>Testar códigos<br>Corrigir bugs relatados |
| **Implantação e manutenção** | Monitorar iterações<br>Colocar código em homologação<br>Implantar projeto<br>Designar membro da equipe para ministrar treinamento |

Fonte: STIC.

De posse do processo padrão de desenvolvimento do STIC e dos elementos que o foram mapeados, buscou-se realizar o experimento em quatro etapas. Na primeira etapa foi apresentada a ferramenta MfTP Tool para o gerente de projetos do STIC que cadastrou todos os elementos necessários para o projeto, tais como: atividades, tarefas, papéis, fases, disciplinas e artefatos de entrada e saída de cada tarefa.

A segunda etapa consistiu em identificar um contexto global utilizado para contextualizar os projetos que a equipe desenvolve. Pode-se observar na Tabela 9, o contexto

global definido pela equipe de desenvolvimento do STIC segundo os valores do *Octopus Model*.

Tabela 9 - Contexto global dos projetos desenvolvidos pelo STIC.

| **FATORES** | **ATRIBUTOS** |
|---|---|
| Tamanho | Pequeno |
| Criticidade | Perda de conforto |
| Arquitetura estável | Estável |
| Modelo de Negócios | Sob demanda pra um cliente |
| Distribuição do Time | Local |
| Taxa de mudanças (% no mês) | Entre 10 e 30 |
| Idade do sistema | Novo desenvolvimento |
| Governança | Mecânica/Formal |

Fonte: STIC.

Na terceira etapa, foram identificados os principais critérios de adaptação que podem justificar uma adaptação no processo padrão adotado pelo setor. Foi identificada uma lista de riscos que podem afetar significativamente os projetos, conforme se pode observar na Tabela 10. Por meio destes riscos, a ferramenta MfTP Tool associada à estratégia QaVaS deve recuperar da base de métodos as atividades (fragmentos) necessárias para preveni-los, validar se tais atividades são completas e consistentes e integrá-las ao processo padrão, formando assim, o processo adaptado.

Tabela 10 - Principais riscos identificados nos projetos realizados pelo setor.

| **RISCOS IDENTIFICADOS** |
|---|
| *Misunderstanding the Requirements* |
| *Requirements Instability* |
| *Lack of user involvement* |

Fonte: STIC.

Por fim, na quarta etapa, de posse do contexto global definido, da identificação dos riscos comuns nos projetos do setor e do processo padrão de desenvolvimento devidamente cadastrado na ferramenta, foi apresentado aos membros da equipe, o módulo para definição de processos completos e consistentes. Para execução deste módulo, foi realizada uma explicação detalhada da estratégia QaVaS e selecionado um projeto para aplicação real da estratégia.

## 7.2 *Aplicação real – Projeto "Project IFF"*

Com o objetivo de validar a estratégia QaVaS, foi solicitado ao engenheiro de projetos da equipe do STIC, a criação de um projeto na ferramenta denominado "*Project IFF*" e sua posterior contextualização a fim de aplicar o experimento. Após a criação do projeto, o mesmo foi caracterizado utilizando os seguintes atributos: i) "*Size = Small*"; ii) "*Team Distribution = Collocated*"; iii) "*Criticality = Confort Loss*"; iv) "*Stable Architecture = Stable*"; v) "Rate of Change = *From* 10 *to* 30"; vi) "*Governance = Mechanical/Formal*"; vii) "*Business Model = In House*" e viii) "*Age of System = New development*". Pode-se visualizar por meio da Figura 50 os formulários de criação (*Create a Project*) e contextualização (*Edit Project Context*) do projeto "*Project IFF*".

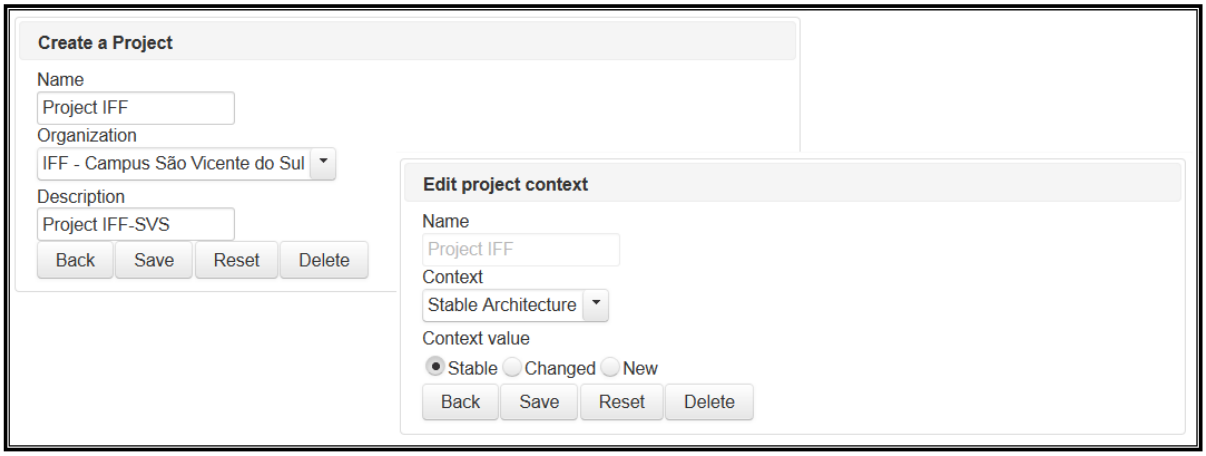



Figura 50 - Formulários de criação e contextualização do Projeto "*Project IFF*".

Fonte: Autor.

Com o projeto criado e contextualizado de acordo com o contexto definido na Tabela 9, foi solicitado ao gerente de projetos que acessasse o módulo para definição de processos completos e consistentes. Conforme descrito no Capítulo 6, o referido módulo é composto por cinco etapas. Na primeira etapa, o projeto "*Project IFF*" precisou ser selecionado para que seu contexto, anteriormente cadastrado, fosse recuperado e assim iniciar a validação da estratégia QaVaS. Pode-se visualizar na Figura 51 a seleção do projeto "*Project IFF*" e seu contexto.

