UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE CIÊNCIAS EXATAS E DA TERRA
PROGRAMA DE PÓS-GRADUAÇÃO EM SISTEMAS E COMPUTAÇÃO

UMA ABORDAGEM ANOTATIVA PARA GERÊNCIA DE VARIABILIDADES EM LINHAS DE PROCESSOS DE SOFTWARE: CONCEPÇÃO, IMPLEMENTAÇÃO E AVALIAÇÃO

Por
FELLIPE ARAÚJO ALEIXO
Tese de Doutorado

NATAL

Agosto, 2013

UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE CIÊNCIAS EXATAS E DA TERRA
PROGRAMA DE PÓS-GRADUAÇÃO EM SISTEMAS E COMPUTAÇÃO

Uma Abordagem Anotativa para a Gerência de Variabilidades em Linhas de Processos de Software: Concepção, Implementação e Avaliação

Banca Examinadora:

_____

Prof. Uirá Kulesza, D.Sc., PPgSC/UFRN (RN)

_____

Profa. Cláudia Maria Lima Werner, D.Sc., COPPE/UFRJ (RJ)

_____

Prof. Edson Alves de Oliveira Junior, D.Sc., DIN/UEM (PR)

_____

Prof. Fernando Marques Figueira Filho, D.Sc., DIMAP/UFRN (RN)

_____

Prof. Eduardo Henrique da S. Aranha, D.Sc., PPgSC/UFRN (RN)

UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE CIÊNCIAS EXATAS E DA TERRA
PROGRAMA DE PÓS-GRADUAÇÃO EM SISTEMAS E COMPUTAÇÃO

FELLIPE ARAÚJO ALEIXO

Uma Abordagem Anotativa para a Gerência de Variabilidades em Linhas de Processos de Software: Concepção, Implementação e Avaliação

Tese de Doutorado submetida à Coordenação do Programa de Pós-Graduação em Sistemas e Computação, do Centro Ciências Exatas e da Terra, da Universidade Federal do Rio Grande do Norte, como parte dos requisitos para obtenção de título de Doutor em Sistemas e Computação.

Orientador: Uirá Kulesza, D.Sc.

NATAL

Agosto, 2013

Resumo

A indústria de software encontra-se, nos dias de hoje, em um cenário altamente dinâmico, o qual reflete o mundo dos negócios e a sociedade como um todo. A demanda por sistemas de software é, dessa forma, cada vez mais crescente e visa atender a diferentes domínios. Nesse cenário, onde sistemas de software complexos precisam ser desenvolvidos com um excelente nível de qualidade e consumindo menor quantidade de recursos; aumenta a importância da definição de processos de software. Porém, tais processos necessitam ser devidamente adaptados aos contextos específicos de cada projeto, de forma a garantir a qualidade dos produtos de software desenvolvidos bem como o uso eficaz dos recursos disponíveis. Para que essa adaptação dos processos de software ocorra de forma eficaz, é necessário promover a reutilização de especificações de processos já existentes, bem como experiências e práticas de sucesso passadas. Este trabalho explora a adoção de técnicas de engenharia de linha de produtos de software de forma a promover a gerência de variabilidades de famílias de processos de software. Para um melhor entendimento do problema em questão foi realizada uma revisão sistemática da literatura, a qual identificou um conjunto de abordagens para a gerência de variabilidades em processos de software e suas principais características. Também foi realizada a proposta de uma abordagem anotativa para a gerência de variabilidades em processos de software, bem como um estudo exploratório visando a concretização dessa abordagem e a implementação de um protótipo de ferramenta para auxiliar na aplicação da mesma. A abordagem anotativa proposta foi, então, avaliada e comparada com a abordagem composicional do EPF *Composer*. Tais avaliações foram conduzidas por meio da realização de estudos empíricos e um experimento controlado. Nos estudos empíricos as abordagens foram avaliadas: (i) qualitativamente – de acordo com um conjunto de critérios de análise da especificação de suas variabilidades; e (ii) quantitativamente por meio da comparação de valores obtidos para métricas de modularidade, tamanho e complexidade para as especificações de uma mesma linha de processo usando as abordagens anotativa e composicionais. O experimento controlado teve como objetivo comparar as abordagens anotativa e composicional sob a perspectiva do esforço e entendimento das abordagens quando utilizadas na especificação de variabilidades em uma linha de processos de software. Os estudos evidenciaram diversos benefícios proporcionados pela abordagem anotativa no contexto de linhas de processos de software e seu potencial de integração com abordagens composicionais para auxiliar na modelagem de variabilidades em processos de software.

Palavras-chave: linhas de processos de software; gerência de variabilidades; técnicas composicionais e anotativas; linhas de produtos de software.

Abstract

Nowadays, the importance of using software processes is already consolidated and is considered fundamental to the success of software development projects. Large and medium software projects demand the definition and continuous improvement of software processes in order to promote the productive development of high-quality software. Customizing and evolving existing software processes to address the variety of scenarios, technologies, culture and scale is a recurrent challenge required by the software industry. It involves the adaptation of software process models for the reality of their projects. Besides, it must also promote the reuse of past experiences in the definition and development of software processes for the new projects. The adequate management and execution of software processes can bring a better quality and productivity to the produced software systems. This work aimed to explore the use and adaptation of consolidated software product lines techniques to promote the management of the variabilities of software process families. In order to achieve this aim: (i) a systematic literature review is conducted to identify and characterize variability management approaches for software processes; (ii) an annotative approach for the variability management of software process lines is proposed and developed; and finally (iii) empirical studies and a controlled experiment assess and compare the proposed annotative approach against a compositional one. One study – a comparative qualitative study – analyzed the annotative and compositional approaches from different perspectives, such as: modularity, traceability, error detection, granularity, uniformity, adoption, and systematic variability management. Another study – a comparative quantitative study – has considered internal attributes of the specification of software process lines, such as modularity, size and complexity. Finally, the last study – a controlled experiment – evaluated the effort to use and the understandability of the investigated approaches when modeling and evolving specifications of software process lines. The studies bring evidences of several benefits of the annotative approach, and the potential of integration with the compositional approach, to assist the variability management of software process lines.

Keywords: software process lines; variability management; compositional and annotative techniques; software product line engineering.

Sumário

Índice de Figuras

Índice de Tabelas

1 Introdução

Nos dias de hoje, os sistemas de software vêm ganhando cada vez mais espaço na vida das pessoas, mesmo que de forma discreta. O desenvolvimento de tecnologias de hardware e software vem estimulando, e sendo estimulado por esse avanço. A cada dia se torna mais comum que uma boa parte das pessoas interaja com algum dispositivo eletrônico, seja um eletrodoméstico, um aparelho de celular, um televisor ou mesmo um automóvel – e em todos eles está presente um sistema de software. Esse cenário gera uma demanda crescente por uma grande variedade de sistemas de software, voltados para as mais diversas plataformas, com diferentes níveis de complexidade, e que atendam aos mais diversos propósitos. Geralmente, essa demanda por sistemas de software está associada a um tempo de entrega cada vez menor, dada a dinâmica das mudanças do mundo moderno e a competição no mundo dos negócios. Por outro lado, deseja-se que o desenvolvimento de tais sistemas de software seja realizado com o menor custo possível, tornando-os mais lucrativos. Diante desse cenário, a tarefa de desenvolver sistemas de software utilizando a menor quantidade de recursos possível, e sem abrir mão da qualidade, não é trivial. Desafios como esses são enfrentados pela engenharia de software moderna.

Algumas estratégias de engenharia de software ganham força no contexto atual do desenvolvimento de sistemas de software, entre elas estão: os (i) processos de desenvolvimento de software (Pressman, 2006); e (ii) as técnicas e estratégias de reúso/reutilização (Braude, 2005). Os processos de desenvolvimento de software propõem-se a oferecer modelos de organização para o ciclo de vida de desenvolvimento e manutenção de um sistema de software. Tais modelos de organização visam orientar equipes de desenvolvimento, para que estas consigam atingir os seus objetivos no menor tempo e com a maior qualidade possível (Malik, 2008). A importância da definição de processos de software pode ser analisada sob dois aspectos: (i) influência direta da qualidade do processo na qualidade dos produtos resultantes da aplicação do mesmo (Malik, 2008); e (ii) possibilitar a avaliação e a evolução/melhoria do processo.

As estratégias de reutilização, com o passar do tempo, foram ganhando uma maior importância, deixando de ser apenas aplicadas de forma oportunista, e passando a ser intencionais e planejadas (Braude, 2005). Dentre as estratégias de reutilização, no contexto do desenvolvimento de sistemas de software, destaca-se a estratégia de linhas

de produtos de software – LPS (Pohl et al., 2005) (Linden et al., 2007) (Clements & Northrop, 2001). Uma LPS representa uma família de sistemas de software que compartilham um conjunto de características comuns, e se diferem por um conjunto de variabilidades. De forma geral, as características – *features* – representam unidades de desenvolvimento de um sistema de software – incrementos na funcionalidade ou em aspectos específicos do desenvolvimento do mesmo (Pohl et al., 2005). Durante o desenvolvimento de uma LPS são definidas e implementadas as características comuns – *features* mandatórias, da mesma forma que são definidas e implementadas as variabilidades – *features* opcionais e alternativas. Após a definição da LPS, é possível produzir sistemas de software – instâncias derivadas ou produtos específicos da LPS – que possuam um conjunto particular de características (Linden et al., 2007). Várias abordagens e ferramentas de LPS têm sido propostas nos últimos anos.

Existem duas abordagens comuns para a definição de LPSs (Kästner et al., 2008a): a abordagem composicional e a abordagem anotativa. A característica da abordagem composicional é que a implementação das *features* ocorre na forma de módulos distintos. Na derivação de instâncias (ou produtos) da LPS, um conjunto de módulos é composto, em tempo de compilação ou em tempo de implantação. As seguintes técnicas de projeto e implementação representam a abordagem composicional no contexto de LPS (Kästner et al., 2008a): (i) tecnologias de componentes; (ii) *frameworks* orientados a objetos; e (iii) programação orientada a aspectos. A característica da abordagem anotativa é que a implementação das *features* ocorre através de anotações implícitas ou explícitas no código fonte, indicando a localização de cada *feature* dentro do código da LPS. São exemplos típicos de anotações explícitas as instruções "*#ifdef*" e "*#endif*" de pré-processadores no estilo C/C++, as quais cercam o código relativo a uma dada *feature*. Outros exemplos segundo a abordagem anotativa para LPS (Kästner et al., 2008a): (i) anotações da linguagem Java; (ii) programação explícita; (iii) Spoon; e (iv) Gears.

1.1 Contextualização do Problema

Embora os processos de software sejam importantes no desenvolvimento de sistemas de software, especificar um processo de software não é uma tarefa simples. A especificação de um processo de software envolve um grande conhecimento nas várias disciplinas da engenharia de software, e na forma como são interligadas e integradas. A

especificação de um processo de software envolve conhecimentos em: gerência de projeto, análise de requisitos, arquitetura de sistemas, projeto detalhado e implementação de sistemas, testes de software, entre outras. Essa complexidade faz com que a tarefa de definição de um processo de desenvolvimento seja executada por um conjunto de pessoas, dado que a gama de conhecimentos é muito abrangente para ser dominada por apenas um profissional. Contudo, uma vez especificado um processo de software, os conhecimentos empregados na definição do mesmo podem ser reutilizados, nas subsequentes aplicações dessa especificação.

Um outro fator que dificulta a tarefa de especificação de processos de desenvolvimento de software é o fato de não haver um padrão consolidado para a modelagem de processos. Existem várias linguagens para especificação de processos de desenvolvimento de software, dentre elas, se destacam: SPEM – *Software Process Engineering Metamodel* (OMG, 2008), UMA – *Unified Method Architecture* (Haumer, 2007), e UML4SPM – *UML for Software Process Modeling* (Bendraou et al., 2005). Mesmo a linguagem SPEM, padrão desenvolvido pela OMG (OMG, 2012), não atende a todas as necessidades e facetas necessárias à especificação de um processo de desenvolvimento de software, como por exemplo aspectos relacionados à execução de processos de software (Chemuturi & Cagley Jr., 2010). A escolha de uma linguagem de especificação de processos de software geralmente se dá em função das necessidades do contexto. Exemplos de necessidades que podem motivar a escolha de uma linguagem de especificação de processos são: (i) inclusão de aspectos relacionados à execução de processos; (ii) adoção de uma dada ferramenta de autoria de processos; e (iii) utilização de especificações de processos anteriormente definidas. Havendo mudanças nas necessidades impostas pelo contexto, é possível a mudança para uma outra linguagem.

Por outro lado, os projetos reais de desenvolvimento de sistemas de software raramente são iguais entre si. Cada projeto de desenvolvimento de sistemas possui suas peculiaridades, tais como: natureza do sistema a ser desenvolvido, perfil da equipe envolvida, definição do orçamento, tempo de entrega, utilização de tecnologias específicas, entre outros. Esse fato implica que um processo de desenvolvimento genérico necessite ser customizado de acordo com as características do projeto. A necessidade de customização, permite a caracterização de variabilidades em processos de software, da mesma forma que em sistemas de software; podendo as mesmas ser representadas como características – *features* – de processo.

Caso as variabilidades em um conjunto de processos de software não sejam gerenciadas, pode-se chegar a um cenário de: (i) vários processos distintos (customizados) compartilhando uma mesma base comum; e (ii) se diferenciando por algumas variabilidades específicas, presentes em alguns membros desse conjunto. Dado que esse conjunto de processos "semelhantes" precisa constantemente passar por manutenções e evoluções, é natural acontecerem problemas como (Armbrust et al., 2009): redundância, inconsistência e um maior custo de manutenção. A redundância é inevitável visto que tais processos customizados compartilham uma série de características comuns; e é agravada quando mais de um processo existente é sujeito a mesma evolução. Uma evolução em um processo customizado, quando não é propagada para as demais customizações, acaba gerando inconsistências. Também são produzidas inconsistências quando evoluções nos processos customizados, buscando resolver um mesmo problema, são realizadas de formas diferentes. Nesse contexto, aumenta o custo de manutenção, visto que vários processos são mantidos paralelamente. Um conjunto de processos semelhantes, compartilhando características comuns e variáveis, pode ser denominado de família de processos de software. À especificação de uma família de processos de software pode ser denominada de linha de processos de software.

Em suma, podem ser destacadas as seguintes dificuldades com relação a gerência de variabilidades em processos de desenvolvimento de software:

- Complexidade inerente da tarefa de definição de processos de software, pelo fato de envolver vários áreas de conhecimento relacionados à engenharia de software;
- Utilização e alternância entre várias linguagem de especificação dos processos de software, devido a um conjunto de necessidades específicas;
- Necessidade de customização de um processo de software, visando atender as necessidades específicas de um dado projeto de desenvolvimento; e
- Problemas de redundância, inconsistência e altos custos de manutenção para manter um conjunto de processos independentes que variam em poucos aspectos.

1.2 Questões de Pesquisa

Com base na contextualização do problema que foi apresentada, foram extraídas quatro questões de pesquisa. Tais questões de pesquisa orientaram todo o

desenvolvimento do trabalho, bem como a concepção, o planejamento e a execução dos estudos realizados durante o desenvolvimento do mesmo, são elas:

Questão de pesquisa 1: quais os tipos de variabilidades que ocorrem em famílias de processos de software?

Questão de pesquisa 2: quais as abordagens existentes para a modelagem e gerência de variabilidades no contexto de processos de software? Quais os benefícios e limitações das abordagens existentes?

Questão de pesquisa 3: qual a viabilidade do uso de técnicas anotativas na definição de uma abordagem, com o devido suporte ferramental, para a gerência de variabilidades em processos de software?

Questão de pesquisa 4: quais os benefícios, complementaridade e limitações do uso de técnicas anotativas e composicionais no contexto de processos de software?

1.3 Objetivos

Neste trabalho, são explorados aspectos da gerência de variabilidade em famílias de processos de desenvolvimento de software, bem como a utilização das técnicas de LPS no contexto de processos de software. A aplicação das técnicas de LPS no contexto de processos de software visa permitir a adaptação e a customização de processos de desenvolvimento de software, que atendam a um dado conjunto de necessidades específicas. Como exemplo concreto de uma família de processos de software, teríamos os processos de software de uma dada empresa de desenvolvimento de sistemas, que desenvolve sistemas de software voltados para uma domínio específico. Trata-se de uma família de processos de software, visto que cada projeto apresenta demandas específicas de um dado cliente, e/ou como forma de otimizar os recursos disponíveis para o desenvolvimento do sistema de software em questão.

O objetivo desta tese de doutorado é (i) identificar as técnicas existentes para a gerência de variabilidades em processos de software; (ii) propor uma abordagem anotativa para a gerência de variabilidades em processos de software; e (iii) avaliar as abordagens composicional e anotativa para a gerência de variabilidades em processos de software, buscando evidenciar os benefícios e desvantagens de cada uma das técnicas investigadas. No contexto deste trabalho, também será buscada a compreensão de (i)

quais são os tipos de variabilidades que ocorrem em processos de software, (ii) em que momento estas variabilidades ocorrem, e (iii) quais as possíveis formas de tratamento para cada tipo de variabilidade.

Podem ser caracterizados como objetivos específicos desse trabalho:

1. Conduzir uma revisão sistemática da literatura que aborde o tema de gerência de variabilidades em processos de software;
2. Conceber e implementar uma abordagem anotativa para a gerência de variabilidades em processos de software, como meio para a definição de linhas de processos de software – LPrSs, e avaliar a sua viabilidade;
3. Investigar o uso das técnicas composicional e anotativa aplicadas à gerência de variabilidades em processos de software, visando a modelagem de LPrSs;
4. Conduzir avaliações empíricas entre exemplares das abordagens composicional e anotativa para a modelagem de LPrSs, evidenciando os pontos fortes e fracos de cada abordagem específica;
5. Caracterizar os diferentes tipos de variabilidades existentes em processos de software.

1.4 Limitações dos Trabalhos Atuais

Simidchieva et al. (Simidchieva et al., 2007) analisam a viabilidade da representação de variações em processos por meio de uma família de processos. Esse trabalho não se restringe a processos de software, tratando processos de forma geral. O trabalho propõe a utilização da linguagem Little-JIL na definição de família de processos, na qual são especificados: (i) passos individuais do processo e a coordenação entre os mesmos; (ii) comportamentos dos agentes que realizam tais passos; e (iii) a estrutura dos artefatos consumidos e produzidos por tais passos. A definição de um processo específico fica a cargo de um meta-processo de coordenação, responsável pela derivação de instâncias de processo. Embora o trabalho explore o contexto de linhas de processos, não se concentra na caracterização das variabilidades de processos nem na comparação com outras abordagens que buscam lidar com os mesmos problemas.

Armbrust et al. (Armbrust et al., 2009) definem uma abordagem de LPrS, denominada SCOPE, e detalha a primeira fase da mesma: definição do escopo da linha de processos. Nesta fase, são definidos os requisitos para a definição da LPrS, fase essa

que é ilustrada por meio de um estudo de caso realizado na agência aeroespacial Japonesa – JAXA. O trabalho restringe-se a detalhar a primeira fase da abordagem proposta, e sua avaliação não explora a comparação com abordagens com a mesma finalidade.

Martínez-Ruíz et al. (Martínez-Ruiz et al., 2009a) definem o framework SPRINTT, visando o suporte à institucionalização de processos. A institucionalização de processos implica na adaptação de processos específicos a partir de um conjunto de processos padronizados, definidos por uma organização. Tais processos padronizados são mantidos e evoluídos continuamente por meio da incorporação das lições aprendidas e das boas práticas. O trabalho define o paradigma *Variant Rich Process* – o qual adapta as técnicas de engenharia de linhas de produtos de software e engenharia de software orientada a aspecto para o contexto de processos de software. A modelagem de processos utiliza a notação vSPEM, permitindo a modelagem das variabilidades convencionais e transversais (*crosscutting*). Para permitir a modelagem de variabilidades transversais, é realizado um mapeamento dos conceitos da engenharia orientada a aspectos para o contexto de processos. O trabalho enfatiza apenas as variabilidades transversais em processos de software. O trabalho também não se propõe à análise e avaliação de outras abordagens com a mesma finalidade.

Barreto et al. (Barreto et al., 2010) propõem o desenvolvimento de LPrSs, objetivando a adaptação de processos que atendam um conjunto de necessidades específicas. A linha de processos é definida em termos de uma arquitetura de processos de software – representada como um conjunto de elementos de processos e a interação entre esses elementos – na qual são representadas as variabilidades do processo. A arquitetura de processos de software definida é utilizada para derivar instâncias de processos a partir da seleção de um conjunto de *features*. O trabalho não aborda nenhuma classificação das variabilidades em processos de software, nem estabelece comparativos com outras abordagens de mesma finalidade.

Simmonds et al. (Simmonds & Bastarrica, 2011a) investigam as ferramentas e notações disponíveis para representar variabilidades em processos de software. Nesse contexto, são testadas ferramentas como: fmp, FaMa-OVM, Clafer e SPLOT; e notações para a definição de processos (contendo variabilidades), tais como: SPEM 2 e vSPEM. Após a análise das ferramentas e notações disponíveis, é proposta a

combinação de algumas delas para a modelagem de variabilidades em processos de software, quais sejam: EPF *Composer*, SPLOT e MODISCO/AMW. O *Eclipse Process Framework* – EPF – *Composer* é utilizado para fazer a definição do processo; o SPLOT para fazer a definição das *features*; e o MODISCO é utilizado para fazer a integração (*weaving*) dos modelos de processo e de *features* em um modelo único. O trabalho realiza análises comparativas entre linguagens de modelagem de processos de software e linguagens de modelagem de *features*, e após a análise dos resultados obtidos, ele seleciona uma linguagem de modelagem para cada domínio. O trabalho propõe também a integração dos dois modelos em um modelo único que representaria as variabilidades em um processo de software. O trabalho, entretanto, de forma similar aos outros trabalhos relacionados, não classifica as variabilidades em processos de software, e não realiza comparativos com outras abordagens similares.

Como limitações comuns aos trabalhos atuais citados anteriormente, temos: (i) o pouco detalhamento dos tipos específicos de variabilidades de processos de software; (ii) a não realização de estudos sistemáticos de avaliação das abordagens propostas para gerência de variabilidades em processos de software; e (iii) a falta de exemplos, e estudos de caso, mais complexos envolvendo a aplicação das abordagens de LPrSs.

1.5 Metodologia

Com vistas à responder as questões de pesquisas e atender os objetivos definidos para este trabalho, foram planejadas e executadas as seguintes atividades:

1. Realização de uma revisão sistemática da literatura, visando identificar e caracterizar as abordagens de gerência de variabilidades em processos de software existentes;
2. Proposição de uma classificação preliminar dos diferentes tipos de variabilidades de LPrSs;
3. Proposição de um abordagem anotativa para a gerência de variabilidades em processos de software, como meio para o desenvolvimento de LPrSs;
4. Realização de um estudo exploratório, o qual visou avaliar a viabilidade da abordagem anotativa proposta para a gerência de variabilidades em LPrSs;
5. Adaptação e validação de um conjunto de critérios e métricas a serem utilizados na comparação das abordagens composicional e anotativa para a gerência de variabilidades em LPrSs; e

6. Realização de estudos e avaliações empíricas visando comparar exemplares das abordagens composicional e anotativa para o desenvolvimento de LPrSs.

1.6 Contribuições Esperadas

No início do desenvolvimento deste trabalho foram definidas as seguintes contribuições esperadas:

1. Um melhor entendimento sobre o domínio de problema relativo à gerência de variabilidades em processos de software;
2. Um panorama das abordagens existentes que atuam no gerenciamento de variabilidades em processos de software;
3. Uma abordagem anotativa para LPrSs, incluindo o suporte ferramental necessário à aplicação da mesma;
4. Critérios, parâmetros e métricas para a comparação das abordagens composicional e anotativa para a gerência de variabilidades em LPrSs; e
5. Resultados, evidências e conclusões relativas aos estudos empíricos comparativos, realizados com exemplares das abordagens composicional e anotativa para a gerência de variabilidades em LPrSs.

1.7 Organização do Documento

O Capítulo 2 apresenta a fundamentação teórica deste trabalho. O Capítulo 3 relata a realização de uma revisão sistemática da literatura, com a identificação das abordagens existentes para a gerência de variabilidades em processos de software. O Capítulo 4 apresenta a proposta de abordagem anotativa para LPrS, bem como relata a execução de um estudo exploratório, o qual avaliou a viabilidade da mesma. Os Capítulos 5 e 6 relatam a execução de estudos comparativos qualitativo e quantitativo, respectivamente, entre as abordagens composicional e anotativa aplicadas à modelagem de LPrSs. O Capítulo 7 relata a realização de um experimento controlado visando comparar as abordagens composicional e anotativa aplicadas à modelagem de LPrSs. Por fim, o Capítulo 8 apresenta as conclusões gerais, uma revisão das contribuições do trabalho; além das lições aprendidas, limitações e trabalhos futuros propostos.

2 Fundamentação Teórica

Este capítulo apresenta uma visão geral dos tópicos de fundamentação teórica necessários ao entendimento dos relatos do desenvolvimento do presente trabalho. O restante do capítulo está organizado da seguinte forma: a Seção 2.1 apresenta uma visão geral sobre o tema de processos de software e aspectos associados; a Seção 2.2 discute as tecnologias para especificação de processos de software; e por fim, a Seção 2.3 apresenta uma visão geral sobre LPrS.

2.1 Processos de Software

Processos de desenvolvimento de software, ou simplesmente processos de software, podem ser definidos como um conjunto de técnicas e tecnologias para apoiar, avaliar e melhorar atividades de desenvolvimento de software (Braude, 2005). A necessidade de buscar a especificação de processos surge do fato da qualidade dos produtos ser influenciada diretamente pelo processo utilizado para a sua produção (Chemuturi & Cagley Jr., 2010), não sendo diferente com os produtos de software. A norma ISO/IEC 15504 (ISO, 2004) define um processo como um conjunto de atividades que se inter-relacionam ou que interagem entre si, as quais transformam entradas em saídas. Tal conjunto de atividades serve de orientação para aqueles que serão os responsáveis pela execução do processo.

Processos são definidos no intuito de melhorar a forma pela qual o trabalho é realizado, visto que ao pensar no processo de forma organizada, é possível antecipar problemas e antever maneiras de prevení-los e resolvê-los. A gerência de tais processos possui quatro responsabilidades principais (Scott, 2003): (i) definir o processo, (ii) controlar o processo, (iii) medir o processo, e (iv) melhorar o processo. Na definição, são especificados os conceitos que irão compor o processo, bem como a organização das atividades do mesmo. Uma vez definido, o processo pode ser usado para orientar uma equipe no desenvolvimento de suas atividades. Ao ser utilizado para orientar o desenvolvimento de atividades de uma equipe, medidas da sua execução podem ser extraídas. Tais medidas são evidências de como está ocorrendo a execução do processo. As informações extraídas da execução e as medidas resultantes são a base para o melhoramento do processo, visando torná-los mais efetivos.

A interação de uma equipe de desenvolvimento com um processo (Braude, 2005) geralmente acontece por meio (i) de um modelo de processo, (ii) de um conjunto de documentos ilustrando o funcionamento do processo, ou (iii) por meio de um site Web, descrevendo os seus elementos e inter-relacionamentos. Importante notar que todas essas formas de interação suscitam suporte ferramental, e que a modelagem do processo, seja por meio de uma ferramenta ou de forma manual, é anterior a qualquer manipulação pela qual possa passar o processo (Chemuturi & Cagley Jr., 2010).

Cada vez mais a indústria de software precisa de técnicas e metodologias para auxiliar a entrega de produtos de software de alta qualidade no menor tempo possível. O uso do processo de software mais adequado é uma das chaves para atingir esse objetivo. Customizar e evoluir processos de software existentes para atender a uma variedade de cenários, tecnologias, culturas e escalas é um desafio que vem sendo recorrentemente exigido da indústria de software (Armbrust, 2010). O desafio envolve a adaptação de modelos de processos de software para atender as necessidades específicas de um dado projeto. Nesse cenário, é muito importante o reúso das experiências passadas na definição de desenvolvimento de processos de software para os novos projetos. O adequado gerenciamento e execução de um processo de software pode resultar em uma maior produtividade e em uma melhor qualidade dos sistemas de software produzidos.

Reusar as especificações de processos de software existentes, de maneira rápida e automática, customizando estas para cada novo projeto, ainda é um desafio para empresas de desenvolvimento de software. Ao longo dos últimos anos, alguns esforços e iniciativas têm sido propostos com o intuito de promover a definição e a reutilização de especificações de processos de software, tais como, o *Rational Unified Process* – RUP (Kruchten, 2004). Em paralelo, ferramentas têm sido propostas para viabilizar a automação durante tal processo de reutilização. O EPF *Composer* (Eclipse Foudation, 2010) e o *Rational Method Composer* (IBM, 2012) são exemplos de tecnologias que permitem a manipulação e a customização dos diferentes elementos presentes na especificação de processos. Um dos exemplos de processos cuja especificação está disponível para ser reutilizada é o OpenUP (EPF, 2011). Trata-se de um processo de desenvolvimento de software iterativo e incremental, derivado do Processo Unificado – *Unified Process* (Scott, 2003).

É possível a partir do OpenUP, derivar novos processos considerando as necessidades específicas de novos projetos. A ferramenta EPF *Composer*, que faz parte do EPF, pode ser usada para manipular os diferentes elementos presentes no processo e auxiliar durante a adaptação do processo para um contexto específico. Atividades como adicionar, remover ou modificar elementos do processo (atividades, papéis, artefatos, passos e guias) podem ser realizadas.

Apesar de oferecer funcionalidades para a manipulação de elementos presentes na especificação do processo, ferramentas como o EPF *Composer* e o *Rational Method Composer* não oferecem funcionalidades e mecanismos que permitam a gerência automática das variabilidades do processo, bem como a derivação automática de versões customizadas do mesmo. Os frameworks de processos existentes, tais como o RUP e OpenUP, oferecem possibilidades de customização e configuração, para a adequação dos mesmos às características dos projetos nos quais serão utilizados. Contudo, a manipulação dos elementos do processo, visando a customização do mesmo, pode se tornar inviável, custosa e sujeita a erros. Alguns dos frameworks de processo citados até explicitam elementos do processo (atividades, tarefas, passos, artefatos) que são opcionais, mas a maioria deles se refere a decisões tomadas durante a execução do processo em um projeto específico, e não durante as atividades de customização do projeto por um engenheiro de processo.

2.2 Tecnologias para Especificação de Processos de Software

A primeira iniciativa de padronização de modelos de processos de desenvolvimento de software foi formalizada pela OMG, na forma de um meta-modelo genérico para a especificação de processo de software. A OMG lançou a versão inicial (1.0) da especificação do SPEM – *Software and Systems Process Engineering Meta-Model* – em 2002. Uma evolução da especificação inicial foi lançada em 2005, intitulada SPEM 1.1, mesmo com pouco esforço de divulgação e algumas deficiências, como a semântica ambígua e a dificuldade de entender (Bendraou et al., 2005). A versão atual do SPEM (2.0) foi lançada em 2007, corrigindo as falhas da versão anterior, além de incluir a compatibilidade com a UML 2. Também foi lançado um conjunto de orientações para a migração de modelos de processo do SPEM 1.1 para o SPEM 2.0 (Bendraou et al., 2005).

O meta-modelo *Unified Method Architecture*, ou UMA, é na verdade uma versão intermediária entre o SPEM 1.1 e o SPEM 2.0. A proposta do UMA, publicada pela IBM (Haumer, 2007), e as suas definições influenciaram a definição da versão 2.0 do SPEM. O meta-modelo UMA foi desenvolvido para possibilitar a modelagem de processos de software nas ferramentas da IBM/Rational (IBM, 2012), mas depois também foi utilizada como base para uma ferramenta de código aberto ligada ao EPF – *Eclipse Process Framework* – chamada de EPF *Composer* (Eclipse Foudation, 2010).

Embora seja um padrão proposto pela OMG, a especificação SPEM provou com o tempo não ser tão genérica como pretendido, abrindo espaço para outras formas de especificação de processos de software (Martínez-Ruiz et al., 2008). As opções ao SPEM exploram características as quais ele não oferece suporte, tais como: (i) suporte à execução do processo; (ii) suporte à definição de variabilidades no processo; (iii) suporte à simulação e verificação de processos, entre outras. A partir do lançamento do meta-modelo SPEM, foram propostas formas alternativas para a especificação de processos de software visando atender a determinadas características não suportadas pelo memso, tais como: UML4SPM (Bendraou et al., 2005), vSPEM (Martínez-Ruiz et al., 2009a), DSL4SPM (Kerzazi & Robillard, 2010), MODAL (Koudri & Champeau, 2010), xSPEM (Bendraou et al., 2007), entre outras. Tal fato acarreta que a escolha da abordagem de especificação de processos passe a depender das necessidades específicas que se deseja atender.

2.3 Linhas de Processos de Software

O termo linhas de processos de software (LPrS) – *software process lines* – foi utilizado inicialmente por Rombach (Rombach, 2005), trabalho no qual propõe a integração de processos de software e linhas de produtos. Rombach (Rombach, 2005) concluiu que as variabilidades de processos, que discriminam as instâncias em uma família de processos de software, seriam em função dos objetivos do produto e do processo, bem como as características específicas do projeto. Na sequência, Washizaki (Washizaki, 2006a) definiu o que seria uma arquitetura para uma LPrS, incorporando similaridades e variabilidades entre possíveis instâncias de processos. A arquitetura, segundo Washizaki (Washizaki, 2006a), seria representada pelo aparato necessário para a definição de uma LPrS. Tal arquitetura teria por base algumas extensões do meta-

modelo SPEM, incluindo elementos, tais como: (i) pontos de variação; (ii) variantes; (iii) elementos opcionais; e (iv) relações de dependência; entre outros.

Em linhas gerais, uma linha de processos pode ser vista como uma família de processos que compartilha elementos de processo comuns e variáveis. A definição de uma LPrS inclui a definição de um subconjunto do processo que ocorre e que é comum a todas as instâncias da linha de processos – denominado geralmente de núcleo ou "*core*". Também são definidas as variabilidades da LPrS associadas à conjuntos de elementos opcionais ou alternativos. A Figura 1 ilustra um fragmento de uma LPrS, na qual a parte mandatória é composta por um papel – "desenvolvedor" – e duas tarefas: "projetar a solução" e "implementar a solução". A Figura 1 também ilustra duas variantes para a referida linha de processos (ambas opcionais): (i) inclusão da prática de testes de desenvolvedor e (ii) inclusão da prática de integrar o código ao repositório e gerar um executável (*build*) após a implementação de uma funcionalidade.

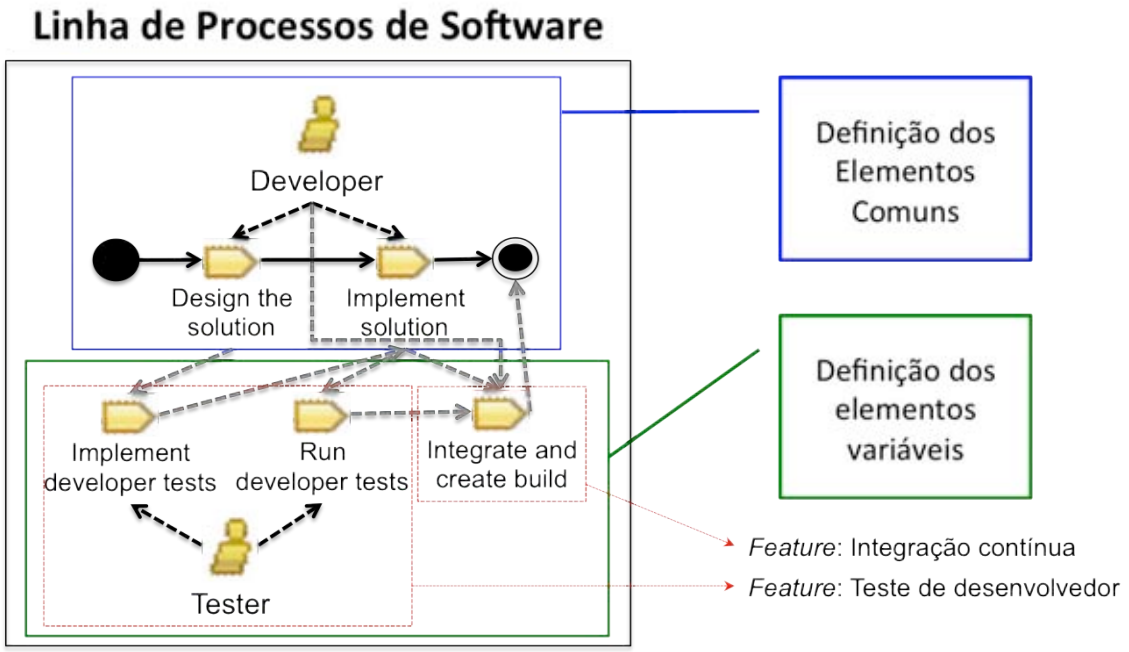


Figura 1. Ilustração de um fragmento de uma LPrS

A inclusão da prática de testes de desenvolvedor leva a inclusão de um papel – "testador" – e duas tarefas "implementar os testes de desenvolvedor" (antes de implementar a solução) e "executar os testes de desenvolvedor" (depois de implementar a solução). A inclusão da prática de integrar o código criado e gerar um executável leva a inclusão da tarefa de "integrar e criar um *build*", de responsabilidades do desenvolvedor, a qual acontece depois de implementar a solução ou depois de executar

os testes de desenvolvedor, se a primeira variabilidade for selecionada. Essa pequena ilustração de linha de processos pode gerar quatro instâncias de processo distintas: (i) sem as variantes, (ii) apenas com a primeira variante, (iii) apenas com a segunda variante e (iv) com ambas as variantes. A Figura 2 ilustra a relação genérica entre variabilidade, variante e ponto de variação. Em processos de software não é diferente, uma variabilidade em um processo de software está relacionada com um ou mais pontos de variação, que por sua vez estão associados a uma ou mais variantes. Um ponto de variação representa um ponto na especificação do processo de software em questão, o qual pode sofrer a inclusão de novos elementos relacionados à referida variabilidade. As variantes representam os elementos de processo e relacionamentos entre elementos de processo que podem ser inseridos na especificação base do processo de software.

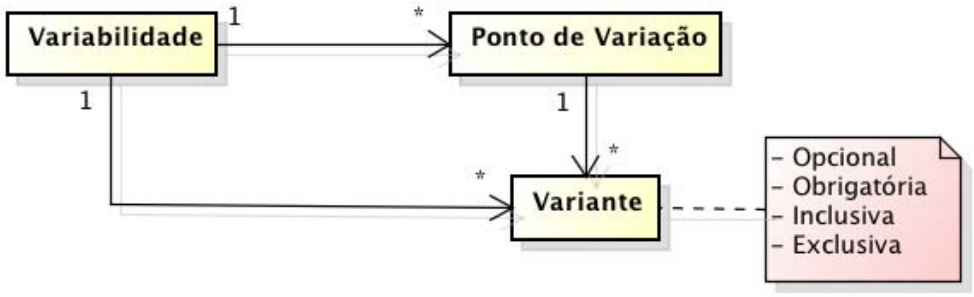


Figura 2. Relação entre variabilidade, variante e ponto de variação

Nos últimos anos, vários trabalhos têm sido publicados (Rombach, 2005) (Armbrust et al., 2009) (Barreto et al., 2010) (Aleixo et al., 2010a) explorando a adaptação de técnicas de linhas de produtos de software voltadas à gerência de variabilidades em linhas de processos. A quantidade e cobertura destes trabalhos sugere a consolidação de LPrSs enquanto um tema de pesquisa relevante, embora ainda em estágio inicial. No decorrer desse trabalho, serão investigadas várias abordagens para a gerência de variabilidades em LPrSs.

3 Gerência de Variabilidades em Processos de Software: Uma Revisão Quasi-Sistemática

Neste capítulo, serão apresentados os resultados de uma revisão *quasi*-sistemática da literatura realizada com o intuito de conhecer, em mais detalhes, as abordagens de gerência de variabilidades em processos de software. As seções deste capítulo estão organizadas da seguinte forma: a Seção 3.1 apresenta o protocolo utilizado para a revisão sistemática; a Seção 3.2 apresenta discussões sobre os resultados alcançados para as questões de pesquisa usadas para caracterizar os estudos primários selecionados; a Seção 3.3 apresenta discussões sobre as lições aprendidas com o estudo e as ameaças a validade do mesmo; a Seção 3.4 apresenta alguns trabalhos relacionados; e por fim, à Seção 3.5 apresenta as conclusões da revisão sistemática.

3.1 Protocolo da Revisão Sistemática

O protocolo adotado para a revisão sistemática é composto pelos seguintes tópicos: descrição do problema (Seção 3.1.1); questões de pesquisa (Seção 3.1.2); equipe que realizou a pesquisa (Seção 3.1.3); estratégia de busca (Seção 3.1.4); critérios de inclusão e exclusão de estudos primários (Seção 3.1.5); processo de seleção dos estudos primários (Seção 3.1.6); e estratégias de extração e síntese dos dados (Seção 3.1.7). Para a definição da estrutura do protocolo da revisão sistemática, foram considerados trabalhos de referência na área (Kitchenham et al., 2010) (Randolph, 2009) (Silva et al., 2011) (Kitchenham, 2004).

3.1.1 Descrição do Problema

A aplicação de técnicas de gerência de variabilidades em processos de software, proporciona a reutilização intencional e sistemática de especificações de processos de software. Também possibilita a definição de LPrSs, e a posterior derivação de instâncias, que são processos customizados, visando atender às necessidades de projetos específicos. A definição de uma LPrS proporciona: (i) a reutilização do conhecimento empregado na especificação da LPrS; (ii) a derivação de uma instância customizada de processo de software, em um tempo menor do que a customização manual; e (iii) a redução dos custos de manutenções, visto que quando são realizadas na LPrS, as mesmas são propagadas para todas as instâncias da família de processos de software. As manutenções e as evoluções em processos de software ocorrem visando permitir que os

mesmos alcancem novos níveis de maturidade, e possam incorporar o conhecimento produzido durante as suas aplicações. Para operacionalizar esse processo de reutilização, vêm sendo propostas algumas abordagens de gerência de variabilidades em processos de software. Com o objetivo de ter uma visão mais consistente e ampla sobre as abordagens de gerência de variabilidades em processos de software, foi idealizada a realização de uma revisão sistemática da literatura. Portanto, o objetivo desta revisão é identificar as abordagens existentes de gerência de variabilidades de processos de software, indicando que técnicas elas utilizam, de que forma vem sendo avaliadas e com que tipos de exemplos de variabilidades tais estudos são realizados.

3.1.2 Questões de Pesquisa

Com base no objetivo definido para esta revisão sistemática, foi definida a seguinte questão principal: “o estudo primário avaliado apresenta uma proposta de abordagem para o gerenciamento de variabilidades em processos de software? O que caracteriza essa proposta?”. Visando complementar esse questionamento, foram definidas 6 (seis) sub-questões (SQ). As sub-questões visaram revelar três aspectos dos estudos primários relativos à gerência de variabilidades em processos de software: (i) caracterização das abordagens propostas e exemplos de sua utilização (SQ1 e SQ2); (ii) classificação dos tipos possíveis de variabilidades de processos de software, bem como exemplos desses tipos específicos (SQ3 e SQ4); e (iii) estudos comparativos entre diferentes abordagens de gerência de variabilidades em processos de software (SQ5 e SQ6). As questões de pesquisa formuladas foram as seguintes:

SQ1: O estudo primário avaliado menciona exemplos práticos de alguma abordagem, proposta para o gerenciamento de variabilidades em processos de software?

SQ2: O estudo primário avaliado apresenta algum suporte ferramental para gerência de variabilidades em processos de software? O que caracteriza tal suporte ferramental?

SQ3: O estudo primário avaliado menciona alguma proposta para a classificação de variabilidades em processos de software? Quais os tipos propostos?

SQ4: O estudo primário avaliado apresenta estratégias de modelagem dos tipos específicos de variabilidades em processos de software? Quais?

SQ5: O estudo primário avaliado apresenta alguma análise ou estudo comparativo entre diferentes abordagens para a gerência de variabilidades em processos de software? De que tipo?

SQ6: O estudo primário avaliado define critérios específicos ou métricas para a comparação entre as diferentes abordagens para a gerência de variabilidades em processos de software? Quais?

3.1.3 Equipe de Pesquisa

A equipe de pesquisadores que realizou a revisão sistemática foi composta por 5 (cinco) membros, quais sejam: 2 (dois) pesquisadores alunos de doutorado, 2 (dois) pesquisadores alunos de mestrado e 1 (um) pesquisador doutor; todos ligados ao Programa de Pós-graduação em Sistemas e Computação (PPgSC) do Centro de Ciências Exatas e da Terra (CCET) da Universidade Federal do Rio Grande do Norte (UFRN). Importante destacar que os membros selecionados para a equipe já acumulavam a experiência de alguns anos no trabalho sob o tema de LPrS.

3.1.4 Estratégia de Busca

As buscas foram realizadas em três etapas. Na primeira etapa, foram utilizados os seguintes mecanismos de busca acadêmica: o *Google Scholar*, ACM *Digital Library* e *IEEEXplore Digital Library*. A escolha por tais mecanismos de busca se deu com base na reconhecida relevância dos mesmos no meio acadêmico. A escolha do *Google Scholar*, por exemplo, foi motivada pela crescente utilização desse mecanismo em revisões da literatura, e pela precisão do seu mecanismo de busca (Walters, 2009) (Lewandowski, 2010). Os artigos retornados na busca foram processados pela equipe responsável pela revisão sistemática, conforme descrito nas próximas seções.

As buscas foram realizadas em meados de Junho de 2012, e aconteceram utilizando variações de três palavras-chave: (i) "*software process lines*", (ii) "*software process variability*" e (iii) "*software process tailoring*". As duas primeiras palavras chave foram escolhidas devido a sua íntima relação com o que estava sendo buscado – gerência de variabilidades em processos de software, e a sua aplicação em LPrSs. Também foi considerado o termo "*software process family*" como sinônimo de "*software process line*". A terceira palavra chave foi escolhida em função do principal

objetivo da gerência de variabilidades em processos de software, que é a customização de processos de software. Não foi restringido um período específico para o ano de publicação dos estudos primários a serem retornados nas buscas. Ao todo, 16 (dezesseis) variações das palavras-chave base foram utilizadas, em função da utilização do plural em alguns dos termos, as quais foram: "*software process line*"; "*software process lines*"; "*software processes line*"; "*software processes lines*"; "*software process family*" ; "*software processes family*"; "*software process families*"; "*software processes families*"; "*software process variability*"; "*software process variabilities*"; "*software processes variability*"; "*software processes variabilities*"; "*software process tailoring*"; "*software processes tailoring*"; "*software process customization*"; e "*software processes customization*". As palavras-chave selecionadas foram buscadas em todo o texto do estudo primário, a saber: título, resumo, palavras-chave e corpo do artigo.

Na segunda etapa foi realizada uma pesquisa manual nos estudos primários publicados nas 10 (dez) últimas edições das principais conferências da área, quais sejam: ICSSP (*International Conference on Software and Systems Process*), PROFES (*International Conference of Product Focused Software Development and Process Improvement*) e ICSE (*International Conference on Software Engineering*). Para a identificação das dez últimas edições destas conferências foi utilizado o DBLP (*The DBLP Computer Science Bibliography*).

As dez últimas edições do ICSSP pesquisadas foram: (i) ICSSP 2012: Zurich, Switzerland; (ii) ICSSP 2011: Honolulu, Hawaii, USA; (iii) ICSP 2010: Paderborn, Germany; (iv) ICSP 2009: Vancouver, Canada; (v) ICSP 2008: Leipzig, Germany; (vi) ICSP 2007: Minneapolis, MN, USA; (vii) SPW/ProSim 2006: Shanghai, China; (viii) ISPW 2005: Beijing, China; (ix) ISPW 1996: Dijon, France; (x) ISPW 1994: Airlie, Virginia, USA.

As dez últimas edições do PROFES utilizadas foram: (i) PROFES 2012: Madrid, Spain; (ii) PROFES 2011: Torre Canne, Itália; (iii) PROFES 2010: Limerick, Irlanda; (iv) PROFES 2009: Oulu, Finlândia; (v) PROFES 2008: Monte Porzio Catone, Itália; (vi) PROFES 2007: Riga, Letônia; (vii) PROFES 2006: Amsterdam, Holanda; (viii) PROFES 2005: Oulu, Filândia; (ix) PROFES 2004: Kansai Science City, Japão; (x) PROFES 2002: Rovaniemi, Finlândia.

As dez últimas edições do ICSE utilizadas foram: (i) ICSE 2012: Zurich, Suiça; (ii) ICSE 2011: Waikiki, Honolulu, Hawaii, USA; (iii) ICSE 2010: Cape Town, África do Sul; (iv) ICSE 2009: Vancouver, BC, Canadá; (v) ICSE 2008: Leipzig, Alemanha; (vi) ICSE 2007: Minneapolis, MN, USA; (vii) ICSE 2006: Shanghai, China; (viii) ICSE 2005: St Louis, Missouri, USA; (ix) ICSE 2004: Edinburgh, Escócia, UK; (x) ICSE 2003: Portland, Oregon, USA.

A terceira, e última etapa das buscas compreendeu a investigação manual das referências bibliográficas dos 231 trabalhos retornados pelos mecanismos de busca. A pesquisa manual nas conferências mais relevantes da área retornou 35 artigos; e a investigação manual das referências bibliográficas dos artigos selecionados retornou outros 138 artigos. Foram utilizados na investigação das referências bibliográficas os mesmos critérios das etapas anteriores. Somando os resultados obtidos nas três etapas de busca, temos um total de 404 estudos primários. Cabe ressaltar que esse montante de artigos inclui alguns artigos duplicados, os quais foram eliminados em uma rodada posterior do estudo. O quantitativo de duplicatas será apresentado nos resultados.

3.1.5 Critério de Inclusão e Exclusão de Estudos Primários

Antes do processo de seleção dos estudos primários foram definidos os critérios de seleção e descarte de estudos primários a serem utilizados. O critério principal para a seleção de um dado estudo primário foi o fato do mesmo responder a alguma das questões de pesquisa definidas. Segundo esse critério, o foco da revisão sistemática concentrou-se em estudos primários que definem abordagens para o gerenciamento das variabilidades em processos de software, com o intuito de (i) definir uma arquitetura de LPrS; ou (ii) organizar componentes de processo reutilizáveis a serem utilizados na customização de processos de software que atendessem a requisitos específicos; ou (iii) customizar instâncias de processos de software para projetos específicos.

Ficou definido que os estudos primários que apresentassem discussões segundo este foco seriam selecionados por esta revisão sistemática da literatura. Foram definidos os seguintes critérios para a exclusão de estudos primários: (i) estar fora do escopo definido para a revisão sistemática; (ii) duplicatas de estudos primários já listados; (iii) não se tratar de um artigo publicado em um *journal*, periódico, conferência ou relatório técnico; e (iv) não estar escrito em inglês.

3.1.6 Processo de Seleção dos Estudo Primários

O processo de seleção iniciou analisando os 404 (quatrocentos e quatro) estudos primários retornados das buscas. Inicialmente, foram identificados os estudos primários que apareciam mais de uma vez na lista. Foram identificados 216 (duzentos e dezesseis) estudos primários repetidos, 53% do total, os quais foram eliminados. Outros 23 (vinte e três) estudos primários, 6% do total, foram eliminados por se tratarem de outros tipos de documentos, tais como: teses, dissertações e patentes. Também foram eliminados 5 (cinco) estudos primários, 5% do total, por não estarem escritos em inglês. Restaram, então, 160 (cento e sessenta) estudos primários a serem lidos e avaliados pelos pesquisadores participantes do estudo.

O processo de seleção dos artigos prosseguiu com a leitura de cada um dos 160 (cento e sessenta) artigos por dois pesquisadores. A seleção de quais pesquisadores seriam atribuídos para quais estudos primários se deu de forma aleatória. O processo de inclusão de cada um dos estudos primários se deu por meio de uma decisão consensual dos dois pesquisadores que analisaram o estudo primário. Cada pesquisador avaliou, em separado, se o foco do estudo primário em questão seria a gerência de variabilidades em processos de software, como técnica de reutilização de elementos de processos de software. Para o caso dos artigos que tiveram duas avaliações iguais, os mesmos foram automaticamente incluídos (avaliações favoráveis) ou excluídos (avaliações contrárias). Para os casos em que um dos pesquisadores foi favorável e outro contra, foi necessária uma discussão entre os pesquisadores, com o objetivo de chegar ao consenso. Havendo um consenso ao final da discussão, o processo se encerra para o referido estudo primário. Não havendo consenso, foi agendada uma discussão com o terceiro pesquisador para o desempate na decisão. Nos casos de não inclusão do estudo primário, ainda assim deveria haver um consenso com relação ao motivo da exclusão. Foram excluídos os estudos primários fora do escopo do estudo, por não abordarem como foco principal o tema de gerência de variabilidades em processos de software; ou que satisfizeram alguns dos critérios de exclusão, definidos anteriormente.

Ao final do processo de leitura dos 160 (cento e sessenta) artigos: (i) 40 (quarenta) artigos foram selecionados; (ii) 120 (cento e vinte) artigos foram descartados. A Tabela 1 apresenta a distribuição dos artigos não selecionados para a etapa de extração de dados. Dos 404 (quatrocentos e quatro) artigos que retornaram das

buscas foram eliminados 364 (trezentos e sessenta e quatro) artigos. Dentre os eliminados: 120 (cento e vinte) foram eliminados por não estarem alinhados com o foco definido pelo estudo; 216 (duzentos e dezesseis) foram eliminados por serem duplicatas de artigos já presentes no estudo; 23 (vinte e três) foram eliminados por não estarem em um dos formatos aceitos para o estudo; e 5 (cinco) foram eliminados por não estarem escritos em inglês, que a língua padrão definida para o estudo.

Tabela 1. Distribuição segundo os critérios dos estudos primários não selecionados

Quantidade	Percentagem	Critério
120	33%	Fora do escopo definido pelo estudo
216	60%	Duplicatas de estudos primários já presentes no estudo
23	6%	Formato diferente dos selecionados para o estudo
5	1%	Escritos em uma língua diferente do inglês
364	100%	TOTAL de artigos eliminados

3.1.7 Estratégias de Extração e Síntese de Dados

O processo de extração e síntese de dados foi realizado nos trabalhos selecionados. Foram extraídas informações, tais como: (i) ano de publicação, (ii) veículo de publicação, (iii) instituição de origem, (iv) se o trabalho é proveniente da academia ou da indústria, (v) país de origem, além das respostas para as questões de pesquisa levantadas inicialmente. Alguns desses resultados foram tabulados e sintetizados na forma de gráficos, para facilitar a interpretação desses resultados. Essas informações foram extraídas dos estudos primários selecionados visando oferecer um panorama sobre a origem e a temporalidade destas publicações. A caracterização da origem dessas publicações permitiu confirmar tratar-se de um tema de pesquisa disseminado por vários países, e não ser exclusivo da academia.

A Figura 3 ilustra a distribuição dos estudos primários selecionados segundo o ano de publicação. Nessa distribuição, pode-se perceber que a grande maioria dos estudos primários relacionados ao tema de gerência de variabilidades em processos de software são muito recentes, o que evidencia ser esse um tema em consolidação. A Figura 4 ilustra a distribuição dos artigos selecionados pelo tipo de instituição de origem (se acadêmica ou ligada à indústria) e por país. Analisando tais informações, percebe-se que embora seja um tema já considerado pela indústria, ainda tem um forte apelo acadêmico, visto ainda encontrar-se em fase de amadurecimento. Outra evidência que pode ser identificada em tal ilustração é o fato do tema estar sendo explorado em vários países, não estando restrito ao contexto de um único país.

