UNIVERSIDADE FEDERAL DA PARAÍBA

CENTRO DE INFORMÁTICA

PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

PROMOVENDO MODULARIDADE EM UM PROCESSO DE ENGENHARIA DE REQUISITOS PARA LINHAS DE PRODUTO DE SOFTWARE

DORGIVAL PEREIRA DA SILVA NETTO

JOÃO PESSOA, PARAÍBA, BRASIL

JULHO/2015

DORGIVAL PEREIRA DA SILVA NETTO

PROMOVENDO MODULARIDADE EM UM PROCESSO DE ENGENHARIA DE REQUISITOS PARA LINHAS DE PRODUTO DE SOFTWARE

Dissertação apresentada ao Centro de Informática da Universidade Federal da Paraíba, como requisito parcial para obtenção do título de Mestre em Informática.

Área de Concentração: Ciência da Computação

Linha de Pesquisa: Computação Distribuída

Prof. DSc. Carla Taciana Lima Lourenço Silva Schuenemann

(Orientadora)

João Pessoa, Paraíba, Brasil

Julho / 2015

S586p Silva Netto, Dorgival Pereira da.
Promovendo modularidade em um processo de Engenharia de Requisitos para linhas de produto de software / Dorgival Pereira da Silva Neto.- João Pessoa, 2015.
186f. : il.
Orientadora: Carla Taciana Lima Lourenço Silva Schuenemann
Dissertação (Mestrado) - UFPB/CI
1. Informática. 2. Computação distribuída. 3. Linhas de produto de software. 4. Modelos de objetivos. 5. Cenários aspectuais. 6. Engenharia de requisitos.

UFPB/BC CDU: 004(043)

Ata da Sessão Pública de Defesa de Dissertação de Mestrado de **DORGIVAL PEREIRA DA SILVA NETTO**, candidato ao título de Mestre em Informática na Área de Sistemas de Computação, realizada em 13 de julho de 2015.

Ao décimo terceiro dia do mês de julho do ano de dois mil e quinze, às treze horas, no Centro de Informática - Universidade Federal da Paraíba (unidade Mangabeira), reuniram-se os membros da Banca Examinadora constituída para julgar o Trabalho Final do **Sr. Dorgival Pereira da Silva,** candidato ao grau de Mestre em Informática, na área de "*Sistemas de Computação*", na linha de pesquisa "*Computação Distribuída*", do Programa de Pós-Graduação em Informática, da Universidade Federal da Paraíba. A comissão examinadora foi composta pelos professores doutores: **Carla Taciana Lima Lourenço Silva Schuenemann (PPGI-UFPB)**, Orientadora e Presidente da Banca, **Alexandre Nóbrega Duarte (PPGI-UFPB)**, examinador interno, **Ayla Débora Dantas de Souza Rebouças (UFPB)**, Examinadora Externa ao Programa, **Danielle Rousy Dias da Silva (UFPB)**, Examinadora Externa ao Programa e **Márcia Jacyntha Nunes Rodrigues Lucena (UFRN),** Examinadora Externa à Instituição. Dando início aos trabalhos, a professora Carla Taciana Lima Lourenço Silva Schuenemann cumprimentou os presentes, comunicou aos mesmos a finalidade da reunião e passou a palavra ao candidato para que o mesmo fizesse, oralmente, a exposição do trabalho de dissertação intitulado "*Promovendo Modularidade em um Processo de Engenharia Requisitos para Linhas de Produto de Software*". Concluída a exposição, o candidato foi arguido pela Banca Examinadora que emitiu o seguinte parecer: "*aprovado*". Assim sendo, eu, Nadja Rayssa Soares de Almeida, Auxiliar em Administração, Secretária do PPGI – Programa de Pós Graduação em Informática, lavrei a presente ata que segue assinada por mim e pelos membros da Banca Examinadora. João Pessoa, 13 de julho de 2015.

Nadja Rayssa Soares de Almeida

Profª Dra. Carla Taciana Lima Lourenço Silva Schuenemann
Orientadora e Presidente da Banca (PPGI-UFPB)

Profº Dr. Alexandre Nóbrega Duarte
Examinador Interno(PPGI-UFPB)

Profª Dra Ayla Débora Dantas de Souza Rebouças
Examinadora Externa ao Programa (UFPB)

Profª Dra. Danielle Rousy Dias da Silva
Examinadora Externa ao Programa (UFPB)

Profª Dra. Marcia Jacyntha Nunes Rodrigues Lucena
Examinadora Externa à Instituição (UFRN)

Resumo

Abordagens de Engenharia de Requisitos Orientadas a Objetivos capturam tanto os objetivos dos interessados (*stakeholders*) como os requisitos do software a ser desenvolvido, de modo que este último corresponda ao que realmente os interessados desejam. Modelos de objetivos são capazes de capturar as similaridades e variabilidades de uma Linha de Produto de Software (LPS), mas não conseguem descrever o comportamento detalhado de suas funcionalidades. Diante dessa limitação, o processo GS2SPL (*Goals and Scenarios to Software Product Lines*) foi definido para obter sistematicamente, a partir de modelos de objetivos, modelos de *features* e especificações de cenários de casos de uso com variabilidade, descritos em PLUSS (*Product Line Use case modeling for Systems and Software engineering*). Entretanto, a variabilidade da LPS e o conhecimento de configuração ficam entrelaçados nos cenários descritos em PLUSS, o que prejudica a manutenção e reuso dos artefatos. A fim de solucionar esse problema, foram propostas técnicas de especificação de cenários de caso de uso com separação de interesses transversais (ou, simplesmente, cenários aspectuais). Uma destas técnicas é o MSVCM (*Modeling Scenario Variability as Crosscutting Mechanisms*), que especifica a variabilidade da LPS separadamente do conhecimento de configuração e define um processo para configurar as especificações de produto. Assim, este trabalho propõe uma extensão do GS2SPL visando obter, sistematicamente, modelos de *features* e especificações de cenários aspectuais em MSVCM, a partir de modelos de objetivos. Esta abordagem chama-se GAS2SPL (*Goals and Aspectual Scenarios to Software Product Lines*) e suas atividades foram descritas utilizando o TaRGeT (*Test and Requirements Generation Tool*) como exemplo. A abordagem GAS2SPL foi avaliada através de um estudo comparativo entre os artefatos do TaRGeT e do *MyCourses- A Course Scheduling System* gerados pelas abordagens GS2SPL e GAS2SPL, levando-se em consideração a modularidade (espalhamento de *features* e entrelaçamento de cenários) e, a expressividade (quão detalhado é o conhecimento de configuração). Depois de realizar a avaliação, percebemos que a abordagem GAS2SPL conseguiu reduzir o espalhamento de *features* e o entrelaçamento de cenários para zero, além de possuir um conhecimento de configuração mais expressivo, pois utiliza menos símbolos para elaborá-lo.

Palavras-chave: Linhas de Produto de Software, Modelos de Objetivos, Cenários Aspectuais, Engenharia de Requisitos.

Abstract

Goal Oriented Requirements Engineering approaches capture both the stakeholders' goals and the requirements of the system-to-be, so that the latter corresponds to the stakeholders desires. Goal models can capture similarities and the variability of a Software Product Line (SPL), but they cannot describe the detailed behavior of its functionality. Due to this limitation, a process called GS2SPL (Goals and Scenarios to Software Product Lines) was defined to systematically obtain, from goal models, feature models and the specification of use case scenarios with variability described in PLUSS (Product Line Use case modeling for Systems and Software engineering). However, the variability of the SPL and the configuration knowledge are tangled an the scenarios described in PLUSS, jeopardizing the maintenance and reuse of artifacts. In order to solve this problem, it was proposed techniques to specific use case scenarios with separation of crosscutting concerns (or just, aspectual scenarios). One of these techniques is called MSVCM (Modeling Scenario Variability as Crosscutting Mechanisms), which specifies the variability and configuration knowledge of a SPL separately, as well as it defines a process to configure the specifications of a product. Thus, this work proposes an extension of the GS2SPL to obtain, systematically, a feature model and a specification of aspectual scenarios in MSVCM, from goal models. This approach is called GAS2SPL (Goals and Aspectual Scenarios to Software Product Lines) and their activities were described using the TaRGeT (*Test and Requirements Generation Tool*) example. GAS2SPL approach was evaluated through a comparative study between TaRGeT and *MyCourses* artifacts generated by GS2SPL and GAS2SPL approaches, taking into account modularity (features scattering and tangling scenarios) and expressiveness (how detailed are the configuration knowledge). After evaluating our approach, we realize that GAS2SPL approach reduced in the features scattering and tangling in the scenarios to zero, addition to own a knowledge configuration more specific because uses less symbols for it elaborate.

Keywords: Software Product Lines, Goal Models, Aspectual Scenarios, Requirements Engineering.

Agradecimentos

A Deus, por me dar forças para não desistir nos momentos mais difíceis e que me permitiu alcançar este objetivo.

Aos meus pais, Paulo e Paula, e a minha namorada, Mayara, por estarem comigo em todos os momentos dessa longa caminhada, me incentivando e apoiando diariamente. Aos meus amigos, pela força para superar as adversidades e pelos momentos de descontração.

A minha Orientadora, Carla Silva, por ter acreditado que podíamos desenvolver esta pesquisa, por todo o auxílio durante esta caminhada de crescimento pessoal e profissional.

Enfim, a todos que contribuíram de alguma maneira para que eu conseguisse alcançar mais uma etapa na carreira acadêmica.

Lista de acrônimos e siglas

BPMN: *Business Process Modeling Notation*

ER: *Engenharia de Requisitos*

GAS2SPL: *Goals and Aspectual Scenarios to Software Product Lines*

GORE: *Goal-Oriented Requirements Engineering*

GS2SPL: *Goals and Scenarios to Software Product Lines*

KAOS: *Keep All Objectives Satisfied*

LPS: *Linha de Produto de Software*

MATA: *Modeling Aspects using a Transformation Approach*

MSVCM: *Modeling Scenario Variability as Crosscutting Mechanisms*

PLUSS: *Product Line Use case modeling for Systems and Software engineering*

RNF: *Requisito Não Funcional*

SD: *Strategic Dependency Model*

SR: *Strategic Rationale Model*

TaRGeT: *Test and Requirements Generation Tool*

Lista de Figuras

Lista de Quadros

Sumário

Capítulo 1 Introdução

A Engenharia de Requisitos (ER) é o ramo da Engenharia de Software que envolve as atividades relacionadas com a definição dos requisitos de software de um sistema, desenvolvidas ao longo do ciclo de vida de software [Kotonya e Sommerville 1998]. A ER pode ser aplicada, também, a Linhas de Produto de Software (LPS) [Pohl, Böckle e Linden 2005]. Uma LPS, também chamada de família de produtos, é um conjunto de sistemas de software que compartilham um conjunto comum e gerenciado de *features* para satisfazer necessidades específicas de um segmento particular de mercado [Czarnecki e Eisenecker 2000]. Uma *feature* pode ser vista como uma propriedade de sistema ou funcionalidade que é relevante para alguns interessados (*stakeholders*) e é utilizada para capturar características comuns e variáveis entre produtos de uma mesma família [Czarnecki e Eisenecker 2000]. Segundo Czarnecki (2004), toda *feature* cobre um conjunto de requisitos. Na ER para LPS, modelos de *feature* são utilizados para capturar as similaridades e variabilidades da linha de produto [Kang et al. 1990]. Entretanto, é um grande desafio estabelecer um relacionamento entre as *features* de um produto de software e os objetivos dos interessados (*stakeholders*) que originaram as respectivas *features* [Silva, Borba e Castro 2010]. *Stakeholders* ou interessados são indivíduos que usam, solicitam e desenvolvem o sistema, que se interessam para que ele seja desenvolvido, por exemplo, analistas, clientes, investidores e usuários [Kotonya e Sommerville 1998]. Neste trabalho quando nos referirmos a *stakeholders* utilizaremos o termo interessados. Com os modelos de *feature* também não é possível definir sistematicamente quais *features* devem ser selecionadas para um produto específico [Souza 2012]. Esse problema se agrava à medida que a LPS cresce, pois, segundo Asadi (2011), o número de *features* aumenta e analisá-las para decidir a melhor configuração para determinado produto não é uma tarefa trivial. Além disso, os interessados nem sempre estão familiarizados com a estrutura dos modelos de *feature* e, geralmente, sentem-se mais confortáveis em expressar suas necessidades em termos de objetivos ou metas que desejam alcançar. Diante disso, pode-se utilizar uma abordagem da Engenharia de Requisitos Orientada a Objetivos (do inglês, *Goal-Oriented Requirements Engineering* ou GORE) [Lamsweerde 2001] como uma forma de descobrir requisitos variáveis e comuns de uma LPS, além de tornar mais sistemática a configuração de um produto específico na família de produtos, pois buscam justificar a existência de *features* e a sua seleção para produtos da LPS através de modelos de objetivos.

1.1. Motivação

Algumas abordagens orientadas a objetivos têm sido propostas para modelar a variabilidade de requisitos, como o modelo de objetivos [Yu et al. 2008], *i** Aspectual [Silva et al. 2008], PL-AOVGraph [Santos, Silva e Batista 2011], G2SPL [Silva, Borba e Castro 2010], o mapeamento objetivo-*features* [Asadi 2011], E-SPL [Lima 2011] e GS2SPL [Souza 2012]. Abordagens GORE capturam tanto os objetivos dos interessados como os requisitos do sistema, de modo que o software a ser desenvolvido corresponda ao que realmente os interessados desejam [Lamsweerde 2001]. De acordo com Souza (2012), modelos de objetivos não são suficientes para modelar os aspectos comportamentais da LPS, eles permitem capturar as funcionalidades que a LPS deve oferecer para atender aos objetivos dos interessados, mas não conseguem descrever o comportamento detalhado de suas funcionalidades. Para modelar esses aspectos comportamentais, pode-se utilizar uma técnica de especificação de cenários, que facilita o entendimento por parte dos interessados e descreve o comportamento das funcionalidades do sistema [Maiden e Alexander 2004].

Mas, utilizar cenários sem relacioná-los a modelos de objetivos também não permite justificar a escolha de uma configuração, nem de relacionar o comportamento do sistema aos objetivos dos interessados. Portanto, faz-se necessário uma abordagem que integre modelos de objetivos, modelos de *feature* e cenários de caso de uso com variabilidade. Segundo Souza (2012), os modelos de objetivos seriam usados para justificar a presença das funcionalidades e atributos existentes nos modelos de *features*, bem como o comportamento dessas funcionalidades modelado nos cenários. Dentre as abordagens com essas características podemos citar o GS2SPL (do inglês, *Goals and Scenarios to Software Product Line*) [Souza 2012] que utiliza a técnica PLUSS [Eriksson, Börstler e Borg 2005] para especificar cenários com variabilidade.

Como Bonifácio e Borba (2009) afirmam, apesar dos benefícios da representação da variabilidade em cenários, existem abordagens que não apresentam uma separação clara entre o gerenciamento de variabilidade e a especificação de cenários de casos de uso. De fato, em muitas técnicas de cenários com variabilidade, como a técnica PLUSS, existem detalhes sobre variantes de produtos e informações de configuração que estão entrelaçadas. Nesse caso, as informações de configuração são chamadas de interesses transversais (do inglês, *crosscutting concerns*). Segundo Clements e Northrop (2002), interesses transversais são aqueles cujo comportamento pode ter impacto ou influenciar múltiplos módulos ou componentes do sistema. Os interesses transversais precisam ser identificados e

modularizados o mais cedo possível, pois, caso sua identificação seja tardia, poderá acarretar um maior custo na resolução de problemas que possam surgir no que diz respeito à evolução e manutenção [Whittle e Araújo 2004]. A separação de interesses é a base da orientação a aspecto, que modulariza os interesses transversais em aspectos para separá-los dos outros componentes do sistema. Cada aspecto encapsula um requisito ou cenário que afeta outros requisitos do sistema, facilitando a compreensão sobre o sistema e permitindo o reuso [Rashid et al. 2002]. De acordo com Oliveira et al. (2010), imaginemos um sistema que possui um requisito que estabelece que a autenticação do administrador do sistema é exigida antes de qualquer alteração de dados numa base de dados. Especificando e modularizando este requisito num aspecto, podemos estabelecer que o código de autenticação deve ser incluído antes da chamada a métodos que atualizem os dados através de mecanismos de composição. Isto permitirá uma reutilização do código e, caso seja necessário, a implementação de novas formas de autenticação. Neste caso, a alteração apenas necessita de ser efetuada em um só local, isto é, no aspecto que modulariza o requisito de autenticação do utilizador. A comunidade que utiliza Orientação a Aspectos no contexto da Engenharia de Requisitos[1] vem crescendo nos últimos anos, evidenciando pontos positivos que a orientação a aspectos traz para esta fase. Na ER, a modularização de interesses transversais em cenários de caso de uso é possível quando técnicas, como o MSVCM (do inglês, *Modeling Scenario Variability as Crosscutting Mechanisms*) [Almeida 2010], são utilizadas. No contexto de linhas de produto, Silva (2008) definiu diretrizes que visam utilizar recursos da orientação a objetivos e aspectos para representar a variabilidade tentando justificar a seleção de uma configuração específica de *features* diante de alternativas, visando melhorar o reuso e reduzir o tempo e os custos associados a evolução e a configuração de *features* para um produto específico. Portanto, este trabalho visa definir uma extensão da abordagem GS2SPL para suportar a especificação de cenários aspectuais usando a técnica MSVCM. A abordagem proposta será demonstrada através da especificação da LPS TaRGeT (*Test and Requirements Generation Tool*) [TaRGeT 2011] e, será avaliada com a aplicação ao *MyCourses* [SCORE 2011].

A partir desta motivação, a próxima seção descreve a questão de pesquisa e os objetivos que queremos alcançar com este trabalho.

---

[1] http://www.springer.com/us/book/9783642386398

1.2. Questão de Pesquisa e Objetivos

Este trabalho propõe responder a seguinte questão: Como integrar modelos de objetivos, modelos de *feature* e cenários de caso de uso com separação de interesses transversais visando promover modularidade em um processo de Engenharia de Requisitos para LPS?

Visando respondê-la, foi definido o objetivo geral que é adaptar uma abordagem existente de Engenharia de Requisitos para Linha de Produto de Software que integre modelos de objetivos, modelos de *feature* e cenários com separação de interesses transversais (ou, simplesmente, cenários aspectuais).

Esta nova abordagem é chamada GAS2SPL (*Goals and Aspectual Scenarios to Software Product Lines*) e obtém, sistematicamente, os cenários MSVCM a partir de modelos de objetivos e modelos de *feature*. A abordagem possibilita, também, a configuração dos produtos variantes da linha com base na prioridade atribuída pelos interessados aos requisitos não funcionais. Além disso, a abordagem possui um artefato para representar o conhecimento de configuração (do inglês, *configuration knowledge*), que relaciona o mapeamento entre o modelo de *feature* e os artefatos da LPS.

1.2.1. Objetivos Específicos

Para atingir o objetivo principal desse trabalho, foram definidos os seguintes objetivos específicos:

- Adaptar e integrar as diretrizes da abordagem GS2SPL para mapear modelos de objetivos em cenários aspectuais especificados com a técnica MSVCM.
- Adaptar o subprocesso de elaboração dos cenários e de configuração dos produtos da abordagem GS2SPL para contemplar os cenários aspectuais.
- Adaptar as diretrizes da técnica MSVCM para tratar da representação do *include* entre casos de uso e do *intertype declaration*.
- Definir a abordagem GAS2SPL.
- Avaliar a abordagem através da comparação entre os artefatos do TaRGeT e do *MyCourses* produzidos pelo GAS2SPL em relação aos artefatos produzidos pelo GS2SPL com relação à modularidade e expressividade.

1.3. Metodologia

Inicialmente, aprofundamos o conhecimento sobre a abordagem GS2SPL e a técnica de especificação de cenários MSVCM, identificando as suas características e oportunidades de melhoria. Analisamos a abordagem GS2SPL para identificar as suas limitações em definir cenários de caso de uso com separação de interesses transversais. Paralelamente, elaboramos cenários de caso de uso da LPS TaRGeT (do inglês, *Test and Requirements Generation Tool*) através da aplicação das diretrizes de Santander (2002). Comparamos os cenários obtidos com os cenários gerados pelo GS2SPL, especificados com a técnica PLUSS [Eriksson, Börstler e Borg 2005]. O propósito desse estudo foi entender como mapear modelos *i** em cenários de caso de uso e, assim, poder definir as diretrizes para gerar os cenários com variabilidade e separação de interesses transversais com a técnica MSVCM. Estendemos a técnica MSVCM para tratar da representação de inclusão (do inglês, *include*) entre casos de usos. Em seguida, considerando a análise das abordagens, elaboramos a abordagem GAS2SPL (do inglês, *Goals and Aspectual Scenarios to Software Product Lines*), que possibilita obter uma especificação de requisitos mais detalhada e modularizada para a LPS e uma configuração de produto mais sistemática, devido à utilização de modelos de objetivos e requisitos não funcionais. Além de permitir a obtenção sistemática de cenários aspectuais, em MSVCM, a partir do modelo de objetivos em conjunto com o modelo de *feature*. Todas as atividades de GAS2SPL foram demonstradas através da utilização da LPS TaRGeT (do inglês, *Test and Requirements Generation Tool*). Por fim, realizou-se a avaliação de GAS2SPL através da aplicação das atividades da abordagem ao TaRGeT e *MyCourses.* Inicialmente, foram elaborados os cenários do *MyCourses* em PLUSS, utilizando o processo definido por GS2SPL e, os cenários, *advices* e *intertype declarations* do *MyCourses* produzidos pela abordagem GAS2SPL. Então, visando comparar os artefatos gerados do TaRGeT e do *MyCourses*, realizou-se uma avaliação quantitativa através da aplicação de métricas que analisaram a modularidade (grau de espalhamento de *features* [Almeida 2010] e grau de entrelaçamento de cenários [Almeida 2010]) e a expressividade (quantidade de símbolos utilizados para especificar o conhecimento de configuração [Alferez et al. 2013]).

1.4. Estrutura da Dissertação

O trabalho divide-se em mais seis capítulos, organizados da seguinte forma:

O Capítulo 2 apresenta os principais conceitos de Engenharia de Requisitos para Linhas de Produto de Software, Engenharia de Requisitos Orientada a Objetivos e cenários de casos de uso com separação de interesses transversais. Além de apresentar alguns trabalhos que tratam da integração de abordagens orientadas a objetivos com técnicas de especificação de cenários com separação de interesses transversais que servirão de base para a abordagem proposta por este trabalho.

O Capítulo 3 descreve detalhadamente as atividades da abordagem GAS2SPL e a aplicação ao exemplo TaRGeT.

O Capítulo 4 apresenta a aplicação da abordagem GAS2SPL ao *MyCourses*.

O Capítulo 5 apresenta a avaliação da abordagem através da aplicação de métricas aos artefatos do TaRGeT e *MyCourses* elaborados pelas abordagens GAS2SPL e GS2SPL.

Por fim, o Capítulo 6 apresenta as considerações obtidas durante o desenvolvimento desse trabalho, assim como as contribuições e limitações, além de citar os trabalhos futuros.

Capítulo 2 Fundamentação Teórica

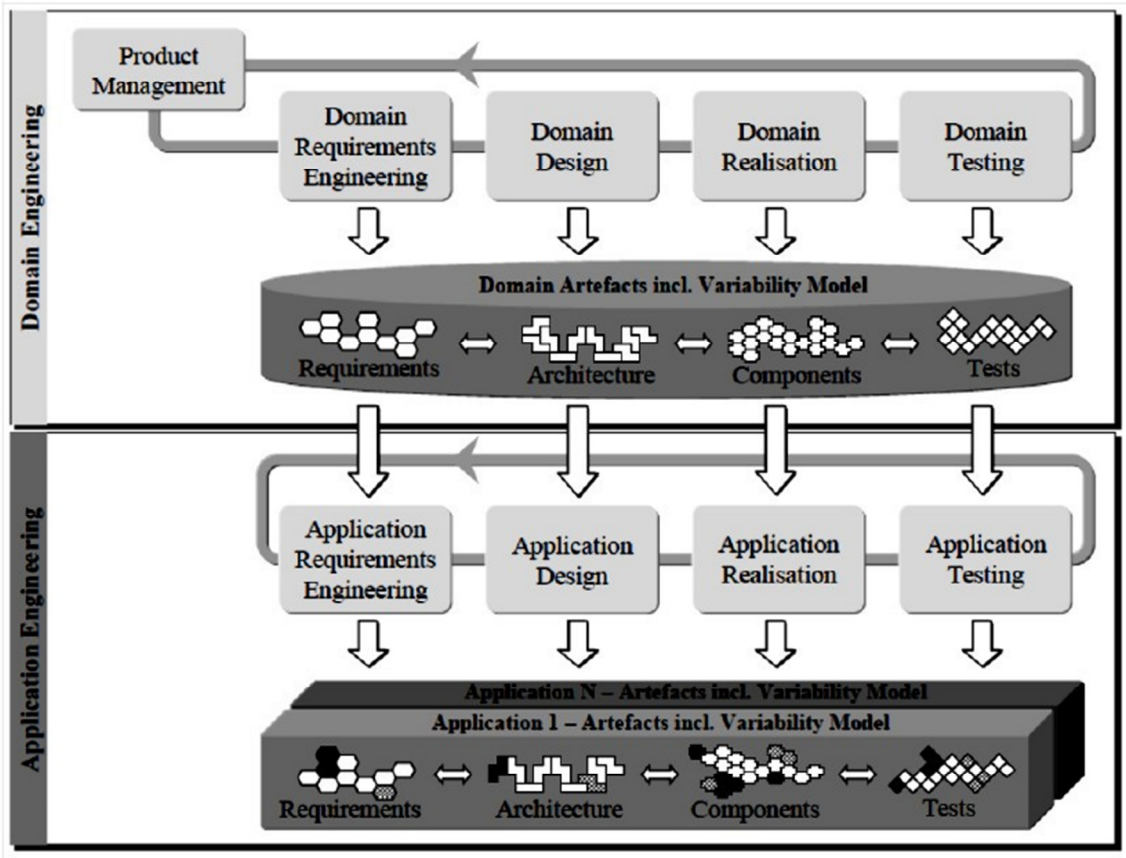
Este capítulo tem como objetivo realizar um levantamento bibliográfico sobre os principais conceitos que estão relacionados ao tema deste trabalho: Linha de Produto de Software, Engenharia de Requisitos para Linhas de Produto de Software, Engenharia de Requisitos Orientada a Objetivos, Orientação a Aspectos e cenários de casos de uso com separação de interesses transversais. Este capítulo apresenta, também, os trabalhos relacionados.

2.1. Linhas de Produto de Software

Uma Linha de Produto de Software (LPS) ou Família de Software, de acordo com Clements e Northrop (2002), é um conjunto de sistemas de software que compartilham um conjunto comum e gerenciado de características, que satisfazem as necessidades específicas de um segmento ou missão particular de mercado e que são desenvolvidos de maneira pré-estabelecida de um conjunto comum de artefatos que são compartilhados por todos os produtos que fazem parte da família (do inglês, *core assets*).

Segundo Pohl, Böckle e Linden (2005), a Engenharia de uma Linha de Produtos de Software define dois processos, a Engenharia de Domínio e a Engenharia de Aplicação. A primeira é o processo de LPS onde os pontos comuns e variáveis da linha de produtos são definidos. Essas semelhanças e diferenças dos produtos de uma LPS são importantes para a análise do domínio. A segunda é o processo de LPS onde as aplicações são construídas através da reutilização dos artefatos de domínio e pela exploração da variabilidade da linha de produtos. Na Figura 1 encontram-se representados os subprocesso da engenharia de domínio e de aplicação, bem como as suas atividades através do *framework* de Engenharia de Linha de Produtos de Software definido por Pohl, Böckle e Linden (2005).

Figura 1 Framework da Engenharia de Linha de Produto de Software [Pohl, Böckle e Linden 2005].



No processo da Engenharia de Domínio, de acordo com Pohl, Böckle e Linden (2005), temos as seguintes atividades: gerenciamento de produto, engenharia de requisitos de domínio, realização de domínio e testes de domínio. A atividade de **Gerenciamento de Produto** lida com os aspectos econômicos da linha de produto de software e, em particular, com a estratégia de mercado. **Engenharia de Requisitos de Domínio** engloba todas as atividades de elicitação, análise, documentação e validação de requisitos da LPS. **Realização de Domínio** trata dos detalhes do projeto e da implementação dos componentes reutilizáveis de software. **Testes de Domínio** consiste em validar e verificar os componentes reutilizáveis.

Por sua vez, na Engenharia de Aplicação as atividades são: engenharia de requisitos da aplicação, projeto da aplicação, realização da aplicação e testes da aplicação. **Engenharia de Requisitos da Aplicação** tem como objetivo elicitar e documentar artefatos de requisitos para uma aplicação específica e, ao mesmo tempo, reusar ao máximo possível os requisitos de domínio. **Projeto da Aplicação** está relacionada à configuração da arquitetura de referência, além da inclusão de adaptações específicas ao produto, para derivar a arquitetura da aplicação. Na **Realização da Aplicação** é feita a seleção e configuração dos componentes de software reutilizáveis e a implementação das características específicas da aplicação. **Teste da**

Aplicação valida a aplicação conforme os requisitos documentados.

Como afirmam Pohl, Böckle e Linden (2005), as principais vantagens das LPS são a redução dos custos de desenvolvimento, o aumento da qualidade dos artefatos e a redução no tempo de entrega. Havendo a reutilização de artefatos não é necessário desenvolver componentes do zero, contribuindo para a redução nos custos. Portanto, há um aumento considerável na qualidade, visto que os *core assets* (ou artefatos de software) de uma LPS são reusados em muitos produtos, significando que foram testados e revisados várias vezes aumentando a chance de detecção e correção de falhas.

Quando estamos realizando o processo da Engenharia de Domínio de uma LPS deve-se analisar as similaridades e variabilidades buscando produzir produtos com os mesmos *core assets*, mas com diferenças que possibilitem o desenvolvimento de aplicações customizadas através do reuso de artefatos pré-definidos e ajustáveis. Portanto, a variabilidade de uma Linha de Produto de Software distingue diferentes aplicações da linha de produto. De acordo com Kang et al. (1990), as similaridades e variabilidades de uma LPS podem ser capturadas através do modelo de *feature* elaborado na etapa de Engenharia de Requisitos do Domínio da Linha de Produto .

Existem diversos estudos de caso que são utilizados na academia para validação de processos de abordagens como, por exemplo, e-shop [Pohl, Metzger, 2006], MobileMedia (2011), *MyCourses* (2011), Car Crash Crisis Management System [Kienzle, Guelfi e Mustafiz 2010, TaRGeT [TaRGeT 2011], entre outros.

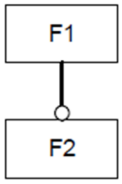
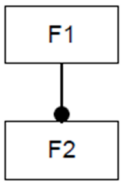
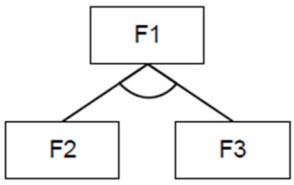
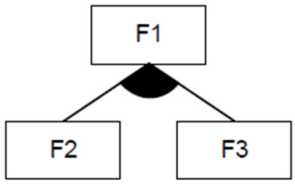
A abordagem GAS2SPL será demonstrada através da especificação da LPS TaRGeT (*Test and Requirements Generation Tool*) [TaRGeT 2011] e, será avaliada com a aplicação a LPS *MyCourses* [SCORE 2011]. A TaRGeT é uma linha de produto desenvolvida pela Motorola[2] para produzir ferramentas que geram suítes de testes automaticamente a partir de documentos de requisitos. Cada suíte de testes gerada é formada por vários casos de teste que são criados de maneira independente. Os requisitos utilizados pela ferramenta, para gerar a suíte de testes, são descritos na forma de casos de uso. Esses casos de uso são escritos em linguagem natural e devem seguir um padrão pré-definido. O *MyCourses* é um sistema para agendamento de cursos, adaptado por Lima (2011) para possuir variabilidade quando a abordagem E-SPL estava sendo desenvolvida. O *MyCourses* será detalhado na seção 5.1.