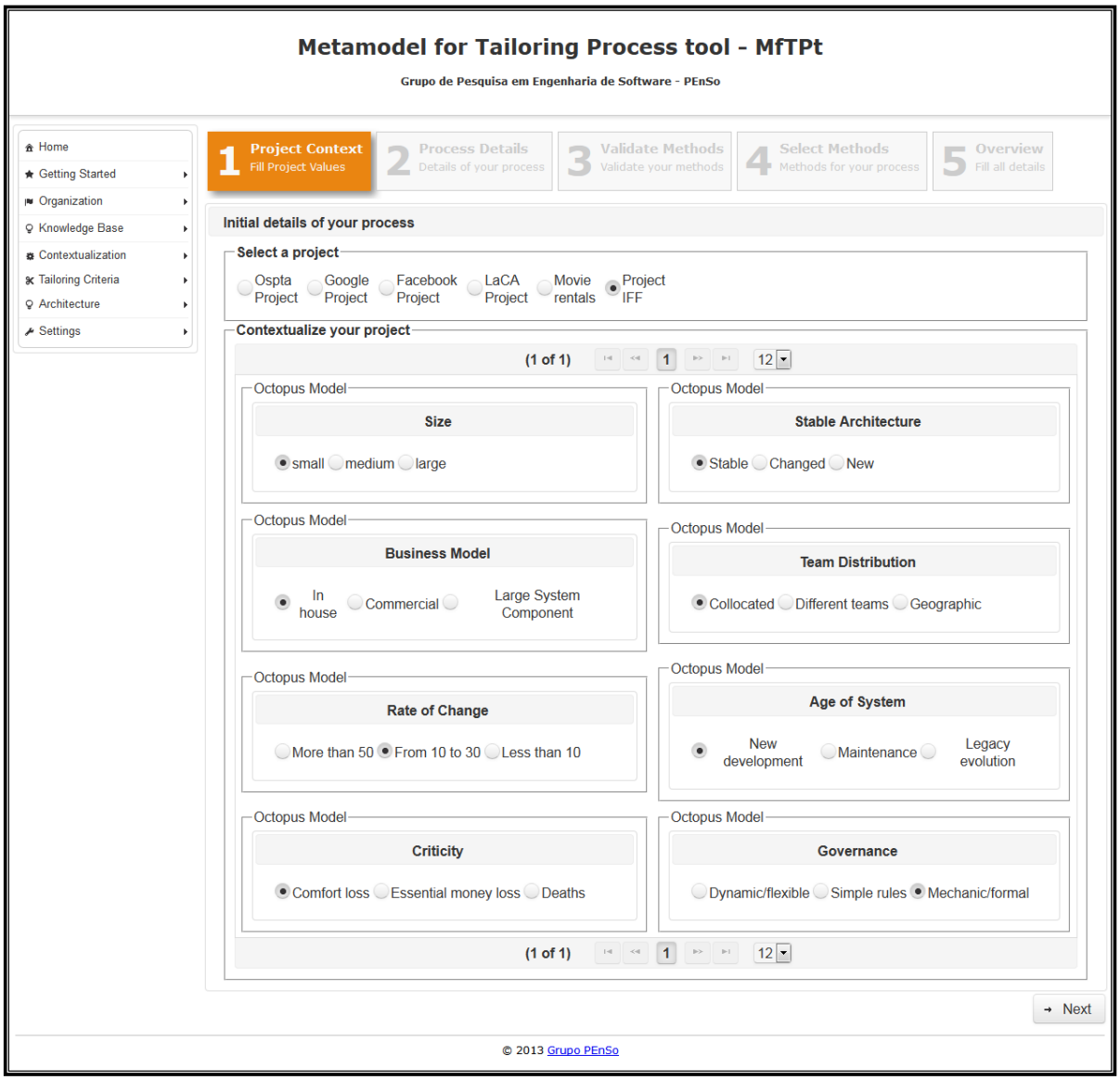



Figura 51 - Seleção do projeto "*Project IFF*" e recuperação do seu contexto.

Fonte: Autor.

A segunda etapa consiste em dar um nome ao processo adaptado, selecionar os critérios de adaptação definidos para o projeto, neste caso os riscos, e selecionar o processo padrão adotado pela equipe para desenvolver seus projetos. Nesta fase, o gerente de projetos selecionou os riscos "*Misunderstanding the Requirements*", "*Requirements Instability*" e "*Lack of user involvement*" e o processo padrão "SPO - STIC (IF São Vicente do Sul)". Pode-se visualizar por meio da Figura 52 a segunda etapa do *Wizard* de validação da estratégia QaVaS para o projeto "*Project IFF*".

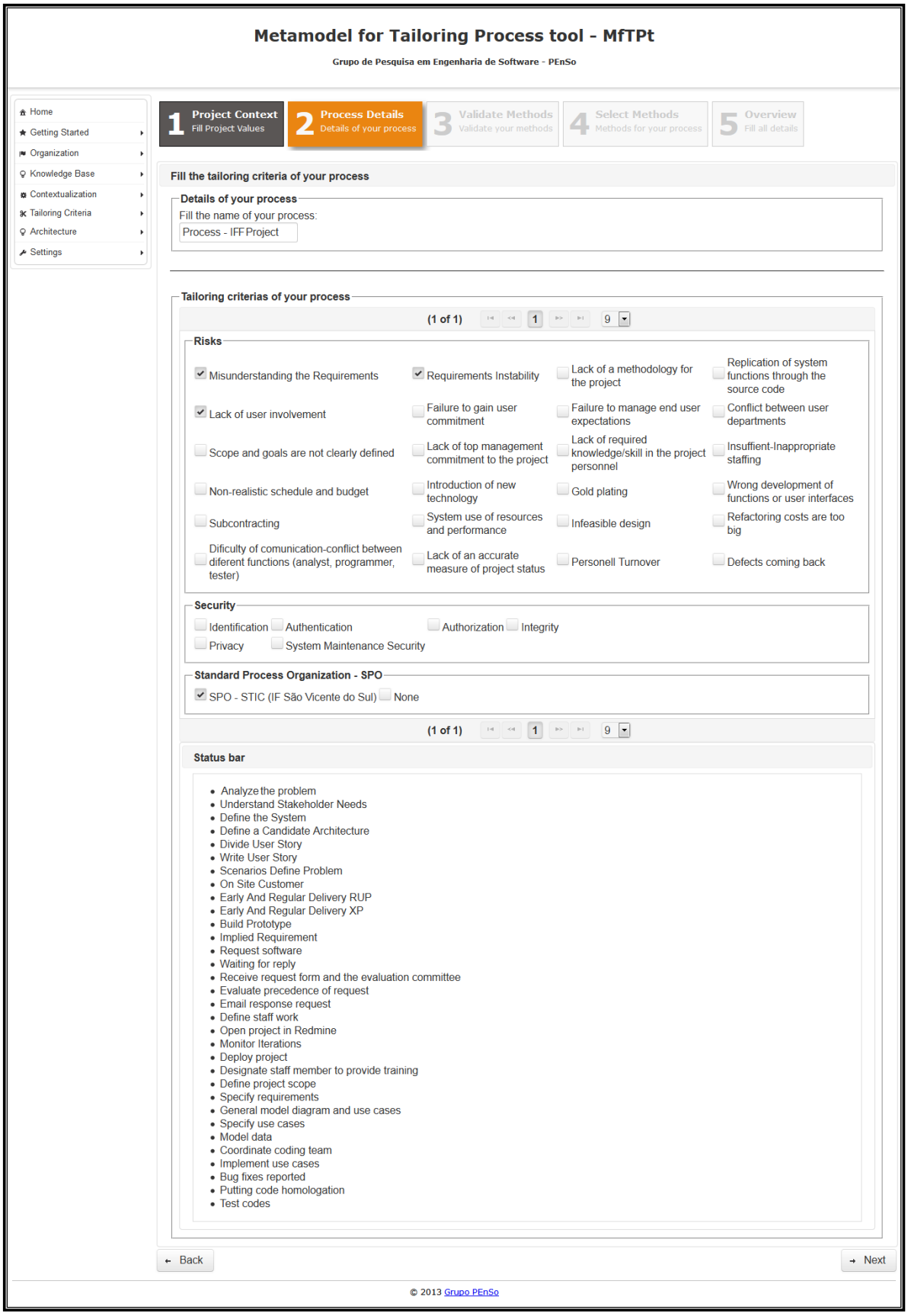



Figura 52 - *Wizard* de validação da estratégia QaVaS para o projeto "*Project IFF*".

Fonte: Autor.

Com a seleção dos riscos "*Misunderstanding the Requirements*", "*Requirements Instability*" e "*Lack of user involvement*", foram recuperados da base de métodos 12 (doze) fragmentos que satisfazem os referidos riscos, são eles: i) "*Analyze the Problem*"; ii) "*Understand Stakeholder Needs*"; iii) "*Define the System*"; iv) "*Divide User Story*"; v) "*Write User Story*"; vi) "*Scenarios Define Problem*"; vii) "*On Site Customer*"; viii) "*Early And Regular Delivery RUP*"; ix) "*Early And Regular Delivery XP*"; x) "*Build Prototype*"; e xi) "*Implied Requirement*". Por sua vez, a seleção do processo padrão "SPO - STIC (IF São Vicente do Sul)" recuperou da base de métodos 20 (vinte) fragmentos que representam as atividades realizadas para desenvolver sistemas no Setor de Informação e Comunicação.

A partir do contexto definido para o projeto e dos 32 (trinta e dois) fragmentos recuperados por meio dos riscos selecionados, na terceira etapa, foi realizada a priorização (coluna *Priorization*) e a avaliação da completude (coluna *Completeness*) dos fragmentos segundo a sistemática descrita nas Seções 5.4 e 5.6 respectivamente. Observa-se por meio da Figura 53 que os fragmentos melhor priorizados são os fragmentos que possuem contexto semelhante ao contexto do projeto. Alguns exemplos deste tipo de fragmento são os fragmentos que representam as atividades "*Define the System*" e "*Write User Story*", o primeiro priorizado em 1,203%, e o segundo em 3,166%, em relação ao contexto definido para o projeto. Ambos os fragmentos foram avaliados em 100% quanto à completude.