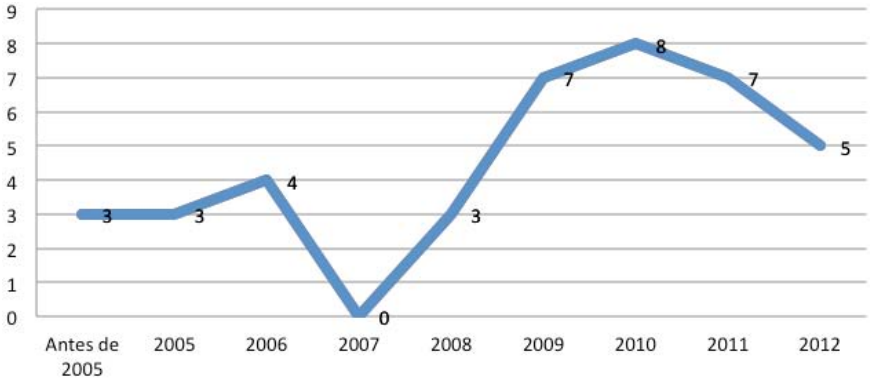


Figura 3. Distribuição dos estudos primários selecionados pelo ano de publicação

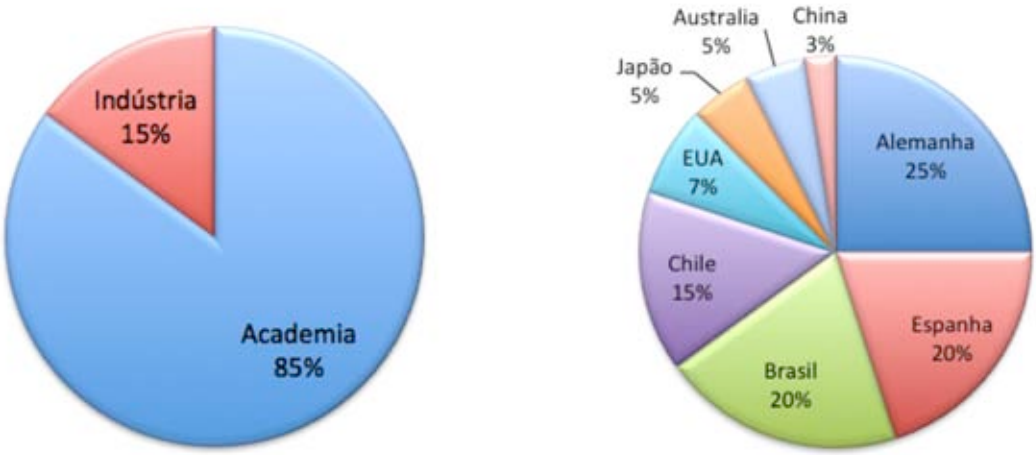


Figura 4. Distribuições dos estudos por tipo de instituição de origem e por país

A Tabela 2 apresenta os representantes da indústria que apoiaram o desenvolvimento de 6 (seis) dos estudos primários selecionados, representando 15% do total de estudos primários selecionados.

Tabela 2. Distribuição dos estudos primários ligados a indústria

Representante da Indústria	Artigos	Referências
Parceria entre o Fraunhofer IESE e a Agência de Exploração Aeroespacial Japonesa - JAXA	2	(Armbrust et al., 2008a) (Armbrust et al., 2009)
Fraunhofer IESE	4	(Jaufman & Münch, 2005) (Ocampo et al., 2005) (Armbrust et al., 2008b) (Armbrust, 2010)

3.2 Resultados Obtidos

Esta seção apresenta as sínteses dos resultados obtidos com a realização da revisão sistemática. Serão apresentados os resultados relativos a principal questão de pesquisa e das suas sub-questões. Os resultados de cada uma das questões de pesquisa serão apresentados em subseções específicas. Os resultados de cada questão de pesquisa serão brevemente discutidos, bem como será discutido o panorama geral que se forma com a junção de todos os resultados.

3.2.1 Abordagens de Gerência de Variabilidades em Processos de Software

A questão principal da revisão sistemática objetivou identificar e caracterizar as propostas de abordagens de gerência de variabilidades em processos de software. Durante a análise dos artigos selecionados, foram identificadas algumas propostas de abordagens para a gerência de variabilidades em processos de software, as quais estão ilustradas na Tabela 3. Na sequência desta seção, serão apresentadas visões gerais sobre cada uma das abordagens identificadas.

Tabela 3. Abordagens identificadas na primeira etapa da revisão sistemática

Ano	Autor Principal	Veículos	Co-autores	Descrição Geral	Ref.
1996	Stanley M. Sutton, Jr.	ISPW	Leon J. Osterweil	Definição de família de processos de forma análoga às famílias de produtos	(Sutton & Osterweil, 1996)
1996	Richard M. Lobsitz	HICSS	-	Definição de método para a montagem de uma definição de processo de software específico dado um projeto	(Lobsitz, 1996)
1999	Wolfgang Hesse	CAiSE	Jörg Noack	Definição de uma abordagem de modelagem de processos de software multi-variantes	(Hesse & Noack, 1999)
2005	Dieter Rombach	ISPW	-	Apenas fomenta a aplicação de SPL ao contexto de processos de software	(Rombach, 2005)
2006	Hinori Washizaki	PROFES e INDIN-IEEE	-	Técnica para definição de arquiteturas de linhas de processos	(Washizaki, 2006a) (Washizaki, 2006b)
2006	Raymond Madachy	SPW/ProSim	-	Definição de modelos reutilizáveis de estrutura e comportamento de processos de software	(Madachy, 2006)
2006	Sebastian Kiebusch	SPW/ProSim	Bogdan Franczyk, Andreas Speck	Definição da métrica de pontos de process-family	(Kiebusch et al., 2006)
2008	Ove Armbrust	*Software Process: Improvement and Practice Journal*, LNCS, ICSP e PROFES	Masafumi Katahira; Yuko Miyamoto; Jürgen Münch; Haruka Nakao; Alexis Ocampo; Olga Jaufman	Abordagem SCOPE – descreve os requisitos para a definição de escopo em uma LPrS, compõem uma abordagem maior	(Jaufman & Münch, 2005) (Ocampo et al., 2005) (Armbrust et al., 2008b) (Armbrust et al., 2008a) (Armbrust et al., 2009) (Armbrust, 2010)
2008	Tomás Matínez-Rúiz	*Software Quality Journal*, SERA, PROFES, ICEIS, EuroPSI e EUROMICRO	Félix García; Mario Piattini; Jürgen Münch;	Paradigma *Variant Rich Process*, baseado nas técnicas de LPrS e orientação a aspectos – propõem o vSPEM	(Martínez-Ruiz et al., 2008) (Martínez-Ruiz et al., 2009a) (Martínez-Ruiz et al., 2009b) (Martínez-Ruiz et al., 2011a) (Martínez-Ruiz et al., 2011b) (Martínez-Ruiz et al., 2012)

2009	Thomas Ternité	EUROMICRO	-	Descreve e compara a utilização de diferentes versões do *V-Modell* para a implementação de LPrS, com relação à representatividade dos modelos	(Ternité, 2009)
2009	Peter Killisperger	ICSP	Markus Stumptner, Georg Peters, Georg Grossmann, Thomas Stückl	Define uma arquitetura baseada em modelos para a instanciação de processos de software	(Killisperger et al., 2009) (Killisperger et al., 2010)
2009	Julio A. Hurtado Alegría	ICSSP e KREUSE	María C. Bastarrica; Alexandre Bergel; Alcides Quispe; Sergio F. Ochoa	Estrutura um processo de referência (Tutelkan) que pode ser adaptado para atender às necessidades específicas. Define uma meta linguagem de especificação de processos de software e suas variabilidades - CASPER	(Alegria & Bastarrica, 2009) (Alegria & Bastarrica, 2010) (Alegria et al., 2011) (Alegria & Bastarrica, 2012)
2009	Jieshan Li	CAiSE	Mingzhi Mao	Apresenta estudo de caso de configuração de processo de software com características baseadas no RUP	(Li & Mao, 2009)
2010	Ahilton Barreto	QUATIC e JUCS	Elaine Nunes; Ana Regina Rocha; Leonardo Murta	Propõe a definição de componentes reutilizáveis de processos (e suas variações), que podem ser combinados para atender contextos específicos	(Barreto et al., 2010) (Barreto et al., 2011)
2010	Fellipe A. Aleixo	ICEIS, LNBP e SBES	Marília A. Freire; Wanderson C. Santos; Uirá Kulesza; Daniel A. da Costa; Edmilson C. Neto	Definição de uma abordagem baseada em modelos para definição de LPrS, bem como a implantação dos processos derivados em motores de workflow (execução)	(Aleixo et al., 2010a) (Aleixo et al., 2010c) (Aleixo et al., 2012a) (Aleixo et al., 2012b)
2010	Andréa Magalhães Magdaleno	ICSE e SSBSE	-	Definição de uma abordagem para customização de processos de software baseada no gerenciamento de informações de contexto	(Magdaleno, 2010a) (Magdaleno, 2010b)
2010	Mario Cervera	ICSP	Manoli Albert, Victoria Torres, Vicente Pelechano	Definição de um framework metodológico e uma infraestrutura de software para a construção de métodos de produção de software	(Cervera et al., 2010)
2011	Jocelyn Simmonds	Relatório Técnico – Universidade do Chile (3)	María C. Bastarrica; Luis Silvestre; Alcides Quispe	Propõe a modelagem de variabilidades em modelos de processo, combinando notações e ferramentas (EPF *Composer*, SPLOT e MODISCO/AMW)	(Simmonds & Bastarrica, 2011a) (Simmonds et al., 2011b) (Simmonds et al., 2012)
2012	Raquel M. Pillat	ICSSP	Toacy C. Oliveira, Fabio L. Fonseca	Definição de uma abordagem de customização de processos de software com base em uma notação baseada em BPMN	(Pillat et al., 2012)

No inicio do seu trabalho, Sutton & Osterweil (Sutton & Osterweil, 1996) concluem que os processos de software também podem ser organizados em termos de famílias, da mesma forma que as famílias de produtos de software, no contexto de linhas de produtos de software. Sugere também a inter-relação entre famílias de produtos e famílias de processos. O trabalho discute que o método Booch de projeto

orientado a objetos pode ser visto como uma linha de processos, visto que não define um processo em específico, ao contrário, define um framework onde várias instâncias podem ser derivadas. Também discute como aconteceriam a reutilização e as customizações para atender as necessidades de projetos específicos. Contudo, o artigo se limita à discussões.

Lobsitz (Lobsitz, 1996) apresenta um método para montagem de uma definição de processo de software específica para um dado projeto. O método em questão consiste na definição dos passos necessários para criar uma definição de processo customizada para um projeto específico, com base em uma definição genérica de processo. Os passos se baseiam na (i) análise dos requisitos do projeto, (ii) definição da arquitetura preliminar para o sistema, (iii) definição das estratégias de teste e aceitação, (iv) definição do arcabouço do plano de projeto e, por fim, (v) a adequação do plano de projeto. O plano de projeto produzido para um projeto específico é feito em BPM.

No seu trabalho, Hesse & Noack (Hesse & Noack, 1999) apresentam uma abordagem para modelagem de processos de software baseada em “multi-variantes”. O contexto de desenvolvimento do trabalho foi uma grande organização bancária. Os autores concluíram que os “ingredientes do processo” (atividades, resultados, técnicas e ferramentas) poderiam ser combinados de diversas formas, a partir de um conjunto de variantes de processo definidas em um segundo nível. Em um primeiro nível, seriam definidos os elementos base a serem usados para montar os processos. Num segundo nível, seriam definidas as variantes de processo, como uma composição dos ingredientes definidos no primeiro nível. Esse trabalho, dentre os selecionados, foi o primeiro a discutir as variabilidades de processos de software e como gerenciá-las.

Rombach (Rombach, 2005) inicia o seu artigo com a seguinte constatação: “processos de software embora também variem entre projetos, ainda não são gerenciados de forma sistemática”. O artigo, portanto, motiva a necessidade de LPrS, de forma similar às linhas de produtos de software. Permitindo a organização de processos de acordo com as suas similaridades e diferenças, permitindo a uma melhor adequação às necessidades de projetos específicos. O autor se detêm a apresentar motivações para o reúso proativo de processos de software, bem como os desafios decorrentes desse caminho. Defende a aplicação das técnicas de linhas de produtos de software, já razoavelmente validadas, ao contexto de processos de software.

No trabalho de Washizaki (Washizaki, 2006a) (Washizaki, 2006b), é proposta uma técnica para a definição de arquiteturas de linhas de processos, incluindo similaridades e variabilidades. Nesse caso, as variabilidades no processo são representadas por meio de uma extensão ao meta-modelo SPEM. A técnica para adequação de processos utiliza duas estratégias: (i) definição de componentes de processo e (ii) geradores. São propostas as extensões ao SPEM para permitir a expressão de similaridades e variabilidades nos fluxos de atividades de processos, representados por meio de diagramas de atividade UML.

Madachy (Madachy, 2006) apresenta uma abordagem que visa promover o reúso de modelos, estruturais e comportamentais, para processos de software. A modelagem em questão utiliza a técnica de modelagem de dinâmica de sistemas, muito utilizada para a definição de modelos que possam ser simulados. O artigo foca na definição de "blocos de construção" – componentes de processo reutilizáveis, incluindo a definição do comportamento usando dinâmica de sistemas. Tais blocos de construção seriam, então, utilizados para montar modelos de processos específicos e simulá-los.

No seu trabalho, Kiebusch et al. (Kiebusch et al., 2006) apresentam a definição de uma métrica, denominada de "pontos de família de processos", ou "*process-family-points*", no contexto da engenharia de família de processos. A análise da referida métrica seria equivalente a análise de pontos de função. Embora o trabalho seja relativo a famílias de processo no contexto de sistemas orientados a processos, tornou-se interessante para a revisão sistemática por ser o único trabalho a definir uma métrica para estimar o tamanho e esforço associados a uma família de processos.

Armbrust et al. (Armbrust et al., 2008a) (Armbrust et al., 2008b) (Armbrust et al., 2009) (Armbrust, 2010) (Armbrust & Rombach, 2011) definem uma abordagem de LPrS denominada de SCOPE, para a qual o autor se detêm a explicar a primeira fase da mesma. Para essa primeira fase, são definidos os requisitos para a definição do escopo da LPrS. Para embasar a proposta, é apresentado um estudo de caso realizado na agência aeroespacial Japonesa – JAXA.

Segundo a abordagem proposta por Matínez-Ruíz (Martínez-Ruiz et al., 2008) (Martínez-Ruiz et al., 2009a) (Martínez-Ruiz et al., 2009b) (Martínez-Ruiz et al., 2011a) (Martínez-Ruiz et al., 2011b) (Martínez-Ruiz et al., 2012), é definido o

framework SPRINTT – visando o suporte à institucionalização de processos. Dentro desse framework, é definido o paradigma *Variant Rich Process*, o qual é suportado por adaptações das técnicas de engenharia de linhas de produtos de software e engenharia de software orientada a aspectos. A modelagem de processos acontece utilizando-se a notação vSPEM, permitindo a modelagem das variabilidades convencionais e transversais (*crosscutting*). Para permitir a modelagem de variabilidades transversais, é realizado um mapeamento dos conceitos da engenharia de software orientada a aspectos (AOSE) para o contexto de processos.

No trabalho proposto por Ternité (Ternité, 2009), é descrita a utilização de versões diferentes do *V-Modell* para a implementação de LPrSs. Tais implementações são realizadas com o intuito de comparar a representatividade dos modelos, com relação a modelagem de variabilidades em processos de software. Segundo a proposta do autor, o trabalho fica restrito ao nível de modelagem.

Killisperger et al. (Killisperger et al., 2009) (Killisperger et al., 2010) apresentam uma arquitetura baseada em meta-modelos para a instanciação de processos de software. O trabalho em questão foi desenvolvido no contexto da Siemens – na qual os processos de software são definidos de acordo com as necessidades especiais de cada projeto. Uma característica importante dos processos dentro da empresa é que se assemelham bastante à *workflows*. O trabalho descreve um framework que executa a instanciação de decisões tomadas por gerentes de projeto, a fim de garantir que os processos derivados sejam gerados de forma consistente e coerente. Para tanto, são checadas restrições cadastradas nos modelos.

A proposta de abordagem apresentada por Alegría et al. (Alegria & Bastarrica, 2009) (Alegria & Bastarrica, 2010) (Alegria et al., 2011) (Alegria & Bastarrica, 2012) estrutura um processo de referência que pode ser adaptado para atender às necessidades de projetos específicos. Processo de referência esse que possui cinco fases: (i) *launching*, (ii) *diagnostic*, (iii) *formulation*, (iv) *implementation* e (v) *closure*. No artigo, a aplicação da abordagem proposta é avaliada. Para essa avaliação são definidas métricas diretas, a saber: (i) fator de reúso, (ii) fator de esforço de adaptação, (iii) fator de esforço de ajustamento; e métricas derivadas: (iv) adaptabilidade, (v) eficiência da adaptação e (vi) efetividade da adaptação.

Em seu trabalho, Li & Mao (Li & Mao, 2009) apresentam um estudo de caso de adequação de processos de software baseados no RUP. O objetivo do trabalho é apresentar como adequar processos de software baseados no RUP com base nas características específicas do software a ser desenvolvido. Os autores discutem como tornar o processo mais adaptável às características do software e às demandas do projeto. São definidos coeficientes de interdependência entre elementos de processo para auxiliar o processo de adequação.

Na abordagem proposta por Barreto et al. (Barreto et al., 2010) (Barreto et al., 2011), é proposto o desenvolvimento de LPrSs, objetivando a adaptação de processos que atendam a um conjunto de necessidades específicas. A linha de processos é definida em termos de uma arquitetura de processos de software – representada como um conjunto de elementos de processos e a interação entre esses elementos – na qual são representadas as variabilidades do processo. A partir da definição de uma arquitetura de processos de software, é possível derivar instâncias de processos, por meio da seleção das *features* desejadas para uma instância em específico.

No trabalho proposto por Aleixo et al. (Aleixo et al., 2010a) (Aleixo et al., 2010c) (Aleixo et al., 2012a) (Aleixo et al., 2012b), é proposta uma abordagem orientada a modelos para a definição, derivação, implantação e execução de processos de software. Nesse trabalho, é adaptada uma ferramenta de derivação de produtos – GenArch – para o contexto de processos de software, com o intuito de modelar LPrS. Nessa abordagem, uma especificação de processo é anotada com variabilidades, as quais são refletidas em um modelo de *features* e um modelo de configuração, que armazena o mapeamento de *features* específicas para elementos de processo. A abordagem prevê a derivação de instâncias da linha de processos e a preparação de tais instâncias para serem implantadas em um motor de *workflow*, onde o referido processo seria executado e monitorado.

Magdaleno (Magdaleno, 2010a) (Magdaleno, 2010b), inicialmente, apresenta uma abordagem de engenharia de linha de processos baseada em informações de contexto. As informações de contexto são utilizadas como parâmetros para a instanciação de processos a partir dessas linhas de processo. Na sequência, propõe a evolução dessa abordagem para adequação de processos de desenvolvimento de software, buscando a otimização. A autora defende um equilíbrio entre colaboração e

disciplina como norteadores para a adequação “otimizada”, bem como a automação de alguns passos desta adequação. Visando esse objetivo, são desenvolvidas as funções utilitárias para calcular a “maior colaboração” e a “maior disciplina”, as quais são consideradas para determinar a seleção de componentes de processo que satisfaçam todas as restrições de um projeto específico, e maximizem a colaboração e a disciplina.

Cervera et al. (Cervera et al., 2010) apresentam um framework metodológico e uma infraestrutura de software para a construção de métodos de produção de software. O framework proposto baseia-se em técnicas da engenharia dirigida por modelos. O framework suporta desde a especificação do método de produção de software até a geração de uma ferramenta CASE que o suporte. O referido framework é dividido em três partes. A primeira parte trata do projeto de método – construção dos elementos (tarefas, produtos, etc.) reutilizáveis, os quais serão armazenados em um repositório de método. A segunda parte trata da configuração de método – os elementos do modelo de método são associados à ativos do repositório de ativos (modelos, meta-modelos, transformações, etc.). Na terceira parte acontece a implementação do método, onde as transformações são executadas para obter uma ferramenta CASE completa, ou parcial.

No trabalho de Simmonds et al. (Simmonds & Bastarrica, 2011a) (Simmonds et al., 2011b) (Simmonds et al., 2012), inicialmente, são pesquisadas as ferramentas e as notações disponíveis para representar variabilidades em processos de software. Nesse contexto, são testadas ferramentas como: fmp, FaMa-OVM, Clafer e SPLOT; e notações para a definição de processos (contendo variabilidades), tais como: SPEM 2 e vSPEM. Após a análise das ferramentas e notações disponíveis, propõe a combinação de algumas delas para a modelagem de variabilidades em processos de software, quais sejam: EPF *Composer*, SPLOT e MODISCO/AMW. O EPF *Composer* é utilizado para fazer a definição do processo; o SPLOT para fazer a definição das *features*; e o MODISCO é utilizado para fazer a integração (*weaving*) dos modelos de processo e de *features* em um modelo único.

Finalmente, Pillat et al. (Pillat et al., 2012) introduzem uma abordagem de adequação de processos de software gerando ao final especificações de fluxos de tarefas segundo a notação BPMN. Para tanto, os autores definem uma extensão à notação BPMN para representar processos de software. Os autores justificam a escolha deste tipo de notação com base na possibilidade de execução dos processos definidos segundo

essa notação. A referida abordagem se propõe a entender como mecanismos de adequação podem ser representados no SPEM, e como mecanismos similares poderiam ser incorporados na notação BPMN. Por fim, é apresentado um estudo de caso.

3.2.2 Exemplos Práticos de Utilização das Abordagens

A primeira sub-questão estava interessada em investigar se os trabalhos analisados apresentam exemplos práticos de alguma abordagem proposta para o gerenciamento de variabilidades em processos de software. A Figura 5 apresenta uma síntese dos resultados para a sub-questão 1.

Figura 5. Síntese dos resultados da sub-questão 1

Ficou evidenciado que mais da metade (57%) dos artigos selecionados apresentaram exemplos práticos, mesmos que superficiais. Porém, um número considerável de artigos (43%) apresentou tais abordagens dissociadas de exemplos práticos. A ausência de exemplos práticos se justifica pelo caráter motivacional e inicial dos trabalhos em questão. O resultado final para esta questão evidencia que foram encontradas propostas em: (i) estágio inicial de caracterização da proposta; (ii) estágio intermediário com exemplos simplificados; e (iii) estágio de consolidação da proposta, com exemplos mais elaborados, e incluindo comparações com outras abordagens.

3.2.3 Suporte Ferramental

A segunda sub-questão estava interessada na apresentação de suporte ferramental para a gerência de variabilidades em processos de software. A Figura 6 apresenta uma síntese das respostas para a sub-questão 2. Conforme apresentado nessa figura, os resultados evidenciam que apenas alguns trabalhos apresentam suporte ferramental para abordagens apresentadas (32%).

Figura 6. Síntese dos resultados da sub-questão 2

A Tabela 4 apresenta as abordagens para as quais foi descrito um suporte ferramental e uma descrição geral das ferramentas envolvidas. O suporte ferramental foi caracterizado pela menção nos estudos primários pesquisados de ferramentas de software que foram desenvolvidas para auxiliar na aplicação de uma dada abordagem.

Tabela 4. Identificação das abordagens com suporte ferramental

Referência	Descrição do Suporte Ferramental
(Armbrust et al., 2008a) (Armbrust et al., 2008b) (Ocampo et al., 2005)	Se propõem a descrever uma abordagem que adapta e implanta o V-Modell XT na empresa Josef Witt GmbH. Para por em prática o processo customizado, foi necessário treinamento nas ferramentas INNOVATOR, Dimensions, Augeo e QA-Center. Em um trabalho anterior do mesmo grupo, Ocampo apresenta uma ferramenta para análise de similaridades em processos de software – SPEARSIM.
(Killisperger et al., 2010)	Cita a implementação de um protótipo de um sistema encarregado de instanciar um processo de referência em uma unidade de negócios da Siemens. Em especial, o referido protótipo foi usado para validar as rotinas de checagem de consistência de uma versão intermediária da instância de processo gerada.
(Cervera et al., 2010)	O suporte à abordagem proposta é proporcionado pelas seguintes ferramentas: (i) o EPF *Composer* como o editor de método; (ii) um repositório de fragmentos de método – utilizados a partir de uma extensão do EPF *Composer* que recupera os fragmentos desejados do repositório; e (iii) um guia (*wizard*) para o projeto de métodos.
(Aleixo et al., 2010a) (Aleixo et al., 2010b) (Aleixo et al., 2012a)	Cita a adaptação da ferramenta de derivação de produtos – GenArch – para o contexto de processos de software – permitindo a: (i) anotação de *features* em elementos de processo e (ii) derivação de instâncias de processos a partir da seleção das *features* desejadas.
(Barreto et al., 2010) (Barreto et al., 2011)	Cita o desenvolvimento de uma ferramenta Web que permite: (i) a definição de componentes de processo, associando à *features*; (ii) definição de uma LPS e (iii) derivação de processos da LPS. Explica que a funcionalidade da ferramenta Web é suportada pela definição de um repositório de componentes de processo reutilizáveis.
(Martínez-Ruiz et al., 2011b)	Cita a adaptação do meta-modelo SPEM para incluir os conceitos da AOSE, possibilitando dessa forma a definição de variabilidades transversais (*crosscutting*) em processos de software, resultando na notação vSPEM. Descreve um editor para a notação vSPEM.

(Alegria et al., 2011) (Alegria & Bastarrica, 2012)	O primeiro trabalho propõem uma estratégia generativa para a adequação de processos de software, utilizando: um modelo de contexto (SPCM), um modelo de processo de software (SPEM), e uma transformação ATL, gerando um processo adaptado (SPEM). O segundo trabalho formaliza a abordagem e a nomeia como CASPER – *Context Adaptable Software Process Engineering*. Inclui como ferramentas os editores específicos para os modelos definidos pela abordagem.
(Simmonds et al., 2012)	Propõem a combinação de notações e ferramentas para a formalização de modelos de processo de software incluindo suas variabilidades – usa: (i) o EPF para a definição do processo de software; (ii) o SPLOT para a modelagem de *features* do processo e (iii) a ferramenta Modisco/AMW para estabelecer restrições entre ambos modelos.

Analisando a distribuição destes artigos com relação ao ano em que foram publicados, temos: 1 (um) em 2005, 1 (um) em 2008, 5 (cinco) em 2010, 3 (três) em 2011 e 3 (três) em 2012. Onde a maioria deles foi publicada de 2010 em diante, este é mais outro indício que embora já existam muitas abordagens publicadas, a grande maioria delas ainda está em fase de amadurecimento.

3.2.4 Classificação de Variabilidades em LPrSs

A sub-questão 3 estava interessada em saber se o artigos avaliados apresentavam alguma proposta para classificação de variabilidades em processos de software. A Figura 7 apresenta uma síntese das respostas para a sub-questão 3. A maioria (75%) não apresenta nenhuma proposta de classificação de variabilidades, os demais artigos (25%) apresentam propostas de classificação de variabilidades em processos de software. Uma classificação das variabilidades em processos de software é útil, pois: (i) pode orientar os engenheiros de processo na identificação das mesmas (*checklist*); (ii) podem ser desenvolvidas técnicas específicas para cada tipo de variabilidade; e (iii) pode auxiliar na seleção da abordagem de LPrS a ser utilizada, dado que alguns tipos de variabilidade podem ser melhor modelados com uma abordagem específica.

A Tabela 5 apresenta as propostas de classificação de variabilidades em processos de software que foram identificadas durante a análise dos trabalhos selecionados. Estão inclusos nessa tabela os trabalhos que diferenciaram, de alguma forma, alguns tipos de variabilidades que ocorrem em processos de software. Estas propostas de classificação de variabilidades foram consolidadas em uma versão preliminar, a qual será apresentada na Seção 8.2.1 deste documento.

Figura 7. Síntese dos resultados da sub-questão 3

Tabela 5. Propostas de classificação de variabilidades em processos de software

Referência	Tipos de Variabilidade de Processos de Software
(Hesse & Noack, 1999)	Classificou os critérios que influenciam na seleção de variantes em dois grande grupos: (i) metas e requisitos do projeto (ex.: necessita ser entregue rapidamente, orçamento limitado, sistema distribuído, altos requisitos de segurança, etc.) e (ii) restrições de processo (ex.: usar bibliotecas de classes específicas, aplicação orientada à processos de negócio, usar estruturas de dados complexas, alta cooperação, etc.).
(Washizaki, 2006a) (Washizaki, 2006b)	A variabilidade dos processos de software é representada por pontos de variação e variantes – os pontos de variação são: (i) atividades; (ii) artefatos de entrada e saída e (iii) papéis.
(Ternité, 2009)	Define os seguintes tipos de variabilidades em processos de software: (i) positiva – são adicionados novos elementos ou relacionamentos; (ii) negativa – elementos ou relacionamentos não excluídos; (iii) extensão – elementos ou relacionamentos são estendidos; (iv) substituição – elementos ou relacionamentos são substituídos.
(Martínez-Ruiz et al., 2008) (Martínez-Ruiz et al., 2009b) (Martínez-Ruiz et al., 2011a) (Martínez-Ruiz et al., 2011b) (Martínez-Ruiz et al., 2012)	Nos trabalhos de Martínez-Ruiz et al. foram caracterizados vários tipos de variabilidades em processos de software. Dentre os citados, destacam-se os seguintes tipos de variabilidade: (i) nos elementos do modelo do processo de software (genericamente); (ii) em atividades e tarefas do processo; (iii) em artefatos; (iv) em recursos (ferramentas para auxiliar na execução de tarefas do processo e orientações sobre as mesmas); (v) no fluxo de controle; (vi) no fluxo de artefatos; (vii) para a manutenção da consistência (em função da dependência entre *features*); (viii) de natureza transversal (*crosscutting*); e (ix) de natureza localizada (pontual).
(Alegria & Bastarrica, 2009) (Alegria & Bastarrica, 2012)	No primeiro trabalho, há uma caracterização de atributos de contexto em três dimensões: (i) dimensão de projeto (ex.: duração e riscos do projeto), (ii) dimensão de equipe (ex.: tamanho e capacidades da equipe), (iii) dimensão do produto (ex.: maturidade e domínio de conhecimento). No segundo trabalho, exemplifica as *features* de processos de software como: (i) propriedades de processo (tipo de ciclo de vida e nível de maturidade), (ii) elementos de método (fragmentos de método), (iii) elementos de processo (componentes de processo), (iv) processo com elementos de método (módulos), (v) elementos de *plugin* de método (componentes reutilizáveis, padrões de processo, processos e configurações), (vi) pacotes de processo e (vii) pacotes e categorias de método.

(Barreto et al., 2010) (Barreto et al., 2011)	Os tipos de variabilidade definem o efeito da escolha de uma *feature*, sejam elas: (i) mandatórias, (ii) opcionais e (iii) alternativas. No segundo trabalho, os componentes de processo são caracterizados como pontos de variação – pois podem ser instanciados de diferentes formas. E cada ponto de variação tem os componentes variantes que se encaixam ao mesmo.
(Simmonds & Bastarrica, 2011a) (Simmonds et al., 2011b) (Simmonds et al., 2012)	No primeiro trabalho, não se define explicitamente tipos específicos de variabilidades em processos de software, apenas discute-se como as mesmas podem ser modeladas com as notações SPEM e vSPEM – mecanismos de variabilidade: *contributes*, *replaces*, *extends* e *extends-replace*, por um lado, e pontos de variação e variantes por outro. No segundo trabalho, apresenta-se os tipos de variabilidades identificados na modelagem de um processo de engenharia de requisitos: (i) atividade de requisitos – opcional; (ii) atividade de projeto – opcional; (iii) tarefa de especificar requisitos – alternativa; (iv) tarefa de estabelecer uma linha de base para os requisitos – alternativa; (v) tarefa de reunião para acordos de integração – opcional; (vi) tarefa de executar os casos de teste – opcional; (vii) artefato de linha de base para requisitos – alternativa; (viii) papel de analista – alternativa. No terceiro trabalho, define-se que a variabilidade dos processos é determinada pela variabilidade dos seus elementos: opcionais e alternativas.

3.2.5 Modelagem de Tipos Específicos de Variabilidades

A sub-questão 4 estava interessada em estratégias apresentadas para a modelagem de tipos específicos de variabilidades em processos de software. A Figura 8 apresenta uma síntese das respostas para a sub-questão 4. Mais uma vez, pode-se perceber que a grande maioria dos artigos (90%) não apresentou nada relativo a modelagem de tipos específicos de variabilidades em processos de software. Apenas 10% apresentou algum tipo de estratégia para a modelagem de tipos específicos de variabilidades, mesmo que sem explicitar muitos detalhes.

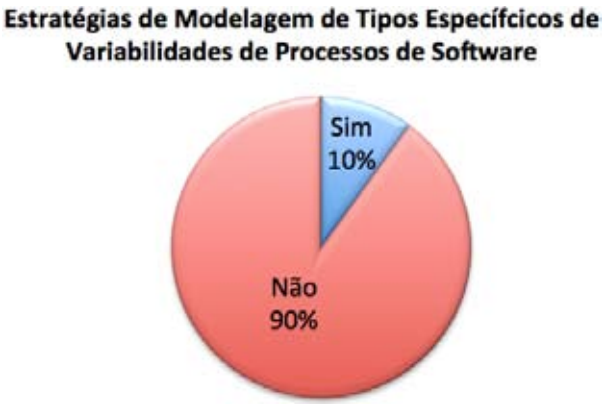


Figura 8. Síntese dos resultados da sub-questão 4

A Tabela 6 apresenta as abordagens que apresentaram estratégias para a modelagem de tipos específicos de variabilidade em processos de software. Nessa mesma tabela, as estratégias serão apresentadas de forma geral.

Tabela 6. Modelagem de tipos específicos de variabilidades de processos de software

Referência	Tipos Específicos de Variabilidade e Estratégias de Modelagem
(Washizaki, 2006a)	Propõe um conjunto de passos para a modelagem das variabilidade em processos de software, que vão desde a definição do núcleo (*core*) do processo com os pontos de variação; até a derivação de processos customizados de acordo com as características desejadas.
(Martínez-Ruiz et al., 2011a)	Apresenta uma validação empírica de que o vSPEM é mais adequado para a modelagem de variabilidades em processo que o SPEM – no qual é modelado um processo com ambas notações, usando os mecanismos de variabilidade do SPEM e pontos de variação e variantes do vSPEM.
(Alegria et al., 2011)	Propõe a customização do processo de software através de uma transformação, que utiliza como entradas um modelo de processo e um modelo de informações contextuais, e gera um processo customizado.
(Simmonds et al., 2012)	O processo é modelado através do EPF e as suas variabilidades através de um modelo de *features* – propõe um processo de fusão (*weaving*) desses modelos, produzindo um modelo de processo com variabilidades.

3.2.6 Análises ou Estudos Comparativos

A sub-questão 5 estava interessada em saber se os artigos avaliados apresentam alguma análise ou estudo comparativo entre diferentes abordagens para a gerência de variabilidades em processos de software. A Figura 9 apresenta uma síntese das respostas para essa sub-questão. Observou-se que a grande maioria (85%) dos artigos não apresentou nenhuma comparação entre abordagens diferentes para a gerência de variabilidades. Apenas 15% apresentou alguma análise ou estudo comparativo entre abordagens diferentes, mesmo que alguns estudos tenham se limitado a analisar ferramentas e estratégias de representação de variabilidades em processos de software.

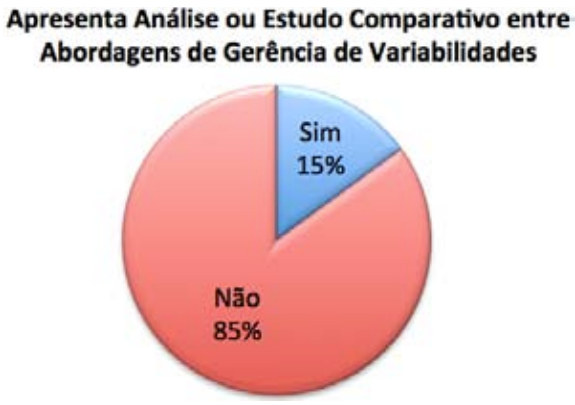


Figura 9. Síntese dos resultados da sub-questão 5

A Tabela 7 lista os artigos que apresentaram análises ou estudos comparativos de aspectos relativos às abordagens para a gerência de variabilidade em processos de software. Também são descritos de forma geral as análises e/ou estudos comparativos que foram identificados nos trabalhos investigados.

Tabela 7. Trabalhos que apresentam análises ou estudos comparativos de diferentes abordagens de gerência de variabilidade em processos de software

Referência	Tipo do Trabalho	Descrição do Trabalho
(Jaufman & Münch, 2005)	Estudo comparativo quantitativo	Uma das seções do referido artigo apresenta um estudo comparativo do método então proposto (EPAc – *Emergent Process Acquisition*) e o método de adequação proposto pelo V-Model. Definiu um modelo GQM para o estudo com o intuito de avaliar a produtividade. No geral, o método proposto apresentou melhores resultados.
(Martínez-Ruiz et al., 2011a)	Estudo empírico	Apresenta um estudo empírico para validar se a notação para processos de software vSPEM é mais apropriada que o SPEM na modelagem de processos de software com variabilidades.
(Simmonds et al., 2011b)	Estudo empírico	Discute os mecanismos de modelagem de variabilidades em processos de software com as notações SPEM e vSPEM. Discute como alguns formalismos usados em LPS podem ser usados para modelar variabilidades expressas em vSPEM: modelo de *features* básico, modelo de *features* baseado em cardinalidade e modelo de variabilidade ortogonal. O trabalho também apresenta uma breve análise do suporte ferramental disponível para a modelagem de *features* e variabilidades ortogonais, sendo eles: fmp, Clafer, SPLOT e FaMa-OVM.
(Simmonds & Bastarrica, 2011a)	Estudo empírico	Investiga como as notações SPEM e vSPEM podem ser utilizadas para modelar diferentes tipos de variabilidade em processos de software. Também são analisadas notações genéricas para a modelagem de variabilidades: modelo de *features* e modelo de variabilidade ortogonal. Por fim, faz uma análise do suporte ferramental disponível, incluindo: Clafer, EPF *Composer*, FaMa-OVM, fmp, Hydra, SPLOT, VEdit e XFeatures.
(Martínez-Ruiz et al., 2012)	Revisão sistemática da literatura	Apresenta uma revisão sistemática da literatura a respeito dos requisitos e abordagens para a adaptação de processos de software – explorando questões como: Quais os tipos de operações de adaptação? Quais as notações de processo utilizadas como base para suportar a adaptação dos mesmos?
(Aleixo et al., 2012a)	Estudo comparativo qualitativo	Apresenta uma comparação qualitativa das abordagens composicional do EPF *Composer* e anotativa do GenArch-P na modelagem de uma mesma LPrS. São definidos sete critérios de comparação com base em um estudo anterior no contexto de linha de produtos de software, são eles: (i) modularidade, (ii) rastreabilidade, (iii) detecção de erros, (iv) granularidade, (v) uniformidade, (vi) adoção e (vii) gerenciamento sistemático de variabilidades. A análise das modelagens indicaram que a abordagem anotativa do GenArch-P obteve os melhores resultados dados os critérios.

Um artigo contendo uma revisão sistemática também fez parte das respostas afirmativas a essa questão. Após a análise da revisão sistemática identificada (Martínez-Ruiz et al., 2012), foi identificado que a mesma está em consonância com os resultados obtidos por meio desse estudo, com relação a: (i) carência por abordagens de gerência sistemática de variabilidades em processos de software; (ii) carência por estudos comparativos entre diferentes abordagens; e (iii) carência de estudos de caso, exemplos práticos, experimentos e aplicações da abordagens em contextos mais complexos.

3.2.7 Critérios e Métricas para Comparação de Abordagens

A sub-questão 6 estava interessada em saber se os artigos avaliados definiram critérios específicos para a comparação de abordagens diferentes para gerência de variabilidades em processos de software. Dado que os estudos que apresentaram análises ou estudos comparativos foram identificados na questão anterior (SQ5), essa sub-questão investigou quais foram os critérios e/ou métricas que foram utilizadas em tais estudos. A Tabela 8 apresenta uma síntese dos critérios e/ou métricas definidas, além de uma breve descrição de como elas foram aplicadas.

Tabela 8. Trabalhos que apresentam critérios para a comparação de diferentes abordagens de gerência de variabilidade em processos de software

Referência	Critérios de Comparação	Como Foram Aplicados
(Jaufman & Münch, 2005)	(i) produtividade, (ii) qualidade do produto desenvolvido e (iii) satisfação da equipe de projeto	Foram utilizados grupos de estudantes para aplicar as abordagens de adequação de processos de software – aquisição de processos emergentes e V-Model. O estudo consistiu de três interações do desenvolvimento incremental de um protótipo.
(Martínez-Ruiz et al., 2011a)	(i) compreensibilidade dos diagramas de processo com variabilidades e (ii) eficiência dos mecanismos de variação	Para medir os critérios estabelecidos foram realizados treinamentos e aplicados questionários, dos quais foram coletados: (i) o tempo de resposta, (ii) a correção das respostas e (iii) a eficiência (razão entre correção e tempo).
(Simmonds et al., 2011b)	Para as abordagens de modelagem de variabilidade: (i) expressividade, (ii) compreensibilidade, (iii) conformidade aos padrões e (iv) disponibilidade de suporte ferramental. Para o suporte ferramental: (i) formatos suportados, (ii) método de definição de features, (iii) análises de consistência realizadas, (iv) tipo de interface, (v) disponibilidade e (vi) usabilidade.	Inicialmente avaliou a adequação de algumas abordagens de modelagem de variabilidade em processos de software: SPEM 2, vSPEM, modelo de features e OVM. Porém, os autores não descreveram em detalhes como foram comparadas as abordagens. A comparação do suporte ferramental foi realizada através da análise das características de cada ferramenta, e compilação destes resultados em uma tabela.
(Simmonds & Bastarrica, 2011a)	(i) viabilidade do uso de determinadas abordagens na especificação de variabilidades em LPrSs	Foram modelados exemplos de variabilidades com as abordagens listadas, e analisados tais resultados. Complementou analisando o suporte ferramental disponível.
(Aleixo et al., 2012a)	(i) modularidade, (ii) rastreabilidade, (iii) detecção de erros, (iv) granularidade, (v) uniformidade, (vi) adoção e (vii) gerenciamento sistemático de variabilidades	Uma mesma LPrS foi modelada utilizando-se as duas abordagens sendo investigadas: a abordagem composicional do EPF *Composer* e a abordagem anotativa do GenArch-P. Ao final, os resultados obtidos foram analisados sob a ótica dos critérios definidos.

(Martínez-Ruiz et al., 2012)	Analisou as publicações em termos de: (i) ano de publicação, (ii) elementos de processo que sofreram variação, (iii) tipos de relacionamento entre elementos que sofreram variação, (iv) tipos de variações direta nos elementos de processo, (v) mecanismos de variação indireta em processos de software, (vi) tipos de variabilidade, (vii) notações utilizadas para a modelagem de variabilidades, (viii) notações de modelos de processo que foram alvo das customizações, (ix) abordagem para a customização de um processo de software, (x) tipo de assistência durante o processo de customização.	As análises desses aspectos foi realizada por meio do desenvolvimento da revisão sistemática da literatura. Tal revisão sistemática teve como questão principal "quais abordagens de customização e adaptação de processos de software podem ser encontradas na literatura e quais requisitos em termos de variabilidade elas irão implicar?". De forma complementar foram definidas as seguintes sub-questões: (i) "quais elementos do modelo de processo de software são utilizados quando adaptações são consideradas?"; (ii) "que tipo de operações de customização são utilizadas nos modelos de processo existentes?"; (iii) "quais notações de modelagem de processos são usadas como base para dar suporte às customizações nos modelos de processo?"; e (iv) "como os processos são customizados para atender às características da organização ou de um projeto?".

Os referidos trabalhos apresentaram as seguintes conclusões: (i) Jaufman et al. (Jaufman & Münch, 2005) concluíram que a abordagem de "adequação de processos emergentes" contribui para um desenvolvimento mais eficiente de sistemas de mais alta qualidade; (ii) Martínez-Ruiz et al. (Martínez-Ruiz et al., 2011a) concluíram que a compreensibilidade dos mecanismos de variabilidade do vSPEM é 1,26 vezes melhor que do SPEM, mas é 3,64 vezes pior em termos da compreensibilidade dos diagramas gerados; (iii) Simmonds et al. (Simmonds et al., 2011b) concluíram que o modelo de *features* seria o mais indicado para a modelagem de variabilidades em processos de software, e que a ferramenta SPLOT seria a melhor para esse propósito; (iv) Simmonds et al. (Simmonds & Bastarrica, 2011a) concluíram que o vSPEM foi mais intuitivo de se utilizar na prática, e que seriam necessários mais estudos de caso para definir as melhores opções em termos da formalização das *features* e do suporte ferramental; (v) Martínez-Ruiz et al. (Martínez-Ruiz et al., 2012) atestaram que nenhuma notação incluindo todos os requisitos para a adaptação de processos de software foi encontrada, com a exceção da linguagem vSPEM; (vi) Aleixo et al. (Aleixo et al., 2012a) concluíram que o GenArch-P alcançou melhores resultados em cinco dos sete critérios definidos, comparado com o EPF *Composer*, e destacou também as possíveis vantagens de integração entre as abordagens composicional e anotativa.

A carência de estudos comparativos envolvendo diferentes abordagens relacionadas com a gerência e variabilidades em processos de software indica que esse

tópico de pesquisa ainda está em fase de amadurecimento. Por outro lado, a existência de estudos comparativos indica que algumas abordagens propostas já se encontram minimamente maduras; e a tendência é que estudos comparativos venham a crescer e trazer um feedback muito importante para o desenvolvimento destas abordagens.

3.3 Proposta de Classificação de Variabilidades em Processos de Software

Esta seção apresenta uma proposta preliminar de classificação de variabilidades em processos de software, com base: (i) nos resultados d a revisão sistemática; (ii) na análise de *frameworks* de processo existentes (OpenUP, RUP, Scrum, *Extreme Programming*) (Eclipse Foundation, 2012); e (iii) na experiência prática de processos derivados de projetos com a indústria, incluindo estudos de extração de linhas de processos a partir de projetos existentes (Aleixo et al., 2010c) (Aleixo et al., 2012a).

O modelo generativo proposto por Czarnecki & Eisenecker (Czarnecki & Eisenecker, 2000) é usado como referência para situar os tipos de variabilidades existentes. O modelo generativo é organizado na forma de: (i) espaço de problema, (ii) espaço de solução e (iii) conhecimento de configuração. O espaço de solução agrupa as *features* de alto nível, relacionadas à definição de um processo de software, tais como: métodos ou técnicas usados em diferentes disciplinas de desenvolvimento; nível de formalismo arquitetural; grau de maturidade desejado para o processo; ou qualidade para a realização de atividades específicas do processo. O espaço de solução engloba os elementos de processo – blocos de construção – que irão compor uma especificação de LPrS, tais como: atividades, tarefas, papéis, artefatos de entrada e saída, práticas, guias, ferramentas, etc. Por sua vez, o conhecimento de configuração determina o mapeamento entre abstrações e *features* do espaço de problema em elementos concretos de definição do processo do espaço de solução.

A Figura 10 ilustra os domínios de classificação de variabilidades em processos de software – espaço de solução e espaço de problema. Alguns trabalhos relacionados (Simidchieva et al., 2007) (Aleixo et al., 2010a) (Barreto et al., 2010) (Martínez-Ruiz et al., 2011a) (Simidchieva & Osterweil, 2011) (Simmonds et al., 2011b) (Armbrust & Rombach, 2011) exploram a ocorrência de variabilidades em LPrS no espaço de

solução. Nesses trabalhos, qualquer elemento da especificação de uma LPrS (atividades, tarefas, papéis, guias, artefatos, etc.) pode ser mandatório, alternativo ou opcional.

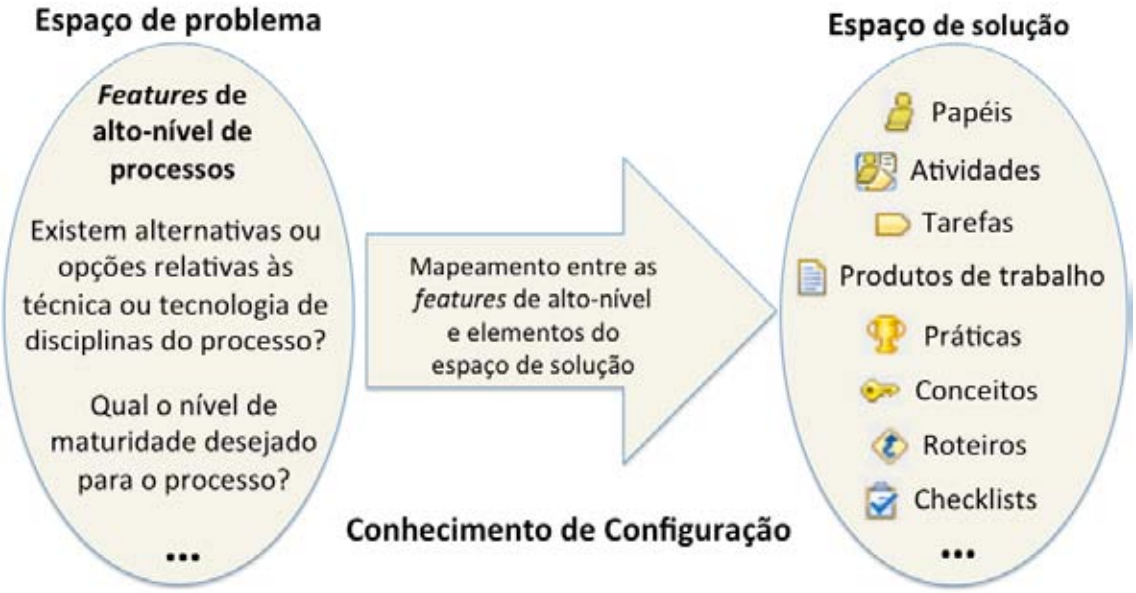


Figura 10. Domínios de classificação de variabilidades em processos de software

Para facilitar o entendimento da classificação proposta, os tipos específicos serão apresentados relacionados ao domínio ao qual pertencem. A Tabela 9 apresenta nossa classificação para variabilidades em processos de software. As variabilidades no espaço de solução foram classificadas segundo os critérios de: (i) relativas aos elementos de processo – atividades, tarefas, papéis, práticas, roteiros, etc.; (ii) granularidade – que dizem respeito aos níveis de granularidade (grossa ou fina); (iii) natureza de especificação – que indicam se a variabilidade ocorre de forma pontual (localizada) ou dispersa (transversal) a um dado elemento de processo; e (iv) tempo de *binding* – que diz respeito ao momento no qual a variabilidade pode ser aplicada ao processo. Os momentos possíveis para a aplicação de variabilidades em uma LPrS são: (a) momento de modelagem/definição do processo; (b) momento de implantação do processo (por exemplo, em uma ferramenta de gerência de projetos ou gerência de mudanças); e (c) momento de execução do processo.

A estratégia de modelar variabilidades no espaço de solução não permite a abstração das escolhas e decisões de alto nível, relativas a conjuntos de elementos de processo. Dessa forma, nossa classificação preliminar considera também variabilidades no espaço de problema, as quais foram classificadas em dois tipos: (i) relativas às técnicas e tecnologias associadas às disciplinas de processo, tais como, requisitos, projeto, implementação, testes e gerência de projeto; e (ii) relativas aos níveis de

modelos de maturidade de processos – indica se uma dada instância de processo precisa atender a um determinado nível de um modelo de maturidade, como por exemplo, os níveis G ou F do MPS.Br.

Tabela 9. Classificação proposta para variabilidades de processos de software

Nível de Abstração	Critério	Classificação
Espaço de Solução	Elementos de processo	(1) papel; (2) atividade, (3) tarefa, (4) artefato e (5) orientação
	Granularidade	(6) fina e (7) grossa
	Natureza de especificação	(8) pontual e (9) transversal
	Tempo de *binding*	(15) modelagem, (16) implantação e (17) execução
Espaço de Problema	Técnicas e tecnologias associadas à disciplina de processo	(10) requisitos, (11) projeto (*design*), (12) implementação, (13) testes, (14) gerenciamento de projetos
	Nível de modelo de maturidade de processos	(20) MPS.Br G, (21) MPS.Br F, (22) MPS.BR E, (23) MPS.Br D (...)

3.3.1 Variabilidades no Espaço de Problema

As variabilidades relativas às disciplinas do processo estão associadas às opções e alternativas relacionadas à alguma disciplina do processo, tais como: requisitos, projeto, implementação, testes, gerência de projetos, gerência de configuração e mudanças. Um exemplo de variabilidade dessa classe é a escolha por uma técnica específica de modelagem de requisitos, dentre várias alternativas. Ainda de acordo com esse exemplo, tem-se que o OpenUP define como técnica padrão para a especificação de requisitos a especificação de casos de uso. Contudo, é possível definir algumas alternativas à essa técnica, como por exemplo: especificação de requisitos por meio de "estórias do usuário" ou por meio de "*product backlog*". A escolha por uma determinada técnica de especificação de requisitos implica na inclusão de vários elementos de processo associados à mesma. Ainda segundo o exemplo apresentado, em caso de seleção da técnica baseada em casos de uso, os seguintes elementos de processo seriam incluídos na instância de processo: a prática "desenvolvimento orientado a casos de uso"; os artefatos "modelo de casos de uso" e "caso de uso"; os exemplos de "evolução do modelo de casos de uso" e "especificação de caso de uso"; a tarefa de "detalhamento de cenários de caso de uso"; entre outros.

Como exemplo de variabilidade associada à disciplina de projeto (*design*) temos, a escolha entre duas alternativas com relação ao grau de formalismo na documentação da arquitetura do sistema a ser desenvolvido: podemos optar por ter uma arquitetura bem documentada ou de acordo com a proposta de projeto ágil (Ambler, 2011). Cada uma das alternativas implicará na inclusão de conjuntos específicos de elementos de processo. No caso da opção pela arquitetura bem documentada, por exemplo, teremos a inclusão dos seguintes elementos: os conceitos “mecanismos de projeto” e “arquitetura de software”; as diretrizes “analisar o projeto” e “evoluir o projeto”; o roteiro “como adotar a prática do projeto evolucionário”; o *template* de “projeto”; as atividades “refinar a arquitetura” e “projetar a solução”; entre outros. No caso de seleção desenvolvimento da arquitetura como um projeto ágil, seriam selecionados os seguintes elementos: o papel de “testador”; as práticas de “desenvolvimento dirigido a testes” e “integração contínua”; os artefatos “casos de teste” e “log de testes”; atividades como “implementar os testes de desenvolvedor” e “integrar e criar um build”; entre outros.

Já as variabilidades relativas a níveis de modelos de maturidade de processos, são caracterizadas pela inclusão de uma série de elementos de processo visando alcançar todos os resultados definidos pelo nível de maturidade especificado no modelo em questão. Para exemplificar esse contexto fazemos uso do modelo de maturidade de processos definido pelo MPS.Br. Como exemplo específico, teríamos a seleção pelo engenheiro de processo do nível G (parcialmente gerenciado) do MPS.Br. Com base em cada resultado a ser alcançado são adicionados (caso o núcleo do processo já não possua) os elementos que deverão prover tal resultado. Por exemplo, para alcançar o resultado 6 para a gerência de projetos – GPR 6, o qual define que os riscos do projeto são identificados e o seu impacto, probabilidade de ocorrência e prioridade de tratamento são determinados e documentados; são adicionados elementos de processo, tais como: a prática “ciclo de vida orientado à riscos”, o artefato “lista de riscos”, a orientação “gerenciando os riscos”, e o roteiro “como adotar a prática de ciclo de vida orientado a riscos”. Para alcançar o resultado 1 para a gerência de requisitos – GRE 1, que define que o entendimento dos requisitos é obtido junto aos clientes e usuários; são adicionados elementos de processo, tais como: o papel “interessado”, o artefato de “glossário”, o conceito de “evoluir para continuamente obter feedback e melhorar”, e a atividade “avaliar resultados”. E assim por diante com os demais resultados esperados para as gerências de projeto e requisitos, próprias do nível G do MPS.Br.