[2] http://twiki.cin.ufpe.br/twiki/pub/TestProductLines/TaRGeTProductLine/TaRGeT_MIT_License.pdf

2.1.1. Modelo de *feature*

Atualmente na ER para LPS, se utiliza de uma técnica bem conhecida para representar os conceitos ou *features* de um domínio do sistema e detalhar as restrições entre essas *features*, o modelo de *feature.* O modelo de *feature* foi proposto por Kang, Cohen, et al. (1990) como parte do método Análise de Domínio Orientado a *Features* (do inglês, *Feature Oriented Domain Analysis* - FODA). Este é utilizado para capturar similaridades e variabilidades de famílias de produtos [Kang et al. 1990]. Um modelo de *feature* é um grafo hierárquico, onde o nó raiz representa um conceito, isto é, o sistema de software que está sendo desenvolvido e os demais nós, as *features*. Cada *feature* pode ser refinada em sub-*features*. Estas, por sua vez, podem ser classificadas em obrigatória (todos os produtos da família devem contê-la), opcional (os produtos podem contê-la ou não) ou alternativa. As *features* alternativas podem ser de dois tipos: Ou-Exclusivo (os produtos devem conter exatamente uma *feature* de um grupo de *features*) e Ou-Inclusivo (os produtos devem conter pelo menos uma de um grupo de *features*). O Quadro 1 apresenta a descrição e a representação dos tipos de *features*.

Quadro 1 Descrição dos tipos de *features*

Tipo de *Feature*	Representação	Descrição
Opcional		Uma *feature* opcional pode ou não estar presente em um produto concreto. A cardinalidade é [0..1]. Por exemplo, ao escolher a *feature* F1, a *feature* F2 pode ser escolhida ou não.
Obrigatória		Uma *feature* obrigatória deve estar presente exatamente uma vez em um produto concreto. A cardinalidade é [1..1]. Por exemplo, ao escolher a *feature* F1, é obrigatório a seleção da *feature* F2.
Ou-Exclusivo		Uma cardinalidade de grupo <1..1> indica que exatamente uma *feature* do grupo pode ser selecionada. Por exemplo, ao escolher a *feature* F1, pode-se escolher apenas uma das *features* F2 ou F3.
Ou-Inclusivo		Uma cardinalidade de grupo <1..k> indica que pelo menos uma e no máximo k *features* do grupo podem ser selecionadas e que k é o número total de *features* no grupo. Por exemplo, ao escolher a *feature* F1, pode-se escolher um das *features* (F2 ou F3) ou ambas.

O modelo de *feature* não permite justificar claramente porque um determinado conjunto de *features* foi selecionado para configurar um produto específico na LPS [Silva, Borba e Castro 2010]. Portanto, é um grande desafio estabelecer um relacionamento entre quais *features* de um produto de software foram geradas a partir de quais objetivos dos interessados [Silva, Borba e Castro 2010]. Também não há uma forma sistemática para descobrir quais *features* farão parte da LPS e nem para determinar quais *features* serão opcionais, obrigatórias ou alternativas [Silva, Borba e Castro 2010]. Segundo Borba (2009), modelos de *feature* não capturam requisitos não funcionais explicitamente e nem a influência positiva ou negativa destes requisitos para alcançar configurações alternativas de uma aplicação na LPS.

Por outro lado, abordagens de Engenharia de Requisitos Orientada a Objetivos (do inglês, *Goal-Oriented Requirements Engineering* ou GORE) [Lamsweerde 2001] podem ser usadas como uma forma de descobrir requisitos variáveis e comuns de uma LPS, além de tornar mais sistemática a configuração de um produto específico na família de produtos. Por isso, algumas abordagens orientadas a objetivos têm sido propostas para modelar a variabilidade de requisitos [Silva, Borba e Castro 2010].

2.2. Engenharia de Requisitos Orientada a Objetivos

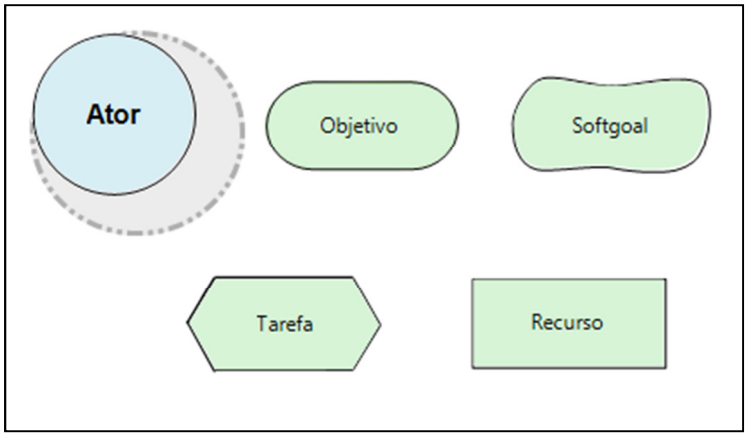
O objetivo do sistema (do inglês, *goals*), de acordo com Chung (2000), é um propósito que o sistema deve alcançar após a execução de uma ação. Os objetivos são refinados em sub-objetivos e decompostos em vários requisitos [Lamsweerde 2001]. Abordagens de Engenharia de Requisitos Orientada a Objetivos (do inglês, *Goal-Oriented Requirements Engineering* ou GORE) podem ser utilizadas para capturar tanto os objetivos dos interessados como os requisitos do sistema, de modo que o software a ser desenvolvido corresponda ao que realmente os interessados desejam [Lamsweerde 2001]. Podemos entender objetivos, neste contexto, como propriedades desejadas do sistema que são expressas pelos interessados [Lamsweerde 2001]. Ainda de acordo com Lamsweerde (2001), essa abordagem considera os objetivos da organização e dos atores como a fonte de requisitos (funcionais e não funcionais). Assim, elicitam-se primeiro objetivos para, em seguida, elicitar requisitos. Existem diversas abordagens GORE, dentre elas podemos citar o *i** [Yu 1995] e o GS2SPL [Souza 2012]. Ambas as abordagens serão apresentadas, de forma resumida, nas seções a seguir.

2.2.1. Framework *i**

O *framework i** [Yu 1995] é uma abordagem orientada a objetivos utilizada para obter uma compreensão aperfeiçoada sobre os relacionamentos da organização, de acordo com os diversos atores do sistema. De acordo com Silva, Borba e Castro (2010), o *i** é uma abordagem de Engenharia de Requisitos Orientada a Objetivos, desenvolvido para modelar e analisar, sob uma visão estratégica e intencional, processos que envolvem vários participantes. Santos (2008) declara que os atores são intencionais por possuírem motivações, desejos, necessidades e razões por trás de suas ações. E, são estratégicos quando não focam apenas o seu objetivo imediato, mas quando se preocupam com as implicações de seu relacionamento estrutural com outros atores. Castro, Alencar e Silva (2006) afirmam que o *framework i** permite compreender melhor os relacionamentos da organização, entender as razões envolvidas nos processos de decisões e descrever potenciais alternativas do sistema pretendido. O *framework i** é uma abordagem centrada nos interessados do sistema e nas dependências entre eles, durante a fase inicial da Engenharia de Requisitos, e pode ser utilizado para modelar o ambiente do sistema. Os engenheiros de requisitos modelam os interessados como atores e suas intenções como objetivos. Os atores são definidos como qualquer entidade para a qual podem ser atribuídas dependências intencionais [Yu et al., 2011].

Yu et al. (2011) definiram quatro elementos intencionais: objetivo (do inglês, *goal*), tarefa (do inglês, *task*), recurso (do inglês, *resource*) e objetivo-*soft* (do inglês, *softgoal*). Um objetivo é uma condição ou estado de coisas que o ator gostaria de alcançar. Uma tarefa especifica um jeito particular de realizar algo. Um recurso é uma entidade, física ou não, que pode ser disponibilizada pelo sistema. Finalmente, um objetivo-*soft* também é uma condição que o ator gostaria de alcançar. Porém, ao contrário do conceito de objetivo, o critério para a condição ser alcançada não é rigorosamente definido, inicialmente. Os elementos de modelagem do *i** são apresentados na Figura 2.

Figura 2 Elementos de modelagem do *framework i**



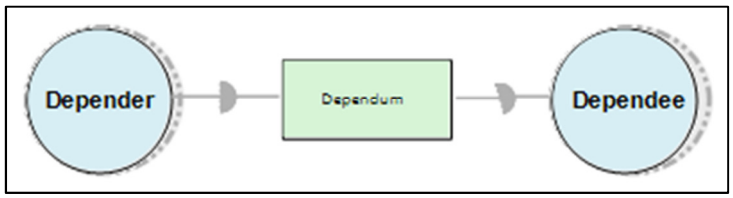
Cada objetivo analisado do ponto de vista do seu ator resulta em um conjunto de dependências entre pares de atores. Um ator pode realizar ações, atingir objetivos e possuir dependências. Desta forma, dependem da ajuda de outros atores para que seus objetivos sejam atingidos, suas tarefas realizadas, recursos sejam fornecidos e os objetivos-*soft* sejam satisfeitos parcial ou totalmente.

O *i** possui dois tipos de modelos, o modelo de Dependências Estratégicas (do inglês, *Strategic Dependency* – SD) que descreve as relações de dependência entre os atores e, o modelo de Razão Estratégica (do inglês, *Strategic Rational* – SR) que ilustra, internamente, como os atores atingem os seus objetivos.

O Modelo de Dependência Estratégica, de acordo com Santander (2002), é composto por nós e ligações. Os nós representam os atores no ambiente e as ligações são as dependências entre os atores.

O ator que depende de alguma forma de outro ator é chamado de *Depender* e o ator que atende e satisfaz o *Depender* é denominado de *Dependee*. O objeto ou elemento de dependência entre *Depender* e *Dependee* é denominado de *Dependum* (Figura 3).

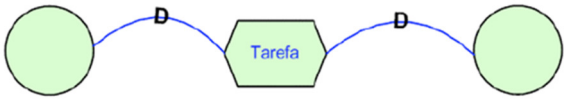
Figura 3 Componentes de uma dependência



As associações permitem descrever graficamente as relações entre atores. Uma de suas classificações é ISA, que representa uma generalização, com um ator a ser um caso especializado de outro ator.

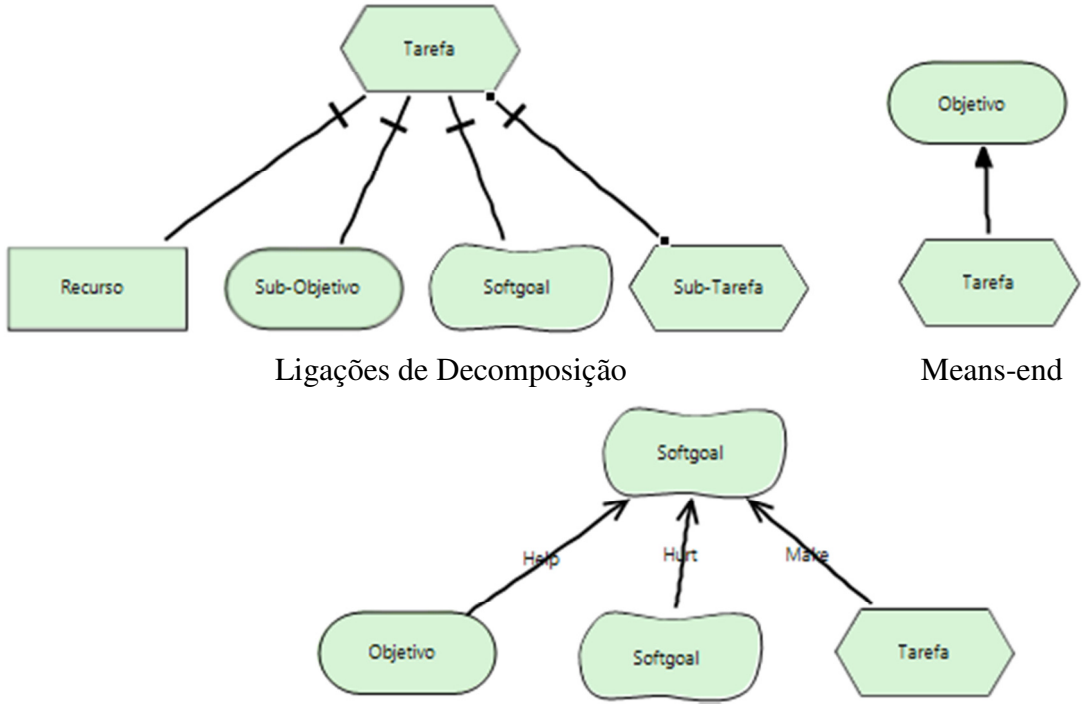
Os tipos de dependência que podem ser representadas pelo *framework i** são: dependência de objetivo, de tarefa, de recurso e de objetivo-*soft* e estão representados no Quadro 2.

Quadro 2 Tipos de dependência entre atores

Dependência	Representação	Descrição
Objetivo		Na dependência de objetivo, o ator *depender* necessita que o ator *dependee* execute as ações necessárias para que sua intenção seja satisfeita.
Tarefa		Na dependência de tarefa, o *dependee* é requisitado a executar uma atividade, sendo informado sobre o que deve ser feito
Recurso		Na dependência de recurso, o *dependee* deve fornecer a entidade (física ou informacional) ao *depender*.
Objetivo-soft		A dependência de objetivo-*soft* é similar à dependência de objetivo, exceto pelo fato de que a condição de sucesso não é definida inicialmente.

O Modelo de Razão Estratégica, de acordo com Santander (2002), permite compreender e modelar de forma mais detalhada as razões associadas com cada ator em relação a processos da organização e como as suas dependências são expressas. Além dos elementos que estão presentes no modelo SD, os elementos intencionais que aparecem no modelo SR são: relações de meio-fim (do inglês, *means-end*), contribuição para objetivos-*soft* e decomposição de tarefa (Figura 4).

Figura 4 Tipos de relações permitidas no modelo SR do framework i*



Ligações de Decomposição

Means-end

Ligações de Contribuição (make, break, some+, some-, help, hurt, unknown, and, or)

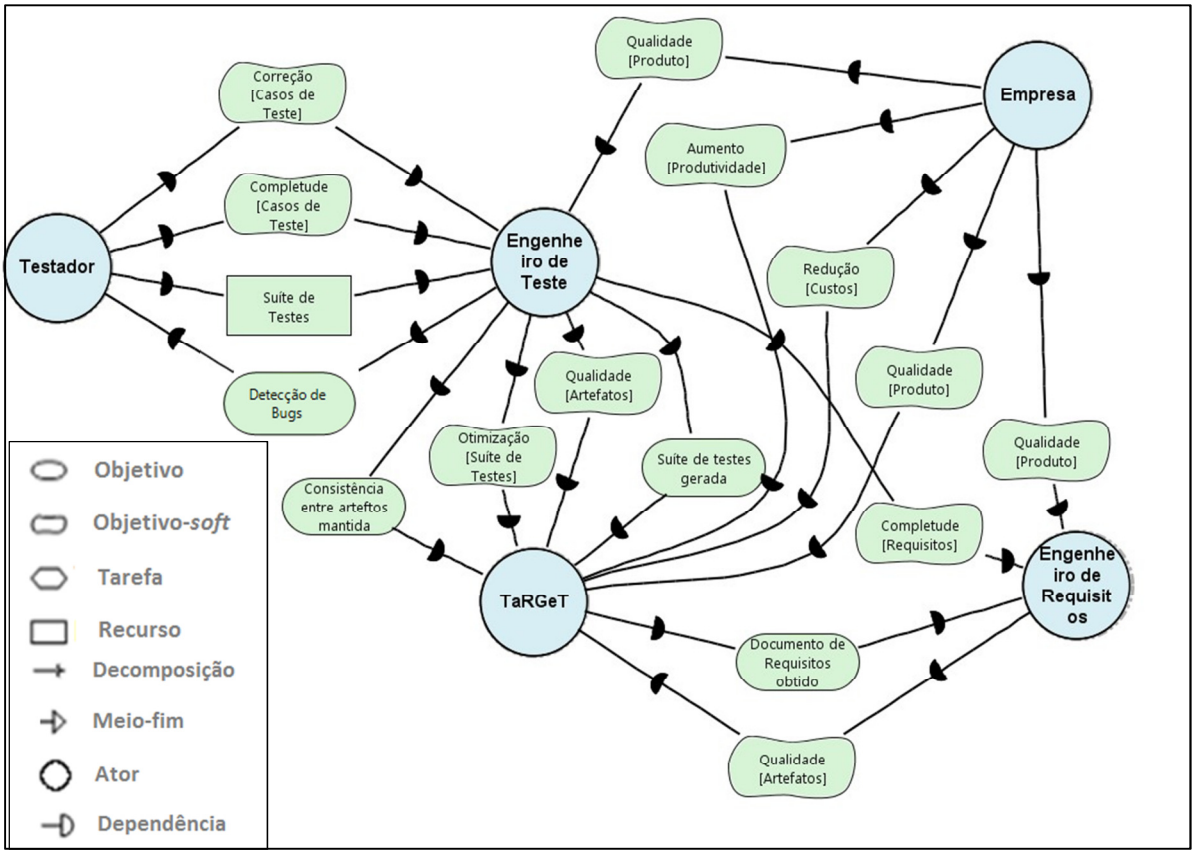
De acordo com Yu et al. (2011), na ligação de decomposição de tarefa, uma tarefa é decomposta em termos de subcomponentes que podem ser: objetivos, tarefas, recursos, e/ou objetivos-*soft*. De acordo com Yu et al. (2011), a ligação meio-fim consiste em um relacionamento entre um nó fim (objetivo, tarefa, recurso, ou objetivo-*soft*) e diversas maneiras para consegui-lo. As ligações de contribuição para objetivo-*soft* são utilizadas para representar o grau de influência (positiva ou negativa) de outros elementos do sistema na satisfação dos objetivos-*soft*. Por possuírem um caráter subjetivo, não existe um critério bem definido para verificar se a condição foi atingida e, a sua satisfação é feita através da análise e interpretação dos interessados. Estas ligações podem ser dos seguintes tipos (Figura 4):

- Make – contribuição positiva com força suficiente para satisfazer um objetivo-*soft*;
- Break – contribuição negativa com força suficiente para negar um objetivo-*soft*;
- Some+ – contribuição positiva cuja força é desconhecida;
- Some- – contribuição negativa cuja força é desconhecida;
- Help – contribuição parcialmente positiva, não suficiente por si só para satisfazer o objetivo-*soft*;

- Hurt – contribuição parcialmente negativa, não suficiente por si só para negar o objetivo-*soft*;
- Unknown – contribuição cuja polaridade é desconhecida;
- And – o ascendente é satisfeito se todos os descendentes forem;
- Or – o descendente é satisfeito se qualquer um dos descendentes for;

Exemplos de modelos SD e SR da linha de produto TaRGeT são apresentados na Figura 5 e Figura 6, respectivamente.

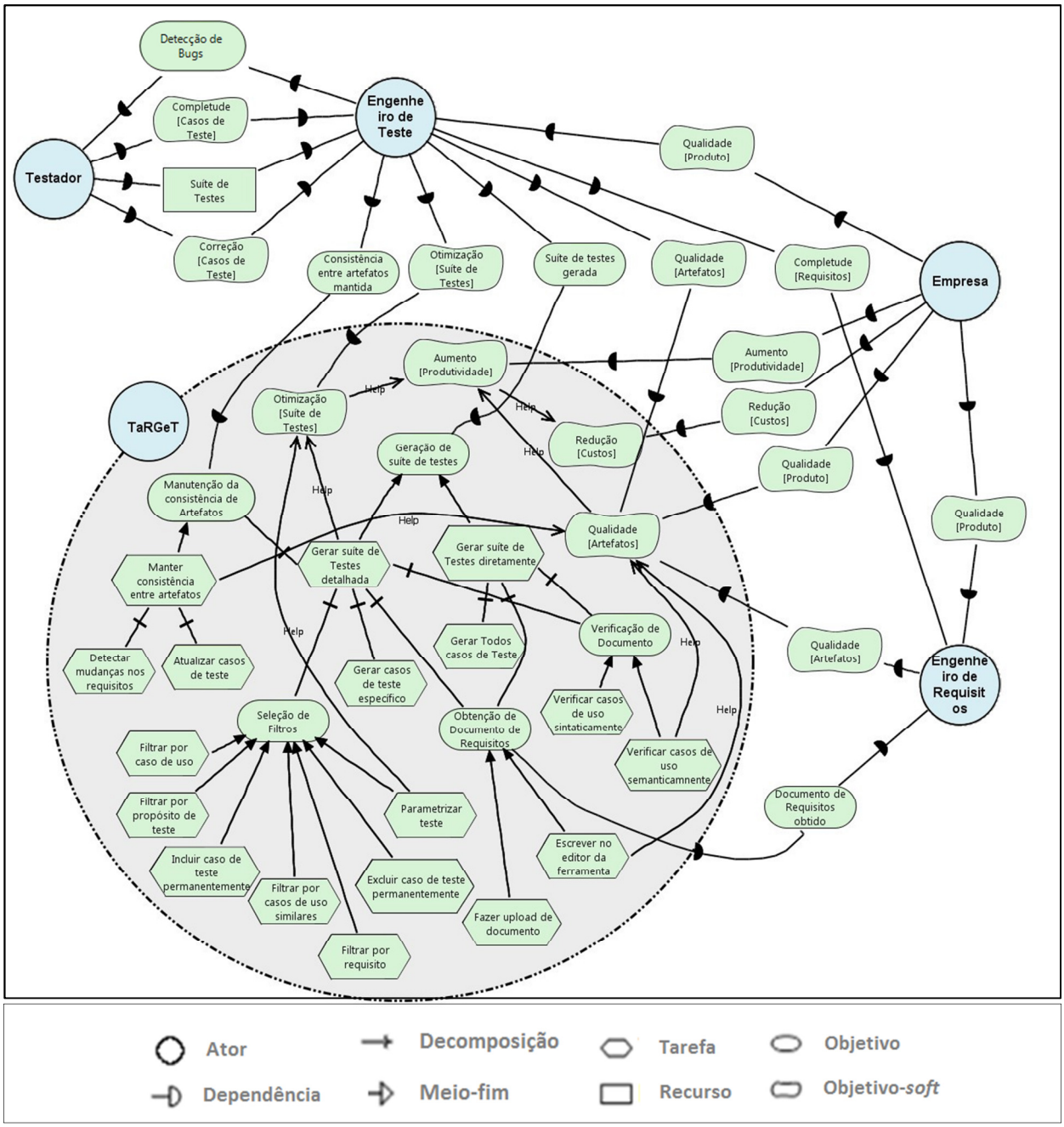
Figura 5 Modelo SD do TaRGeT [Souza 2012]



Observando a Figura 5 podemos perceber que temos cinco círculos que representam os atores envolvidos: a própria TaRGeT, o testador, a empresa que utilizará o TaRGeT, o engenheiro de requisitos e o engenheiro de testes. Em seguida, foram modeladas as dependências entre os atores. Como pode ser visto na figura, **Empresa** depende de **Engenheiro de Requisitos** e de **Engenheiro de Testes** para satisfazer o objetivo-*soft* **Qualidade [Produto]**. **Empresa** também espera que **TaRGeT** contribua para os objetivos-*soft* **Aumento [Produtividade]**, **Redução [Custos]** e **Qualidade [Produto]**. **Engenheiro de**

Requisitos espera que **TaRGeT** contribua para **Qualidade [Artefatos]** e este, por sua vez, depende de **Engenheiro de Requisitos** para seu objetivo **Documento de Requisitos Obtido** ser atendido. **Engenheiro de Testes** depende de **TaRGeT** para atender aos objetivo-*soft* **Qualidade [Artefatos]** e **Otimização [Suíte de Testes]**, bem como para alcançar os objetivos **Suíte de Testes Gerada** e **Consistência entre Artefatos Mantida**. **Engenheiro de Teste** também deseja que **Engenheiro de Requisitos** colabore para o objetivo-*soft* **Completude [Requisitos]**, além de depender de **Testador** para atender ao objetivo **Detecção de Bugs**. Por fim, **Testador** precisa que **Engenheiro de Testes** forneça o recurso **Suíte de Testes** e contribua para a satisfação dos objetivos-*soft* **Correção [Casos de Teste]** e **Completude [Casos de Teste]**.

Figura 6 Modelo SR do TaRGeT [Souza 2012]



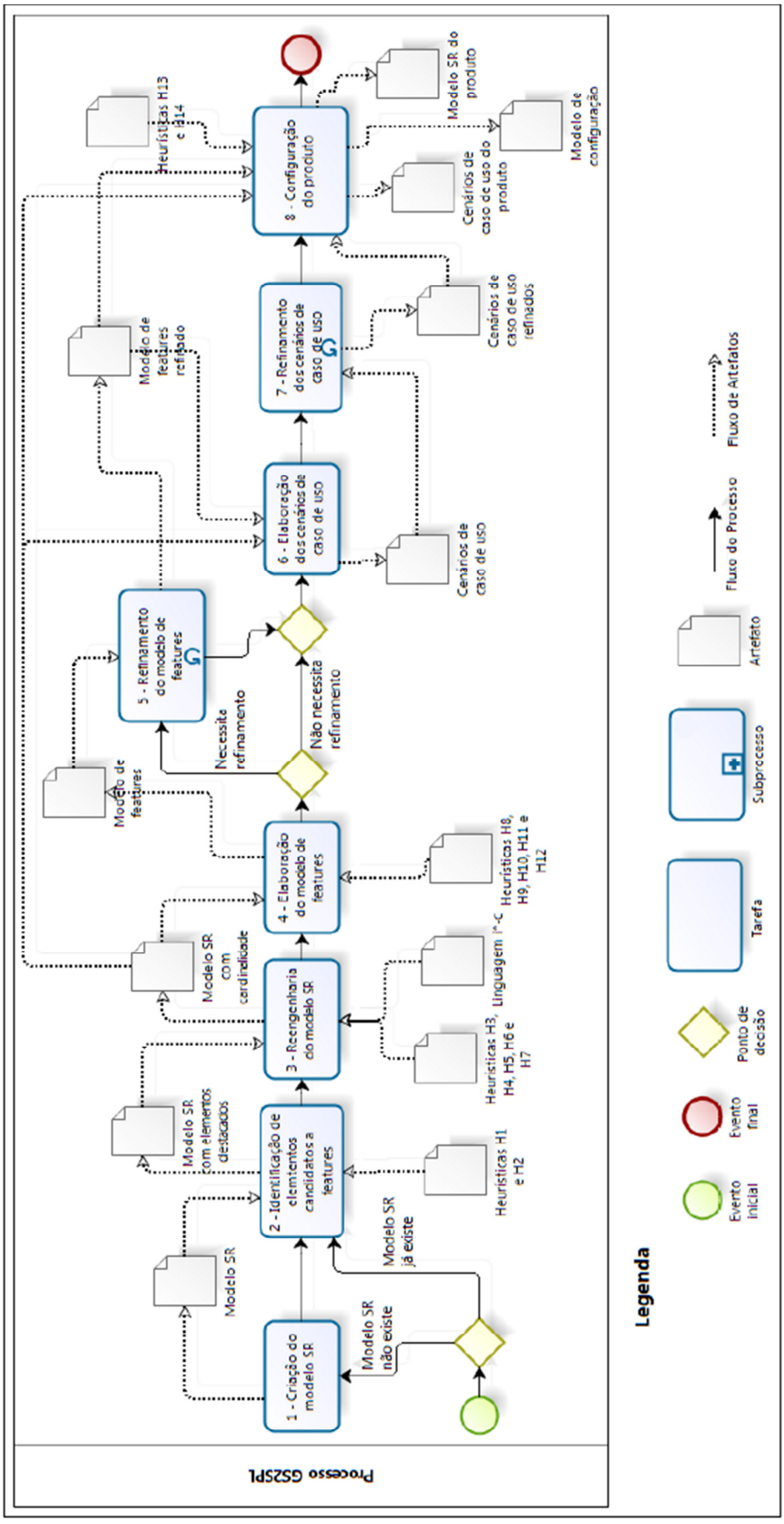
Como podemos ver na Figura 6 o objetivo principal de **TaRGeT** é **Geração de Suíte de Testes**, que vai atender à dependência **Suíte de Testes Gerada** com o ator **Engenheiro de Testes**. Este objetivo é atendido pela execução das tarefas **Gerar Suíte de Testes Detalhada** ou **Gerar Suíte de Testes Diretamente**, pois a mesma está ligada a eles por uma relação do tipo meio-fim. A tarefa **Gerar Suíte de Testes Detalhada** é decomposta em **Manutenção da Consistência de Artefatos**, **Seleção de Filtros**, **Gerar Caso de Teste Específico** e **Obtenção de Documento de Requisitos**. O objetivo **Manter Consistência entre Artefatos** que

atende à dependência **Consistência entre Artefatos Mantida** é realizado pela tarefa **Manter Consistência entre Artefatos**, que é decomposta nas tarefas **Detectar Mudanças nos Requisitos** e **Atualizar Casos de Teste**. O objetivo **Seleção de Filtros** está relacionado às tarefas **Filtrar por Caso de Uso**, **Parametrizar Testes**, **Filtrar por Propósito de Teste**, **Incluir Caso de Teste Permanentemente**, **Excluir Caso de Teste Permanentemente**, **Filtrar por Casos de Uso Similares** ou **Filtrar por Requisito** por meio de uma ligação meio-fim. **Obtenção de Documento de Requisitos**, que atende à dependência **Documento de Requisitos Obtido**, pode ser satisfeito pelas tarefas **Fazer *Upload* de Documento** ou **Escrever no Editor da Ferramenta**. **Gerar Suíte de Testes Diretamente** é decomposta em **Gerar Todos os Casos de Teste**, **Obtenção de Documento de Requisitos** e **Verificação de Documento**. O objetivo **Verificação de Documento** pode ser alcançado pela tarefa **Verificar Casos de Uso Sintaticamente** ou **Verificar Casos de Uso Semanticamente**. Quanto aos objetivos-*soft*, **Otimização [Suíte de Testes]** recebe contribuição positiva das tarefas **Parametrizar Teste** e **Gerar Suíte de Teste Detalhada**, e contribui positivamente para **Aumento [Produtividade]**. **Qualidade [Artefatos]** recebe contribuição positiva de **Verificar Casos de Uso Semanticamente**, **Escrever no Editor da Ferramenta** e **Manter Consistência entre Artefatos**, e também contribui positivamente para **Aumento [Produtividade]**. O objetivo-*soft* **Aumento [Produtividade]** contribui positivamente para **Redução [Custos]**.

2.2.2. GS2SPL

A *Goals and Scenarios to Software Product Lines* (GS2SPL) [Souza 2012], é uma abordagem da Engenharia de Requisitos Orientada a Objetivos (GORE) que integra uma abordagem GORE (G2SPL) [Silva, Borba e Castro 2010] com uma técnica de especificação de cenários de caso de uso com variabilidade, além de incluir um subprocesso para configuração de aplicações específicas de uma LPS com base na priorização de requisitos não funcionais. A abordagem tem como objetivo incluir entre seus artefatos, não só modelos *i** e de *feature*, como também cenários de caso de uso com variabilidade e a configuração definida com base na prioridade atribuída pelos interessados aos requisitos não funcionais. A Figura 7 apresenta o modelo do processo da abordagem.

Figura 7 Processo da abordagem GS2SPL



A abordagem é dividida em oito atividades, cada uma possuindo heurísticas que definem seus passos internos. Destas, sete fazem parte da Engenharia de Domínio de LPS e uma da Engenharia de Aplicação. As atividades serão apresentadas, de forma resumida, a seguir.

A primeira atividade da abordagem, **criação do modelo SR,** tem como objetivo construir o modelo SR do *framework i**. Utilizamos os artefatos gerados na etapa de elicitação de requisitos para a elaboração do modelo SR. De acordo com Souza (2012), para criar o modelo SR pode-se utilizar qualquer técnica de elicitação de requisitos, como entrevistas ou observações, desde que seja possível capturar os atores, seus objetivos e a maneira com que eles interagem entre si para alcançar as suas metas. Ainda segundo Souza (2012), deve-se criar primeiro o modelo SD, apresentado na Figura 5, e depois expandir a fronteira de cada ator para analisar como as dependências são atendidas internamente, desenvolvendo o modelo SR. Como forma de facilitar o entendimento, apresentamos na Figura 6, apenas a expansão da fronteira do ator **TaRGeT** representado por um círculo tracejado.