Por sua vez, os fragmentos incompletos que permaneceram com grau de completude superior a 60% e que respeitaram os itens eliminatórios definidos pelo processo de validação da completude foram marcados com a cor amarela. Na Figura 53, pode-se visualizar os fragmentos incompletos passíveis de reparação. São eles: i) "*Build Prototype*"; ii) "*Implied Requirement*"; iii) "*Evaluate precedence of request*"; iv) "*Monitor iterations*"; e v) "*Coordinate coding team*". Neste caso, coube ao gerente de projetos do STIC completar os referidos fragmentos durante a adaptação do processo e posteriormente selecioná-los.

Da mesma forma, os fragmentos incompletos "*Early and Regular Delivery RUP*" e "*Early and Regular Delivery XP*" foram destacados em vermelho e eliminados do processo, impedindo que o gerente de projetos do STIC os selecione ou repare. Tais fragmentos, foram avaliados em 24% quanto a sua completude por não possuírem tarefas (item eliminatório), artefatos (classificatório) e papéis associados (classificatório).

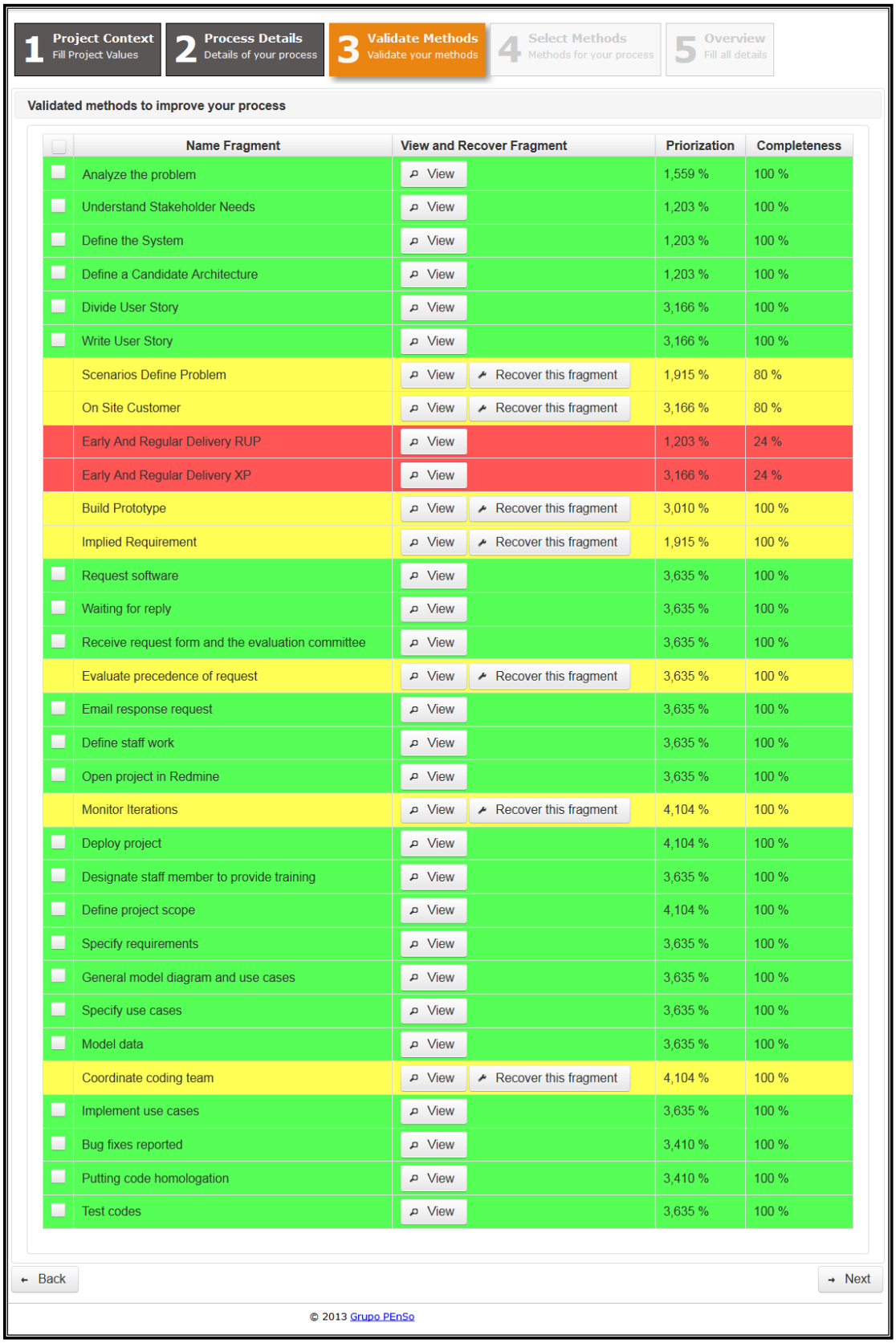



Figura 53 - Priorização dos fragmentos e avaliação da completude para o projeto "*Project IFF*".

Fonte: Autor.

Após a reparação e seleção dos fragmentos completos na terceira etapa, teve início a quarta etapa da estratégia que foi a validação dos fragmentos usando o processo de validação da consistência. Nesta etapa, os conceitos apresentados na Seção 5.7 foram empregados a fim de eliminar possíveis inconsistências nos fragmentos selecionados por meio dos riscos e também nos fragmentos pertencentes ao processo padrão. A Figura 54 mostra os fragmentos completos e consistentes validados pela estratégia QaVaS para compor o processo adaptado para o projeto "*Project IFF*".

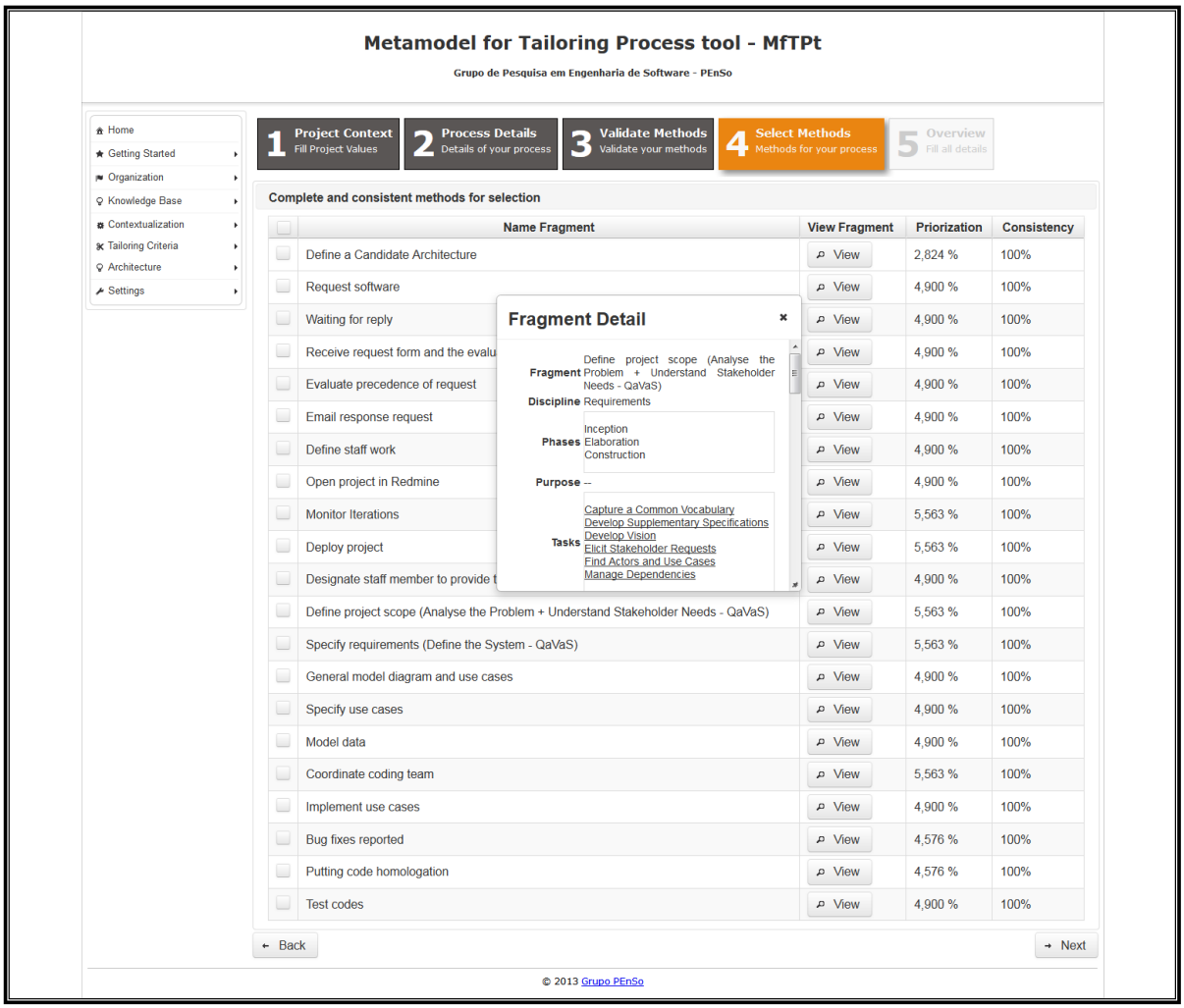



Figura 54 - Fragmentos completos e consistentes validados por meio da estratégia QaVaS.
Fonte: Autor.