3.3.2 Variabilidades no Espaço de Solução

As variabilidades dos elementos de processo caracterizam-se pela possibilidade de incluir ou não elementos de processos específicos em uma dada instância de uma linha de processos. Nesse tipo de variabilidade, podemos ter: (i) definição de elementos de processo opcionais; ou (ii) definição de alternativas para um dado elemento de processo. Teríamos como exemplos de elementos que poderiam ser definidos como opcionais: um papel, uma atividade, uma tarefa, um artefato ou mesmo uma orientação. Como exemplo de alternativas para um dado elemento, poderíamos ter um conjunto de *templates* para ilustrar um mesmo artefato, variando em níveis de complexidade e formalismo. Nesse caso, o engenheiro de processos poderia escolher qual *template* específico seria o mais indicado para um dado projeto, com base no contexto do mesmo.

Já a classificação de variabilidades de processo por granularidade, diferencia as variabilidades como granularidade grossa e granularidade fina. Uma variabilidade de granularidade grossa se caracteriza por estar associada a pacotes (conjuntos) de elementos de processo. Por exemplo, tomando por base a linha de processo baseada no OpenUP, temos a escolha entre uma das várias alternativas de ferramentas de especificação de requisitos. Onde a seleção de uma dada ferramenta implica na inclusão de um pacote de elementos de processo, contendo elementos de processo específicos como: os exemplos de “evolução do modelo de casos de uso”, “modelo de casos de uso – fase de elaboração” e “modelo de casos de uso – fase de concepção”; e a orientação “utilizando a modelagem visual”. Já uma variabilidade de granularidade fina se caracteriza pela especialização do conteúdo de elementos de processos já existentes no núcleo da referida linha de processos. Como exemplo de variabilidade de granularidade fina, na linha de processos baseada no Scrum, temos a opção de incluir o “*planning poker*” para a definição dos itens a serem trabalhados na próxima “*sprint*”. Essa variabilidade específica implica na alteração da tarefa “reunião de planejamento da *sprint*”, incluindo o passo referente a realização do “*planning poker*” e uma breve instrução de como realizá-lo.

Segundo o critério de natureza de especificação, foi possível classificar as variabilidades como: (i) localizada (pontual), variabilidade com um interesse específico; ou (ii) transversal (*crosscutting*), variabilidade que agrupa vários interesses. Uma variabilidade localizada ou pontual é aquela que, dada a sua natureza, pode ser

classificada facilmente segundo um outro critério. Por outro lado, as variabilidades transversais por agregarem interesses diversos, geram uma dificuldade para a sua classificação de acordo com os demais critérios. Como exemplos de variabilidades transversais temos: (i) seleção de métricas específicas; e (ii) tratamento de exceções de processo. Por exemplo, a seleção de uma métrica específica, afeta simultaneamente disciplinas como: projeto, implementação e gerência de projeto. Uma métrica geralmente é responsável de fornecer parâmetros para a gerência de projeto, com base em informações das atividades de projeto e implementação, como o tempo de duração.

As variabilidades de processo de software também podem ser caracterizadas em relação ao tempo de *binding*. Segundo esse critério, a variabilidade pode ocorrer em três momentos distintos, durante: (i) a modelagem e definição do processo; (ii) a implantação do processo, em algum mecanismo que dê suporte a sua execução; e (iii) a execução do processo usando mecanismos que permitam a monitoração dessa execução. A implantação e a execução do processo de software, por exemplo, podem acontecer em um motor de workflow, conforme descrito em Aleixo et al. (Aleixo et al., 2010a) e Freire et al. (Freire et al., 2011), assim como o monitoramento pode ser realizado usando ferramentas de gerência de mudanças ou gerência do processo de software. Como exemplo de uma variabilidade que ocorre em tempo de modelagem, temos uma opção extraída da linha de processos baseada no Scrum. Trata-se da opção de escolha ou não, pela utilização de relatórios de visão geral de progresso. Caso essa opção seja selecionada são incluídos os artefatos "*product burndown*", "*sprint burndown*" e "*release burndow*", além da adição dos passos específicos relativos à atualização dos relatórios na tarefa "reunião diária". Com relação às variabilidades de tempo de implantação, poderíamos ter como exemplos a opção de utilização de um sistema de controle de versão específico e a possível integração com outras ferramentas ligadas ao desenvolvimento de software. Como exemplo de variabilidade de tempo de execução podemos ter a escolha entre alternativas de tratamento para possíveis exceções durante a execução do processo e a opção de inclusão de um novo *milestone* de projeto, como o da liberação de um novo release.

3.4 Ameaças à Validade do Estudo

As ameaças à validade do estudo foram classificadas segundo quatro diferentes grupos, os quais serão apresentados e discutidos no decorrer desta seção. No que diz

respeito à **validade de construção** (*construct validity*) identificamos as seguintes ameaças: (i) utilização de apenas três mecanismos de busca e (ii) expressividade das *strings* de busca informadas aos mecanismos escolhidos. Para as duas ameaças identificadas nessa classe, os tratamentos foram os mesmos – de antemão os três mecanismos de busca foram escolhidos (*Google Schollar*, *ACM Digital Library*, *IEEEXplore Digital Library*) com base em sua reputação no meio acadêmico e qualidade dos seus resultados. Para complementar o conjunto dos artigos resultantes dos mecanismos de busca, foram investigados manualmente os trabalhos das dez últimas edições das conferências mais importantes da área (ICSSP, PROFES e ICSE), bem como foram investigados manualmente as referências bibliográficas dos artigos até então selecionados. Ainda assim, trabalhos futuros podem complementar esta revisão sistemática, considerando a utilização de outros motores de busca, tais como: (a) *Science Direct*, (b) CiteSeerX, (c) ISI *Web of Science*, e (d) Scopus. Da mesma forma, trabalhos futuros poderão complementar essa revisão sistemática investigando manualmente outras conferências importantes da área como SPICE e ICEIS.

A expressividade das *strings* de busca utilizadas foi uma preocupação que perdurou desde a concepção até a análise dos resultados desta revisão sistemática. As *strings* de busca escolhidas, assim o foram devido a sua íntima relação com o que se desejava encontrar – abordagens de gerência de variabilidades em processos de software, e possivelmente a sua aplicação na definição de LPrSs. Após a execução das buscas e início da análise dos artigos retornados, outros termos se identificaram interessantes, tais como: "*process domain engineering*"; "*software process flexibility*"; "*software process variant*"; "*software process adaptation*"; "*software process reconfiguration*" e "*software process component*". Para complementar os resultados obtidos com esta revisão sistemática da literatura, outras podem ser realizadas incluindo alguns desses termos, e confrontando os seus resultados com os aqui apresentados.

Com relação à **validade interna** (*internal validity*) do estudo, foi identificada a ameaça da (i) seleção subjetiva dos artigos. Essa ameaça foi tratada com a definição dos critérios de seleção dos artigos e a participação de mais de um pesquisador no processo de decisão de um dado artigo; com a possibilidade de um voto de minerva de um pesquisador mais experiente. O principal critério de seleção dos artigos foi baseado na primeira questão de pesquisa – se o trabalho avaliado apresentava alguma estratégia ou técnica de gerência de variabilidade em processos de software. Um outro ponto que

contribuiu para o tratamento da referida ameaça foi a experiência da equipe que participou do estudo, visto que essa equipe, quando da realização do estudo, já acumulava a experiência de alguns anos no trabalho sob o tema de LPrS.

Com relação à **validade externa** (*external validity*) do estudo, foi identificada a ameaça do (i) estabelecimento do contexto no qual os resultados podem ser generalizados. Para o tratamento dessa ameaça, houve o cuidado da não generalização dos resultados. As conclusões do estudo serão no contexto dos trabalhos resultantes do processo de busca definido para o estudo, e servirão de indícios para o panorama geral dos trabalhos publicado sobre a gerência de variabilidades em processos de software e a definição de LPrSs.

Com relação à **confiança** (*reliability*) de que o referido estudo possa ser repetido e encontrar os mesmos resultados, foi definido e tornado público o protocolo do estudo, no qual foram registradas as informações do planejamento e execução do mesmo.

3.5 Trabalhos Relacionados

Martínez-Ruiz et al. (Martínez-Ruiz et al., 2012) apresentam uma revisão sistemática da literatura sobre o tema de requisitos e mecanismos para a adaptação de processos de software. A questão principal dessa revisão sistemática da literatura buscou identificar abordagens de customização e adaptação de processos de software, e como estas definem variabilidades em processos de software. Foram exploradas as seguintes sub-questões: (i) “quais elementos do modelo de processo de software são utilizados quando uma adaptação é considerada?”; (ii) “quais os tipos de mecanismos de customização são utilizados nos modelos de processo existentes?”; (iii) “quais as notações para modelagem de processos de software são utilizadas para dar suporte à customização de modelos de processo?”; e (iv) “como os processos são customizados para atender as características de uma organização ou projeto específico?”.

A análise dos resultados de tal revisão sistemática permitiu as seguintes conclusões: (i) 32 estudos primários foram encontrados; (ii) permitiu a definição de alguns “requisitos” que as notações voltadas para a customização de processos de software precisam atender; (iii) identificado um interesse crescente em customização de processos de software, principalmente nos últimos anos; (iv) identificado um número crescente de aplicação da customização de processos de software na indústria; (v) as

variabilidades em processos de software ocorrem mais frequentemente em atividades, seguido de artefatos e papéis; (vi) no que diz respeito às variabilidades nos relacionamentos entre elementos, são mais frequentes as variações nos fluxos de controle e de artefatos; (vii) poucos estudos propõem a definição de notações especiais para a representação das variabilidades em processos de software; (viii) a maioria dos estudos propôs ferramentas para auxiliar na customização dos processos de software; e por fim, (ix) as ações visando a manutenção da consistência dos processos deve acontecer em paralelo com as ações de customização.

Rocha & Fantinato (Rocha & Fantinato, 2013) apresentam uma revisão sistemática da literatura sobre o tema do uso das técnicas de linhas de produto de software (SPL) para o gerenciamento de processos de negócio (BPM). Para a orientação desse estudo foram definidas quatro questões de pesquisa, as quais são: (i) “quais fases do ciclo de vida BPM são atendidas com a aplicação da abordagem SPL para este domínio?”; (ii) “quais os conceitos de SPL que são aplicados ao domínio de BPM?”; (iii) “as abordagens de SPL para BPM são todas baseadas em SOA?”; e (iv) “quais as limitações das abordagens de SPL para BPM?”. Após a análise dos trabalhos que retornaram das buscas, 63 trabalhos satisfizeram os critérios de seleção. Desses, apenas 15 artigos trataram os aspectos específicos do contexto avaliado. Foi identificado que a estratégia de linhas de produtos de software apenas atendem parcialmente o ciclo de vida dos processos de negócio, visto que a última fase dos processos de negócio não é tratada pelas abordagens identificadas. Os resultados indicaram que as abordagens de linhas de produtos de software para o contexto de gerenciamento de processos de negócio ainda estão em um estágio inicial e em processo de amadurecimento.

3.6 Conclusões

Na realização desta revisão sistemática da literatura a respeito de abordagens de gerência de variabilidades em processos de software, foram obtidos os seguintes resultados: (i) 404 (quatrocentos e quatro) artigos retornaram das buscas realizadas; (ii) destes, 40 (quarenta) foram selecionados; (iii) 216 (duzentos e dezesseis) foram excluídos por se tratarem de réplicas de trabalhos já listados; (iv) 5 (cinco) artigos retornados estavam escritos em outra língua que não o inglês; (v) 23 (vinte e três) artigos não se enquadravam nos formatos previamente especificados; (vi) 120 (cento e vinte) artigos foram excluídos por não estarem alinhados com o foco definido para o

estudo; e (vii) ao final, foram identificadas 19 (dezenove) abordagens de gerência de variabilidades em processos de software.

Os resultados obtidos apresentam um indício do panorama do nicho de pesquisa no qual está inserida esta tese – permitindo que sejam tiradas algumas conclusões importantes, quais sejam: (i) a aplicação das técnicas de linhas de produtos de software no contexto de processos de software vem sendo explorada por vários trabalhos de pesquisa; (ii) já existem algumas abordagens propostas para a gerência de variabilidades em LPrSs; (iii) boa parte das abordagens identificadas ainda se encontra em fase de amadurecimento, visto que para as mesmas só foram apresentados exemplos genéricos e simplificados da sua aplicação, e sem menção a algum suporte ferramental; e (iv) há uma carência de estudos para todas as abordagens identificadas, bem como de estudos comparativos, voltados a evidenciar em que cenários ou para quais tipos de variabilidades uma abordagem é melhor do que outra. Tais evidências reforçam a importância da realização de estudos mais aprofundados de aplicação de tais abordagens, além de estudos comparativos identificando pontos fortes e pontos fracos de cada abordagem, bem como os benefícios e limitações da sua aplicação.

Cabe observar que durante o processo de seleção dos estudos primários foram identificados alguns trabalhos no contexto de processos genéricos, tais como: (i) processos de negócio e (ii) sistemas orientados à processos. Embora abordando temas interessantes e relacionados aos trabalhos no contexto de processos de software, foram descartados em função do escopo definido para esta versão da revisão sistemática. Uma análise futura desses trabalhos poderá trazer contribuições importantes para o contexto da gerência de variabilidades em processos de software; tais como a adequação de técnicas, métodos ou ferramentas.

Como resultado marginal da revisão sistemática temos que a mesma servirá de base para uma classificação preliminar de variabilidades de processos de software. A classificação dos tipos de variabilidades em processos de software é importante pois pode: (i) ajudar na modelagem de linhas de processo de software, ajudando na localização das variabilidades, ilustrando onde e como elas podem acontecer; (ii) na definição de estratégias específicas de modularização para cada tipo específico de variabilidade; e (iii) servir de base para a avaliação de abordagens de gerência de variabilidades em processos de software.

4 Uma Abordagem Anotativa para a Gerência de Variabilidades em Linhas de Processos de Software

Este capítulo apresenta uma proposta de abordagem para a gerência de variabilidades em LPrSs. A definição dessa abordagem visou aplicar as técnicas e princípios da estratégia de linha de produtos de software no contexto de processos de software. O objetivo principal dessa iniciativa foi promover o reúso sistemático das definições de processos de software, visando a derivação de instâncias customizadas de processos de software, que atendam às necessidades de projetos de desenvolvimento de software específicos. A abordagem em questão se baseou nos princípios da abordagem anotativa (Pohl et al., 2005), onde os ativos que compõem uma linha de produto são decorados com informações que os associam à determinadas *features*. Também é apresentado neste capítulo um estudo exploratório visando validar a referida proposta de abordagem, por meio da: (i) implementação do suporte ferramental necessário e da (ii) modelagem de uma LPrS exemplo com tal ferramental.

A organização deste capítulo obedece a seguinte estrutura: a Seção 4.1 apresenta uma introdução ao capítulo; a Seção 4.2 descreve a proposta de abordagem para a gerência de variabilidades em LPrSs; a Seção 4.3 apresenta o estudo exploratório realizado para validar a proposta de abordagem em questão; a Seção 4.4 detalha a realização do estudo exploratório, além de analisar e discutir a evolução da proposta de abordagem; a Seção 4.5 apresenta a análise dos resultados alcançados pelo estudo exploratório; a Seção 4.6 apresenta algumas discussões e limitações do estudo; e por fim, a Seção 4.7 apresenta as conclusões do estudo exploratório.

4.1 Introdução

A análise das abordagens identificadas na revisão sistemática da literatura, apresentada no Capítulo 3 possibilitou caracterizar três momentos distintos em que pode ocorrer a gerência de variabilidades: (i) especificação da LPrS; (ii) implantação de uma instância de processo em um ambiente de suporte à execução; e (iii) execução de uma instância do processo em um ambiente que apoie a sua execução. Cada um dos momentos em que pode ocorrer a gerência de variabilidades possui um conjunto de características e necessidades específicas. O primeiro momento é o da especificação, ou modelagem, da LPrS. Nesse momento, são tratadas, principalmente, variabilidades que

afetam a estrutura da família de processos sendo modelada, como por exemplo: (i) quais papéis podem ou devem fazer parte de processos membros da família sendo modelada; (ii) quais atividades podem ou devem fazer parte de processos membros da família sendo modelada; e (iii) quais disciplinas podem ou devem fazer parte de processos membros da família sendo modelada. Após a modelagem da LPrS, com a devida gerência das variabilidades; esta pode ser utilizada na derivação de instâncias dde processo (customizadas), os quais são considerados membros da LPrS.

O segundo momento é o da implantação de uma instância de processo derivada a partir da LPrS em um ambiente que ofereça suporte à execução do mesmo, como, por exemplo: um sistema ou motor de *workflow*. Nesse momento, são tratadas as variabilidades relativas às especificidades do processo que está sendo preparado para ser executado, como, por exemplo: (i) quantidades de iterações previstas para cada fase, (ii) definição dos responsáveis pelos papéis do processo, e (iii) estrutura de repositório para o armazenamento dos artefatos produzidos pelo processo.

O terceiro, e último, momento é o da execução de um processo implantado em um ambiente que dê suporte para tal. Nesse momento, são tratadas, principalmente, variabilidades relativas a condições excepcionais da execução do processo, como, por exemplo: (i) alteração na quantidade de iterações de uma dada fase, com base em informações das reuniões de planejamento; (ii) opção de retroceder no fluxo de atividades, para contemplar problemas identificados em atividades específicas; e (iii) opção por não executar uma determinada tarefa, em função de características específicas do sistema de software sendo desenvolvido e/ou da equipe de desenvolvimento.

Portanto, temos que as variabilidades em LPrSs podem ser gerenciadas em três momentos distintos: (i) definição, (ii) implantação e (iii) execução. Foi definido como escopo para esta proposta de abordagem de gerência de variabilidades em LPrSs o momento da definição da mesma.

A revisão sistemática da literatura, citada anteriormente, identificou que até a realização da mesma não existiam abordagens segundo a estratégia anotativa visando a gerência de variabilidades de LPrS. Também não foram identificados trabalhos apresentando questionamentos e avaliações sobre a possibilidade da aplicação de tal abordagem no contexto de LPrS. Por outro lado, temos que a abordagem anotativa

demonstra estar consolidada no contexto de linhas de produtos de software, embasando ferramentas, tais como: CIDE (Kästner et al., 2010) e GenArch (Cirilo et al., 2008). Neste cenário, vislumbrou-se a possibilidade da aplicação da estratégia anotativa no contexto de LPrS.

O referido cenário, e a oportunidade identificada pela revisão sistemática da literatura, motivaram a concepção de uma abordagem anotativa para a gerência de variabilidades em LPrSs. Também foi idealizada a realização de um estudo exploratório visando atestar a viabilidade da referida abordagem. A realização do estudo buscou evidenciar que a estratégia anotativa poderia ser aplicada à gerência de variabilidades em LPrS. Segundo o escopo definido para a proposta de abordagem, o gerenciamento de variabilidades ficou restrito ao momento de modelagem da LPrS. Contudo, o estudo exploratório buscou evidências da viabilidade da abordagem anotativa na: (i) modelagem de uma LPrS, (ii) derivação de uma instância de processo específica e (iii) implantação dessa instância em um ambiente de suporte à execução.

4.2 Abordagem para a Gerência de Variabilidades em LPrSs

Após a identificação e a análise de alguns trabalhos relacionados (Rombach, 2005) (Washizaki, 2006a) (Simidchieva et al., 2007) (Armbrust et al., 2009) (Martínez-Ruiz et al., 2009a) (Ternité, 2009), foi esboçada a abordagem proposta. A aplicação da abordagem na modelagem de uma LPrS exemplo foi realizada em estágios, foram eles: (i) identificação das similaridades e variabilidades em uma família de processos; (ii) decoração dos elementos de processos associados às *features* da LPrS; (iii) derivação automática de uma instância de processo customizado, de acordo com o conjunto das *features* selecionadas; (iv) geração do modelo de *workflow* correspondente à instância de processo; e (v) implantação e execução do modelo de *workflow* correspondente.

Dentre as razões que motivaram a escolha da abordagem anotativa, podem ser destacadas: (i) o fato de ser uma abordagem em evidência no que diz respeito a engenharia de linhas de produtos de software; (ii) não terem sido identificadas outras propostas de abordagens anotativas para a gerência de LPrSs; (iii) por ser uma estratégia que independe da forma de representação do processo de software utilizado; e (iv) o conhecimento da abordagem anotativa pela equipe envolvida na elaboração da proposta. Cabe observar que parte da equipe participou da concepção da ferramenta GenArch,

voltada à derivação de produtos, segundo a abordagem anotativa. A Figura 11 apresenta os estágios de aplicação dessa proposta de abordagem anotativa para a gerência de variabilidades em LPrSs. Ela ilustra os 5 (cinco) estágios de aplicação da abordagem proposta, os quais serão explicados em mais detalhes nos próximos parágrafos.

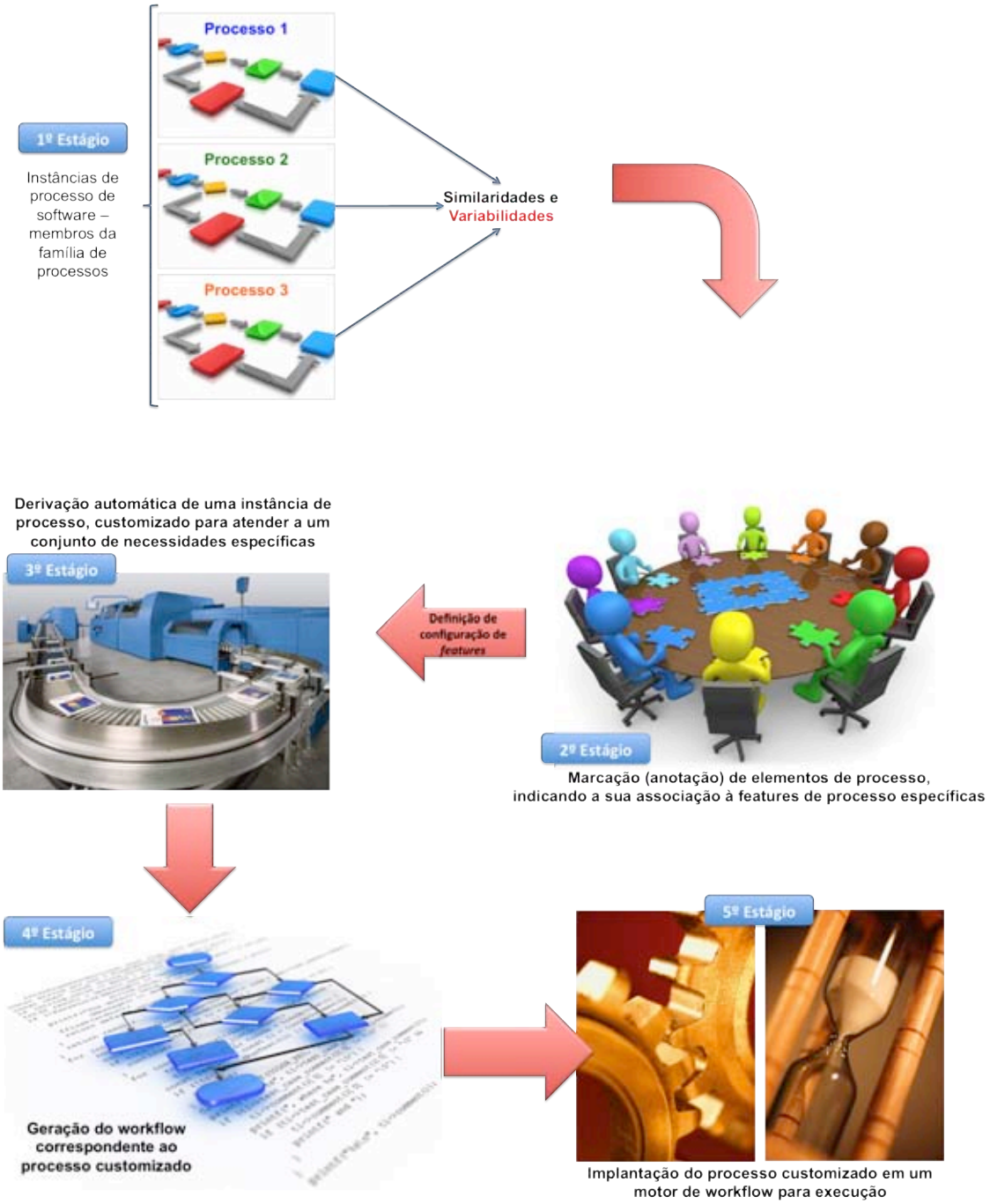


Figura 11. Estágios de aplicação da abordagem proposta

No primeiro estágio de aplicação da abordagem, acontece a escolha dos processos de software existentes que servirão de base para a definição da LPrS, como membros da família de processos de software desejada. Os processos selecionados são

analisados visando identificar as similaridades e as variabilidades entre os mesmos. No momento da identificação das variabilidades, são mapeados também os elementos de processo associados a tais variabilidades.

Após a identificação das variabilidades, tem início o segundo estágio da aplicação da abordagem, onde os elementos de processo variáveis são anotados, indicando que estão associados à uma dada *feature* de processo. Cada anotação contém o nome da *feature* à qual o elemento de processo está associado, bem como algumas propriedades da *feature*, como o seu tipo e informações hierárquicas. As anotações devem ser realizadas nos elementos de processo, mas não devem alterar a sua estrutura interna. A especificação do processo de software em questão não deve ser alterada, nem se tornar inválida após a inserção das anotações. O segundo estágio de aplicação da abordagem – anotação dos elementos de processo com informações das *features* correspondentes – pode ser denominado de modelagem da LPrS.

Após a modelagem da LPrS, tem início o terceiro estágio da aplicação da abordagem com a identificação das necessidades específicas de um dado projeto de desenvolvimento, para o qual se deseja derivar um processo da LPrS. A definição de uma nova configuração do modelo de *features*, a qual permite que as *features* desejadas sejam selecionadas. A configuração de *features* produzida nessa atividade é, então, utilizada como base para a derivação automática de uma instância da LPrS. O processo de derivação automática é realizado por meio da manipulação dos modelos, representando: (i) o processo, (ii) as *features* e (iii) a associação entre as *features* e os elementos do processo. O resultado final – a instância da LPrS – é um processo de software customizado, que atende as necessidades do projeto de desenvolvimento alvo.

O quarto estágio de aplicação da abordagem é responsável pela geração de um modelo de fluxo de atividades (*workflow*) referente à instância de processo que foi derivada no estágio anterior. A geração do modelo de *workflow* deve ser feita por meio de uma transformação entre modelos. Nessa transformação entre modelos ocorre um mapeamento dos elementos que definem o fluxo de atividades do processo, para elementos correspondentes em um *workflow*. O resultado final dessa transformação é um modelo de *workflow* que espelha o fluxo de atividades definido para a instância de processo, utilizada como entrada.

No quinto, e último, estágio de aplicação da abordagem, o modelo de *workflow* gerado no estágio anterior é utilizado como entrada para a geração de todos os artefatos necessários para a implantação do mesmo em um motor de *workflow*. O passo seguinte, é a implantação do referido *workflow* em um motor que dê suporte à execução do mesmo. Com o *workflow* devidamente implantado em um motor de execução, pode ser iniciada a execução de uma instância do mesmo. O objetivo é que o referido *workflow* auxilie e oriente os participantes do processo na realização do fluxo de atividades definido para o mesmo.

4.3 Estudo Exploratório sobre a Viabilidade da Abordagem Proposta

Esta Seção apresenta a definição de um estudo exploratório visando atestar a viabilidade da abordagem proposta. São apresentadas: as questões de pesquisa definidas para o estudo (Seção 4.3.1); as fases propostas para a realização do mesmo (Seção 4.3.2); bem como, os procedimentos de análise dos resultados do estudo (Seção 4.3.3).

4.3.1 Questões de Pesquisa

Visando orientar a execução do estudo exploratório foram definidas duas questões de pesquisa:

- **Questão de pesquisa 1**: Qual a viabilidade do uso dos mecanismos anotativos propostos para a gerência de variabilidades no contexto de LPrS?
- **Questão de pesquisa 2**: Quais os benefícios e limitações de tais mecanismos quando usados no contexto de LPrS?

4.3.2 Etapas do Estudo e Procedimentos de Avaliação

No intuito de responder às questões de pesquisa definidas para o estudo exploratório, foram definidas quatro etapas: (1ª) implementação e aplicação de uma abordagem anotativa para LPrSs; (2ª) análise do resultado da implementação e das limitações da versão implementada da abordagem; (3ª) definição de um conjunto de propostas de melhoria da abordagem; e (4ª) discussão sobre os resultados e limitações do estudo exploratório. A análise dos resultados do estudo exploratório de aplicação da abordagem proposta abrangeu três aspectos: (i) modelagem de variabilidades em

processos de software, (ii) derivação automática de instâncias de processos, e (iii) implantação e execução do processo gerado em um motor de *workflow*, permitindo a sua execução e monitoramento.

Na primeira etapa do estudo exploratório, aconteceu o desenvolvimento do suporte ferramental para a abordagem proposta. O suporte ferramental para a abordagem proposta baseou-se na adaptação de uma ferramenta anotativa de derivação de produtos, denominada GenArch (Cirilo et al., 2008). Foi utilizada como base a ferramenta GenArch para possibilitar a gerência sistemática de variabilidades em LPrSs. Na segunda etapa do estudo exploratório, após o desenvolvimento do suporte ferramental, a abordagem proposta foi utilizada na definição de uma LPrS baseada no OpenUP (EPF, 2011). O OpenUP é um processo de software inspirado no processo unificado – UP (Scott, 2003), disponibilizado pela iniciativa EPF, no formato *open source*, para ser utilizado ou customizado pelo público em geral.

Na segunda etapa do estudo exploratório, foram analisados os resultados da aplicação da abordagem anotativa desenvolvida na primeira etapa, com base em critérios de avaliação estabelecidos. Para atestar a viabilidade da abordagem proposta e do suporte ferramental desenvolvido foram considerados os seguintes critérios: (i) capacidade de modelagem de *features* de processos de software e associação de elementos de processo a tais *features* e (ii) possibilidade de derivação de instâncias da LPrS a partir da seleção das *features* desejadas. Também foram identificadas e discutidas as limitações identificadas da versão implementada para a abordagem.

Na terceira etapa do estudo exploratório, foram apresentadas propostas de melhorias para a abordagem proposta e para o suporte ferramental desenvolvido, visando lidar com as limitações identificadas. Também foram discutidos os impactos de tais melhorias sobre cada um dos estágios propostos para a abordagem. Na quarta, e última, etapa do estudo exploratório foi realizada uma análise dos resultados alcançados pelo estudo, em contraposição às questões de pesquisa estabelecidas. Os detalhes relativos à realização das etapas descritas são apresentados do decorrer deste capítulo.

4.4 Desenvolvimento do Estudo Exploratório

Esta Seção detalha a execução das etapas propostas para o estudo. A Seção 4.2.1 apresenta a realização da primeira etapa do estudo exploratório, na qual foi

implementada e aplicada uma abordagem anotativa para gerência de variabilidades em LPrSs. Ao longo da apresentação dos estágios propostos para a aplicação da abordagem, também é ilustrado um exemplo de aplicação da mesma na modelagem de uma LPrS baseada no OpenUP. A Seção 4.2.2 apresenta a segunda etapa deste estudo exploratório, na qual ocorreu a análise e discussão sobre as limitações identificadas na versão inicial da abordagem. A identificação das limitações ocorreu através da análise da aplicação da abordagem na modelagem de uma LPrS baseada no OpenUP. E, por fim, a Seção 4.2.3 descreve a terceira etapa deste estudo exploratório, na qual é definido um conjunto de propostas de melhorias para a versão implementada da abordagem e como tais propostas afetam os estágios previstos para a aplicação da abordagem. A quarta e última etapa que discute os resultados do desenvolvimento deste estudo exploratório, é apresentada na Seção 4.3.

4.4.1 Implementação e Aplicação de uma Versão Inicial da Abordagem

Durante a realização do estudo exploratório foi implementado um protótipo do suporte ferramental necessário para a abordagem e, em seguida, a abordagem proposta foi aplicada na definição de uma LPrS. É importante ressaltar que a abordagem visou apenas o gerenciamento de variabilidades considerando a modelagem da LPrS. A Figura 12 ilustra uma visão geral do suporte ferramental desenvolvido para a abordagem, e das tecnologias utilizadas. Os rótulos presentes na Figura 12 ilustram os estágios definidos para a abordagem.

Na implementação do suporte ferramental da abordagem foram utilizadas e adaptadas ferramentas existentes. Para a especificação e modelagem do processo foi adotada a plataforma Eclipse *Process Framework* (EPF) (Eclipse Foundation, 2012). Mais especificamente foi utilizada a ferramenta EPF *Composer*, que gera como resultado um modelo segundo o meta-modelo UMA – *Unified Method Architecture*. O meta-modelo UMA define uma notação para a representação de processos de software, compostos por (i) conteúdo de métodos e (ii) processos (Haumer, 2007).

A gerência de variabilidades foi realizada com o auxílio de mecanismos e técnicas oferecidos pela ferramenta de derivação de produtos, denominada GenArch (Cirilo et al., 2008). Tal ferramenta foi adaptada para permitir a criação de relações de

mapeamento entre *features* existentes em um modelo de *features*, e elementos de processos especificados em um modelo EPF. A ferramenta GenArch também foi utilizada para habilitar a seleção das *features* relevantes em face de demandas específicas, permitindo assim a derivação automática de especificações de processo.

Figura 12. Visão geral do suporte ferramental para a abordagem

Para avaliar os impactos da gerência de variabilidades em processos de software durante os momentos de implantação e execução, a abordagem propõe a extração de um modelo de *workflow* a partir do modelo UMA. Para tal, é aplicada uma transformação de modelos no modelo UMA resultante da derivação automática, gerando como resultado um modelo de *workflow* correspondente. O modelo de *workflow* gerado sofre então um processo de preparação para que o mesmo possa ser implantado em um motor de *workflow*. Por fim, o *workflow* é implantado em um motor de *workflow* específico e a sua execução é acompanhada por meio do mesmo.

Cada um dos estágios que compreendem a abordagem será explicado em maiores detalhes nas próximas seções, ilustrando a sua implementação e a sua aplicação para a gerência de variabilidades.

4.4.1.1 Estágio 1: Identificação e Modelagem de Similaridades e Variabilidades

De forma similar ao desenvolvimento de linhas de produtos de software, o primeiro passo para a definição de uma LPrS é o estudo das similaridades e variabilidades entre os processos que fazem parte de uma mesma família. A abordagem proposta utilizou a forma de representação de processos de desenvolvimento de software proposta pelo EPF. Dentre os processos disponibilizados pelo EPF, o OpenUP foi selecionado como sendo a família de processos a ser estudada, dado que o mesmo pode ser customizado de diversas formas para atender às necessidades específicas de projetos reais de desenvolvimento.

O EPF *Composer* usa uma abordagem baseada em formulários para a definição e apresentação dos elementos de processo, tais como, papéis, tarefas e artefatos. Com o EPF *Composer*, um engenheiro de processo pode definir um novo processo, reusando elementos de outros processos previamente definidos na ferramenta. Ao finalizar a edição de um processo, é possível gerar uma publicação do mesmo, na forma de um site Web. O EPF *Composer* pode ser usado também para exportar a definição do processo, escolhendo-se um local na estrutura de diretórios da máquina. A definição de um processo usando o EPF é fisicamente constituída por uma estrutura de diretórios, contendo arquivos XMI que obedecem o meta-modelo UMA.

Embora o EPF forneça algum apoio para especificar e definir processos de software, ele não permite a representação e a personalização automática de linhas ou famílias de processos de software. Ferramentas de derivação de produtos podem, então, ser usadas para permitir a seleção das *features* relevantes de uma família ou linha de processo existente, permitindo assim a derivação automática de especificações personalizadas do processo de software, abordando cenários e projetos específicos.

Neste contexto, o OpenUP pode ser visto como uma linha de processos que permite a customização de um novo processo de acordo com as demandas específicas do projeto a ser desenvolvido. Porém, tratar as variabilidades e similaridades desse framework de processos diretamente no EPF não é uma tarefa trivial, pois tais características não estão logicamente organizadas. O processo OpenUP explicita que algumas atividades ou passos do processo são opcionais durante a sua execução, entretanto, ele não define quais elementos dos processos (atividades, passos, artefatos, guias, etc.) são variabilidades do ponto de vista da linha de processos, podendo estar ou não presente durante a sua instanciação para um projeto específico. Assim, para modelar o OpenUP como uma linha de processos, o primeiro passo é definir claramente quais são as suas *features* similares e variáveis.

Para o estudo exploratório de aplicação da abordagem, tomou-se como base um pequeno conjunto de projetos de pesquisa e desenvolvimento reais que seguiam o OpenUP. Os referidos projetos, estavam ligados ao Núcleo de Desenvolvimento de Software da Diretoria Acadêmica de Gestão e Tecnologia da Informação do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte – NUDES / DIATINF / CNAT / IFRN (CNPq, 2012). Cada um dos projetos se propôs a executar customizações do OpenUP. Além disso, foi também considerada a experiência profissional dos envolvidos na definição de processos e metodologias de desenvolvimento de software baseadas no RUP.

A seleção de projetos de pesquisa e desenvolvimento de software reais, que fazem uso de customizações do OpenUP, permitiu a análise de similaridades e variabilidades nessa família de processos. A estrutura de divisão de trabalho do OpenUP (*work breakdown structure*) serviu de base para a modelagem das similaridades e variabilidades. Como resultado dessa análise foi produzida uma matriz que explicita quais atividades, tarefas, artefatos de entrada e saída e passos relacionadas às tarefas do

OpenUP representam similaridades e variabilidades da linha de processo que o define. No total, foram identificadas 586 *features*, delas: 273 *features* mandatórias, 239 *features* opcionais e 74 *features* alternativas.

A Figura 13 apresenta uma visão parcial da matriz de similaridades e variabilidades obtida como resultado desse processo de análise de domínio (Kang et al., 1990) da linha de processo. A matriz de similaridades completa, bem como outras informações sobre a execução do estudo estão disponíveis em um site web mantido pelos pesquisadores responsáveis (Aleixo, 2013). Na análise de domínio, as tarefas que foram executadas em todos os projetos investigados foram classificadas como mandatórias, e aquelas que não foram executadas em pelo menos um dos projetos estudados, foram classificadas como opcionais. Pode-se observar, por exemplo, diversas tarefas relacionadas ao detalhamento dos cenários de um caso de uso (*Detail Use-Case Scenarios*), criação de casos testes (*Create Test Cases*) e esboço da arquitetura (*Outline the Architecture*) foram classificadas como opcionais para a fase de Concepção. Isso ocorreu porque em pelo menos um dos projetos a referida atividade não foi executada. Vale ressaltar que a especificação do OpenUP dentro do EPF não torna explícita a presença de tais similaridades e variabilidades, dificultando assim sua customização automática para novos projetos.

Breakdown Element	Index	Pred.	Feature	Arquivo em que acontece a referência
Inception Phase	1			/EPF_Practices/process.openup.base/deliveryprocesses/openup_lifecycle/model.xmi
Inception Iteration [1..n]	2			/EPF_Practices/process.openup.base/deliveryprocesses/openup_lifecycle/model.xmi
1. Initiate Project	3		mandatória	/EPF_Practices/process.openup.base/capabilitypatterns/inception_phase_iteration/model.xmi
1.1 Develop Technical Vision	4		mandatória	/EPF_Practices/process.openup.base/capabilitypatterns/initiate_project/model.xmi
1.2 Plan Project	5		opcional	/EPF_Practices/process.openup.base/capabilitypatterns/initiate_project/model.xmi
1.2.1 [template] Word template			alternativa	
1.2.2 [template] MS-Project template			alternativa	
2. Plan and Manage Iteration	6		mandatória	/EPF_Practices/process.openup.base/capabilitypatterns/inception_phase_iteration/model.xmi
2.1 Plan Iteration	7		mandatória	/EPF_Practices/process.openup.base/capabilitypatterns/plan_manage_iteration/model.xmi
2.1.1 [template] Full Risk List	-		alternativa	
2.1.2 [template] Top 10 Risk List	-		alternativa	
2.2 Manage Iteration	8		mandatória	/EPF_Practices/process.openup.base/capabilitypatterns/plan_manage_iteration/model.xmi
2.3 Assess Results	9		mandatória	/EPF_Practices/process.openup.base/capabilitypatterns/plan_manage_iteration/model.xmi
3. Identify and Refine Requirements	10	3	mandatória	/EPF_Practices/process.openup.base/capabilitypatterns/inception_phase_iteration/model.xmi
3.1 Identify and Outline Requirements	11		mandatória	/EPF_Practices/process.openup.base/capabilitypatterns/identify_and_refine_requirements/model.xmi
3.2 Detail Use-Case Scenarios	12		opcional	/EPF_Practices/process.openup.base/capabilitypatterns/identify_and_refine_requirements/model.xmi
3.3 Detail System-Wide Requirements	13		opcional	/EPF_Practices/process.openup.base/capabilitypatterns/identify_and_refine_requirements/model.xmi
3.4 Create Test Cases	14		opcional	/EPF_Practices/process.openup.base/capabilitypatterns/identify_and_refine_requirements/model.xmi
4. Agree on Technical Approach	15	3	opcional	/EPF_Practices/process.openup.base/capabilitypatterns/inception_phase_iteration/model.xmi
4.1 Outline the Architecture	16		opcional	/EPF_Practices/process.openup.base/capabilitypatterns/develop_architecture/model.xmi
Lifecycle Objectives Milestone	17	2		/EPF_Practices/process.openup.base/deliveryprocesses/openup_lifecycle/model.xmi

Figura 13. Fragmento da matriz de similaridades e variabilidades

No que se refere às atividades do OpenUP, a seguinte estratégia foi usada para realizar a sua modelagem: (i) as atividades que tiveram pelo menos uma das suas tarefas classificada como mandatória, foram classificadas como mandatórias também; e (ii) no caso onde todas as tarefas da atividade foram classificadas como opcionais, a atividade

foi também classificada como opcional. A Figura 13 exemplifica tal caso, por exemplo, mostrando que a atividade "*Identify and Refine Requirements*" é mandatória porque possui pelo menos uma tarefa obrigatória (*Identify and Outline Requirements*). Já a atividade "*Agree on Technical Approach*" foi classificada como opcional já que todas as tarefas agregadas são opcionais.

Durante o estudo, também foram identificados elementos do processo que representavam possíveis *features* alternativas, como, por exemplo, vários tipos diferentes de modelos para um dado artefato. A Figura 13, também ilustra que os *templates* de artefatos podem representar elementos de processo alternativos para a LPrS, como, por exemplo, o documento de planejamento do projeto. Segundo esse exemplo, o documento de planejamento do projeto tanto pode ser baseado em um *template* do *Word*, como em um *template* do *MS Project*. Outro exemplo de *features* alternativas encontradas foram os próprios guias de técnicas ou tecnologias, os quais são muitas vezes oferecidos pelos processos para apoiar a realização de alguma atividade.

Durante o estudo das variabilidades e similaridades nas instâncias selecionadas do OpenUP, também foi feita a identificação dos trechos do modelo de processo (segundo o meta-modelo UMA) os quais eram correspondentes às atividades ou tarefas consideradas variantes. Tal informação foi bastante importante para a fase subsequente de gerência de variabilidades da linha de processos detalhadas na próxima seção.

4.4.1.2 Estágio 2: Gerência Automatizada de Variabilidades

Após a identificação e modelagem de similaridades e variabilidades da linha de processo, a atividade seguinte trata de prover a gerência automática de variabilidades, pro meio da especificação de modelos que podem ser manipulados por uma ferramenta de derivação. A concretização desse estágio habilita os engenheiros de processo a poderem gerenciar automaticamente as variabilidades do processo. Para a concretização desse estágio da abordagem, foi selecionada a ferramenta de derivação de produtos GenArch. Esta seção detalha a adaptação da ferramenta GenArch para a utilização no contexto de processos de desenvolvimento de software, além do processo de utilização da mesma para atingir os objetivos desse estágio da abordagem.

Anotando Modelos de Processo com Variações. A ferramenta GenArch foi escolhida nessa etapa, por duas razões principais: (i) ela foi desenvolvida usando

tecnologias compatíveis com a especificação de processos do framework EPF, tais como, o *Eclipse Modeling Framework* (EMF) (Steinberg et al., 2008) e o openArchitectureWare (oAW) (openArchitectureWare.org, 2012); e (ii) ela oferece facilidades de extensão que facilitam a adaptação da ferramenta para manipular os artefatos de especificação de processos oferecidos pelo EPF. O GenArch foi inicialmente desenvolvido para trabalhar com linhas de produtos de software implementadas usando as linguagens Java e AspectJ. Mais recentemente, a ferramenta foi também adaptada para trabalhar com frameworks de infraestrutura (Cirilo et al., 2011).

A ferramenta GenArch oferece suporte para a derivação automática de produtos durante a engenharia de aplicação, a partir dos artefatos de implementação produzidos na engenharia de domínio para arquiteturas de LPSs. A ferramenta permite a criação automática de três modelos (característica, configuração e arquitetura) que auxiliam nesse processo de derivação, a partir da análise sintática de anotações Java. Tais anotações são inseridas dentro de artefatos que implementam a arquitetura da LPS, e indicam que variabilidades cada um deles implementa. Para permitir o uso da ferramenta GenArch na gerência de variabilidades de uma linha de processos, foi necessário adaptá-la com a criação de novas funcionalidades que permitissem a manipulação de artefatos de uma especificação de processo geradas no EPF *Composer*. Uma especificação de processo no EPF é representada por um modelo UMA. O modelo UMA é armazenado em uma estrutura de diretórios contendo um conjunto de arquivos XMI. Cada arquivo XMI, que define um modelo, representa uma parte bem definida (atividade, tarefa, artefato) do processo. Foi necessário estender a ferramenta para permitir a leitura e manipulação dessa estrutura de diretórios e arquivos XMI. No lugar das anotações Java, da versão convencional, foram usados comentários XML contendo as mesmas informações contidas nas anotações. Tais anotações, na forma de comentários XML, podem então ser lidos e processados para automaticamente produzir os modelos usados na derivação automática.

Assim, após a adaptação da ferramenta GenArch para permitir a gerência de variabilidades em linhas de processos especificadas no EPF, a sua utilização em tal contexto, passou a obedecer os seguintes passos:

(i) *exportação do processo que se deseja gerenciar variabilidades, a partir da ferramenta EPF Composer* – dado que estamos interessados em gerenciar variabilidades em um processo EPF, esse passo é importante para permitir a exportação dos artefatos que representam o processo de forma a promover a gerência de suas variabilidades. A ferramenta permite essa exportação na forma de um *Method Plug-in*, que é um container de conteúdos de métodos e do processo, especificados como uma série de arquivos XMI;

(ii) *inclusão dos comentários/anotações XML dentro dos elementos de processos (arquivos XMI) para indicar explicitamente a que variabilidades cada um deles está relacionado*. A utilização de comentários foi motivada pelo fato de serem ignorados na interpretação da especificação do processo. Este passo foi realizado com o auxílio de uma matriz de variabilidades, gerada a partir da análise das customizações necessárias ao processo em questão, para atender às necessidades dos diferentes tipos de projetos. Esse procedimento de adicionar anotações nos arquivos XMI para indicar cada variabilidade foi relativamente complexa, pois exigiu um conhecimento razoável da estrutura física de armazenamento dos arquivos relativos a especificação de processo no UMA. Por fim, os diferentes arquivos XMI relativos a diferentes elementos do processo foram devidamente marcados com comentários XML. A Figura 14, por exemplo, ilustra um arquivo XMI com um comentário que indica que tal artefato representa a *feature* "*initiate_project*", cujo *feature*-pai é a "*activities*", sendo a mesma obrigatória.

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- @Feature(name=initiate_project, parent=activities, type=mandatory) -->

<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:org.eclipse.epf.uma="http://www.eclipse.org/epf/uma/1.0.5/uma.ecore"
       xmlns:org.eclipse.epf.uma.resourcemanager=
       "http:///org/eclipse/epf/uma/resourcemanager.ecore"
       xmlns:rmc="http://www.ibm.com/rmc" rmc:version="7.5.0"
       xmlns:epf="http://www.eclipse.org/epf"
       epf:version="1.5.0">
  <org.eclipse.epf.uma.resourcemanager:ResourceManager xmi:id="_eZJ_INOEEdyqlogshP814g"
      guid="_eZJ_INOEEdyqlogshP814g">
    <resourceDescriptors xmi:id="_eZJ_IdOEEdyqlogshP814g" id="-kpXvvHptrggypecQEamfhw"
        uri="content.xmi"/>
  </org.eclipse.epf.uma.resourcemanager:ResourceManager>
  <org.eclipse.epf.uma:ProcessComponent xmi:id="_eWxZgNOEEdyqlogshP814g"
       name="initiate_project"
       guid="_eWxZgNOEEdyqlogshP814g">
```

Figura 14. Anotação em um arquivo XMI

(iii) importação dos artefatos que especificam a linha de processos na forma de um projeto GenArch no Eclipse – este passo consistiu na importação dos arquivos XMI

anotados que representam o processo na forma de um projeto EPF dentro do Eclipse, bem como na análise e processamento de seus respectivos comentários (anotações), de forma a promover a geração automática dos modelos de derivação do GenArch. Os modelos gerados para o estudo de caso OpenUP são detalhados na próxima seção.

Criação Automática de Modelos de Derivação. A ferramenta GenArch promove a automação do processo de derivação por meio da modelagem e especificação de três modelos: (i) modelo de características (*features*) – que indicam as características comuns (similaridades) e variáveis (variabilidades) existentes na linha de processo/produto, além de permitir a definição de restrições (*constraints*) entre os mesmos; (ii) modelo de arquitetura – que representa visualmente os artefatos usados na estruturação da linha de processos/produtos; e (iii) modelo de configuração – que especifica o mapeamento entre variabilidades presentes no modelo de característica e artefatos presentes no modelo de arquitetura. Tais modelos representam a realização do modelo generativo proposto por Czarnecki e Eisnecker (2000) e adotado por várias ferramentas de derivação existentes.

No caso de linhas de processos definidas usando o framework EPF, a ferramenta GenArch permite a criação automática dos modelos de derivação a partir do processamento de comentários/anotações nos arquivos XMI, que representam os elementos do processo sendo considerados. Esta seção discute e apresenta em mais detalhes, quais são tais modelos e como eles são usados para gerenciar as variações no GenArch. As Figuras 14, 15 e 16 apresentam uma visão parcial dos modelos de característica, configuração e arquitetura, respectivamente, considerando a linha de processos OpenUP.

A Figura 15 ilustra o modelo de *features* parcial do OpenUP, indicando suas similaridades e variabilidades. O modelo de *features* ilustra todas as *features* mandatórias, opcionais e alternativas encontradas durante a análise dos arquivos XMI contidos na estrutura de diretórios do processo. Para cada comentário/anotação representando uma variabilidade encontrada em um dos arquivos XMI, uma *feature* correspondente é adicionada ao modelo de característica.

O modelo de arquitetura de processo representa todos os elementos encontrados durante a análise da estrutura de diretórios presentes na especificação de processo no

EPF. A Figura 16 ilustra tal modelo contendo a estrutura de arquivos da especificação EPF do OpenUP que foi processada pelo GenArch. É possível visualizar diferentes níveis da estrutura de organização dos arquivos do nosso processo. O componente `process.openup.base`, presente na Figura 16, define o processo. Nos demais componentes, o EPF especifica partes reutilizáveis do processo, tal como, por exemplo, o componente `practice.mgmt.iterative_dev.base` que visa criar elementos do processo comuns a diversas iterações.

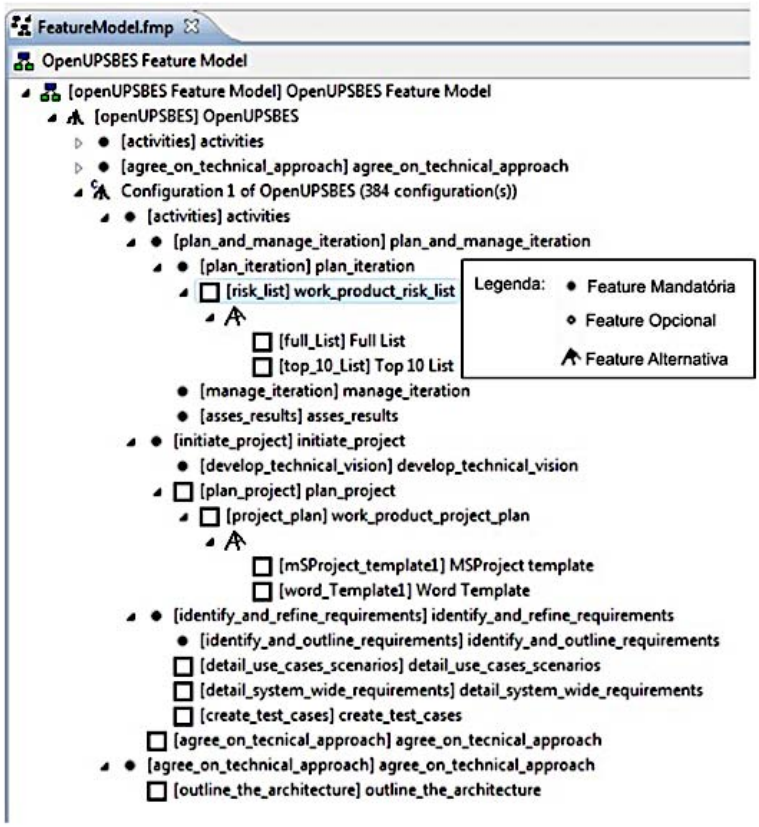


Figura 15. Modelo de features gerado pelo GenAcrh

Já o modelo de configuração, que representa a LPrS, é usado para relacionar elementos específicos do modelo de arquitetura de processo às *features* específicas presentes no modelo de *features*. A Figura 17 apresenta uma visão parcial do modelo de configuração do OpenUP, destacando os elementos do processo que tem dependência de alguma *feature*. A hierarquia no modelo de configuração ilustra a dependência entre elementos e *features* específicas. Por exemplo, dada a primeira *feature* que aparece no diagrama da Figura 17: `plan_and_manage_iteration`, a seleção ou não desta *feature* implica na inclusão ou não de um determinado trecho ou do arquivo `model.xmi` completo. Todos os modelos gerados são posteriormente usados para a customização de um processo durante o processo de derivação automático.

Modelagem de Variabilidades em Diferentes Granularidades. A modelagem das variabilidades existentes na linha de processos OpenUP exigiu a gerência de variabilidades em dois diferentes níveis de granularidade: (i) nível de arquivos XMI, que representam elementos de processo EPF (granularidade grossa); e (ii) nível de fragmentos em arquivos XMI, que contém informações de detalhamento de um dado elemento de processo e que são impactadas durante o processo de gerência de variabilidades (granularidade fina).

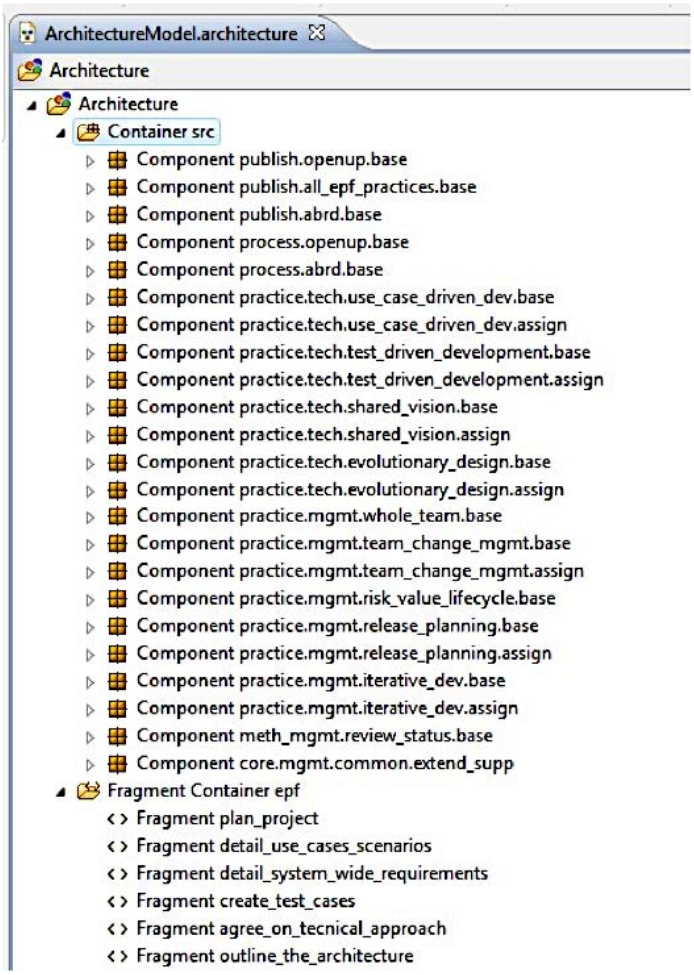


Figura 16. Modelo de Arquitetura gerado pelo GenArch

No caso de gerência de variabilidades de granularidade grossa, as seções anteriores ilustraram que o uso de comentários/anotações em arquivos XMI foi suficiente para permitir a criação automática dos modelos de derivação, assim garantindo que elementos de processo, tais como, atividades e tarefas pudessem ser tratados como variabilidades da linha de processos. Entretanto, alguns elementos de processo, assim como relacionamentos existentes entre tais elementos e que também podem representar variações, não são representados usando arquivos XMI individuais.

Por exemplo, os passos definidos para uma tarefa são especificados dentro do próprio arquivo XMI que descreve a atividade. Outros exemplos de elementos de processo que não são definidos dentro de arquivos XMI exclusivos, são: passos de uma atividade, artefatos de entrada e saída de uma atividade, papéis participantes de uma atividade, conceitos e guias relacionados a uma atividade.

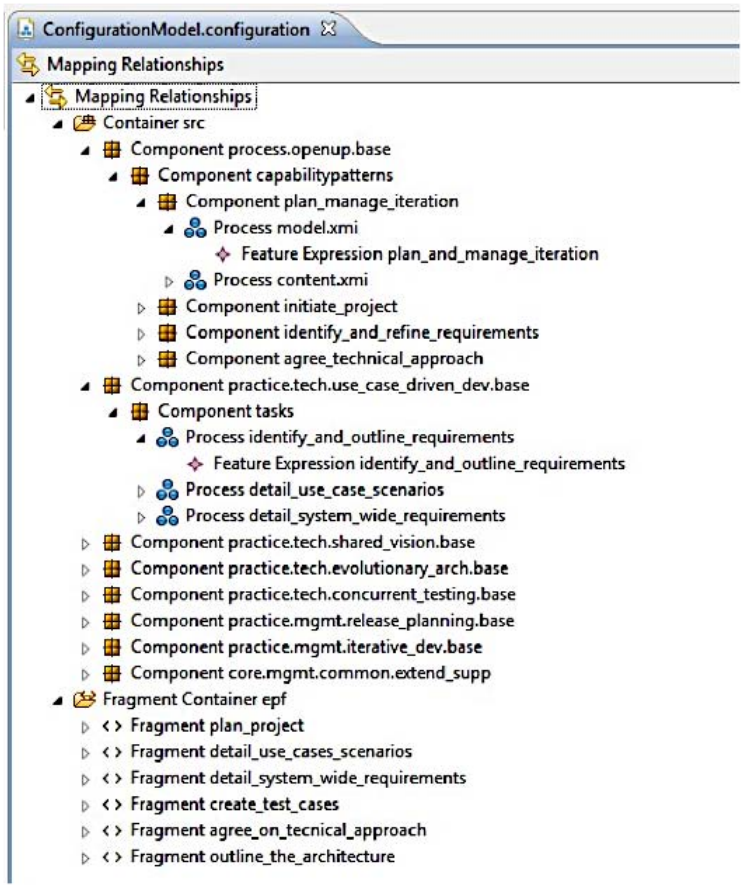


Figura 17. Modelo de configuração gerado pelo GenArch

Para garantir a gerência de variabilidades de menor granularidade, associadas a trechos da especificação do processo (arquivos XMI), foi utilizado outro recurso da ferramenta GenArch, denominado de fragmento (Cirilo et al., 2008). Um fragmento é caracterizado como um trecho de texto de qualquer tipo de artefato (código, arquivo de configuração, texto, etc.) que pode ser inserido (ou não) dependendo da escolha de uma dada variabilidade, durante o processo de derivação. Ferramentas como pure::varians (pure-systems, 2012) e Gears (BigLever Software, 2011) também oferecem mecanismos parecidos com um propósito similar.

As Figuras 16 e 17 também ilustram as definições de fragmentos dentro dos modelos de arquitetura e configuração da linha de processos. Cada um dos fragmentos

representa um trecho da especificação do processo, os quais estão associados a uma variabilidade. Cada trecho, portanto, representa uma variante a ser inserida dentro da especificação do processo, caso a variabilidade correspondente seja selecionada. Usando a ferramenta GenArch, é possível selecionar um trecho de um arquivo específico e extrair tal trecho automaticamente para um fragmento, realizando assim uma refatoração nos artefatos da LPS para expor tal variação. Tal refatoração transforma o arquivo cujo fragmento foi extraído em um *template* na linguagem XPand (Eclipse Foundation, 2012), além de criar as respectivas entradas no modelo de arquitetura e configuração para representar o fragmento.

A Figura 18 ilustra como fica um arquivo XMI após a extração de alguns trechos e associação destes aos fragmentos específicos. O trecho de arquivo ilustrado representa as tarefas relacionadas à atividade "*Identify and Refine Requirements*" da fase de concepção do OpenUP. Das quatro tarefas associadas a esta atividade, três foram identificadas como opcionais. Podemos observar no arquivo, por exemplo, que o trecho referente à tarefa "*Detail Use-Case Scenarios*" foi extraído para um fragmento denominado `detail_use_case_scenarios`, que foi associado com a *feature* de mesmo nome. Este mesmo procedimento foi repetido em relação às tarefas "*Detail Use Case Scenarios*" e "*Detail System-Wide Requirements*". Esta associação possibilita que cada um desses fragmentos seja inserido no arquivo da especificação final da instância de processo apenas se as *features* a eles relacionadas forem selecionadas.