Segundo Silva, Borba e Castro (2010), modelos *i** não foram desenvolvidos com a intenção de modelar LPS, apesar de ser capaz de modelar variabilidade, e, portanto, faz-se necessário ajustar este modelo para que se possa capturar informação relativa à presença opcional e obrigatória de algumas *features* em um produto da LPS. Assim, foi definida uma extensão da linguagem *i**, chamada *i**-c (*i** com cardinalidade) [Borba 2009], onde adicionou-se cardinalidade, permitindo capturar mais informação sobre a variabilidade de uma LPS.

A segunda atividade do processo, **identificação dos elementos candidatos a *features***, recebe como artefato de entrada o modelo SR (Figura 6) gerado na atividade anterior e produz o mesmo modelo com alguns elementos destacados. Os elementos destacados são aqueles que podem originar *features*.

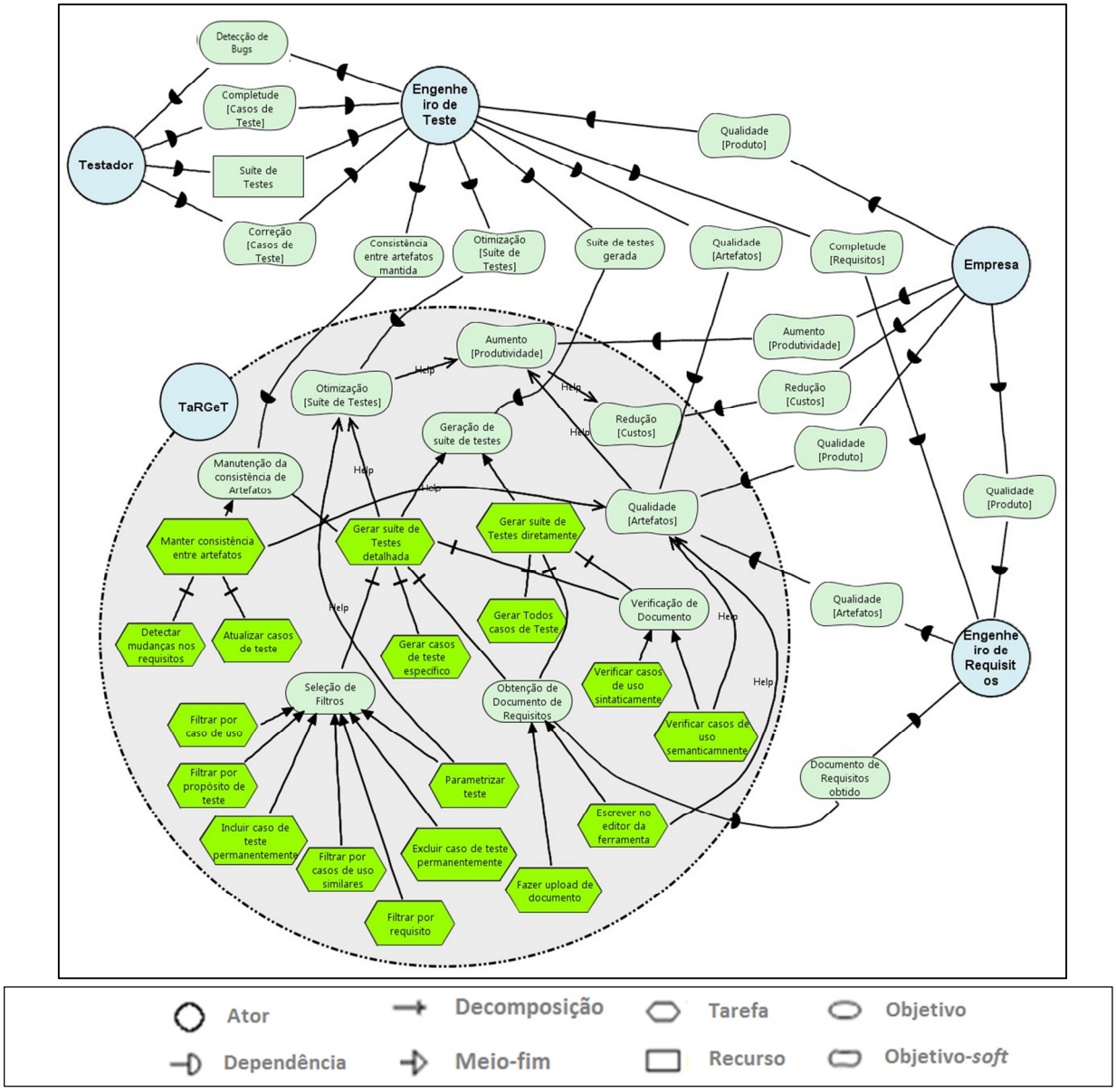
De acordo com Souza (2012), os elementos do tipo objetivo, em *i*,* não podem ser *features*, pois representam intenções. Os objetivos-*soft*, em *i*,* representam requisitos não funcionais da LPS e, não podem ser *features,* pois nos modelos *i** existem tarefas que operacionalizam os objetivos-*soft*. Então, *features* podem ser obtidas, apenas, a partir dos elementos do tipo tarefa e recurso, Borba (2009) propôs duas heurísticas que são apresentadas a seguir.

H1: As *features* podem ser extraídas a partir dos elementos do tipo tarefa, que determinam funcionalidades do sistema e, a partir de elementos do tipo recurso que determinam características do sistema.

H2: As *features* podem ser extraídas a partir de elementos internos (tarefas e recursos) do ator que representa o sistema e das dependências (particularmente do *dependum*) ligadas diretamente a esse ator.

Analisando o modelo da Figura 6 percebemos que não há dependência do tipo recurso ou tarefa, mas há várias tarefas como elementos internos do ator que representa a TaRGeT e todas elas devem ser destacadas como possíveis *features*. A Figura 8 mostra o modelo SR da TaRGeT após a aplicação dessas heurísticas.

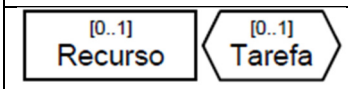
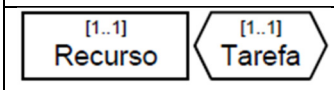
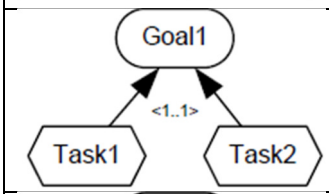
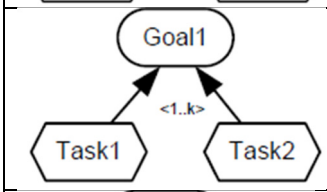
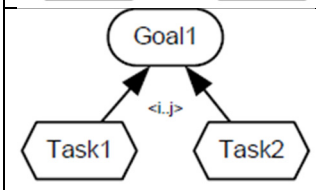
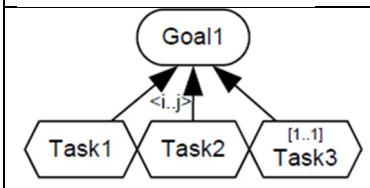
Figura 8 Modelo SR do TaRGeT com os candidatos a *features* destacados [Souza 2012]



Depois de destacar no modelo SR os elementos que possivelmente irão gerar *features*, o próximo passo (**reengenharia do modelo SR**) é reestruturar o modelo para inserir cardinalidade e, dessa forma, poder capturar a variabilidade da LPS que não pode ser

expressa apenas com o modelo *i** original de Yu (1995). Para isto, vamos utilizar conceitos de cardinalidade da linguagem *i*-c* (*i** com cardinalidade) [Silva, Borba e Castro 2010]. O Quadro 3 apresenta os tipos de cardinalidade definidos por Silva, Borba e Castro (2010).

Quadro 3 Cardinalidade para modelos i*, adaptado de Silva, Borba e Castro (2010)

Notação	Conceito nos modelos de *features*	Em um produto específico
	*Feature* opcional	Pode estar presente ou não
	*Feature obrigatória*	Deve estar presente exatamente uma vez
	*Feature* agrupada alternativa *xor* (ou-exclusivo)	Aparece exatamente 1 alternativa do agrupamento de *features*
	*Feature* agrupada alternativa *or* (ou-inclusivo)	Aparece pelo menos 1 e no máximo k alternativa do agrupamento de *features*
	*Feature* agrupada com cardinalidade	Pode aparecer entre i e j alternativas do agrupamento de *features*
	*Feature* agrupada com cardinalidade	Pode aparecer entre i e j alternativas do agrupamento de *features*, sendo "Task3" uma alternativa obrigatória

Assim como na etapa anterior, utilizamos algumas heurísticas propostas por Borba (2009) e outras definidas por Souza (2012) para realizar essa atividade. As heurísticas são apresentadas a seguir.

H3: Todos os elementos do tipo tarefa e recurso do modelo SR que foram destacados como possíveis *features* serão modificados para conter cardinalidade. Os demais elementos do modelo não sofrerão modificações.

H4: Se a quantidade de sub-elementos de uma ligação meio-fim que está destacada como possível *feature*, for maior do que 1, então esses sub-elementos serão agrupados e haverá cardinalidade no relacionamento. Contudo, a cardinalidade pode estar presente nos sub-elementos se alguns deles forem obrigatórios e outros forem opcionais. Se a quantidade

de sub-elementos for exatamente 1, então a cardinalidade pertencerá ao sub-elemento e não ao relacionamento.

H5: Os sub-elementos de uma ligação de decomposição de tarefa que estão destacados como possíveis *features*, serão modelados como *features* obrigatórias, significando que a cardinalidade pertencerá aos sub-elementos, que terão cardinalidade [1..1].

H6: As dependências (*dependum*), que estão destacadas como possíveis *features*, terão a cardinalidade no próprio elemento.

H7: Os elementos destacados como possíveis *features*, que não sejam sub-elementos de nenhum outro elemento que também seja *feature* e não esteja agrupado, terão a cardinalidade pertencentes a eles próprios. As *features* obtidas através deles estarão ligadas diretamente a *feature* raiz no modelo de *features*.

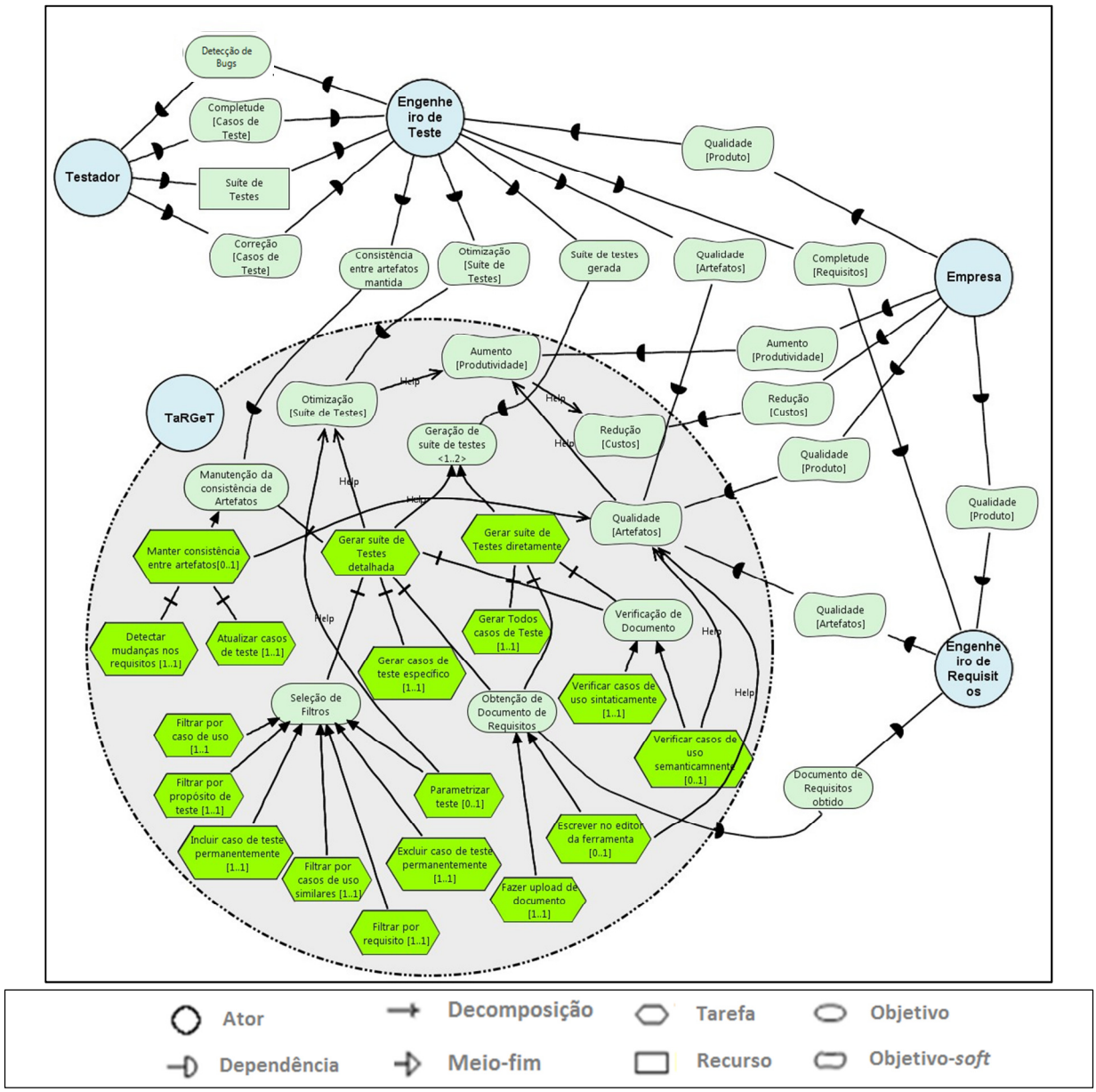
Como afirma Souza (2012), a cardinalidade dos elementos pertencentes a outros tipos de relacionamento e a cardinalidade das ligações meio-fim devem ser decididas pelo engenheiro de domínio de acordo com as informações e análises que ele possui sobre a LPS em questão.

Aplicando as heurísticas ao TaRGeT, através de **H3** descobre-se que todas as tarefas destacadas conterão cardinalidade. Por **H4**, tem-se que a cardinalidade de **Manter Consistência entre Artefatos** deve ficar no próprio elemento e que **Gerar Suíte de Testes Detalhada** e **Gerar Suíte de Testes Diretamente** serão agrupadas e a cardinalidade pertencerá ao relacionamento. A cardinalidade do grupo formado por **Gerar Suíte de Testes Detalhada** e **Gerar Suíte de Testes Diretamente** é <1..2>, pois deve existir ao menos um tipo de geração de suíte de teste, mas pode ser qualquer uma das duas opções ou até mesmo ambas. De acordo com a heurística **H4**, as tarefas relacionadas por ligação meio-fim aos objetivos **Seleção de Filtros**, **Obtenção de Documento de Requisitos** e **Verificação de Documento** poderiam ser agrupadas e ter a cardinalidade posta no relacionamento. Porém, nesses três agrupamentos existem tarefas que são obrigatórias e outras que são opcionais. Por isso, também conforme a heurística **H4**, optou-se por colocar a cardinalidade nas tarefas, ao invés de colocá-las no relacionamento. As tarefas **Manter Consistência entre Artefatos**, **Parametrizar Teste**, **Escrever no Editor da Ferramenta** e **Verificar Casos de Uso Semanticamente** caem no caso em que a cardinalidade deve estar no elemento. Todas estas tarefas têm cardinalidade [0..1] porque são opcionais, ou seja, pode haver produtos

TaRGeT sem essas funcionalidades. Já as demais tarefas que são sub-elementos em ligações meio-fim têm cardinalidade [1..1] porque são obrigatórias.

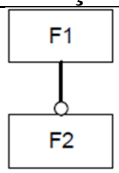
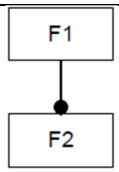
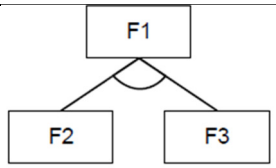
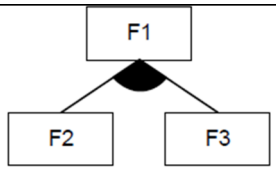
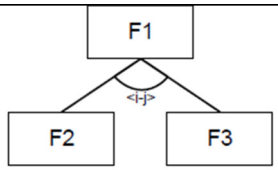
A partir de **H5**, tem-se que a cardinalidade das tarefas **Detectar Mudanças nos Requisitos**, **Atualizar Casos de Teste**, **Gerar Casos de Teste Específicos** e **Gerar Todos os Casos de Teste** será [1..1], pois são sub-elementos de decomposições de tarefas. Como não foram destacadas dependências, **H6** não é aplicada. A Figura 9 mostra o modelo SR com cardinalidade obtido após a aplicação das heurísticas desta atividade.

Figura 9 Modelo SR com cardinalidade do TaRGeT [Souza 2012]



Na atividade de **elaboração do modelo de *feature***, as heurísticas foram mantidas como definidas no processo de Borba (2009). Seu propósito é elaborar o modelo de *feature* de uma LPS, a partir do modelo SR com cardinalidade, usando algumas heurísticas e um quadro de mapeamento (Quadro 4) entre a cardinalidade dos modelos *i** e dos modelos de *features*.

Quadro 4 Mapeamento entre a cardinalidade nos modelos *i e nos modelos de *features*, extraída de Borba (2009)**

Cardinalidade no modelo i*		**Cardinalidade no modelo de *features***	
Tipo	**Valor**	**Notação**	**Explicação**
Elemento	[0..1]		Uma *feature* opcional pode ou não estar presente em um produto concreto. A cardinalidade é [0..1]
Elemento	[1..1]		Uma *feature* obrigatória deve estar presente exatamente uma vez em um produto concreto. A cardinalidade é [1..1]
Grupo	<1..1>		Uma cardinalidade de grupo <1..1> indica que exatamente uma *feature* pode ser selecionada do grupo
Grupo	<1..k>		Uma cardinalidade de grupo <1..k> indica que pelo menos uma e no máximo k *features* podem ser selecionadas do grupo e que k é o número total de *features* no grupo
Grupo	<i..j>		Uma cardinalidade de grupo é um intervalo inserido em um grupo de *features* que indica quantas *features* podem ser selecionadas do grupo em um produto concreto

As heurísticas definidas por Borba (2009) para elaboração de modelo de *feature* a partir do modelo SR são as seguintes:

H8: A *feature* raiz (em inglês, *root feature*) de um modelo de *feature* será o sistema que está sendo modelado.

H9: Os nomes das *features* podem ser extraídos através das propriedades que descrevem os elementos intencionais. Se o elemento intencional for um recurso, o nome da *feature* será o mesmo nome do recurso. Se o elemento intencional for uma tarefa, o nome da *feature* poderá ser uma simplificação do nome da tarefa, caso seja necessário. Para facilitar a construção do modelo de *feature*, aconselha-se criar uma tabela com informações

sobre a *feature*, o elemento que a originou e a cardinalidade. O Quadro 5 pode ser utilizado como um *template* da tabela a ser preenchida servindo para manter o rastreamento entre as *features* e as tarefas ou recursos que as originaram.

H10: Todas as linhas da tabela serão mapeadas para o modelo de *feature* com o nome preenchido na coluna **Feature**. Quando duas ou mais *features* tiverem o mesmo nome, deve-se analisar se possuem mesma semântica. Em caso positivo, elas deverão se tornar apenas uma *feature*. Do contrário, deve-se adaptar seus nomes para que reflitam a funcionalidade ou atributo desejado.

H11: Todas as *features* da tabela serão relacionadas à *feature* raiz quando não tiverem o campo **Elemento Pai** preenchido.

H12: Todas as *features* da tabela serão sub-*features* quando tiverem a coluna **Elemento Pai** preenchida com o mesmo nome. Essas sub-*features* serão relacionadas à *feature* correspondente ao valor do campo **Elemento Pai**.

De acordo com as heurísticas definidas nesta atividade, *features* opcionais são obtidas de elementos com cardinalidade [0..1], enquanto *features* obrigatórias são obtidas de elementos com cardinalidade [1..1]. Elementos envolvidos em relacionamento meio-fim com cardinalidade tornam-se *features* alternativas com a cardinalidade equivalente.

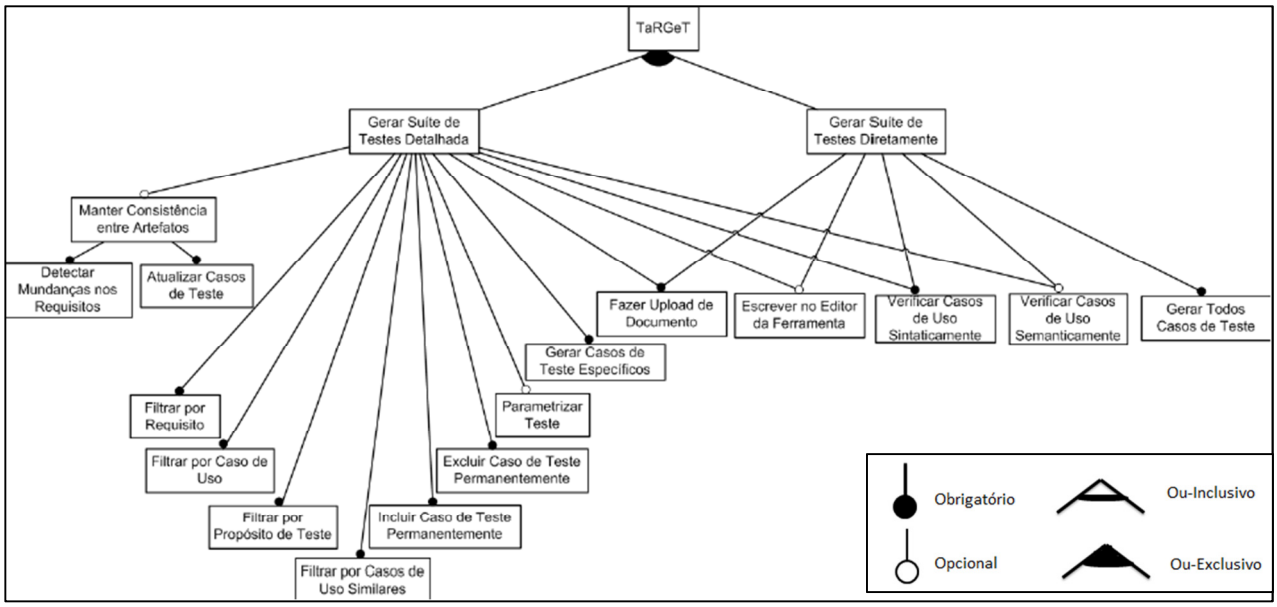
Com base no modelo SR com cardinalidade, apresentado na Figura 9, utilizando-se das cinco heurísticas desta atividade, **H8** a **H12**, e levando-se em consideração o mapeamento entre a cardinalidade de modelos *i** e de modelos de *features* (Quadro 4), elaboramos o modelo de *feature* da LPS TaRGeT. Por **H8**, tem-se que a *feature* raiz será **TaRGeT** e com **H9** constrói-se o Quadro 5.

Quadro 5 Rastreamento entre tarefas/recursos e as features de TaRGeT

Elemento	**Cardinalidade**		**Elemento Pai**	***Feature* Relacionada**
	Tipo	**Valor**		
Gerar Suíte de Testes Detalhada	Grupo	<1..2>	-	Gerar Suíte de Testes Detalhada
Gerar Suíte de Testes Diretamente	Grupo	<1..2>	-	Gerar Suíte de Testes Diretamente
Manter Consistência entre Artefatos	Elemento	[0..1]	Gerar Suíte de Testes Detalhada	Manter Consistência entre Artefatos
Detectar Mudanças nos Requisitos	Elemento	[1..1]	Manter Consistência entre os Artefatos	Detectar Mudanças nos Requisitos
Atualizar Casos de Teste	Elemento	[1..1]	Manter Consistência entre os Artefatos	Atualizar Casos de Teste
Filtrar por Requisito	Elemento	[1..1]	Gerar Suíte de Testes Detalhada	Filtrar por Requisito
Filtrar por Caso de Uso	Elemento	[1..1]	Gerar Suíte de Testes Detalhada	Filtrar por Caso de Uso
Filtrar por Propósito de Teste	Elemento	[1..1]	Gerar Suíte de Testes Detalhada	Filtrar por Propósito de Teste
Filtrar por Casos de Uso Similares	Elemento	[1..1]	Gerar Suíte de Testes Detalhada	Filtrar por Casos de Uso Similares
Incluir Caso de Teste Permanentemente	Elemento	[1..1]	Gerar Suíte de Testes Detalhada	Incluir Caso de Teste Permanentemente
Excluir Caso de Teste Permanentemente	Elemento	[1..1]	Gerar Suíte de Testes Detalhada	Excluir Caso de Teste Permanentemente
Parametrizar Teste	Elemento	[1..1]	Gerar Suíte de Testes Detalhada	Parametrizar Teste
Gerar Casos de Teste Específicos	Elemento	[1..1]	Gerar Suíte de Testes Detalhada	Gerar Casos de Teste Específicos
Fazer Upload de Documento	Elemento	[1..1]	Gerar Suíte de Testes Detalhada; Gerar Suíte de Testes Diretamente	Fazer *Upload* de Documento
Escrever no Editor da Ferramenta	Elemento	[0..1]	Gerar Suíte de Testes Detalhada; Gerar Suíte de Testes Diretamente	Escrever no Editor da Ferramenta
Verificar Casos de Uso Sintaticamente	Elemento	[1..1]	Gerar Suíte de Testes Detalhada; Gerar Suíte de Testes Diretamente	Verificar Casos de Uso Sintaticamente
Verificar Casos de Uso Semanticamente	Elemento	[0..1]	Gerar Suíte de Testes Detalhada; Gerar Suíte de Testes Diretamente	Verificar Casos de Uso Semanticamente
Gerar Todos os Casos de Teste	Elemento	[1..1]	Gerar Suíte de Testes Diretamente	Gerar Todos os Casos de Teste

O Quadro 5 deve ser analisado através das heurísticas **H10**, **H11** e **H12**, para produzir o modelo de *feature* contido na Figura 10. Como é possível perceber, na Figura 10, o modelo de *feature* gerado pode conter algumas inconsistências, como, por exemplo, *features* com mais de uma *feature* pai. Esta e outras inconsistências serão resolvidas na próxima atividade.

Figura 10 Modelo de *feature* gerado para TaRGeT [Souza 2012]



O objetivo da atividade de **refinamento do modelo de *feature*** é reorganizar e refinar o modelo de *feature* obtido anteriormente. Esta atividade pode ser executada várias vezes, até que o engenheiro de domínio esteja satisfeito com o modelo de *feature*. É considerada opcional, pois só é executada caso o engenheiro de domínio identifique algum problema ou incoerência no modelo gerado e/ou exista a necessidade de incorporar *features* que não são capturadas pelo modelo SR. Deve-se averiguar a necessidade de modelar *features* referentes a funcionalidades ou restrições que geralmente não são capturadas em modelos *i**. Como *features* referentes a restrições de ambiente, tecnologias, idiomas, sistemas monetários e de medidas. Também deve-se avaliar se alguma *feature* refere-se a uma funcionalidade complexa que pode ser decomposta. É aconselhável haver novas interações com os interessados nesta etapa, para completar o modelo de *features* obtido do modelo SR e verificar se ele reflete as suas reais necessidades. Para Borba (2009), as situações em que uma reorganização do modelo de *features* é necessária são: sub-*feature* com mais de um pai, *features* repetidas, *features* colocadas em lugar inapropriado, *features* com nomes diferentes, mas com mesma semântica, *features* que podem ser agrupadas por fazerem referência ao mesmo conceito. Não foram definidas heurísticas para identificar essas situações ou sobre o que fazer quando

elas surgem, mas em Silva, Borba e Castro (2011) foram dadas algumas sugestões do que pode ser feito:

(i) se uma sub-*feature* estiver relacionada a mais de uma *feature*, provavelmente esta sub-*feature* será realocada;

(ii) se existirem *features* repetidas no modelo, as repetições serão eliminadas;

(iii) se existirem diferentes *features* com mesma semântica, essas *features* serão representadas por uma só.

Aconselha-se que o engenheiro de domínio, após refinar o modelo de *feature*, compare o modelo anterior com o refinado para verificar se há produtos que poderiam ser configurados anteriormente, mas não são possíveis com o modelo refinado. Caso isto aconteça, cabe ao engenheiro avaliar se esses produtos devem fazer parte do escopo da LPS. Se sim, deve-se refatorar o modelo para que seja possível configurar todos os produtos pretendidos.

Também é aconselhável consultar os interessados para tentar descobrir *features* que não estão presentes no modelo SR, pois nem sempre todas as funcionalidades e atributos de um sistema são capturados pelo modelo de objetivos. Isto pode ocorrer porque não são atributos diretamente ligados aos objetivos dos interessados ou porque os objetivos que dariam origem a essas *features* não foram elicitados e modelados anteriormente. Assim, o modelo de *feature* gerado na atividade anterior pode estar incompleto.

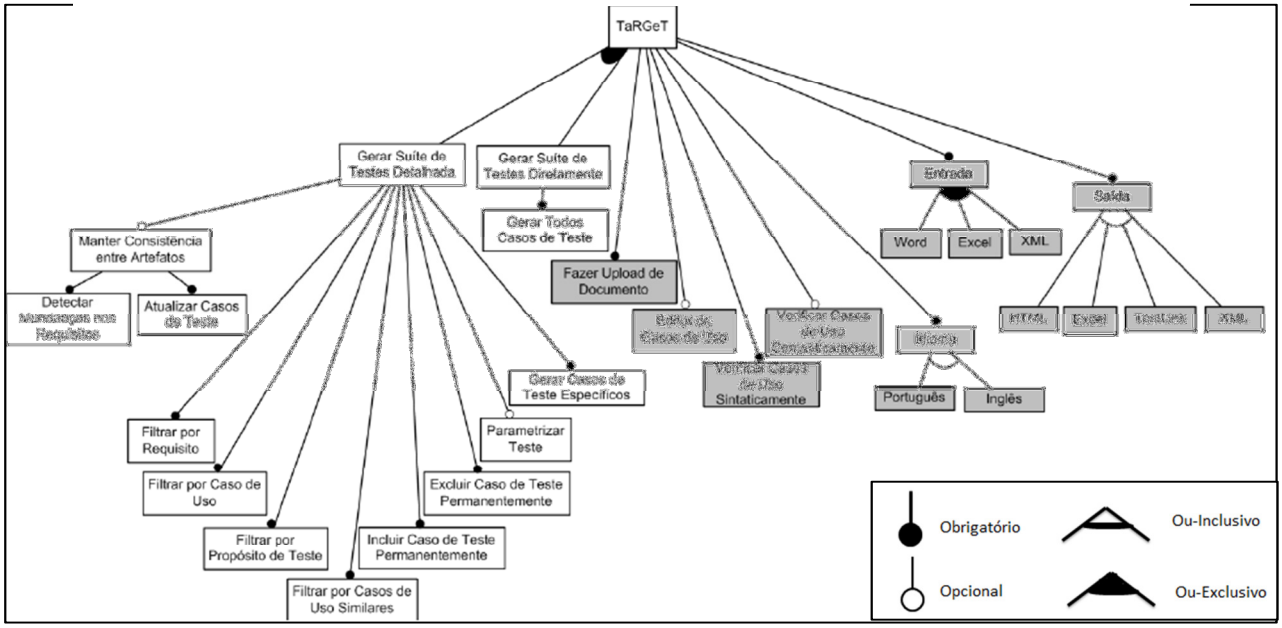
Para o exemplo em questão, TaRGeT, verificou-se que algumas *features* possuíam mais de uma *feature* pai. Para resolver este problema, optou-se por ligá-las ao elemento de nível acima das *features* pai, uma vez que as *features* não representam características específicas de uma determinada *feature*, mas funcionalidades do sistema como um todo. Também optou-se por mudar o nome da *feature* **Escrever no Editor da Ferramenta** para **Editor de Casos de Uso**, para deixar mais claro que a funcionalidade do sistema é o próprio editor, enquanto escrever é a ação do usuário.

Outras modificações foram feitas com base em requisitos levantados durante as entrevistas realizadas por Souza (2012), mas que não estão diretamente ligados aos objetivos dos atores que foram capturados no modelo SD e, portanto, não estão presentes no modelo SR. São *features* referentes a tipos de artefatos de entrada e saída e a idiomas da ferramenta.