Ao analisar a quarta etapa, observou-se que dos 32 (trinta e dois) fragmentos recuperados por meio dos riscos definidos para o projeto ou por meio do processo padrão, 4 (quatro) fragmentos foram excluídos do processo já na terceira etapa por serem incompletos.

Dos 28 (vinte e oito) fragmentos resultantes, ao aplicar o processo de validação da consistência, restaram apenas 21 (vinte e um) fragmentos, sendo os fragmentos "*Define project scope (Analyze the Problem + Understand Stakeholder Needs – QaVaS)*" e "*Specify requirements (Define the System - QaVaS)*" novos fragmentos completos e consistentes criados por meio da estratégia QaVaS.

Por sua vez, após selecionar os fragmentos completos e consistentes e avançar para a quinta etapa do *Wizard*, o gerente de projetos do STIC, concluiu a adaptação do processo de software "*Process – Project IFF*" definido para o projeto "*Project IFF*". Pode-se visualizar por meio da Figura 55 a quinta etapa do *Wizard* para definição de processos completos e consistentes.

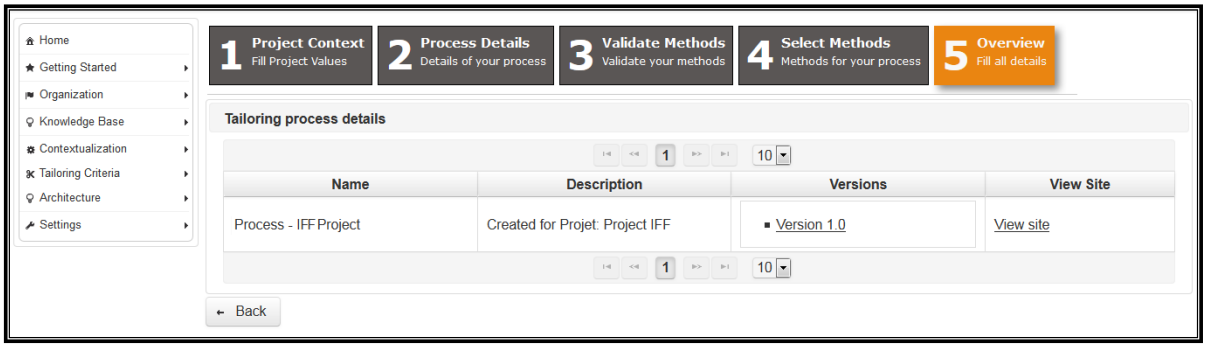



Figura 55 - Quinta etapa do *Wizard* para definição de processos completos e consistentes.
Fonte: Autor.

## 7.3 *Análise da estratégia*

A fim de avaliar o real impacto da estratégia QaVaS, após sua execução na ferramenta MfTP Tool, foi aplicado um questionário (Apêndice B) ao gerente de projetos do STIC mediante o uso da técnica de entrevista. O referido questionário foi criado para este trabalho e por meio de suas perguntas buscou-se: i) conhecer a experiência/conhecimento dos participantes envolvidos no experimento; ii) avaliar a importância dada pelos entrevistados em relação à adaptação de processos de software; iii) avaliar a importância dada pelos entrevistados em relação à garantia da consistência e completude em processos de software; iv) avaliar se a estratégia QaVaS auxilia na consistência, completude e consequentemente na

melhoria da qualidade dos processos de software gerados; e v) avaliar a usabilidade da estratégia e da ferramenta usada para validar a mesma.

Após analisar as respostas do gerente de projetos, destaca-se as seguintes conclusões:

i) A definição de um processo que atenda às necessidades do campus é de fundamental relevância, entretanto, devido às especificidades existentes em cada projeto, este processo precisa ser constantemente adaptado. A adaptação é realizada selecionando elementos de processo obrigatórios e opcionais, gerando assim, uma complexidade em definir que elementos podem ser utilizados para compor o processo.

ii) Durante a atividade de adaptar o processo padrão, o entrevistado entende que este é composto por atividades e por elementos de processo. Quando solicitado para classificar os elementos relevantes e indispensáveis para a definição de uma atividade, obteve-se a lista de classificação que pode ser observada na Tabela 11.

Tabela 11 - Lista de classificação dos elementos relevantes e indispensáveis segundo análise do gerente de projetos do STIC.

| **Avaliação do item** | **Quais são indispensáveis?** | **Grau de relevância** |
|---|---|---|
| Possuir um nome único para que não seja confundida com outra atividade | Indispensável | 3 |
| Possuir uma fonte de origem | Dispensável | 1 |
| Estar associado a uma disciplina da engenharia de software | Dispensável | 2 |
| Possuir fases no ciclo de vida | Dispensável | 2 |
| Possuir tarefas associadas | Indispensável | 3 |
| Possuir ao menos um artefato | Indispensável | 3 |
| Possuir como artefato de entrada artefatos de saída provenientes de outros fragmentos | Indispensável | 3 |
| Possuir papéis associados as tarefas | Indispensável | 3 |
| Possuir uma descrição e um propósito | Dispensável | 1 |
| Possuir critérios de adaptação e um contexto associado | Indispensável | 3 |

Fonte: Gerente de projetos do STIC.

iii) Na adaptação do processo padrão, o risco do responsável por adaptar o processo selecionando elementos incompletos e consistentes é considerado alto pelo entrevistado. Da mesma forma, a existência destes elementos no processo já adaptado pode vir a não só desmotivar a equipe, como atrasar o projeto e aumentar o custo do mesmo.

iv) A adoção de uma estratégia que possibilite avaliar quais elementos são completos e consistentes ajudaria a reduzir a complexidade na adaptação de processos e a evitar os riscos acima expostos. Da mesma forma, a seleção de elementos completos e consistentes voltados a atender às características do projeto e a posterior priorização destes elementos possibilita: a) a melhoria da qualidade do processo adaptado para este fim; e b) avaliar o grau de conhecimento do responsável por adaptar o processo.

v) Por sua vez, o entrevistado julga que a sistemática proposta pela estratégia QaVaS é de grande importância para adaptação dos processos, uma vez que por meio da validação da consistência e completude elimina duplicidades, evitando que sejam realizadas atividades incompletas e diminui a complexidade na adaptação dos processos de uma organização fornecendo assim, segurança para as pessoas executoras do processo.

vi) Outro fator importante destacado foi a possibilidade de reparar os elementos de processo considerados incompletos e persisti-los após a reparação. Segundo o entrevistado, o momento escolhido para esta reparação é o ideal, uma vez que a necessidade de reparar um elemento incompleto é na fase que os elementos de processos estão sendo escolhidos para compor o processo adaptado. Em relação aos elementos incompletos que devem ser descartados a avaliação foi satisfatória, pois é de entendimento que elementos sem os itens considerados indispensáveis não podem ser utilizados para compor um processo.

vii) A utilização da ferramenta MfTP Tool e seu módulo para validação da definição de processos completos e consistentes foram considerados essenciais para o entendimento do trabalho e para validação da estratégia. A usabilidade da ferramenta foi elogiada e o *Wizard* para definição de processos completos e consistentes destacou-se por:

a) Guiar o gerente de projetos na tarefa de adaptar o processo padrão do STIC para um projeto específico;

b) Informar o gerente de projetos quais atividades (fragmentos) são completas e incompletas, possibilitando, mediante avaliação, recuperar as incompletas em tempo de execução.

c) Recuperar da base de dados as atividades já cadastradas que satisfazem requisitos específicos do projeto ou que integram o processo padrão da organização;

d) Priorizar as referidas atividades de acordo com o contexto definido para o projeto;

e) Possibilitar eliminação de inconsistência (duplicidades) e diminuir a complexidade na seleção dos elementos de processo;

viii) Entre as dificuldades relatadas destaca-se a dificuldade em cadastrar e definir os elementos da ferramenta em um primeiro momento, até se obter um repositório satisfatório para a reutilização destes elementos. Segundo o gerente de projetos, à medida que o repositório estiver com um volume adequado de informação, este aspecto negativo será eliminado.

ix) O entrevistado sugeriu para ser adicionado na ferramenta um módulo para gestão de processos. O referido módulo além de armazenar os artefatos produzidos possibilitaria avaliar o processo definido para o projeto.