```
    <processElements xsi:type="org.eclipse.epf.uma:WorkProductDescriptor" xmi:id="_EOwqqdOLEdyqlogshP8l4g"
        name="use_case_model" guid="_EOwqqdOLEdyqlogshP8l4g" presentationName="Use-Case Model"
        isPlanned="false" superActivities="_xxcpgdOEEdyqlogshP8l4g">
      <WorkProduct href="uma://_OOB2AAouEdyhlpBPrduOMw#_W2SgEDR5EdutE_HNDIJk5Q"/>
    </processElements>


    «REM»Code extracted to 'epfFragmentos/detail_use_case_scenarios' frament«ENDREM»
«LET fragment("epfFragmentos/detail_use_case_scenarios",architecture) AS e»«e.content»«ENDLET»


    «REM»Code extracted to 'epfFragmentos/detail_system_wide_requirements' frament«ENDREM»
«LET fragment("epfFragmentos/detail_system_wide_requirements",architecture) AS e»«e.content»«ENDLET»



    «REM»Code extracted to 'epfFragmentos/detail_system_wide_requirements' frament«ENDREM»
«LET fragment("epfFragmentos/detail_system_wide_requirements",architecture) AS e»«e.content»«ENDLET»


    <processElements xsi:type="org.eclipse.epf.uma:WorkProductDescriptor" xmi:id="_HDCWUdOLEdyqlogshP8l4g"
        name="technical_specification_slot" guid="_HDCWUdOLEdyqlogshP8l4g" presentationName="[Technical Spec
        isPlanned="false" superActivities="_xxcpgdOEEdyqlogshP8l4g">
      <WorkProduct href="uma://_-bv0ELXuEduGP_98XMd0fg#_i3vkoLS-EduDY8LNbMCDBA"/>
    </processElements>
```

Figura 18. Arquivo XMI após a retirada de fragmentos - Template XPand

4.4.1.3 Estágio 3: Derivação e Customização Automática de Processos

Esse estágio da abordagem consiste do processo de customização e derivação de um processo de software, a partir dos modelos de derivação e artefatos que representam a implementação da linha de processos. Assim, após a realização dos estágios anteriores, o engenheiro de processos pode selecionar o conjunto de características variáveis (variabilidades) que ele deseja para o seu processo, usando a ferramenta GenArch. Isso consiste basicamente na seleção explícita de quais *features* opcionais e alternativas se deseja ter no processo considerado. A seleção de *features* é baseada nas demandas específicas de um dado projeto, bem como na experiência dos engenheiros de processo, responsáveis pela customização do processo que será usado em tal projeto.

Importante lembrar que no modelo de configuração estão definidos os mapeamentos das *features* para os elementos do processo, sejam eles arquivos individuais ou trechos (fragmentos) de arquivos específicos. São esses mapeamentos que são processados pela ferramenta para decidir quais arquivos e fragmentos devem entrar na especificação de processo sendo gerada. Como resultado dessa geração automática, é criada uma estrutura de diretórios semelhante à definida para a linha de processos OpenUP, mas com as devidas adaptações realizadas levando em consideração as *features* selecionadas. Tal seleção implica na inclusão ou não de determinados arquivos ou trechos de arquivos (fragmentos) na especificação EPF que representa o processo desejado. Após a derivação, uma das possibilidades é a importação da especificação de processo resultante pelo EPF *Composer*, e a publicação do *site web* do processo. Mesmo tendo sido gerado o site do processo, segundo a abordagem proposta, a especificação de processo resultante deve ser preparada para ser implantada em um motor de *workflow*. Essa preparação é detalhada na próxima seção.

4.4.1.4 Estágio 4: Geração do Modelo de Workflow Correspondente

Para possibilitar que a especificação de processo resultante pudesse ser executada, e ter a sua execução acompanhada, a abordagem propõe a sua implantação em um motor de *workflow*. A estratégia adotada para implementar tal funcionalidade é aplicar uma transformação modelo para modelo da especificação EPF para uma especificação de *workflows* correspondente. O fluxo de atividades proposto para o processo é replicado em uma especificação de *workflow* com as mesmas características.

No estudo exploratório realizado, foi implementada uma ferramenta específica para gerar especificações de *workflows* que possam ser executados no jBPM *workflow engine* (Salatino, 2009) oferecido pela JBoss, a partir de especificações de processos EPF. Tal ferramenta realiza transformações de modelo para modelo e modelo para texto, recebendo um modelo segundo o meta-modelo UMA e devolvendo uma especificação de *workflow* em jPDL. As transformações foram escritas: (i) na linguagem QVT para a transformação do modelo de processo EPF gerado pelo GenArch para um modelo de workflow segundo um meta-modelo jPDL; e (ii) na tecnologia Acceleo para permitir a transformação de especificações jPDL para formulários *Java Server Faces* – JSF – que são instalados e executados no jBPM.

A Tabela 10 apresenta o mapeamento dos elementos de processo, segundo o meta-modelo UMA, e os elementos de *workflow*, segundo um meta-modelo jPDL. Mapeamento esse que foi implementado pela transformação em QVT correspondente. Os elementos gerados no modelo jPDL foram necessários para a geração dos arquivos de especificação jPDL, os quais serão implantados no motor de *workflow*, o qual proporcionará a execução do mesmo. O modelo jPDL gerado pela transformação é utilizado pela segunda transformação que gera a especificação do *workflows* em jPDL, um conjunto de páginas JSF representando as atividades e um descritor de implantação. O descritor de implantação é responsável por relacionar uma atividade específica do *workflow* à uma JSF contendo as informações sobre tal atividade. As páginas JSF permitem que o fluxo do processo seja monitorado, por meio do console Web do jBPM. Essa funcionalidade de monitoramento é responsável por armazenar informações sobre a execução das tarefas, ou mesmo decisões tomadas durante a execução do processo.

Tabela 10. Mapeamento de elementos UMA em elementos JPDL

Elemento de Processo – UMA	**Elemento de Workflow – jPDL**
Activity	*Task-node*
Workorder	*Transition*
Activities (com mais de um predecessor)	*Join*
Activities (com mais de um sucessor)	*Fork*
Activity (sem predecessor)	*Start-state*
Activity (sem sucessor)	*End-state*
Task	*Task-node*
Step	*Task*

Para possibilitar a geração do arquivo com a definição de processo segundo o *schema* jPDL, e as páginas JSF correspondentes às atividades do *workflow*, foi implementada uma transformação modelo para texto usando a linguagem *Acceleo*

(Obeo, 2012). *Acceleo* é uma ferramenta de geração de código, que permite transformar modelo em código. Esse processo de geração também gerou o arquivo `forms.xml`, que é um arquivo XML responsável por relacionar cada página JSF com uma tarefa específica do *workflow*. Como resultado final é criado um projeto jPDL no Eclipse contendo todos os arquivos gerados pela ferramenta desenvolvida. Projeto esse que já está pronto para ser implantado no motor de *workflow* do jBPM.

A Figura 19 ilustra um trecho do arquivo jPDL que define o processo. Nessa definição do processo em jPDL podem ser notados: (i) os *task-nodes* representando as atividades do processo de software; (ii) as variáveis de controle em cada *task-node*, responsáveis por armazenar informações sobre a execução e (iii) as transições que são realizadas quando uma dada tarefa é dada como concluída. No projeto jPDL do Eclipse, é possível ter uma visão gráfica do processo contido na definição. A visão gráfica do fragmento de definição de processo em jPDL da Figura 19 pode ser vista na Figura 20.