É possível que os objetivos que geraram as *features* fossem identificados no modelo SR se houvesse um detalhamento na etapa de elaboração do modelo SD. De acordo com Souza (2012), os modelos SR do *i** tendem a ficar bastante complexos à medida que vários objetivos são adicionados, alguns analistas preferem concentrar-se aos objetivos principais ou mais

relevantes. Portanto, para evitar o aumento de sua complexidade, optou-se por não acrescentar os objetivos relativos às *features* descobertas no modelo SR. A Figura 11 apresenta o modelo de *feature* refinado cujos elementos destacados correspondem às alterações feitas em relação ao modelo anterior.

Figura 11 Modelo de *feature* da TaRGeT refinado [Souza 2012]



Na Figura 11, as *features* obrigatórias são: **Entrada**, **Saída**, **Idioma**, **Verificar Casos de Uso Sintaticamente**, **Fazer Upload de Documento**. Temos como *features* opcionais **Editor de Casos de Uso**, **Verificar Casos de Uso Semanticamente**. Pode-se selecionar apenas uma *feature* (**Português** ou **Inglês**) quando **Idioma** é selecionado. Assim, como **Saída** é selecionado **HTML**, **Excel**, **TestLink** ou **XML**. Podemos selecionar no mínimo uma e no máximo duas opções para gerar suíte de testes (**Gerar Suíte de Testes Detalhadamente** e **Gerar Suíte de Testes Diretamente**). Se a escolha foi **Gerar Suíte de Testes Diretamente**, torna-se obrigatório selecionar **Gerar Todos os Casos de Teste**. Caso a escolha seja a *feature* **Gerar Suíte de Testes Detalhadamente**, torna-se obrigatório selecionar as *features* **Filtrar por Requisito, Filtrar por Caso de Uso**, **Filtrar por Propósito de Teste**, **Filtrar por Casos de Uso Similares**, **Incluir Caso de Teste Permanentemente**, **Excluir Caso de Teste Permanentemente**, **Gerar Casos de Teste Específicos**. Temos ainda, como *features* opcionais **Parametrizar Teste** e **Manter Consistência entre Artefatos**. Caso a *feature* **Parametrizar Teste** seja selecionada, torna-se obrigatório selecionar **Detectar Mudanças nos Requisitos** e **Atualizar Casos de Teste**. Quando a *feature* **Entrada** é selecionada, podemos escolher no mínimo uma e no máximo três opções dentre **Word**,

Excel e **XML**.

Para **elaborar cenários de caso de uso**, a abordagem utiliza o modelo de *feature* e o modelo SR com cardinalidade e gera um documento contendo a descrição dos casos de uso utilizando a abordagem PLUSS [Eriksson, Börstler e Borg 2005]. Esta atividade foi realizada com base nas descrições disponíveis na documentação do TaRGeT[3] e, em informações coletadas de entrevistas realizadas por Souza (2012) com membros do projeto.

As diretrizes da atividade de elaboração dos cenários de caso de uso da abordagem GS2SPL, definidas por Souza (2012) são:

Passo 1 - Descoberta de Atores

Diretriz 1: Todo ator *i** é candidato a ser mapeado para um ator de caso de uso.

Diretriz 2: O ator *i** candidato deve ser externo ao sistema de software pretendido, caso contrário, ele não pode ser mapeado para um ator de caso de uso.

Diretriz 3: O ator *i** candidato deve ter ao menos uma dependência (tarefa, objetivo, recurso e objetivo-*soft*) com o ator do sistema de software, caso contrário, ele não pode ser mapeado para um ator de caso de uso.

Diretriz 4: Após a análise das dependências, se algum ator tiver apenas dependências de objetivo-*soft* com o ator da LPS pretendida, ele não deve ser mapeado para um ator de caso de uso. Contudo, os requisitos não funcionais derivados dessas dependências permanecem.

Por exemplo, analisando a Figura 6 percebemos que o ator **Empresa** não pode ser mapeado para um ator de caso de uso, pois possui apenas dependências do tipo objetivo-*soft* com o ator **TaRGeT**. Mas, os respectivos objetivos-*soft* que representam requisitos não funcionais continuam sendo importantes para a criação dos produtos da linha.

Diretriz 5: Atores *i**, que são relacionados através do mecanismo ISA[4] (é um) nos modelos de objetivos e mapeados individualmente para atores em casos de uso (aplicando as diretrizes 1, 2 e 3), serão relacionados no diagrama de casos de uso através do relacionamento do tipo <<*generalização*>>.

---

[3] http://twiki.cin.ufpe.br/twiki/bin/view/ProjetoProcad/TaRGeT

[4] ISA é um tipo de associação, presente nos modelos *i**, entre atores e representa uma generalização, com um ator sendo um caso especializado de outro ator.

Passo 2 - Descoberta de Casos de Uso para os Atores

Diretriz 6: Para cada ator de caso de uso descoberto no passo 1, deve-se analisar todas as suas dependências (*dependum*) com o ator que representa a LPS, buscando descobrir casos de uso para o ator analisado.

Diretriz 6.1: Dependências de objetivo – objetivos em *i** são mapeados para casos de uso;

Diretriz 6.2: Dependências de tarefa – neste caso, deve-se investigar se a tarefa precisa ser decomposta em subtarefas. Se a execução da tarefa requer vários passos (posteriormente mapeados em passos de caso de uso), ela pode ser mapeada em caso de uso. Do contrário, esta dependência representa um passo do caso de uso relacionado ao elemento interno ao qual a dependência está ligada;

Diretriz 6.3: Dependências de recurso – se um ator depende de outro para obter um recurso, deve-se descobrir se são necessárias várias interações para a obtenção desse recurso. Se for preciso várias interações do ator com o sistema, esta dependência pode ser mapeada em caso de uso. Do contrário, esta dependência representa um passo do caso de uso relacionado ao elemento interno ao qual a dependência está ligada;

Diretriz 6.4: Dependências de objetivos-*soft* – tipicamente, uma dependência de objetivos-soft em *i** representa um requisito não funcional para o sistema. Consequentemente, estas dependências não representam casos de uso do sistema, mas requisitos não funcionais associados a casos de uso específicos ou ao sistema como um todo. O requisito não funcional será associado ao mesmo caso de uso que contém o elemento interno ligado à dependência de objetivo-*soft*.

Passo 3 - Descoberta e Descrição de Cenários dos Casos de Uso

Diretriz 7: Após descobrir quais dependências originam casos de uso, deve-se preencher o campo "*Feature*" da descrição dos casos de uso. Além disso, os passos descobertos com a aplicação das próximas diretrizes devem ser anotados com as *features* relacionadas.

Diretriz 7.1: Se a dependência for de tarefa ou recurso, a coluna é preenchida com o nome da *feature* que ela originou;

Diretriz 7.2: Se a dependência for de objetivo, é necessário verificar a que elemento ela está relacionada dentro da fronteira do ator do sistema. Se este elemento for uma tarefa ou recurso, preenche-se com o nome da *feature* que ele originou. Se o elemento for um objetivo, analisa-se a(s) tarefa(s) nas quais ele foi refinado;

Diretriz 7.3: Os passos derivados de elementos que também originaram *features* devem ser anotados com as mesmas, colocando o nome da *feature* entre colchetes [ ] na descrição do passo;

Diretriz 8: Analisar os subcomponentes em uma ligação de decomposição de tarefa em um possível mapeamento para passos na descrição do cenário primário de casos de uso.

Diretriz 8.1: Se esta tarefa que foi decomposta satisfaz alguma dependência que foi previamente mapeada em um caso de uso, seus subcomponentes são mapeados como passos obrigatórios do cenário primário desse caso de uso;

Diretriz 8.2: Cada passo derivado de uma decomposição de tarefa será identificado por um número único e sem parênteses, de acordo com a notação de passos obrigatórios de PLUSS;

Diretriz 8.3: Se um objetivo-*soft* for um sub-elemento de uma decomposição de tarefa, ele pode ser associado como requisito não funcional ao caso de uso em questão.

Diretriz 9: Se houver apenas um sub-elemento na ligação meio-fim, analisa-se a cardinalidade dele.

Diretriz 9.1: Se a cardinalidade for [1..1], ele será mapeado para um passo obrigatório do cenário. Logo, este passo será identificado por um número único e sem parênteses;

Diretriz 9.2: Se a cardinalidade for [0..1], ele será mapeado para um passo opcional do cenário. Logo, este passo será identificado por um número único, porém entre parênteses, conforme a notação PLUSS para passos opcionais;

Diretriz 10: Se existir mais de um sub-elemento na ligação meio-fim, olha-se a cardinalidade de grupo.

Diretriz 10.1: Se a cardinalidade for <1..1>, os sub-elementos serão mapeados em passos alternativos dos quais apenas um pode ser escolhido para um passo obrigatório. Portanto, serão identificados por números iguais e sem parênteses;

Diretriz 10.2: Se a cardinalidade for <i..j> e $i = 0$ e $j = 1$, os sub-elementos serão mapeados em passos alternativos dos quais apenas um deve ser escolhido para um passo opcional. Portanto, serão identificados por números iguais, mas entre parênteses;

Diretriz 10.3: Se a cardinalidade for <i..j> e $i = 0$ e $j \neq 1$, os sub-elementos serão mapeados em passos alternativos dos quais no máximo j devem ser escolhidos para um passo opcional. Portanto, serão identificados por números iguais, mas entre parênteses, seguidos de uma letra minúscula, sendo cada alternativa com uma letra diferente e a primeira

começa pela letra *a*;

Diretriz 10.4: Se a cardinalidade for *<i..j>* e $i \neq 0$ e $j > 1$, incluindo o caso específico da cardinalidade *<1..k>*, os sub-elementos serão mapeados em passos alternativos dos quais no mínimo *i* e no máximo *j* devem ser escolhidos para um passo obrigatório. Portanto, serão identificados por números iguais seguidos de uma letra minúscula, sendo cada alternativa com uma letra diferente e a primeira começa pela letra *a*, e sem parênteses;

Diretriz 10.5: Se não houver cardinalidade de grupo e ela estiver apenas nos elementos, os sub-elementos serão mapeados em passos alternativos. Eles serão identificados por números iguais seguidos de uma letra minúscula, sendo cada alternativa com uma letra diferente e a primeira começa pela letra *a*, e sem parênteses. Entretanto, aqueles que tiverem cardinalidade [0..1] terão a numeração entre parênteses, o que denota sua opcionalidade.

Diretriz 11: Deve-se analisar o modelo SR à procura de informações adicionais e relevantes para os casos de uso

Diretriz 11.1: Analisar ligações de contribuição para objetivo-*soft*. Caso o elemento do qual partiu a contribuição tenha sido mapeado em um caso de uso, o objetivo-*soft* pode ser associado a este caso de uso como requisito não funcional;

Diretriz 11.2: analisar os elementos que fazem parte de cada caso de uso e suas ligações com demais elementos ou dependências. Estas ligações podem originar pré-condições para os casos de uso descobertos, especialmente as ligações do tipo decomposição de tarefa, que indicam que o sub-elemento é necessário para a execução da tarefa.

Diretriz 12: Investigar a possibilidade de derivar novos casos de uso a partir da observação dos passos dos cenários daqueles casos de uso que são complexos. Cada passo do caso de uso deve ser analisado para verificar a possibilidade do mesmo gerar um novo caso de uso.

Diretriz 12.1: Um novo caso de uso será gerado a partir do passo analisado, se for possível definir os passos que representam a necessidade de interações adicionais entre o ator e o sistema para se atingir o objetivo desejado;

Diretriz 12.2: Se o novo caso de uso foi derivado de um passo obrigatório, o mesmo deve estar relacionado ao caso de uso do qual ele fazia parte através do mecanismo *<<include>>*;

Diretriz 12.3: Se o novo caso de uso foi derivado de um passo opcional, o mesmo deve estar relacionado ao caso de uso do qual ele fazia parte através do mecanismo <<*extend*>>.

Finalizada a atividade de **elaboração dos cenários de casos de** uso, a próxima atividade é o **refinamento dos cenários de caso de uso**. Esta atividade é necessária, pois o artefato gerado na etapa anterior é incompleto, faltando algumas respostas do sistema e o cenário secundário (excepcional e alternativo). A Figura 12 apresenta o cenário de caso de uso completo, gerado ao final dessa atividade.

Figura 12 Cenário de caso de uso Obter Documento de Requisitos do TaRGeT especificado com a técnica PLUSS

UC 04 Obter Documento de Requisitos

ID	Ação do Usuário	Resposta do Sistema
1 (a)	*Upload* do documento [Documento de Requisito]	Solicita que o usuário indique o caminho do documento de requisitos
2 (b)	Seleciona a opção de abrir o editor de caso de uso da ferramenta [Editor da Ferramenta]	Abre o editor de caso de uso
2 (a)	Fornece localização do arquivo	Arquivo é importado para a ferramenta
2 (b)	Escreve caso de uso no editor	-
3 (a)	-	Documento é carregado
3 (b)	Seleciona a opção para salvar o documento	Documento é salvo

Por fim, **Configuração do produto**, é executada sempre que um novo produto da LPS tem de ser derivado. Ao final do processo, as *features* mais relevantes e os casos de uso dos objetivos dos interessados são obtidos de maneira sistemática. A seguir as três subatividades de configuração do produto são apresentadas.

A primeira atividade, **escolha de configuração específica**, tem o objetivo de permitir ao interessado escolher, de acordo com as suas necessidades, as funcionalidades que farão parte do produto. O interessado utiliza o modelo SR, simplificando o processo de escolha, pois ele seleciona os objetivos que deseja satisfazer, ao invés de analisar diretamente o modelo de *feature*. Assim como em algumas atividades anteriores, utilizamos as heurísticas definidas por Borba (2009) para guiar esta atividade.

H18: Todos os elementos intencionais que possuírem cardinalidade [1..1] deverão estar presentes no modelo de configuração do produto, desde que seus elementos pais também tenham sido selecionados. Os elementos com cardinalidade [0..1] poderão ou não ser selecionados, ficando a critério da necessidade do interessado.

H19: Os sub-elementos relacionados com as ligações meio-fim deverão estar

presentes no modelo de configuração de acordo com as escolhas e necessidades dos interessados, mas obedecendo aos limites impostos pela cardinalidade do relacionamento (se ela existir): <*1..1*>, <*1..k*> ou <*i..j*>.

Como afirma Souza (2012), devido à cardinalidade presente tanto nos elementos como nas ligações meio-fim, pode existir mais de uma configuração possível para a aplicação desejada, as quais diferem apenas nos níveis de satisfação dos objetivos-*soft*. Portanto, todas as variantes que atendem as necessidades do interessado devem ser configuradas em modelos SR e classificadas de acordo com as prioridades dos objetivos-*soft*. A próxima atividade tem como objetivo realizar essa classificação.

Para ilustrar esta atividade no TaRGeT utilizaremos a configuração de um produto elaborada por Souza (2012), pois não tivemos contato com interessados que pudessem auxiliar na realização desta atividade da abordagem.

Suponhamos que algum cliente requisite uma ferramenta TaRGeT que não possua a opção de gerar suítes de teste diretamente, isto é, só possua a opção de gerar suítes de teste detalhada, pois este cliente deseja que seus testes sejam específicos para cada situação. Ele também deseja que seja possível manter os casos de teste atualizados a cada mudança nos requisitos e não quer um editor embutido na ferramenta. Além disso, a ferramenta deve ser em inglês, deve permitir a geração de casos de teste apenas em XML (do inglês, *eXtended Markup Language*).

Neste caso, seguindo as heurísticas **H18** e **H19**, nota-se que a alternativa **Gerar Suíte de Testes Diretamente**, e todos os seus sub-elementos que não estão ligados a outros elementos que foram selecionados, podem ser removidos do produto segundo a requisição do cliente, pois não ferirá a cardinalidade do grupo em que ela está inserida. Tem-se também que a *feature* opcional **Manter Consistência entre Artefatos** estará presente no produto, bem como todos os elementos com cardinalidade [1..1]. A alternativa **Escrever no Editor da Ferramenta** e sua *feature* relacionada não estarão presentes. As *features* relacionadas aos artefatos de entrada e saída em XML e ao idioma inglês, embora não estejam presentes no modelo SR, devem ser selecionadas no modelo de *feature*, que será configurado na terceira atividade. Ainda assim, há várias configurações possíveis, pois existem outros itens opcionais que o cliente não havia considerado. Para auxiliar a escolha da configuração que o produto deve ter, pode-se analisar as variantes possíveis, dentro daquilo que o cliente já escolheu, com base na prioridade dada por ele aos objetivos-*soft*. No caso deste cliente fictício, ele não especificou se desejava ou não que houvesse a possibilidade de parametrizar os testes ou de verificar

semanticamente os casos de uso. Portanto, há quatro variantes a serem consideradas: (V1) com verificação semântica dos casos de uso e sem parametrização de testes, (V2) com parametrização de testes e sem verificação semântica dos casos de uso, (V3) com as duas opções, e (V4) sem nenhuma das duas opções.

Para o exemplo TaRGeT, as figuras: Figura 13, Figura 14, Figura 15 e Figura 16 representam os modelos SR das variantes.

Figura 13 Modelo SR da Variante V1 do TaRGeT [Souza 2012]

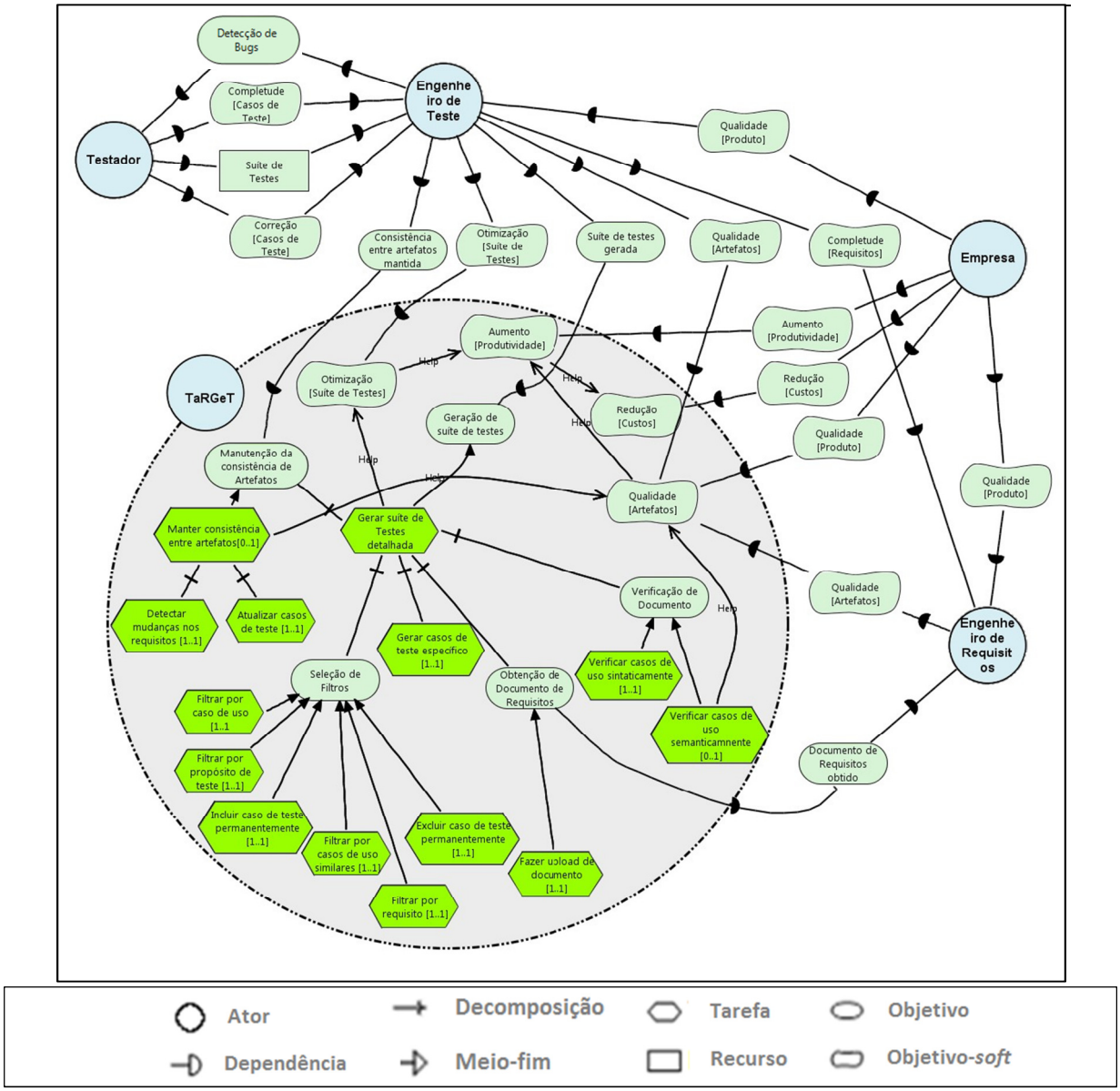


Figura 14 Modelo SR da Variante V2 do TaRGeT [Souza 2012]

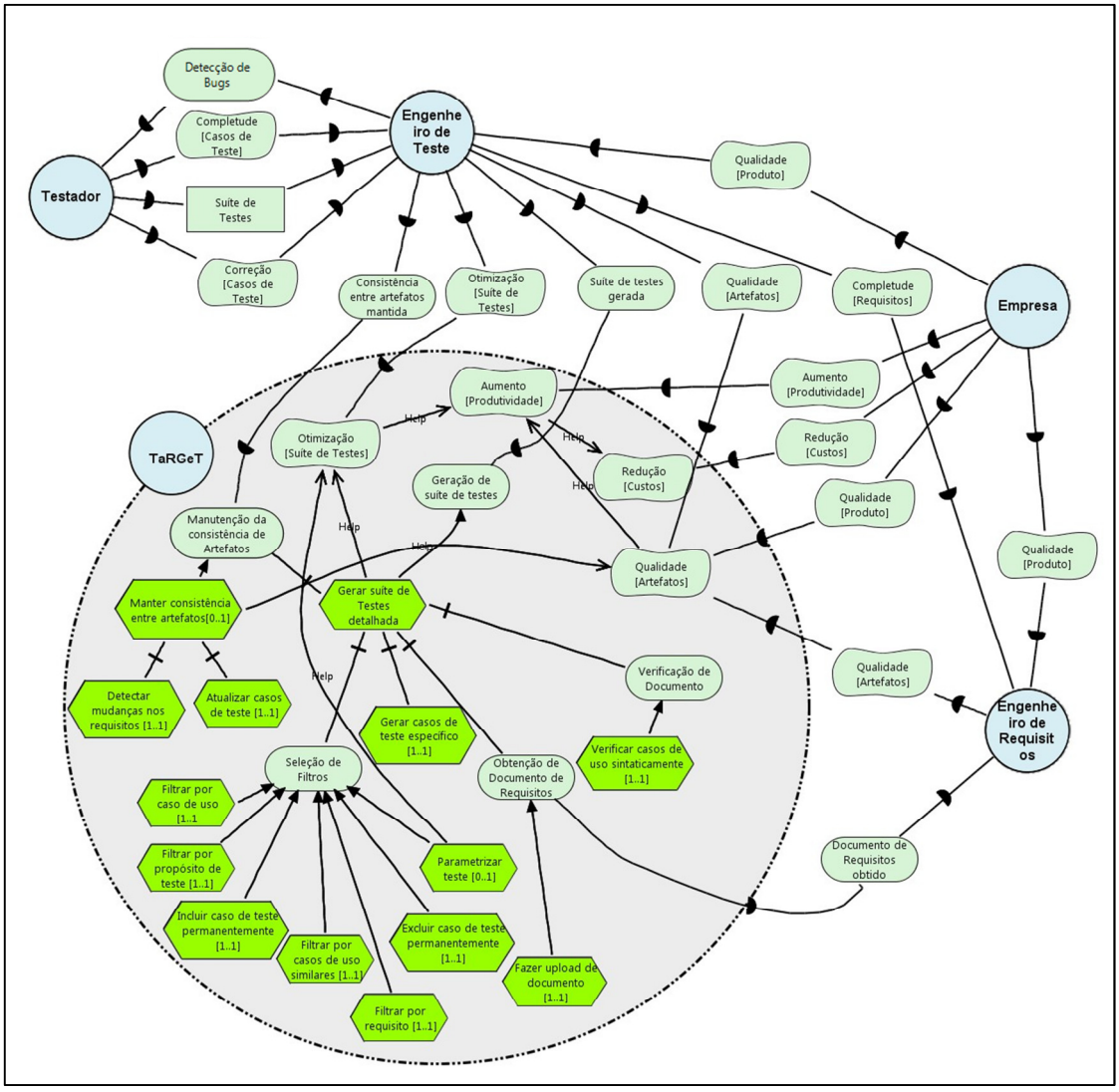


Figura 15 Modelo SR da Variante V3 do TaRGeT [Souza 2012]

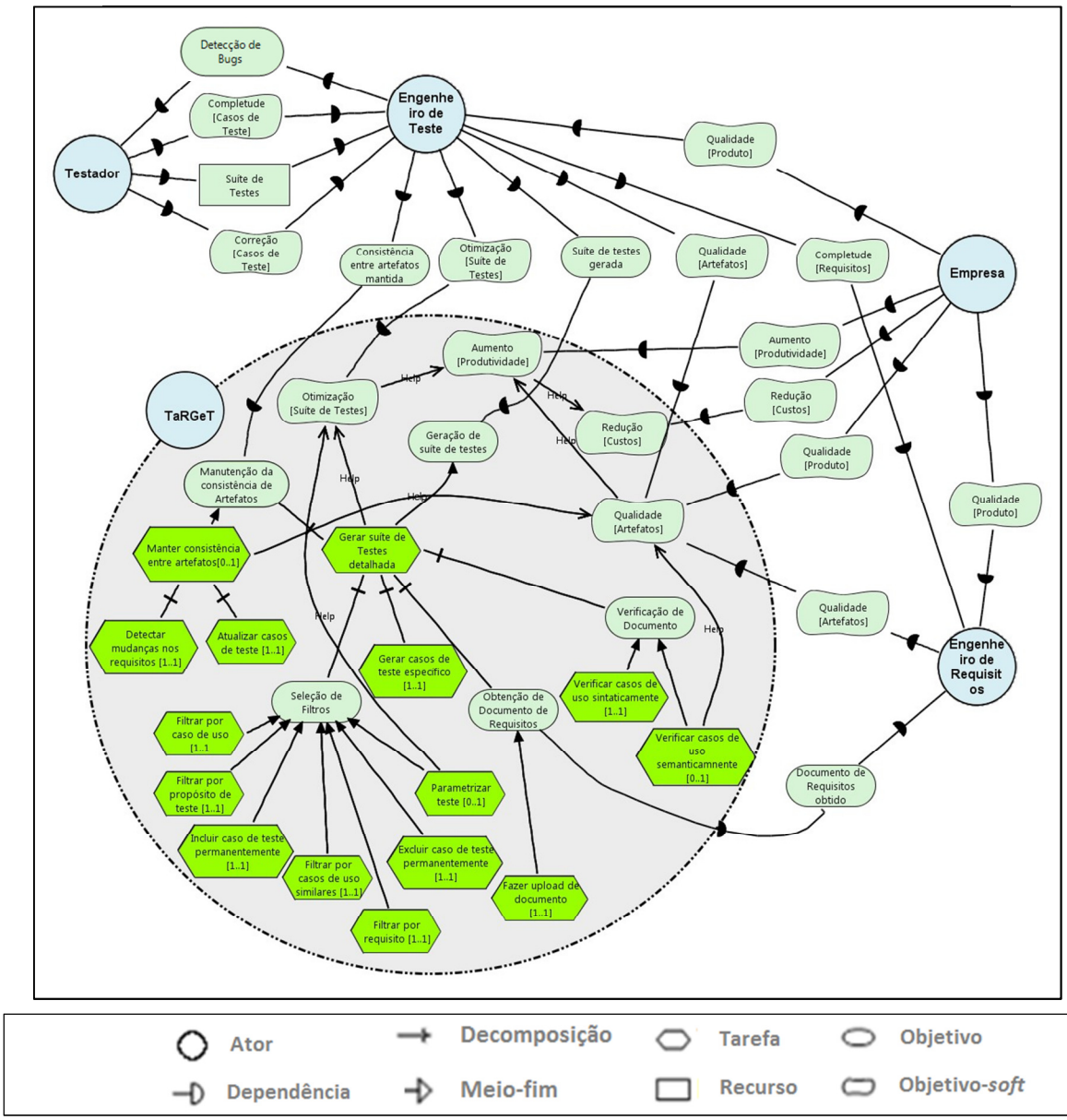
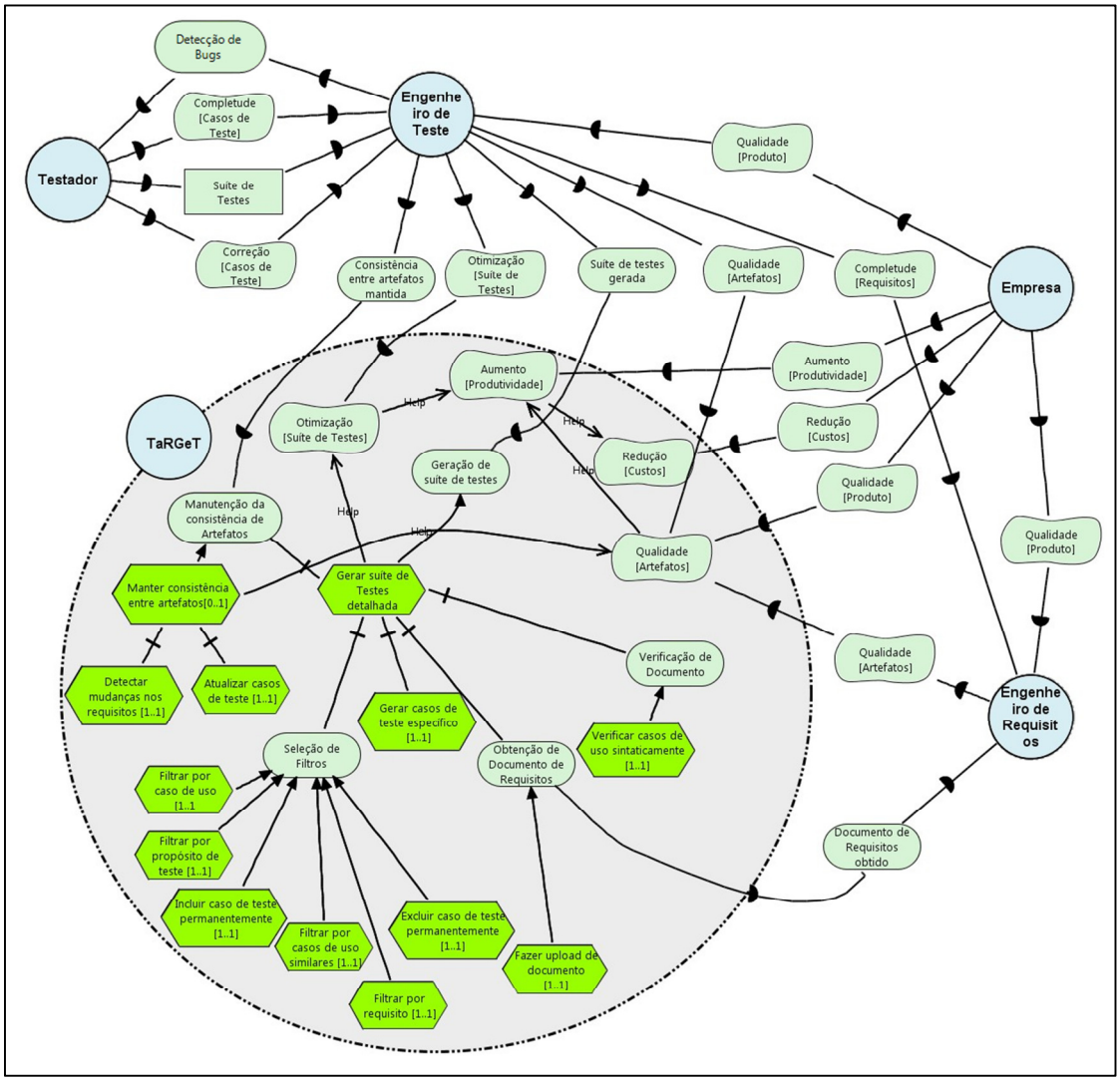


Figura 16 Modelo SR da variante V4 do TaRGeT [Souza 2012]



Segundo Souza (2012) o interessado analisa os objetivos-*soft* presentes no modelo SR e atribui valor de prioridade a cada um, então as variantes serão classificadas. Dessa maneira, a escolha da variante é feita em um nível mais alto de abstração, sem que o interessado precise conhecer as *features* de cada variante. Para a realização da atividade de **priorização da variante** utiliza-se como artefato de entrada os critérios de contribuição (Quadro 6) e os modelos SR das variantes obtidos na atividade anterior. O artefato gerado nesta atividade é o modelo SR da configuração escolhida após a priorização.