Neste capítulo foram apresentados os resultados do experimento realizado no Setor de Informação e Comunicação do Instituto Federal Farroupilha, Campus São Vicente do Sul, no qual foi validado a estratégia proposta por meio do módulo para definição de processos completos e consistentes, existente na ferramenta MfTP Tool. No Capítulo 8, são apresentados os trabalhos relacionados que motivaram a proposta apresentada neste trabalho.

# 8 TRABALHOS RELACIONADOS

Independente do processo de software escolhido pela organização, este processo não pode ser utilizado sem antes sofrer alguma customização. Inúmeras abordagens visam adaptar o processo com base nas características do projeto ou organização, entretanto, poucas são as pesquisas que buscam validar critérios de qualidade para os componentes formadores deste processo adaptado.

Azzouz; Kraiem e Ghezala (2012) apresentam em sua proposta um conjunto de critérios que permitem avaliar a construção de um método ou processo em termos de qualidade dentro da abordagem SME. A completude, a consistência e outros critérios são explorados em duas etapas. A primeira etapa é realizada no momento da caracterização do processo, na qual requisitos para caracterizar a qualidade em processos são utilizados. Por sua vez, a segunda etapa, é realizada na construção do método e consiste em avaliar e validar as decisões tomadas na etapa inicial.

Os autores salientam a falta de obras cobrindo o assunto de critérios de qualidade na construção de processos de software, embora na área de engenharia de software o conceito de qualidade seja amplamente discutido. Como contribuição é apresentado um metamodelo para vincular o fragmento de processo a critérios de validação da qualidade na etapa de construção de métodos por meio da abordagem SME.

Este trabalho apresenta um conceito mais amplo comparado ao trabalho de Azzouz; Kraiem e Ghezala (2012), uma vez que apresenta um metamodelo desenvolvido para: i) suportar abordagens multicritérios (critérios de qualidade, riscos, segurança, entre outros); ii) suportar abordagens de adaptação e criação de processos como SME ou linhas de processo de software; e iii) possibilitar o versionamento dos fragmentos envolvidos no processo de construção de um novo fragmento de acordo com os critérios envolvidos. Além disso, a proposta deste trabalho satisfaz duas das validações contempladas pelos autores que é a validação da consistência e da completude, o que possibilita a construção de métodos completos e mais aderentes ao produto a ser desenvolvido.

Barreto; Murta e Rocha (2011) apresentam uma abordagem para definição de processos com foco na reutilização. Por meio de conceitos como componentes de processo, arquiteturas linhas de processo e características para descrever os requisitos do processo, os autores buscam garantir a consistência externa dos elementos que compõem uma linha de processos de software, garantindo que o que é requerido por uma parte do processo seja

produzido por outra. Contudo, a referida abordagem não valida a completude interna dos elementos selecionados para compor o processo, nem a consistência semântica destes elementos, permitindo que o processo formado por meio desta abordagem possua ambiguidades ao longo de sua execução.

Kornyshova (2013) apresenta a abordagem denominada MADISE. Por meio da integração de conceitos de SME e linhas de processo de software para engenharia os autores buscam eliminar inconsistências e construir a linha de processos de software para determinado projeto usando para isso famílias de métodos. Uma família de métodos é um conjunto de componentes de método, extraídos de processos ou melhores práticas, que possuem o mesmo propósito ou propósitos similares agrupados em famílias.

A principal vantagem em utilizar uma família de métodos é que a mesma permite reduzir a complexidade na gestão da grande variedade de métodos facilitando a reutilização dos componentes de métodos, eliminando inconsistências e propiciando a adaptação (KORNYSHOVA, 2013). Todavia, a abordagem MADISE não verifica o grau de completude dos elementos formadores das famílias de método. Isso possibilita que o engenheiro de métodos selecione para a linha de processo, famílias de métodos incompletos e por sua vez afete o processo, impactando diretamente na qualidade do mesmo.

A Tabela 12 apresenta possíveis aplicações da QaVaS em cada uma das abordagens.

Tabela 12 - Possíveis aplicações da estratégia QaVaS nas abordagens estudadas.

(continua)

| **Abordagem** | **Aplicação da estratégia QaVaS** |
|---|---|
| Barreto et al. (2011) | Sugere-se a aplicação da estratégia QaVaS para validar a completude e consistência dos componentes de processo que formam a linha de processos impedindo que o engenheiro de processos: i) selecione para compor a linha de processos componentes incompletos e com relacionamento faltando; ii) selecione componentes com tarefas similares gerando ambiguidades e por sua vez insatisfação para quem for usar a linha; iii) altere um componente de processos que já foi instanciado em uma ou mais linhas de processo e assim deixe tais linhas incoerentes em relação ao objetivo para o qual foram criadas. |
| Azzouz et al. (2012) | Sugere-se que a estratégia QaVaS adapte-se a todos os critérios de qualidade para construção de processos propostos por Azzouz et al. (2013). Tal evolução propiciará com que a estratégia QaVaS valide uma gama maior de critérios e contribua assim para o aumento da qualidade dos elementos formadores do processo em que ela for aplicada. |

(conclusão)

| | |
|---|---|
| Kornyshova (2012) | Sugere-se que a estratégia QaVaS seja utilizada na etapa de construção das famílias de métodos contribuindo para que somente componentes de métodos completos façam parte das famílias de métodos. A estratégia QaVaS permitirá também uma efetiva recuperação dos componentes incompletos e eliminará inconsistências internas acabando com possíveis redundâncias que podem ocorrer nas famílias de métodos. |

# 9 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

A criação e adaptação de processos vêm sendo fonte inesgotável de pesquisas na área de engenharia de software. No entanto, ao analisar os trabalhos relacionados apresentados no Capítulo 8 desta dissertação, pode-se verificar que a validação da completude e consistência dos elementos que compõem este processo é ainda tema pouco discutido e explorado.

Por meio de uma revisão da literatura foi possível observar que tanto a criação ou adaptação de processos quanto a seleção dos elementos que compõem estes processos, embora complexa, é ainda executada de maneira *ad-hoc* deixando a responsabilidade unicamente para o engenheiro de processos. Isto possibilita que por erro humano, o processo gerado possa vir a ter elementos incompletos e/ou inconsistentes impactando diretamente na qualidade do produto construído e no tempo gasto para execução do processo, desperdiçando assim, tempo e dinheiro.

Este trabalho propõe uma estratégia para validação da consistência e completude dos fragmentos que possam vir a fazer parte dos processos construídos ou adaptados. Por meio da estratégia QaVaS, buscou-se melhorar a qualidade do processo adaptado utilizando a seleção de elementos de processo completos e consistentes.

A estratégia proposta utiliza uma ontologia, denominada OntoSME, que foi modelada e implementada neste trabalho. Esta ontologia se mostrou eficaz para o reconhecimento da similaridade semântica e estrutural dos fragmentos de processo, e propiciou representar o conhecimento inicial para o domínio de processos de software adaptados a partir da abordagem SME. Entretanto, verificou-se que o sucesso para o reconhecimento da similaridade estrutural e semântica depende diretamente do conhecimento armazenado na ontologia. Neste caso, torna-se importante em trabalhos futuros expandir este conhecimento.