```
<?xml version="1.0" encoding="UTF-8"?>

<process-definition xmlns="" name="EPFArticle">
    <start-state name="start">
        <transition to="Plan the Project"></transition>
    </start-state>
    <!--  -->
    <task-node name="Request Change">
        <task name="Request Change" swimlane="start">
            <controller>
                <variable name="Request Change" access="read,write,required" />
            </controller>
        </task>
        <transition name="Request Change" to="Asses Result"></transition>
    </task-node>
    <!--  -->
    <task-node name="Plan the Project">
        <task name="Plan the Project" swimlane="start">
            <controller>
                <variable name="Plan the Project" access="read,write,required" />
            </controller>
        </task>
        <transition name="Plan the Project" to="Request Change"></transition>
    </task-node>
    <!--  -->
    <task-node name="Asses Result">
        <task name="Asses Result" swimlane="start">
            <controller>
                <variable name="Asses Result" access="read,write,required" />
            </controller>
        </task>
    </task-node>
```

Figura 19. Fragmento do Arquivo JPDL de Definição de Processo

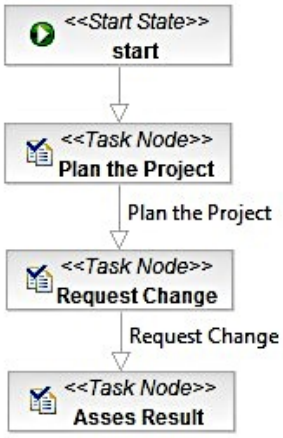


Figura 20. Visualização Gráfica do Fragmento de Definição de Processo

4.4.1.5 Estágio 5: Implantação e Execução do Modelo de Workflow Correspondente

O arquivo de definição de processo, ilustrado na Figura 19, é essencial para a implantação e execução do processo correspondente no motor de *workflow*. Além do arquivo de definição de processo, serão implantados no motor de *workflow* as páginas JSF e o arquivo de configuração `forms.xml`. Após a implantação desses elementos no motor de *workflow*, o usuário pode iniciar a execução de uma nova instância desse processo. A Figura 21 mostra o resultado da implantação do fragmento de processo ilustrado anteriormente.

Quando é iniciada a execução de uma nova instância do processo, o usuário pode visualizar o atual estado do processo e ter acesso a detalhes a respeito da atividade que precisa ser realizada. As informações e detalhes de cada atividade do processo foram extraídas das informações contidas na especificação EPF (UMA) do mesmo. Após a execução de cada atividade, o usuário informa o fim da mesma ao motor de *workflow*. Quando o usuário informa o fim de uma dada atividade, o motor de *workflow* solicita ao usuário informações adicionais sobre esta atividade, por meio de um formulário JSF específico. As informações passadas pelo usuário ao final de cada atividade são armazenadas pelo motor de *workflow* em um banco de dados interno, específico para esta finalidade. As informações adicionais armazenadas são devidamente relacionadas a uma instância especifica do processo, e podem ser recuperadas posteriormente. Ao concluir uma tarefa, o motor de *workflow* exibe a tarefa

seguinte que precisa ser realizada. Esses passos são repetidos até que a atividade que representa o estado final do processo seja alcançada.

Figura 21. Resultado da Implantação de um Processo no jBPM

4.4.2 Análise da Versão Inicial da Abordagem

Após a concepção e implementação da abordagem, a mesma foi aplicada na modelagem de uma linha de processos, a qual foi ilustrada no decorrer da explicação da abordagem. A partir do uso e aplicação da abordagem, foi possível identificar algumas limitações, as quais serão apresentadas e discutidas no decorrer desta seção, quais sejam: (i) a abordagem proposta utiliza a estratégia extrativa (Pohl et al., 2005) para a construção de uma LPrS; (ii) alto custo para a anotação manual dos arquivos XMI de especificação do processo; (iii) verificação de inconsistências no fluxo de atividades de alguns processos derivados; e (iv) qualquer alteração na especificação do processo base para a linha de processos requer que o mesmo seja novamente importado pela ferramenta, para que tais alterações possam ser refletidas para os membros da família.

Opção pelo uso exclusivo da estratégia extrativa. Das três abordagens primárias para o gerenciamento de escopo de uma LPS (Pohl et al., 2005) – proativa, reativa e extrativa – foi escolhida a abordagem extrativa para compor esse trabalho. A opção de se limitar à abordagem reativa se deu pelo fato que essa abordagem simplifica o processo de definição de escopo para uma LPrS. Tal processo é tão rico e complexo,

que encontramos trabalhos correlatos que limitam-se a discutir tal processo, como o trabalho proposto por Amrbrust (Armbrust et al., 2009). Segundo a abordagem extrativa, o responsável pela fase de definição de escopo limita-se a analisar um conjunto de processos membros da família desejada, tomando por base as definições dos elementos do processo e a estrutura de fluxo de trabalho do processo como um todo. Ao final dessa análise, este conclui quais elementos são mandatórios (ocorrem em todos os membros analisados), opcionais (ocorrem em apenas alguns membros) ou alternativos (cada membro pode trabalhar com elementos diferentes para a mesma finalidade). Após a estruturação da LPrS, são estudadas possíveis distorções em processos derivados, observando os aspectos da composição segura (Thaker et al., 2007). Distorções essas que podem dar início à uma reação no sentido de corrigir/aperfeiçoar a linha de processos. Para extrapolar essa limitação da abordagem proposta, ao final do trabalho, na seção de reflexões sobre a abordagem, será discutida como a mesma poderia ser adaptada para utilizar a abordagem proativa de LPS.

Alto custo da anotação manual dos arquivos XMI. A anotação manual dos arquivos XMI de uma especificação de processo de software, indicando a variabilidade específica de cada um dos elementos do processo, provou ser uma tarefa inviável. A opção de anotar arquivos do modelo de processo, utilizando comentários XML se deu com o intuito de facilitar a adaptação da ferramenta GenArch para o contexto de processos de software. Já estava previsto que a estratégia de anotação dos elementos de um processo de software precisaria ser melhorada, tanto que essa limitação foi um dos alvos principais dos melhoramentos na proposta inicial da abordagem. Foi proposta a anotação de variabilidades em uma visualização simplificada do modelo de processo, conforme será detalhado na próxima seção.

Inconsistências identificadas no fluxo de atividades de processos derivados. Também pode ser verificada a inconsistência no fluxo de atividades de alguns processos derivados. Essa inconsistência deu-se em situações em que a não opção por uma dada *feature* resultou na remoção de uma atividade do meio do fluxo de atividades. Esse fato causou a quebra do fluxo de atividades da instância de processo. Isso ocorreu porque, na primeira implementação da abordagem, a reconstrução e manutenção da consistência do fluxo de atividades, quando da retirada de uma atividade intermediária, não foi realizada. Haveria a possibilidade de corrigir esse problema inserindo vários trechos de código em uma dada atividade, fazendo referência a todas as possíveis atividades, e

isolar cada trecho desses em um fragmento, e associá-lo a uma *feature* específica. Tal problema não foi solucionado, durante a implementação da primeira versão da abordagem, visto que agravaria a limitação anteriormente citada. Fazendo com que o engenheiro de processo, responsável pela linha de processos, necessitasse de um conhecimento vasto sobre a estrutura dos arquivos XMI dos modelos UMA. Além de obrigar o engenheiro de processos à inserir manualmente uma série de trechos de código (XML) em vários arquivos XMI. A seção seguinte discute como a evolução da abordagem proposta, citada no parágrafo anterior, resolve também esta limitação.

Necessidade de um novo processo de importação em caso de evolução do modelo de processo original. Outra limitação identificada diz respeito aos ajustes/evolução do modelo de processo utilizado como base para uma família de processos, que uma vez realizados implicam que o mesmo seja novamente importado pela ferramenta. Após a evolução no processo base, torna-se necessário um novo processo de importação e geração dos modelos de derivação, para que as alterações no modelo base possam ser refletidas para os membros da família. Essa limitação se deu devido ao fato que uma vez importado o processo para a ferramenta GenArch, não foi prevista a possibilidade do mesmo ser aberto para edição na ferramenta EPF *Composer*. Para a solução dessa limitação foram elaboradas as opções de importação do modelo UMA e exportação para o modelo UMA, para o modelo base para a LPrS. Detalhes dessa nova funcionalidade estão descritas na próxima seção.

4.4.3 Evoluções na Versão Inicial da Abordagem

Com base na análise da aplicação da versão inicial da abordagem na modelagem de uma família de processos, foram realizadas melhorias para a abordagem. O principal melhoramento realizado foi a definição de um meta-modelo de processos simplificado, a ser utilizado nos demais estudos desta tese. Os objetivos desse modelo simplificado de processo são: (i) servir como uma versão simplificada do processo, a qual poderá ser visualizada graficamente; (ii) permitir que as anotações de variabilidades possam ser realizadas nesse modelo; e (iii) permitir a evolução da especificação original do processo de forma independente da linha de processo.

O modelo simplificado de processo substituiu os modelos de arquitetura (de processo) e de configuração, gerados automaticamente pela ferramenta GenArch. A

substituição do modelo de arquitetura dá-se pelo fato do modelo simplificado do processo também representar a estrutura do processo, não a estrutura física dos arquivos mas sim dos elementos descritos por tais arquivos. No modelo simplificado do processo, há uma referência aos arquivos físicos que representam os mesmos. A substituição do modelo de configuração se dá pelo fato do modelo simplificado permitir a inserção das informações de variabilidades, já associadas a um dado elemento.

A geração do modelo simplificado de processo, a partir de uma especificação de processo, passou a ser o primeiro passo da abordagem. O modelo simplificado de processo deverá ser anotado com as informações de variabilidade. Após a anotação de variabilidades, a ferramenta GenArch deverá gerar de forma automática o modelo de *features* correspondente. Para a derivação de instância da linha de processo a ferramenta GenArch deverá utilizar a especificação original do processo, o modelo simplificado de processo (anotado) e uma configuração de *features*.

A utilização de um modelo simplificado de processo pode resolver uma boa parte das limitações identificadas na versão inicial da abordagem. Primeiro, não será mais preciso a edição manual de arquivos XMI, o trabalho de anotação das variabilidades será feito no próprio modelo simplificado. O modelo simplificado ajudará também na manutenção da consistência dos processos a serem derivados pela linha de processos. Por exemplo: quando uma atividade, que compõe o fluxo de atividades, for considerada como opcional, além da informação de qual elemento será removido do processo a ser derivado, serão registradas as alterações de referência que serão necessárias para reestabelecer o fluxo de atividades.

Por fim, o modelo simplificado, é desvinculado da forma de especificação do processo utilizada, permitindo que sejam utilizados processos segundo diferentes formas de especificação como base para uma linha de processos. Essa desvinculação permite também que a especificação de processo evolua independentemente da modelagem da linha de processos. Nesses casos, de evolução da especificação original do processo após a configuração da linha de processos, torna-se necessário fazer uma sincronização entre o modelo simplificado e a nova especificação original do processo. Nessa sincronização todas as alterações estruturais sofridas pelo modelo de processo são refletidas no modelo simplificado. Após a anotação dos novos elementos, estão prontos para serem gerados os novos modelos de configuração e de *features*. Nas subseções a

seguir serão citados os impactos das mudanças realizadas em cada um dos estágios da abordagem.

4.4.3.1 Impactos das Evoluções nos Estágios da Abordagem

Nenhuma mudança substancial é percebida no **Estágio 1**. Duas novas ferramentas adicionadas à abordagem podem ser utilizadas nesse estágio, são elas: (i) geração automática do modelo simplificado do processo e (ii) visualização gráfica do modelo simplificado do processo. O modelo simplificado do processo base para a família de processos de software pode ser útil nesse estágio da abordagem, servindo de base para o estudo de similaridades e variabilidades. A sua utilização pode ser útil pois os elementos do processo, e os seus relacionamentos hierárquicos, ficam mais evidentes por meio da visualização do modelo. Nesse modelo simplificado os elementos que compõem o processo são agrupados pelo tipo (papéis, práticas, tarefas, etc.).

O grande impacto dos melhoramentos realizados na abordagem acontece no **Estágio 2**, pois a gerência automática de variabilidade passa a ocorrer por meio da edição de um modelo, e não mais por meio da manipulação direta de arquivos XMI. A anotação de variabilidades dos elementos do processo ocorre diretamente no modelo simplificado do processo. Esse nível de abstração permite que o engenheiro de processo não precise conhecer a estrutura da representação de baixo nível utilizada pelo processo. O trabalho de anotação de variabilidade em um modelo a parte da especificação do processo segundo o meta-modelo UMA não é intrusiva, permitindo que a especificação do processo possa ser alterada. É bem verdade que após qualquer alteração no modelo do processo original é necessário uma sincronização, preferencialmente automática, com o modelo simplificado, para que este reflita as alterações sofridas pelo mesmo.

Após a anotação do modelo simplificado, os dois modelos – o modelo de processo original (UMA) e o modelo simplificado – são processados pela ferramenta GenArch, que gera então o modelo de *features*. A ferramenta GenArch, de derivação de produtos, necessitou passar por novas adaptações para estar de acordo com o modelo simplificado de processo. Com esses modelos – (i) especificação do processo (UMA), (ii) modelo simplificado do processo anotado com as variabilidades, e (iii) modelo de *features* – a ferramenta GenArch já está pronta para derivar processos, com base em configurações de *features*.

Do ponto de vista do usuário da abordagem, o **Estágio 3** não sofreu alteração com relação a versão inicial proposta para a abordagem. O processo continua partindo da instanciação do modelo de *features* gerado pela ferramenta GenArch, seguido pela seleção das características desejadas para o novo processo. Por fim, após a seleção de *features*, a ferramenta GenArch é utilizada para derivar um novo processo, com base nas *features* selecionadas.

Do ponto de vista da infraestrutura da abordagem, a ferramenta GenArch precisou ser adaptada para trabalhar com o modelo simplificado de processo, que substituiu os modelos de arquitetura e configuração. O processo de derivação da ferramenta passou a obedecer a seguinte sequência de ações: (i) com base no modelo simplificado do processo, serão selecionados os elementos de processo que irão fazer parte do processo sendo derivado; (ii) para cada elemento associado à uma *feature*, é verificado se a *feature* em questão foi selecionada, não tendo sido selecionada os elementos correspondentes não serão incluídos na instância de processo derivada; (iii) em caso de uma atividade sendo removida, o fluxo de atividades é corrigido no modelo de processo. Ao final desse processo, todos os elementos compatíveis com as *features* selecionadas pelo usuário são copiados para um diretório específico, cujo resultado final é uma especificação de processo de software, segundo o UMA.

Os melhoramentos propostos para a abordagem não geram nenhum impacto direto sobre o **Estágio 4** da abordagem. Mas, percebeu-se um benefício com relação à transformação do fluxo de atividades de processo para um *workflow*. No estágio anterior, ao se corrigir o fluxo de atividades, após a retirada de uma dada atividade do processo, isso implicou que o *workflow* correspondente também ficasse consistente. Não foi implementado até este ponto do trabalho, mas percebeu-se ser possível a inserção de informações pertinentes à execução no modelo simplificado do processo. A inclusão de informações relativas à execução no modelo simplificado de processo permitiria incrementar o processo de geração do *workflow* correspondente. Essa possibilidade será avaliada nos trabalhos futuros de melhoria da abordagem.

O **Estágio 5** permanece como definido na versão inicial da abordagem. Embora caibam melhoramentos para esse estágio; visando permitir a gerência de variabilidades em tempo de implantação e execução de processos; tais melhoramentos estão fora do escopo definido para esta tese.

4.5 Análise dos Resultados do Estudo Exploratório

Todas as fases propostas para o estudo exploratório foram realizadas conforme planejado. Foi proposta e implementada uma abordagem anotativa para a gerência de variabilidades em LPrSs. A versão inicial da abordagem foi aplicada na modelagem de uma linha de processos baseada no OpenUP. Por fim, foi possível propor alguns melhoramentos na versão inicial da abordagem e analisar quais seriam os impactos de tais melhoramentos em cada um dos estágios da abordagem. Com base nas informações da execução das fases planejadas para o estudo exploratório, foram respondidas as questões de pesquisa definidas.

Com relação a primeira questão de pesquisa, "*qual a viabilidade e como mecanismos de gerência de variabilidades anotativas podem ser usados para o contexto de linhas de processo de software?*", a implementação do suporte ferramental necessário para a abordagem e a aplicação da mesma na modelagem de uma LPrS demonstrou a viabilidade da aplicação das técnicas anotativas ao contexto de LPrSs. A adaptação da ferramenta de derivação de produtos de software – GenArch – ao contexto de linhas de processo de software, respondeu ao "como?" da primeira questão de pesquisa. Apresentamos indícios, por meio de um estudo exploratório, que a abordagem anotativa pode ser utilizada na modelagem de linhas de processo de software. As limitações apresentadas durante a aplicação da abordagem, serviram de base para a avaliação de como foram empregadas as técnicas anotativas no contexto de processos de software, e como melhor empregá-las.

Para responder a segunda questão de pesquisa, "*quais os benefícios e limitações de tais mecanismos quando usados no contexto de linhas de processo de software?*", a versão inicial da abordagem foi aplicada e analisada, visando identificar os benefícios e limitações da mesma. O principal benefício identificado foi que a abordagem anotativa permitiu o gerenciamento sistemático de variabilidades em processos de software. A utilização da abordagem possibilitou a execução de tarefas, tais como: (i) a definição de variabilidades (por meio da anotação da especificação do processo); (ii) a representação das variabilidades da linha de processos na forma de um modelo de *features*; (iii) a escolha de *features* para uma nova instância da linha de processos; e (iv) a derivação de uma instância de processo com as variações definidas pelas *features* selecionadas. A execução das tarefas supracitadas permitem um gerenciamento sistemático de

variabilidades durante a modelagem de uma LPrS. As limitações identificadas durante a aplicação da versão inicial da abordagem, discutidas na Seção 4.2.2, ressaltaram o fato de que a aplicação das técnicas anotativas ao contexto de LPrS não é trivial, mas é possível e viável, devendo ser explorada com maior profundidade em estudos posteriores.

4.6 Discussões e Limitações do Estudo

Esta seção apresenta e discute aspectos técnicos, lições aprendidas, assim como novas perspectivas que surgiram ao longo do desenvolvimento da abordagem apresentada neste trabalho. Também são apresentadas e discutidas algumas limitações do estudo. As discussões apresentadas nesta seção abordam a utilização da abordagem anotativa, mas também extrapolam para tratar da implementação de LPrSs de forma geral. As discussões serão apresentadas na forma de tópicos, os quais são relevantes para o desenvolvimento de abordagens de gerência automática de variabilidades em processos de software. Tais discussões deverão nortear os rumos a serem tomadas pelas pesquisas futuras.

Especificação Multi-nível do Modelo de *Features*. Uma das possibilidades que está sendo atualmente explorada e que pode agregar valor à abordagem proposta é a possibilidade de expressar as *features* da linha de processos na forma de perguntas. Essas perguntas seriam respondidas na configuração de um novo processo, trazendo facilidades para a tomada de decisão e interação do engenheiro de processos durante a etapa de derivação da instância de processo. O foco principal de tais perguntas pode ser em tornar explícitas *features* de alto nível relacionadas às diretrizes do projeto. Um exemplo de pergunta seria: “Qual a linguagem de programação utilizada no projeto?”. Por meio da resposta fornecida pelo usuário, diferentes atividades, tarefas, artefatos e guias do processo seriam automaticamente selecionados e customizados para garantir o atendimento de tal requisito. No caso de seleção da linguagem Java, por exemplo, a tarefa de “realizar testes unitários utilizando JUnit” seria incluída no processo. A proposta inicial seria organizar o modelo de características em dois diferentes níveis: (i) das perguntas de mais alto nível; e (ii) do nível de variabilidades encontradas diretamente na linha de processo.

Gerência de Variações em Métricas de Processo. A importância da utilização de métricas para a engenharia de software já é um assunto consolidado, seja para medir produtos, processos ou recursos. No que diz respeito a processos, sua aplicação pode ser útil para entender e aperfeiçoar o processo de desenvolvimento, avaliar sua produtividade, identificar as melhores práticas, entre outros. Dada a grande variedade de métricas existentes, selecionar quais métricas devem ser aplicadas em determinado processo de desenvolvimento de software não é uma tarefa trivial. De acordo com as necessidades de cada projeto, uma série de métricas alternativas e complementares podem ser aplicadas. Neste cenário, é interessante que as variabilidades nas métricas de um processo possam também ser gerenciadas durante a fase de reutilização e customização de uma linha de processo usando uma abordagem de gerência de variabilidades, tal como a apresentada neste trabalho. Atualmente, diversas métricas de avaliação de produtividade estão sendo investigadas pelo grupo para serem tratadas como variabilidades da linha de processo OpenUP, e desta forma habilitar o engenheiro de processo a derivar um processo contendo métricas alinhadas com as fases, atividades, técnicas e tecnologias que ele contém.

No decorrer da realização e durante a análise dos resultados deste estudo exploratório foram identificadas algumas limitações do mesmo, quais sejam: (i) a concepção e implementação da abordagem ficou restrita à experiência da equipe responsável na área; (ii) a natureza do estudo não permitiu a extração de dados quantitativos para reforçar as respostas das questões de pesquisa; e (iii) a abordagem anotativa sendo proposta não foi comparada com outras abordagens similares, no intuito de discutir a sua aplicabilidade frente a outras alternativas.

A equipe responsável pelo desenvolvimento da abordagem tem conhecimento na área de linhas de produtos de software. Um dos pesquisadores que integrou a equipe responsável pela execução deste estudo, participou da idealização da ferramenta GenArch (Cirilo et al., 2008) (Cirilo et al., 2011). Os demais pesquisadores envolvidos no desenvolvimento trabalham juntos na idealização e adaptação das técnicas e mecanismos de linhas de produto de software ao contexto de processos de software, desde o seu início (Aleixo et al., 2010c). A experiência da equipe responsável pela execução do estudo com a ferramenta GenArch e com os conceitos de LPrS contribuiu para minimizar essa ameaça à validade do estudo.

Mesmo não tendo sido possível a extração de dados quantitativos, a validade do estudo se mantém, visto que por sua natureza um estudo exploratório tem por missão principal proporcionar a familiaridade com o problema, com vistas a torná-lo mais explícito ou a construir hipóteses (Kitchenham, 2004). O estudo proporcionou um melhor entendimento do problema de modularização de variabilidades em LPrSs, por meio da concepção, implementação e aplicação da abordagem proposta. Estudos comparativos entre a abordagem proposta e uma abordagem composicional para a gerência de variabilidades em processos de software – EPF *Composer* – serão apresentadas nos capítulos subsequentes.

4.7 Conclusões

Com base nos resultados obtidos e apresentados neste capítulo, é possível afirmar que o estudo exploratório cumpriu o seu papel, relacionado à avaliação preliminar da concepção e implementação de uma abordagem anotativa para a gerência de variabilidades em processos de software. A melhor compreensão do domínio do problema motivou a definição de novas questões de pesquisa, tais como: (i) quais vantagens e limitações da abordagem anotativa comparada à abordagem composicional para a gerência de variabilidades em LPrSs? (ii) quais as propostas de abordagem existentes, em qual classificação se encaixam e em que se diferenciam umas das outras? (iii) quais os tipos de variabilidade em processos de software existentes e como as referidas abordagens são utilizadas para modelar tais tipos de variabilidade? Tais questões de pesquisa foram exploradas nos demais estudos realizados durante o desenvolvimento deste trabalho, ou serão exploradas na forma de trabalhos futuros.

5 Estudo Comparativo das Abordagens Composicional e Anotativa para a Modelagem de LPrSs

Este capítulo apresenta um estudo comparativo qualitativo da utilização das abordagens composicional e anotativa na modelagem de LPrSs. A abordagem composicional é representada pelo EPF *Composer* (Eclipse Foudation, 2010), uma ferramenta industrial para engenharia de processos de software. A abordagem anotativa, por sua vez, é representada pelo GenArch-P (acrônimo de *GenArch Process*) (Aleixo et al., 2010c). O GenArch-P é uma adaptação da ferramenta de derivação de produtos – GenArch (Cirilo et al., 2008), a qual oferece suporte para a criação de modelos generativos que representam as variabilidades de um processo e associa tais variabilidade a elementos específicos de processo.

O capítulo está organizado da seguinte forma: a Seção 5.1 apresenta uma visão geral das duas abordagens orientadas a modelo investigadas. A Seção 5.2 descreve a configuração do estudo, a LPrS alvo e os critérios de comparação. A Seção 5.3 descreve a realização do estudo e os resultados obtidos. Finalmente, conclusões e discussões sobre o estudo realizado são apresentadas na Seção 5.4.

5.1 Abordagens Orientadas a Modelos para LPrSs

Esta seção apresenta de forma geral os representantes das abordagens investigadas que serão utilizados no estudo. A subseção 5.1.1 descreve o EPF *Composer*, representante da abordagem composicional. São apresentados alguns conceitos e mecanismos importantes desta abordagem. Já a subseção 5.1.2 apresenta os conceitos que dão suporte ao GenArch-P, representante da abordagem anotativa.

5.1.1 Abordagem Composicional: EPF Composer

O EPF *Composer* é uma ferramenta de autoria de processos que faz parte da iniciativa *Eclipse Process Framework* (Eclipse Foundation, 2012), que visa dotar os engenheiros de processo de conteúdo e ferramentas para a autoria, edição e customização de processos de software. No EPF os conteúdos de um processo são organizados como “conteúdo de método” (*method content*) e na forma de fluxos de atividades. Os elementos de processo que fazem parte do conteúdo de método podem ser agrupados em “pacotes de conteúdo” (*content packages*). Esses elementos de

processo descrevem o que deve ser produzido, as habilidade necessárias, e explicações sobre como uma tarefa deve ser realizadas, entre outros. Variabilidade de processos, segundo o EPF, são tratadas por meio de mecanismos de refinamento específicos, denominados de "variabilidade de conteúdo de método" (*method content variability*), as quais são: "*contributes*", "*extends*", "*replace*", e "*extends and replace*" (Haumer, 2007) (Eclipse Foudation, 2010), descritas nos próximos parágrafos.

Usando o mecanismo de variabilidade "*contributes*", um elemento de processo especializado pode adicionar conteúdo às propriedades de um outro elemento de processo (elemento base), de uma forma em que não altera o conteúdo do elemento base. Por exemplo, se o elemento "contribuinte" define uma descrição em uma de suas propriedades, esse texto é adicionado ao fim do texto da mesma propriedade do elemento base. Quando o processo é publicado, o elemento base aparece com as novas contribuições. Adotando o mecanismo de variabilidade "*extends*" um elemento especializado herda conteúdo de um elemento base, podendo sobrescrever as propriedades que desejar. Quando o processo é publicado, ambos os elementos (extensão e base) são publicados no processo. O mecanismo de variabilidade "*replace*" permite que um elemento tome o lugar de um outro elemento. Quando o processo é publicado, o elemento "substituto" é apresentado, enquanto o elemento base é deixado intocado, mas não é exibido. Finalmente, o mecanismo de variabilidade "*extends and replace*" combina os efeitos dos dois últimos mecanismos. Enquanto o mecanismo "*replace*" permite que um elemento tome o lugar de outro (não aproveitando nenhuma de suas propriedades), este tipo de variabilidade permite que as propriedades não definidas no elemento "substituto" sejam herdadas do elemento base. As propriedades definidas no elemento "substituto" sobrescrevem as propriedades do elemento base.

5.1.2 Abordagem Anotativa: GenArch-P

O GenArch-P é uma ferramenta de derivação de processos de software, adaptada de uma abordagem orientada a modelo de derivação para linhas de produtos, denominada de GenArch (Cirilo et al., 2008). A ferramenta trabalha com a abordagem anotativa, onde elementos de processos de software variáveis são anotados e associados à *features* específicas. A abordagem anotativa é utilizada em várias ferramentas para linhas de produtos de software, tais como: pure::variants, Gears e CIDE. O GenArch-P especifica dois modelos: (i) o modelo de processo, e (ii) o modelo de *features*. Alguns

elementos do modelo de processo são anotados para refletir que estão associados a *features* opcionais e/ou alternativas da LPrS. As anotações no GenArch-P são realizadas em elementos do modelo de processo, associando-o à uma definição de variabilidade, a qual descreve o tipo de variação do elemento e o nome da *feature* associada. Após a anotação de variabilidade nos elementos do modelo de processo, este modelo se torna a representação do conhecimento de configuração. O conhecimento de configuração define o mapeamento entre as *features* e os elementos de processo associados.

Após a anotação das variabilidades no modelo de processo, o GenArch-P gera automaticamente o modelo de *features* correspondente. O modelo de *features* pode ser usado para modelar e especificar as similaridades e as variabilidades em uma linha de produtos de software. Esse modelo apresenta todas as possíveis opções que podem ser selecionadas, em termos de *features*. Para derivar um novo processo, o usuário cria uma nova configuração de *features* e seleciona as *features* desejadas para o novo processo. Após isso, a ferramenta GenArch-P está pronta para derivar automaticamente instâncias de processo de acordo com as *features* selecionadas.

5.2 Configuração do Estudo

Para permitir a análise comparativa das abordagens composicional e anotativa para a modelagem de LPrSs, foi desenvolvido um estudo comparativo qualitativo das abordagens EPF *Composer* e GenArch-P. Nesse estudo foram modeladas duas linhas de processos, com diferentes tipos de variabilidades de processo, utilizando as duas abordagens avaliadas. Uma das linhas de processos modelada foi baseada em uma família de processos OpenUP, a outra linha foi baseada no *framework* de processos Scrum. A LPrS baseada no OpenUP foi extraída a partir de três projetos de pesquisa e desenvolvimento que foram desenvolvidos no IFRN e instituições parceiras. Já a linha de processos do Scrum foi baseada em variações de tal processo encontradas na Internet e literatura (Scrum.org, 2012) (Scrum Alliance, Inc., 2012) (Mountain Goat Software, 2012) (Eclipse Foundation, 2012) (Clifton & Dunlap, 2003). Após a modelagem das LPrSs usando o EPF *Composer* e o GenArch-P, foi realizada uma análise comparativa dos resultados finais com base em critérios adaptados de trabalhos anteriores (Kästner & Apel, 2008b) (Kästner, 2010). A Seção 5.2.1 descreve as fases do estudo e discute os procedimentos de avaliação. A Seção 5.2.2 apresenta as LPrSs utilizadas no estudo. A Seção 5.2.3 descreve os critérios de comparação adotados.

5.2.1 Fases do Estudo e Procedimentos de Análise

O estudo foi organizado em quatro fases principais: (1ª) definição de escopo para as LPrSs escolhidas, selecionando as instâncias de processo a serem usadas como base para a extração das linhas de processos escolhidas; (2ª) identificação e modelagem das variabilidades das linhas de processos, utilizando as abordagens investigadas; (3ª) derivação de processos customizados, para analisar os mecanismos de gerência de variabilidades adotados pelas abordagens investigadas; e (4ª) análise dos resultados e avaliação segundo os critérios definidos. A seguir são apresentados mais detalhes de cada uma das fases do estudo. O estudo foi realizado por cinco pesquisadores: um pesquisador doutor, dois estudantes de doutorado (líderes de modelagem) e dois estudantes de mestrado (auxiliares de modelagem).

Na primeira fase, foi realizada a seleção dos processos que foram utilizados de base para as duas LPrSs. Também foi coletada a documentação desses projetos, os quais serviram de base para a análise de similaridades e variabilidades entre os membros das famílias, possibilitando a extração da LPrS. Na segunda fase, foram identificadas e modeladas cada uma das variabilidades das LPrSs com a respectiva definição dos seus elementos comuns e variáveis. Na terceira fase do estudo, foram derivadas instâncias de processos das linhas de processos criadas, onde cada instância de processo foi baseada em uma configuração de *features*. Tal derivação de instâncias foi útil para validar as linhas de processos modeladas. Na quarta, e última fase, foram sumarizados os resultados do estudo, os quais foram analisados na perspectiva dos critérios definidos. Os critérios de avaliação utilizados foram escolhidos por permitirem a análise da estrutura, modularidade e o efetivo gerenciamento de variabilidades das abordagens investigadas.

5.2.2 LPrSs Alvo

Na primeira fase do estudo, aconteceu a definição das duas LPrSs que foram usadas para permitir a comparação das abordagens investigadas. Foram definidas duas LPrSs: a primeira extraída de customizações do OpenUP e a segunda extraída de customizações do Scrum. A especificação completa das linhas de processos a serem modeladas com as abordagens selecionadas foi realizada usando a técnica extrativa (Linden et al., 2007). Essa técnica envolveu a análise de similaridades e variabilidades das instâncias de processo representando membros da família de processos desejada. Os

elementos de processo comuns aos exemplares analisados formaram o núcleo da linha de processos, enquanto os elementos variáveis entre um exemplar e outro foram modelados separadamente e preferencialmente associados a *features* de alto nível.

5.2.2.1 Linha de Processos Baseada no OpenUP

O escopo da linha de processos baseada no OpenUP foi definido por meio da seleção de três processos baseados no OpenUP, que foram utilizados em projetos de pesquisa e desenvolvimento. Os processos baseados no OpenUP foram extraídos de projetos de pesquisa e desenvolvimento desenvolvidos em cooperação entre o IFRN e instituições parceiras. O primeiro projeto tratou do desenvolvimento do um sistema de software para auditoria em redes de telefonia. Esse sistema realiza a contagem, sumarização e a análise de registros de conexão telefônica, criados por um hardware específico. O segundo projeto envolveu o desenvolvimento de um módulo de um sistema distribuído, responsável pelo recolhimento e armazenamento das informações relacionadas às instituições federais de educação profissional e tecnológica do Brasil. O terceiro, e último, projeto envolveu a implementação de um sistema integrado de gestão acadêmica e administrativa, para as instituições federais de educação profissional e tecnológica no Brasil. A Tabela 11 mostra exemplos de *features* de alto nível e alguns elementos variáveis associados a elas, para a linha de processos baseada no OpenUP.

Tabela 11. Exemplo de features de alto nível da linha de processos do OpenUP e exemplos de elementos associados a tais features

Exemplos de *Features*	**Alguns elementos de processos associados**
Técnica de especificação de requisitos por meio de casos de uso (alternativa)	**(1)** Papel: Analyst **(2)** Prática técnica: Use Case Driven Development **(3)** Produto de trabalho: Use-Case Model **(4)** Produto de trabalho: Use Case **(5)** Checklist: Use Case **(6)** Checklist: Use Case Model **(7)** Conceito: Actor **(8)** Conceito: Use Case
Elementos adicionais oriundos do processo Scrum	**(1)** Produto de trabalho: Work Items List **(2)** Orientação: Risk List **(3)** Orientação: Work Items List **(4)** Conceito: Retrospective **(5)** Conceito: Risk **(6)** Exemplo: Iteration Burndown Report **(7)** Exemplo: Project Burndown Report

Para essa linha de processos foram identificados: (i) 75 elementos de processo como parte do núcleo da linha de processos; (ii) 9 *features* opcionais; (iii) 9 *features* alternativas; (iv) 4 OR-*features*. A Tabela 12 apresenta os nomes das *features*

identificadas e o número de elementos de processo associados a cada uma delas, caso uma *feature* possua várias alternativas, as mesmas também são apresentadas. Durante a análise da LPrS baseada no OpenUP também foram identificadas restrições entre *features*. Por exemplo, as *features* representando a utilização dos frameworks JEE e JUnit requerem a seleção da *feature* que define a utilização da linguagem de programação Java. Outro exemplo de restrição pode ser visto na seleção da documentação da arquitetura de acordo com o projeto ágil (*agile design*) (Ambler, 2011), que desabilita a seleção da *feature* relativa a ferramenta de especificação de projeto, visto que segundo metodologias ágeis o projeto é geralmente realizado no papel ou quadros brancos (Ambler, 2011).

Tabela 12. Features da linha de processos baseada no OpenUP

Feature	**Alternativas (se existirem)**	**Elementos relacionados**
Influência de outros processos		
Elementos adicionais oriundos do Scrum		25
Técnicas e tecnologias de requisitos		
Técnica de especificação	Casos de uso	27
	Estórias de usuário (*users stories*)	27
	Product backlog	27
Ferramenta de especificação	Astah Community	24
	Rational Software Architect	24
	Borland Together	24
	ArgoUML	24
Técnicas e tecnologias de projeto		
Documentação da arquitetura	Projeto ágil (*agile design*)	64
	Arquitetura bem documentada	16
Técnicas e tecnologias de implementação		
Linguagem de programação (OR-*feature*)	Java	7
	C#	7
	Ruby	7
	Phyton	7
Utilização do framework Java EE		4
Utilização da IDE Eclipse		5
Utilização do framework JUnit		15
Técnicas e tecnologias de integração contínua		
Utilização da ferramenta Hudson		8
Técnicas e tecnologias de métricas		
Utilização da ferramenta Maven (métricas de código mineradas do SVN)		5
Aplicação da métrica para avaliar o progresso das atividades		3
Aplicação da métrica para avaliar o cumprimento de *deadlines*		3
Aplicação da métrica para avaliar a duração das atividades principais		3

5.2.2.2 Linha de Processos Baseada no Scrum

O escopo da linha de processos baseada no Scrum foi definido baseado na experiência dos pesquisadores e algumas instanciações do processo Scrum publicadas na Internet, tais como: (i) Scrum.org (Scrum.org, 2012); (ii) Scrum Alliance (Scrum

Alliance, Inc., 2012); (iii) Scrum definido pela Mountain Goat Software (Mountain Goat Software, 2012); (iv) Scrum definido pelo EPF (Eclipse Foundation, 2012); e (v) Scrum definido pelo CodeProject (Clifton & Dunlap, 2003).

A Tabela 13 apresenta as *features* identificadas na linha de processos baseada no Scrum. Diferentemente da linha de processos baseada no OpenUP, dada a sua natureza simplificada, não foram identificadas features que afetassem mais de um elemento do processo – *features* de alto-nível. As *features* da linha de processos baseada no Scrum atendem a um grande leque de customizações do Scrum, das mais tradicionais até as mais minimalistas. Outro aspecto da linha de processo baseada no Scrum é que alguns elementos apresentam variações internas, como por exemplo: (i) a reunião de planejamento de próxima *sprint* pode opcionalmente incluir a prática de "jogo do planejamento" (*planning poker*); (ii) o *product backlog* (que armazena a lista de itens a serem desenvolvidos) pode ser priorizada pela complexidade dos seus itens, ou alternativamente pelo valor do negócio de cada item; (iii) um item do *product backlog* é geralmente definido como um módulo de código a ser desenvolvido, mas também pode ser de outros tipos, tais como: um plano da fase implementação, relatório de implementação, um projeto detalhado, a definição de uma interface com o usuário ou testes unitários (*features* relacionadas – OR-*feature*).

Tabela 13. Features da linha de processos baseada no Scrum

Elemento do processo Scrum	Tipo do Elemento	Tipo da *Feature*
Sprint planning meeting	Cerimônia	Mandatória
Daily meeting	Cerimônia	Opcional
Sprint review meeting	Cerimônia	Opcional
Retrospective	Cerimônia	Opcional
Product backlog	Artefato	Mandatória
Product backlog item	(Artefato) Conceito	Mandatória
Sprint backlog	Artefato	Mandatória
Burn down charts	Artefato	Opcional
Impediment registry	Artefato	Opcional
Scrum board	Artefato	Opcional
Release plan	Artefato	Opcional
Story cards	Artefato	Opcional
Scrum master	Papel	Mandatória
Scrum team	Papel	Mandatória
Product owner	Papel	Opcional
Velocity	Métrica	Opcional

5.2.3 Critérios de Comparação

Objetivando promover a avaliação de técnicas de implementação de linhas de produtos de software, Kästner et al. (Kästner & Apel, 2008b) (Kästner, 2010) (Kästner

et al., 2010) definiram um conjunto de critérios de comparação para esse propósito. Neste trabalho, alguns desses critérios foram adaptados para o contexto de LPrS, quais sejam: (i) modularidade, (ii) rastreabilidade, (iii) detecção de erros, (iv) granularidade, (v) uniformidade e (vi) adoção. Adicionalmente, as abordagens também foram analisadas com relação ao (vii) suporte à gerência sistemática de variabilidades. A seguir tais critérios são resumidamente explicados.

O critério de **modularidade** permite a análise do grau de modularização dos elementos de processos associados à *features* específicas, usando as diferentes abordagens. Em linhas de produtos de software uma modularização adequada permite o isolamento da implementação de uma *feature* específica e reduz a complexidade geral do código (Kästner, 2010). Com o apropriado isolamento das *features*, o desenvolvedor pode entender, checar e testá-las de forma independente, contribuindo dessa forma para menores custos de desenvolvimento e manutenção. De forma análoga, no contexto de LPrS, a modularização dos elementos de processos associados à *features* específicas possibilita um melhor entendimento e facilita as manutenções e evoluções da especificação da linha de processos.

O critério de **rastreabilidade** analisa o quão fácil é a visualização do mapeamento entre *features* e elementos de processo. Segundo Kästner (Kästner, 2010), a visualização dos elementos associados à uma dada *feature* permite ao desenvolvedor trabalhar com segurança em tais elementos para melhorar ou consertar um erro identificado em uma *feature*. Tal visualização também é importante para LPrSs no sentido de promover o rastreamento dos elementos de processo existentes que geralmente estão associados a *features* específicas de alto-nível.

O critério de **detecção de erros** tem o objetivo de analisar como as abordagens existentes oferecem suporte à checagem de consistência de uma linha de processos e dos seus processos derivados. Segundo Kästner (Kästner, 2010), a detecção de erros é útil para evitar os altos custos de manutenção na sequência do ciclo de desenvolvimento, corrigindo erros de tipo, sintáticos e semânticos o quanto antes possível. No contexto do nosso estudo, esse critério está relacionado à checagem de consistência na definição da linha de processos, e seus elementos, bem como nos processos derivados de tais linhas. Exemplos de falta de consistência podem ser: (i) na definição da linha de processos – a impossibilidade de garantir as restrições entre as

features; e (ii) em uma instância de processo – a referência a um elemento não presente, ou uma quebra abrupta no fluxo de atividades do processo.

O critério de **granularidade** se refere ao suporte da abordagem para a associação de *features* a elementos de diferentes granularidades (Kästner, 2010). Uma LPrS também pode ser composta de variabilidades de granularidades grossa e fina. Por isso, é fundamental que as abordagens para a modelagem de linhas de processos ofereçam suporte e diferentes mecanismos para modularizar variabilidades de processos em diferentes níveis de granularidade.

Segundo Kästner (Kästner, 2010), o critério de **uniformidade** é definido como a habilidade de oferecer suporte uniforme à múltiplas linguagens (de código e não-código). No contexto de linhas de processo de software, esse critério é responsável por avaliar o quanto uma abordagem específica é independente das possíveis formas de especificação de processos de software. Para cada abordagem, é analisada a possibilidade de se trabalhar com processos de software especificados segundo diferentes meta-modelos. O fato da abordagem estar restrita a um único meta-modelo para a especificação de processos de software torna a abordagem menos genérica, e desconsidera as vantagens de outras formas de especificação de processos de software.

O critério de **adoção** analisa a dificuldade de adoção de uma abordagem existente em termos da quantidade de conhecimentos necessários para a aplicação da abordagem. De forma similar a Kästner (Kästner, 2010), foi levantada a quantidade de conceitos, mecanismos e ferramentas necessárias durante a aplicação da abordagem. Para a análise desse critério, não foi levada em consideração a usabilidade das ferramentas disponibilizadas pelas diferentes abordagens. Foi levado em consideração apenas a complexidade de conceitos e mecanismos que precisam ser entendidos para a utilização das ferramentas propostas.

No critério de **gerência sistemática de variabilidades**, foram analisados os mecanismos da abordagem visando o gerenciamento sistemático e efetivo de variabilidades em LPrSs. Dois aspectos são analisados nesse critério: (i) os mecanismos oferecidos pela abordagem para especificação de variabilidades; e (ii) o suporte da abordagem para a derivação automática de processos a partir dos ativos de processos existentes (seleção de *features* e resolução de variabilidades).

5.3 Resultados do Estudo

Esta seção apresenta os resultados da modelagem das linhas de processos selecionadas utilizando o EPF *Composer* (Seção 5.3.1) e GenArch-P (Seção 5.3.2). Adicionalmente, também são apresentados e discutidos os resultados obtidos para as duas abordagens com relação aos critérios de comparação (Seção 5.3.3). Finalmente, algumas questões em aberto são apresentadas e discutidas (Seção 5.3.4).

5.3.1 Modelagem com EPF Composer

Engenharia de linha de processos. A definição das linhas de processos foi realizada tomando por base os *plugins* de método (*method plugins*) originais do EPF (Eclipse Foudation, 2010) para o OpenUP e Scrum. O primeiro passo da definição da LPrS foi a criação de um novo *plugin* de método para cada linha de processos. Logo após, foram criados os elementos de processo mandatórios. Estes elementos mandatórios foram organizados em um pacote de conteúdo denominado de “core” – representando o núcleo da LPrS. Para a definição dos elementos de processo relacionados a *features* opcionais e alternativas, foram criados pacotes de conteúdo específicos para cada *feature*. Para facilitar a organização, o tipo e o nome de cada *feature* foi usado para dar nome aos pacotes de conteúdo.

A Figura 22 apresenta um conjunto de pacotes de conteúdo criados para definir a linha de processos baseada no OpenUP. A definição da linha de processos baseada no Scrum foi diferente em dois aspectos, quais sejam: (i) não foram identificadas *features* de alto-nível, e (ii) o processo Scrum não impõe um fluxo de atividades específico. A execução de um processo Scrum é baseada em ciclos de trabalho, chamados de *sprints*, nos quais os requisitos de um produto de software (*product backlog*) são transformados em incrementos parciais no produto final. A análise de variabilidades encontrou elementos de processo opcionais e alternativos. Para acomodar a variabilidade restrita a um único elemento, foram criados os pacotes de conteúdo representando cada *feature*.

A Figura 23 apresenta as estruturas de pacotes de conteúdo criadas para abrigar os elementos variáveis do processo do Scrum. Cada pacote de conteúdo, além do elemento variável, também inclui as especializações dos elementos de processo mandatórios que irão se relacionar com o elemento variável.

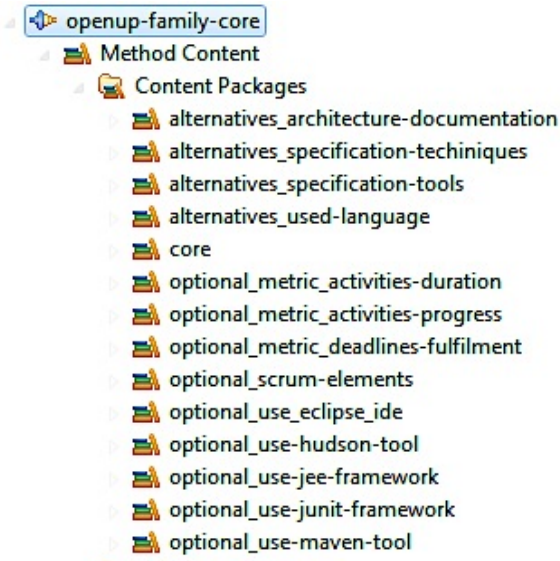


Figura 22. Pacotes de conteúdo para a linha de processos baseada no OpenUP

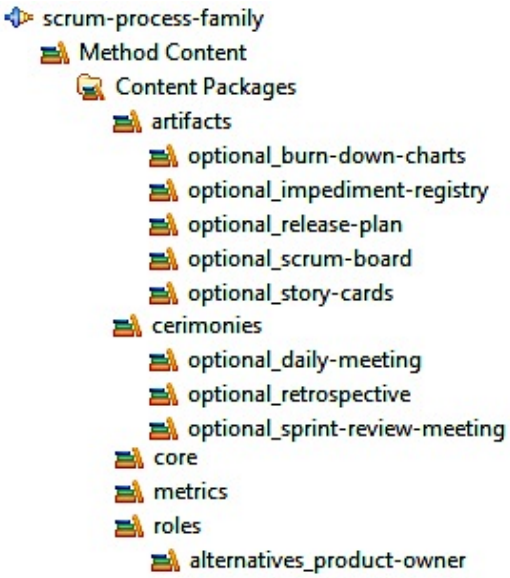


Figura 23. Pacotes de conteúdo para a linha de processos baseada no Scrum

A Figura 24 ilustra em detalhes um dos pacotes de conteúdo criados para a linha de processos baseada no Scrum, representando a cerimônia "*daily meeting*". Neste pacote de conteúdo, podem ser vistas a inclusão da atividade correspondente e a extensão dos elementos do núcleo do processo que passarão a se relacionar com esta nova atividade, são eles: (i) o Scrum *master* e a sua equipe que serão os realizadores da atividade e (ii) o artefato de *Sprint Backlog*, visto que o mesmo será usado como entrada para a referida atividade.

O fluxo de atividades da linha de processos foi especificado utilizando as estruturas de padrões de capacidade (*capability patterns*) providas pelo EPF. Um padrão

de capacidade serve para encapsular um fluxo de atividades, o qual pode ser composto, ou especializado, por outros padrão de capacidade. Inicialmente, foi definido o fluxo de atividades mandatório para a linha de processo – fluxo de atividades “core”. Um padrão de capacidade, denominado de “core”, foi criado para representar tal fluxo. A seguir, foram criados alguns padrões de capacidade para encapsular os incrementos ao fluxo de atividades “core” demandados por algumas *features*. Uma *feature* específica não necessariamente causa um incremento no fluxo de atividades mandatório, podendo apenas definir formas diferentes de realizar tais atividades. Os nomes das *features* foram usados para dar nome aos padrões de capacidade criados. Devido ao fato da linha de processos do Scrum não impor um fluxo de atividades, não foram utilizados padrões de capacidade nessa LPrS.

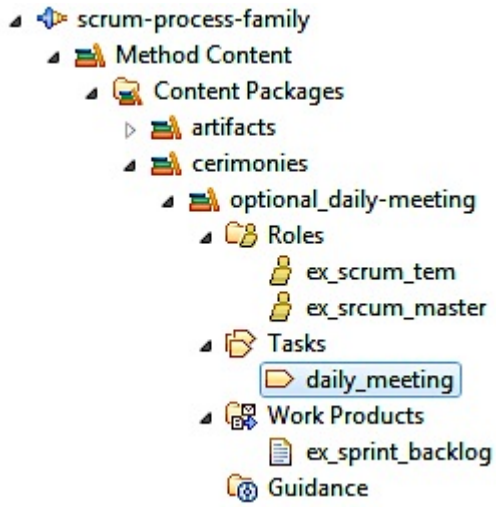


Figura 24. Exemplo de um pacote de conteúdo da LPrS baseada no Scrum

As variabilidades de granularidade fina, nos elementos do processo Scrum, foram modularizadas pelo EPF por meio da definição de elementos especializados que contribuem, estendem ou substituem os elementos originais. Por exemplo, a adição das descrições de novos tipos de itens de trabalho (*work items*) do artefato de *product backlog* foi realizada por meio da especialização do elemento de processo correspondente com o mecanismo de variabilidade “*contributes*”. O elemento especializado adiciona texto à descrição principal do *product backlog*, descrevendo um novo tipo de item de trabalho. Nesse exemplo, um outro elemento é afetado pela inclusão de um novo tipo de item de trabalho, justamente a tarefa que gera e refina o *product backlog* deve incluir um novo passo que instrui como tal tipo de item de trabalho é identificado e como tem que ser o registro do mesmo.

A Figura 25 apresenta o conjunto de padrões de capacidade criados para a linha de processos baseada no OpenUP. Cada um deles usa o mecanismo de variabilidade "*contributes*" para realizar um incremento ao "core". O mecanismo de variabilidade "*contributes*" foi o único utilizado nesse caso, porque o conteúdo do fluxo de atividades do "core" precisa ser mantido e os incrementos demandados pelas *features* devem ser adicionados ao mesmo. A Tabela 14 apresenta um resumo de como os mecanismos EPF foram utilizados para implementar cada tipo de *feature*.

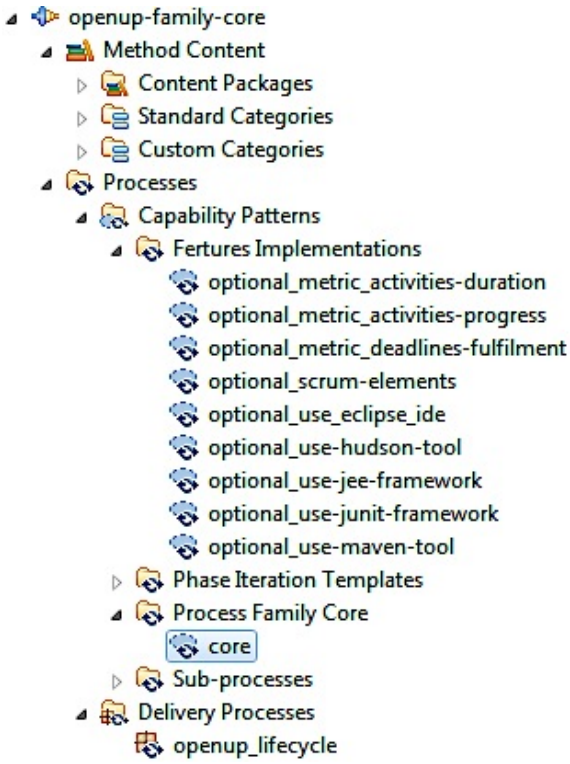


Figura 25. Padrões de capacidade para a LPrS baseada no OpenUP

Tabela 14. Mecanismos EPF usados para o gerenciamento das variabilidades

Tipo de *feature*	**Mecanismo EPF utilizado**
Alternativa	(1) Estrutura hierárquica de pacotes de conteúdo (2) Elementos de processo representando cada alternativa (3) Padrões de capacidade necessários, encapsulando os incrementos no fluxo de atividades do "core", hierarquicamente estruturado
Opcional	(1) Estrutura de pacote de conteúdo (2) Elementos de processo relacionados (3) Padrão de capacidade encapsulando os incrementos ao fluxo de atividades do "core", se necessário
OR-*feature*	(1) Estrutura hierárquica de pacotes de conteúdo (2) Elementos de processo associados a cada opção (3) Padrões de capacidade necessários, encapsulando os incrementos no fluxo de atividades do "core", hierarquicamente estruturado

Derivação de processos. Para permitir a derivação de processos a partir da LPrS, o EPF *Composer* oferece a funcionalidade de definição de configuração. Na

definição de uma configuração para um processo, o engenheiro de processos seleciona visualmente as estruturas que deseja que façam parte do processo sendo derivado. Contudo, o EPF *Composer* não provê suporte à representação das variabilidades de forma explícita, como no modelo de *features*. Também não provê suporte ao mapeamento explícito entre *features* específicas e elementos de processo associados – conhecimento de configuração. Devido a esse fato, os engenheiros de processo precisam saber quais seleções precisam ser feitas para refletir as *features* desejadas. A Figura 26 apresenta o exemplo de uma seleção de *feature* alternativa, ilustrando como é feita a seleção de uma alternativa específica. Nesse exemplo, é selecionada a técnica de casos de uso (Cockburn, 2001) como a técnica de especificação de requisitos. A Figura 27 apresenta um exemplo de seleção de *features* opcionais.

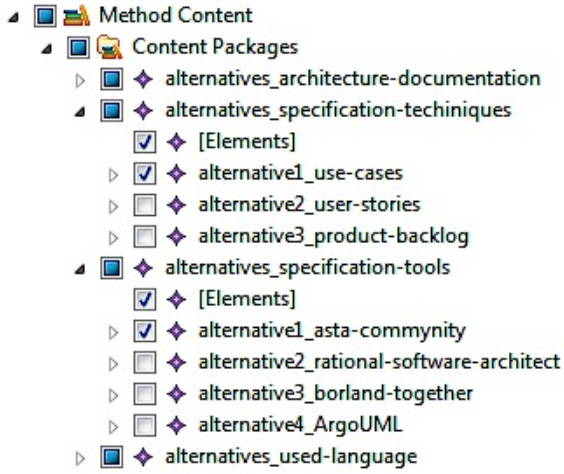


Figura 26. Seleção de feature alternativa da LPrS baseada no OpenUP

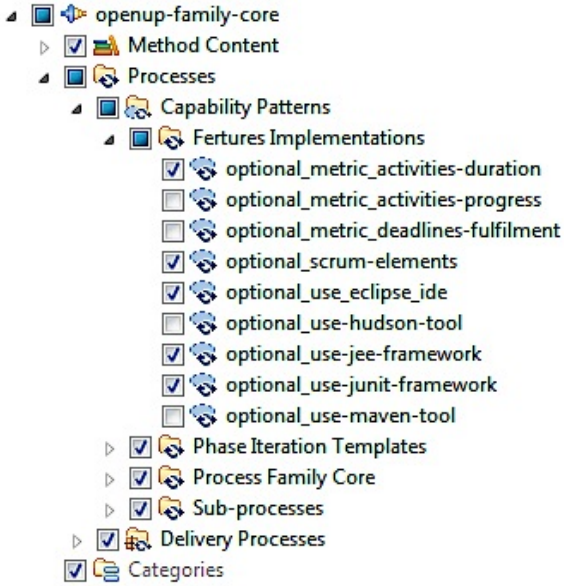


Figura 27. Seleção de features opcionais da LPrS baseada no OpenUP

A Figura 27 mostra a seleção de cinco das nove *features* opcionais disponíveis, quais sejam: (i) a aplicação da métrica para avaliar a duração das atividades principais; (ii) a inclusão dos elementos de processo oriundos do Scrum; (iii) a utilização da IDE Eclipse; e assim por diante. Após a definição da configuração, o EPF gera um *site Web* de acordo com a configuração definida, e os correspondentes fluxos de atividades e elementos de processo (atividades, passos, papéis, entre outros).

5.3.2 Modelagem com GenArch-P

A definição de LPrS utilizando o GenArch-P requer, inicialmente, a definição de uma especificação de processo de software contendo: (i) todos os elementos de processo mandatórios que representam o núcleo da linha de processos; e (ii) os elementos de processo opcionais e alternativos que estão associados a *features* de alto-nível.

Engenharia de linha de processos. Inicialmente, um novo projeto GenArch-P foi criado para cada linha de processos, contendo as especificações de processo representando as famílias de processo OpenUP e Scrum. Após esse passo, a ferramenta GenArch-P foi utilizada para gerar modelos de processo simplificados, abstraindo a estrutura das especificações de processo de software utilizadas.

A Figura 28 apresenta o modelo simplificado do GenArch-P para a linha de processos baseada no OpenUP, no qual os elementos estão hierarquicamente distribuídos para compor uma visão geral do processo. Para associar elementos de processo à *features* específicas, os engenheiros de processo devem anotar tais elementos com expressões de *features*, especificando dessa forma o conhecimento de configuração. Mais de uma *feature* pode ser associada a um dado elemento de processo. Durante a atividade de anotação dos elementos de processo, os engenheiros de processo podem também especificar restrições e dependências entre *features*. A modelagem da linha de processos baseada no Scrum usando a versão do GenArch-P disponibilizada para o estudo falhou em modularizar *features* de granularidade fina (que acontecem internamente aos elementos). A abstração proporcionada pelo modelo de processo simplificado, não permitiu a anotação de fragmentos do conteúdo de elementos de processo. Na versão atual do GenArch-P, a menor granularidade passível de anotação são os elementos de processo como um todo, e não os atributos dos mesmos.

A Figura 28 também ilustra os elementos de processo anotados com as respectivas associações às variabilidades da linha de processos. Após as anotações, esse modelo passa a representar o conhecimento de configuração – o qual especifica o mapeamento entre *features* e elementos de processo. Por exemplo, a *feature* que representa a documentação arquitetural segundo o projeto ágil (Ambler, 2011) está associada aos seguintes elementos de processo: (i) prática de teste concorrente (*concurrent testing*); (ii) prática de integração contínua (*continuous integration*); (iii) prática de desenvolvimento orientado à testes (*test-driven development*); (iv) papel de testador (*tester*); além de outros elementos de processo não presentes na Figura 28.

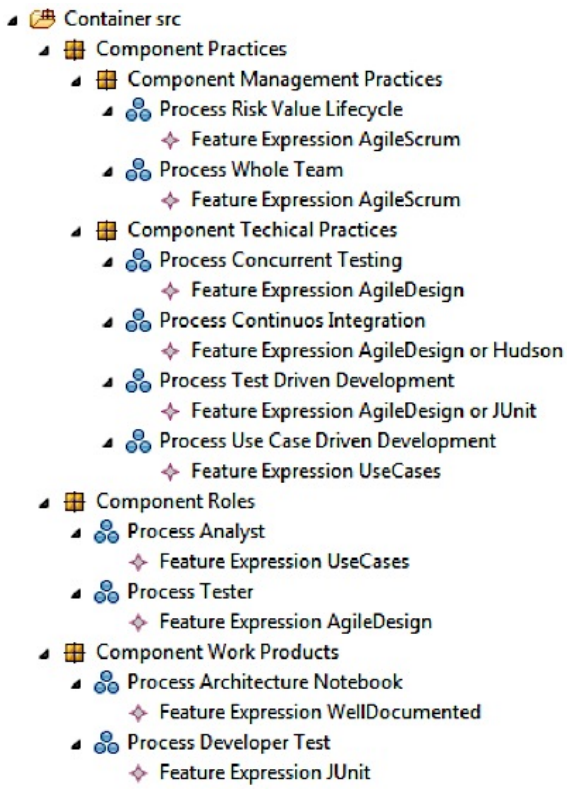


Figura 28. Fragmento do modelo simplificado de processo referente a linha de processos baseada no OpenUP com anotações de features

A Figura 29 e a Figura 30 apresentam a visão parcial dos modelos de *features* gerados para as linhas de processos do OpenUP e Scrum, respectivamente. Eles representam as diferentes variabilidades associadas aos elementos de processo e as suas respectivas restrições. O modelo de *features* e o modelo de processo anotado orientam a ferramenta durante a derivação automática de processos.

Derivação de processos. O primeiro passo para a derivação automática de processos utilizando o GenArch-P é a criação de uma nova configuração de *features* a

partir do modelo de *features*. Uma configuração de *features* permite que os engenheiros de processo possam selecionar as *features* desejadas para uma nova instância de processo de software.

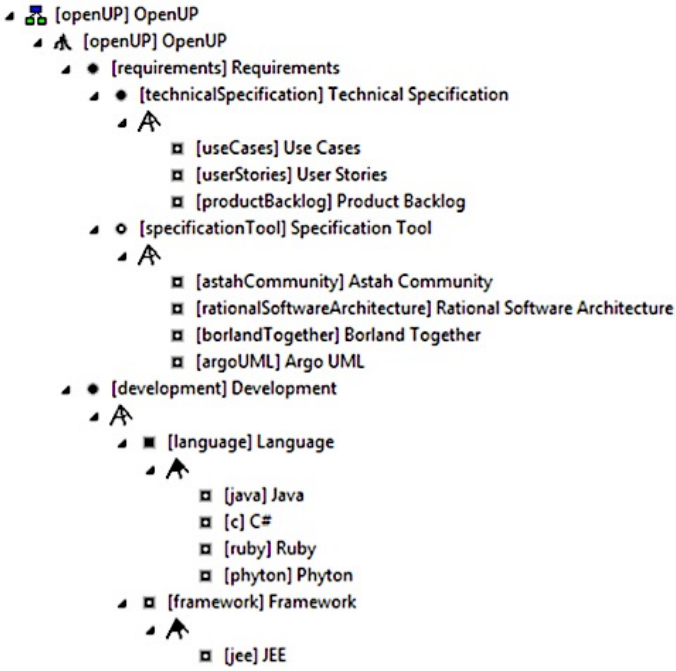


Figura 29. Fragmento do modelo de features para a família de processos OpenUP

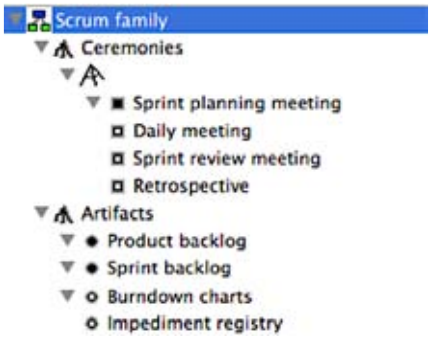


Figura 30. Fragmento do modelo de features para a família de processos Scrum

No GenArch-P, o modelo de *features* é gerado pelo *plug-in* FMP (Generative Software Development Lab, 2012), o qual torna a seleção de *features* dinâmica, visto que as restrições associadas às *features* são analisadas no momento em que as seleções são realizadas. Esse fato pode ocasionar que a seleção de uma *feature* específica implique na adição ou remoção de outras *features*, além de que algumas *features* podem se tornar não selecionáveis. Após a seleção de *features*, o GenArch-P automaticamente deriva a especificação de processo de software correspondente – contendo os elementos de processo associados às *features* selecionadas. A especificação de processo gerada pode então ser editada e processada pelos engenheiros de processo responsáveis.

5.3.3 Análise dos Critérios

Nesta seção, são apresentadas análises dos resultados da aplicação das abordagens na modelagem das linhas de processos selecionadas para o estudo, segundo a perspectiva dos critérios definidos para a análise comparativa, e as conclusões extraídas desta análise.

Modularidade. Usando o EPF *Composer* é possível definir elementos de processo de software em pacotes de conteúdo específicos, como ilustrados na Figura 22 e na Figura 23. Também é possível agrupar fragmentos de fluxos de atividades em padrões de capacidade. Estruturas essas que podem ser incluídas em diferentes configurações – que determinam como irão ser formadas as instâncias da linha de processos (ver Figura 26 e Figura 27). Estes mecanismos de modularização não permitem a associação explícita dos elementos neles contidos à *features* específicas, mas eles permitem o agrupamento dos elementos associados à uma dada *feature*. Portanto, conclui-se que o EPF *Composer* provê um suporte útil para a modularização dos elementos de processo relacionados à uma *feature* específica. A Tabela 15 resume os resultados dos mecanismos de modularização do EPF *Composer* usados na modelagem das LPrSs alvos do estudo.

Tabela 15. Resultados quantitativos da utilização do EPF Composer

Mecanismos do EPF	**Linha de processos OpenUP**	**Linha de processos Scrum**
Elementos de processo base (mandatórios)	75	6
Elementos de processo em pacotes de conteúdo (granularidade grossa)	227	32
Elementos de processo com variabilidade de conteúdo (granularidade fina)	117	19
Padrões de capacidade	9	0

O GenArch-P não provê mecanismos de modularização explícitos para os elementos de processo associados a *features* específicas. Os engenheiros de processo apenas interagem com uma abstração de especificação de processo – o modelo de processo simplificado, o qual representa uma visão geral da estrutura do processo e seus elementos. Trabalhando com esse modelo, o engenheiro de processo abstrai os detalhes e a organização de baixo nível da especificação do processo. Dessa forma, eles não precisam ter conhecimento sobre o baixo nível da organização do processo, para anotar expressões de *features* em elementos de processos e derivar instâncias da linha de

processos. A funcionalidade do GenArch-P independe da especificação original do processo de software e como esta esteja previamente organizada (bem modularizada ou não). Em consequência disso, conclui-se que essa abordagem possui um fraco suporte para modularização. A Tabela 16 resume os resultados dos mecanismos de modularização do GenArch-P usados na modelagem das LPrSs alvos do estudo.

Tabela 16. Resultados quantitativos da utilização do GenArch-P

Mecanismos do GenArch-P	**Linha de processos OpenUP**	**Linha de processos Scrum**
Elementos de processo com anotações	152	25
Elementos de processo sem anotações	75	6

Comparando os resultados quantitativos para os mecanismos de modularização, pode ser observado que embora o EPF *Composer* ofereça mecanismos explícitos para modularizar as variabilidades das linhas de processos, o GenArch-P requer a anotação de um número reduzido de elementos de processo (152) comparado ao alto número de elementos de processo representando variabilidades de granularidade grossa (227) e fina (117) no EPF. No EPF *Composer*, o alto número de elementos é justificado pelo fato que elementos mandatórios de processo podem ser especializados em mais pacotes de conteúdo, especificando diferentes variabilidades.

Rastreabilidade. Usando o EPF *Composer*, a rastreabilidade é alcançada por meio da organização dos elementos em estruturas específicas: pacotes de conteúdo e padrões de capacidade. Para facilitar a rastreabilidade, os nomes dessas estruturas devem refletir os nomes das *features* correspondentes (ver Figura 22, Figura 23 e Figura 25). Mesmo que o EPF *Composer* não ofereça um mecanismo explícito para rastrear elementos de processo a partir das *features*, a organização adequada destes elementos permite o rastreamento da linha de processos. Conclui-se, portanto, que o EPF *Composer* possui um suporte parcial para rastreabilidade.

Por outro lado, usando o GenArch-P, os engenheiros de processo podem visualizar todos os mapeamentos entre *features* e elementos de processo por meio do modelo simplificado de processo, devidamente anotado. O GenArch-P permite encontrar os elementos de processo associados à *features* específicas, por meio da navegação nesse modelo. Por essa característica, pode-se dizer que o GenArch-P oferece um bom suporte para rastreabilidade.

Detecção de erros. O EPF *Composer* não oferece nenhum mecanismo explícito para a detecção de erros em processos derivados da linha de processos. Por exemplo, é considerado um erro a inclusão de duas (ou mais) *features* mutuamente exclusivas – como as *features* alternativas. A única checagem de consistência realizada pelo EPF *Composer* é a checagem de dependência durante o processo de configuração. Nessa checagem de dependência é analisado se todos os *plugins* de método necessários para uma configuração foram corretamente selecionados. Não há nenhum mecanismo adicional de checagem de consistência das seleções realizadas na configuração da linha de processos. Por exemplo, na definição de uma configuração, é possível selecionar duas ou mais alternativas para uma *feature* onde apenas uma deveria ser selecionada. A abordagem não provê suporte em garantir a semântica associada aos diferentes tipos de *features*. Outro exemplo dessa limitação é que a seleção da *feature* opcional que representa a utilização do *framework* Java EE não está atrelada a seleção da *feature* alternativa relativa à escolha da linguagem de programação Java – a ser utilizada durante a execução do processo. Isso significa que o EPF não pode representar restrições entre *features*. Por esse fato, conclui-se que o EPF *Composer* oferece um fraco suporte para a detecção de erros.

O GenArch-P também não provê um mecanismo explícito para a detecção de erros para a linha de processos, nem para as instâncias de processo derivadas. Mas, durante a anotação dos elementos com expressões de *feature*, é possível a definição de restrições associadas a cada uma das *features*. Tais restrições permitem garantir que elas serão respeitadas durante o processo de seleção de *features* em uma nova configuração de *features* (pelo *plugin* FMP), tais como relacionamentos “requer” ou “exclui”. Por tais fatos, conclui-se que o GenArch-P possui um suporte parcial para a detecção de erros de consistência.

Granularidade. Em termos de granularidade dos elementos de processo associados a *features* específicas, o EPF *Composer* permite trabalhar com diferentes níveis de granularidade, mesmo com algumas pequenas limitações. Os mecanismos de pacotes de conteúdo e padrões de capacidade dão suporte a granularidade grossa, por meio do agrupamento de elementos de processo correlacionados. Os mecanismos de variabilidade de conteúdo do EPF dão suporte a granularidade fina, alterando a estrutura de elementos de processo das seguintes formas: (i) adicionando conteúdo aos atributos de um elemento de processo – base (com a variabilidade de conteúdo “*contributes*”); (ii)

criando especializações de um elemento de processo, possivelmente redefinindo alguns dos seus atributos (com a variabilidade de conteúdo "*extends*"); (iii) criando um substituto para um elemento de processo (com a variabilidade de conteúdo "*replace*"); e (iv) criando um substituto para um elemento de processo, que herda alguns dos seus atributos (com a variabilidade de conteúdo "*extends and replace*"). A pequena limitação citada anteriormente é em função das alterações em um elemento de processo estarem condicionadas às formas proporcionadas pelos mecanismos de variabilidade de conteúdo. Dessa forma, pode-se dizer que o EPF *Composer* possui um bom suporte para lidar com elementos de processo com granularidades fina e grossa. A Tabela 17 apresenta uma visão geral da granularidade de *features* para cada uma das linhas de processos modeladas, além de um resumo dos mecanismos EPF utilizados para modularizar as mesmas. Tal resumo de estratégias pode servir de diretriz para lidar com esses tipos de variabilidade.

Tabela 17. Resultados da modelagem de granularidade com o EPF Composer

	Linha de processos OpenUP	**Linha de processos Scrum**
***Features* de granularidade fina**	0	25
***Features* de granularidade grossa**	22	0

Já o GenArch-P restringiu a granularidade do que pode ser associado a *features* específicas no nível dos elementos apresentados no modelo de processo simplificado. A primeira atividade da abordagem proposta pelo GenArch-P é a análise da especificação original do processo de software para a geração do modelo de processo simplificado, o qual modela a estrutura do processo. O modelo de processo simplificado é criado segundo um meta-modelo genérico, o qual também possibilita a anotação dos elementos de processo, associando os mesmos a *features* específicas. A implementação do GenArch-P utilizada no estudo oferece suporte apenas para a anotação de elementos de processos, como atividades e tarefas, mas não possibilita a anotação de atributos desses elementos. Considerando tais fatos, conclui-se que o GenArch-P atualmente provê um suporte parcial para lidar com diferentes granularidades de elementos de processo. A Tabela 18 apresenta um resumo dos resultados quantitativos da modelagem de *features* de diferentes granularidades com o GenArch-P. Como observado anteriormente, apenas um nível de granularidade foi considerado para o GenArch-P – o nível de elemento de processo.

Tabela 18. Resultados da modelagem de granularidade com o GenArh-P

	Linha de processos OpenUP	**Linha de processos Scrum**
***Features* associadas a propriedades de elementos ou conjunto de elementos de processo**	-	-
***Features* associadas a elementos de processo**	152	25

Comparando os resultados da Tabela 17 e Tabela 18, é possível concluir que os mecanismos de modularização do EPF também ajudam na modelagem de *features* de granularidade grossa. O trabalho com o GenArch-P é mais repetitivo, visto que cada elemento de processo associado à uma dada *feature* precisa ser anotado individualmente. A modelagem da linha de processos baseada no Scrum mostrou que se o número *features* de granularidade fina for muito maior que o número de *features* de granularidade grossa, a quantidade de esforço em ambas abordagens é equivalente.

Uniformidade. Por trabalhar exclusivamente com o meta-modelo UMA (*Unified Method Architecture*), o EPF *Composer* não oferece suporte para outras formas de especificação de processos de software que não seja o UMA. Esse tipo de abordagem possui as suas vantagens, pois pode explorar todos os pontos fortes da linguagem de especificação de processos utilizada. A desvantagem é que a ferramenta condiciona o uso de uma determinada linguagem de especificação de processos de software. Com base em tais observações, conclui-se que o EPF *Composer* não oferece suporte uniforme a diferentes formas de especificação de processos de software.

Por outro lado, o GenArch-P trabalha com um modelo de processo simplificado que abstrai a linguagem de especificação do processo. A representação da estrutura do processo e seus elementos não possui qualquer dependência com a forma na qual o processo é especificado. É requerida apenas uma extensão para a ferramenta que implemente a funcionalidade de importação, que permite a tradução de uma especificação de processo de software para o modelo de processo simplificado proposto do GenArch-P. Conclui-se, portanto, que o GenArch-P oferece suporte ao critério de uniformidade.

Adoção. O EPF *Composer* introduz vários conceitos e mecanismos que precisam ser conhecidos pelos engenheiros de processo. Como exemplos de tais conceitos, temos: conteúdo de método, pacotes de conteúdo e padrões de capacidade.

Adicionalmente, os engenheiros de processo também precisam entender os mecanismos de variabilidade para lidar com a definição de variabilidades de processos de software. Por fim, mas não menos importante, a atividade de definição de uma configuração de processo também requer o entendimento das funcionalidades de customização e composição de elementos de processo. Além disso, o conhecimento de configuração, necessário quando um engenheiro de processos está definindo uma nova instância de um processo, é de única responsabilidade do mesmo. Embora o EPF ofereça suporte ferramental a todos esses conceitos e mecanismos, há uma complexidade inerente envolvida na adoção de todos eles para promover a gerência automática de variabilidades em LPrSs. Por forçar os engenheiros de processo a conhecer tal conjunto, não trivial, de conceitos, funcionalidades e mecanismos, conclui-se que o EPF *Composer* oferece um fraco suporte para o critério de adoção. Nosso estudo contribui para atenuar esse problema por meio de uma clara indicação de quais mecanismos podem ser utilizados para modelar diferentes tipos de variabilidade (Tabela 14).

A ferramenta GenArch-P automatiza boa parte das tarefas envolvidas na definição de uma LPrS. Apenas algumas seleções de opções de menu são necessárias para acionar as funcionalidades principais da ferramenta. Inicialmente, a ferramenta analisa uma especificação de processo de software existente, e gera um modelo de processo simplificado. Após isso, os engenheiros de processo anotam os elementos de processo que representam variabilidades, associando os mesmos a expressões de *features*. Por fim, os engenheiros de processo utilizam a ferramenta para gerar uma instância de processo da LPrS. Nesse passo final, é apenas necessária a criação de uma nova configuração de *features*, realizar a seleção das *features* desejadas e o GenArch-P automaticamente gera uma instância de processo que reflete as escolhas realizadas. A abordagem GenArch-P requer apenas que o usuário conheça as variabilidades da LPrS, não necessitando de conhecimentos adicionais, como por exemplo: as técnicas e mecanismos composicionais. Oferecendo desta forma, um bom suporte à adoção.

Gerência sistemática de variabilidades. Usando o EPF *Composer*, os engenheiros de processos não podem especificar as variabilidades existentes usando modelos explícitos, como o modelo de *features*, e consequentemente não podem associar elementos de processo a tais *features*. O gerenciamento de variabilidades no EPF *Composer* é refletido apenas na modularização e agrupamento de elementos de processo que representam cada uma das variabilidades de uma linha de processos. Por

outro lado, o GenArch-P permite a gerência explícita de *features* durante todas as fases da modelagem de uma LPrS. Utilizando essa ferramenta são possíveis ações, tais como: (i) representação de *features* com suas respectivas restrições; (ii) definição do mapeamento entre *features* e elementos de processo relacionados; e (iii) derivação automática de processos baseada em uma seleção de *features*. Assim, é possível concluir que o GenArch-P oferece suporte à gerência sistemática de variabilidades.

A Figura 31 apresenta um visão geral dos resultados obtidos com o estudo. Segundo esta forma de visualização dos resultados, os três níveis de conformidade para os critérios observados: (i) fraco ou nenhum suporte, (ii) suporte parcial e (iii) bom suporte foram representados por níveis nas escalas relativas à cada critério. O nível mais externo representa um “bom suporte”. O nível intermediário representa um “suporte parcial”. Por fim, o nível mais interno (diferente da origem) representa um “fraco ou nenhum suporte”.

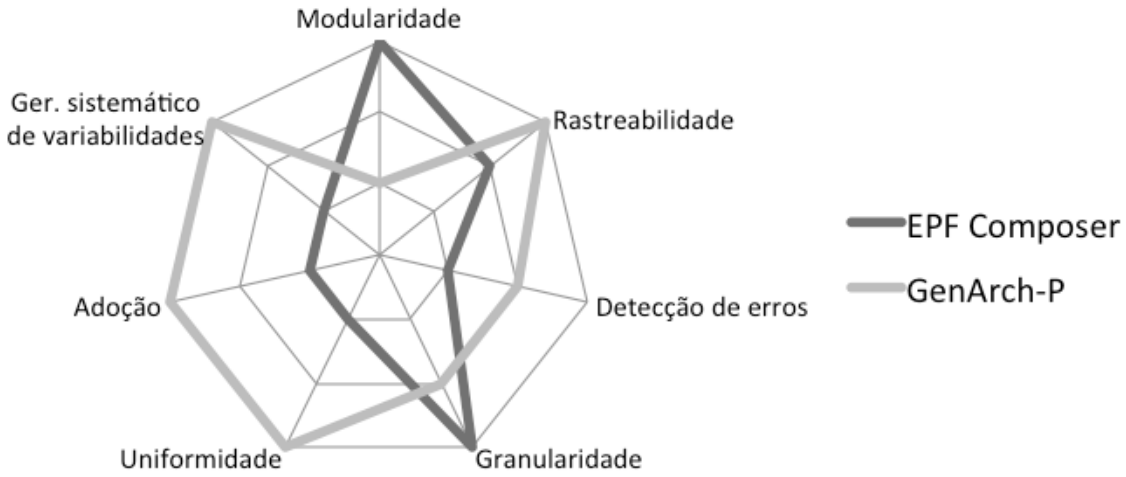


Figura 31. Visão geral dos resultados do estudo comparativo

5.4 Questões em Aberto e Discussões

Os resultados do estudo permitiram a observação de limitações existentes na definição de LPrSs nas abordagens investigadas. Nesta seção, são apresentadas tais limitações e possíveis soluções para questões em aberto. As limitações principais do EPF *Composer* identificadas durante a realização do estudo foram: (i) o fato do EPF não prover suporte ou gerenciamento sistemático de variabilidades; e (ii) a complexidade de utilização, em função da diversidade e complexidade dos conceitos e mecanismos que permitiram a modelagem da linha de processos. Ficou claro que o EPF *Composer* não foi concebido para a implementação de LPrSs, o que é compreensível pois esse não é

um dos objetivos do *framework* nem da ferramenta. Este fato explica o fraco suporte para a modelagem de *features*, que afeta negativamente a gerência sistemática de variabilidades da abordagem. Contudo, o estudo ajudou a esclarecer que mesmo com essas limitações, o EPF *Composer* pode ser usado para modelar LPrSs. A introdução da modelagem e o explícito mapeamento de *features* para elementos de processo podem contribuir diretamente para o uso da ferramenta para a engenharia de LPrSs, dessa forma motivando o seu uso pela comunidade de processos de software. O problema da complexidade de uso do EPF *Composer* pode ser superado com diretrizes que ajudem a orientar em como as diferentes variabilidades de processos podem ser modeladas usando suas abstrações e mecanismos. Nosso estudo também contribuiu nesse sentido, proporcionando um conjunto preliminar de diretrizes para a modularização de LPrSs com o EPF *Composer* (Seção 5.3.1). Mesmo que a curva de aprendizado da ferramenta seja acentuada no início, com o passar do tempo o esforço pode ser compensado.

De forma similar, a ferramenta GenArch-P também apresentou algumas limitações durante a realização do estudo comparativo. As limitações mais importantes da versão utilizada foram: (i) a menor granularidade possível de ser anotada com variabilidades foi a de elementos de processo; (ii) não ajudou em uma melhor modularização da especificação original do processo de software; (iii) a versão do GenArch-P utilizada no estudo só estava preparada para manipular modelos de acordo com um único meta-modelo para especificação de processos de software. O problema da modelagem de variabilidade de granularidade fina usando o GenArch-P pode ser resolvido com a melhoria da análise da especificação de processo original, possibilitando que sejam representados no modelo de processo simplificado elementos de menor granularidade. Por exemplo, uma vez que atributos de um elemento de processo, como uma tarefa, possam também ser vistos como elementos de menor granularidade, esses poderão ser anotados com expressões de *feature*. O problema de modularização é uma característica intrínseca da abordagem anotativa, compensada com a utilização de ferramentas de análise de variabilidades e detecção de erros. O problema do suporte a um único meta-modelo de processo pode ser facilmente resolvido com o desenvolvimento e acoplamento de novos módulos analisadores. Módulos esses responsáveis por analisar uma especificação de processo de software específica e transformá-la em uma instância do modelo de processo simplificado. O GenArch-P possui uma utilização simplificada e automatiza muitas das tarefas necessárias para a

implementação de uma LPrS. Além disso, uma das vantagens da ferramenta GenArch-P é o fato da mesma abstrair e não depender de uma linguagem de especificação de processos de software.

5.5 Conclusões do Estudo

Este capítulo apresentou os resultados de um estudo comparativo de modelagem de linhas de processo de software usando as abordagens composicionais e anotativas. As comparações foram baseadas em critérios de comparação previamente adotados na análise de implementações de linhas de produtos de software (Kästner, 2010) (Kästner & Apel, 2008b) (Kästner et al., 2010). Duas modernas e recentes propostas de abordagens, denominadas de EPF *Composer* e GenArch-P, foram selecionadas para especificar LPrSs baseadas no OpenUP e Scrum. Analisando-se os resultados finais do estudo, constatou-se que o GenArch-P apresentou melhores resultados que o EPF *Composer*, com relação a maioria dos critérios avaliados, quais sejam: (i) rastreabilidade, (ii) detecção de erros, (iii) uniformidade, (iv) adoção e (v) gerenciamento sistemático de variabilidades. É importante enfatizar que, no que diz respeito aos critérios adotados, ainda se faz necessária a realização de novos estudos de caso e experimentos para o melhor entendimento da efetividade das abordagens de modelagem de LPrSs. Os resultados obtidos não podem ser generalizados dado o escopo restrito do estudo. Há a necessidade da realização de mais estudos empíricos para alcançar mais evidências dos benefícios da abordagem composicional.

O estudo mostrou que as abordagens composicional e anotativa possuem as suas próprias vantagens e limitações quando utilizadas na modelagem de LPrSs. Tal fato levou à indagação “como combinar as vantagens dessas duas abordagens?” e a vislumbrar a possibilidade de integração das duas abordagens. A modularidade da abordagem composicional poderia ser somada à flexibilidade da anotação de *features* da abordagem anotativa. Proporcionando, por exemplo, a anotação das estruturas modulares da abordagem composicional. Adicionalmente, deve ser enfatizada a necessidade de melhorar as funcionalidades de detecção de erros de consistência nas abordagens anotativas existentes.

6 Estudo Comparativo Quantitativo das Abordagens Composicional e Anotativa para LPrSs

O estudo comparativo qualitativo apresentado no capítulo anterior, onde as abordagens, composicional do EPF *Composer* e anotativa do GenArch-P, foram analisadas e comparadas, apontou para importantes evidências. Nesse estudo, as abordagens em questão foram avaliadas segundo critérios relativos à experiência da modelagem de uma LPrS. Contudo, as conclusões desse estudo não puderam ser melhor embasadas pela falta de dados quantitativos para dar suporte a tais conclusões. Para suprir essa lacuna, foi idealizada a realização de um novo estudo comparativo. Desta vez, produzindo dados quantitativos, que permitissem conclusões melhor embasadas sobre os pontos fortes e fracos das abordagens do EPF *Composer* e do GenArch-P. Dados quantitativos que pudessem também servir de evidências iniciais para a generalização das conclusões para as abordagens composicional e anotativa.

A definição deste estudo tomou por base a metodologia GQM (Basili et al., 1994) e também buscou orientações em trabalhos anteriores no contexto de linhas de produtos de software (Figueiredo & al, 2008) (Kästner & Apel, 2008b) (Kästner et al., 2010) (Bonifácio & Borba, 2009). Durante a execução do mesmo, foi modelada uma evolução da LPrS utilizada no estudo anterior com a abordagem EPF *Composer* (Eclipse Foundation, 2012) e com a abordagem GenArch-P (Aleixo et al., 2010a) (Aleixo et al., 2010b), as quais representam abordagens concretas das técnicas composicional e anotativa, respectivamente. Após tal modelagem, foram quantificadas métricas usadas para avaliar atributos relevantes da especificação de LPrSs, tais como: modularidade, tamanho e complexidade. Por fim, os resultados obtidos para cada uma das métricas foram analisados e discutidos. Discussões essas que permitiram que os resultados do estudo anterior fossem confrontados pelos dados quantitativos obtidos nesse estudo (Aleixo et al., 2012a) (Aleixo et al., 2012b).

A sequência desse capítulo está organizada da seguinte forma. A Seção 6.1 apresenta a configuração deste estudo comparativo. A Seção 6.2 apresenta detalhes sobre os procedimentos de modelagem da LPrS alvo, pelas abordagens do EPF *Composer* e do GenArch-P. A Seção 6.3 apresenta os resultados do estudo, com a análise de cada um dos atributos avaliados. A Seção 6.4 apresenta discussões adicionais sobre os resultados obtidos. A Seção 6.5 apresenta as ameaças à validade do estudo que

foram identificadas e como essas foram tratadas. A Seção 6.6 apresenta e discute alguns trabalhos relacionados. E por fim, a Seção 6.7 apresenta as conclusões e trabalhos futuros derivados desse estudo.

6.1 Configurações do Estudo

Esta seção apresenta as configurações do estudo, em termos de seu: objetivo principal e questões de pesquisa, fases do estudo e procedimentos de avaliação, o conjunto de métricas adotado, as abordagens de LPrS sob avaliação, e a linha de processos alvo modelada com as abordagens sendo investigadas.

6.1.1 Objetivo do Estudo e Questões de Pesquisa

A metodologia GQM (Basili et al., 1994) foi utilizada para estruturar o estudo. Inicialmente, foi definido o objetivo principal do estudo, como: “comparar (propósito) as técnicas composicional e anotativa (alvo) na modelagem de variabilidades em uma LPrS (processo), na perspectiva dos usuários dessas abordagens (ponto de vista)”. Para atingir esse objetivo, foram definidas três questões de pesquisa (QP):

(i) QP1 – qual das técnicas investigadas – composicional ou anotativa – provê um melhor suporte para a modularização de LPrSs?

(ii) QP2 – qual das técnicas investigadas pode produzir uma especificação de LPrS mais concisa, com respeito ao número total de elementos de processo?

(iii) QP3 – qual das técnicas investigadas possui aplicação mais complexa, durante a especificação de uma LPrS, no que diz respeito ao número de mecanismos de gerenciamento de variabilidades e na quantidade de aplicações desses mecanismos?

Para ajudar a responder tais questões, foi adaptado um conjunto de métricas para quantificar os atributos de modularidade, tamanho e complexidade, respectivamente. O atributo de modularidade foi medido por meio da quantidade de “grupos subsequentes” de elementos de processo associados à uma *feature* específica. O atributo de tamanho, por outro lado, foi quantificado por meio da contagem do número de elementos de processos usados para especificar a LPrS. Finalmente, o atributo de complexidade foi quantificado por meio do número de mecanismos de gerenciamento de variabilidades

proposto por cada abordagem, e pela quantidade de aplicações desses mecanismos na modelagem da LPrS alvo. Tais métricas são discutidas em mais detalhes na Seção 6.1.3.

6.1.2 Fases do Estudo e Procedimentos de Avaliação

O estudo foi organizado em três fases: (i) modelagem da LPrS alvo utilizando as abordagens investigadas; (ii) coleta das métricas; e (iii) análise, discussões e conclusões a respeito dos resultados das métricas. A definição da LPrS alvo considerou um estudo anterior realizado pelo nosso grupo de pesquisa (Aleixo et al., 2012a). A modelagem da LPrS alvo utilizando as abordagens investigadas pode ser considerada uma evolução dos resultados alcançados pelo estudo comparativo qualitativo anteriormente realizado (Aleixo et al., 2012a). Para a avaliação dos resultados obtidos, cada uma das métricas selecionadas foi quantificada visando identificar: (i) qual abordagem obteve melhores e piores resultados considerando os atributos sob avaliação; e (ii) quais foram as causas e razões identificadas para tais resultados. Por fim, os resultados obtidos e as respectivas conclusões foram confrontados com os resultados obtidos no estudo anterior.

6.1.3 Métricas Adotadas no Estudo

Após o término do estudo comparativo qualitativo anteriormente realizado (Aleixo et al., 2012a), percebeu-se a falta de resultados quantitativos para proporcionar uma maior confiança na análise dos resultados obtidos. Essa foi a principal motivação para o desenvolvimento deste estudo comparativo quantitativo. Visando embasar as respostas às questões de pesquisa levantadas no estudo, foi adaptado um conjunto de métricas. Prioritariamente, buscou-se a adaptação de métricas já utilizadas em outros estudos, especificamente foram considerados estudos empíricos relacionados ao projeto e implementação de linhas de produtos de software (Figueiredo & al, 2008) (Kästner & Apel, 2008b) (Kästner, 2010) (Bonifácio & Borba, 2009). Tais métricas foram adaptadas ao contexto de LPrS, e abordaram três importantes atributos da modelagem de LPrSs: (i) modularidade, (ii) tamanho, e (iii) complexidade.

O atributo de modularidade está preocupado com o uso de estruturas modulares visando o isolamento dos elementos de processo associados à uma *feature* de processo específica. A modularidade em uma LPrS é importante porque ela pode tornar mais fácil o gerenciamento de variabilidades. Se uma abordagem permite o agrupamento de elementos de processo associados às *features* de uma LPrS, utilizando para tal uma

quantidade menor de “estruturas modulares”, esta apresenta uma melhor modularidade. A menor quantidade de módulos (grupos de elementos de processos) pode facilitar o entendimento e gerenciamento dos mesmos, além de facilitar o acesso aos elementos de processo dispostos em tais módulos.

O atributo de tamanho está preocupado com o número de elementos de processo envolvidos na modelagem de uma LPrS. Este atributo é importante porque quanto maior uma especificação de LPrS, torna-se mais difícil o entendimento e gerenciamento desta especificação. Da mesma forma, quanto maior o número de elementos na especificação de uma LPrS, maior é o trabalho de um engenheiro de processo durante a modelagem da mesma.

O atributo de complexidade, no contexto desse estudo, está relacionado aos mecanismos usados para a especificação das variabilidades em processos de software. A utilização de uma maior quantidade de mecanismos para especificação de variabilidades pode dificultar a compreensão e o gerenciamento dessa especificação. O uso de diferentes mecanismos, associado à quantidade de aplicações individuais de cada um deles durante a modelagem de uma LPrS pode contribuir para o aumento da complexidade de gerenciamento e manutenção dessa especificação. A Tabela 19 apresenta as métricas adotadas para o estudo, associadas aos atributos sendo investigados, além de uma breve explicação sobre cada uma delas.

Tabela 19. Conjunto de Métricas Adotado no Estudo

Atributo	**Métrica**	**Significado**
Modularidade	(1) número de agrupamentos de elementos de processo associados a uma *feature* específica	Conta o número de grupos adjacentes de elementos de processo associados a uma *feature* específica
Tamanho	(2) número de elementos de processo	Conta o número total de elementos de processo usados na modelagem da LPrS
Complexidade	(3) tipos de mecanismos utilizados para especificar variabilidades em processos de software	Conta os diferentes mecanismos de cada abordagem utilizados para modelar variabilidades na LPrS
	(4) número de utilizações dos mecanismos de variabilidade em processos de software	Conta o número de utilizações dos mecanismos para a modelagem de variabilidades na LPrS

Para avaliar a modularidade das abordagens investigadas, foi adaptada uma métrica de separação de interesses (Figueiredo & al, 2008) (Garcia et al., 2005), previamente utilizada em vários estudos empíricos no contexto de linhas de produtos de software, para ser aplicada no contexto de especificações de LPrS. Este atributo foca na alternância entre grupos de elementos de processos associados à uma *feature* específica.

Um grande número de alternâncias entre grupos de elementos de processo associados a *features* diferentes torna mais difícil o seu gerenciamento, em função do espalhamento dos elementos de processo associados à uma dada *feature*.

O tamanho das especificações de LPrS em cada uma das abordagens investigadas foi quantificado por meio do número total de elementos de processo utilizados em tais especificações. De forma similar ao atributo de modularidade, quanto maior o número de elementos de processo utilizados na especificação, mais difícil são as tarefas de especificar e evoluir esta LPrS. Desta forma, a abordagem que especificar uma mesma LPrS com um menor número de elementos de processo é considerada a melhor de acordo com este critério.

Duas métricas foram definidas para avaliar a complexidade em modelar uma LPrS: (i) o número de mecanismos diferentes que a abordagem disponibiliza para a modelagem de variabilidades em processos de software; e (ii) o número de utilizações desses mecanismos durante a modelagem da referida LPrS. Quanto maior for o número de mecanismos de modelagem de variabilidades em processos de software e o número de utilizações desses mecanismos, maior será o tempo necessário para entender os mecanismos disponíveis e para escolher o mais apropriado em cada situação.

6.1.4 Abordagens de Modelagem de LPrSs

As abordagem selecionadas foram as mesmas do estudo anterior: a abordagem composicional do EPF *Composer* (Eclipse Foundation, 2012) e a abordagem anotativa do GenArch-P (Aleixo et al., 2010a) (Aleixo et al., 2010b). Essas duas abordagens foram selecionadas baseadas nos seguintes argumentos: (i) ambas possuem suporte ferramental; (ii) ambas possuem documentação acessível; e (iii) considerando a experiência da equipe responsável pela execução do estudo em utilizar tais abordagens.

A adaptação da abordagem anotativa para o contexto de LPrS é uma proposta recente (Aleixo et al., 2010a). Porém, no contexto de linhas de produtos de software, a abordagem anotativa já encontra-se consolidada, em ferramentas como: CIDE (Kästner et al., 2010), pure::variants (pure-systems, 2012) e GenArch (Cirilo et al., 2008). Nesse estudo, foi utilizada a GenArch para processos – GenArch-P – que representa uma adaptação da ferramenta GenArch para o contexto de processos de software.

6.1.5 A LPrS Alvo

Foi escolhida uma LPrS baseada no OpenUP para ser modelada nesse estudo empírico. Três instâncias de processo de software, utilizadas em projetos de pesquisa e desenvolvimento executados no IFRN foram tomados como base para a modelagem da LPrS. Estes três projetos de pesquisa e desenvolvimento envolveram: (i) o desenvolvimento de um sistema de software que realiza auditorias em uma rede telefônica; (ii) desenvolvimento de um módulo de um sistema distribuído, responsável pela recuperação e armazenamento das informações relacionadas à rede nacional de educação profissional e tecnológica do Brasil; e (iii) desenvolvimento de um sistema integrado de gerenciamento acadêmico e administrativo para as instituições federais de educação profissional e tecnológica do Brasil. Essa mesma linha de processos foi utilizada em um trabalho anterior da mesma equipe de trabalho (Aleixo et al., 2012a) (Aleixo et al., 2012b).

Durante a especificação da LPrS, a técnica extrativa (Linden et al., 2007) foi utilizada. Foram analisadas as comunalidades e variabilidades entre as instâncias de processos de software selecionadas, como membros de uma família de processos de software. A realização dessa tarefa demandou um engenheiro de processo especialista, que foi responsável por analisar a documentação e registros dos membros dessa família de processos de software. Nessa tarefa, foram extraídas similaridades e variabilidades nas práticas e atividades entre as instâncias de processo de software utilizadas, em um trabalho análogo ao da engenharia de domínio. As comunalidades e variabilidades da LPrS foram registradas e associadas à *features* de processo de software. A Tabela 20 apresenta as *features* identificadas para esta LPrS e um quantitativo dos elementos de processo associados a cada uma delas. As *features* identificadas foram classificadas de acordo com as disciplinas de processo ou com as práticas às quais estas pertenciam. Foram identificados 76 elementos de processo como parte do núcleo da LPrS, compondo os elementos mandatórios às instâncias de processo de software da mesma. No que diz respeito às variabilidades, foram encontradas 9 *features* alternativas mutuamente exclusivas (XOR) (números de 1 a 9), 4 *features* alternativas cumulativas (OR) (números de 10 a 13), e 9 *features* opcionais (números de 14 a 22).

Durante a análise de similaridades e variabilidades, também foram identificadas restrições entre *features* da LPrS. Por exemplo, a *feature* representando o uso opcional

dos *frameworks* Java EE e JUnit requer a seleção da feature que representa o uso da linguagem de programação Java.

Tabela 20. Features identificadas para a LPrS alvo

***Feature* de Processo**	**Alternativas Específicas**
Classificação: Técnicas e Tecnologias de Requisitos	
Técnica de especificação	(1) Casos de Uso
	(2) Estórias do Usuário
	(3) *Product Backlog*
Ferramenta de especificação	(4) *Astah Community*
	(5) *Rational Software Architect*
	(6) *Borland Together*
	(7) ArgoUML
Classificação: Técnicas e Tecnologias de Projeto	
Estilo de documentação arquitetural	(8) Projeto Ágil (*Agile Design*)
	(9) Arquitetura "bem documentada"
Classificação: Técnicas e Tecnologias de Implementação	
Linguagem de programação	(10) Java
	(11) C#
	(12) Ruby
	(13) Phyton
(14) Utilização do *framework* Java EE	
(15) Utilização da IDE Eclipse	
(16) Utilização do *framework* JUnit	
Classificação: Influências de Outros Processos	
(17) Elementos adicionais do processo SCRUM	
Classificação: Técnicas e Tecnologias de Integração Contínua	
(18) Utilização da ferramenta Hudson	
Classificação: Técnicas e Tecnologias de Métricas	
(19) Utilização da ferramenta Maven (métricas de código coletadas a partir do SVN)	
(20) Métrica para avaliar o progresso das atividades	
(21) Métrica para avaliar o cumprimento dos *deadlines*	
(22) Métrica para avaliar a duração das atividades	

6.2 Modelagem da LPrS Alvo

Nesta seção, serão apresentadas as estratégias e resultados da modelagem da LPrS baseada no OpenUP utilizando o EPF *Composer* (Seção 6.2.1) e o GenArch-P (Seção 6.2.2).

6.2.1 Abordagem Composicional do EPF Composer

O primeiro passo da modelagem foi a criação de um novo *plugin* de método baseado no OpenUP original para representar a LPrS, tornando a evolução da especificação da LPrS independente da evolução do *plugin* do OpenUP. O passo seguinte foi a definição dos elementos de processo mandatórios, agrupados no pacote de conteúdo denominado de "*core*". Os demais elementos de processo, associados à *features* específicas, foram organizados nos pacotes de conteúdo correspondentes.

Para as *features* de processo que implicaram em alterações no fluxo de atividades mandatório da LPrS, o comportamento adicional foi modelado e isolado na forma de um padrão de capacidade (*capability pattern*). O EPF *Composer* disponibiliza um mecanismo de composição semelhante ao dos elementos estáticos para a definição do fluxo de atividades do processo. Segundo esse mecanismo o fluxo de atividades mandatório pode ser representado por um conjunto de padrões de capacidade, os quais podem ser estendidos, refinados ou mesmo substituídos por outros padrões de capacidade, incluindo desta forma o comportamento variável. Os padrões de capacidade utilizados também tiveram os seus nomes representando as *features* que os motivaram.

Cada pacote de conteúdo, representando uma *feature* de processo específica, agregou os seguintes tipos de elementos de processo: (i) elementos adicionais, criados especificamente para a *feature* em questão; (ii) elementos especializados, criados para aplicar variabilidade de conteúdo em elementos mandatórios; e (iii) elementos relacionados a mais de uma *features*, que se repetem nos respectivos pacotes de conteúdo. A Figura 32 apresenta uma visão geral dos pacotes de conteúdo criados na modelagem da LPrS utilizando o EPF *Composer*.

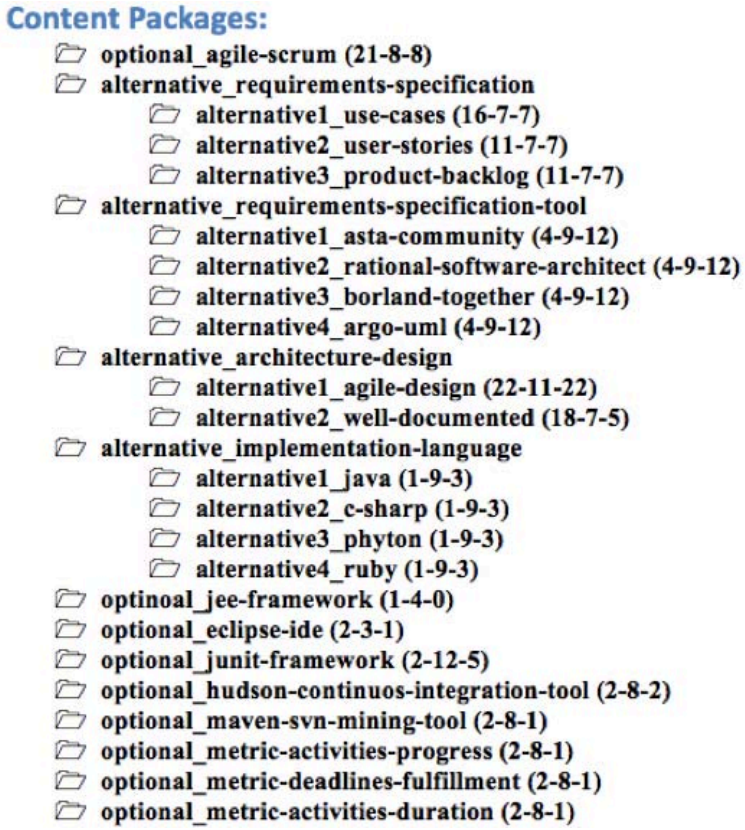


Figura 32. Organização dos elementos de processo com o EPF Composer

Para cada pacote de conteúdo ilustrado na Figura 32, também está ilustrado o número de elementos agrupados dentro do mesmo. O número dos elementos de processo é apresentado entre parênteses com três números separados por um hífen. Por exemplo, a sequência de números "(2-3-1)" para a *feature* opcional representando o uso da IDE Eclipse significa que este pacote de conteúdo contém: 2 elementos de processos específicos para esta *feature*; 3 elementos de processo que aplicam variabilidade de conteúdo em elementos mandatórios; e 1 elemento associado a outras *features*. Nesse exemplo, temos que: (i) os elementos de processo específicos são a diretriz "*Using Eclipse Plugins*" e o roteiro "*How to Create Applications on Eclipse*"; (ii) os elementos de processo que aplicam variabilidade de conteúdo são o conceito "*Coding Standard*", a atividade da fase de elaboração denominada "*Develop a Solution Increment*", e a atividade da fase de construção denominada "*Develop a Solution Increment*"; e (iii) o elemento de processo associado a outras *features* é o conceito de "*Refactoring*".

6.2.2 Abordagem Anotativa do GenArch-P

A modelagem da LPrS usando o GenArch-P envolveu, inicialmente, a definição da especificação do processo de software correspondente à LPrS. Nessa definição, foram criados todos os elementos de processo necessários, os mandatórios e os variáveis. A organização desses elementos na especificação do processo de software não é relevante, visto que o usuário só irá interagir com um modelo representando tal especificação, na qual os elementos de processo são agrupados pelo tipo. No próximo passo da modelagem da LPrS com o GenArch-P, a especificação do processo de software é submetida à ferramenta para que seja analisada. Nessa análise, é gerado automaticamente o modelo representando a especificação do processo, apresentando todos os elementos contidos na mesma. Os elementos de processo que aparecem no modelo tanto podem representar definições e descrições de papéis, artefatos, práticas orientações e etc. (elementos estáticos), ou atividades específicas do fluxo de atividades.

A Figura 33 apresenta uma ilustração de um grupo de elementos de processo, os quais representam artefatos, ou produtos de trabalho, e como estes foram associados à *features* de processo específicas. Para associar elementos de processo a *features* de processo específicas com o GenArch-P, cada elemento deve ser anotado com uma expressão de *features*. No modelo representando a especificação de processo, a anotação de um elemento de processo é feita por meio da criação de um elemento

"*FeatureExpression*" interno ao referido elemento de processo. O elemento "*FeatureExpression*" tem apenas um atributo, que é a própria expressão de *features* – como nome da *feature* relacionada, ou vários nomes de *features* conectados por operadores lógicos. Tais anotações são a representação do conhecimento de configuração – mapeamento entre *features* e ativos da LPrS (elementos de processo). As *features* são identificadas por um nome, e podem ser tanto opcionais como alternativas.

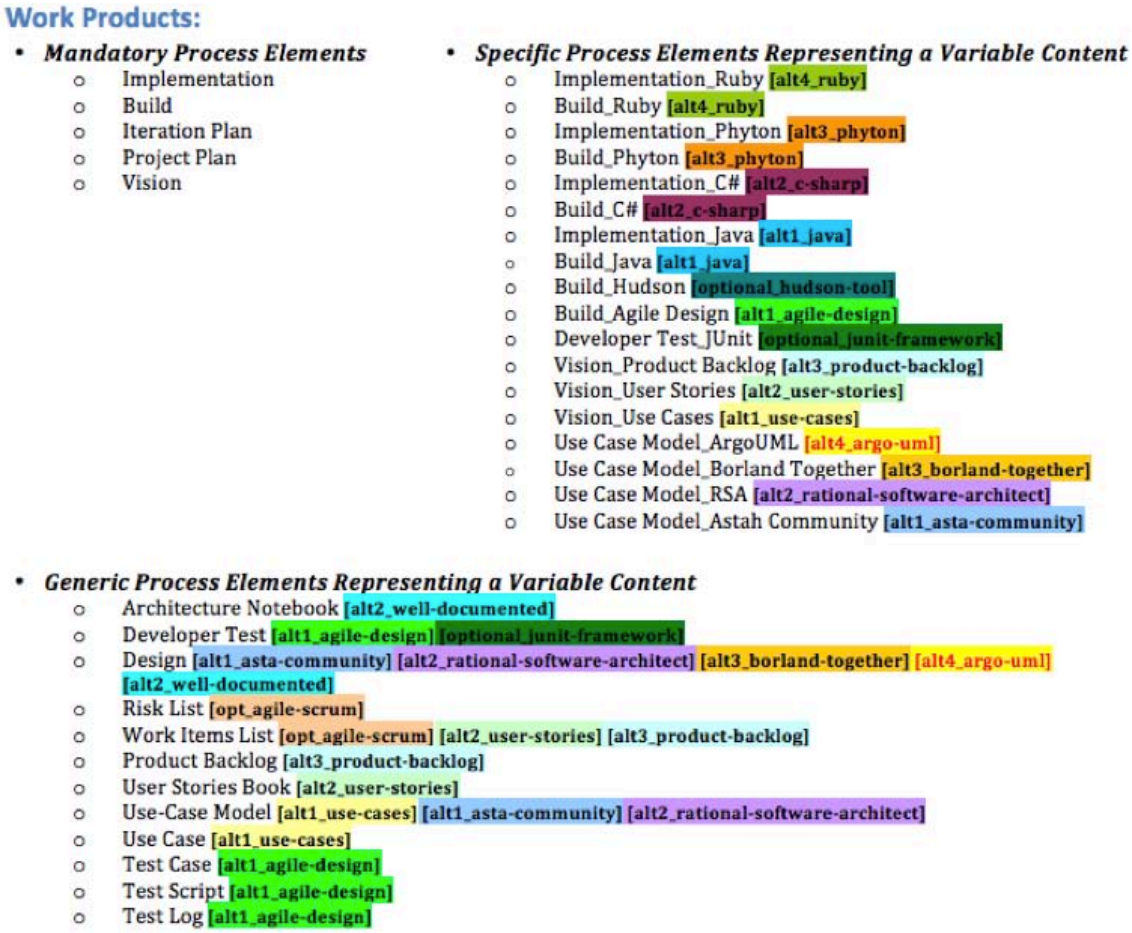


Figura 33. Ilustração de elementos de processo e anotações no GenArch-P

Os elementos de processo ilustrados na Figura 33 representam produtos de trabalho, ou artefatos, da LPrS modelada com o GenArch-P. Esses elementos estão apresentados em três grupos: (i) elementos mandatórios; (ii) elementos específicos com conteúdo variável; e (iii) elementos genéricos com conteúdo variável. Um exemplo de um artefato mandatório é a "*Implementation*", o que indica que sempre haverá uma implementação, independentemente das variações do processo. A diferença principal entre os elementos de processo específicos e genéricos é que os elementos específicos são criados para adicionar conteúdo relativo a uma *feature* em particular. Por exemplo, "*Implementation_Java*" e "*Build_Java*" descrevem especificamente a implementação e

o processo de *build* na linguagem Java. Por outro lado, um mesmo elemento de processo genérico pode ser associado a mais de uma *feature*, se necessário.

6.3 Resultados do Estudo

Após a modelagem da LPrS baseada no OpenUP utilizando as duas abordagens sendo investigadas, foram coletadas métricas das duas modelagens resultantes. A Tabela 21 apresenta um resumo dos resultados obtidos para cada uma das métricas. As seções seguintes discutem os resultados obtidos visando responder as questões de pesquisa definidas para o estudo.

Tabela 21. Resultados quantitativos para o conjunto de métricas adotado

Atributo	**Métrica**	**Abordagens de LPrS**	
		Composicional	**Anotativa**
Modularidade	Alternâncias entre grupos de elementos de processo associados à *features* específicas	21	161
Tamanho	Número de elementos de processo	516	303
Complexidade	Número de mecanismos para modelar variabilidades em processos de software	6	2
	Número de utilizações dos mecanismos para modelar variabilidades em processos de software	463	368

6.3.1 Análise de Modularidade

Visando responder a primeira questão de pesquisa (QP1), foi analisado o grau de separação de interesses das abordagens investigadas. No EPF *Composer*, a separação de interesses é quantificada pelas alternâncias entre pacotes de conteúdo, a qual foi contabilizada em 21 alternâncias. Por outro lado, no GenArch-P não é possível a definição de estruturas modulares, apenas anotações podem ser adicionadas aos elementos de processo. Para facilitar a visualização dos elementos de processo associados à uma mesma *feature*, as anotações foram representadas com uma mesma cor. Adicionalmente, os elementos anotados com uma mesma *feature* foram movidos no modelo para ficarem próximos uns dos outros, formando grupos de elementos. Dessa forma, na abordagem anotativa do GenArch-P, foram contabilizadas as alternâncias entre esses grupos de elementos de processo (com a mesma cor). O número de alternâncias total no GenArch-P foi de 161.

Com esses resultados, foi possível concluir que a abordagem composicional do EPF *Composer* apresentou um melhor resultado que a abordagem anotativa do GenArch-P, considerando a métrica de modularidade. Isso mostra que o EPF *Composer*

oferece uma melhor separação de interesses/*features* comparado ao GenArch-P. Com base nos resultados obtidos, tem-se um forte indício de que a modularidade é uma fraqueza significativa da abordagem anotativa.

6.3.2 Análise de Tamanho

Visando responder a segunda questão de pesquisa (QP2), foi analisado o número total de elementos de processo nas LPrSs modeladas com as duas abordagens investigadas. As especificações de processo em ambas as abordagens é baseada no SPEM. No SPEM, um processo de software é especificado por meio da definição e ligação de elementos de processo de software, tais como: tarefas, atividades, papeis, artefatos, guias e orientações. Durante a definição de tais elementos de processo, também são especificados os seus relacionamentos, como por exemplo: que papel é responsável pelo desenvolvimento de uma dada tarefa, que artefatos são consumidos ou produzidos por essa tarefa, e que guias e orientações são disponibilizados para auxiliar da realização desta tarefa e confecção de um dado artefato.

Nas duas abordagens investigadas, o número de elementos de processo foi maior do que a especificação original do processo OpenUP, que baseou a modelagem da LPrS. No GenArch-P, esse fato é justificado pela necessidade de modelagem dos elementos de processo variantes. Entretanto, o EPF *Composer* utilizou um número de elementos ainda maior. Essa maior quantidade de elementos de processo utilizando o EPF *Composer* foi causada, além da modelagem dos elementos de processo variantes, pela necessidade de duplicação de certos elementos de processo, relacionados a mais de uma *feature*. Considerando estes resultados, foi observado que a abordagem anotativa do GenArch-P obteve um melhor resultado segundo a métrica de tamanho, modelando a mesma LPrS com um número menor de elementos de processo.

6.3.3 Análise de Complexidade

Para responder a terceira questão de pesquisa (QP3), foi analisado o conhecimento necessário e um parâmetro relativo à quantidade de trabalho para a efetiva gerência de variabilidades na linha de processos alvo. A abordagem composicional do EPF *Composer* introduziu os conceitos de (i) pacotes de conteúdo, (ii) variabilidade de conteúdo, além de quatro tipos específicos de variabilidade de conteúdo – (iii) *contributes*, (iv) *extends*, (v) *replace*, e (vi) *extends and replace*. Esses conceitos,

definidos pela abordagem do EPF *Composer*, foram usados como mecanismos para a modelagem de variabilidades na especificação da LPrS. A abordagem anotativa do GenArch-P permitiu a modelagem de variabilidades da LPrS alvo por meio da anotação de elementos de processo relacionados a (i) *features* opcionais e (ii) *features* alternativas. Como resultado da contabilização dos mecanismos disponibilizados por cada abordagem para a gerência de variabilidades, foram identificados 6 mecanismos para o EPF *Composer*, e 2 mecanismos para o GenArch-P.

A segunda métrica para o atributo de complexidade contabilizou a utilização desses mecanismos de modelagem de variabilidades na modelagem da LPrS alvo. A abordagem composicional do EPF *Composer* modelou as variabilidades da linha de processos baseada no OpenUP utilizando 22 pacotes de conteúdo, os quais agruparam 440 elementos de processo variáveis, num total de 462 aplicações dos mecanismos de modelagem de variabilidade. É importante mencionar que em 173 dos 440 elementos de processo foram aplicados o mecanismo de variabilidade de conteúdo. Por outro lado, a abordagem anotativa do GenArch-P modelou as variabilidades da LPrS alvo utilizando 368 anotações de expressões de *features*, opcionais e alternativas. Também é importante mencionar que as 368 anotações foram aplicadas em 150 elementos de processo, porque alguns elementos de processos foram anotados com mais de uma *feature*. Considerando esses resultados, é possível concluir que a abordagem do GenArch-P obteve melhores resultados considerando as duas métricas de complexidade.

6.4 Discussões

A abordagem anotativa do GenArch-P obteve melhores resultados em dois dos três atributos analisados durante esse estudo, quais sejam: tamanho e complexidade. Por outro lado, a abordagem composicional do EPF *Composer* obteve melhores resultados considerando a métrica de modularidade. No estudo comparativo realizado anteriormente (Aleixo et al., 2012b), as abordagens do EPF *Composer* e do GenArch-P foram qualitativamente comparadas de acordo com um conjunto de critérios, tais como: modularidade, granularidade e adoção. Com base nos resultados quantitativos desse estudo, serão revisitadas e discutidas as conclusões apresentadas no estudo anterior sobre tais critérios.

O critério de modularidade utilizado no estudo anterior qualificou as abordagens investigadas segundo os níveis de suporte ao isolamento dos elementos de processo associados a uma *feature* de processo específica. Foi concluído que a abordagem composicional possuía um bom suporte ao critério de modularidade, enquanto a abordagem anotativa apenas um suporte parcial para esse critério. Com a realização desse estudo, foram contabilizados os agrupamentos de elementos de processo associados à *features* específicas – 22 na abordagem composicional e 165 na abordagem anotativa. De acordo com esses resultados quantitativos, é possível reforçar a evidência de que a abordagem composicional possui um suporte melhor ao critério de modularidade, do que a abordagem anotativa. E dada a discrepância entre os resultados, é possível diminuir a classificação da abordagem anotativa de suporte parcial, para fraco ou nenhum suporte ao critério de modularidade.

O critério de granularidade utilizado no estudo anterior qualificou as abordagens investigadas segundo os níveis de suporte à modelagem de variabilidades em diferentes granularidades (fina e grossa). O estudo anterior concluiu que a abordagem composicional possuía um bom suporte ao critério de granularidade, enquanto a abordagem anotativa possuía um suporte parcial segundo esse critério. O resultado quantitativo, obtido por uma das métricas de complexidade apresentadas nesse estudo, revelou que a abordagem composicional utilizou 22 pacotes de conteúdo (granularidade grossa), agrupando 440 elementos de processo, onde 173 desses elementos aplicavam variabilidade de conteúdo em elementos mandatórios (granularidade fina). Por outro lado, a abordagem anotativa utilizou 368 anotações (granularidade fina) para modelar variabilidades. Tais resultados quantitativos atestam as evidências do estudo anterior, segundo o critério de granularidade. Entretanto, observou-se que tal variedade de mecanismos da abordagem do EPF *Composer*, para modelar variabilidades de granularidades fina e grossa, contribuiu de forma negativa para a complexidade da modelagem de LPrSs comparando com a abordagem do GenArch-P

O critério de adoção utilizado no estudo anterior qualificou as abordagens investigadas segundo os níveis de suporte à adoção das abordagens investigadas. Quanto maior o suporte à adoção oferecido por uma abordagem, mais fácil seria a aplicação prática da mesma. O estudo anterior apresentou evidências de que a abordagem composicional possuía um fraco suporte à adoção. Os resultados quantitativos obtidos nesse estudo revelaram que a abordagem composicional gerou

uma especificação de LPrS com 516 elementos de processo, contra apenas 303 elementos de processo na especificação da mesma LPrS pela abordagem anotativa. Os resultados quantitativos também revelaram que a abordagem composicional ofereceu 6 mecanismos para a modelagem de variabilidades, contra apenas 2 mecanismos da abordagem anotativa. Como a abordagem anotativa produziu uma especificação de LPrS menor e ofereceu um menor número de mecanismos para modelar variabilidades, é possível reforçar as evidências do estudo anterior, o qual afirmou que a abordagem anotativa oferece um melhor suporte à adoção. Contudo, os mesmos resultados quantitativos mostram que a discrepância entre as abordagens não é tão grande. Desta forma, a abordagem composicional pode ter elevada a sua classificação do estudo anterior, passando a ser classificada como tendo suporte parcial ao critério de adoção.

A apresentação destas considerações sobre alguns dos resultados do estudo anterior não afetam as conclusões do mesmo, o qual identificou que de uma forma geral a abordagem anotativa obteve melhores resultados que a abordagem composicional. Os resultados quantitativos obtidos com a realização desse estudo, também apontaram na mesma direção. Por outro lado, também foi observado que considerando os critérios de modularidade e granularidade, o EPF *Composer* obteve melhores resultados que o GenArch-P. Esses pontos fortes da abordagem composicional do EPF *Composer* e os melhores resultados, no geral, da abordagem anotativa do GenArch-P, lavaram a equipe à considerar a integração dessas duas abordagens.

Uma possível abordagem integrada poderia ser baseada principalmente nas características da abordagem anotativa, para se beneficiar dos seguintes pontos fortes: (i) o uso de anotações tornam a abordagem independente do modelo utilizado para a especificação do processo de software, podendo ser adaptada para vários tipos de modelo de processos de software; (ii) a abstração de alguns detalhes (de baixo nível) da especificação do processo de software pode tornar a modelagem da LPrS mais simples; e (iii) a habilidade de explicitamente definir e modelar *features* de processo tornam a modelagem da LPrS mais consistente. Por outro lado, algumas características da abordagem composicional poderiam ser consideradas, quais sejam: (i) a habilidade de definir e visualizar estruturas modulares oferecidas pelo modelo de processo de software utilizado no baixo nível, possibilitando também a anotação de tais estruturas com expressões de *features*; e (ii) a habilidade de definir variabilidades de granularidade fina, a serem aplicadas por meio de um mecanismo de refinamento dos elementos de

processo ligado ao modelo de processo de software utilizado no baixo nível, como o mecanismo de variabilidade de conteúdo oferecido pelo EPF *Composer*.

6.5 Ameaças à Validade do Estudo

As ameaças à validade do estudo foram analisadas de acordo com a seguinte taxonomia: validade de construção, validade interna e validade externa. Uma ameaça de construção foi a forma como as abordagens investigadas foram utilizadas. A utilização e a escolha dos melhores mecanismos de cada abordagem foi discutida entre os pesquisadores participantes do estudo, durante as revisões das modelagens realizadas com cada uma das abordagens. Outra ameaça à validade de construção está relacionada com as métricas definidas para o estudo. Ao escolher as métricas a serem utilizadas para avaliar as abordagens investigadas, evitou-se a definição de um novo conjunto de métricas. Optou-se por adaptar as métricas utilizadas em um trabalho anterior, no contexto de linhas de produtos de software. Os resultados obtidos com o conjunto de métricas adotado no estudo permitiram a análise dos atributos selecionados, bem como que fossem respondidas as questões de pesquisa. Contudo, uma validação teórica e experimental do conjunto de métricas está planejada na forma de trabalhos futuros.

Uma ameaça à validade interna foi que a mesma equipe foi responsável pela modelagem da LPrS com ambas as abordagens investigadas. A escolha da equipe responsável pela condução do estudo foi realizada visando a minimização dessa ameaça; foi dada a preferência para os pesquisadores que demonstrassem um bom conhecimento nas duas abordagens sendo estudadas. Os membros da equipe de pesquisa, responsável pela execução do estudo, também contribuíram para o desenvolvimento da ferramenta GenArch-P; e possuíam uma boa experiência usando esta abordagem. Por outro lado, o EPF *Composer* é uma iniciativa consolidada na comunidade de processos de software. Vários guias, instruções e exemplos estão disponíveis na Internet, os quais também foram fundamentais para garantir o melhor entendimento das vantagens e desvantagens dessa abordagem. A equipe de pesquisa responsável pela execução do estudo também possuía uma boa experiência no uso do EPF *Composer*, devido ao fato do mesmo já ter sido utilizado em estudos anteriores.

Uma ameaça à validade externa foi a natureza acadêmica dos projetos utilizados como base para a definição da LPrS, definida como alvo para o estudo. Mesmo que tais

projetos tenham sido executados em instituições acadêmicas, as demandas o calendário destes projetos foram bastante similares a um projeto de desenvolvimento executado na “indústria”. Visando minimizar esta ameaça, está planejada a replicação desse estudo utilizando uma LPrS extraída a partir de uma instituição devotada ao desenvolvimento de software – como legítimo representante da indústria de software. Outra ameaça à validade externa do estudo é que as conclusões obtidas estão restritas às abordagens de modelagem de LPrS do EPF *Composer* e GenArch-P. Tais resultados apenas servem de evidências iniciais com relação às abordagens composicional e anotativa.

6.6 Conclusões do Estudo e Trabalhos Futuros

Após a análise e discussão dos resultados do estudo, foram devidamente respondidas as questões de pesquisa definidas para este estudo. A resposta para a primeira questão de pesquisa foi que a abordagem composicional do EPF *Composer* conseguiu produzir uma LPrS melhor modularizada – obtendo melhores resultados para a métrica de modularidade do que a abordagem anotativa do GenArch-P. A modularização é uma forte característica da abordagem composicional, a qual motiva a idealização da integração das técnicas composicional e anotativa para modelagem de LPrSs. A resposta para a segunda questão de pesquisa foi que a abordagem anotativa do GenArch-P produziu uma LPrS com um menor tamanho em termos de número de elementos de processo. Esse número reduzido de elementos pode auxiliar o engenheiro de processo no gerenciamento da complexidade dessa tarefa. A resposta para a terceira questão de pesquisa foi que a abordagem anotativa do GenArch-P possibilitou a modelagem da LPrS com menor complexidade – segundo os resultados da métrica de complexidade. A maior complexidade da abordagem do EPF *Composer* é relativa: (i) ao maior número de mecanismos de modelagem de variabilidades em processos de software, e (ii) à maior utilizações desses mecanismos na modelagem da LPrS.

Em termos gerais, o estudo trouxe evidências preliminares que com base nos atributos selecionados para o estudo, e no contexto específico da LPrS alvo modelada pelas duas abordagens, que a abordagem anotativa do GenArch-P obteve melhores resultados. A análise dos pontos fortes e fracos das abordagens investigadas sugeriu que a integração das características das abordagens composicional e anotativa poderia produzir uma melhor solução para a modelagem de LPrS no que se refere a lidar com os atributos internos de modularidade e tamanho.

7 Experimento Controlado de Análise das Abordagens Composicional e Anotativa para LPrSs

Em estudos comparativos anteriores, apresentados nos Capítulos 5 e 6 (Aleixo et al., 2012a) (Aleixo et al., 2012b) (Aleixo et al., 2013), foram apresentadas importantes evidências relativas aos pontos fortes e fracos das abordagens composicional do EPF *Composer* e anotativa do GenArch-P. Contudo, nesses estudos, as abordagens foram utilizadas apenas pela equipe de pesquisadores, responsáveis pela condução dos estudos. Os pesquisadores em questão buscaram de forma criteriosa utilizar o melhor de cada abordagem, especificamente: técnicas, mecanismos e ferramentas propostos pelas mesmas. Após a aplicação das abordagens pela equipe de pesquisadores sobre linhas de processos de software, os resultados foram analisados. Os resultados obtidos nessas análises serviram de parâmetro para as discussões e conclusões de tais estudos. Uma das lacunas identificadas nesses estudos está no fato de que os mesmos não envolveram o público externo, e por conseguinte não puderam ser analisados aspectos, tais como, compreensibilidade e esforço necessário para a utilização dessas abordagens.

Visando suprir essa lacuna, e uma análise também da utilização das abordagens composicional e anotativa, foi planejada a execução de um experimento controlado. O objetivo do experimento controlado foi o de avaliar a utilização das referidas abordagens. Dado que as abordagens específicas do EPF *Composer* e GenArch-P já haviam sido utilizadas nos estudos comparativos anteriores, optou-se por continuar analisando tais abordagens específicas. Optou-se por utilizar as abordagens específicas do EPF *Composer* e GenArch-P, representando as abordagens composicional e anotativa, respectivamente. A escolha se deu em função de já haverem sido realizadas análises comparativas das mesmas, o que possibilita que os resultados anteriores sejam revisitados e complementados pelos resultados obtidos por este estudo em particular.

As seções desse capítulo estão organizadas segundo o modelo proposto por Wohlin (Wohlin, 2012) para a documentação de experimentos controlados, quais sejam: introdução (Seção 7.1), definição do experimento (Seção 7.2), planejamento do experimento (Seção 7.3), linhas de processos de software alvo (Seção 7.4), realização do experimento (Seção 7.5), apresentação dos resultados (Seção 7.6), análises qualitativas (Seção 7.7), e conclusões e trabalhos futuros (Seção 7.8).

7.1 Introdução

Já foi discutido anteriormente que o uso de abordagens de gerência de variabilidades em LPrSs pode trazer uma série de benefícios, tais como: (i) a reutilização do conhecimento, representado pela especificação de uma família de processos de software; (ii) redução no tempo de instanciação de um processo que atenda às necessidades específicas de um dado projeto; (iii) facilidade para manutenção das evoluções de uma família de processos; entre outras vantagens. A importância desses benefícios estimulam as pesquisas na área e o surgimento de várias propostas visando a solução do problema da gerência de variabilidades em LPrSs.

A revisão sistemática da literatura, descrita no Capítulo 3, sobre o tema de linhas de processos de software apresentou, dentre outras, as seguintes conclusões: (i) a necessidade crescente de estratégias e técnicas de reúso em especificações de processos de software; e (ii) a existência de várias propostas de abordagens de gerência de variabilidades em processos de software. A maioria dos trabalhos que apresentam propostas de abordagens de gerência de variabilidades em processos de software reflete trabalhos ainda em amadurecimento. Consequentemente, foi identificada uma carência na definição de critérios que estabelecem parâmetros para a comparação das abordagens, bem como de estudos empíricos ou análises comparativas entre as diferentes abordagens propostas. Diante desse contexto, este capítulo apresenta o relato da execução de um experimento controlado com o objetivo de: **avaliar** as abordagens composicional e anotativa para a gerência de variabilidades em processos de software; **com o propósito de** comparar tais abordagens; **no que diz respeito aos** atributos de esforço e compreensibilidade; **do ponto de vista de** potenciais usuários destas abordagens, **no contexto de** alunos de pós-graduação na área de computação com os mais diversos perfis, mas com experiência anterior em engenharia de software e LPrS. Serão analisadas as abordagens composicional e anotativa por meio de exemplares destas abordagens: o EPF *Composer*, representando a abordagem composicional, e o GenArch-P, representando a abordagem anotativa.

7.2 Definição do Experimento

O atributo de esforço foi analisado segundo dois aspectos: (i) o tempo na modelagem de uma LPrS alvo; e (ii) o tempo na realização de alterações na modelagem nessa mesma LPrS. Com relação ao atributo de compreensibilidade, foram coletadas as

impressões gerais (qualitativas) dos participantes ao utilizar cada uma das abordagens sendo investigadas. Ainda com relação ao atributo de compreensibilidade, foi analisada a corretude na execução das tarefas propostas. De acordo com o objetivo definido para o experimento – avaliação dos atributos de esforço e compreensibilidade – foram definidas três questões de pesquisa (QP), as quais orientaram a execução do estudo:

QP1. A abordagem do GenArch-P possibilita a modelagem de *features* em uma LPrS em menos tempo que a abordagem do EPF *Composer*?

QP2. A abordagem do GenArch-P possibilita que alterações na modelagem de *features* sejam realizadas em menos tempo que na abordagem do EPF *Composer*?

QP3. Quais as impressões dos usuários das abordagens investigadas com relação à utilização e compreensibilidade das mesmas?

Visando responder às duas primeiras questões de pesquisa, relativas ao atributo de esforço, foram definidas duas métricas, são elas: (i) tempo de modelagem de uma LPrS; e (ii) tempo da realização de alterações em uma LPrS. Já para responder a terceira questão de pesquisa, foi idealizado um questionário para ser respondido pelos participantes do experimento, visando coletar as suas impressões no uso de cada uma das abordagens. A partir das respostas desse questionário foi realizada uma avaliação qualitativa das impressões gerais e compreensibilidade das abordagens investigadas.

7.3 Planejamento do Experimento

O experimento foi organizado segundo o modelo de quadrado latino (Ryan, 2011). De acordo com essa forma de organização, foram tabulados os fatores: (i) participante e (ii) LPrS alvo, os quais apareceram nas linhas e colunas do quadrado latino, respectivamente. As abordagens investigadas apareceram apenas uma vez nas linhas e colunas do quadrado latino. Desta forma, cada quadrado latino foi organizado na forma de uma matriz dois por dois (duas linhas e duas colunas), contendo: (i) dois participantes; (ii) duas LPrSs alvo; e (iii) cada participante devendo aplicar cada uma das abordagem investigada com uma LPrS alvo diferente e de forma alternada. Para possibilitar resultados com um bom nível de confiança foi planejada a utilização de 6 (seis) réplicas desta configuração de quadrado latino, com um total de 12 (doze) participantes. A escolha pelo formato de quadrado latino se deu pelo fato dessa forma

de organização possibilitar um bom controle da variância dos fatores envolvidos no experimento (Leroy, 2011). O restante desta seção apresenta o planejamento do experimento, desde a definição das hipóteses a serem testadas, até a apresentação das ameaças à validade identificadas para o estudo, além das iniciativas visando a minimização de tais ameaças.

7.3.1 Definição das Hipóteses

Para as duas primeiras questões de pesquisa foram definidas métricas de tempo, visando mensurar o esforço de utilização das abordagens investigadas na modelagem e realização de alterações em uma LPrS. Para essas duas questões, foram formuladas as seguintes hipóteses:

Hipóteses relativas a QP1: **H1** – em média, a abordagem GenArch-P possibilita a realização das tarefas de modelagem de uma LPrS em menos tempo ($tModel_{GenArch-P} < tModel_{EPF\ Composer}$); **H2** – em média, a abordagem EPF *Composer* possibilita a realização das tarefas de modelagem de uma LPrS em menos tempo ($tModel_{EPF\ Composer} < tModel_{GenArch-P}$); e a hipótese nula (**H0**) para esta questão de pesquisa é que o tempo, em média, de realização das tarefas de modelagem de uma LPrS utilizando as abordagens investigadas é equivalente ($tModel_{GenArch-P} \approx tModel_{EPF\ Composer}$).

Hipóteses relativas à QP2: **H1** – em média, a abordagem GenArch-P possibilita a realização das tarefas de alteração em uma LPrS em menos tempo ($tAltera_{GenArch-P} < tAltera_{EPF\ Composer}$); **H2** – em média, a abordagem EPF *Composer* possibilita a realização das tarefas de alteração em uma LPrS em menos tempo ($tAltera_{EPF\ Composer} < tAltera_{GenArch-P}$); e a hipótese nula (**H0**) para esta questão de pesquisa é que o tempo, em média, de realização das tarefas de alterações em uma LPrS utilizando as abordagens investigadas é equivalente ($tAltera_{GenArch-P} \approx tAltera_{EPF\ Composer}$).

Essas duas hipóteses serão testadas por meio da realização da análise de variância dos tempos obtidos pelos participantes do experimento na realização das tarefas de modelagem e alteração de uma LPrS – ANOVA (Hair et al., 2007). Para a

interpretação do resultado da ANOVA será utilizado o nível de significância de 5% (cinco por cento) para a rejeição da hipótese nula (**H0**) – valor-p <= 0,05.

Hipóteses relativas à QP3: **H1** – a abordagem GenArch-P se mostrará mais fácil de entender e utilizar; **H2** – a abordagem EPF *Composer* se mostrará mais fácil de entender e utilizar; e a hipótese nula (**H0**) é de que as duas abordagens se mostrarão equiparadas em termos da dificuldade de compreensão e utilização. A validação dessas hipóteses foi realizada através da análise e discussão das respostas dos participantes no questionário de avaliação das abordagens; bem como análise e discussão dos resultados de corretude das tarefas de modelagem e alteração das LPrSs alvo. Com relação à corretude, foi analisado o percentual de tarefas de modelagem e alteração que foram respondidas de forma errada, para cada uma das abordagens. Por fim, a análise geral dos resultados das avaliações e da corretude das tarefas de modelagem e alterações na LPrS, foi utilizada na comprovação dessa hipótese.

7.3.2 Seleção das Variáveis

Foi considerada variável dependente para o experimento: o tempo que cada participante levou para a execução de cada uma das tarefas de modelagem e alterações em uma LPrS alvo. Por outro lado, foram consideradas como variáveis independentes: (i) as LPrSs alvo utilizadas na realização das tarefas; e (ii) as abordagens específicas utilizadas. A variável independente “LPrS alvo” é um fator com dois tratamentos – duas LPrSs distintas, ambas baseadas no OpenUP. A variável independente “Abordagem” também é um fator com dois tratamentos – EPF *Composer* e GenArch-P. A configuração do estudo visou reduzir o efeito de aprendizagem das abordagens sob investigação; da mesma forma que buscou a minimização do efeito da LPrS modelada.

7.3.3 Seleção dos Participantes

Foi selecionado um grupo de 12 (doze) alunos de mestrado e doutorado ligados ao Programa de Pós-graduação em Sistemas e Computação da UFRN (PPgSC/UFRN), todos da área de pesquisa de Engenharia de Software, para participar do experimento. Destes, foram selecionados 8 (oito) alunos sem nenhuma experiência em linhas de produtos de software e em LPrSs; e 4 (quatro) alunos com experiência em linhas de produtos de software, mas com pouco conhecimento em LPrSs. De forma geral, os discentes do PPgSC/UFRN possuem um bom conhecimento em engenharia de software.

O conhecimento em engenharia de software desses alunos, em parte, é fruto das experiências anteriores dos mesmos (cursos de graduação e projetos de pesquisa e/ou desenvolvimento), bem como das próprias disciplinas do programa. Das 6 (seis) réplicas de quadrado latino planejadas para o experimento, 2 (duas) delas foram compostas pelos participantes mais experientes e 4 (quatro) delas pelos participantes menos experientes. A distribuição dos tratamentos aos participantes em cada uma das réplicas de quadrado latino foi aleatorizada por meio de sorteios.

7.3.4 Instrumentação

O objetivo da instrumentação é prover meios para a realização do experimento e o seu monitoramento. Os instrumentos de um experimento podem ser classificados em (Wohlin, 2012): objetos, orientações e instrumentos de medida.

A realização do experimento ocorreu em um dos laboratórios do curso de Ciência da Computação do Departamento de Informática e Matemática Aplicada da UFRN. Cada participante utilizou um computador (**objeto**), nos quais foram instaladas as ferramentas relativas às duas abordagens. Como **orientação** para a realização do experimento cada participante utilizou: (i) um roteiro descrevendo as atividades a serem realizadas; (ii) os slides utilizados nos treinamentos das abordagens; e (iii) anotações de cada participante, feitas durante o treinamento das abordagens. Como **instrumento de medida** foi utilizada, por cada participante, uma aplicação de “cronômetro”, com o tempo registrado em minutos e segundos. Os roteiros, além das atividades a serem executadas, também instruíam sobre os momentos em que os participantes deveriam zerar, iniciar, parar e ler o cronômetro, bem como o devido registro dos tempos auferidos em um determinado local do próprio roteiro.

Ao final dos roteiros, foi adicionado um questionário onde os participantes foram solicitados a responder cinco questões com um valor numérico de 1 a 5; onde 1 (um) representava “muito ruim” e 5 (cinco) representava “muito bom (boa)”. As questões utilizadas foram as seguintes: (i) “como você qualifica o uso da referida abordagem para a realização das tarefas solicitadas?”; (ii) “como você qualifica a facilidade de entender como a abordagem expressa o conhecimento de configuração?”; (iii) “como você qualifica o acesso ao conhecimento de configuração com a referida abordagem?”; (iv) “como você qualifica a tarefa de explicar para um leigo o resultado

da modelagem de uma LPrS com a referida abordagem?"; (v) "como você recomendaria a abordagem para um colega interessado em realizar tarefas equivalentes?". Nesse mesmo questionário, os participantes foram solicitados a registrar os pontos fortes e pontos fracos que identificaram para a abordagem que acabaram de utilizar. As perguntas definidas para o questionário visaram caracterizar aspectos, tais como: (i) se a compreensão de cada uma das abordagens pelos participantes foi fácil ou difícil (1ª e 4ª questões); (ii) se foi fácil para os participantes acessar o conhecimento de configuração por meio de cada abordagem (2ª e 3ª questões); além de (iii) uma avaliação geral da abordagem na opinião do participante (5ª questão e pontos fortes e fracos).

7.3.5 Ameaças à Validade

Esta seção apresenta e discute as ameaças à validade do estudo que foram identificadas, e como as mesmas foram tratadas. As ameaças à validade do estudo foram classificadas de acordo com as seguintes nomenclaturas: (i) validade interna, (ii) validade externa, (iii) validade de construção e (iv) validade da conclusão.

A primeira ameaça à validade interna do estudo foi com relação ao controle da influência dos fatores envolvidos, tais como: LPrSs alvo e participantes. Uma das motivações para a adoção da estratégia de quadrado latino foi justamente proporcionar um maior controle da influência desses fatores, evidenciando o fator de interesse – que era o uso de uma determinada abordagem. Cada réplica de quadrado latino avaliou: (i) as duas abordagens, (ii) utilizadas por dois participantes e (iii) em tarefas relacionadas a duas LPrSs alvo; fatores esses devidamente aleatorizados. Foram utilizadas 6 (seis) réplicas ao todo – 4 (quatro) réplicas com participantes sem conhecimento prévio de LPrS; e 2 (duas) réplicas com participantes com conhecimento mínimo de LPrS. Contudo, está planejada a repetição do experimento com um número maior de participantes, visando aumentar a confiança nos resultados obtidos.

A segunda ameaça à validade interna do estudo foi o fato do tempo de realização das atividades serem coletados pelos próprios participantes. Para minimizar essa ameaça, foi instituído o uso de uma mesma aplicação para a coleta dos tempos. A ferramenta em questão registra o tempo em minutos, segundos e centésimos de segundo, embora os participantes só precisassem registrar os minutos e segundos. Além disso, os participantes foram instruídos no início da atividade de aquecimento (*warm-*

up), e novamente no início do experimento sobre como deveria ser feita a coleta do tempo. A execução do experimento foi acompanhada, para assegurar que os participantes estivessem registrando os tempos de forma correta. A terceira ameaça à validade interna do estudo foi com relação ao controle das perturbações não desejadas para o experimento, como por exemplo, o surgimento de uma dúvida na utilização de uma das abordagens durante a execução do experimento. Para minimizar essa ameaça, a realização do experimentos foi precedida de atividades de aquecimento, nas quais os participantes realizaram tarefas equivalentes às que realizariam durante o experimento. O aquecimento serviu para a ambientação dos participantes na abordagem e para que pudessem ser sanadas as últimas dúvidas.

A primeira ameaça à validade externa do estudo foi com relação ao número e o perfil dos participantes envolvidos. Foram selecionados para participar do estudo apenas alunos de pós-graduação do PPgSC/UFRN, proporcionando uma homogeneidade nos perfis dos participantes. A homogeneidade no perfil dos participantes foi interessante, a princípio, como forma de minimizar o efeito deste perfis nos resultados do estudo. Embora o número dos participantes não tenha sido muito grande, segundo a estratégia do quadrado latino todos eles utilizaram as duas abordagens. Contudo, está planejada para a repetição do experimento com um número ainda maior de participantes, preferencialmente composto de profissionais ligados à indústria de software, visando verificar se o número e o perfil dos participantes, vai impactar de alguma forma nos resultados obtidos.

Os desvios do comportamento esperado para os participantes e experimentador, os quais poderiam ameaçar a validade da construção do experimento, foram controlados através das seguintes ações específicas: (i) o experimento foi planejado e discutido por cerca de três meses, antes de ser executado; (ii) foram elaborados roteiros detalhados para os participantes do experimento; (iii) durante o treinamento das abordagens os participantes foram encorajados a tirar todas as possíveis dúvidas com relação às abordagens; e (iv) durante as seções de *warm-up*, o resultado gerado pelos participantes foi corrigido *in loco*, e os participantes foram estimulados a realizar as correções.

A primeira ameaça à validade de conclusão estava relacionada à habilidade de chegar a uma conclusão a respeito dos relacionamentos entre o tratamento e o resultado do experimento. Para minimizar essa ameaça foi adotado o modelo de quadrado latino

visando reduzir a influência dos outros fatores envolvidos no experimento sobre o tratamento – as abordagens composicional e anotativa para a gerência de variabilidades em LPrS. A segunda ameaça à validade de conclusão foi a confiabilidade das medidas. Para minimizar essa ameaça foram tomadas as seguintes precauções: (i) a coleta do tempo de execução das tarefas foi solicitada nas seções de aquecimento, para que os participantes se familiarizassem com tal prática; (ii) foi utilizada a mesma aplicação de cronômetro por todos os participantes; (iii) os instantes em que o cronômetro deveria ser zerado e que o tempo deveria ser coletado estavam descritos no roteiro adotado pelos participantes; (iv) antes de cada seção, o pesquisador ressaltou novamente os momentos de zerar o cronômetro e marcar o tempo; e (v) durante a realização das seções do experimento o pesquisador verificou se os tempos estavam sendo marcados corretamente.

7.4 As Linhas de Processos Alvo

Foram utilizadas três LPrSs na execução do experimento. As três LPrSs utilizadas foram definidas como subconjuntos de uma LPrS baseada no OpenUP (EPF, 2011). A primeira das LPrSs foi utilizada no aquecimento para o experimento – para esta LPrS foi solicitado ao participante a modelagem (i) da *feature* opcional “Utilização do framework JEE” e (ii) da *feature* alternativa “Linguagem de Programação”; com as alternativas “Java” e “C#”. As outras duas LPrSs foram utilizadas na execução do experimento e foram definidas visando permitir que os participantes pudessem realizar as tarefas propostas em um intervalo de tempo não superior a duas horas. A duração máxima de duas horas teve por objetivo garantir que o experimento não se tornasse cansativo, o que prejudicaria os resultados do mesmo.

A modelagem da “LPrS-1”, incluiu a definição: (i) da *feature* opcional “Utilização do *framework* JUnit”; (ii) da *feature* alternativa “Técnica de especificação de requisitos”, com as alternativas “casos de uso”, “estórias do usuário” e “*product backlog*”; (iii) da *feature* opcional “Utilização da ferramenta Eclipse IDE”; além das seguintes alterações na *feature* alternativa já citada – (iv) a remoção da alternativa “*product backlog*”; (v) o retorno de alguns elementos de processo a condição de mandatórios; (vi) a inclusão da alternativa “cartões CRC”; e (vii) associação de elementos de processo mandatórios à *features* alternativas específicas.

A modelagem da “LPrS-2”, incluiu a definição: (i) da *feature* opcional “Utilização da ferramenta Hudson”; (ii) da *feature* alternativa “Estilo de especificação da arquitetura”, com as alternativas “projeto ágil” e “arquitetura bem documentada”; (iii) da *feature* opcional “Incorporação de influências do processo Scrum”; além das seguintes alterações na *feature* alternativa já citada – (iv) a remoção da alternativa “arquitetura bem documentada”; (v) o retorno de alguns elementos de processo a condição de mandatórios; (vi) a inclusão da alternativa “estilo Booch”; e (vii) associação de elementos de processo mandatórios à *features* alternativas específicas.

7.5 Realização do Experimento

Esta seção apresenta as etapas que marcaram a execução do experimento. Serão descritas as etapas de (i) preparação para a execução do experimento (Seção 7.5.1), (ii) execução propriamente dita (Seção 7.5.2) e (iii) validação dos dados obtidos para o procedimento de análise de variância (ANOVA) dos mesmos (Seção 7.5.3).

7.5.1 Preparação

O planejamento, que antecedeu a preparação do experimento, definiu: (i) que o experimento seria configurado na forma de quadrado latino; (ii) que seriam utilizados 12 (doze) participantes, distribuídos em 6 (seis) réplicas de quadrado latino – com duas delas compostas por participantes mais experientes e quatro por participantes menos experientes; (iii) que o experimento aconteceria em 4 (quatro) dias, não ultrapassando duas horas em cada dia; e (iv) que as tarefas de modelagem e alterações na LPrS alvo deveriam ser projetadas para serem concluídas em menos de duas horas.

Como tarefas de preparação para a realização do experimento, foram realizadas as seguintes tarefas: (i) definição do cronograma (dia-a-dia) para o experimento; (ii) definição das LPrSs alvo a serem utilizadas na realização do experimento; (iii) definição dos roteiros de atividades para o aquecimento e para a execução do experimento, com as duas LPrSs alvo; e (iv) aleatorização das réplicas de quadrado latino a serem utilizadas.

Foi definida a seguinte cronologia para o experimento: 1º dia (1 hora e meia) – treinamento introdutório, explicando a fundamentação de LPrS e dando uma visão geral das abordagens sob investigação; 2º dia (1 hora por abordagem) – treinamento prático de utilização das abordagens; 3º dia (2 horas) – realização de aquecimento com a

abordagem sorteada para o dia (30 minutos) e realização do roteiro de atividades com a abordagem sorteada para o dia (1 hora e meia); e por fim o 4º dia (2 horas) – realização de aquecimento com a abordagem diferente da já utilizada (30 minutos) e realização do roteiro de atividades com a abordagem diferente da já utilizada (1 hora e meia).

Para a aleatorização das réplicas de quadrado latino compostas pelos participantes menos experientes foram realizados os seguintes sorteios: (i) posição dos participantes nas 4 (quatro) réplicas correspondentes; (ii) a ordem das LPrSs em cada réplica; e (iii) a ordem das abordagens a serem utilizadas por cada participante. Foi garantido que as alternativas possíveis tivessem o mesmo número de sorteios. A Figura 34 ilustra o resultado da aleatorização das LPrSs e abordagens para as réplicas de quadrado latino contendo os participantes menos experientes.

RÉPLICA 1	SPrL_alvo_1	SPrL_alvo_2
Participante_1	GenArch-P	EPF Composer
Participante_2	EPF Composer	GenArch-P
RÉPLICA 2	**SPrL_alvo_2**	**SPrL_alvo_1**
Participante_1	EPF Composer	GenArch-P
Participante_2	GenArch-P	EPF Composer
•••		
RÉPLICA 4	**SPrL_alvo_2**	**SPrL_alvo_1**
Participante_1	EPF Composer	GenArch-P
Participante_2	GenArch-P	EPF Composer

Figura 34. Resultado da aleatorização da execução do experimento

O sorteio dos participantes para ocupar as 8 (oito) posições possíveis nas 4 (quatro) réplicas de quadrado latino obedeceu os seguintes passos: (i) os nomes dos participantes foram listados em ordem alfabética em uma planilha; (ii) foi criada uma coluna denominada de “aleatório”, contendo o resultado da expressão randômica – “random()” (gerando um valor real aleatório entre zero e um); (iii) em seguida, os participantes foram ordenados pela coluna “aleatório”; e (iv) cada quadrado foi composto por dois participantes conforme a ordem obtida, com os primeiros dois compondo a primeira réplica e assim por diante.

O sorteio das posições das LPrSs alvo também seguiu um procedimento semelhante, onde: (i) os nomes das LPrSs alvo foram listados em uma planilha; (ii) foi criada uma coluna denominada “aleatório”, contendo o resultado da função “*random*()”;

(iii) em seguida, as LPrSs alvo foram ordenadas pela coluna “aleatório”; e por fim (iv) as LPrSs alvo foram usadas nessa ordem na primeira réplica de quadrado latino e na ordem inversa na segunda réplica, e assim por diante. Aleatorização semelhante foi realizada para as abordagens sob investigação. As duas réplicas de quadrado latino, compostas pelos participantes mais experientes, foram aleatorizadas de forma semelhante. Inicialmente foram aleatorizados os seus participantes e, em seguida, as abordagens a serem utilizadas em cada uma das rodadas por cada participante.

7.5.2 Execução

Ocorreram apenas duas exceções ao que havia sido planejado para o experimento. A primeira exceção foi a realização de um dia a mais de treinamento para os participantes com menos experiência, visto que os mesmos ainda demonstraram insegurança ao final da realização do treinamento planejado para o experimento. O treinamento aconteceu com 13 (treze) participantes, sendo um deles como reserva. A segunda exceção aconteceu antes da execução da primeira rodada do experimento, onde um dos participantes informou que precisaria viajar, nesse caso foi convocado o reserva. A execução do experimento aconteceu em quatro dias, no primeiro dia foi feita uma introdução à LPrS e treinamento com as abordagens. No segundo dia, foi repetido o treinamento com as abordagens. No terceiro dia, foi realizado (i) o aquecimento com a abordagem a ser utilizada no dia, (ii) a execução do roteiro de tarefas e (iii) aplicação do questionário de avaliação da abordagem utilizada. No quarto dia, foi realizado (i) o aquecimento com a outra abordagem, (ii) a execução do roteiro com de tarefas e (iii) a aplicação do questionário de avaliação da abordagem utilizada.

Os tempos das atividades foram registrados, pelos participantes, no próprio roteiro que receberam, assim como as respostas ao questionário de avaliação. Ao final da execução, os participantes devolveram os roteiros. Também foi recolhido o ambiente que os mesmos utilizaram para a realização das tarefas do roteiro, visando identificar se as tarefas foram realizadas de forma correta.

7.5.3 Validação dos Dados

Para a validação dos dados obtidos, foram realizadas duas verificações: (i) se o experimento ocorreu conforme foi planejado e instruído e (ii) se a coleta de dados ocorreu conforme foi instruída. A execução do experimento foi acompanhada por um

pesquisador responsável, o qual instruiu sobre a aferição dos tempos de realização das tarefas. Dado que tudo ocorreu conforme planejado e não ocorreu nenhuma anormalidade da execução do mesmo, sendo os dados obtidos considerados válidos.

7.6 Apresentação dos Resultados

Os resultados do experimento serão apresentados e discutidos (i) levando em conta cada uma das tarefas realizadas no experimento; bem como as totalizações referentes (ii) às tarefas de modelagem da LPrS alvo e (iii) às tarefas de alterações na modelagem dessa mesma LPrS. Ao final serão discutidas as influências da experiência dos participantes nos resultados, bem como das LPrS escolhidas nos resultados obtidos.

7.6.1 Resultados Obtidos pelos Participantes do Experimento

A Tabela 22 apresenta os resultados obtidos pelos 12 (doze) participantes do experimento, nas duas rodadas de execução do experimento. Por uma questão de impessoalidade foram omitidos os nomes dos mesmos. A Tabela 22 registra os tempos de execução, em segundos, de cada uma das tarefas descritas no roteiro do experimento.

Tabela 22. Resultados obtidos pelos participantes do experimento

1ª RODADA									
Participante	**Abordagem**	**LPrS**	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**
P1:Réplica1	Genarch-P	1	323	536	168	160	77	153	151
P2:Réplica1	EPF Comp.	1	714	1227	355	44	146	325	637
P1:Réplica2	EPF Comp.	2	536	725	150	12	43	453	145
P2:Réplica2	Genarch-P	2	294	328	311	181	80	377	180
P1:Réplica3	Genarch-P	1	540	720	180	360	180	240	180
P2:Réplica3	EPF Comp.	1	542	894	165	4	80	243	667
P1:Réplica4	EPF Comp.	2	707	1476	379	120	90	253	220
P2:Réplica4	Genarch-P	2	287	305	321	88	43	242	122
P1:Réplica5	Genarch-P	1	313	656	185	135	129	159	175
P2:Réplica5	EPF Comp.	1	560	889	222	40	203	245	262
P1:Réplica6	EPF Comp.	2	411	861	377	214	99	267	215
P2:Réplica6	Genarch-P	2	287	410	187	91	20	155	89
2ª RODADA									
P1:Réplica1	EPF Comp.	2	712	536	231	109	51	198	176
P2:Réplica1	Genarch-P	2	671	728	315	220	163	288	172
P1:Réplica2	Genarch-P	1	293	468	182	9	94	168	168
P2:Réplica2	EPF Comp.	1	922	935	291	175	208	300	279
P1:Réplica3	EPF Comp.	2	1082	1141	285	75	81	366	212
P2:Réplica3	Genarch-P	2	839	400	214	30	55	295	89
P1:Réplica4	Genarch-P	1	858	1593	257	75	137	205	365
P2:Réplica4	EPF Comp.	1	369	449	153	120	55	133	188
P1:Réplica5	EPF Comp.	2	330	604	257	172	53	230	134
P2:Réplica5	Genarch-P	2	318	449	344	266	62	219	152
P1:Réplica6	Genarch-P	1	561	859	304	182	148	216	266
P2:Réplica6	EPF Comp.	1	519	935	204	7	104	240	274

Conforme já foi citado, as tarefas solicitaram: (i) a modelagem de duas *features* opcionais (T1 e T3); (ii) a modelagem de uma *feature* alternativa (T2); e (iii) quatro tarefas de alteração da modelagem da *feature* alternativa (T4 – exclusão de alternativa, T5 – desassociar elementos de processo de uma *feature* específica, T6 – inclusão de alternativa, e T7 – associar elementos de processo à *feature* específica). Os tempos estão apresentados, na referida tabela, apenas em segundos, embora tenham sido originalmente registrados na forma de minutos e segundos. Os tempos foram convertidos em segundos, para facilitar a manipulação e análise estatística dos mesmos.

As análises dos resultados foram realizadas tomando por base: (i) os tempos de cada tarefa individualmente, (ii) o somatório dos tempos das tarefas de modelagem, e (iii) o somatório dos tempos das tarefas de alterações. Foram realizadas análises de variância (ANOVA) dos resultados obtidos, validando, ou não, as hipóteses definidas.

7.6.1.1 Resultados da Primeira Tarefa de Modelagem da LPrS Alvo

A Figura 35 apresenta um sumário dos tempos para a primeira tarefa de modelagem (de uma *feature* opcional). A análise visual deste sumário não possibilita nenhuma conclusão prévia; entretanto, é possível perceber que a abordagem GenArch-P possui alguns resultados pontuais melhores do que a abordagem EPF *Composer*.

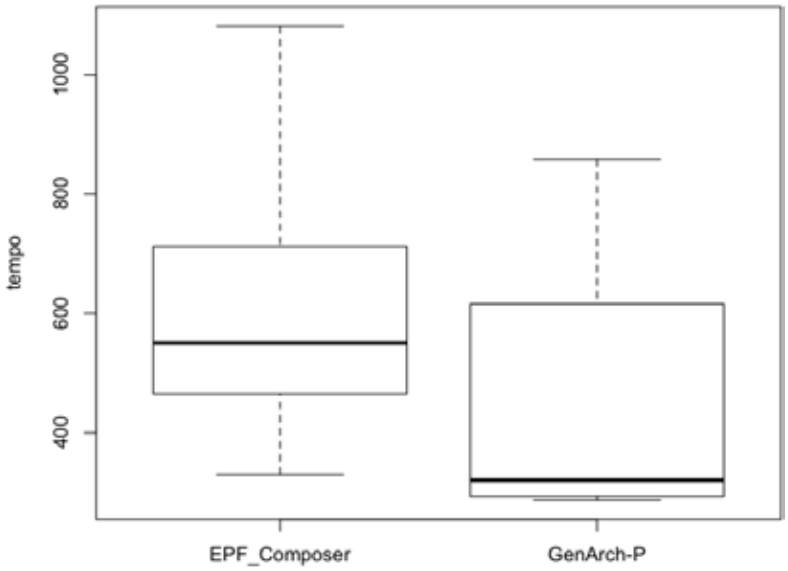
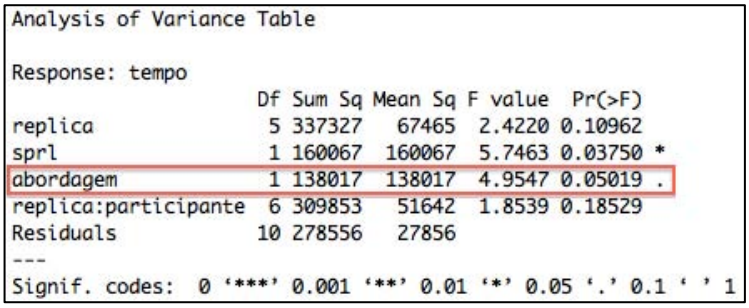


Figura 35. Sumário dos resultados para a primeira tarefa do experimento (T1)

A Figura 36 ilustra a tabela de análise de variância (ANOVA) para os resultados da modelagem da primeira *feature* opcional. O valor-p calculado foi de 5% (0,05019), o qual permite rejeitar a hipótese nula. Portanto, os resultados apontam que a abordagem

do GenArch-P permitiu que a referida tarefa fosse realizada de forma mais rápida, do que com a abordagem do EPF *Composer*; restando assegurar a adequação da ANOVA.