Quadro 6 Critérios de Contribuição, extraída de Lima (2011)

Contribuição	**Make**	**Break**	**Help**	**Hurt**	**Some+**	**Some-**	**Unknown**
Valor	1,00		0,75		0,50		0,25

A priorização definida por Lima (2011) é dada pela equação a baixo:

$$priority\,(v) = \sum_{sg \epsilon v} percentPos(v, sg) \times priority(sg) - \sum_{sg \epsilon v} percentNeg(v, sg) \times priority(sg)$$

Sendo,

$$percentPos(v, sg) = \frac{numContPos(sg)}{numContPosTotal} \times tipoDeContPos(sg)$$

$$percentNeg(v, sg) = \frac{numContNeg(sg)}{numContNegTotal} \times tipoDeContNeg(sg)$$

O *numContPosTotal* e o *numContNegTotal* referem-se, respectivamente, ao número total de contribuições positivas e negativas no modelo de objetivos. O *numContPos(sg)* é o número de contribuições positivas para cada objetivo-*soft* e o *tipoDeContPos(sg)* leva em consideração um fator percentual para os tipos de contribuições positivas definidas no Quadro 6. Analogamente, *numContNeg(sg)* é o número de contribuições negativas para cada objetivo-*soft* e o tipoDeContNeg(sg) leva em consideração um fator percentual para os tipos de contribuições negativas definidas no Quadro 6. Além disso, deve-se considerar que o valor de *priority(sg)* deve estar no intervalo de [0, 10], sendo 10 a prioridade máxima.

As variantes são classificadas de acordo com o valor obtido aplicando-se *priority(v)*, quanto maior o valor, melhor a classificação da variante. Como afirma Souza (2012), cabe ao analista a decisão final, mas a priorização facilita a escolha, pois avalia as contribuições das variantes para todos os objetivos-*soft*, levando em consideração a prioridade atribuída a cada um deles.

Antes de aplicar a priorização, é necessário que o cliente defina a prioridade de cada objetivo-*soft*. Para aplicar a priorização ao exemplo, foi definida a prioridade de cada objetivo-*soft* envolvido, conforme mostrado no Quadro 7.

Quadro 7 Prioridade dos objetivos-soft para o TaRGeT

Objetivos-*soft*	**Prioridade [0, 10]**
Otimização [Suíte de Testes] (OST)	10
Aumento [Produtividade] (AP)	9
Redução [Custos] (RC)	9
Qualidade [Artefatos] (QA)	6

O cálculo da priorização das variantes é o seguinte:

$$\begin{aligned} priority\,(V1) &= [(percentPos(V1, OST) \times priority(OST)) \\ &+ (percentPos(V1, AP) \times priority(AP)) \\ &+ (percentPos(V1, QA) \times priority(QA)) \\ &+ (percentPos(V1, RC) \times priority(RC))] \end{aligned}$$

$$\begin{aligned} priority\,(V1) &= [(\frac{1}{6} \times 0,75 \times 10) + (\frac{2}{6} \times 0,75 \times 9) + (\frac{2}{6} \times 0,75 \times 6) + (\frac{1}{6} \times 0,75 \times 9)] \\ &= [1,25 + 2,25 + 1,50 + 1,125] = 6,125 \end{aligned}$$

$$\begin{aligned} priority\,(V2) &= [(percentPos(V2, OST) \times priority(OST)) \\ &+ (percentPos(V2, AP) \times priority(AP)) \\ &+ (percentPos(V2, QA) \times priority(QA)) \\ &+ (percentPos(V2, RC) \times priority(RC))] \end{aligned}$$

$$\begin{aligned} priority\,(V2) &= [(\frac{2}{6} \times 0,75 \times 10) + (\frac{2}{6} \times 0,75 \times 9) + (\frac{1}{6} \times 0,75 \times 6) + (\frac{1}{6} \times 0,75 \times 9)] \\ &= [2,50 + 2,25 + 0,75 + 1,125] = 6,625 \end{aligned}$$

$$\begin{aligned} priority\,(V3) &= [(percentPos(V3, OST) \times priority(OST)) \\ &+ (percentPos(V3, AP) \times priority(AP)) \\ &+ (percentPos(V3, QA) \times priority(QA)) \\ &+ (percentPos(V3, RC) \times priority(RC))] \end{aligned}$$

$$\begin{aligned} priority\,(V3) &= [(\frac{2}{7} \times 0,75 \times 10) + (\frac{2}{7} \times 0,75 \times 9) + (\frac{2}{7} \times 0,75 \times 6) + (\frac{1}{7} \times 0,75 \times 9)] \\ &= [1,25 + 2,25 + 1,50 + 1,125] = 6,328 \end{aligned}$$

$$\begin{aligned} priority\,(V4) &= [(percentPos(V4, OST) \times priority(OST)) \\ &+ (percentPos(V4, AP) \times priority(AP)) \\ &+ (percentPos(V4, QA) \times priority(QA)) \\ &+ (percentPos(V4, RC) \times priority(RC))] \end{aligned}$$

$$\begin{aligned} priority\,(V4) &= [(\frac{1}{5} \times 0,75 \times 10) + (\frac{2}{5} \times 0,75 \times 9) + (\frac{1}{5} \times 0,75 \times 6) + (\frac{1}{5} \times 0,75 \times 9)] \\ &= [1,50 + 2,70 + 0,90 + 1,35] = 6,45 \end{aligned}$$

Dentre as variantes analisadas, V2 (Figura 14), com valor 6,625 para a priorização, é a melhor opção para o cliente que deseja prioridade máxima para **Otimização [Casos de Teste]**. Dessa forma, essa foi a configuração escolhida.

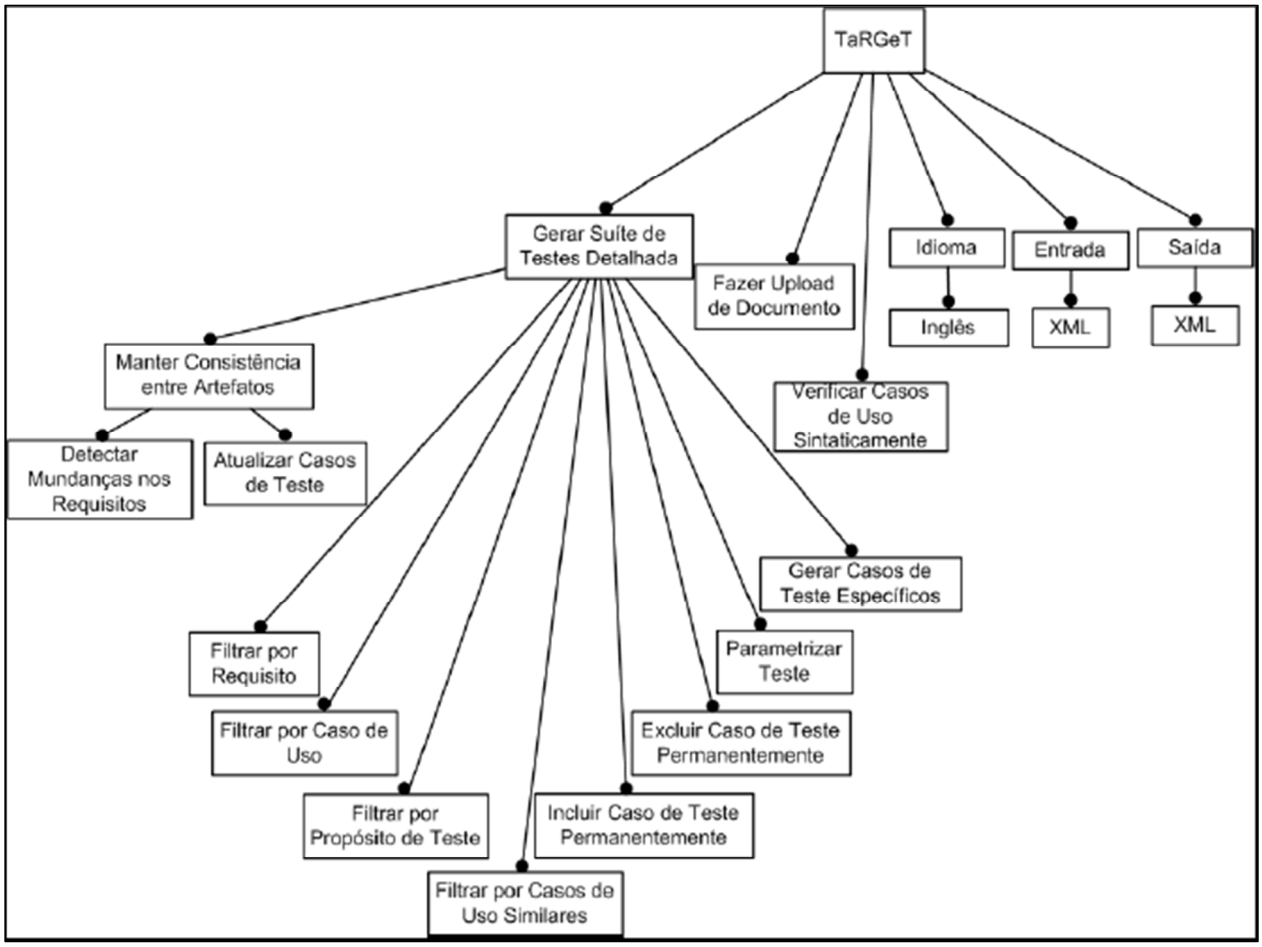
A atividade de **configuração dos artefatos do produto** visa configurar todos os artefatos de requisitos desenvolvidos para a LPS, apenas para a variante escolhida na atividade de priorização. Ela é composta de três etapas: elaboração do modelo de configuração do produto (construído com base no modelo de *feature* e no modelo SR da variante), remoção da cardinalidade de elementos e relacionamentos no modelo SR e configuração dos cenários de caso de uso do produto (baseado nas *features* escolhidas para a variante). Estes passos são detalhados a seguir.

Passo 1: Elaborar Modelo de Configuração de Produto

O modelo de configuração do produto é um diagrama das *features* selecionadas para um produto específico [Borba, 2009]. Para elaborá-lo deve-se analisar o modelo SR da variante (Figura 14) para extrair as *features* que estarão presentes no modelo de configuração. Para descobrir as *features* associadas ao modelo SR, deve-se consultar a tabela de rastreamento entre *features* e os elementos que as geraram (Quadro 5). Depois, essas *features* devem ser comparadas com o modelo de *feature* (Figura 11) para verificar se as *features* da configuração satisfazem as relações do modelo de *feature*. Por fim, gera-se um modelo de configuração válido com as features selecionadas.

Ao aplicar este passo ao exemplo, o modelo de configuração do produto em questão é apresentado na Figura 17.

Figura 17 Modelo de configuração do produto TaRGeT [Souza 2012]

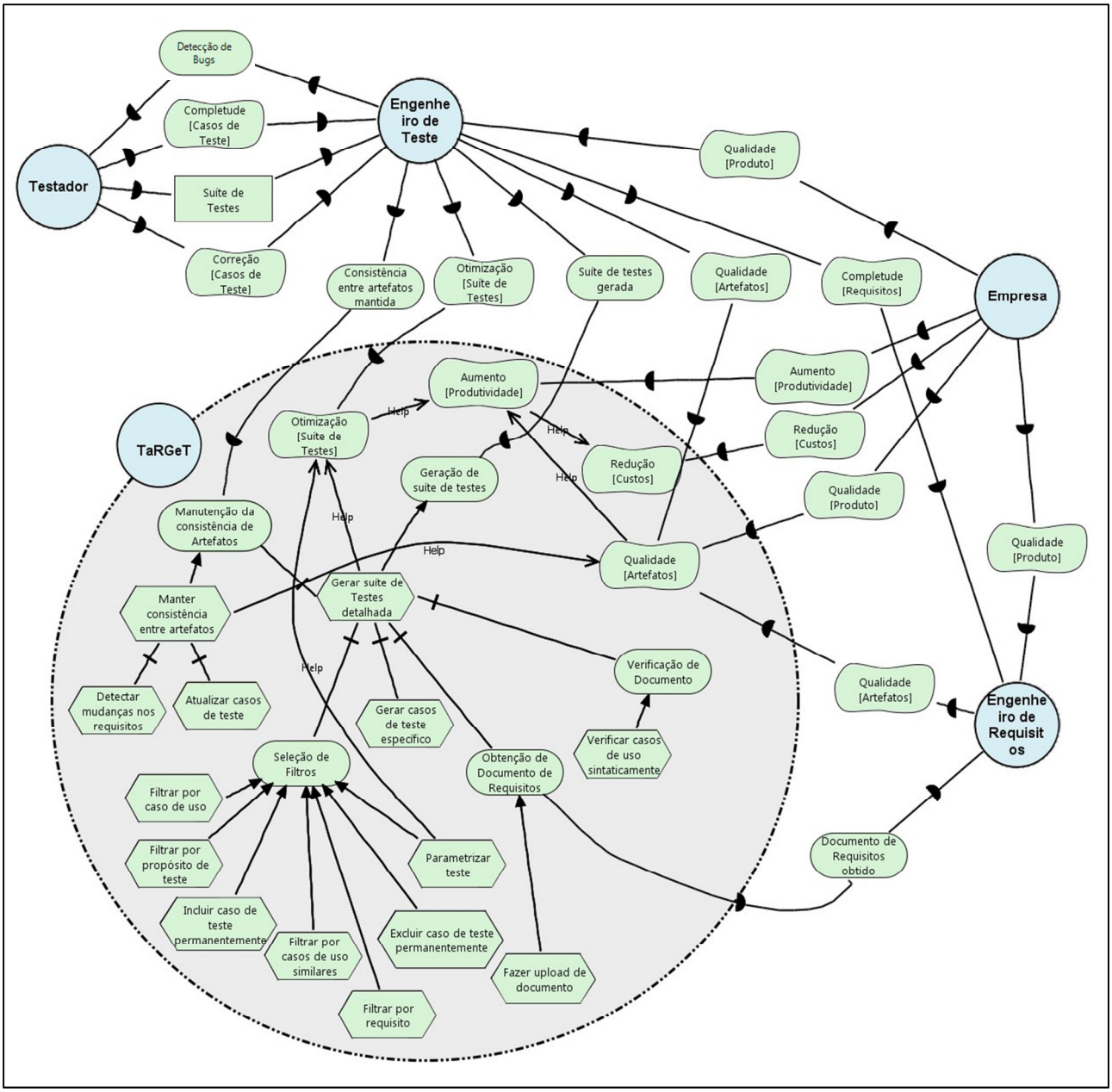


Passo 2: Remover a cardinalidade de elementos e relacionamentos no modelo SR

Uma vez que a configuração foi escolhida, deve-se remover todas as indicações de cardinalidade do modelo de objetivos para que ele passe a ser um modelo SR em *i**-wiki [Grau et al. 2008]. Também é necessário que se removam as marcações feitas para destacar elementos que originaram *features*. Segundo Souza (2012), o modelo SR do produto serve para manter um rastreamento entre as intenções dos interessados e os requisitos do produto.

A Figura 18 mostra o modelo SR final do produto correspondente à configuração selecionada.

Figura 18 Modelo SR do produto TaRGeT [Souza 2012]

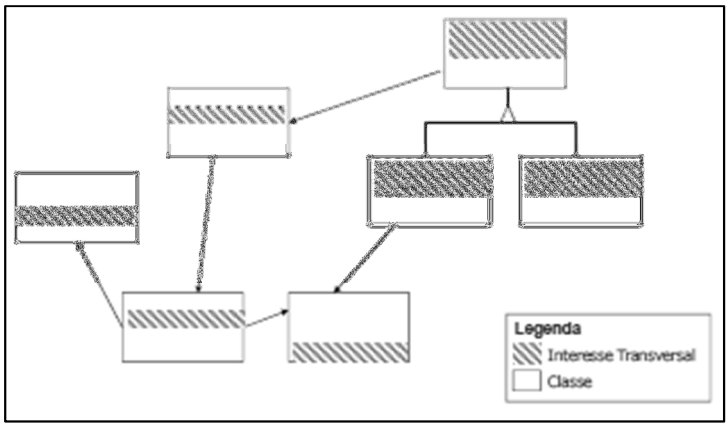


Analisando o cenário da LPS produzido pela abordagem GS2SPL (Figura 12) e conforme afirma Almeida (2010), percebe-se que a notação PLUSS [Eriksson, Börstler e Borg 2005] utiliza um único artefato para representar todas as configurações válidas relacionadas a um cenário. Misturando comportamentos comuns, variantes e informações de configuração (referências a *features* dentro de colchetes). Com isso, todas as variações são descritas no mesmo artefato dificultando o entendimento do produto que está sendo especificado e aumentando problemas de manutenção. Como afirmam Alferez et al. (2013), essa técnica não é adequada para modularizar a natureza transversal de certas *features*, tem pobre legibilidade e leva a baixa manutenabilidade. Diante disso devemos utilizar uma técnica de especificação de cenários que se preocupa com a separação de interesses transversais.

2.3. Orientação a Aspectos

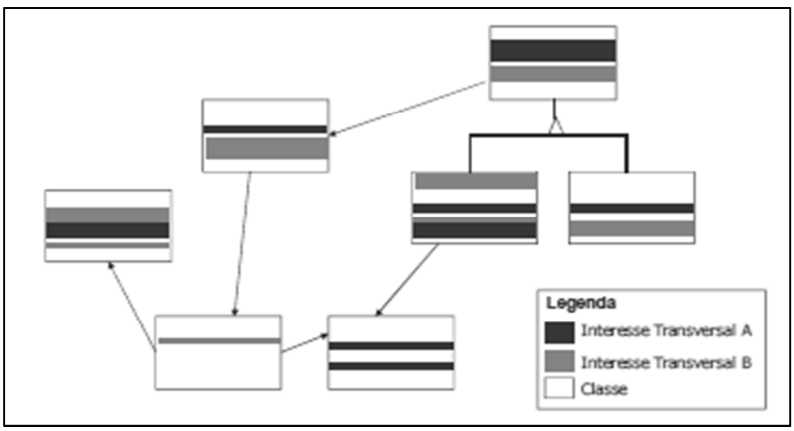
O termo separação de interesses (do inglês, *separation of concerns*) foi empregado pela primeira vez por Dijkstra (1976) no sentido de dividir um problema em partes menores visando reduzir sua complexidade, sendo cada parte capaz de representar um interesse. Interesses (do inglês, *concerns*) podem ser entendidos como requisitos funcionais ou não funcionais. Quando estes interesses estão espalhados (do inglês, *scattering*) ou entrelaçados (do inglês, *tangling*) a outros interesses, são chamados transversais [Clements e Northrop 2002]. Um interesse está espalhado (do inglês, *scattering*) quando afeta vários componentes do sistema. A Figura 19 apresenta um interesse transversal espalhado entre várias classes de um sistema.

Figura 19 Exemplo de espalhamento. Adaptado de Figueiredo (2006)



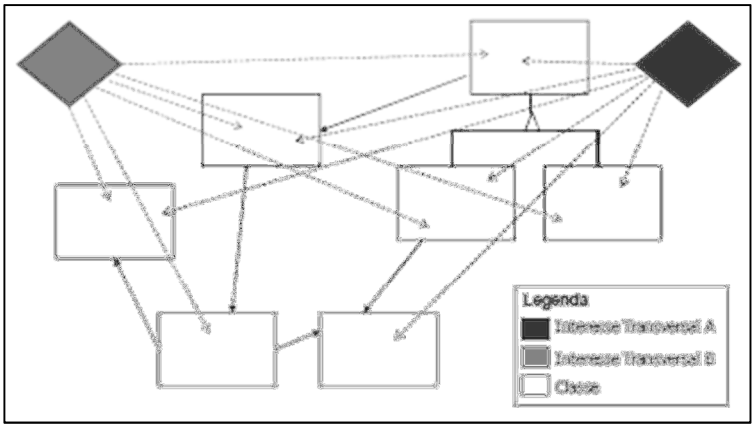
Um interesse transversal é dito entrelaçado (do inglês, *tangling*) quando se mistura com outros interesses dentro de um módulo. O entrelaçamento favorece o aumento de dependência entre os componentes, tornando-os menos reusáveis e mais propensos a erros [Kiczales et al, 1997]. A Figura 20 apresenta dois interesses transversais entrelaçados entre várias classes de um sistema.

Figura 20 Exemplo de entrelaçamento. Adaptado de Figueiredo (2006)



A separação de interesses é a base da Orientação a Aspecto [Kiczales et al, 1997]. Neste paradigma, os interesses transversais são modularizados em aspectos para separá-los dos outros componentes do sistema. Cada aspecto encapsula um requisito ou cenário que afeta outros requisitos do sistema, facilitando a compreensão sobre o sistema e permitindo o reuso de artefatos [Rashid et al. 2002]. Um aspecto pode afetar a estrutura estática ou dinâmica de uma ou mais classes ou objetos. A estrutura dinâmica é alterada pela especificação de conjuntos de pontos de junção (do inglês, *join points*) que identificam pontos de junção e, adendos (do inglês, *advices*) que implementam o comportamento a ser adicionado ao ponto de junção. A estrutura estática é alterada através de declarações de atributos e métodos, chamada de inter-tipo (do inglês, *intertype declarations*). A Figura 21 apresenta os dois interesses modularizados em aspectos.

Figura 21 Exemplo de uma modularização em aspectos. Adaptado de Figueiredo (2006)



Um ponto de junção (do inglês, *join point*) é qualquer ponto identificável e bem definido onde um interesse vai interceptar durante a execução de um programa. Um conjunto de pontos de junção (do inglês, *pointcut*) são utilizados para identificar um grupo de *join points*, em que os comportamentos adicionais inseridos pelos aspectos devem ser executados. A seguir, é apresentado um exemplo do código em AspectJ [Goetten and Winck, 2006] de um conjunto de pontos de junção.

```
public pointcut printObject() : call (String *.toString());
```

O exemplo acima apresenta os elementos de um conjunto de ponto de junção. A declaração de restrição de acesso público (*public*), mas também podem ser privado (*private*) e protegido (*protected*). A palavra reservada *pointcut* que informa que é uma representação de um ponto de junção. Em seguida, um identificador (*printObject ()*) que pode ou não vir

acompanhado por parâmetros. Depois dos (:) vem o tipo e os pontos de junção que são todas as chamadas a métodos *toString()*.

Os *advices*, em AspectJ, especificam o comportamento do aspecto e o momento em que estas operações são realizadas. Eles podem ser de três tipos: anterior (do inglês, *before*), posterior (do inglês, *after*) ou ao redor (do inglês, *around*) que são executados no lugar do ponto de junção.

O *before* é executado antes da ocorrência do ponto de junção (*joinpoint*) [Goetten e Winck 2006]. A seguir um exemplo de um *advice before*.

```
before() : printObject() {
   System.out.println("Antes de imprimir!");
}
```

O *after* é o bloco de código executado no fim da execução do ponto de junção [Goetten e Winck 2006]. Dependendo da maneira como terminou a execução do ponto de junção há três variações do *after*. O primeiro executa depois do ponto de junção, independentemente de como ele retornou. O segundo executa depois do ponto de junção se ele tiver terminado normalmente (sem exceção). O terceiro executa depois do ponto de junção somente se tiver saído com uma exceção.

```
after() : printObject() {
   // executa independente de como retornar
   // o ponto de junção
}

after() returning : printObject() {
   // executa se o ponto de junção terminar normalmente
}

after() throwing : printObject() {
   // executa se o ponto de junção sair com uma exceção
}
```

O último tipo é o *around* que é executado durante a execução do ponto de junção ou bloco de código, ou seja, o *around* cerca toda a execução do ponto de junção [Goetten e Winck 2006].

As declarações inter-tipo (*intertype declarations*) permitem inserir novos atributos e métodos nas classes básicas do programa e, também podem mudar a hierarquia das classes. Neste exemplo, o *intertype declaration* adiciona uma variável (*senha*), um método (*verificar(int s)*) e uma superclasse a classe Aluno (*Aluno extends Pessoa*).

```
private int Aluno.senha;

public boolean Aluno.verificar(int s) {
   return (senha == s);
}

declare parents : Aluno extends Pessoa;
```

2.4. Especificação de Cenários com Interesses Transversais

Para complementar a especificação de requisitos obtida com abordagens GORE, o comportamento dinâmico da LPS pode ser descrito através de uma técnica de especificação de cenários. Os cenários descrevem o comportamento das funcionalidades do sistema e são muito utilizados na ER de sistemas, pois são de fácil compreensão por parte dos interessados [Maiden e Alexander 2004].

Como afirma Bonifácio e Borba (2009), apesar dos benefícios da representação da variabilidade em cenários, existem abordagens que não apresentam uma separação clara entre gerenciamento de variabilidade e especificação de cenários de casos de uso. De fato, em muitas técnicas de cenários com variabilidade, existem detalhes sobre variantes de produtos e informações de configuração que estão entrelaçadas. Nesse caso, as informações de configuração são chamadas de interesses transversais (do inglês, *crosscutting concerns*) e podem comprometer a manutenibilidade e o entendimento dos artefatos da LPS [Bonifácio e Borba 2009]. Os interesses transversais são requisitos cujo comportamento pode ter impacto ou influenciar múltiplos módulos ou componentes do sistema [Kiczales et al. 1997]. Exemplos comuns de interesses transversais são: rastreamento, auditoria, persistência, distribuição e tratamento de erros. Os interesses transversais precisam ser identificados e modularizados mais cedo possível, pois, caso sua identificação seja tardia poderá acarretar um maior custo na resolução de problemas que possam surgir no que diz respeito à evolução e manutenção [Araujo, Whittle e Kim 2004].

A separação de interesses é a base da orientação a aspectos, que modulariza os interesses transversais em aspectos para separá-los dos outros componentes do sistema. Na ER, cada aspecto encapsula um requisito que afeta outros requisitos do sistema, facilitando a compreensão sobre o sistema e o reuso [Rashid et al. 2002].

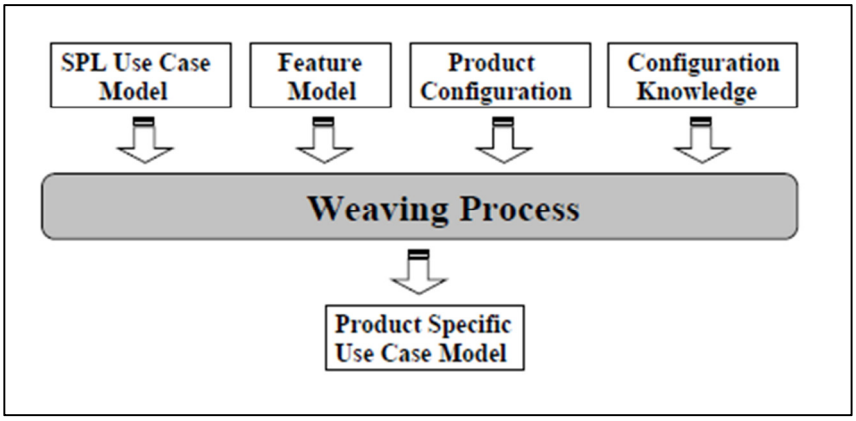
2.4.1. MSVCM

A técnica MSVCM (*Modeling Scenario Variability as Crosscutting Mechanisms*) elaborada por Almeida (2010) trata do gerenciamento de variabilidade de cenário de caso de uso, melhorando a separação de interesses entre o gerenciamento de variabilidades e as especificações de cenários de caso de uso. MSVCM é uma abordagem composicional, ou seja, aplica técnicas orientadas a aspectos para a especificação de variabilidade de cenários. Essa abordagem utiliza diferentes módulos para representar as características comuns (ou

seja, cenários base) e variabilidades (*advices*), mas também usa anotações nos cenários base para mostrar onde o *advice* deve ser aplicado. O gerenciamento da variabilidade é tratado como um fenômeno transversal e, por isso, precisa ser separado.

O MSVCM utiliza um processo de combinação (do inglês, *weaving process*), ilustrado na Figura 22, que recebe como entrada os artefatos: modelo de caso de uso (do inglês, *SPL use case model*), modelo de *feature* (do inglês, *feature model*), configuração do produto (do inglês, *product configuration*) e conhecimento de configuração (do inglês, *configuration knowledge*). E, ao final, gera um modelo de caso de uso de um produto específico.

Figura 22 Processo da técnica MSVCM. Extraído de Bonifácio e Borba (2009)



O modelo de *feature* é utilizado para verificar se uma configuração de produto é um membro válido de uma LPS. Foram definidas funções usando a linguagem de programação Haskell[5] para verificar se todas as restrições definidas no modelo de *feature* foram atendidas para uma configuração do produto. O **modelo de caso de uso** da LPS define cenários que descrevem o comportamento esperado dos membros da LPS. Este artefato possui um modelo de caso de uso (ações do usuário e respostas do sistema) e um modelo de caso de uso aspectual que possui uma lista de *advices* (que representa a variabilidade e é utilizado para ampliar o comportamento dos cenários existentes).

A Figura 23 apresenta um exemplo de cenário e *advice* definidos através da técnica MSVCM para a LPS TaRGeT. Há duas maneiras de gerar uma suíte de testes: de forma detalhada ou gerando todos os casos de teste. Assim, analisa-se o modelo de *feature* e identifica-se que **Gerar Suíte de Testes Detalhada** é uma *feature* obrigatória e, portanto, deverá ser representada por um cenário. No caso, o cenário principal do caso de uso **Suíte de testes**

[5] http://www.haskell.org/haskellwiki/Haskell

gerada detalhadamente. A *feature* **Manter consistência entre artefatos** é opcional e, representa uma variabilidade, por isso, é mapeada como um *advice*. Portanto, o cenário **Suíte de testes gerada detalhadamente** é composto pelo *advice* **Manter consistência entre artefatos**.

Figura 23 Cenário de caso de uso especificado com MSVCM

UC02
Suíte de testes gerada detalhadamente

SC_02
Descrição: Gerar suíte de teste detalhadamente

ID	Ação do Usuário	Resposta do Sistema
SC_02.1	Selecionar uma das formas de verificar documento	Informa que casos de uso foram verificados e exibe a janela de seleção de filtros com botão para confirmar a geração de suíte de testes
SC_02.2	Seleciona filtros	-
SC_02.3	@consistencia	-
SC_02.4	Confirma a geração da suíte de testes	Gerar caso de teste de acordo com os filtros selecionados

ADV01
Descrição: Manter consistência entre os artefatos
Pointcut: Before @consistencia

ID	Ação do Usuário	Resposta do Sistema
ADV01.1	Detecta que houve mudanças nos requisitos em relação à versão anterior do documento	-
ADV01.2	Atualiza casos de teste relacionados aos casos de uso modificados	-

Apenas o modelo de *feature* não é capaz de representar como os produtos serão gerados a partir dos artefatos da LPS. Por isso, o **Conhecimento de Configuração** realiza o mapeamento entre o modelo de *feature* e os artefatos de implementação. Ele é elaborado de forma separada e corresponde a uma lista de itens de configuração que relaciona expressões de *features* a tarefas (ou transformações) usadas para gerar automaticamente um membro da LPS [Almeida 2010].

O conhecimento de configuração definido no MSVCM utiliza três transformações existentes no Hephaestus para lidar com os diferentes tipos de variabilidade: *evaluate advice*, *select scenario* e *bind parameter* [Bonifácio, Teixeira e Borba 2009].

Casos de uso aspectuais especificam o comportamento opcional (*advices*) exigido para a configuração das *features* e tratam da variabilidade em controle de fluxo. Variabilidade em controle de fluxo é utilizada para modularizar passos opcionais e alternativos que estão

entrelaçados com especificações comuns de um cenário [Bachmann e Bass 2001]. Para isto, foi definida a transformação *evaluate advice*, que faz a composição de *advices* a *join points* específicos.

Parametrização de cenário define o comportamento comum e, lida com variabilidade em dados. Variabilidade em dados ocorre sempre que dois ou mais cenários compartilham o mesmo comportamento (a sequência de passos) e diferem apenas em relação a valores de um mesmo conceito [Bachmann e Bass 2001]. Então, Bachmann e Bass (2001) definiram a transformação *bind parameters*, responsável por ligar parâmetros a *features*.