A modelagem de um metamodelo, denominado MfTP, colaborou para oferecer um alto grau de formalismo para a estratégia QaVaS. Este formalismo, por sua vez, proporcionou o reuso de fragmentos completos e consistentes e contribuiu para se obter um melhor suporte para a garantia da consistência, completude e consequentemente um aumento significativo da qualidade nos fragmentos de processo criados por meio da estratégia.

Por meio das métricas atribuídas para avaliar o grau de completude dos fragmentos foi possível avaliar quais fragmentos são completos, quais podem ser completados e quais devem ser eliminados do processo adaptado. A seleção dos fragmentos completos que satisfazem

características específicas do projeto, como riscos, por exemplo, e a priorização destes fragmentos contribuiu para diminuir a complexidade na definição do processo, impedindo assim, que o engenheiro de processos selecionasse para compor o processo fragmentos incompletos. Do mesmo modo, a criação de regras de qualidade para criação de novos fragmentos associada ao processo de validação da consistência e a priorização dos fragmentos candidatos completos possibilitou a eliminação das duplicidades nos fragmentos, criando assim, novos fragmentos completos e consistentes.

Por sua vez, conclui-se que a partir do experimento realizado por meio da ferramenta MfTP, que a estratégia QaVaS aliada a metodologias de adaptação de processos pode contribuir para a melhoria da qualidade em processos de software definidos para projetos específicos da organização informando ao engenheiro de processos quais fragmentos são adequados (completos e consistentes), e possibilitando a eliminação de inconsistências internas nestes elementos.

Para o meio científico, a estratégia QaVaS contribuiu para a área de engenharia de métodos, considerada essencial para evolução da área de engenharia de software, considerando que o resultado é uma nova proposta para integração, validação e melhoria contínua dos processos de software, buscando facilitar a definição de processos e diminuir o custo e o esforço associado a esta atividade.

Para a indústria, a estratégia contribuiu com a elaboração de uma ferramenta que propicia: i) auxiliar na construção ou adaptação do processo com base nas necessidades da organização ou projeto, elaborando um processo consistente, completo e mais aderente ao produto a ser desenvolvido; e ii) auxiliar médias e pequenas empresas a definir seu processo de desenvolvimento de forma interativa e de fácil compreensão.

Trabalhos futuros incluem: i) melhorar o mecanismo de consultas usado na identificação da similaridade, uma vez que a criação de tarefas em formato de “texto não estruturado” pode acarretar que tarefas similares passem desapercebidas no processo de validação da consistência; ii) o desenvolvimento de um módulo na ferramenta MfTP Tool para inclusão de novos conhecimentos na ontologia OntoSME; e iii) evolução da estratégia QaVaS para ser aderente a outros critérios que caracterizam qualidade em processos de software, como por exemplo, a coesão e a dependência. Sendo a primeira definida pela capacidade que um fragmento tem de ser coerente em relação ao propósito para o qual foi criado, e a segunda, definida pela relação que um fragmento tem com os outros para existir.

# REFERÊNCIAS BIBLIOGRÁFICAS


AHARONI, A.; REINHARTZ-BERGER, I. **Representation of Method Fragments: A Comparative Study**. Situational Method Engineering: Fundamentals and Experiences. Springer US. p. 130-145. Anais 2007.

ARMBRUST, O. et al. **Scoping software process lines**. Software Process: Improvement and Practice, v. 14, n. 3, p. 181-197, 2009.

ATKINSON, C.; GUTHEIL, M.; KIKO, K. **On the relationship of ontologies and models.** Proceedings of the 2nd Workshop on MetaModelling and Ontologies WoMM06 LNI P96 Gesellschaft fur Informatik Bonn (2006). **Anais** 2006. Disponível em: <http://subs.emis.de/LNI/Proceedings/Proceedings96/GI-Proceedings-96-3.pdf>.

AZZOUZ, S.; KRAIEM, N.; GHEZALA, H. BEN. **Defining the foundations of situational method quality**. International Journal of Computers & Technology, 2012.

BAJEC, M.; VAVPOTIČ, D.; KRISPER, M. **Practice-driven approach for creating project-specific software development methods**. Information and Software Technology, v. 49, n. 4, p. 345–365, abr. 2007.

BARRETO, A.; MURTA, L.; ROCHA, A. R. **Software Process Definition: a Reuse-based Approach**. Journal Of Universal Computer Science, v. 17, n. 13, p. 1765–1799, 2011.

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

BRASIL, M. A. B.; FONTOURA, L. M.; ALVARO, L. **Uma Proposta para Melhoria da Qualidade de Processos de Software com base em MPS-BR**. SBQS 2013 - XII Simpósio Brasileiro de Qualidade de Software, p. 15, 2013.

BRASIL, M. A. B.; PEREIRA, G. V.; FONTOURA, L. M. **Software process tailoring using Situational Method Engineering based on criteria of quality improvement**. 2012, XXVIII Conferencia Latinoamericana En Informatica (CLEI), p. 1–8, out. 2012.

BRICKLEY, D.; GUHA, R. V. **RDF Vocabulary Description Language 1.0: RDF Schema**. 2004.

BRINKKEMPER, S.; SAEKI, M.; HARMSEN, F. **Assembly techniques for method engineering**. Information Systems Engineering, p. 381–400, 1998.

BUCHER, T. et al. **Situational Method Engineering - On the Differentiation of "Context" and "Project Type"**. Situational Method Engineering: Fundamentals and Experiences: Proceedings of the IFIP WG 8.1 Working Conference. Anais 2007.

CASTANO, S.; ANTONELLIS, V. de. **A constructive approach to reuse of conceptual components**. Proceedings Advances in Software Reuse, 1993.

CONSORTIUM, W. W. W. OWL 2 Web Ontology Language Document Overview. **OWL 2 Web Ontology Language**, p. 1–7, 2012.

COPLIEN; O. J.; ALEXANDER, A. **Word On**. Software patterns. 1996.

DEVEDZIĆ, V. **Understanding ontological engineering**. Communications of the ACM, 2002.

FETTKE, P. **Business Process Modeling Notation, version 2.0**. WIRTSCHAFTSINFORMATIK, v. 50, n. 6, p. 504–507, 7 dez. 2008.

FONTOURA, L. **PRiMA: Project Risk Management Approach**. Universidade Federal do Rio Grande do Sul (UFRG), 2006.

FONTOURA, L.; PRICE, R. **Systematic Approach to Risk Management in Software Projects through Process Tailoring**. SEKE, 2008.

FUGGETTA, **A. Software Process: A Roadmap**. International Conference on Software Engineering, 2000.

GHOLAMI, M. F.; SHARIFI, M.; JAMSHIDI, P. **Enhancing the OPEN Process Framework with service-oriented method fragments**. Software and Systems Modeling, v. 13, p. 361–390, 2014.

GINSBERG, M.; QUINN, L. **Process Tailoring and the the Software Capability Maturity Model**. 1995.

GRUBER, T. R. **Toward principles for the design of ontologies used for knowledge sharing?** International Journal of Human-Computer Studies, v. 43, p. 907–928, 1995.
GRUNINGER, M. **Designing and Evaluating Generic Ontologies**. 2007.

GUPTA, D.; DWIVEDI, R. **A Step Towards Method Configuration from Situational Method Engineering**. v. 2, n. 1, p. 51–59, 2012.

HARMSEN, F.; BRINKKEMPER, S.; OEI, H. **Situational Method Engineering for Information System Project Approaches**. A.A. Verrijn Stuart and T.W. Olle (Eds.), Methods and Associated Tools for the Information Systems Life Cycle. Proceedingsof the IFIP WG 8.1 Working Conference, Maastricht, Netherlands, September 1994,IFIP Transactions A-55, North-Holland, ISBN 0-444-82074-4, pp. 169-194, 1994.

HAYES, P. RDF Semantics. **W3C Recommendation**, v. 10, p. 1–45, 2004.

HENDERSON-SELLERS, B. **Bridging metamodels and ontologies in software engineering**. Journal of Systems and Software, v. 84, n. 2, p. 301–313, 2011.