```
Analysis of Variance Table

Response: tempo
                    Df Sum Sq Mean Sq F value  Pr(>F)
replica              5 337327   67465  2.4220 0.10962
sprl                 1 160067  160067  5.7463 0.03750 *
abordagem            1 138017  138017  4.9547 0.05019 .
replica:participante 6 309853   51642  1.8539 0.18529
Residuals           10 278556   27856
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figura 36. ANOVA para a primeira tarefa do experimento (T1)

Para verificar a adequação da utilização da tabela ANOVA para a análise dos resultados obtidos, foram analisados os resíduos desses resultados com relação aos pressupostos de Independência, Homocedasticidade e Normalidade (Hair et al., 2007). Os resíduos são obtidos por meio da diferença entre as observações individuais e a média da abordagem correspondente. A ferramenta R (Teetor, 2011), também utilizada na geração das tabelas ANOVA, possibilitou a geração dos gráficos que permitiram a análise dos resíduos dos resultados obtidos no experimento. A validade do pressuposto de Independência pôde ser verificada por meio dos gráficos de “Resíduos vs. Preditos”, nos quais, os resíduos deveriam estar distribuídos em torno de uma faixa horizontal centrada em 0 (zero) (Hair et al., 2007). A validade do pressuposto de Homocedasticidade pôde ser verificada por meio dos gráficos de “Resíduos vs. Abordagens”; nos quais, as abordagens envolvidas deveriam demonstrar variâncias semelhantes, com a média de resíduos próxima de zero (Hair et al., 2007). A validade do pressuposto de Normalidade pôde ser verificada por meio dos gráficos de probabilidade normal para os resíduos (Hair et al., 2007). Nesse gráfico, os resíduos são representados em função do seu valor esperado, o qual é calculado supondo que os resíduos seguem uma distribuição normal. Considera-se válida a suposição de Normalidade se os pontos do gráfico estiverem dispostos na forma de uma linha reta. Na visualização dessa reta, devem ser enfatizados os valores centrais do gráfico e não dos extremos (Hair et al., 2007). Para complementar a validação do pressuposto de Normalidade, foi realizado o teste proposto por Shapiro-Wilk (Field et al., 2012).