A transformação de seleção de cenário *select scenario* é usada para selecionar cenários em configuração de produto específico e, está relacionado a variabilidade em função. Variabilidade em função ocorre quando uma função particular pode existir em alguns produtos e não em outros [Bachmann e Bass 2001].

Configuração do produto identifica um membro específico de uma LPS que é caracterizado por uma configuração de *features* que deve cumprir os relacionamentos e restrições especificados no modelo de *feature* correspondente. Visando tratar da variabilidade em cenário textual foi definida uma maneira de mapear cada variabilidade encontrada: um cenário opcional, um passo parametrizado ou um *advice*. Isto é utilizado para a definição do conhecimento de configuração.

Na especificação de cenário apresentada anteriormente na Figura 23 não há referência explícita a *features*. Esta relação é feita pelo Conhecimento de Configuração, como mostra a Figura 24.

Figura 24 Conhecimento de Configuração da LPS TaRGeT

Expressão de feature	Transformações
TaRGeT	select scenario SC04, SC05 evaluate advice ADV08, ADV09
Gerar suíte de testes detalhada	select scenario SC02 evaluate advice ADV02, ADV03, ADV04, ADV05, ADV06, ADV07
Parametrizar Teste	evaluate ADV01
Gerar suíte de testes diretamente	select scenario SC03
Fazer upload de documento	select scenario SC04
Editor de Caso de Uso	evaluate advice ADV08
Manter Consistência entre Artefatos	select scenario SC01
Checar Caso de Teste Sintaticamente	select scenario SC05
Checar Caso de Teste Semanticamente	evaluate advice ADV09

De acordo com Alferez et al. (2013), mecanismos orientados a aspectos do MSVCM oferecem grandes contribuições para melhorar a modularização de variabilidades na especificação de cenários. O MSVCM favorece a separação de interesses entre o gerenciamento da variabilidade da LPS e as especificações dos cenários da LPS, lidando com a variabilidade dos cenários como sendo uma composição de diferentes artefatos. Além de apoiar a separação de interesses mencionada, esta técnica também ajuda a especificar precisamente como compor diferentes representações a fim de gerar cenários específicos para um único produto da LPS.

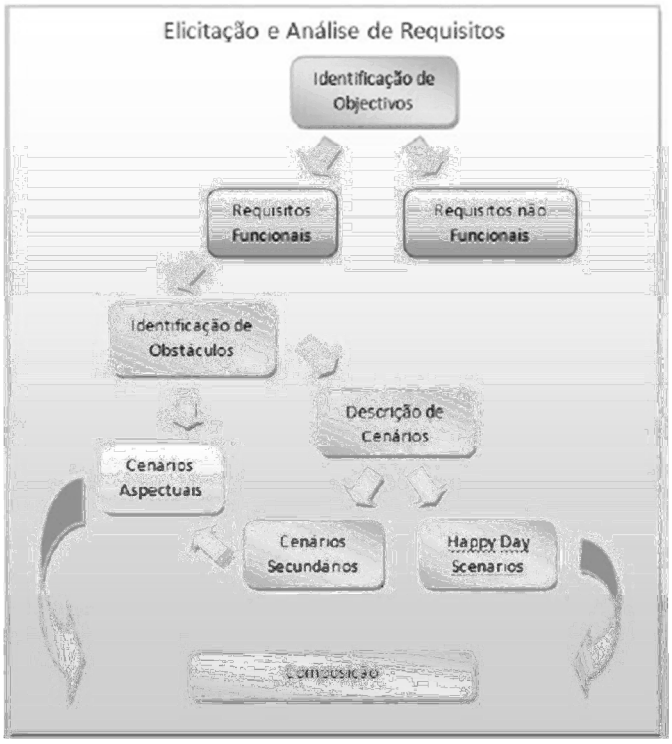
2.5. Trabalhos Relacionados

Os trabalhos relacionados considerados foram abordagens que davam suporte a especificação de cenários aspectuais, pois a comunidade que trabalha com GORE e LPS é pequena e o trabalho mais completo de engenharia de requisitos para SPL de que temos conhecimento é o GS2SPL, que foi estendido pela abordagem proposta.

2.5.1. SceneKaos

A abordagem proposta por Oliveira, Araújo e Silva (2010) visa integrar uma técnica orientada a objetivos, KAOS (*Keep All Objectives Satisfied*) [Lapouchnian 2005], e uma técnica de especificação de cenários aspectuais, MATA (*Modeling Aspects Using a Transformation Approach*) [Whittle e Jayaraman 2007], levando em consideração os interesses transversais. Como resultado dessa integração, definiu-se a abordagem *SceneKaos*, cujas atividades situam-se no contexto da Elicitação e Análise de Requisitos. A abordagem possui um processo, ilustrado na Figura 25, onde, incialmente, identificam-se os objetivos do sistema, que podem gerar requisitos funcionais ou não funcionais. Em seguida, identificam-se os obstáculos e, elaboram-se os cenários aspectuais e as descrições dos cenários, que podem ser cenários principais ou *happy day* cenários (cenários onde tudo ocorre como planejado, sem exceções) e, cenários secundários, onde são tratadas as exceções (obstáculos).

Figura 25 Processo SceneKAOS, extraído de Oliveira, Araújo e Silva (2010)



Inicialmente, no processo da abordagem *SceneKAOS*, identificam-se os objetivos e requisitos do sistema através da realização de uma reunião com os interessados para obter o máximo de informação sobre o sistema (características e exceções que deverão ser tratadas e resolvidas). Estas exceções são modeladas utilizando o conceito de obstáculos de KAOS. Segundo Oliveira, Araújo e Silva (2010) obstáculos são situações que têm como consequência o impedimento da realização do objetivo ou da funcionalidade.

Tendo como base os objetivos, requisitos e os obstáculos identificados, elaboram-se os cenários. Estes permitem obter uma descrição detalhada de como os objetivos serão detalhados, tomando por base o modelo de objetivos do KAOS. É estabelecida uma relação entre os objetivos e os cenários principais identificados. Em seguida, estabelece-se a relação entre os obstáculos e os cenários secundários. Esta relação permite identificar as ações que podem impedir a realização dos objetivos, bem como, identificar os obstáculos que podem ocorrer e em que situações podem prejudicar a realização dos objetivos.

Ao analisar os objetivos e a descrição dos seus cenários, podemos encontrar comportamentos comuns entre vários cenários. Estes cenários são chamados cenários aspectuais e indicam a existência de requisitos comuns entre objetivos. Estes requisitos podem ser modularizados em aspectos e descritos utilizando a técnica MATA. Modularizar estes comportamentos comuns em cenários aspectuais permite reutilizá-lo em diferentes objetivos do

sistema. Além disso, facilita a escrita do código e também a sua manutenção.

De acordo com Oliveira, Araújo e Silva (2010) com a utilização de *SceneKAOS* pode-se modelar cenários aspectuais a partir do modelo de objetivos KAOS, permitindo obter um modelo de requisitos com um nível elevado de detalhamento e bem modularizado através de cenários aspectuais especificados com a técnica MATA.

No entanto, essa abordagem considera apenas os modelos de objetivos e as especificações de cenários com separação de interesses transversais, descartando a representação de variabilidade necessária para especificar uma LPS e, consequentemente, não existe um processo para configurar produtos variantes de uma LPS.

2.5.2. Oliveira et al. (2006)

O trabalho de Oliveira et al. (2006), une o *framework i**, o *framework* AspecT e a descrição textual de cenários para atingir uma melhor compreensão nas fases iniciais de um projeto, dos interesses e agentes envolvidos no processo de realização dos objetivos e das suas influências nas fases posteriores do desenvolvimento. De acordo com Oliveira et al. (2006), neste trabalho, o *framework i** modela contextos organizacionais utilizando os relacionamentos estratégicos entre atores. O *framework* AspecT é utilizado na implementação de sistemas multi-agente, pois permite modelar propriedades dos agentes de maneira modular. Os cenários produzem uma descrição estruturada de uma situação que ocorre no mundo real.

Os sistemas multi-agentes possuem características que podem ser tratadas como interesses transversais como, por exemplo, autonomia, adaptação e interação. Portanto, utilizar abordagens orientadas a aspectos para tratar destes interesses transversais é um estratégia que vem sendo bastante utilizada tanto no projeto como na implementação. Nesta abordagem, objetos encapsulam as funcionalidades básicas do sistema como, por exemplo, crenças e recursos; enquanto mecanismos orientados a aspectos são usados para modularizar os interesses dos agentes como, por exemplo, interação, adaptação, autonomia, aprendizado, mobilidade e colaboração.

O processo da abordagem consiste de três etapas: definir o estado dos agentes e comportamentos; capturar e associar propriedades dos agentes; criar tipos de agentes. A primeira atividade prepara a definição de cenários que utiliza símbolos UofD (do inglês, *Universe of Discourse*). UofD de acordo com Oliveira et al. (2006) são todas as fontes de informação e todas as pessoas relacionadas ao sistema. A segunda atividade utiliza os cenários, de acordo com os conceitos do framework i* e dos sistemas multi-agentes para produzir os modelos de

Dependência Estratégica (SD) e Razão Estratégica (SR). A terceira atividade implementa o sistema utilizando os cenários e os modelos, de acordo com os conceitos dos sistemas multi-agentes e AspecT e, dos cenários. Contudo, esta abordagem não dá suporte à especificação de linhas de produto.

2.5.3. MATA

Outro trabalho que se preocupa com a separação de interesses transversais na especificação de cenários é a técnica Modelagem de Aspectos Usando uma Abordagem de Transformação (do inglês, *Modeling Aspects Using a Transformation Approach - MATA*) [Whittle e Jayaraman 2007]. MATA é uma abordagem de modelagem orientada a aspectos que utiliza transformações de grafos para especificar e compor aspectos visando tratar a variabilidade da linha de produto em modelos UML. Em um grafo, cada nó e aresta podem ser rotulados com o par (valor, tipo), dado o valor do atributo e seu tipo. Portanto, um metamodelo UML pode ser representado como um tipo grafo. Cada metaclasse torna-se um nó do grafo e cada meta-associação torna-se uma aresta do grafo. A especificação de cenários aspectuais em MATA é feita da seguinte forma: um cenário base, portanto não aspectual, pode ser especificado, por exemplo, através de um diagrama de sequência, enquanto que um cenário aspectual é descrito também através de um diagrama de sequência enriquecido com papéis (do inglês, *roles*) que funcionam como variáveis que deverão ser instanciadas no momento da composição do cenário aspectual com o cenário base. MATA, atualmente, suporta composição dos seguintes diagramas UML, classe, sequência e estado.

Porém esta abordagem sozinha não é adequada para especificar cenários de variabilidade de uma LPS. Para capturar a variabilidade da linha de produto ela precisaria ser usada em conjunto ao menos com o modelo de *feature*.

2.5.4. Comparativo das abordagens

Buscando apresentar um comparativo entre as abordagens citadas nesta seção e o GS2SPL e MSVCM, elaboramos o Quadro 8 com os seguintes critérios: (*i*) utiliza abordagens orientada a objetivos; (*ii*) utiliza cenários de caso de uso; (*iii*) suporte a cenários aspectuais; (*iv*) representação da variabilidade da LPS através de modelo de *features*; (*v*) aborda a fase de Configuração de Produto da Engenharia de Aplicação; (*vi*) representação do conhecimento de configuração (*configuration knowledge*) separadamente; (*vii*) utiliza requisitos não funcionais na configuração. É utilizado **SIM** para identificar que a abordagem cumpre positivamente este critério.

Quadro 8 Comparação entre as abordagens

Abordagens / Critérios	[Oliveira et al. 2006]	*SceneKaos*	MATA	GS2SPL	MSVCM	GAS2SPL
Utiliza abordagens orientadas a objetivos	**SIM**	**SIM**	**-**	**SIM**	**-**	**SIM**
Utiliza cenários de caso de uso	**SIM**	**SIM**	**SIM**	**SIM**	**SIM**	**SIM**
Suporte a cenários aspectuais	**-**	**SIM**	**SIM**	**-**	**SIM**	**SIM**
Representação da variabilidade da LPS através do modelo de *features*	**-**	**-**	**-**	**SIM**	**SIM**	**SIM**
Aborda a fase de Configuração de Produto da Engenharia de Aplicação	**-**	**-**	**-**	**SIM**	**SIM**	**SIM**
Representação do conhecimento de configuração (*configuration knowledge*) separadamente	**-**	**-**	**-**	**-**	**SIM**	**SIM**
Utiliza requisitos não funcionais na configuração	**-**	**-**	**-**	**SIM**	**-**	**SIM**

A abordagem proposta nesta dissertação (GAS2SPL) visa preencher as lacunas deixadas pelas abordagens apresentadas anteriormente. Assim, esta abordagem utilizará os objetivos dos interessados para identificar as *features* e, mapear requisitos funcionais e não funcionais. Os requisitos não funcionais terão valores atribuídos pelos interessados, representando a prioridade de cada um de acordo com o objetivo do interessado para o produto da LPS a ser derivado. A variabilidade da linha de produto será representada pelo modelo de *feature*, que irá conter as restrições que devem estar presentes nos produtos da linha. O conhecimento de configuração, utilizado para relacionar a variabilidade da linha de produto e os artefatos, guiando a geração de produtos será um artefato que permite separar a variabilidade da linha de produto do conhecimento de configuração. A abordagem especifica os cenários de caso de uso utilizando recursos da orientação a aspecto para separar os interesses transversais, reduzindo o espalhamento de *features* e o entrelaçamento de cenários.

2.6. Considerações Finais

Neste capítulo foram apresentados alguns conceitos relacionados a Engenharia de Requisitos e a Linhas de Produto de Software. Descreveram-se, de forma resumida, as abordagens orientadas a objetivos, o framework *i** e a abordagem GS2SPL. Além de conceitos relacionados a especificação de cenários, com a apresentação da técnica de especificação de cenários com separação de interesses transversais, MSVCM. Enumerou-se as diretrizes propostas por Santander (2002) para mapear diagramas *i** em casos de uso.

A abordagem GS2SPL leva em consideração os objetivos dos interessados para identificar as *features* e elaborar os cenários, mas não utiliza recursos da orientação a aspectos nestes artefatos. Assim, a variabilidade e as partes comuns na linha estão presentes no mesmo cenário de caso de uso, dificultando o entendimento e a evolução da LPS, assim como a configuração dos produtos. Se o engenheiro utilizar somente o GS2SPL, se beneficiará com a identificação sistemática das features e cenários a partir dos modelos de objetivos, a configuração do produto será dirigida pela satisfação de requisitos não funcionais, mas não terá a separação das *features* nos cenários promovida pelo MSVCM. Assim, se usar somente o MSVCM, não terá os benefícios providos pelo GS2SPL. Fazendo a integração das duas técnicas será possível somar os benefícios de ambas e poder derivar os artefatos de requisitos da LPS a partir do modelo de objetivos, que é um modelo mais próximo da realidade dos interessados por capturar os seus objetivos.

Neste capítulo também foram apresentados alguns trabalhos relacionados à abordagem proposta neste trabalho, bem como, uma tabela comparativa das abordagens mostrando suas características em relação aos critérios apresentados. Assim, o objetivo da comparação foi identificar quais abordagens davam suporte a LPS e qual o grau de suporte, visando enfatizar que o GAS2SPL é o processo de engenharia de requisitos de LPS mais completo e que também dá suporte a especificação de cenários aspectuais.

No próximo capítulo é detalhada a nova abordagem, baseada no GS2SPL, e na técnica de especificação de cenário com separação de interesses transversais, MSVCM. A abordagem GAS2SPL (do inglês, *Goals and Aspectual Scenarios to Software Product Lines*) integra modelos de objetivos, modelos de *feature,* cenários de caso de uso com separação de interesses transversais (ou, simplesmente, cenários aspectuais). Além de representar o conhecimento de configuração em um artefato separado e, a configuração de produto que leva em consideração a prioridade atribuída pelos interessados aos requisitos não funcionais. Suas atividades são aplicadas ao TaRGeT [TARGET 2011].

Capítulo 3 Goals and Aspectual Scenarios to Software Product Lines (GAS2SPL)

Este capítulo apresenta a proposta desta dissertação, a abordagem GAS2SPL (do inglês, *Goals and Aspectual Scenarios to Software Product Lines*). Esta abordagem faz parte da Engenharia de Domínio e de Aplicação, porém apenas para a fase de requisitos, do *framework* para LPS de Pohl, Böckle e Linden (2005). GAS2SPL é uma abordagem orientada a objetivos estendida da abordagem GS2SPL [Souza 2012] visando incluir especificações de cenários com separação de interesses transversais através da utilização da técnica MSVCM [Almeida 2010] adaptada ao processo de GS2SPL. Visando tratar da inserção de novos atributos ou alteração na hierarquia de classes na especificação de cenários alternativos, em MSVCM, fez-se necessário estender esta técnica para tratar do *intertype declaration*. A abordagem proposta permite obter, sistematicamente, cenários aspectuais a partir do modelo de objetivos e modelo de *feature*. Além de possuir um artefato para representar o conhecimento de configuração separadamente da variabilidade da linha de produto.

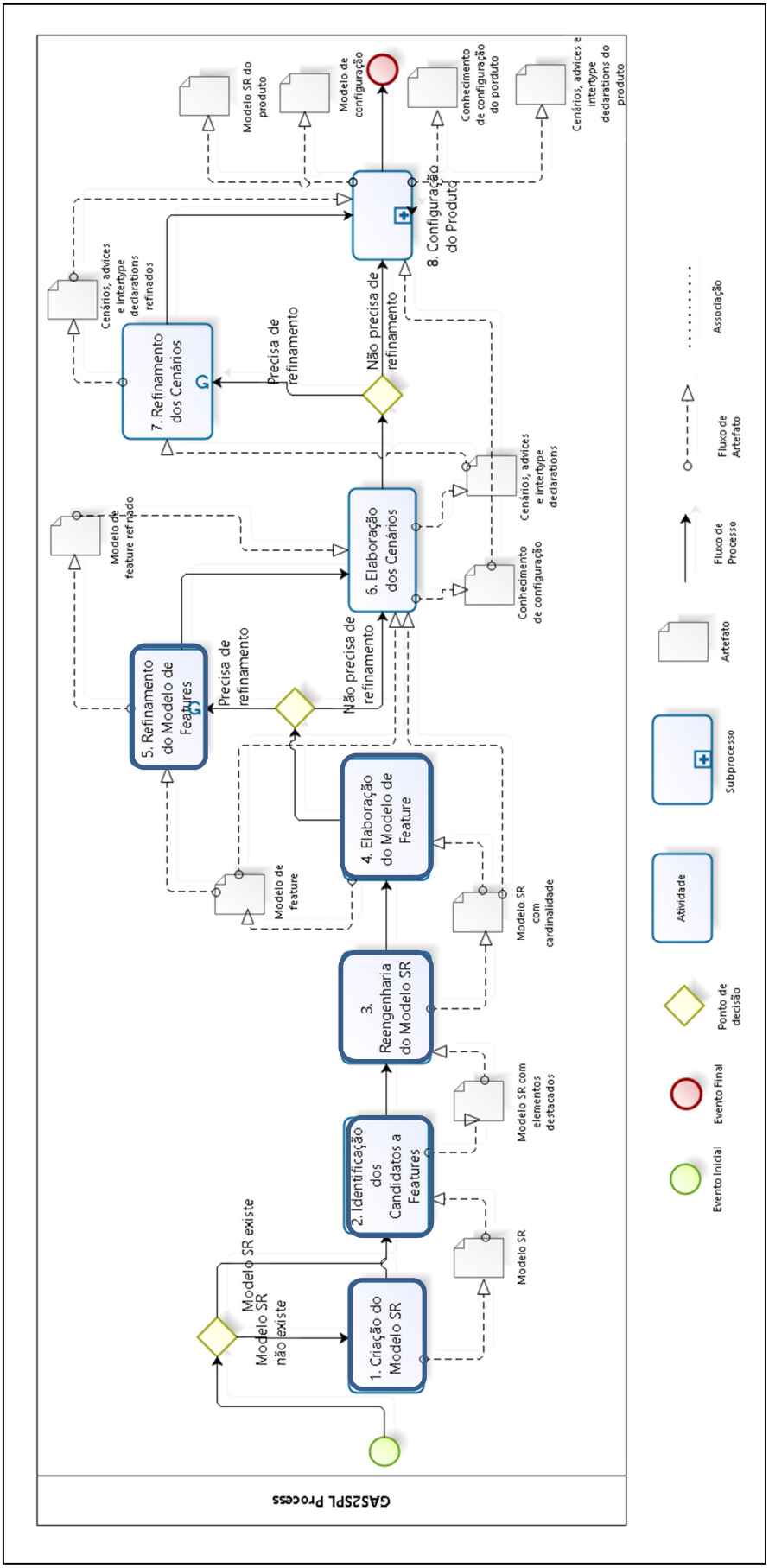
Neste capítulo será apresentado, em detalhes, o processo da abordagem GAS2SPL (*Goals and Aspectual Scenarios to Software Product Lines*). Este processo orienta a criação de cenários aspectuais a partir de modelos *i** com cardinalidade.

3.1. Visão geral da abordagem

A Figura 26 apresenta o modelo do processo da abordagem, elaborado usando a notação BPMN (*Business Process Modeling Notation*)[6] [Pohl e Metzger 2008]. O processo consiste de oito atividades, sendo as sete primeiras, relacionadas a Engenharia de Domínio e, a última, um subprocesso relacionado a Engenharia de Aplicação de LPS. Na Figura 26 as atividades com as bordas destacadas representam as abordagens herdadas do GS2SPL sem modificações, as demais atividades sofreram alterações em relação à GS2SPL para tratar do mapeamento de cenários aspectuais, *advices* e *intertype declarations*, do conhecimento de configuração e das subatividades do processo de configuração do produto.

[6] http://www.omg.org/bpmn/

Figura 26 Modelo de processo da abordagem GAS2SPL



O Quadro 9 apresenta as atividades da abordagem GAS2SPL e a sua classificação em relação ao processo da abordagem GS2SPL: **modificada** (atividades que existiam em GS2SPL, mas foram adaptadas para GAS2SPL), **herdada sem modificação** (atividades que são utilizadas como foram especificadas em GS2SPL) e, **nova** (atividades específicas da abordagem GAS2SPL). As cinco primeiras atividades (Criação do Modelo SR, Identificação dos elementos candidatos a *features*, Reengenharia do Modelo SR, Elaboração do modelo de *features* e Refinamento do Modelo de *features*) são herdadas do GS2SPL sem modificações. Para definir as atividades seguintes utilizaram-se contribuições dos processos do MSVCM, das diretrizes de mapeamento de *i** para cenários de caso de uso de Santander (2002) e do GS2SPL. Nas seções seguintes cada atividade será detalhada utilizando como exemplo a LPS TaRGeT [TARGET 2011].

Quadro 9 Atividades da abordagem GAS2SPL

Atividade	Classificação
Criação do Modelo SR	Herdada sem modificação
Identificação dos Elementos Candidatos a *Features*	Herdada sem modificação
Reengenharia do Modelo SR	Herdada sem modificação
Elaboração do Modelo de *features*	Herdada sem modificação
Refinamento do Modelo de features	Herdada sem modificação
Elaboração dos Cenários	Nova
Refinamento dos Cenários	Modificada
Configuração do Produto	Modificada

A expressão *select scenario* definida por Bachmann e Bass (2001) e utilizada no conhecimento de configuração do MSVCM definido por Almeida (2010) foi adaptada para também tratar da composição de *intertype declarations*, pois o MSVCM teve que ser estendido para tratar de *intertype declarations*.

As seções a seguir apresentam a atividade definida pela abordagem GAS2SPL (Elaboração dos Cenários) e, as atividades que sofreram modificações em relação à abordagem GS2SPL (Refinamento dos Cenários e Configuração dos Produtos).

3.2. Elaboração dos Cenários

A atividade de **elaboração dos cenários** é a principal modificação em relação ao processo da abordagem GS2SPL [Souza 2012]. Ao invés de utilizar a abordagem PLUSS [Eriksson, Börstler e Borg 2005] para especificar os cenários de caso de uso, como em GS2SPL, os cenários serão elaborados com a utilização da técnica MSVCM que realiza a separação de interesses transversais através de cenários aspectuais e *advices* [Almeida 2010].

Esta atividade tem a finalidade de identificar os casos de uso e elaborar os respectivos cenários, *advices* e *intertype declarations*. A atividade de elaboração dos cenários também é responsável por construir o conhecimento de configuração. Este tem como propósito modelar a variabilidade da linha de produto, através da relação das *features* a cenários alternativos, *intertype declarations* e *advices*. O conhecimento de configuração relaciona expressões de *features* a transformações, transformando os artefatos gerados na atividade anterior em artefatos de um produto [Almeida 2010]. Uma expressão de *feature* pode ser entendida como um conjunto de restrições que deve ser satisfeita.

A expressão *select scenario* utilizada no conhecimento de configuração do MSVCM definido por Almeida (2010) foi adaptada para tratar, também, da composição de cenários alternativos e *intertype declarations*. De acordo com Almeida (2010), se o conhecimento de configuração não for especificado de maneira correta, produtos inconsistentes podem ser criados.

Elaboramos algumas diretrizes de mapeamento de elementos do modelo SR para cenários com variabilidade e separação de interesses transversais. Utilizaremos como artefato de entrada o modelo SR com cardinalidade, elaborado na seção 2.2.2 (apresentado na Figura 9), cujos elementos originarão os casos de uso e, posteriormente, os cenários, *advices* e *intertype declarations* descritos com a técnica MSVCM.

As diretrizes definidas por Souza (2012) adaptaram as diretrizes de Santander (2002) para obter cenários com variabilidade especificados em PLUSS a partir de modelos *i**-c. Aqui propomos uma nova adaptação das diretrizes de Souza (2012) e Santander (2002) para mapear modelos *i***-c* para cenários de caso de uso com variabilidade e separação de interesses transversais descritos em MSVCM. É importante destacar que as diretrizes definidas por Santander (2002) tinham como propósito mapear modelos *i** em cenários de casos de uso e não suportam a especificação da variabilidade presente em uma LPS. A técnica escolhida para especificar os cenários de caso de uso no trabalho de Souza (2012), o PLUSS, não apresenta uma separação adequada entre a variabilidade de LPS e o conhecimento de configuração. Como estamos tratando da modularização de *features* de natureza transversal, com a utilização dessa técnica não seríamos capazes de atingir o objetivo desta dissertação.

O Quadro 10 resume as modificações realizadas nas diretrizes desta atividade na abordagem GS2SPL e que sofreram alterações em seu formato original. As diretrizes podem ser classificadas em: mantida (M), removida (R) ou alterada (A).

Quadro 10 Alterações nas diretrizes da atividade de Elaboração dos Cenários de Caso de Uso da abordagem GS2SPL

Diretriz	Descrição	Situação
1 – 4	Identificação dos atores de caso de uso	M
5	Criação do diagrama de caso de uso	R
6	Análise das dependências entre atores	A
7	Associação das *features* a tarefas, recursos e objetivos que as originaram	R
8	Análise dos subcomponentes de uma decomposição de tarefa	A
9	Análise dos sub-elementos na ligação meio-fim (único elemento)	A
10	Análise dos sub-elementos na ligação meio-fim (mais de um elemento)	A
11	Procura de informações adicionais para o caso de uso	A
12	Derivação de novos casos de uso	M

As Diretrizes de **1** a **4** foram mantidas conforme a abordagem de Souza (2012). A Diretriz **5** de Souza (2012), apresentada na seção 2.2.2, que trata da criação do diagrama de caso de uso foi removida, pois para utilizarmos este diagrama teríamos que estender a notação UML para tratar as ligações definidas na abordagem GAS2SPL. Assim, não usaremos esse diagrama.

A diretriz **6** de Souza (2012), apresentada na seção 2.2.2, sofreu pequenas alterações na abordagem GAS2SPL (Diretriz **5**), pois agora também analisa as dependências de um ator como *depender* em relação ao ator que representa o sistema (como na Diretriz 6 de Santander (2002)).

A diretriz **7** de Souza (2012), apresentada na seção 2.2.2, que trata da inserção de referências a *features* nos cenários de caso de uso foi removida, pois no MSVCM existe um artefato que faz o relacionamento entre *features* e os cenários de caso de uso, o conhecimento de configuração.

As antigas Diretrizes **8**, **9**, **10** e **11** de Souza (2012), apresentadas na seção 2.2.2, foram as que sofreram maiores modificações, pois elas agora consideram a especificação de cenários, *advices* e *intertype declarations*. Os passos da Diretriz **12**, apresentada na seção 2.2.2, que tratavam do diagrama de caso de uso foram removidos.

As diretrizes da abordagem GAS2SPL para a atividade de elaboração de cenário são apresentadas a seguir.

Aplicando as quatro primeiras diretrizes de Souza (2012) ao exemplo TaRGeT, obtemos, pela **Diretriz 1**, os atores *Testador*, *Engenheiro de Testes*, *Engenheiro de Requisitos*, *Empresa* e *TaRGeT*. Mas, de acordo com a **Diretriz 2**, o ator deve ser externo ao sistema,

então o ator *TaRGeT* não pode ser mapeado como um ator de caso de uso, restando *Testador*, *Engenheiro de Testes*, *Engenheiro de Requisitos* e *Empresa*. Analisando a **Diretriz 3** percebemos que o ator *Testador* não possui nenhuma dependência com o ator *TaRGeT* e, por isso, ele não pode ser mapeado como um ator de caso de uso. Então, apenas os atores *Engenheiro de Testes*, *Engenheiro de Requisitos* e *Empresa* ainda podem ser mapeados para atores de casos de uso. A **Diretriz 4** exclui aqueles atores que possuem apenas dependência do tipo objetivo-*soft* com o ator *TaRGeT*, com isso o ator *Empresa* não pode ser mapeado como ator de caso de uso. Chegamos ao fim dessa etapa apenas com os atores *Engenheiro de Testes* e *Engenheiro de Requisitos* que atenderam todas as diretrizes e podem ser mapeados como atores de caso de uso.

Diretriz 5: Para cada ator de caso de uso descoberto no passo anterior, deve-se analisar as suas dependências (*dependum*) em relação ao ator sistema (ator -> *dependum* -> sistema) e do sistema com ele (sistema -> *dependum* -> ator), buscando-se descobrir casos de uso para o ator analisado. A fim de organizar os dados extraídos com esta diretriz, as informações relacionadas aos atores, o nome e o tipo de suas dependências, além da diretriz usada para mapear a dependências, deve-se usar o Quadro 11.

Quadro 11 Template para dependência entre atores

Ator	Dependência	Tipo de Dependência	Diretriz a ser usada

Diretriz 5.1: Dependência de objetivo – objetivos em *i** são mapeados para casos de uso.

De acordo com a **Diretriz 5** iremos analisar as dependências entre os atores *Engenheiro de Testes* e *Engenheiro de Requisitos* com o ator *TaRGeT*. Aplicando a Diretriz **5.1** temos as informações contidas no Quadro 12.

Quadro 12 Dependência entre atores do TaRGeT (Diretriz 5.1)

Ator	Dependência	Tipo de Dependência	Diretriz a ser usada
Engenheiro de Teste	Consistência entre Artefatos Mantida	Objetivo	5.1
Engenheiro de Teste	Suíte de Teste Gerada	Objetivo	5.1
Engenheiro de Requisitos	Documento de Requisitos Obtido	Objetivo	5.1

Diretriz 5.2: Dependências de tarefa - neste caso, deve-se investigar se a tarefa precisa ser decomposta em subtarefas. Se a execução da tarefa requer vários passos (posteriormente mapeados em passos de caso de uso), ela pode ser mapeada para um caso de uso. Do contrário, esta dependência representa um passo do caso de uso referente ao elemento interno (tarefa, recurso ou objetivo) ao qual a dependência está ligada.

Esta diretriz não se aplica ao exemplo TaRGeT, pois, o seu modelo SR não apresenta dependências de tarefa entre os atores.

Diretriz 5.3: Dependências de recurso – se um ator depende de outro para obter um recurso, deve-se descobrir se são necessárias várias interações para a obtenção desse recurso. Se for preciso várias interações do ator com o sistema, esta dependência pode ser mapeada para um caso de uso. Do contrário, esta dependência representa um passo do caso de uso referente ao elemento interno (tarefa, recurso ou objetivo) ao qual a dependência está ligada.