HENDERSON-SELLERS, B.; GONZALEZ-PEREZ, C.; RALYTE, J. **Comparison of Method Chunks and Method Fragments for Situational Method Engineering**. 19th Australian Conference on Software Engineering (ASWEC 2008), 2008.

HENDERSON-SELLERS, B.; RALYTÉ, J. **Situational Method Engineering: State-of-the-Art Review**. Journal of Universal Computer Science, v. 16, n. 3, p. 424–478, 2010.

HORRIDGE, M. **A Practical Guide To Building OWL Ontologies Using Protégé 4 and CO-ODE Tools**. Edition 1.2. The University Of Manchester, v. 1.3, p. 0–107, 2011.

HORROCKS, I. et al. **SWRL: A Semantic Web Rule Language Combining OWL and Rule.** MLW3C Member submission 21, 2004.

IBM. **Rational Unified Process (software), version 7.0**. IBM Rational, 2006.

ISO/IEC 15504. Iso/iec tr 15504-6. **Office**, v. 1998, 1998.

ISO/IEC 24744. **ISO/IEC Guide 24744:2007, Software Engineering - Metamodel for Development Methodologies**. International Organization for Standardization, 2007.

ISO/IEC/IEEE 29148. Systems and software engineering - Life cycle processes - Requirements engineering. **ISO/IEC/IEEE 29148:2011(E)**, p. 1–94, 2011.

JAUFMAN, O.; MÜNCH, J. **Acquisition of a Project-Specific Process** (F. Bomarius, S. Komi-Sirviö, Eds.). 6th International Conference on Product Focused Software Development and Process Improvement PROFES 05. 2005. Disponível em: <http://www.springerlink.com/content/vjby7fl2phjj1xjj/>.

JENA. **Jena Semantic Web Framework**. IEEE Internet Computing, p. 55-59, 2007. Disponível em: <http://jena.sourceforge.net/>.

JENERS, S.; LICHTER, H.; PYATKOVA, E. **Automated Comparison of Process Improvement Reference Models Based on Similarity Metrics**. 2012 19th Asia-Pacific Software Engineering Conference, p. 743–748, Dez. 2012.

KALUS, G.; KUHRMANN, M. **Criteria for Software Process Tailoring A Systematic Review**. Proceedings of the 2013 International Conference on Software and System Process - ICSSP 2013, p. 171, 2013.

KORNYSHOVA, E.; Ralyte, J. & Deneckere, R. (2013). **Constructing Method Families Based on the Variability Analysis**. Proceedings of the RCIS 2013 Conference Forum. 2013.

KORNYSHOVA, E.; DENECKÈRE, R. **Decision-making method family MADISE:** Validation within the requirements engineering domain. Research Challenges in Information Science (RCIS), 2012 Sixth International Conference on (pp. 1-10). IEEE. 2012.

KORNYSHOVA, E.; DENECKÈRE, R.; ROLLAND, C. **Method families concept:** Application to decision-making methods. In Enterprise, Business-Process and Information Systems Modeling (pp. 413-427). Springer Berlin Heidelberg. 2011.

KRUCHTEN, P. **Contextualizing agile software development**. EuroSPI Conference, p. 1–12, 2010.

MAGDALENO, A. M. **An Optimization-based Approach to Software Development Process Tailoring**. 2nd International Symposium on Search Based Software Engineering, p. 40–43, set. 2010.

MARTINEZ-RUIZ, T. et al. **Applying AOSE Concepts to Model Crosscutting Variability in Variant-Rich Processes**. 37th EUROMICRO Conference on Software Engineering and Advanced Applications, p. 334–338, Ago. 2011.

MCGUINNESS, D. L.; VAN HARMELEN, F. **OWL Web Ontology Language Overview**. W3C recommendation, v. 10, p. 1–22, 2004.

MÜNCH, J.; ARMBRUST, O.; KOWALCZYK, M.; SOTO, M. **Software Process Definition and Management**. Springer, 2012.

NOY, N.; MCGUINNESS, D. **Ontology development 101: A guide to creating your first ontology**. p. 1–25, 2001.

O'CONNOR, M.; DAS, A. **SQWRL: A Query Language for OWL**. OWLED, v. 23, p. 3–10, 2009.

PAULK, Mark C. et al. **Capability maturity model, version 1.1**. Software, IEEE, v. 10, n. 4, p. 18-27, 1993.

PEREIRA, G. V. **Abordagem multicritérios para adaptação de processos de software baseada em situational method engineering**. 2012.

PEREIRA, G.; FONTOURA, L. **Defining Agile and Planned Method Fragments for Situational Method Engineering**. Simpósio Brasileiro de Sistemas de Informação, 2012.

RALYTÉ, J.; ROLLAND, C. **An assembly process model for method engineering**. Internation Conference on Advanced Information Systens Engineering, Switzerland, p. 24, 2001.

ROMBACH, D. **Integrated software process and product lines**. Unifying the Software Process Spectrum, n. May, 2006.

SEI. **CMMI for Development, Version 1.3**. Carnegie Mellon University. Disponível em: <http://repository.cmu.edu/sei/287/>, 2010.

SOFTEX. **MPS.BR – Melhoria de Processo do Software Brasileiro. Guia geral**. SEI. **Software & Systems Process Engineering Meta-Model Specification Verson 2.0**. Continuous Representation, Pittsburgh: Software Engineering Institute, Carnegie Mellon University, 2012.

SOMMERVILLE, I. **Software Engineering - 9ª Edition**. Harlow, England: Addison-Wesley, 2010.

TSUMAKI, T.; TAMAI, T. **Framework for matching requirements elicitation techniques to project characteristics**. Software Process Improvement and Practice, v. 11, p. 505–519, 2006.

VAN DER AALST, W. M. P. **Business Process Management:** A Comprehensive Survey. ISRN Software Engineering, p. 1–37, 2013.

VARGAS, R. V. **Utilizando a programação multicritério (Analytic Hierarchy Process-AHP) para selecionar e priorizar projetos na gestão de portfólio**. In: PMI Global Congress. p. 1–22. 2010.

VIDAL, L. A.; MARLE, F.; BOCQUET, J. C. **Using a Delphi process and the Analytic Hierarchy Process (AHP) to evaluate the complexity of projects**. Expert Systems with Applications, v. 38, p. 5388–5405, 2011.

WANGENHEIM, A. V.; WANGENHEIM, C. G. **Raciocínio Baseado Em Casos**. 1. ed. Barueri. 2003.

XU, P. Knowledge Support in Software Process Tailoring. **Proceedings** of the 38th Annual Hawaii International Conference on System Sciences. Anais IEEE, 2005.

XU, P.; RAMESH, B. **Using process tailoring to manage software development challenges**. IT Professional, v. 10, n. 4, p. 39–45, jul. 2008.

YAGUCHI, Y. et al. **Word Space:** A New Approach to Describe Word Meanings. The Sixth IEEE International Conference on Computer and Information Technology (CIT'06), 2006.

YOON, I. C.; MIN, S. Y. & BAE, D. H. (2001, December). **Tailoring and verifying software process**. Software Engineering Conference, 2001. APSEC 2001. Eighth Asia-Pacific (pp. 202-209). IEEE. 2001.

ZOWGHI, D.; FIRESMITH, D.; HENDERSON-SELLERS, B. Using the OPEN process framework to produce a situation-specific requirements engineering method. **Proceedings** of SREP, p. 1–18, 2005.

# APÊNDICE A – PROPRIEDADE DE DADOS E DE OBJETOS DA ONTOLOGIA ONTOSME

Tabela 13 - Propriedade de dados e de objetos da ontologia.