A análise dos resíduos para a primeira tarefa do experimento, verificou que embora os dois primeiros pressupostos (Independência e Homocedasticidade) fossem válidos, não foi possível validar o pressuposto de Normalidade. A não validação do

pressuposto de Normalidade foi comprovada pelo teste Shapiro-Wilk, que retornou um valor-p igual a 0,02648 (2%); o qual indica que não pode ser rejeitada a hipótese nula – de que a probabilidade dos resíduos não comportam como uma distribuição normal. Tal análise, inviabiliza uma conclusão, de base estatística, para os resultados apresentados pela ANOVA. Embora a análise dos resíduos indique a não adequação da tabela ANOVA para embasar as conclusões, a distribuição dos tempos (Figura 35) mostra que a abordagem GenArch-P obteve vários resultados pontuais melhores do que a abordagem EPF *Composer*.

7.6.1.2 Resultados da Segunda Tarefa de Modelagem da LPrS Alvo

A Figura 37 apresenta um sumário dos tempos obtidos na realização da modelagem de uma *feature* alternativa com as abordagens investigadas. Por meio do sumário dos resultados, é possível perceber a presença de um ponto discrepante (*outlier*). Ao analisar esse resultado discrepante, visando identificar as suas causas, percebeu-se que o participante que o gerou (participante1 da réplica 4) levou um tempo acima da média para realizar essa mesma tarefa com as duas abordagens. O mesmo levou 1476 segundos para concluir a segunda tarefa com o EPF *Composer* e 1593 segundos para concluir uma tarefa equivalente com o GenArch-P. Conclui-se que tal fato não compromete os resultados, visto tratar-se de uma característica específica do participante em questão.

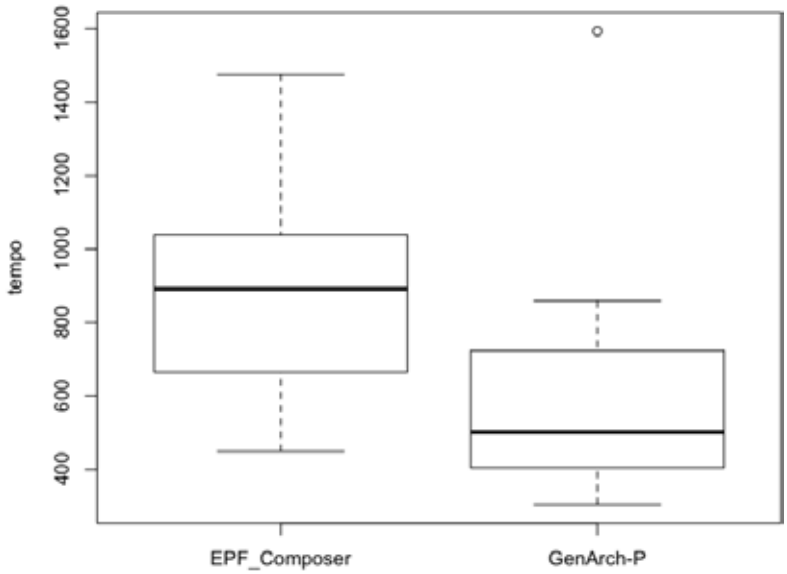
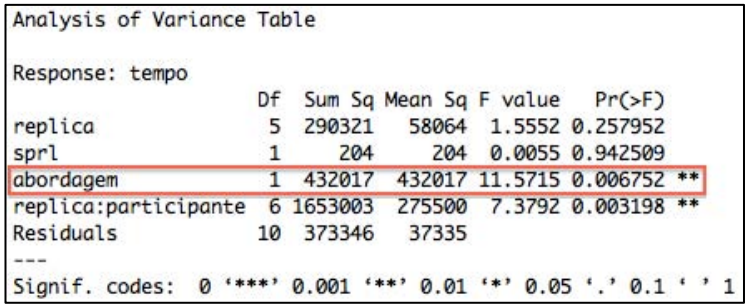


Figura 37. Sumário dos resultados para a segunda tarefa do experimento (T2)

A Figura 38 apresenta a tabela ANOVA para os resultados da segunda tarefa do experimento, a qual gerou um valor-p = 0,006752 (0,7%). Com um valor-p inferior ao desejado (5%), é possível rejeitar a hipótese nula em detrimento da hipótese de que a abordagem GenArch-P permitiu que a *feature* em questão fosse modelada de forma mais rápida, do que com a abordagem EPF *Composer*, caso a ANOVA seja adequada.



```
Analysis of Variance Table

Response: tempo
                     Df  Sum Sq Mean Sq F value   Pr(>F)
replica               5  290321   58064  1.5552 0.257952
sprl                  1     204     204  0.0055 0.942509
abordagem             1  432017  432017 11.5715 0.006752 **
replica:participante  6 1653003  275500  7.3792 0.003198 **
Residuals            10  373346   37335
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figura 38. ANOVA para a segunda tarefa do experimento (T2)

A análise dos pressupostos de Independência, Homocedasticidade e Normalidade, identificou a validade dos dois primeiros e um indício de falha do pressuposto de Normalidade. A comprovação veio com o teste de normalidade dos resíduos de Shapiro-Wilk, que gerou um valor-p igual a 0,0199 (2%); o qual não permite a rejeição da hipótese nula – de que os resíduos não se comportam como uma distribuição normal. Esse resultado não permite uma conclusão, com base estatística, sobre qual a abordagem mais rápida. Embora a análise dos resíduos indique a não adequação da tabela ANOVA para embasar as conclusões, a distribuição dos tempos (Figura 37) mostra que a abordagem GenArch-P obteve vários resultados pontuais melhores do que a abordagem EPF *Composer*.

7.6.1.3 Resultados da Terceira Tarefa de Modelagem da LPrS Alvo

A Figura 39 apresenta um sumário dos tempos para a modelagem da segunda *feature* opcional na LPrS alvo. Após a análise visual dos resultados, tem-se a impressão de que os tempos para as duas abordagens investigadas foi semelhante. Para confirmar essa impressão foi realizada a análise de variância desses resultados. A tabela ANOVA gerou um valor-p = 0,7627. Com um valor-p de 76%, não é possível rejeitar a hipótese nula em detrimento da hipótese de que ambas as abordagens possibilitaram a modelagem dessa *feature* opcional com tempos equivalentes.

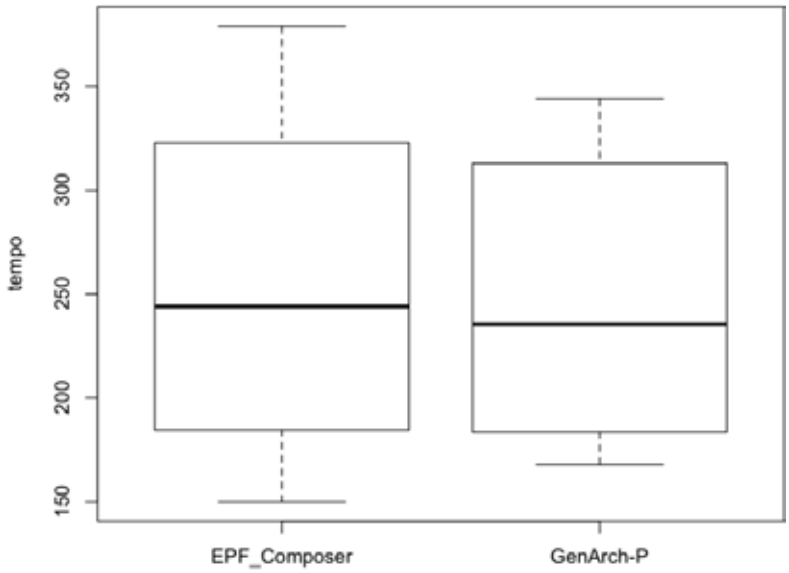


Figura 39. Sumário dos resultados da terceira tarefa do experimento (T3)

```
Analysis of Variance Table

Response: tempo
                     Df Sum Sq Mean Sq F value Pr(>F)
replica               5  12643  2528.6  0.5731 0.7200
sprl                  1     57    57.0  0.0129 0.9117
abordagem             1    425   425.0  0.0963 0.7627
replica:participante  6  69864 11644.0  2.6389 0.0843 .
Residuals            10  44124  4412.4
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figura 40. ANOVA para a terceira tarefa do experimento (T3)

Para validar a adequação da ANOVA para análise dos resultados, foram analisados os resíduos. Tal análise identificou a validade dos pressupostos de Independência, Homocedasticidade e Normalidade. Comprovando a validade do pressuposto de Normalidade, o teste de normalidade de Shapiro-Wilk gerou um valor-p igual a 0,9968. O valor-p de 99% permite rejeitar a hipótese nula – de que os resíduos não se comportam como uma distribuição normal. A validação dos referidos pressupostos atestam a adequação da ANOVA na análise dos resultados obtidos, nos quais conclui-se que as abordagens avaliadas propiciaram que essa tarefa específica fosse realiza com tempos equivalentes. Nenhuma delas proporcionou que a referida tarefa fosse concluída mais rapidamente.

7.6.1.4 Totalização dos Resultados para as Tarefas de Modelagem

Para possibilitar uma análise do tempo total das atividades de modelagem da LPrS alvo, foram totalizados os tempos para as três primeiras tarefas do experimento. Nessa totalização, constam os tempos de modelagem das duas *features* opcionais (T1 e

T3) e de modelagem da *feature* alternativa (T2). A Figura 41 apresenta um sumário dos tempos para as tarefas de modelagem da LPrS alvo. Segundo a análise visual do sumário dos resultados, foi percebido que a abordagem GenArch-P tem um ponto discrepante. A análise desse ponto discrepante, identificou que o mesmo foi gerado pelo mesmo motivo exposto na análise da segunda tarefa do experimento. Da mesma forma que anteriormente, esse ponto discrepante não invalida a análise dos resultados, visto se tratar de uma característica relacionada ao participante envolvido. A Figura 42 apresenta a tabela ANOVA para a totalização das tarefas de modelagem da LPrS alvo, a qual gerou um valor-p = 0,007823. O valor-p de 0,8% permite a rejeição da hipótese nula, em favor da hipótese de que a abordagem GenArch-P possibilitou que a modelagem da LPrS alvo fosse realizada em menos tempo, do que a abordagem EPF *Composer*.

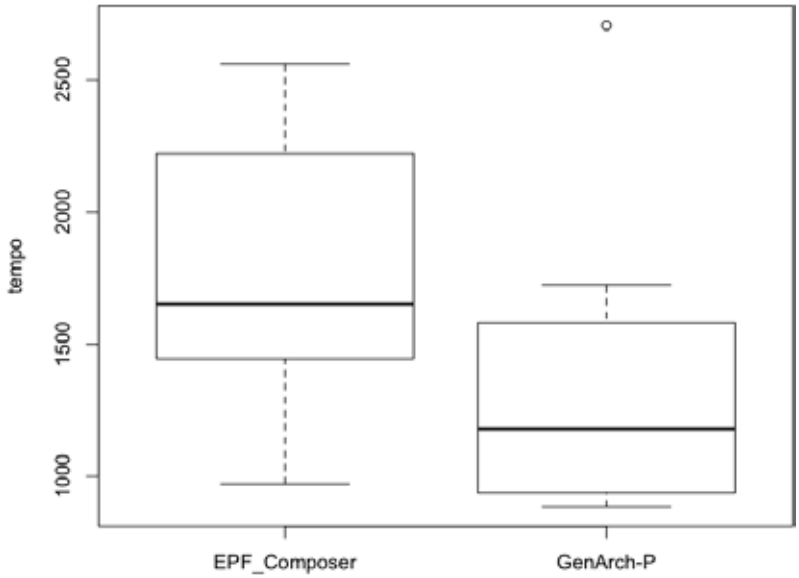


Figura 41. Sumário dos resultados para as tarefas de modelagem da LPrS

```
Analysis of Variance Table

Response: tempo
                     Df  Sum Sq Mean Sq F value   Pr(>F)
replica               5  903585  180717  1.9358 0.174988
sprl                  1  148365  148365  1.5892 0.236057
abordagem             1 1025480 1025480 10.9845 0.007823 **
replica:participante  6 3856817  642803  6.8854 0.004146 **
Residuals            10  933572   93357
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figura 42. ANOVA para as tarefas de modelagem da LPrS

Importante notar que não foi possível rejeitar as hipóteses nulas quando as tarefas de modelagem das *features* opcionais foram analisadas de forma isolada. Já quando as tarefas de modelagem das *features* opcionais são consideradas no computo

do tempo total de modelagem – é possível concluir que a abordagem GenArch-P apresenta uma vantagem com relação a abordagem do EPF *Composer*. Para validar a adequação da ANOVA na análise da totalização dos tempos das tarefas de modelagem da LPrS alvo, procedeu-se a análise dos resíduos. Tal análise identificou a validade dos pressupostos de Independência, Homocedasticidade e Normalidade. Comprovando a validade do pressuposto de Normalidade, o teste Shapiro-Wilk obteve um valor-p igual a 0,06744. Com um valor-p de 6,7% foi possível rejeitar, dentro da significância desejada (5%), a hipótese nula, validando desta forma o pressuposto de Normalidade da probabilidade dos resíduos. Validados esses pressupostos, verifica-se a adequação da ANOVA na análise da totalização dos tempos das tarefas de modelagem da LPrS alvo. Desta forma, está validada a conclusão, com base nos resultados da ANOVA, de que a abordagem GenArch-P proporcionou que as tarefas de modelagem da LPrS alvo fossem realizadas em tempo menor do que com a abordagem EPF *Composer*.

7.6.1.5 Resultados da Primeira Tarefa de Alteração da LPrS Alvo

A Figura 43 apresenta um sumário dos tempos para a realização da primeira tarefa de alteração da LPrS alvo – exclusão de uma alternativa. A análise visual dos resultados gerou a impressão de que a abordagem EPF *Composer* possibilitou que essa tarefa fosse realizada mais rapidamente.

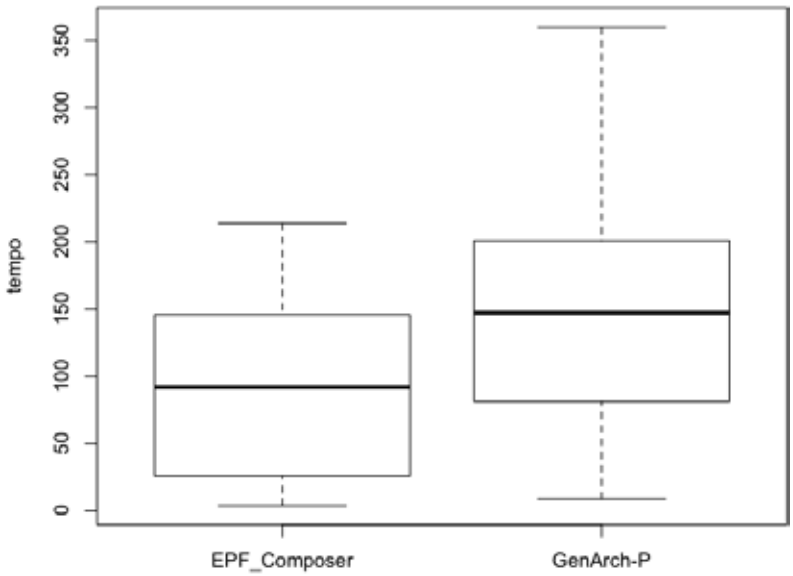
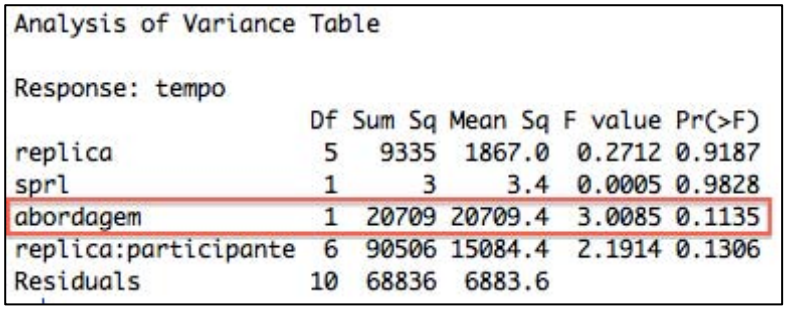


Figura 43. Sumário dos resultados da primeira tarefa de alteração (T4)

A Figura 44 apresenta a tabela ANOVA para os resultados da primeira tarefa de alteração da LPrS alvo, a qual apresentou um valor-p = 0,1135. O valor-p de 11% não

permite a rejeição da hipótese nula, o que evidencia que as abordagens possibilitaram a execução da referida tarefa em tempos equivalentes.



Analysis of Variance Table

Response: tempo

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
replica	5	9335	1867.0	0.2712	0.9187
sprl	1	3	3.4	0.0005	0.9828
abordagem	1	20709	20709.4	3.0085	0.1135
replica:participante	6	90506	15084.4	2.1914	0.1306
Residuals	10	68836	6883.6		

Figura 44. ANOVA para a primeira tarefa de alteração da LPrS (T4)

Visando atestar a adequação da ANOVA para a análise dos resultados da primeira tarefa de alteração da LPrS alvo, foram analisados os resíduos. Tal análise indicou a validade dos pressupostos de Independência, Homocedasticidade e Normalidade. Comprovando a validade do princípio de Normalidade, o teste Shapiro-Wilk obteve um valor-p igual a 0,9929 (99%); o qual permite a rejeição da hipótese nula – de que os resíduos não se comportam com uma distribuição normal. A validade desses pressupostos atestam a adequação da ANOVA na análise dos resultados para a primeira tarefa de alteração da LPrS alvo.

7.6.1.6 Resultados da Segunda Tarefa de Alteração da LPrS Alvo

A Figura 45 apresenta um sumário dos tempos que os levaram para realizar a segunda tarefa de alteração da LPrS alvo – desassociar elementos de processos relacionados a uma dada *feature*. A análise visual do sumário dos tempos dessa tarefa, gera a impressão de que o EPF *Composer* possibilitou a realização da tarefa num tempo menor, do que com o GenArch-P. A Figura *46* apresenta a tabela ANOVA para análise dos resultados da segunda tarefa de alteração da LPrS alvo, a qual gerou um valor-p = 0,9319. O valor-p de 93% não possibilita rejeitar a hipótese nula, o que evidencia que ambas as abordagens possibilitaram a conclusão da tarefa em tempos equivalentes.

Para validar a adequação da ANOVA para a análise dos resultados para a segunda tarefa de alteração da LPrS alvo, procedeu-se com a análise dos resíduos. Tal análise, permitiu identificar a validade dos pressupostos de Independência, Homocedasticidade e Normalidade. Comprovando a validade do pressuposto de Normalidade, o teste Shapiro-Wilk obteve um valor-p igual a 0,9478 (94%); o qual

possibilita que seja rejeitada a hipótese nula – de que os resíduos não se comportam como uma distribuição normal. A validação desses pressupostos, atestam a validade da ANOVA na análise desses resultados, embora que não possibilite a conclusão em prol de uma das abordagens investigadas.

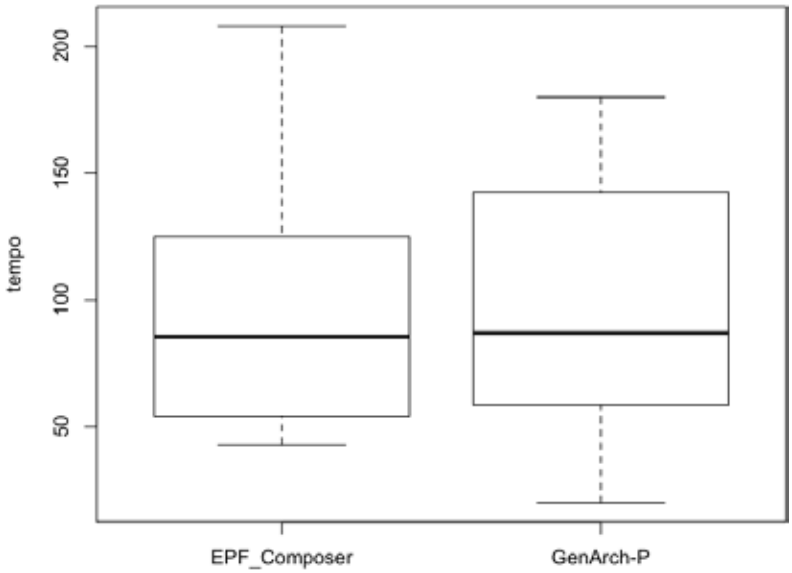


Figura 45. Sumário dos resultados da segunda tarefa de alteração da LPrS (T5)

```
Analysis of Variance Table

Response: tempo
                     Df Sum Sq Mean Sq F value Pr(>F)
replica               5   2671   534.2  0.1577 0.9726
sprl                  1     18    18.4  0.0054 0.9427
abordagem             1     26    26.0  0.0077 0.9319
replica:participante  6  27524  4587.4  1.3541 0.3198
Residuals            10  33877  3387.7
```

Figura 46. ANOVA para a segunda tarefa de alteração da LPrS (T5)

7.6.1.7 Resultados da Terceira Tarefa de Alteração da LPrS Alvo

A Figura 47 apresenta um sumário dos tempos que os participantes levaram para a conclusão da terceira tarefa de alteração da LPrS alvo – inclusão de uma nova alternativa. A análise visual do sumário dos resultados, gera a impressão de que as abordagens possibilitaram tempos equivalentes para a conclusão da referida tarefa, com uma leve vantagem para a abordagem GenArch-P. A Figura 48 apresenta a tabela ANOVA para os resultados da referida tarefa, a qual apresenta um valor-p = 0,1701. O valor-p de 17% não possibilita a rejeição da hipótese nula, a qual atesta que os tempos proporcionados pelas abordagens investigadas podem ser considerados equivalentes.

Desta forma, os resultados apontam novamente para uma equivalência entre as abordagens identificadas, possibilitando realizar as tarefas solicitadas em tempos similares. Antes da conclusão definitiva, é necessários analisar os resíduos e atestar a adequabilidade da ANOVA para o referido conjunto de resultados.

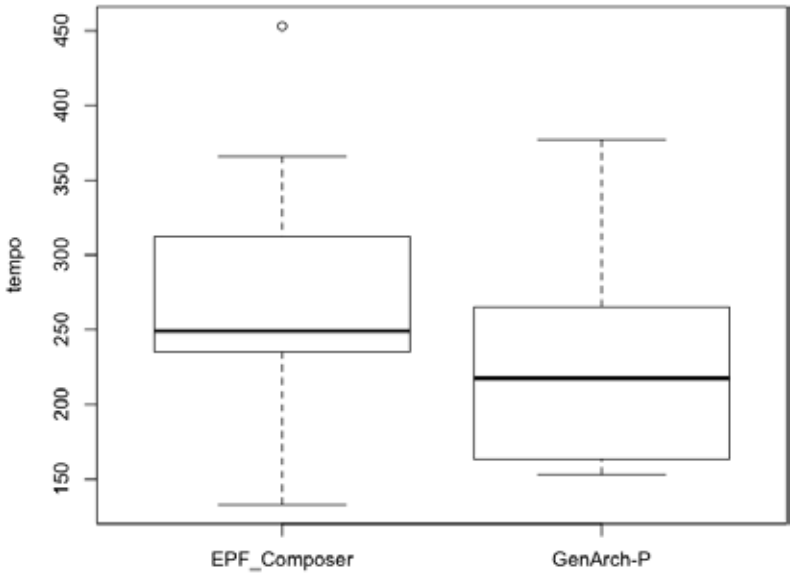


Figura 47. Sumário dos resultados da terceira tarefa de alteração da LPrS (T6)

```
Analysis of Variance Table

Response: tempo
                     Df Sum Sq Mean Sq F value Pr(>F)
replica               5  43767  8753.4  1.5982 0.2467
sprl                  1   2688  2688.2  0.4908 0.4995
abordagem             1  11971 11970.7  2.1857 0.1701
replica:participante  6  24165  4027.6  0.7354 0.6331
Residuals            10  54769  5476.9
```

Figura 48. ANOVA para a terceira tarefa de alteração da LPrS (T6)

Para validar a adequação da ANOVA para análise dos resultados da terceira tarefa de alteração da LPrS alvo foi realizada a análise dos resíduos dos resultados obtidos. Tal análise identificou a validade dos pressupostos de Independência, Homocedasticidade e Normalidade. Comprovando a validade do pressuposto de Normalidade, o teste Shapiro-Wilk obteve um valor-p igual a 0,9983 (99%); o qual possibilita rejeitar a hipótese nula – de que os resíduos não se comportam como uma distribuição normal.

A validade desses pressupostos atesta a adequação da ANOVA na análise dos resultados referentes a essa tarefa específica do experimento, embora que não ateste a vantagem de uma dada abordagem em detrimento da outra.

7.6.1.8 Resultados da Quarta Tarefa de Alteração da LPrS Alvo

A Figura 49 apresenta um sumário dos tempos levados pelos participantes do experimento para concluir a quarta tarefa de alteração da LPrS alvo – associação de novos elementos de processo às *features* existentes. A análise visual do sumário dos resultados para a referida tarefa do experimento, gera a impressão de que a abordagem GenArch-P possibilitou que a referida tarefa fosse realizada de forma mais rápida, do que a abordagem do EPF *Composer*. A Figura 50 apresenta a tabela ANOVA para análise dos resultados para essa referida tarefa do experimento, a qual apresenta um valor-p = 0,1143. O valor-p de 11% não permitiu que fosse rejeitada a hipótese nula, a qual indica que as abordagens proporcionaram tempos equivalentes para a conclusão dessa tarefa. Antes da conclusão definitiva, é necessário analisar os resíduos e comprovar que os mesmos atendem aos pressupostos básicos, os quais atestam a adequação do uso ANOVA para análise dos respectivos resultados.

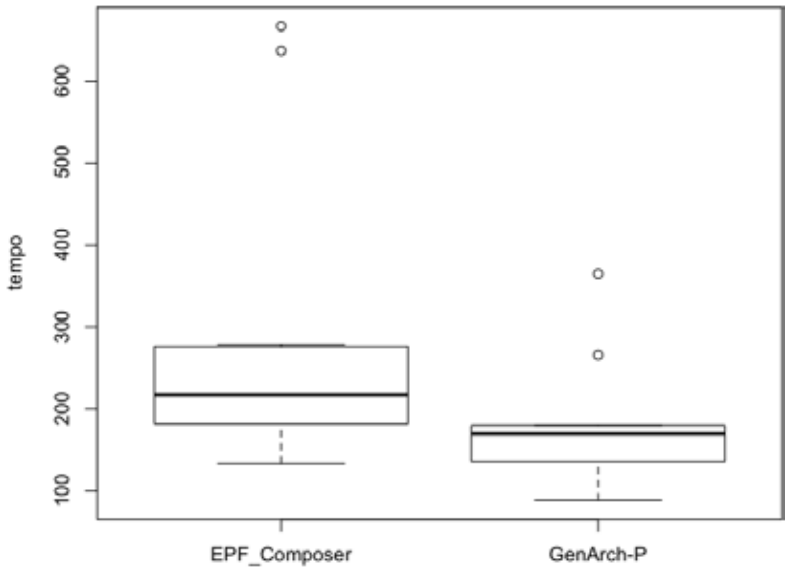


Figura 49. Sumário dos resultados da quarta tarefa de alteração da LPrS (T7)

```
Analysis of Variance Table

Response: tempo
                     Df Sum Sq Mean Sq F value Pr(>F)
replica               5  41438    8288  0.3522 0.8695
sprl                  1  13443   13443  0.5714 0.4672
abordagem             1  70417   70417  2.9929 0.1143
replica:participante  6 121677   20280  0.8619 0.5533
Residuals            10 235279   23528
```

Figura 50. ANOVA para a quarta tarefa de alteração do experimento (T7)

Para atestar a adequação da ANOVA para os resultados obtidos pela quarta tarefa de alteração da LPrS alvo, procedeu-se a análise dos resíduos desses resultados. Tal análise identificou a validade dos pressupostos de Independência, Homocedasticidade e Normalidade. Comprovando a validade do pressuposto de Normalidade, o teste Shapiro-Wilk retornou um valor-p igual a 0,9423 (94%); o qual possibilita a rejeição da hipótese nula – de que os resíduos não se comportam como uma distribuição normal. A validação desses pressupostos atestam a adequação da ANOVA para os resultados da referida tarefa.

7.6.1.9 Totalização dos Resultados para as Tarefas de Alteração

Para possibilitar uma visão geral das tarefas de alteração da LPrS alvo utilizando as duas abordagens investigadas, os tempos das tarefas correspondentes (T4 a T7) foram totalizados. A Figura 51 apresenta um sumário dos tempos totalizados para as tarefas de alteração do experimento. A análise visual do sumário dos tempos para as tarefas de alteração, geram a impressão de que os referidos tempos foram equivalentes para ambas abordagens investigadas, com uma leve vantagem para a abordagem GenArch-P.

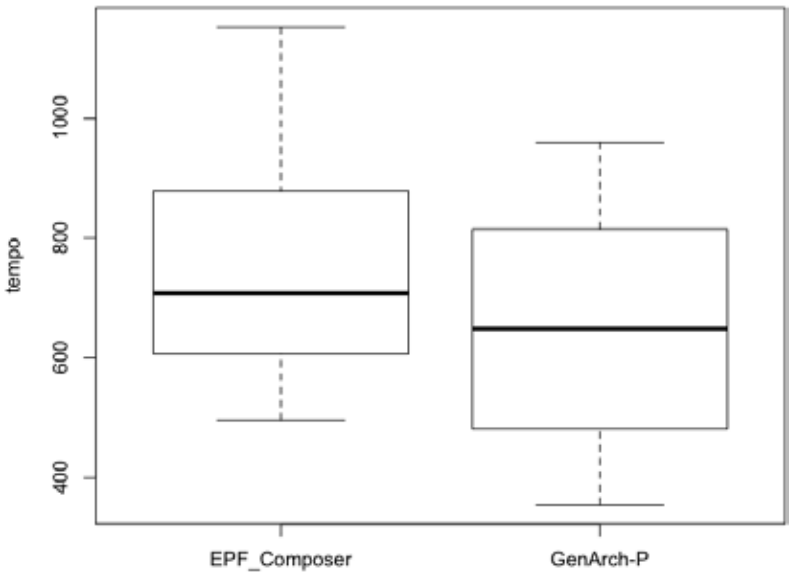


Figura 51. Sumário dos resultados para as tarefas de alteração da LPrS

A Figura 52 apresenta a tabela ANOVA para os resultados das tarefas de alteração da LPrS alvo, a qual apresenta um valor-p = 0,12969. O valor-p de 13% não possibilita a rejeição da hipótese nula – indicando que os tempos proporcionados pelas abordagens investigadas para a realização das tarefas de alteração foram equivalentes. Dessa forma, não pode ser demonstrado que uma das abordagens possui uma vantagem

em relação a outra. Esse resultado pode ter sido causado pela natureza pontual das tarefas de alteração da modelagem da LPrS, propostas para o experimento.

```
Analysis of Variance Table

Response: tempo
                    Df Sum Sq Mean Sq F value  Pr(>F)
replica              5  99013   19803  0.9698 0.48033
sprl                 1  27337   27337  1.3388 0.27414
abordagem            1  55681   55681  2.7268 0.12969
replica:participante 6 514888   85815  4.2025 0.02266 *
Residuals           10 204200   20420
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Figura 52. ANOVA para as tarefas de alteração da LPrS

Uma das análises possíveis para os resultados obtidos, é que a quantidade de amostras utilizada não permitiu uma conclusão dessa questão. Outra forma de analisar tais resultados, é concluindo que uma vez modelada uma LPrS, as abordagens investigadas possibilitam que alterações na modelagem da LPrS sejam realizadas em tempos equivalentes.

Visando atestar a adequação da ANOVA para os resultados totalizados das tarefas de alteração propostas para o experimento, procedeu-se a análise dos resíduos referentes a esses resultados. Tal análise possibilitou identificar a validade dos pressupostos de Independência, Homocedasticidade e Normalidade. Comprovando o pressuposto de Normalidade, o teste Shapiro-Wilk retornou um valor-p igual a 0,947 (94%); o qual possibilita a rejeição da hipótese nula – de que os resíduos não se comportam como uma distribuição normal. A validação destes pressupostos, atesta a adequação da ANOVA na análise dos resultados da totalização das tarefas de alteração da LPrS alvo, embora que com a mesma não tenha sido possível afirmar que há uma vantagem (menor tempo) na utilização de uma abordagem em detrimento de outra.

7.6.2 Influência da Experiência dos Participantes e das LPrSs Alvo

Para analisar a influência da experiência dos participantes e das LPRS Alvo nos resultados obtidos, foi tomado por base a significância calculada para esses parâmetros nas tabelas de análise de variância ANOVA. A Tabela 23 apresenta para cada uma das tarefas do experimento, bem como as totalizações das tarefas de modelagem e alteração, as variáveis controladas ordenadas pela significância calculada nas tabelas ANOVA. As variáveis de maior significância e influência nos resultados aparecem primeiro.

Tabela 23. Ordem de significância das variáveis controladas nas tabelas ANOVA

Tarefa	Variáveis controladas ordenadas por significância	Adequação da ANOVA
Tarefa 1 (modelagem)	LPrS > Abordagem > Réplica > **Participante** > Resíduos	Não
Tarefa 2 (modelagem)	Abordagem > **Participante** > Réplica > Resíduos > LPrS	Não
Tarefa 3 (modelagem)	**Participante** > Réplica > Resíduos > Abordagem > LPrS	Sim
Tarefas de modelagem	Abordagem > **Participante** > Réplica > LPrS > Resíduos	Sim
Tarefa 4 (alteração)	Abordagem > **Participante** > Resíduos > Réplica > LPrS	Sim
Tarefa 5 (alteração)	**Participante** > Resíduos > Réplica > Abordagem > LPrS	Sim
Tarefa 6 (alteração)	Abordagem > Réplica > Resíduos > **Participante** > LPrS	Sim
Tarefa 7 (alteração)	Abordagem > Resíduos > **Participante** > LPrS > Réplica	Sim
Tarefas de alteração	**Participante** > Abordagem > LPrS > Resíduos > Réplica	Sim

Para compreender a influência da experiência dos participantes nos resultados obtidos, foi analisado o grau de significância da variável "replica:participante" em cada uma das tabelas ANOVA (Hair et al., 2007) geradas a partir dos resultados obtidos no experimento. Por meio dessa análise, foi possível perceber que tal influência existe, mesmo que varie um pouco entre as tarefas solicitadas no experimento controlado. Das sete tarefas do experimento controlado, a significância da variável "participante" esteve entre as duas primeiras em quatro delas. Na totalização das tarefas de modelagem a significância da variável "participante" ficou em segundo lugar. E na totalização das tarefas de alteração a significância da variável "participante" ficou em primeiro lugar. Tais resultados permitem concluir que a experiência e o conhecimento prévio dos participantes influenciou nos resultados obtidos no experimento controlado. Concluímos também que a estratégia de linha de produtos de software foi fielmente mapeada para o contexto de processos de software, tanto que permitiu uma rápida familiarização daqueles que possuíam conhecimento prévio em linha de produtos de software.

Da mesma forma, para compreender a influência das LPrSs alvo que foram utilizadas nos resultados obtidos, foi analisada a significância da variável correspondente nas tabelas ANOVA. Por meio dessa análise, foi possível perceber que tal influência foi muito pequena. Das sete tarefas do experimento, a significância da variável "LPrS alvo" esteve entre as duas últimas em seis delas. Tais resultados permitem concluir que as LPrSs alvo utilizadas pelos participantes influenciaram muito pouco nos resultados obtidos pelo experimento.

7.6.3 Resultados Segundo o Aspecto de Corretude

Para a análise do aspecto de corretude foram verificadas as LPrSs resultantes de cada participante e averiguado se a abordagem em questão foi usada corretamente para

a solução de cada uma das tarefas propostas pelo experimento. A Tabela 24 ilustra o resultado da análise de corretude das tarefas realizadas por cada um dos participantes do experimento. Foram destacadas as tarefas nas quais a abordagem em questão não foi utilizada de forma correta.

Tabela 24. Tabela de análise da corretude na utilização das abordagens

1ª RODADA									
Participante	**Abordagem**	**LPrS**	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**
P1:Réplica1	Genarch-P	1	Ok	Ok	Ok	Ok	Ok	Ok	Ok
P2:Réplica1	EPF Comp.	1	Ok	Ok	Ok	Ok	Ok	Ok	Ok
P1:Réplica2	EPF Comp.	2	Erro	Ok	Ok	Ok	Erro	Ok	Ok
P2:Réplica2	Genarch-P	2	Ok	Ok	Ok	Ok	Ok	Ok	Ok
P1:Réplica3	Genarch-P	1	Ok	Ok	Ok	Ok	Ok	Ok	Ok
P2:Réplica3	EPF Comp.	1	Ok	Ok	Ok	Erro	Ok	Ok	Erro
P1:Réplica4	EPF Comp.	2	Ok	Ok	Ok	Ok	Ok	Ok	Ok
P2:Réplica4	Genarch-P	2	Ok	Erro	Ok	Ok	Ok	Ok	Erro
P1:Réplica5	Genarch-P	1	Ok	Ok	Ok	Ok	Ok	Ok	Ok
P2:Réplica5	EPF Comp.	1	Ok	Ok	Ok	Ok	Ok	Ok	Ok
P1:Réplica6	EPF Comp.	2	Ok	Ok	Ok	Ok	Ok	Ok	Ok
P2:Réplica6	Genarch-P	2	Ok	Ok	Ok	Ok	Ok	Ok	Ok
2ª RODADA									
P1:Réplica1	EPF Comp.	2	Ok	Ok	Ok	Ok	Ok	Ok	Ok
P2:Réplica1	Genarch-P	2	Ok	Erro	Ok	Erro	Erro	Ok	Ok
P1:Réplica2	Genarch-P	1	Ok	Ok	Ok	Erro	Ok	Ok	Ok
P2:Réplica2	EPF Comp.	1	Ok	Ok	Ok	Ok	Ok	Ok	Ok
P1:Réplica3	EPF Comp.	2	Ok	Ok	Ok	Ok	Ok	Ok	Ok
P2:Réplica3	Genarch-P	2	Ok	Ok	Ok	Ok	Ok	Ok	Ok
P1:Réplica4	Genarch-P	1	Ok	Ok	Ok	Ok	Ok	Ok	Ok
P2:Réplica4	EPF Comp.	1	Ok	Ok	Ok	Ok	Ok	Ok	Ok
P1:Réplica5	EPF Comp.	2	Ok	Ok	Ok	Ok	Ok	Ok	Ok
P2:Réplica5	Genarch-P	2	Ok	Ok	Ok	Ok	Ok	Ok	Ok
P1:Réplica6	Genarch-P	1	Ok	Ok	Ok	Ok	Ok	Ok	Ok
P2:Réplica6	EPF Comp.	1	Ok	Ok	Ok	Ok	Ok	Ok	Ok

Dos 12 (doze) participantes que utilizaram a abordagem EPF *Composer*, apenas 2 (dois) deles cometeram erros na execução das tarefas propostas no experimento. No caso do GenArch-P, dos 12 (doze) participantes apenas 3 (três) deles cometeram erros. Dos participantes que cometeram erros na utilização das abordagens, apenas um deles cometeu erros com as duas abordagens (participante 1 da réplica 2). Dado tratar-se de tarefas não realizadas anteriormente por nenhum dos participantes já era esperado que fossem cometidos erros na utilização das referidas abordagens. Por esse motivo, foram selecionadas tarefas mais simples e os treinamentos de utilização das abordagens foi mais criterioso. Dados os resultados apresentados na Tabela 24, a taxa de corretude calculada para a abordagem EPF *Composer* foi de 95% (80 tarefas corretas de 84), enquanto para a abordagem GenArch-P foi de 93% (78 tarefas corretas de 84).

Com taxas de corretude bem próximas uma da outra, e com números de participantes que cometeram erros também próximos, concluímos que ambas abordagens foram, no geral, bem compreendidas pelos participantes do experimento. Desta forma, podemos concluir também que ambas abordagens apresentaram um grau de complexidade moderado, o que permitiu que os participantes compreendessem a proposta de cada uma delas, e as aplicassem com sucesso na solução das tarefas propostas pelo experimento controlado.

7.7 Análises Qualitativas

Após a realização do experimento com cada uma das abordagens, os participantes responderam a um questionário de avaliação qualitativa da abordagem que acabaram de utilizar. O questionário possuía 7 (sete) questões ao todo. Nas cinco primeiras o participante respondia com um número de 1 a 5, onde "1" representava "muito ruim" e "5" representava "muito bom (boa)". As duas últimas questões foram discursivas e solicitavam ao participante escolher pontos fracos e fortes para a abordagem que acabou de utilizar. Esta seção apresenta uma consolidação das respostas dos participantes, bem como uma análise e discussão sobre as mesmas. Um dos objetivos dessa avaliação qualitativa é relacionar, sempre que possível, as suas respostas com os resultados da realização das tarefas de modelagem e alteração da LPrS alvo.

A primeira questão da avaliação qualitativa perguntou "como você qualifica o uso dessa abordagem para a realização das tarefas solicitadas?". A Figura 53 apresenta uma consolidação das respostas dos participantes para essa questão.

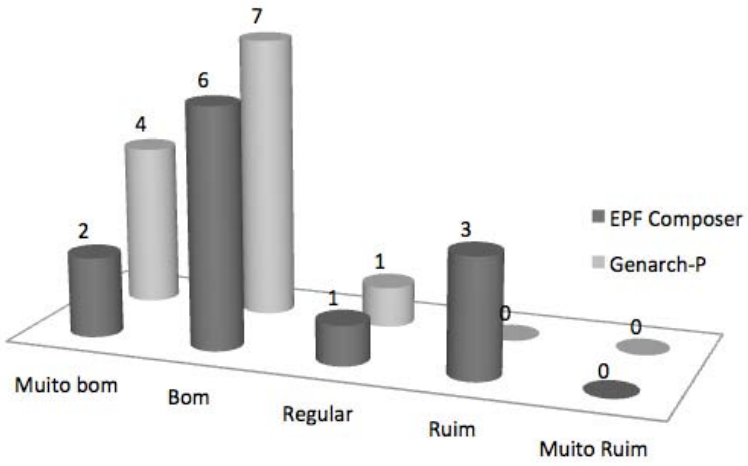


Figura 53. Respostas para a primeira questão da avaliação qualitativa

Nessa consolidação podemos perceber que a abordagem GenArch-P foi melhor avaliada, obtendo 11 (onze) respostas entre “muito boa” e “boa”, contra 8 (oito) do EPF *Composer*. Importante notar que a maioria dos resultados, para as duas abordagens, esteve entre “muito boa” e “boa”, significando que ambas cumpriram o seu papel e permitiram a modelagem de variabilidades em uma LPrS.

A segunda questão da avaliação qualitativa perguntou “como você qualifica a facilidade de entender como essa abordagem expressa o conhecimento de configuração (mapeamento entre *features* e elementos de processo)?”. A Figura 54 apresenta uma consolidação das respostas dos participantes para essa questão. A abordagem do GenArch-P obteve melhores resultados segundo a avaliação dos participantes nessa questão; foram 9 (nove) respostas entre “muito boa” e “boa”, contra 6 (seis) do EPF *Composer*. Embora com uma melhor avaliação no geral, um dos participantes julgou a forma de representar o conhecimento de configuração com o GenArch-P “ruim”, o que não aconteceu com o EPF *Composer*. Esse fato indica que o GenArch-P gerou dificuldades para pelo menos um participante, possivelmente por ter uma curva de aprendizado mais acentuada.

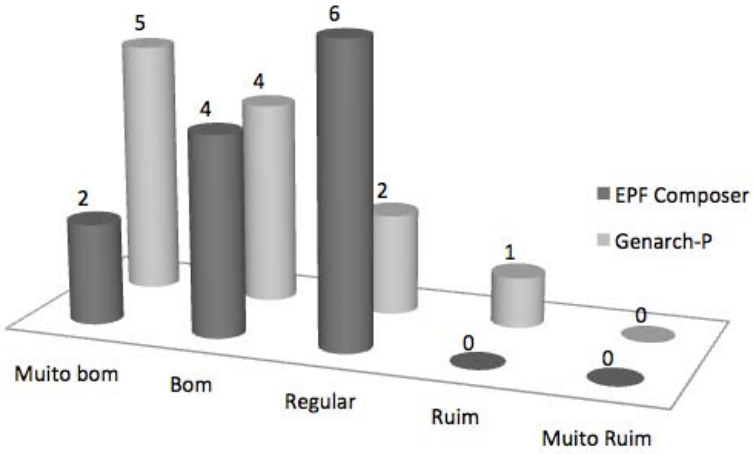


Figura 54. Respostas para a segunda questão da avaliação qualitativa

A terceira questão do questionário de avaliação qualitativa perguntou “como você qualifica o acesso (por meio de pesquisas) ao conhecimento de configuração?”. Os participantes foram instruídos desde o treinamento que o conhecimento de configuração era representado pelo mapeamento de *features* para os elementos de processo. Para responder a essa questão, os mesmos foram instruídos a avaliar “como foi o acesso aos elementos de processo associados, dado uma *feature*” e “como foi o acesso à *feature*

correspondente, dado um elemento de processo". A Figura 55 apresenta uma consolidação das respostas dos participantes para essa questão.

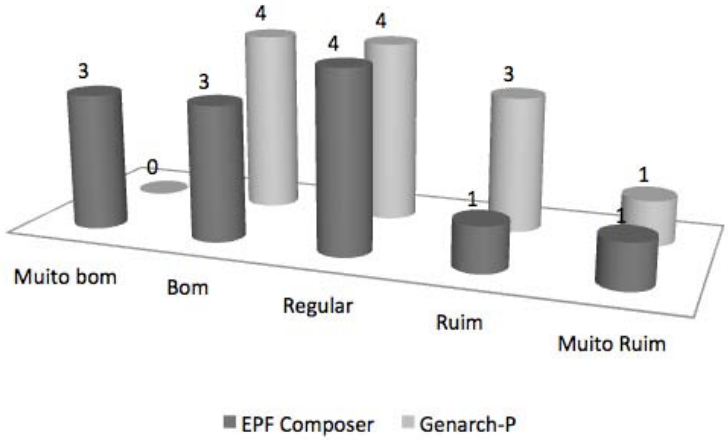


Figura 55. Respostas para a terceira questão da avaliação qualitativa

O EPF *Composer* teve 6 (seis) respostas entre "muito bom" e "bom", contra 4 (quatro) do GenArch-P; indicando que o mesmo oferece um melhor acesso ao conhecimento de configuração. Importante notar que 4 (quatro) participantes qualificaram o acesso ao conhecimento de configuração proporcionado pelo GenArch-P como "ruim" ou "muito ruim", contra 2 (dois) do EPF *Composer*. A avaliação negativa do GenArch-P, segundo relato dos participantes, foi devido ao fato da ferramenta não "destacar" os elementos de processo associados a uma dada *feature*. A descrição da ferramenta cita a utilização de cores para destacar os elementos de processo anotados com uma dada *feature*, mas a versão utilizada não possuía tal funcionalidade.

A quarta questão da avaliação qualitativa perguntou "como você qualifica a tarefa de explicar para um leigo o resultado da modelagem de uma LPrS com essa abordagem?". A Figura 56 apresenta uma consolidação das respostas dos participantes para essa questão. Com relação ao número respostas entre "muito bom" e "bom", percebemos 5 (cinco) respostas para o EPF *Composer*, contra 4 (quatro) para o GenArch-P; concluímos que um número maior de participantes entendeu melhor a proposta do EPF *Composer*. Porém, se for considerado o intervalo entre "muito bom" e "regular"; percebemos 11 (onze) respostas para o GenArch-P, contra 8 (oito) para o EPF *Composer*. Dados esses dois pontos de vista, concluímos que ambas abordagens foram bem compreendidas, embora também tenham apresentado dificuldades significativas. Uma análise mais profunda será realizada durante a discussão dos pontos fortes e fracos identificados pelos participantes do experimento.