Esta diretriz não se aplica ao exemplo TaRGeT, pois o seu modelo SR não apresenta dependências de recurso entre os atores

Diretriz 5.4: Dependências de objetivo-*soft* – tipicamente uma dependência de objetivo-*soft* em *i** representa um requisito não funcional para o sistema. Consequentemente, esta dependência não representa um caso de uso do sistema.

Seguindo a aplicação das diretrizes apresentadas ao TaRGeT, de acordo com a **Diretriz 5.4**, o mapeamento é apresentado a seguir (Quadro 13).

Quadro 13 Dependência entre atores do TaRGeT

Ator	Dependência	Tipo de Dependência	Diretriz a ser usada
Engenheiro de Teste	Consistência entre Artefatos Mantida	Objetivo	5.1
Engenheiro de Teste	Otimização [Suíte de Teste]	Objetivo-*soft*	5.4
Engenheiro de Teste	Suíte de Teste Gerada	Objetivo	5.1
Engenheiro de Teste	Qualidade [Artefatos]	Objetivo-*soft*	5.4
Engenheiro de Requisitos	Qualidade [Artefatos]	Objetivo-*soft*	5.4
Engenheiro de Requisitos	Documento de Requisitos Obtido	Objetivo	5.1

Analisando o Quadro 13 e sabendo que objetivos-*soft* representam os requisitos não funcionais e, portanto, não originam casos de uso. Os casos de uso gerados nessa atividade são: *Consistência entre Artefatos Mantida*, *Suíte de Testes Gerada* e *Documento de Requisitos Obtido*.

Diretriz 5.5: O caso de uso deve ser identificado por um número único seguindo, preferencialmente, o formato *UCnúmeroDoCasoDeUso* e o seu nome. O nome do caso de uso (<*nomeDoCasoDeUso*>), como sugere Cockburn (2000), pode ser um objetivo descrito com uma frase curta contendo um verbo na voz ativa. Os cenários devem seguir o *template* do Quadro 14.

Quadro 14 Template para a elaboração do cenário

<*UCnúmeroDoCasoDeUso*> <*nomeDoCasoDeUso*> <*Ator*> <*siglaTipoCenário.númeroDoCenário*> <*Descrição*>		
ID	**Ação do Usuário**	**Resposta do Sistema**
<*siglaTipoCenário.númeroDoCenário.númeroDoPasso*>		

Cada cenário deve possuir a identificação do caso de uso que está associado através das referências *UCnúmeroDoCasoDeUso* e *nomeDoCasoDeUso*. O *Ator* faz referência aos atores que podem acessar este caso de uso. O *TipoDoCenário* é utilizado para indicar se o cenário está representando o Cenário Principal ou um Cenário Secundário, que inclui alternativo e excepcional. Exceto para o caso do cenário principal, que é único, os demais cenários devem possuir um identificador único e sugere-se que siga o formato *siglaTipoCenário.númeroDoCenário*, por exemplo, CE.01 (cenário excepcional 01) ou CA.02 (cenário alternativo 02). Para o cenário principal na sigla deve haver apenas *siglaTipoCenário*, por exemplo, CP (cenário principal). Além desse identificador, um cenário deve conter uma *Descrição*, que pode ser o nome do cenário ou alguma informação pertinente sobre o objetivo do cenário. Os seus passos devem ser identificados por um número único, seguindo o formato *siglaTipoCenário.númeroDoCenário.númeroDoPasso.* Se o cenário for o principal, o formato a ser seguido para identificar os passos é *CP.númeroDoPasso.*

Aplicando esta diretriz aos casos de uso identificados na diretriz **5** para o TaRGeT, os cenários dos casos de uso gerados para *Suíte de Teste* Gerada, *Documento de Requisito* Obtido, *Consistência entre Artefatos Mantida* são apresentados nas figuras Figura 27, Figura 28 e Figura 29, respectivamente.

Figura 27 Caso de uso Suíte de Teste Gerada

UC01 **Suíte de Teste Gerada** **Ator:** Engenheiro de Teste **Cenário Principal 01** **Descrição:** Gerar suíte de teste		
ID	**Ação do Usuário**	**Resposta do Sistema**

Figura 28 Caso de uso Documento de Requisitos Obtido

UC02 **Documento de Requisitos Obtido** **Ator:** Engenheiro de Requisitos **Cenário Principal 01** **Descrição:** Obter documento de requisitos		
ID	**Ação do Usuário**	**Resposta do Sistema**

Figura 29 Caso de uso Consistência entre Artefatos Mantida

UC03 **Consistência entre artefatos mantida** **Ator:** Engenheiro de Teste **Cenário Principal 01** **Descrição:** Manter consistência entre artefatos		
ID	**Ação do Usuário**	**Resposta do Sistema**

Diretriz 5.6: O caso de uso derivado do objetivo principal do modelo SR será mapeado, no conhecimento de configuração, através da transformação *select scenario*, pois este é um caso de uso obrigatório. Os demais casos de uso não serão mapeados no conhecimento de configuração, pois dependem da existência de seus elementos pais para existirem. A transformação deve seguir o *template select scenario* < UC*númeroDoCasoDeUso.identificadorDoCenárioPrincipal*>.

Diretriz 5.7: O conhecimento de configuração deve seguir o *template* do Quadro 15.

Quadro 15 Template para o conhecimento de configuração

Expressão de Feature	Transformações

Na coluna *Expressão de Feature* devem estar todas as *features* e, na coluna *Transformações* as expressões *select scenario*, *evaluate advice* e *bind parameter* e seus respectivos cenários e *intertype declarations*, *advices* e parâmetros associados.

Aplicando as diretrizes **5.6** e **5.7 a**o *TaRGeT*, obtemos o seguinte conhecimento de configuração (Quadro 16).

Quadro 16 V1 do Conhecimento de Configuração do TaRGeT

Expressão de Feature	Transformação
Suíte de Teste Gerada	*select scenario* UC01.CP

Diretriz 6: Analisar o ator que representa o sistema para extrair informações que possam gerar mais casos de uso, *advices* e *intertype declarations* para a LPS.

Diretriz 6.1: Cada objetivo interno ao ator sistema, que não possuir ligação direta com uma dependência, será mapeado para um caso de uso do sistema.

Aplicando esta diretriz ao TaRGeT, obtêm-se os casos de uso *Documento Verificado e Filtro Selecionado* apresentados na Figura 30 e

Figura 31, respectivamente.

Figura 30 Caso de uso Documento Verificado

UC04
Documento Verificado
Ator: Engenheiro de Teste
Cenário Principal 01
Descrição: Verificação de documento

ID	Ação do Usuário	Resposta do Sistema

Figura 31 Caso de uso Filtro Selecionado

UC05
Filtro Selecionado
Ator: Engenheiro de Teste
Cenário Principal 01
Descrição: Seleção de Filtros

ID	Ação do Usuário	Resposta do Sistema

Diretriz 6.2: Elementos com cardinalidade [1..1] que são sub-elementos de um elemento com cardinalidade [1..1], representam a parte comum da LPS e podem ser capturados como o cenário principal ou cenários alternativos do caso de uso relacionado ao elemento raiz (daqui em diante chamado de caso de uso base).

Diretriz 6.2.1: Os sub-elementos com cardinalidade [1..1] de uma decomposição de tarefa serão mapeados para passos do cenário principal do caso de uso base.

Diretriz 6.2.2: Se uma ligação meio-fim tiver mais de um sub-elemento e a cardinalidade estiver nos sub-elementos, cada sub-elemento com cardinalidade [1..1] será mapeado para um cenário alternativo do caso de uso base.

Diretriz 6.2.3: Se uma ligação meio-fim tiver apenas um sub-elemento e esse sub-elemento tiver cardinalidade [1..1], ele será mapeado para o cenário principal do caso de uso base.

Essas diretrizes não se aplicam ao exemplo TaRGeT, pois o seu modelo SR não apresenta elemento com cardinalidade [1..1] que seja sub-elemento de outro elemento com cardinalidade [1..1].

Diretriz 6.3: Os sub-elementos com cardinalidade [0..1] ou envolvidos em relacionamentos meio-fim com cardinalidade <*i..j*> representam a variabilidade da LPS e podem ser capturados como *advices* ou *intertype declarations.* O *intertype declaration* cria cenários alternativos e referencia-os no passo do cenário principal do caso de uso base onde eles devem acontecer representando o objetivo principal do relacionamento meio-fim (daqui em diante chamado de caso de uso base). Um *advice* adiciona passos aos cenários do caso de uso base.

Diretriz 6.3.1: Se a cardinalidade estiver na ligação meio-fim, os sub-elementos serão mapeados para *intertype declarations*, responsáveis por criar pelo menos *i* e no máximo *j* cenários alternativos no caso de uso base, respeitando a cardinalidade <*i*, *j*> definida na ligação meio-fim.

Diretriz 6.3.2: Um cenário alternativo criado com o *intertype declaration* deve seguir o *template* do Quadro 17.

Quadro 17 Template para elaboração do intertype declaration

<ITDnúmeroDoIntertypeDeclaration> *<Descrição>*		
ID	**Ação do Usuário**	**Resposta do Sistema**
<ITDnúmeroDoIntertypeDeclaration.númeroDoPasso>		

O *intertype declaration* deve ter um identificador único e sugere-se que siga o formato *ITDnúmeroDoIntertypeDeclaration*. Além do identificador, o *template* deve conter uma *Descrição*, que pode ser o objetivo do cenário. Os passos do *intertype declaration* devem ser identificados por um número único, seguindo o formato *ITDnúmeroDoIntertypeDeclaration.númeroDoPasso*. A criação do cenário alternativo será realizada através da inclusão da anotação *@criar* no passo do cenário principal e o identificador do *intertype declaration*. A inclusão é feita na Engenharia de Domínio, antes da configuração de um produto da LPS.

Aplicando as diretrizes 6.3.1 e 6.3.2 ao TaRGeT, os *intertype declarations* gerados são apresentados na Figura 32 e Figura 33.

Figura 32 *Intertype declaration* Gerar Suíte de Teste Detalhada

ITD01 **Descrição:** Gerar suíte de teste detalhada		
ID	**Ação do Usuário**	**Resposta do Sistema**
ITD01.01		

Figura 33 *Intertype declaration* Gerar Suíte de Teste Diretamente

ITD02 **Descrição:** Gerar suíte de teste diretamente		
ID	**Ação do Usuário**	**Resposta do Sistema**
ITD02.01		

Diretriz 6.3.3: O *intertype declaration* deve ser mapeado, no conhecimento de configuração, através da aplicação da transformação ***select scenario***, adaptada de Bachmann e Bass (2001). A transformação para o *intertype declaration* deve seguir o *template select scenario* <ITD*númeroDoITD.*UC*númeroDoCasoDeUso.identificadorDoCenárioPrincipal*>.

Aplicando as diretrizes **6.3.3** e **5.7** ao *TaRGeT*, obtemos o seguinte conhecimento de configuração (Quadro 18).

Quadro 18 V2 do Conhecimento de Configuração do TaRGeT

Expressão de Feature	Transformação
Suíte de Teste Gerada	*select scenario* UC01.CP
Gerar Suíte de Teste Detalhada	*select scenario* ITD01.UC01.CP
Gerar Suíte de Teste Diretamente	*select scenario* ITD02.UC01.CP

Diretriz 6.3.4: Se uma ligação meio-fim tiver mais de um sub-elemento e a cardinalidade estiver nos sub-elementos, cada sub-elemento com cardinalidade [0..1] será mapeado para um *intertype declaration*.

Aplicando as diretrizes 6.3.2 e 6.3.4 ao TaRGeT, os *intertype declarations* gerados são apresentados na Figura 34, Figura 35 e Figura 36.

Figura 34 *Intertype declaration* Escrever no Editor

ITD03 **Descrição:** Escrever no editor		
ID	**Ação do Usuário**	**Resposta do Sistema**
ITD03.01		

Figura 35 *Intertype declaration* Parametrizar Teste

ITD04 **Descrição:** Parametrizar teste		
ID	**Ação do Usuário**	**Resposta do Sistema**
ITD04.01		

Figura 36 *Intertype declaration* Checar Caso de Uso Semanticamente

ITD05 **Descrição:** Checar caso de uso semanticamente		
ID	**Ação do Usuário**	**Resposta do Sistema**
ITD05.01		

Aplicando as diretrizes **6.3.3** e **5.7 a**o *TaRGeT*, obtemos o seguinte conhecimento de configuração (Quadro 19).

Quadro 19 V3 do Conhecimento de Configuração do TaRGeT

Expressão de Feature	Transformação
Suíte de Teste Gerada	*select scenario* UC01.CP
Gerar Suíte de Teste Detalhada	*select scenario* ITD01 UC01.CP
Gerar Suíte de Teste Diretamente	*select scenario* ITD02 UC01.CP
Escrever no Editor da Ferramenta	*select scenario* ITD03.UC02.CP
Parametrizar Teste	*select scenario* ITD04.UC05.CP
Checar Caso de Teste Semanticamente	*select scenario* ITD05 UC04.CP

Diretriz 6.3.5: Se uma ligação meio-fim tiver apenas um sub-elemento e esse sub-elemento tiver cardinalidade [0..1], ele será mapeado para um *advice*.

Diretriz 6.3.6: Cada sub-elemento com cardinalidade [0..1] de uma ligação de decomposição de tarefa será mapeado para um *advice* que irá acrescentar passos ao cenário principal do caso de uso base.

Esta diretriz não se aplica ao exemplo TaRGeT, pois o seu modelo SR não apresenta sub-elementos com cardinalidade [0..1] de uma decomposição de tarefas.

Diretriz 6.3.7: A sintaxe do *advice* é bastante similar a utilizada para descrever um *intertype declaration*, mas no *template* a ser seguido são introduzidos novos campos para identificar o tipo do *advice* e os *pointcuts*. Os *advices* devem seguir o *template* do Quadro 20.

Quadro 20 *Template* para elaboração do advice

<ADVnúmeroDoAdvice> *<Descrição>* *<TipoDoAdvice> <Pointcut>*		
ID	**Ação do Usuário**	**Resposta do Sistema**
<ADVnúmeroDoAdvice.númeroDoPasso>		

O *advice* deve ter um identificador único e sugere-se que siga o formato *ADVnúmeroDoAdvice*. Além do identificador, um *advice* deve conter uma *Descrição*, que pode ser alguma informação que explique o seu objetivo. Um *advice* varia conforme o momento de sua execução, portanto, o *TipodoAdvice* pode ter um dos modificadores *before*, *after* ou *around* [Bodkin e Laddad 2004]. Além do modificador, o *pointcut* deve possuir um nome seguindo o formato *@ADVnúmeroDoAdvice*. O nome do *pointcut* deverá ser o identificador do *advice*, pois é um identificador único e assim, evitará duplicidade de referências. No cenário do caso de uso base associado ao *advice* deve haver uma anotação que corresponda ao mesmo nome

do *pointcut,* seguindo o formato @*ADVnúmeroDoAdvice*, representando um *joinpoint*. Portanto, o advice poderá ser adicionado antes (*before*), depois (*after*) ou substituindo (*around)* o joinpoint indicado no caso de uso base. Os passos do *advice* devem ser identificados por um número único seguindo o formato *ADVnúmeroDoAdvice.númeroDoPasso*.

Aplicando as diretrizes **6.3.5** e **6.3.7** ao TaRGeT, o *advice* gerado é apresentado na Figura 37.

Figura 37 *Advice* do caso de uso Consistência entre Artefatos Mantida

ADV01 **Descrição:** Manter consistência entre os artefatos **After @*ADV01***		
ID	**Ação do Usuário**	**Resposta do Sistema**
ADV01.01		

Diretriz 6.3.8: O *advice* deve ser mapeado, no conhecimento de configuração, através da transformação ***evaluate advice***, definida por Bachmann e Bass (2001). Esta transformação faz a composição de *advices* a *join points* específicos e, trata da variabilidade em controle de fluxo e é utilizada para modularizar passos opcionais e alternativos que estão entrelaçados com especificações comuns de um cenário [Bachmann e Bass 2001]. A transformação para o *advice* deve seguir o *template evaluate advice* <ADV*númeroDoADV*>.

Aplicando as diretrizes **5.7** e **6.3.8** ao *TaRGeT*, obtemos o seguinte conhecimento de configuração (Quadro 21).

Quadro 21 V4 do Conhecimento de Configuração do TaRGeT

Expressão de Feature	**Transformação**
Suíte de Teste Gerada	*select scenario* UC01.CP
Gerar Suíte de Teste Detalhada	*select scenario* ITD01 UC01.CP
Gerar Suíte de Teste Diretamente	*select scenario* ITD02 UC01.CP
Checar Caso de Teste Semanticamente	*select scenario* ITD05 UC04.CP
Parametrizar Teste	*select scenario* ITD04.UC05.CP
Escrever no Editor da Ferramenta	*select scenario* ITD03.UC02.CP
Manter Consistência entre os Artefatos	*evaluate advice* ADV01

Diretriz 6.3.9: Um cenário pode conter parâmetros, que também é um ponto de variação. Um parâmetro ocorre quando dois ou mais cenários compartilham a mesma sequência de passos e diferem apenas em relação às variáveis manipuladas. O parâmetro é adicionado a um passo do cenário principal do caso de uso base e deve seguir o formato

<<nomeDoParametro>>.

Essa diretriz não se aplica ao exemplo TaRGeT, pois o seu modelo SR não apresenta cenários que compartilham a mesma sequência de passos e utilizam variáveis diferentes.

Diretriz 6.3.10: Caso haja algum parâmetro nos cenários a sua associação deve ser feita através de ***bind parameter***, definida por Bachmann e Bass (2001). Esta transformação ocorre sempre que dois ou mais cenários compartilham o mesmo comportamento (a sequência de passos) e difere apenas em relação a valores de um mesmo conceito [Bachmann e Bass 2001]. Ela é responsável por ligar parâmetros a *features*. A transformação utilizada para parâmetros deve seguir o *template bind parameter <siglaDoParâmetro><nomeDoParâmetro>*.

Esta diretriz não se aplica ao *TaRGeT*, pois não foram mapeados parâmetros na diretriz anterior.

Diretriz 6.4: Elementos com cardinalidade [1..1] que não são sub-elementos de elementos com cardinalidade [1..1] também representam a variabilidade da LPS e podem ser mapeados para *advices*, *intertype declarations* ou cenários de casos de uso opcionais.

Diretriz 6.4.1: Os sub-elementos com cardinalidade [1..1] de uma decomposição de uma tarefa opcional (com cardinalidade [0..1] ou alternativa de uma ligação meio-fim com cardinalidade) serão mapeados para passos de um *advice* ou um *intertype declaration* obtidos a partir da tarefa raiz. Para definir se a tarefa raiz é um cenário alternativo, um *advice* ou *intertype declaration*, deve-se aplicar as sub-diretrizes da diretriz 6.3.

Aplicando a diretriz 6.4.1 ao TaRGeT, obtêm-se uma nova versão de alguns cenários previamente identificados, conforme apresentado na Figura 38, Figura 39 e Figura 40.

Figura 38 *Advice* do caso de uso Consistência entre Artefatos Mantida

ADV01 **Descrição:** Manter consistência entre os artefatos **After @*ADV01***		
ID	**Ação do Usuário**	**Resposta do Sistema**
ADV01.01	Detectar mudança nos requisitos	
ADV01.02	Atualizar casos de teste	

Figura 39 Versão 2 do *ITD* Gerar Suíte de Teste Detalhada

ITD01 **Descrição:** Gerar suíte de teste detalhada		
ID	**Ação do Usuário**	**Resposta do Sistema**
ITD01.01	Gerar casos de teste específico	

Figura 40 Versão 2 do *ITD* Gerar Suíte de Teste Diretamente

ITD02 **Descrição:** Gerar suíte de teste diretamente		
ID	**Ação do Usuário**	**Resposta do Sistema**
ITD02.01	Gerar todos os casos de teste	

Diretriz 6.4.2: Se uma ligação meio-fim tiver mais de um sub-elemento e a cardinalidade estiver nos sub-elementos, cada sub-elemento com cardinalidade [1..1] será mapeado para um cenário alternativo do caso de uso base. A descrição do cenário alternativo deve conter o nome do elemento que o originou. O caso de uso base é opcional, porque o elemento que o originou não é sub-elemento de um elemento com cardinalidade [1..1] (não é filho de um elemento obrigatório da LPS). Para cada cenário alternativo criado, deve-se atualizar o cenário principal do caso de uso base, indicando o passo onde o cenário alternativo pode ocorrer. A indicação deve seguir o formato [CA.<númeroDoCenário>].

Aplicando a diretriz 6.4.2 ao TaRGeT, os cenários alternativos gerados são apresentados nas Figuras 41-51.

Figura 41 Cenário alternativo Filtro por Propósito de Teste

UC05 Filtro Selecionado **Ator:** Engenheiro de Teste **Cenário Alternativo 01** **Descrição:** Filtrar por propósito de teste		
ID	**Ação do Usuário**	**Resposta do Sistema**
CA01.01		

Figura 42 Cenário alternativo Incluir Caso de Teste Permanentemente

UC05 Filtro Selecionado **Ator:** Engenheiro de Teste **Cenário Alternativo 02** **Descrição:** Incluir caso de teste permanentemente		
ID	**Ação do Usuário**	**Resposta do Sistema**
CA02.01		

Figura 43 Cenário alternativo Excluir Caso de Teste Permanentemente

UC05 Filtro Selecionado **Ator:** Engenheiro de Teste **Cenário Alternativo 03** **Descrição:** Excluir caso de teste permanentemente		
ID	**Ação do Usuário**	**Resposta do Sistema**
CA03.01		

Figura 44 Cenário alternativo Filtro por Caso de Uso Similar

UC05 Filtro Selecionado **Ator:** Engenheiro de Teste **Cenário Alternativo 04** **Descrição:** Filtrar por caso de uso similar		
ID	**Ação do Usuário**	**Resposta do Sistema**
CA04.01		

Figura 45 Cenário alternativo Filtro por Requisito

UC05 Filtro Selecionado **Ator:** Engenheiro de Teste **Cenário Alternativo 05** **Descrição:** Filtrar por requisito		
ID	**Ação do Usuário**	**Resposta do Sistema**
CA05.01		

Figura 46 Cenário alternativo Filtro por Requisito

UC05 Filtro Selecionado **Ator:** Engenheiro de Teste **Cenário Alternativo 06** **Descrição:** Filtrar por caso de uso		
ID	**Ação do Usuário**	**Resposta do Sistema**
CA06.01		

Figura 47 Cenário alternativo Checar Caso de Teste Sintaticamente

UC04 Documento Verificado **Ator:** Engenheiro de Teste **Cenário Alternativo 01** **Descrição:** Checar Caso de teste sintaticamente		
ID	**Ação do Usuário**	**Resposta do Sistema**
CA01.01		

Figura 48 Cenário alternativo Fazer *Upload* de Documento

UC02 Documento de Requisitos Obtido **Ator:** Engenheiro de Teste **Cenário Alternativo 01** **Descrição:** *Upload* de Documento		
ID	**Ação do Usuário**	**Resposta do Sistema**
CA01.01		

Figura 49 Versão 2 do caso de uso Filtro Selecionado

UC05 **Filtro Selecionado** **Ator:** Engenheiro de Teste **Cenário Principal** **Descrição:** Seleção de Filtros		
ID	**Ação do Usuário**	**Resposta do Sistema**
CP.01	[CA.01][CA.02][CA.03][CA.04][CA.05][CA.06]	

Figura 50 Versão 2 do caso de uso Documento Verificado

UC04 **Documento Verificado** **Ator:** Engenheiro de Teste **Cenário Principal** **Descrição:** Verificação de documento		
ID	**Ação do Usuário**	**Resposta do Sistema**
CP.01	[CA.01]	

Figura 51 Versão 2 do caso de uso Documento de Requisitos Obtido

UC02 **Documento de Requisitos Obtido** **Ator:** Engenheiro de Requisitos **Cenário Principal** **Descrição:** Obter documento de requisitos		
ID	**Ação do Usuário**	**Resposta do Sistema**
CP.01	[CA.01]	

Diretriz 6.4.3: Se uma ligação meio-fim tiver apenas um sub-elemento e esse sub-elemento tiver cardinalidade [1..1], ele será mapeado para o cenário principal do caso de uso base. Nesse caso, temos um caso de uso opcional, visto que o elemento que o originou não é sub-elemento de um elemento com cardinalidade [1..1] (não é filho de um elemento obrigatório da linha).

Essa diretriz não se aplica ao exemplo TaRGeT, pois o seu modelo SR não apresenta um único sub-elemento com cardinalidade [1..1] de uma ligação meio-fim.

Diretriz 6.4.4: Se a cardinalidade estiver na ligação meio-fim, os sub-elementos serão mapeados para *intertype declarations* e irão criar *j* cenários alternativos no caso de uso base, respeitando a cardinalidade <*i*, *j*> definida na ligação meio-fim. Nesse caso, temos um caso de uso opcional, visto que o elemento que o originou não é sub-elemento de um elemento com cardinalidade [1..1] (não é filho de um elemento obrigatório da LPS).

Essa diretriz não se aplica ao exemplo TaRGeT, pois o seu modelo SR não apresenta uma ligação meio-fim com cardinalidade cujo elemento raiz seja filho de elemento opcional da

linha de produto.

Diretriz 6.4.5: Os cenários alternativos, de acordo com Almeida (2010) podem ser mapeados, no conhecimento de configuração, através da aplicação da transformação ***select scenario***, adaptada de Bachmann e Bass (2001). A transformação para os cenários alternativos devem seguir o *template select scenario* <CA*númeroDoCA*.UC*númeroDoCasoDeUso.CP*>.

Aplicando as diretrizes **6.4.5** e **5.7 a**o *TaRGeT*, obtemos o seguinte conhecimento de configuração (Quadro 22).

Quadro 22 V5 do Conhecimento de Configuração do TaRGeT

Expressão de Feature	Transformação
Suíte de Teste Gerada	*select scenario* UC01.CP
Gerar Suíte de Teste Detalhada	*select scenario* ITD01 UC01.CP
Gerar Suíte de Teste Diretamente	*select scenario* ITD02 UC01.CP
Checar Caso de Teste Semanticamente	*select scenario* ITD05 UC04.CP
Parametrizar Teste	*select scenario* ITD04.UC05.CP
Escrever no Editor da Ferramenta	*select scenario* ITD03.UC02.CP
Manter Consistência entre os Artefatos	*evaluate advice* ADV01
Fazer *Upload* de Documento	*select scenario* CA01.UC02.CP
Checar Caso de Teste Permanentemente	*select scenario* CA01.UC04.CP
Filtrar por Propósito de Teste	*select scenario* CA01.UC05.CP
Incluir Caso de Teste Permanentemente	*select scenario* CA02.UC05.CP
Excluir Caso de Teste Permanentemente	*select scenario* CA03.UC05.CP
Filtrar por Caso de Uso Similar	*select scenario* CA04.UC05.CP
Filtrar por Requisito	*select scenario* CA05.UC05.CP
Filtrar por Caso de Uso	*select scenario* CA06.UC05.CP

Apenas o cenário de caso de uso UC01 é obrigatório, os demais cenários de caso de uso (UC02, UC03, UC04 e UC05) são opcionais e, podem estar presentes no conhecimento de configuração do produto, caso uma de suas sub-*features* seja selecionada na atividade de configuração do produto.

As diretrizes seguintes se encarregam de analisar outros relacionamentos internos não considerados nas diretrizes anteriores.

Diretriz 6.5: Analisar os sub-elementos de uma decomposição de tarefa que não são objetivos. Estes objetivos são casos de uso que devem ser incluídos em outros casos de uso utilizando a anotação @incluir que tem a mesma semântica do relacionamento <<*include*>> do caso de uso UML.

Diretriz 6.5.1: Se uma tarefa possuir um sub-elemento do tipo objetivo, o caso de uso correspondente a esse objetivo será incluído através da anotação @incluir <*UCnúmeroDoCasoDeUso*> <*nomeDoCasoDeUso*> no cenário associado a tarefa que foi decomposta. Essa tarefa pode ser um passo do cenário de caso de uso, pode ser um ITD ou um advice, mapeamentos obtidos pela aplicação das diretrizes anteriores. O <*UCnúmeroDoCasoDeUso*> <*nomeDoCasoDeUso*> representa o caso de uso que será incluído.

Aplicando a diretriz 6.5.1 ao TaRGeT, obtêm-se uma nova versão de alguns cenários previamente identificados, conforme apresentado na Figura 52 e

Figura 53.

Figura 52 Versão 3 do *ITD* Gerar Suíte de Teste Detalhada

ITD01 **Descrição:** Gerar suíte de teste detalhada		
ID	**Ação do Usuário**	**Resposta do Sistema**
ITD01.01	@incluir UC02 Documento de Requisitos Obtido	
ITD01.02	@incluir UC04 Documento Verificado	
ITD01.03	@incluir UC05 Filtro Selecionado	
ITD01.04	Gerar casos de teste específico	
ITD01.05	@incluir UC03 Consistência entre artefatos mantida	

Figura 53 Versão 3 do *ITD* Gerar Suíte de Teste Diretamente

ITD02 **Descrição:** Gerar suíte de teste diretamente		
ID	**Ação do Usuário**	**Resposta do Sistema**
ITD02.01	@incluir UC02 Documento de Requisitos Obtido	
ITD02.02	@incluir UC04 Documento Verificado	
ITD02.02	Gerar todos os casos de teste	

A ordem dos passos é decidida pelo analista, de acordo com o conhecimento que ele tem sobre o problema/solução que está sendo especificado.

Os cenários, *advices* e *intertype declarations* gerados através da utilização das diretrizes apresentadas são incompletos, isto ocorre devido o modelo de objetivo possuir um nível de abstração alto e não detalhar o passo a passo da realização de algumas tarefas.

À medida que se tenta aumentar o detalhamento dos modelos *i**, pode-se aumentar a complexidade deste modelo, prejudicando a legibilidade.

A próxima atividade é responsável por preencher estas lacunas.

3.3. Refinamento dos Cenários

Esta atividade tem o propósito de preencher as lacunas que ficaram nos cenários, *advices* e *intertype declarations* após a execução da atividade anterior. Isto ocorre, pois, segundo Souza (2012), os modelos de objetivos não são suficientes para capturar aspectos comportamentais. A atividade de refinar os cenários de caso de uso pode ser executada várias vezes pelo engenheiro de domínio até que este fique satisfeito com as descrições obtidas.

O engenheiro de domínio pode analisar novamente todos os artefatos que foram utilizados para criar o modelo SR (especificação de requisitos, reuniões com interessados, entre outros), bem como realizar novas entrevistas ou utilizar outras técnicas de elicitação de requisitos com os interessados.

Analisando os artefatos gerados na atividade anterior percebemos que existem lacunas na *Resposta do Sistema*, na descrição de passos dos *advices* e *intertype declarations* e, nos passos de cenários alternativos. Assim como na atividade de Souza (2012) não foram definidas heurísticas, mas, de acordo com Souza (2012), deve-se tentar responder algumas perguntas ao analisar cada caso de uso. Estas perguntas são:

- Quais as possíveis exceções ou falhas que podem ocorrer durante a execução desse passo?
- Se houver algum tipo de exceção, qual é a resposta do sistema nesta situação?
- Este passo pode ser refinado em outros? Quais?
- Existe algum caminho alternativo aos passos do cenário principal que o ator possa realizar e que não represente uma exceção?

Para respondê-las e elaborar os cenários, *advices* e *intertype declarations* mais completos para o TaRGeT, foram analisadas as informações disponíveis em Souza (2012) e TARGET (2011). Com isso, preenchemos as lacunas relacionadas aos cenários e as respostas fornecidas pelo sistema. As Figuras 54-72 apresentam os cenários, *advices* e *intertype declarations* após a execução desta atividade.