(continua)

| Classe | Atributo | Tipo de atributo | Função |
|---|---|---|---|
| *Artifacts* | *name* | Dados (*String*) | Nome do artefato |
| | *is_recomended_to* | Objeto (*Task*) | Para quais atividades o artefato é recomendado |
| | *is_synonym* | Objeto (*Artifacts*) | Exibe os artefatos que são similares ao artefato em questão |
| *Discipline* | *name* | Dados (*String*) | Nome da disciplina |
| | *description* | Dados (*String*) | Descrição da disciplina |
| *Phases* | *name* | Dados (*String*) | Nome da fase |
| | *description* | Dados (*String*) | Descrição da fase |
| *Purpose* | *name* | Dados (*String*) | Propósito para o qual o fragmento pode ser criado. Optou-se por transformar o propósito em classe concreta para trabalhos futuros. |
| *Roles* | *name* | Dados (*String*) | Nome do papel |
| | *description* | Dados (*String*) | Descrição do papel |
| | *has_source* | Objeto (*Source*) | Indica se o papel em questão tem origem a partir de melhores práticas, modelos de processo, padrões de processo ou modelos de referência. |
| *Source* | *name* | Dados (*String*) | Nome da melhor prática, do modelo de processo, do padrão de processo ou do modelo de referência. |
| *Tasks* | *name* | Dados (*String*) | Nome da tarefa |
| | *is_ synonym* | Objeto (*Task*) | Especifica quais tarefas são similares conceitualmente ou semanticamente a tarefa em questão. |
| | *use_verb* | Objeto (*Verb*) | Especifica quais substantivos fazem uso do verbo em questão dentro do conjunto de substantivos e verbos mapeados. |
| | *use_noun* | Objeto (*Noun*) | Especifica quais verbos fazem uso do substantivo em questão dentro do conjunto de substantivos e verbos mapeados. |
| *Tasks* | *has_disciplines* | Objeto (*Discipline*) | Especifica as disciplinas que a tarefa está associada. |

(continuação)

| | | | |
|---|---|---|---|
| *Nouns* | *name* | Dados (*String*) | Especifica o nome do substantivo |
| | *used_in* | Dados (*String*) | Especifica quando o substantivo é usado (atividade/fragmento ou tarefa). |
| | *is_ synonym* | Objeto (*Nouns*) | Especifica os substantivos que são sinônimos ou similares conceitualmente ao substantivo em questão. |
| | *has_source* | Objeto (*Source*) | Indica se o substantivo em questão tem origem a partir de melhores práticas, modelos de processo, padrões de processo ou modelos de referência. |
| | *use_verb* | Objeto (*Verbs*) | Especifica quais verbos fazem uso do substantivo em questão dentro do conjunto de substantivos e verbos mapeados. |
| | *disciplines_in_activities* | Objeto (*Discipline*) | Especifica as disciplinas que o substantivo aparece quando usado dentro das atividades/fragmentos |
| | *disciplines_in_tasks* | Objeto (*Discipline*) | Especifica as disciplinas que o substantivo aparece quando usado dentro das tarefas. |
| *Verbs* | *name* | Dados (*String*) | Nome do verbo. |
| | *used_in* | Dados (*String*) | Especifica quando o verbo é usado (atividade/fragmento ou tarefa). |
| | *is_ synonym* | Objeto (*Verbs*) | Indica os verbos que são sinônimos do verbo em questão. |
| | *has_source* | Objeto (*Source*) | Indica se o verbo em questão tem origem a partir de melhores práticas, modelos de processo, padrões de processo ou modelos de referência. |
| | *nouns_in_activities* | Objeto (*Nouns*) | Especifica quais substantivos fazem uso do verbo em questão dentro do conjunto de substantivos e verbos mapeados quando este verbo for utilizado dentro de uma atividade/fragmento. |

(conclusão)

| | | | |
|---|---|---|---|
| *Verbs* | *nouns_in_tasks* | Objeto (*Nouns*) | Especifica quais substantivos fazem uso do verbo em questão dentro do conjunto de substantivos e verbos mapeados quando este verbo for utilizado dentro de uma tarefa. |
| | *disciplines_in_activities* | Objeto (*Discipline*) | Especifica as disciplinas que o verbo está associado em nível de atividades/fragmentos. |
| | *disciplines_in_tasks* | Objeto (*Discipline*) | Especifica as disciplinas que o verbo está associado em nível de tarefas. |

Fonte: Autor.

# APÊNDICE B – QUESTIONÁRIO REFERENTE A UTILIZAÇÃO DA FERRAMENTA MFTP TOOL PARA VERIFICAÇÃO DA CONSISTÊNCIA E COMPLETUDE DE PROCESSOS DE SOFTWARE UTILIZANDO A ESTRATÉGIA QAVAS

## I. Processos de Software

- **Objetivos:**
  - **Medir a experiência/conhecimento dos participantes envolvidos no experimento;**
  - **Avaliar a importância da adaptação de processos de software no desenvolvimento de projetos.**

1. Qual seu nível de experiência em desenvolvimento de software?

( ) Muito experiente (mais de 5 anos)

( ) Experiente ( entre 2 e 5 anos)

( ) Pouco experiente (menos de 2 anos )

( ) Não tenho experiência

2. Como foi elaborado o processo de software usado atualmente?

3. Em todos os projetos é utilizado o mesmo processo?

   a. Caso a resposta seja **sim**, o processo atende as diferentes características de todos os projetos?

   b. Caso a resposta seja **não**, como é efetuada a adaptação? Esta adaptação é considerada fácil ou complexa?

4. Quais atributos são considerados relevantes para a definição de uma atividade em um processo de software? Classifique-os de 1 a 3 sendo 3 o peso atribuído aos itens de maior relevância e quais são considerados indispensáveis.

( ) Possuir um nome único para que não seja confundida com outra atividade.

( ) Uma propriedade para informar sua origem (Ex.: RUP, XP e Scrum).

( ) Estar associada a uma disciplina da engenharia de software.

( ) Estar associada a fases no ciclo de vida de um processo de software.

( ) Possuir uma ou mais tarefas associadas.

( ) Produzir ao menos um artefato ou produto de trabalho.

( ) Relacionar-se com artefatos de entrada e saída de outras atividades.

( ) Possuir papéis associados a cada tarefa que a compõem.

( ) Possuir uma descrição contendo informações adicionais sobre este atividade.

( ) Possuir um propósito indicando para o que ela foi criada.

( ) Estar associada a informações que descrevem o tipo de projeto.

**II. Estratégia QaVaS:**

- **Objetivos:**
  - **Avaliar a importância da consistência e completude em processos de software;**
  - **Avaliar se a estratégia QaVaS auxilia na consistência e completude dos processos de software gerados.**

1. A utilização da estratégia QaVaS auxilia na definição de um processo de desenvolvimento?

2. Existe o risco do responsável em adaptar o processo selecionar elementos de processo incompletos ou inconsistentes?

a. Caso a resposta seja **sim**:

- A existência de elementos incompletos e/ou inconsistentes pode vir a desmotivar a equipe, atrasar o projeto ou impactar diretamente nos custos do projeto?
- Adaptar um processo com elementos completos e consistentes melhora a qualidade do processo?
- A estratégia QaVaS pode ajudar a diminuir este risco?

3. Ao utilizar a estratégia por meio da estratégia QaVaS:

a. Acredita que a complexidade em adaptar um processo foi reduzida?

b. Acredita que é uma vantagem somente selecionar elementos de processo que sejam completos?

c. Acredita que eliminar elementos inconsistências (elementos duplicados) do processo de software pode ajudar na implementação do processo?

d. Acredita que a seleção de elementos completos e consistentes associado às caraterísticas do projeto e a critérios de adaptação podem melhorar a qualidade do processo adaptado?

4. É importante ter um conjunto de elementos de processo completos e consistentes que possam ser reusados para definir novos elementos ou adaptar processos existentes (importância do repositório)?

5. A atividade de reparar os fragmentos incompletos, proposto pela estratégia QaVaS, é realizada no momento certo?

6. A estratégia QaVaS auxiliou o engenheiro de processos a selecionar as atividades que vão fazer parte do processo adaptado, impedindo que este profissional selecione atividades incompletas ou com duplicidades (inconsistentes)?

7. O processo gerado utilizando a estratégia QaVaS foi satisfatório?

**III. Ferramenta MfTP Tools QaVaS**

- **Objetivos:**
  - **Avaliar a usabilidade da ferramenta;**
  - **Avaliar a usabilidade da estratégia proposta (QaVaS).**

1. A utilização da ferramenta é de fácil entendimento e utilização?

2. O módulo para adaptação de processos e validação da consistência e completude dos elementos de processo utilizando a estratégia QaVaS é de fácil entendimento e utilização?

3. O módulo para adaptação de processos utilizando a estratégia QaVaS permite gerar um processo adequado às necessidades da empresa?

4. Pontos fortes e oportunidades de melhoria da ferramenta?

**IV. Críticas e sugestões:**