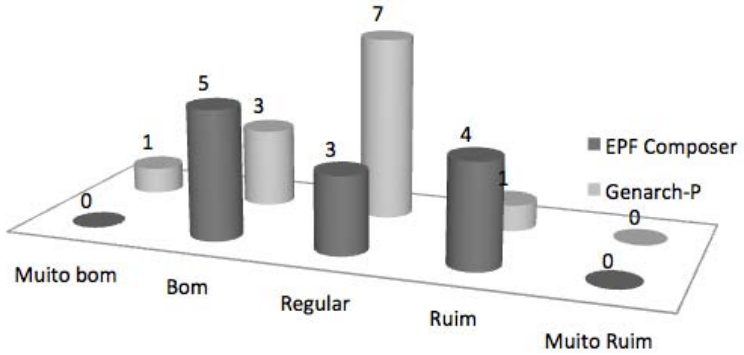


Figura 56. Respostas para a quarta questão da avaliação qualitativa

A quinta questão do questionário de avaliação qualitativa perguntou “como você recomendaria essa abordagem para um colega interessado em realizar tarefas equivalentes?”. A Figura 57 apresenta uma consolidação das respostas para tal questão. Analisando as resposta dos participantes para essa questão, temos que o GenArch-P teve 8 (oito) respostas entre “muito boa” e “boa”, contra 6 (seis) do EPF *Composer*. Esse resultado geral, indica que embora o GenArch-P tenha obtido uma leve vantagem com relação ao EPF *Composer*; as duas abordagens obtiveram resultados equivalentes. O fato das duas abordagens terem obtido resultados equivalentes, possibilita a conclusão de que ambas deram suporte, no mínimo regular para as tarefas solicitados no experimento. Relacionando as duas últimas questões da avaliação qualitativa, podemos deduzir que os participantes que tiveram um pouco mais dificuldades no entendimento do EPF *Composer*, recomendaram a mesma de forma negativa; o que não aconteceu com o GenArch-P.

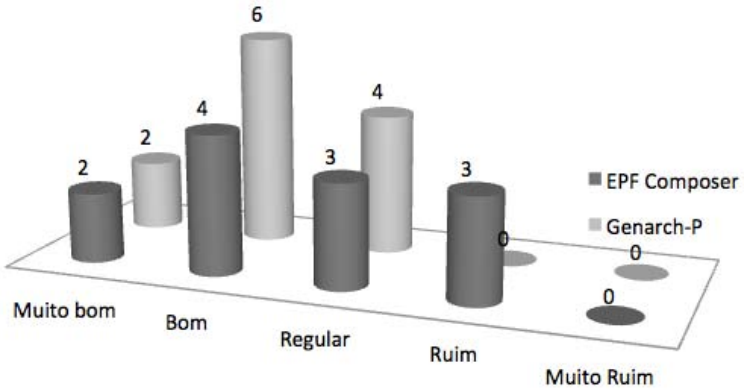


Figura 57. Respostas para a quinta questão da avaliação qualitativa

As duas últimas questões permitiram que o participante descrevesse com as suas palavras os pontos fortes e fracos da abordagem que acabou de utilizar. As respostas dos mesmos foram consolidadas em duas tabelas, uma para cada abordagem sendo

investigada. Aos participantes foi solicitado informar pelo menos um ponto forte e um ponto fraco, mas ficaram livres para apresentar mais de uma observação por categoria. A Tabela 25 sumariza os pontos fortes e fracos identificados pelos participantes do experimento para o EPF *Composer*. As observações estão ordenadas pelo número de vezes em que as mesmas foram citadas pelos participantes.

Analisando as respostas dos participantes, é possível chegar a algumas conclusões. O fato do EPF *Composer* estar inserido na plataforma Eclipse é um aspecto positivo, visto que ofereceu aos participantes um ambiente familiar e amigável, como o sistema de busca. Outro ponto importante, que já foi destacado em estudos anteriores (Aleixo et al., 2012a) (Aleixo et al., 2012b) (Aleixo et al., 2013), a modularidade proporcionada pelo EPF *Composer* na modelagem das variabilidades torna essa tarefa mais fácil. Estando os elementos de processo, associados à uma *feature* específica, agrupados e isolados em um pacote de conteúdo. Este fato possibilitou aos participantes um rápido e fácil acesso a tais elementos (conhecimento de configuração). O fato de permitir a edição da especificação do processo de software, e dos seus elementos, foi um outro aspecto considerado como positivo para o EPF *Composer*; embora essa tarefa demande um tempo adicional até o usuário entender a forma de organização dos elementos que compõem o processo.

Tabela 25. Pontos fortes e fracos identificados para a abordagem EPF Composer

Pontos Fortes	**Pontos Fracos**
Sistema de busca por elementos de processo é bem eficiente (6)	A utilização de caracteres curinga na especificação da pesquisa desejada não é simples (5)
Fácil definição de variabilidades e agrupamento dos elementos de processo relacionados – mover e copiar para pacotes de conteúdo (6)	Dificuldade em desfazer associação de elementos de processo à *features* específicas (5)
Interface amigável (2)	Impossibilidade de definição dos tipos de variabilidade – opcional, alternativa, etc. (4)
Organização dos elementos por tipo (2)	Complexidade para mover elementos de processo (1)
Possibilidade de visualização e edição do conteúdo dos elementos de processo (1)	O conceito de mover elementos de processo parece um pouco estranho (1)
Fácil de trabalhar após alguns treinos (1)	A organização em pacotes de conteúdo dificulta a visão global do processo – implica na necessidade de utilização do sistema de busca (1)

Alguns aspectos negativos foram identificados para o EPF *Composer*, dentre eles, um dos principais foi o fato da não possibilidade de definição explícita das *features*. O EPF *Composer* também não possibilitou a definição das propriedades dessas *features*, tais como: tipo, hierarquia e dependência. Esse fato implicou que o usuário ficou responsável por manter e aplicar o conhecimento de configuração; estratégia esta

propensa a falhas e erros. Ao mesmo tempo que a distribuição dos elementos de processo pelos pacotes de conteúdo, associados à *features* específicas, facilita a visualização e acesso aos mesmos; gera uma dificuldade quando deseja-se desfazer essa associação. Nesse caso, o elemento de processo precisa ser movido para um outro pacote de conteúdo (representando o conteúdo mandatório), ou mesmo excluído da especificação do processo. Um outro aspecto negativo é que as ações de mover e copiar elementos de processo entre pacotes de conteúdo, é propensa a erros – um elemento pode ser apagado por engano, ou mesmo copiado (ou movido) para o lugar errado. Da mesma forma como foi feito para o EPF *Composer*, as respostas dos participantes com relação ao GenArch-P foram consolidadas em uma tabela. A Tabela 26 apresenta os pontos fortes e fracos identificados para a abordagem do GenArch-P.

Tabela 26. Pontos fortes e fracos identificados para a abordagem GenArch-P

Pontos Fortes	**Pontos Fracos**
Utilização de anotações para relacionar elementos de processo à *features* específicas (6)	Não há validação, nem auxílio, na definição das expressões de *features* (5)
Fácil entendimento (5)	Busca de elementos de processo é custosa (5)
Interface gráfica amigável - visualização dos elementos de processo (4)	Processo de remoção de feature demorado, pois implica na remoção individual das expressões de *features* associadas (3)
Flexibilidade proporcionado pelo uso de expressões de *features* (2)	Dificuldade de localizar as referências a uma determinada *feature* (3)
Visualização do modelo de *features* (2)	Mais difícil de compreender, a princípio (2)
	Não é possível definir *features* mandatórias (1)
	Trabalhar simultaneamente com dois modelos (1)

A análise das respostas dos participantes permitiu a identificação de alguns aspectos positivos do GenArch-P, o principal deles foi com relação à utilização de anotações para associar elementos de processo à *features* específicas. A gerência de variabilidades segundo essa abordagem utiliza apenas um mecanismo – anotação de elementos de processo – o que facilita o entendimento da abordagem. O modelo de processo disponibilizado pelo GenArch-P apresenta os elementos de processo agrupados pelos seus tipos, facilitando a visualização dos mesmos. A associação de um elemento de processo a uma *feature* específica é feita através da anotação do referido elemento com uma expressão de *features*. As expressões de *features* permitem que sejam definidas associações de um dado elemento de processo à várias *features*, bem como a definição de dependência entre *features*. A visualização do modelo de *features* é um outro aspecto positivo do GenArch-P, pois dá uma idéia clara de todos os possíveis processos de software que podem ser derivados a partir da LPrS.

Por outro lado, alguns aspectos negativos também foram identificados para o GenArch-P, sendo o principal deles o fato de não haver validações, nem auxílios, no momento da definição das expressões de *features*. Um outro aspecto negativo que foi percebido pelos participantes do experimento foi o fato de que a única opção de busca oferecida pelo GenArch-P ser a busca no modelo de processo (XMI) e não na representação visual que o usuário trabalha (editor visual do modelo). Uma outra funcionalidade ausente no GenArch-P que fez falta aos participantes do experimento foi a possibilidade de destacar (com cores, ou algo do gênero) os elementos de processo associados a uma dada *feature*. A localização manual, ou mesmo por meio de pesquisas, dos elementos associados a uma dada *feature* foi considerada custosa pelos participantes. Alguns participantes indicaram à dificuldade inicial de entender a abordagem; mas esse fato foi atribuído a inexperiência de alguns participantes nos conceitos fundamentais de linhas de produtos de software.

A impossibilidade de definições de *features* mandatórias foi um problema para um dos participantes, visto que é possível que em uma das evoluções de uma LPrS uma dada *feature* mandatória seja alterada para opcional. O fato de trabalhar com dois modelos – modelo de processo e modelo de *features* – também foi um problema para um dos participantes do experimento. Dado que o modelo de *features* poderia ser gerado automaticamente a partir da definição de características das *features* no próprio modelo de processo, durante o processo de modelagem das variabilidades.

7.8 Conclusões do Estudo e Trabalhos Futuros

Este capítulo relatou a motivação, planejamento, execução e análise dos resultados de um estudo empírico, na forma de experimento controlado, visando a comparação de duas abordagens para a gerência de variabilidades em LPrS. O experimento foi executado conforme planejado e possibilitou a comparação das abordagens nos aspectos de modelagem de uma LPrS alvo, e de alterações específicas na modelagem dessa LPrS. Como conclusão geral para o capítulo, temos que ficou evidenciado que ambas abordagens – EPF *Composer* (composicional) e GenArch-P (anotativa) – possibilitaram a gerência de variabilidades em uma LPrS. Visando responder às questões de pesquisa definidas para o experimento, foram analisados os resultados da execução do experimento.

Com relação a primeira questão de pesquisa, os resultados permitiram concluir que o GenArch-P, de fato, possibilitou a modelagem da LPrS alvo em menos tempo (H1) do que com o EPF *Composer*. Também foram confrontados os resultados da execução das tarefas propostas para o experimento, com as respostas do questionário de avaliação qualitativa das abordagens utilizadas. A partir dessa análise, foi possível conjecturar algumas novas hipóteses sobre as suas possíveis causas, são elas: (i) a utilização de um único mecanismo para a definição de variabilidades – anotação dos elementos de processo – fez a diferença para a abordagem do GenArch-P; e (ii) as tarefas de distribuir – copiando e movendo – os elementos de processo pelos pacotes de conteúdo representando *features* específicas se mostrou um pouco complexa.

Com relação a segunda questão de pesquisa, os resultados obtidos acabaram por rejeitar a hipótese de que o GenArch-P possibilitaria que tarefas de alteração na modelagem da LPrS alvo fossem realizadas em menos tempo, do que com o EPF *Composer*. Os resultados evidenciaram tempos equivalentes para as tarefas de alteração da LPrS alvo, realizadas utilizando-se as duas abordagens (H0). A confrontação desse resultado com as respostas dos participantes na avaliação qualitativa permitiu conjecturar novas hipóteses sobre as causas desse fato. As hipóteses sobre as possíveis causas desse resultado são: (i) o GenArch-P não ofereceu nenhuma funcionalidade para destacar os elementos associados a *features* específicas – fato esse que comprometeu o acesso ao conhecimento de configuração; (ii) embora bem modularizados os elementos associados às *features* específicas – as ações de copiar e mover elementos, para desassociá-los dessas *features* se mostraram complexas.

Visando responder à terceira questão de pesquisa, foram analisadas as respostas dos participantes do experimento na avaliação qualitativa, realizada após a execução das tarefas propostas para o experimento. Foi possível concluir que, de fato, ambas as abordagens, no geral, se mostraram fáceis de entender e utilizar. É verdade que alguns participantes apresentaram dificuldades pontuais, mas essas podem ser justificadas em termos dos pontos fracos identificados para as abordagens. Analisando as respostas à avaliação qualitativa, foi possível definir as necessidades e expectativas de um usuário ao especificar uma LPrS, independentemente da abordagem que utilizar. Essas necessidades e expectativas identificadas foram: (i) definir explicitamente as *features* e suas propriedades; (ii) associar, e desassociar, elementos de processo à *features* específicas de forma simples e rápida; e (iii) visualizar e navegar no conhecimento de

configuração (mapeamento bidirecional entre elementos de processos e *features* da LPrS) de forma fácil. Dado que uma abordagem qualquer atenda a tais necessidades e expectativas, ela terá boas chances de ser bem sucedida.

Dadas as necessidades e expectativas levantadas para os usuários interessados em modelar LPrSs, é possível identificar as evoluções necessárias para as abordagens investigadas, visando atender a tais necessidades e expectativas. Segundo essa análise, o EPF *Composer* precisa: (i) possibilitar a definição explícita de *features*, suas propriedades e as restrições associadas; (ii) possibilitar a visualização do modelo de *features* correspondente; (iii) possibilitar o rastreamento (e a opção de desfazer) as operações de copiar ou mover elementos de processo; (iv) possibilidade de associar explicitamente pacotes de conteúdo a *features*; e (v) possibilitar uma checagem de consistência, para um determinado conjunto de seleções de *features* de processo.

Já para o GenArch-P, podem ser vislumbradas as seguintes evoluções: (i) possibilitar uma melhor visualização do conhecimento de configuração – destacando os elementos de processo associados à uma dada *feature*, utilizando cores por exemplo; (ii) possibilitar a definição de grupos de elementos de processo e a anotação desses grupos com expressões de features; (iii) possibilitar a navegação entre o modelo de *features* e o modelo de processo; e (iv) possibilitar a definição de *features* mandatórias.

Como trabalhos futuros cogita-se: (i) replicar o experimento com um conjunto de participantes ligados diretamente à indústria de software; (ii) o aperfeiçoamento da abordagem anotativa GenArch-P, evoluindo nas áreas identificadas como pontos fracos para a mesma; (iii) propor e implementar a integração de características importantes da abordagem composicional na abordagem GenArch-P; e (iv) planejar e executar novos estudos empíricos comparando as diferentes abordagens de gerência de variabilidades em LPrSs.

8 Conclusões

Este trabalho apresentou os resultados de um conjunto de estudos realizados sobre o tema de gerência de variabilidades em processos de software. Para o desenvolvimento do trabalho foram definidas quatro questões de pesquisa, as quais nortearam o planejamento e a execução de cada um dos estudos. Os estudos realizados tiveram como objetivo a coleta de evidências, visando embasar as respostas para tais questões de pesquisa.

A primeira questão de pesquisa estava interessada em saber "***quais são os tipos de variabilidade que ocorrem em famílias de processos de software?***". Visando responder a essa questão, foi realizada uma revisão sistemática da literatura (Capítulo 3). Os resultados obtidos com a revisão sistemática foram confrontados com a experiência prática advinda das modelagens de LPrSs, realizadas nos estudos subsequentes. Em tais estudos foram modeladas LPrSs com variabilidades de alto nível, e o critério utilizado para a classificação dessas variabilidades foi a relação das mesmas com as disciplinas de processo de software. Por exemplo, as variabilidades foram classificadas em: (i) técnicas e tecnologia de requisitos; (ii) técnicas e tecnologias de projeto; (iii) técnicas e tecnologias de implementação; e assim por diante. A Seção 3.3 discute uma proposta preliminar para uma classificação das variabilidades em uma LPrS, a ser aprofundada e refinada por meio de trabalhos futuros.

A segunda questão de pesquisa estava interessada em saber "***quais as abordagens existentes para a modelagem e a gerência de variabilidades no contexto de processos de software? Quais os benefícios e as limitações das abordagens existentes?***". Essa questão de pesquisa foi explorada por meio da revisão sistemática da literatura (Capítulo 3). A revisão sistemática identificou 19 (dezenove) abordagens de gerência de variabilidades em LPrSs. Com relação aos benefícios e as limitações dessas abordagens, foram selecionados dois exemplares de abordagens para serem investigados mais profundamente – o EPF *Composer* e o GenArch-P. O EPF *Composer* representando a abordagem que explora técnicas composicionais para a estruturação de suas variabilidades. Por outro lado, o GenArch-P representando a abordagem que adota técnicas anotativas para a definição das mesmas variabilidades. Para complementar a resposta a essa questão foram realizados estudos comparativos (qualitativos e quantitativos) dessas duas abordagens específicas. Também foram identificados indícios

que podem vir a permitir a generalização das respectivas conclusões para as abordagens composicional e anotativa.

A terceira questão de pesquisa estava interessada em saber "***qual a viabilidade do uso de técnicas anotativas na modelagem e definição de uma abordagem para LPrS?***". O Capítulo 4 apresentou uma proposta de abordagem anotativa para a gerência de variabilidades em LPrSs, bem como um estudo exploratório visando atestar a viabilidade da mesma. Os resultados obtidos pelo estudo exploratório atestaram a viabilidade da abordagem proposta. Por conseguinte, foi validada a aplicação das técnicas anotativas na definição de uma abordagem para a modelagem de LPrSs. Nesse estudo, uma ferramenta de derivação de produtos foi adaptada para possibilitar a sua utilização no contexto de processos de software. O suporte ferramental desenvolvido permitiu a anotação de especificações de processos de software com expressões de *features*, e a posterior derivação de instâncias da LPrS modelada. No mesmo estudo, foram identificadas algumas limitações no suporte ferramental desenvolvido para a abordagem, as quais foram atacadas por meio da proposta de uma série de melhoramentos. Evoluções do suporte ferramental da abordagem foram utilizadas na execução dos estudos comparativos subsequentes.

A quarta questão de pesquisa estava interessada em saber "***quais os benefícios, complementaridade e limitações do uso de técnicas anotativas e composicionais no contexto de LPrS?***". Visando responder a essa questão, inicialmente foi proposta uma abordagem anotativa para a gerência de variabilidades em LPrSs (Capítulo 4). Na sequência, as abordagens composicional e anotativa foram confrontadas em estudos comparativos, visando ressaltar os pontos fortes e fracos de cada uma das abordagens. O primeiro estudo realizado, foi um estudo comparativo qualitativo, o qual avaliou as abordagens segundos os critérios de: (i) modularidade, (ii) rastreabilidade, (iii) detecção de erros, (iv) granularidade, (v) uniformidade, (vi) adoção e (vii) gerenciamento sistemático de variabilidades. O segundo estudo realizado foi um estudo comparativo quantitativo, o qual avaliou o resultado da modelagem de uma LPrS alvo com as abordagens investigadas, sob os aspectos de modularidade, tamanho e complexidade. Para o atributo de modularidade, foi utilizada a métrica de número de agrupamentos de elementos de processos associados a uma dada *feature*. Para o atributo de tamanho, foi utilizada a métrica do número total de elementos de processo contidos na especificação da LPrS. Para o atributo de modularidade foram utilizadas as métricas do (i) número de

mecanismos de modelagem de variabilidade e do (ii) quantitativo de aplicações desses mecanismos na modelagem da LPrS alvo. O terceiro, e último, estudo comparativo teve a forma de um experimento controlado, o qual avaliou os atributos de esforço e compreensibilidade das abordagens investigadas. Para o atributo de esforço foram utilizadas métricas de tempo para (i) as tarefas de modelagem e para (ii) as tarefas de alteração de uma LPrS alvo. Para o atributo de compreensibilidade, foi idealizado um questionário para capturar as impressões do usuário no uso das abordagens investigadas.

As abordagens composicional e anotativa foram representadas pelo EPF *Composer* e GenArch-P, respectivamente. Os resultados evidenciaram vantagens e desvantagens das duas abordagens, com melhores resultados no geral para a abordagem anotativa do GenArch-P. Dentre os resultados específicos obtidos pela abordagem GenArch-P, destacam-se: (i) produziu uma especificação de LPrS com um menor número de elementos de processo (atributo de tamanho); (ii) oferece um menor número de mecanismos para a modelagem de variabilidades em uma LPrS, e na modelagem de uma LPrS alvo implicou em um menor número de aplicações destes mecanismos (atributo de complexidade); (iii) proporcionou que uma LPrS alvo fosse modelada e alterada em menos tempo (atributo de esforço); e por fim, (iv) foi mais fácil de entender e aplicar, na opinião dos participantes do experimento controlado que foi realizado (atributo de compreensibilidade).

8.1 Trabalhos Relacionados

Bendraou et al. (Bendraou et al., 2010) apresentam uma comparação entre seis linguagens de modelagem de processos de software baseadas na UML. O estudo, em questão, investigou como as abordagens EPF *Composer* e GenArch-P atendem a importantes requisitos da modelagem de processos, quais sejam: (i) riqueza semântica, (ii) modularidade, (iii) executabilidade, (iv) conformidade ao padrão UML e (v) formalidade. Os autores concluíram as abordagens investigadas apresentaram vantagens e desvantagens; ficando a cargo dos gerentes de projeto, com base nos resultados da avaliação, escolher a abordagem mais adequada às suas necessidades. Por outro lado, no trabalho apresentado neste documento foram comparadas abordagens, de mais alto nível, voltadas à gerência de variabilidades em processos de software. Contudo, é importante notar que os requisitos mencionados por Bendraou et al. também podem ser aplicados às abordagens de LPrS, dado que também são especificados processos.

Simmonds et al. (Simmonds & Bastarrica, 2011a) (Simmonds et al., 2011b) apresentam os resultados de investigações sobre formas de representação de variabilidades em processos de software, quais sejam: (i) SPEM 2.0, (ii) vSPEM, (iii) OVM, e (iv) modelo de *features*. Foi analisada a expressividade de cada notação, bem como a compreensibilidade de cada especificação. Também foi realizada uma avaliação do suporte ferramental para a modelagem de variabilidades em processos de software. Em suas conclusões, os autores reconheceram o poder das linguagens de propósito geral (OVM e modelo *features*) para expressar as variabilidades e suas propriedades. Afirmaram que o SPEM 2 com o EPF *Composer* proporcionam uma interessante interface com o usuário baseada em formulários, mas os conceitos relacionados às variabilidades são complexos. No que diz respeito ao vSPEM, eles concluem que é altamente promissor, porque este permite a modelagem explícita de variabilidades de processos de software. Finalmente, considerando a definição de uma "cadeia" de ferramentas para implementação de uma plataforma para a customização automática de processos de software; a melhor escolha para a modelagem de variabilidades em processos de software foram: (i) o modelo de *features* e (ii) a ferramenta SPLOT. Por outro lado, os estudos comparativos apresentados no presente trabalho focaram em uma análise quantitativa ou qualitativa mais rigorosa de abordagens de gerência de variabilidades em LPrSs, incluindo a abordagem proposta neste trabalho.

Martínez-Ruiz et al. (Martínez-Ruiz et al., 2011a) apresentam os resultados de um estudo empírico avaliando duas notações para a modelagem de processos de software: SPEM e vSPEM. O objetivo principal desse trabalho foi apresentar uma validação empírica, na qual é checado se a forma de representação de variabilidades em processos de software do vSPEM é mais apropriada do que a do SPEM, no que diz respeito à modelagem das mesmas. Foram analisados aspectos como: (i) a compreensibilidade dos mecanismos de modelagem de variabilidades, e (ii) a compreensibilidade da notação como um todo. O trabalho concluiu que a compreensibilidade dos mecanismos de variabilidade do vSPEM foi 126,99% maior que a do SPEM. Por outro lado, a compreensibilidade dos diagramas do vSPEM foi 34,87% menor do que a do SPEM. Por outro lado, no experimento controlado, descrito nesse documento, foi avaliada a compreensibilidade das abordagens do EPF *Composer* e GenArch-P, quando utilizadas na modelagem de LPrSs. A avaliação da complexidade se deu em função da utilização da mesma e não especificamente das notações utilizadas.

8.2 Revisão das Contribuições

Esta Seção apresenta um sumário das contribuições deste trabalho, na seguinte ordem: (i) levantamento das abordagens de gerenciamento de variabilidades em LPrSs existentes (Seção 8.2.1); (ii) apresentação de uma proposta de abordagem anotativa para a gerência de variabilidades em LPrSs, e da ferramenta GenArch-P (Seção 8.2.2); (iv) técnicas de modelagem de LPrSs com o EPF *Composer* (Seção 8.2.3); e (v) estudos e análises comparativas das técnicas composicional e anotativa aplicadas à gerência de variabilidade em LPrSs (Seção 8.2.4).

8.2.1 Panorama das Abordagens de Gerenciamento de Variabilidades em LPrSs

As 19 (dezenove) abordagens de gerência de variabilidades em processos de software identificadas, embora não representem a totalidade das abordagens com esse propósito, oferecem um bom panorama das soluções propostas para esse problema. A compreensão dessas abordagens, suas características e seus *modus operandi*, foram fundamentais para: (i) o entendimento do domínio do problema; e (ii) a concretização da proposta da abordagem anotativa GenArch-P. Dentre as abordagens identificadas, foi dada especial atenção àquelas com suporte ferramental disponível. Contudo, não foi possível analisar e comparar todas as abordagens com suporte ferramental disponível; as comparações ficaram restritas a duas abordagens, representando as técnicas composicional e anotativa.

8.2.2 Abordagem Anotativa para Gerência de Variabilidades em LPrSs e a Ferramenta GenArch-P

A abordagem de gerência de variabilidades em LPrSs idealizada e proposta neste trabalho, avaliou a viabilidade do uso das técnicas anotativas no contexto de processos de software (Aleixo et al., 2010a) (Aleixo et al., 2010b) (Aleixo et al., 2010c). A classificação das abordagens de LPrS em composicionais e anotativas, representando as principais técnicas utilizadas atualmente no projeto e implementação de linhas de produtos de software, auxiliou na observação das características dessas abordagens que auxiliam e dificultam a modelagem de LPrSs. Uma conclusão importante com a análise dos resultados gerais dos estudos realizados é que: a abordagem anotativa no contexto

de LPrSs não é só uma alternativa viável, mas pode facilitar e acelerar o trabalho de um engenheiro de processo na tarefa de modelagem de uma LPrS.

A ferramenta GenArch-P foi desenvolvida para dar suporte à aplicação da abordagem anotativa para a gerência de variabilidades em LPrSs, que foi proposta no trabalho. O GenArch-P é fruto da adaptação da ferramenta GenArch (Cirilo et al., 2008) para operar com processos de software. O GenArch é uma ferramenta de derivação de produtos, baseada nas técnicas anotativas para linha de produtos de software. O GenArch-P sofreu várias evoluções durante o desenvolvimento desse trabalho. As evoluções da ferramenta aconteceram sempre em resposta às deficiências identificadas para a mesma, quando da sua aplicação na realização de um estudo. A versão inicial para a ferramenta, possibilitou: (i) a modelagem de uma LPrS, com a anotação de uma especificação de um processo de software, com expressões de *features*; e (ii) a derivação de uma instância de processo correspondente à uma configuração de *features*. As versões seguintes da ferramenta, utilizadas nos estudos relatados nesse trabalho, foram frutos de evoluções que visaram otimizar a tarefa de "modelagem" da LPrS. A modelagem de uma LPrS envolve as ações de: (i) definir e excluir *features* de processo; (ii) associar e desassociar elementos de processo à *features* específicas; e (iii) realizar manutenções e evoluções em modelagens previamente realizadas.

8.2.3 Estratégias para a Modelagem de LPrS com o EPF Composer

O EPF *Composer* é uma ferramenta de autoria de processos, disponibilizada por uma iniciativa da Fundação Eclipse, denominada de *Eclipse Process Framework*. Essa ferramenta se destina à especificação de processos de software. Com o EPF *Composer* é possível editar e alterar uma especificação de processo, usando como base outras especificações. A ferramenta disponibiliza o mecanismo de variabilidade de conteúdo, o qual permite que conteúdos de um elemento base sejam estendidos e reutilizados por elementos especializados. Durante a realização deste trabalho foi desenvolvido e validado um conjunto de estratégias que guiam os engenheiros de processos na modelagem das variabilidades, opcionais e alternativas, de LPrS com o EPF *Composer* (Aleixo et al., 2012a) (Aleixo et al., 2012b). O conjunto de estratégias de modelagem das variabilidades foi apresentado nos relatos de realização dos estudos comparativos, nos quais o EPF *Composer* representou a abordagem composicional.

8.2.4 Estudos e Análises Comparativas das Técnicas Composicional e Anotativa na Gerência de Variabilidades em LPrSs

Foram realizados três tipos de estudos comparativos de duas abordagens específicas para a gerência de variabilidade em LPrSs (Aleixo et al., 2012a) (Aleixo et al., 2012b) (Aleixo et al., 2013). As abordagens foram o EPF *Composer* e o GenArch-P, os quais representam, respectivamente, a abordagem composicional e a abordagem anotativa. Os seguintes estudos foram realizados: um estudo comparativo qualitativo, um estudo comparativo quantitativo e um experimento controlado. O planejamento, os critérios e as métricas utilizados pelos mesmos podem servir de base para a realização de novos estudos, bem como a replicação desses estudos com outras abordagens de LPrS. Da mesma forma, os resultados obtidos por tais estudos podem servir de base para o refinamento e a evolução dos critérios e métricas utilizados na comparação das abordagens. Por outro lado, as evidências geradas por tais estudos também podem ser utilizadas na elaboração de novas hipóteses; as quais irão motivar e direcionar a realização de novos estudos.

8.3 Limitações do Trabalho

Esta seção apresenta as principais limitações identificadas para este trabalho. Uma limitação identificada para este trabalho está no fato de que os estudos comparativos realizados consideraram apenas dois exemplares de abordagens. Para que as conclusões do trabalho possam ser estendidas e generalizadas, é necessário considerar outros exemplos de abordagens que representem as técnicas composicionais e anotativas. Esse fato se deveu, principalmente, ao estágio de desenvolvimento das abordagens que foram identificadas, bem como a carência de ferramentas automatizadas oferecidas pelas mesmas. A maioria das abordagens encontra-se ainda em fase de amadurecimento e desenvolvimento de suporte ferramental. Além disso, é importante que também nos novos estudos sejam especificadas outras LPrSs.

Outra limitação identificada para este trabalho está no fato de que os estudos comparativos realizados ficaram restritos ao ambiente acadêmico. Foi planejada a replicação de um dos estudos em uma instituição voltada especificamente ao desenvolvimento de sistemas. Entretanto, mudanças da direção da tal instituição, bem como outros fatores externos, impediram a realização do mesmo. É muito importante que os resultados obtidos nos estudos comparativos sejam confrontados com os

resultados de um possível estudo de caso conduzido junto à indústria de desenvolvimento de software, ou mesmo sejam conduzidos novos estudos empíricos considerando especificações de LPrSs provenientes de empresas.

Outra limitação identificada para este trabalho está no fato de que não foram tratadas todas as deficiências identificadas para a ferramenta GenArch-P, como, por exemplo, a carência por uma visualização mais clara do conhecimento de configuração, a qual foi identificada durante a realização do experimento controlado. A análise dos resultados do experimento, bem como o *feedback* dos participantes do mesmo, possibilitaram a conclusão de que esta deficiência específica gerou uma perda de tempo, a qual poderia alterar os resultados obtidos, caso fosse considerada. Uma possível solução para esta deficiência específica passa por melhorar a representação do conhecimento de configuração, por meio de algum mecanismo que possibilite uma separação visual de interesses. Por exemplo, colorindo com uma cor especifica os elementos de processo “anotados” com uma dada *feature*, ou oferecendo suporte ferramental que facilitasse o processo de gerência de variabilidades.

Outra limitação identificada diz respeito aos possíveis complementos aos resultados obtidos com a revisão sistemática da literatura (apresentada no Capítulo 3). A revisão sistemática atingiu o seu propósito oferecendo um panorama das abordagens existentes para a gerência de variabilidades em processos de software, do ponto de vista do suporte ferramental existente, dos tipos de variabilidades de processos explorados pelos trabalhos existentes, dos estudos comparativos existentes na área.

Durante o desenvolvimento do trabalho, foi observada a necessidade de extensão e melhoria da revisão sistemática especificamente nos seguintes aspectos: (i) levantamento de características que possibilitem uma comparação inicial das abordagens existentes; (ii) cenários em que tais abordagens foram aplicadas; (iii) formas de aplicação dessas abordagens; (iv) em que fase do ciclo de vida de LPrS é utilizado o suporte ferramental das mesmas; e (v) como as abordagens foram demonstradas em suas publicações. Estes aspectos deverão motivar a realização de complementos desta revisão sistemática, ou mesmo a realização de novas revisões sistemáticas.

8.4 Trabalhos Futuros

Visando suprir as limitações identificadas nesse trabalho e como forma de dar continuidade ao trabalho aqui iniciado, foram previstos os seguintes trabalhos futuros:

(i) ***Evolução da Abordagem Anotativa***: implementar e validar evoluções na abordagem anotativa, com base nas limitações e possíveis melhoramentos identificados durante a execução deste trabalho, tais como: especificação multi-nível de *features* (definir um nível superior à seleção individual das *features*, como por exemplo definir perguntas de alto-nível que impliquem na seleção automática de uma série de *features*); propor mecanismos para a visualização das variabilidades da LPrS (visualizações das *features* e seus elementos associados, do impacto da remoção de um elemento de processo, dos elementos de processo ligados a mais de uma *feature*, entre outras); e checagem de consistência e detecção de erros no processo de derivação de instâncias de processos de software;

(ii) ***Integração da Abordagem Anotativa com a Composicional***: conceber, implementar e validar uma proposta de abordagem que integre as características positivas das abordagens composicional e anotativa, identificadas neste trabalho, objetivando a gerência de variabilidades em LPrSs;

(iii) ***Definição de um Suíte de Critérios e Métricas para Avaliação de LPrSs***: refinar e evoluir os critérios, métricas e parâmetros utilizados nos estudos comparativos apresentados no decorrer do documento, visando a sua aplicação em novos estudos comparativos;

(iv) ***Realização de Novos Estudos Comparativos***: realizar novos estudos comparativos das abordagens de gerência de variabilidades em processos de software, preferencialmente no contexto da indústria de software, e utilizando outros exemplares das abordagens composicional e anotativa. Para os experimentos controlados que venham a ser realizados, utilizar um ambiente de suporte para a definição e a replicação de tais experimentos (Freire et al., 2013); e

(v) ***Realização de Novas Revisões Sistemáticas***: buscando responder questões mais específicas, objetivando: (a) comparar as abordagens existentes de LPrS, identificando cenários em que uma abordagem é “melhor” que a

outra; (b) consolidar a proposta de classificação das variabilidades em processos de software; (c) em que são utilizados os suportes ferramentais das abordagens existentes; entre outros.

(vi) ***Aplicação dos Resultados no Contexto de Processos de Negócio***: dada a similaridade de linguagens de modelagem de especificação de processos de negócio com processos de software, é possível vislumbrar a exploração dos resultados obtidos com a abordagem anotativa neste trabalho, para a área de processos de negócio.

9 Referências Bibliográficas

Alegria, J.A.H. & Bastarrica, M.C., 2009. Process Model Tailoring as a Means for Process Knowledge Reuse. In *2nd Workshop on Knowledge Reuse (KREUSE)*. Falls Church, Virginia, USA, 2009. Springer-Verlag.

Alegria, J.A.H. & Bastarrica, M.C., 2010. *Tutelkán Implementation Process: Adapting a Reusable Reference Software Process in the Chilean Software Industry*. Technical Report. Santiago, Chile: Universidad de Chile.

Alegria, J.A.H. & Bastarrica, M.C., 2012. Building software process lines with CASPER. In *International Conference on Software and System Process*. Zurich, Suíça, 2012. IEEE.

Alegria, J.A.H., Bastarrica, M.C., Quispe, A. & Ochoa, S.F., 2011. An MDE Approach to Software Process Tailoring. In *International Conference on Software and Systems Process (ICSSP 2011)*. Honolulu, HI, USA, 2011. ACM.

Aleixo, F.A., 2013. *Software Process Lines*. [Online] Available at: https://sites.google.com/site/softwareprocesslines/ [Accessed 2013].

Aleixo, F.A. et al., 2012b. A Comparative Study of Compositional and Annotative Modelling Approaches for Software Process Lines A Comparative Study of Compositional and Annotative Modelling Approaches for Software Process Lines. In *26º Simpósio Brasileiro de Engenharia de Software*. Natal, RN, Brazil, 2012b. IEEE Digital Library.

Aleixo, F.A., Freire, M.A., Santos, W.C.d. & Kulesza, U., 2010a. Automating the Variability Management, Customization and Deployment of Software Processes: A Model-Driven Approach. *Lecture Notes in Business Information Processing*, pp.372-87.

Aleixo, F.A., Freire, M.A., Santos, W.C.d. & Kulesza, U., 2010b. A Model-driven Approach to Managing and Customizing Software Process Variabilities. In *12th International Conference on Enterprise Information Systems*. Funchal, Madeira, Portugal, 2010b. SciTePress.

Aleixo, F.A., Freire, M.A., Santos, W.C. & Kulesza, U., 2010c. An Approach to Manage and Customize Variability in Software Processes. In *Brazilian Symposium on Software Engineering*. Salvador - BA, Brazil, 2010c.

Aleixo, F.A. et al., 2012a. Modularizing Software Process Lines using Model-driven Approaches - A Comparative Study. In *ICEIS 2012 - Proceedings of the 14th International Conference on Enterprise Information Systems*. Wroclaw, Poland, 2012a. SciTePress.

Aleixo, F.A., Kulesza, U. & Oliveira Junior, E.A.d., 2013. Modeling Variabilities from Software Process Lines with Compositional and Annotative Techniques: A Quantitative Study. In Heidrich, J., Oivo, M., Jedlitschka, A. & Baldassarre, M.T., eds. *International Conference in Product-Focused Software Process Improvement*. Paphos, Chipre, 2013. Springer.

Ambler, S.W., 2011. *Agile Design*. [Online] Available at: http://www.agilemodeling.com/essays/agileDesign.htm [Accessed 27 January 2012].

Armbrust, O., 2010. Determining Organization-Specific Process Suitability. In *International Conference on Software Process (ICSP 2010)*. Paderborn, Germany, 2010. Springer.

Armbrust, O. et al., 2008b. Experiences and Results from Tailoring and Deploying a Large Process Standard in a Company. *Software Process: Improvement and Practice*, 13(4), pp.301-09.

Armbrust, O. et al., 2008a. coping Software Process Models - Initial Concepts and Experience from Defining Space Standards. In *International Conference on Software Process (ICSP 2008)*., 2008a. Springer.

Armbrust, O. et al., 2009. Scoping software process lines. *Software Process: Improvement and Practice*, 14-3, pp.181-97.

Armbrust, O. & Rombach, H.D., 2011. The right process for each context: objective evidence needed. In *International Conference on Software and Systems Process*. Honolulu, HI, USA, 2011. ACM.

Barreto, A., Duarte, E., Rocha, A.R. & Murta, L., 2010. Supporting the Definition of Software Processes at Consulting Organizations via Software Process Lines. In *7th International Conference on the Quality of Information and Communications Technology*. Porto, Portugal, 2010. IEEE Computer Society.

Barreto, A., Murta, L.G.P. & Rocha, A.R.C.d., 2011. Software Process Definition: a Reuse-based Approach. *The Journal of Universal Computer Science*, 17 (13), pp.1765-99.

Basili, V.R., Caldiera, G. & Rombach, H.D., 1994. The Goal Question Metric Approach. In *Encyclopedia of Software Engineering*. Wiley.

Bendraou, R., Combemale, B., Crégut, X. & Gervais, M.-P., 2007. Definition of an Executable SPEM 2.0. In *APSEC 2007*. Nagoya, Japan, 2007. IEEE Computer Society.

Bendraou, R., Gervais, M.P. & Blanc, X., 2005. UML4SPM: A UML 2.0-based Meta-model for Software Process Modelling. In *MoDELS'05*., 2005. Springer-Verlag.

Bendraou, R., Jézéquel, J.-M., Gervais, M.-P. & Blanc, X., 2010. A Comparison of Six UML-Based Languages for Software Process Modeling. *Transactions on Software Engineering*, 36(5), pp.662-75.

BigLever Software, 2011. *BigLever Software Gears*. [Online] Available at: http://www.biglever.com/solution/product.html [Accessed 27 January 2012].

Bonifácio, R. & Borba, P., 2009. Modeling Scenario Variability as Crosscutting Mechanisms. In ACM, ed. *International Conference on Aspect-Oriented Software Development (AOSD)*. Charlottesville, Virginia, USA, 2009.

Braude, E., 2005. *Projeto de Software*. Bookman.

Cervera, M., Albert, M., Torres, V. & Pelechano, V., 2010. A Methodological Framework and Software Infrastructure for the Construction of Software Production Methods. In *International Conference on Software Process*. Paderborn, Alemanha, 2010. Springer.

Chemuturi, M. & Cagley Jr., T.M., 2010. *Mastering Software Project Management: Best Practices, Tools and Techniques*. J. Ross Publishing.

Cirilo, E., Kulesza, U., Garcia, A. & Lucena, C.J.P.d., 2011. GenArch+: an Extensible Infrastructure for Building Framework-based Software Product Lines. In *10th International Conference on Aspect-Oriented Software Development (AOSD 2011)*. Porto de Galinhas, Brazil, 2011. ACM.

Cirilo, E., Kulesza, U. & Lucena, C.J.P.d., 2008. A Product Derivation Tool Based on Model-Driven Techniques and Annotations. *The Journal of Universal Computer Science*, 14-8, pp.1344-67.

Clements, P. & Northrop, L., 2001. *Software Product Lines: Practices and Patterns*. Addison-Wesley.

Clifton, M. & Dunlap, J., 2003. *What is SCRUM?* [Online] Available at: http://www.codeproject.com/Articles/4798/What-is-SCRUM [Accessed September 2012].

CNPq, 2012. *Grupo de Pesquisa - Núcleo de Desenvolvimento de Software (NUDES)*. [Online] Available at: http://dgp.cnpq.br/buscaoperacional/detalhegrupo.jsp?grupo=2815103NKG0WWE [Accessed September 2012].

Cockburn, A., 2001. *Writing Effective Use Cases*. Addison-Wesley.

Czarnecki, K. & Eisenecker, U.W., 2000. *Generative Programming - Methods, Tools and Applications*. Addison-Wesley.

Eclipse Foudation, 2010. *Eclipse Process Framework (EPF) Composer - Installation, Introduction, Tutorial and Manual*. [Online] Available at: http://www.eclipse.org/epf/general/EPF_Installation_Tutorial_User_Manual.pdf [Accessed 27 January 2012].

Eclipse Foundation, 2012. *Eclipse Process Framework Project (EPF)*. [Online] Available at: http://www.eclipse.org/epf/ [Accessed 27 January 2012].

Eclipse Foundation, 2012. *EPF Practices Libraries Download*. [Online] Available at: http://www.eclipse.org/epf/downloads/praclib/praclib_downloads.php [Accessed 27 January 2012].

Eclipse Foundation, 2012. *Get Involved with M2T Xpand*. [Online] Available at: http://wiki.eclipse.org/Xpand [Accessed September 2012].

EPF, 2011. *OpenUP Published Web Site*. [Online] Available at: http://epf.eclipse.org/wikis/openup/ [Accessed 27 January 2012].

Field, A., Miles, J. & Field, Z., 2012. *Discovering Statistics Using R*. Editora SAGE.

Figueiredo, E. & al, e., 2008. Evolving Software Product Lines with Aspects: An Empirical Study on Design Stability. In ACM, ed. *International Conference on Software Engineering (ICSE)*. Leipzig, Germany, 2008.

Freire, M.A. et al., 2013. *A Model-Driven Approach to Specifying and Monitoring Controlled Experiments in Software Engineering*. Paphos, Chipre: Springer.

Freire, M.A. et al., 2011. Automatic Deployment and Monitoring of Software Processes: A Model-Driven Approach. In *23rd International Conference on Software Engineering & Knowledge Engineering (SEKE'2011)*. Miami Beach, USA, 2011. Knowledge Systems Institute Graduate School.

Garcia, A. et al., 2005. Modularizing design patterns with aspects: a quantitative study. In Press, A., ed. *Aspect-Oriented Software Development Conference (AOSD)*. Chicago, USA, 2005.

Generative Software Development Lab, 2012. *fmp: Feature Modeling Plug-in*. [Online] Available at: http://gp.uwaterloo.ca/fmp [Accessed 27 January 2012].

Hair, J.F., Tatham, R.L., Anderson, R.E. & Black, W.C., 2007. *Analise Multivariada de Dados*. Bookman.

Haumer, P., 2007. *Eclipse Process Framework Composer - Part 1: Key Concepts*. [Online] Available at: http://www.eclipse.org/epf/general/EPFComposerOverviewPart1.pdf [Accessed 27 January 2012].

Hesse, W. & Noack, J., 1999. A Multi-variant Approach to Software Process Modelling. In *11th International Conference on Advanced Information Systems Engineering*. London, UK, 1999. Springer-Verlag.

IBM, 2012. *IBM - RUP - Rational Method Composer - Software*. [Online] Available at: http://www.ibm.com/software/awdtools/rmc/ [Accessed 27 January 2012].

ISO, 2004. *ISO/IEC 15504: Information Technology - Process Assessment*. ISO/IEC.

Jaufman, O. & Münch, J., 2005. Acquisition of a Project-specific Process. In *6th International Conference on Product Focused Software Process Improvement*. Oulu, Finland, 2005. Springer.

Kang, K.C. et al., 1990. *Feature-oriented domain analysis (FODA) feasibility study*. SEI.

Kerzazi, N. & Robillard, P., 2010. Multi-Perspective Software Process Modeling. In *8th ACIS International Conference on Software Engineering Research, Management and Applications (SERA 2010)*. Montreal, Canada, 2010. IEEE Computer Society.

Kästner, C., 2010. *Virtual Separation of Concerns: Toward Preprocessors 2.0*. Magdeburg, Germany: Dissertation, Otto-von-Guericke-Universität.

Kästner, C. & Apel, S., 2008b. Integrating Compositional and Annotative Approaches for Product Line Engineering. In *GPCE Workshop on Modularization, Composition and Generative Techniques for Product Line Engineering (McGPLE)*. Passau, Germany, 2008b. University of Passau.

Kästner, C., Apel, S. & Kuhlemann, M., 2008a. Granularity in software product lines. In Wilhelm Schäfer, M.B.D.V.G., ed. *International Conference on Software Engineering*. Leipzig, Alemanha, 2008a. ACM.

Kästner, C., Apel, S., Rosenthal, M. & Dreiling, A., 2010. *CIDE: Virtual Separation of Concerns (Preprocessor 2.0)*. [Online] Available at: http://wwwiti.cs.uni-magdeburg.de/iti_db/research/cide/ [Accessed 27 January 2012].

Kiebusch, S., Franczyk, B. & Speck, A., 2006. Process-Family-Points. In *International Software Process Workshop and International Workshop on Software Process Simulation and Modeling*. Shanghai, China, 2006. Springer.

Killisperger, P. et al., 2009. Meta Model Based Architecture for Software Process Instantiation. In *International Conference on Software Process*. Vancouver, Canadá, 2009. Springer.

Killisperger, P. et al., 2010. A Framework for the Flexible Instantiation of Large Scale Software Process Tailoring. In *International Conference on Software Process*. Paderborn, Alemanha, 2010. Springer.

Kitchenham, B., 2004. *Procedures for Performing Systematic Reviews*. Technical Report. Keele University.

Kitchenham, B. et al., 2010. Systematic Literature Reviews in Software Engineering - A Tertiary Study. *Information & Software Technology*, pp.792-805.

Koudri, A. & Champeau, J., 2010. MODAL: A SPEM Extension to Improve Co-design Process Models. *Lecture Notes in Computer Science*, pp.248-59.

Kruchten, P., 2004. *The Rational Unified Process: An Introduction*. Addison-Wesley Professional.

Leroy, G., 2011. *Designing User Studies in Informatics*. Springer.

Lewandowski, D., 2010. Google Scholar as a Tool for Discovering Journal Articles in Library and Information Science. *Online Information Review*, 34 (2), pp.250 - 262.

Li, J. & Mao, M., 2009. A Case Study on Tailoring Software Process for Characteristics Based on RUP. In *International Conference on Computational Intelligence and Software Engineering*. Wuhan, China, 2009. IEEE.

Linden, F.J.v.d., Schmid, K. & Rommes, E., 2007. *Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering*. Springer.

Lobsitz, R.M., 1996. A Method for Assembling a Project-Specific Software Process Definition. In *29th Annual Hawaii International Conference on System Sciences*. Maui, Hawaii, 1996. IEEE.

Madachy, R.J., 2006. Reusable Model Structures and Behaviors for Software Processes. In *International Software Process Workshop and International Workshop on Software Process Simulation and Modeling*. Shanghai, China, 2006. Springer.

Magdaleno, A.M., 2010a. Balancing Collaboration and Discipline in Software Development Processes. In *32nd ACM/IEEE International Conference on Software Engineering*. Cape Town, África do Sul, 2010a. ACM.

Magdaleno, A.M., 2010b. An Optimization-based Approach to Software Development Process Tailoring. In *2nd International Symposium on Search Based Software Engineering*. Benevento, Itália, 2010b. IEEE.

Malik, 2008. *Software Quality : A Practitioner's Approach*. Tata McGraw-Hill Education.

Martínez-Ruiz, T., García, F. & Piattini, M., 2008. Towards a SPEM v2.0 Extension to Define Process Lines Variability Mechanisms. In R. Lee, ed. *Software Engineering Research, Management and Applications*. Springer Berlin Heidelberg. pp.115-30.

Martínez-Ruiz, T., García, F. & Piattini, M., 2009a. Process Institutionalization using Software Process Lines. In *ICEIS 2009*. Milan, 2009a.

Martínez-Ruiz, T., García, F. & Piattini, M., 2009b. An Enhanced Variability Mechanisms to Manage Software Process Lines. In *16 EuroSPI Conference Industrial Proceedings*. Alcala de Henares, Espanha, 2009b. Springer.

Martínez-Ruiz, T., García, F., Piattini, M. & Münch, J., 2011a. Modelling Software Process Variability: an Empirical Study. *IET Software*, 5 (2), pp.172-87.

Martínez-Ruiz, T., García, F., Piattini, M. & Münch, J., 2011b. Applying AOSE Concepts to Model Crosscutting Variability in Variant-Rich Processes. In *37th EUROMICRO Conference on Software Engineering and Advanced Applications*. Oulu, Finlândia, 2011b. IEEE.

Martínez-Ruiz, T., Münch, J., García, F. & Piattini, M., 2012. Requirements and Constructors for Tailoring Software Processes: A Systematic Literature Review. *Software Quality Journal*, 20 (1), pp.229-60.

Mountain Goat Software, 2012. *Srcum - Beginners Guide to Scrum*. [Online] Available at: http://www.mountaingoatsoftware.com/topics/scrum [Accessed September 2012].

Obeo, 2012. *Acceleo: MDA Generator*. [Online] Available at: http://www.acceleo.org/pages/home/en [Accessed September 2012].

Ocampo, A., Münch, J. & Bella, F., 2005. Software Process Commonality Analysis. *Software Process: Improvement and Practice*, 10, pp.273-85.

OMG, 2008. *Software & Systems Process Engineering Metamodel (SPEM)*. [Online] Available at: http://www.omg.org/spec/SPEM/ [Accessed September 2012].

OMG, 2012. *Object Management Group*. [Online] Available at: http://www.omg.org/ [Accessed September 2012].

openArchitectureWare.org, 2012. *openArchitectureWare*. [Online] Available at: http://www.openarchitectureware.org/ [Accessed September 2012].

Pillat, R.M., Oliveira, T.C. & Fonseca, F.L., 2012. Introducing Software Process Tailoring to BPMN: BPMNt. In *International Conference on Software and System Process*. Zurich, Suíça, 2012. IEEE.

Pohl, K., Böckle, G. & Linden, F.v.d., 2005. *Software product line engineering: foundations, principles, and techniques*. Berlin, Germany: Springer-Verlang.

Pressman, R.S., 2006. *Engenharia de Software*. 6th ed. Mcgraw Hill - Artmed.

pure-systems, 2012. *pure:variants - Variant Management*. [Online] Available at: http://www.pure-systems.com/pure_variants.49.0.html [Accessed 27 January 2012].

Randolph, J.J., 2009. A Guide to Writing the Dissertation Literature Review. *Practical Assessment, Research & Evaluation Journal*, 14.

Rombach, H.D., 2005. Integrated Software Process and Product Lines. In *Unifying the Software Process Spectrum, International Software Process Workshop*. Beijing, China, 2005. Springer.

Ryan, T., 2011. *Estatística Moderna Para Engenharia*. Elsevier Brasil.

Salatino, M., 2009. *jBPM Developer Guide*. Packt Publishing Ltd.

Scott, K., 2003. *Processo Unificado Explicado*. Bookman.

Scrum Alliance, Inc., 2012. *Scrum Alliance - Transforming the World of Work*. [Online] Available at: http://scrumalliance.org [Accessed September 2012].

Scrum.org, 2012. *Scrum.org - Improving the Profession of Software Development*. [Online] Available at: http://www.scrum.org [Accessed September 2012].

Silva, F.Q.B.d. et al., 2011. Six Years of Systematic Literature Reviews in Software Engineering: An Updated Tertiary Study. *Information & Software Technology*, pp.899-913.

Simidchieva, B.I., Clarke, L.A. & Osterweil, L.J., 2007. Representing Process Variation with a Process Family. In *International Conference on Software Process*. Minneapolis, MN, USA, 2007. Springer.

Simidchieva, B.I. & Osterweil, L.J., 2011. Characterizing process variation. In *33rd International Conference on Software Engineering*. Waikiki, Honolulu, HI, USA, 2011. ACM.

Simmonds, J. & Bastarrica, M.C., 2011a. *Modeling Variability in Software Process Lines*. Santiago, Chile: Universidad de Chile.

Simmonds, J., Bastarrica, M.C., Silvestre, L. & Quispe, A., 2011b. *Analyzing Methodologies and Tools for Specifying Variability in Software Processes*. Santiago, Chile: Universidad de Chile.

Simmonds, J., Bastarrica, M.C., Silvestre, L. & Quispe, A., 2012. *Modeling Variability in Software Process Models*. Technical Report. Santiago, Chile: Universidad de Chile Universidad de Chile.

Steinberg, D., Budinsky, F., Merks, E. & Paternostro, M., 2008. *EMF: Eclipse Modeling Framework*. 2nd ed. Pearson Education.

Sutton, S.M.J. & Osterweil, L.J., 1996. Product Families and Process Families. In *10th International Software Process Workshop*. Dijon, France, 1996. IEEE.

Teetor, P., 2011. *R Cookbook*. O'Reilly Media, Inc.

Ternité, T., 2009. Process Lines: A Product Line Approach Designed for Process Model Development. In *35th Euromicro Conference on Software Engineering and Advanced Applications*. Patras, Greece, 2009. IEEE Computer Society.

Thaker, S., Batory, D., Kitchin, D. & Cook, W., 2007. Safe Composition of Product Lines. In *6th International Conference on Generative Programming and Component Engineering (GPCE '07)*., 2007. ACM.

Walters, W.H., 2009. Google Scholar Search Performance: Comparative Recall and Precision. *portal: Libraries and the Academy*, 9 (1), pp.5-24.

Washizaki, H., 2006a. Building Software Process Line Architectures from Bottom Up. In *Product-Focused Software Process Improvement, 7th International Conference*. Amsterdam, The Netherlands, 2006a. Springer.

Washizaki, H., 2006b. Deriving Project-Specific Processes from Process Line Architecture with Commonality and Variability. In *2006 IEEE International Conference on Industrial Informatics*., 2006b. IEEE.

Wohlin, C., 2012. *Experimentation in Software Engineering*. Springer.