Figura 54 Caso de uso Suíte de Teste Gerada

UC01 **Suíte de Teste Gerada** **Ator:** Engenheiro de Teste **Cenário Principal 01** **Descrição:** Gerar suíte de teste		
ID	**Ação do Usuário**	**Resposta do Sistema**
CP01		

Figura 55 Caso de uso Documento de Requisitos Obtido

UC02 **Documento de Requisitos Obtido** **Ator:** Engenheiro de Requisitos **Cenário Principal 01** **Descrição:** Obter documento de requisitos		
ID	**Ação do Usuário**	**Resposta do Sistema**
CP.01	[CA.01]	Documento carregado

Figura 56 Caso de uso Consistência entre Artefatos Mantida

UC03 **Consistência entre artefatos mantida** **Ator:** Engenheiro de Teste **Cenário Principal 01** **Descrição:** Manter consistência entre artefatos		
ID	**Ação do Usuário**	**Resposta do Sistema**
CP01		

Figura 57 Caso de uso Documento Verificado

UC04 **Documento Verificado** **Ator:** Engenheiro de Teste **Cenário Principal** **Descrição:** Verificação de documento		
ID	**Ação do Usuário**	**Resposta do Sistema**
CP.01	[CA.01]	Caso de teste verificado sintaticamente

Figura 58 Caso de uso Filtro Selecionado

UC05 **Filtro Selecionado** **Ator:** Engenheiro de Teste **Cenário Principal** **Descrição:** Seleção de Filtros		
ID	**Ação do Usuário**	**Resposta do Sistema**
CP.01	[CA.01][CA.02][CA.03][CA.04][CA.05][CA.06]	Filtros selecionados

Figura 59 *Intertype Declarations* Gerar Suíte de Teste Detalhada

ITD01 **Descrição:** Gerar suíte de teste detalhada		
ID	**Ação do Usuário**	**Resposta do Sistema**
ITD01.01	@incluir UC02 Documento de Requisitos Obtido	Obtém documento de requisitos
ITD01.02	@incluir UC04 Documento Verificado	Verifica documento de requisito
ITD01.03	@incluir UC05 Filtro Selecionado	Seleciona filtro
ITD01.04	Gerar casos de teste específico	Gerar casos de teste para casos de uso específicos do documento de requisitos
ITD01.05	@incluir UC03 Consistência entre artefatos mantida	Mantém a consistência entre os artefatos

Figura 60 *Intertype Declarations* Gerar Suíte de Teste Diretamente

ITD02 **Descrição:** Gerar suíte de teste diretamente		
ID	**Ação do Usuário**	**Resposta do Sistema**
ITD02.01	@incluir UC02 Documento de Requisitos Obtido	Obtém documento de requisitos
ITD02.02	@incluir UC04 Documento Verificado	Verifica documento de requisito
ITD02.02	Gerar todos os casos de teste	Gerar casos de teste para todos os casos de uso do documento de requisitos

Figura 61 *Intertype declaration* Escrever no Editor

ITD03 **Descrição:** Escrever no editor		
ID	**Ação do Usuário**	**Resposta do Sistema**
ITD03.01	Seleciona opção para abrir o editor de caso de uso da ferramenta	Abre janela do editor de caso de uso
ITD03.02	Escrever no editor da ferramenta	-
ITD03.03	Seleciona a opção de salvar o documento	Documento de requisito salvo

Figura 62 *Intertype declaration* Parametrizar Teste

ITD04 **Descrição:** Parametrizar teste		
ID	**Ação do Usuário**	**Resposta do Sistema**
ITD04.01	**-**	Inclui parâmetros na suíte de testes

Figura 63 *Intertype declaration* Checar Caso de Uso Semanticamente

ITD05 **Descrição:** Checar caso de teste semanticamente		
ID	**Ação do Usuário**	**Resposta do Sistema**
ITD05.01	**-**	Verifica a descrição dos casos de uso contém palavras que possam gerar ambiguidade (dicionário interno)

Figura 64 *Advice* do caso de uso Consistência entre Artefatos Mantida

ADV01 **Descrição:** Manter consistência entre os artefatos **After *@ADV01***		
ID	**Ação do Usuário**	**Resposta do Sistema**
ADV01.01	Detectar mudança nos requisitos	Detecta que houve mudança nos requisitos em relação à versão anterior do documento
ADV01.02	Atualizar casos de teste	Atualiza casos de teste relacionados aos casos de uso modificados

Figura 65 Cenário alternativo Filtro por Propósito de Teste

UC05 Filtro Selecionado **Ator:** Engenheiro de Teste **Cenário Alternativo 01** **Descrição:** Filtrar por propósito de teste		
ID	**Ação do Usuário**	**Resposta do Sistema**
CA01.01	-	Seleciona apenas os casos de uso que possuem o passo selecionado

Figura 66 Cenário alternativo Incluir Caso de Teste Permanentemente

UC05 Filtro Selecionado **Ator:** Engenheiro de Teste **Cenário Alternativo 02** **Descrição:** Incluir caso de teste permanentemente		
ID	**Ação do Usuário**	**Resposta do Sistema**
CA02.01	-	Seleciona caso de teste para inclusão permanente

Figura 67 Cenário alternativo Excluir Caso de Teste Permanentemente

UC05 Filtro Selecionado **Ator:** Engenheiro de Teste **Cenário Alternativo 03** **Descrição:** Excluir caso de teste permanentemente		
ID	**Ação do Usuário**	**Resposta do Sistema**
CA03.01	-	Seleciona caso de teste para exclusão permanente

Figura 68 Cenário alternativo Filtro por Caso de Uso Similar

UC05 Filtro Selecionado **Ator:** Engenheiro de Teste **Cenário Alternativo 04** **Descrição:** Filtrar por caso de uso similar		
ID	**Ação do Usuário**	**Resposta do Sistema**
CA04.01	-	Seleciona casos de uso similares aos casos de uso selecionados

Figura 69 Cenário alternativo Filtro por Requisito

UC05 Filtro Selecionado **Ator:** Engenheiro de Teste **Cenário Alternativo 05** **Descrição:** Filtrar por requisito		
ID	**Ação do Usuário**	**Resposta do Sistema**
CA05.01	-	Marca os requisitos selecionados

Figura 70 Cenário alternativo Filtro por Caso de Uso

UC05 Filtro Selecionado **Ator:** Engenheiro de Teste **Cenário Alternativo 06** **Descrição:** Filtrar por caso de uso		
ID	**Ação do Usuário**	**Resposta do Sistema**
CA06.01	-	Marca os casos de uso selecionados para geração de teste

Figura 71 Cenário alternativo Verificar Caso de Teste Sintaticamente

UC04 Documento Verificado **Ator:** Engenheiro de Teste **Cenário Alternativo 01** **Descrição:** Checar Caso de teste sintaticamente		
ID	**Ação do Usuário**	**Resposta do Sistema**
CA01.01	-	Verifica se os casos de uso seguem o *template* suportado pela ferramenta

Figura 72 Cenário alternativo Fazer *Upload* de Documento

UC02 Documento de Requisitos Obtido **Ator:** Engenheiro de Teste **Cenário Alternativo 01** **Descrição:** *Upload* de Documento		
ID	**Ação do Usuário**	**Resposta do Sistema**
CA01.01	Fazer *upload* de documento	Solicita que o usuário indique o caminho do documento de requisitos
CA01.02	Fornece caminho do arquivo	Arquivo é importado para a ferramenta

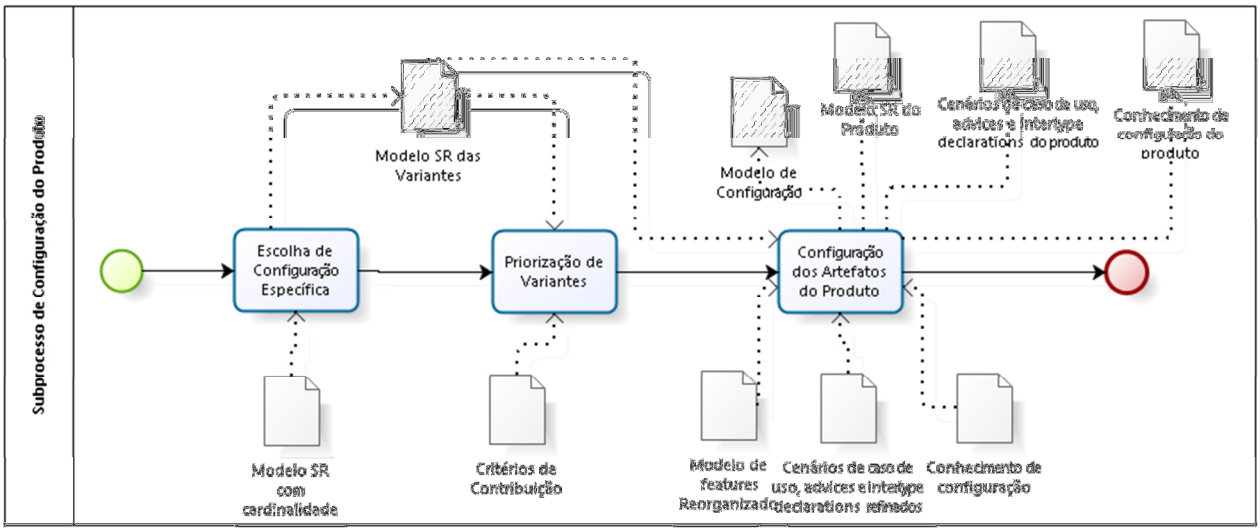
Ao finalizar esta atividade, os pontos comuns e variáveis da LPS estão definidos. Com isso, encerram-se as atividades relacionadas a Engenharia de Domínio na abordagem. A próxima atividade está relacionada à Engenharia de Aplicação, onde serão elaboradas as aplicações da linha de produto.

3.4. Configuração do Produto

Como dito anteriormente, configuração do produto é um subprocessso que dá suporte a Engenharia de Aplicação, sendo responsável por selecionar os requisitos que farão parte de um produto específico da LPS e elaborar seus artefatos. Este subprocesso deve ser executado sempre que um novo produto da LPS for requisitado. As atividades deste subprocesso são: **Escolha de Configuração Específica**, **Priorização de Variantes** (ambas apresentadas na seção 2.2.2), herdadas de Souza (2012) sem modificações e, **Configuração dos Artefatos** herdada de Souza (2012), mas, com alterações. Como artefatos de entrada utiliza-se o modelo SR com cardinalidade, o modelo de *feature* e as descrições dos cenários de casos de uso. Os artefatos de saída são o modelo SR, o modelo de configuração e as descrições dos cenários de caso de uso que foram escolhidos para o produto. A Figura 73, apresenta o subprocesso, em BPMN, de configuração do produto.

Nas subseções a seguir, será apresentada apenas a atividade de **configuração dos artefatos do produto**.

Figura 73 Modelo do subprocesso Configuração do Produto



3.4.1. Configuração dos Artefatos do Produto

1. Configurar cenários dos casos de uso dos produtos

Inicialmente, iremos verificar o artefato conhecimento de configuração (Quadro 22), elaborado na atividade 3.2, para determinar se a expressão de *feature* está relacionada ao cenário de caso de uso visando identificar se o mesmo estará presente na especificação do produto. Se a *feature* relacionada ao cenário não constar no modelo de configuração do

produto (Figura 17) obtido no Passo 1 desta atividade (apresentado na Seção 2.2.2), esse caso de uso deve ser removido da especificação do produto. O Quadro 23 apresenta o conhecimento de configuração do produto TaRGeT.

Quadro 23 Conhecimento de Configuração do produto do TaRGeT

Expressão de Feature	Transformação
Gerar Suíte de Teste Detalhada	*select scenario* ITD01 UC01.CP
Parametrizar Teste	*select scenario* ITD04.UC05.CP
Fazer Upload de Documento	*select scenario* CA01.UC02.CP
Checar Caso de Teste Sintaticamente	*select scenario* CA04.UC04.CP
Propósito de Teste	*select scenario* CA01.UC05.CP
Incluir Caso de Teste Permanentemente	*select scenario* CA02.UC05.CP
Excluir Caso de Teste Permanentemente	*select scenario* CA03.UC05.CP
Caso de Uso Similar	*select scenario* CA04.UC05.CP
Filtrar por Requisito	*select scenario* CA05.UC05.CP
Filtrar por Caso de Uso	*select scenario* CA06.UC05.CP
Manter Consistência dos Artefatos	*evaluate advice* ADV01

Após determinar quais casos de uso estarão presentes, os passos dos cenários, *intertype declarations* ou *advices* de cada um deles devem ser analisados. Se a *feature* relacionada ao passo não constar no modelo de configuração do produto, o passo deve ser removido do cenário. Posteriormente, deve-se atualizar a numeração dos passos, numerando-os sequencialmente. As figuras 74-88 apresentam as descrições dos cenários, *intertype declarations* e *advices*.

Figura 74 Caso de uso Suíte de Teste Gerada

UC01 **Suíte de Teste Gerada** **Ator:** Engenheiro de Teste **Cenário Principal 01** **Descrição:** Gerar suíte de teste		
ID	**Ação do Usuário**	**Resposta do Sistema**
CP01		

Figura 75 Caso de uso Documento de Requisitos Obtido

UC02 **Documento de Requisitos Obtido** **Ator:** Engenheiro de Requisitos **Cenário Principal 01** **Descrição:** Obter documento de requisitos		
ID	**Ação do Usuário**	**Resposta do Sistema**
CP.01	[CA.01]	Documento carregado

Figura 76 Caso de uso Documento Verificado

UC04 **Documento Verificado** **Ator:** Engenheiro de Teste **Cenário Principal** **Descrição:** Verificação de documento		
ID	**Ação do Usuário**	**Resposta do Sistema**
CP.01	[CA.01]	Caso de teste verificado sintaticamente

Figura 77 Caso de uso Filtro Selecionado

UC05 **Filtro Selecionado** **Ator:** Engenheiro de Teste **Cenário Principal** **Descrição:** Seleção de Filtros		
ID	**Ação do Usuário**	**Resposta do Sistema**
CP.01	[CA.01][CA.02][CA.03][CA.04][CA.05][CA.06]	Filtros selecionados

Figura 78 *Intertype Declarations* Gerar Suíte de Teste Detalhada

ITD01 **Descrição:** Gerar suíte de teste detalhada		
ID	**Ação do Usuário**	**Resposta do Sistema**
ITD01.01	@incluir UC02 Documento de Requisitos Obtido	Obtém documento de requisitos
ITD01.02	@incluir UC04 Documento Verificado	Verifica documento de requisito
ITD01.03	@incluir UC05 Filtro Selecionado	Seleciona filtro
ITD01.04	Gerar casos de teste específico	Gerar casos de teste para casos de uso específicos do documento de requisitos
ITD01.05	@incluir UC03 Consistência entre artefatos mantida	Mantém a consistência entre os artefatos

Figura 79 *Intertype declaration* Parametrizar Teste

ITD04 **Descrição:** Parametrizar teste		
ID	**Ação do Usuário**	**Resposta do Sistema**
ITD04.01	-	Inclui parâmetros na suíte de testes

Figura 80 *Advice* do caso de uso Consistência entre Artefatos Mantida

ADV01 **Descrição:** Manter consistência entre os artefatos **After *@ADV01***		
ID	**Ação do Usuário**	**Resposta do Sistema**
ADV01.01	Detectar mudança nos requisitos	Detecta que houve mudança nos requisitos em relação à versão anterior do documento
ADV01.02	Atualizar casos de teste	Atualiza casos de teste relacionados aos casos de uso modificados

Figura 81 Cenário alternativo Filtro por Propósito de Teste

UC05 Filtro Selecionado **Ator:** Engenheiro de Teste **Cenário Alternativo 01** **Descrição:** Filtrar por propósito de teste		
ID	**Ação do Usuário**	**Resposta do Sistema**
CA01.01	-	Seleciona apenas os casos de uso que possuem o passo selecionado

Figura 82 Cenário alternativo Incluir Caso de Teste Permanentemente

UC05 Filtro Selecionado **Ator:** Engenheiro de Teste **Cenário Alternativo 02** **Descrição:** Incluir caso de teste permanentemente		
ID	**Ação do Usuário**	**Resposta do Sistema**
CA02.01	-	Seleciona caso de teste para inclusão permanente

Figura 83 Cenário alternativo Excluir Caso de Teste Permanentemente

UC05 Filtro Selecionado **Ator:** Engenheiro de Teste **Cenário Alternativo 03** **Descrição:** Excluir caso de teste permanentemente		
ID	**Ação do Usuário**	**Resposta do Sistema**
CA03.01	-	Seleciona caso de teste para exclusão permanente

Figura 84 Cenário alternativo Filtro por Caso de Uso Similar

UC05 Filtro Selecionado **Ator:** Engenheiro de Teste **Cenário Alternativo 04** **Descrição:** Filtrar por caso de uso similar		
ID	**Ação do Usuário**	**Resposta do Sistema**
CA04.01	-	Seleciona casos de uso similares aos casos de uso selecionados

Figura 85 Cenário alternativo Filtro por Requisito

UC05 Filtro Selecionado **Ator:** Engenheiro de Teste **Cenário Alternativo 05** **Descrição:** Filtrar por requisito		
ID	**Ação do Usuário**	**Resposta do Sistema**
CA05.01	-	Marca os requisitos selecionados

Figura 86 Cenário alternativo Filtro por Requisito

UC05 Filtro Selecionado **Ator:** Engenheiro de Teste **Cenário Alternativo 06** **Descrição:** Filtrar por requisito		
ID	**Ação do Usuário**	**Resposta do Sistema**
CA06.01	-	Marca os casos de uso selecionados para geração de teste

Figura 87 Cenário alternativo Verificar Caso de Teste Sintaticamente

UC04 Documento Verificado **Ator:** Engenheiro de Teste **Cenário Alternativo 01** **Descrição:** Checar Caso de teste sintaticamente		
ID	**Ação do Usuário**	**Resposta do Sistema**
CA01.01	-	Verifica se os casos de uso seguem o *template* suportado pela ferramenta

Figura 88 Cenário alternativo Fazer *Upload* de Documento

UC02 Documento de Requisitos Obtido **Ator:** Engenheiro de Teste **Cenário Alternativo 01** **Descrição:** *Upload* de Documento		
ID	**Ação do Usuário**	**Resposta do Sistema**
CA01.01	Fazer *upload* de documento	Solicita que o usuário indique o caminho do documento de requisitos
CA01.02	Fornece caminho do arquivo	Arquivo é importado para a ferramenta

Depois de finalizar este passo, a configuração dos artefatos de requisitos do produto foi terminada. E os artefatos aqui gerados (modelo de objetivos, modelo de configuração, modelo de casos de uso e descrições dos cenários de casos de uso e *advices*) servirão de entrada para as próximas fases da EA da LPS. Estas fases não serão discutidas neste trabalho, pois o GAS2SPL contempla apenas a fase de requisitos, tanto para a Engenharia de Domínio como para a Engenharia de Aplicação.

3.5. Considerações Finais

Neste capítulo, foram apresentadas, detalhadamente, as atividades da abordagem GAS2SPL (*Goals and Aspectual Scenarios to Software Product Lines*), seus artefatos de entrada e saída, além de todas as diretrizes e passos que devem ser seguidos. O TaRGeT foi utilizado para explicar a aplicação de cada atividade e o subprocesso de configuração de produto. Em relação aos artefatos gerados, obtivemos alguns cenários, *intertype declaration* e *advices* incompletos, isto ocorre devido o modelo de objetivo possuir um nível de abstração alto e não detalhar o passo a passo da realização de algumas tarefas. À medida que se tenta aumentar o detalhamento dos modelos *i**, pode-se aumentar a complexidade deste modelo, prejudicando a legibilidade.

No subprocesso de configuração de produto poderíamos utilizar o Hephaestus [Bonifácio, Teixeira e Borba 2009] para gerar artefatos (modelo de *feature*, conhecimento de configuração e modelo de caso de uso) do produto configurado. Mas, resolveu-se não utilizá-lo para contemplarmos a configuração através da priorização atribuída pelos interessados aos requisitos não funcionais.

O próximo capítulo apresenta a aplicação da abordagem GAS2SPL ao *MyCourses* – Um sistema de agendamento de cursos [SCORE 2011].

Capítulo 4 Exemplo de Aplicação

Neste capítulo será apresentada como cada atividade da abordagem foi executada e os artefatos gerados da LPS *MyCourses* – Um sistema para agendamento de cursos [SCORE 2011] à abordagem GAS2SPL.

4.1. *MyCourses*

O agendamento de cursos é uma tarefa difícil e propensa a erros quando feita manualmente ou semi-manualmente [SCORE 2011]. Por esta razão, um programa que pode produzir o agendamento de cursos automaticamente com requisitos e cumprindo restrições é muito importante. A partir desta motivação surgiu o *MyCourses* – Um sistema para agendamento de cursos [SCORE 2011]. O *MyCourses* permite inserir dados e requisitos de uma maneira simples usando uma interface web, realiza o agendamento automático de cursos, notifica usuários de mudanças no agendamento, verifica conflitos de agendamento, calcula e propõe um horário, permite atualização manual de um agendamento e, finalmente, apresenta os resultados dos agendamentos homologados para os interessados. Há ainda a possibilidade de realizar um agendamento manualmente e a exportação dos agendamentos para outros sistemas. De acordo com Lima (2011), os atores do *MyCourses* são o administrador do sistema, o gerente de programa, os professores e os alunos. A seguir, são apresentadas algumas características destes atores:

- **Administrador do Sistema**: é uma pessoa que gerencia e mantém o sistema. O administrador tem total controle sobre o sistema, podendo adicionar, editar e excluir novos usuários, atribuindo papéis a usuários novos e existentes. O administrador é responsável pela configuração do sistema incluindo ativar e desativar as restrições, períodos da universidade, dias escolares e feriados.
- **Gerente de Programa**: é responsável pelo gerenciamento de programas na universidade. O gerente de programa pode criar um novo curso ou modificar os existentes, identificar o programa e o seu período de execução (começo e fim do ano), adicionar um curso e atribuir um professor a ele. O gerente de programa pode executar o agendamento automático do curso, modificar o agendamento manualmente, aceitar ou descartar uma proposta de curso feita por um professor e homologar os agendamentos.
- **Professor**: é o responsável por detalhes de um curso atribuído a ele. O professor pode adicionar ou modificar elementos de um curso existente, definir outras pessoas que

estão participando no processo de ensino.

- **Aluno**: consultam ou são notificados sobre o resultado das homologações ou alterações no sistema de agendamento.

Todos os usuários podem realizar a exportação dos dados do agendamento. Alunos e Professores podem exportar os dados em formato PDF, por sua vez, Gerentes de Programa e Administradores de Sistema terão acesso a todos os formatos disponíveis (PDF, XML e SQL). Dentre as funcionalidades citadas anteriormente, a principal é o agendamento de curso. Esta funcionalidade é responsável por manipular dados de usuários, programas, cursos e recursos, combiná-los e realizar o agendamento. O sistema deve disponibilizar meios de realizar a reserva de recursos. Dependendo da maneira como a instituição de ensino administra os seus programas, há duas maneiras de fazê-la: possibilitar que se aproveite a maior parte dos recursos (manter ocupado a maior parte do tempo) ou, possibilitar que a maior parte do tempo os recursos estejam disponíveis (manter livre a maior parte do tempo). O *MyCourses* deve disponibilizar uma maneira de gerenciar conflitos de membros (levando em consideração a disponibilidade), gerenciar conflito de recursos (por exemplo, impedir que o mesmo espaço físico seja ocupado por mais de um evento).

Os requisitos não funcionais do *MyCourses* são:

- **Desempenho**: o sistema deve realizar o agendamento e disponibilizar resultados com um bom tempo de resposta.
- **Satisfação dos usuários**: o agendamento realizado de maneira automática e gerenciado os conflitos deve obter bons níveis de satisfação dos usuários.
- **Otimização dos recursos**: independente da maneira que a instituição trata a alocação de recursos, o sistema deverá manter consistentes as reservas de recursos.
- **Permitir alteração manual:** quando o agendamento for realizado de maneira semiautomática deverá permitir a definição de horários fixos de acordo com as definições do usuário.
- **Julgamento justo:** o agendamento será efetivado apenas se houver a homologação por parte do gerente do programa. Visando evitar que o professor agende e homologue o próprio horário.

Nas próximas seções será apresentada a aplicação da abordagem GAS2SPL no *MyCourses*, detalhando como cada atividade foi executada e os artefatos gerados.

4.2. Aplicando o GAS2SPL ao *MyCourses*

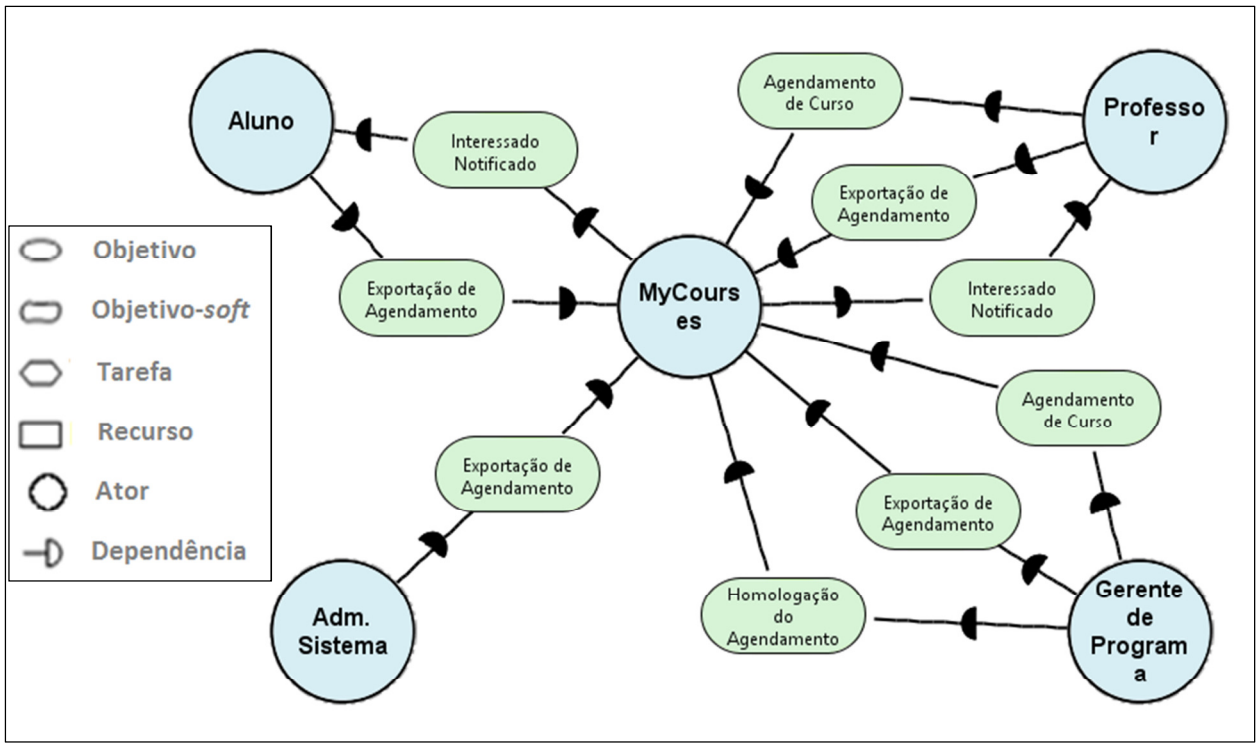
Utilizaremos as descrições sobre o *MyCourses* disponíveis na seção anterior para a modelagem dos artefatos.

4.2.1. Criação do Modelo SR

Utilizaremos as diretrizes apresentadas na seção 2.2.2 para elaborar o modelo de objetivos do framework *i** para o *MyCourses*. Inicialmente, será elaborado o Modelo SD (Figura 89) e, baseado neste modelo, será construído o Modelo SR.

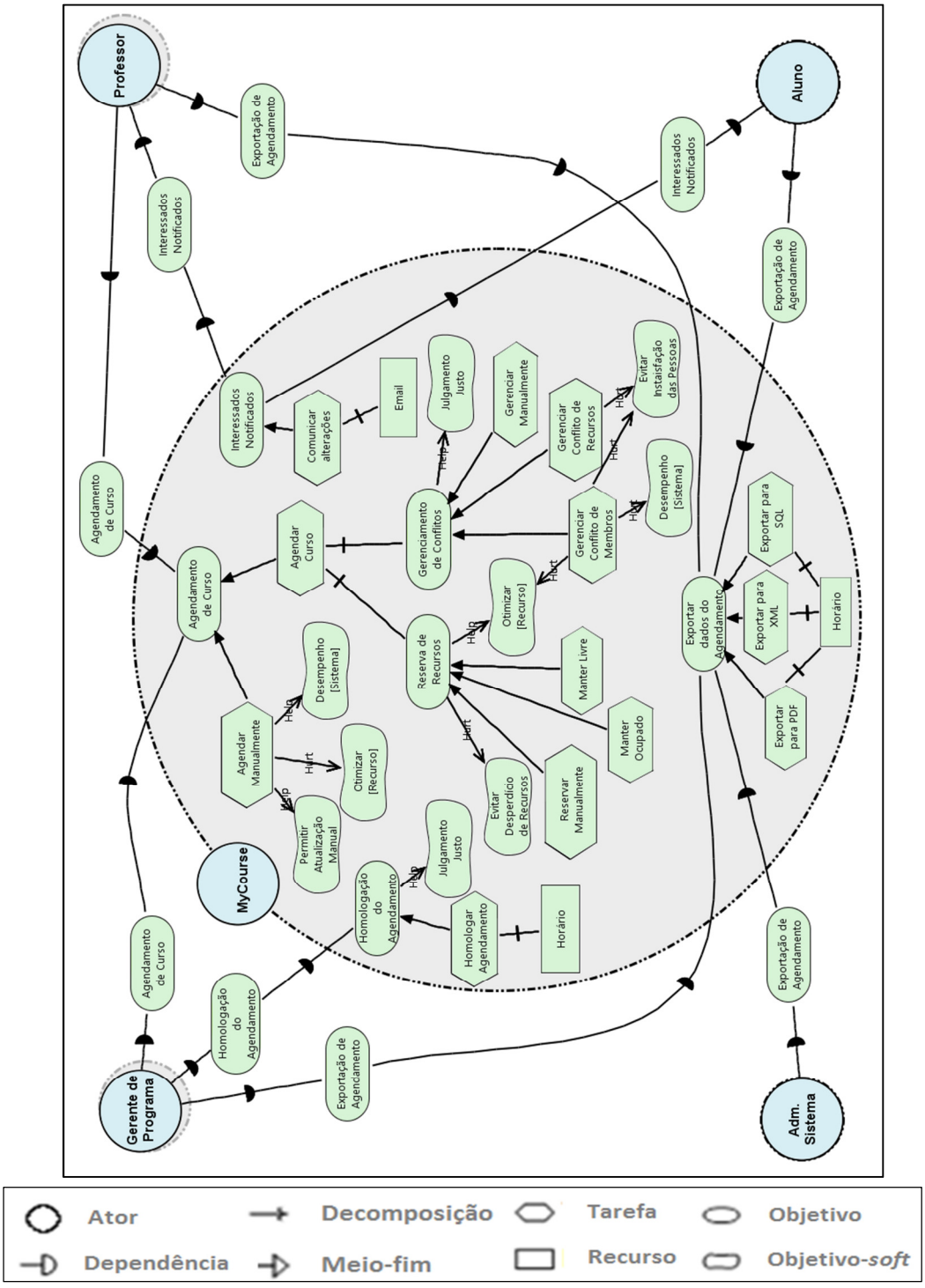
A partir da documentação do *MyCourses* apresentada na seção 4.1 identificamos cinco atores (representados, no modelo SD, por círculos): Aluno, Professor, Administrador do Sistema, Gerente de Programa e o próprio sistema, *MyCourses*. Em seguida, foram modeladas as dependências entre os atores. Observando a Figura 89 percebemos que o ator **Administrador do Sistema** depende do ator ***MyCourses*** para que o seu objetivo ***Exportação de Agendamento*** seja satisfeito. O **Aluno** utiliza o ***MyCourses*** para ***Exportação de Agendamento***. Por sua vez, o ***MyCourses*** notifica os atores **Aluno** e **Professor** quando há alguma modificação ou homologação nos cursos em que eles estão inseridos (***Interessado Notificado***). Os atores **Professor** e **Gerente de Programa** dependem do ator ***MyCourses*** para que os objetivos ***Exportação de Agendamento*** e ***Agendamento de Curso*** aconteçam. O **Gerente de Programa** utiliza o ***MyCourses*** para validar os agendamentos solicitados (***Homologação do Agendamento***).

Figura 89 Modelo SD do MyCourses



Após a conclusão do modelo SD, expande-se as fronteiras dos atores para analisar como as dependências são atendidas internamente. Como forma de facilitar o entendimento, apresentamos na Figura 90 apenas a expansão da fronteira do ator ***MyCourses*** representado por um círculo tracejado.

Figura 90 Modelo SR do MyCourses



4.2. Aplicando o GAS2SPL ao *MyCourses*

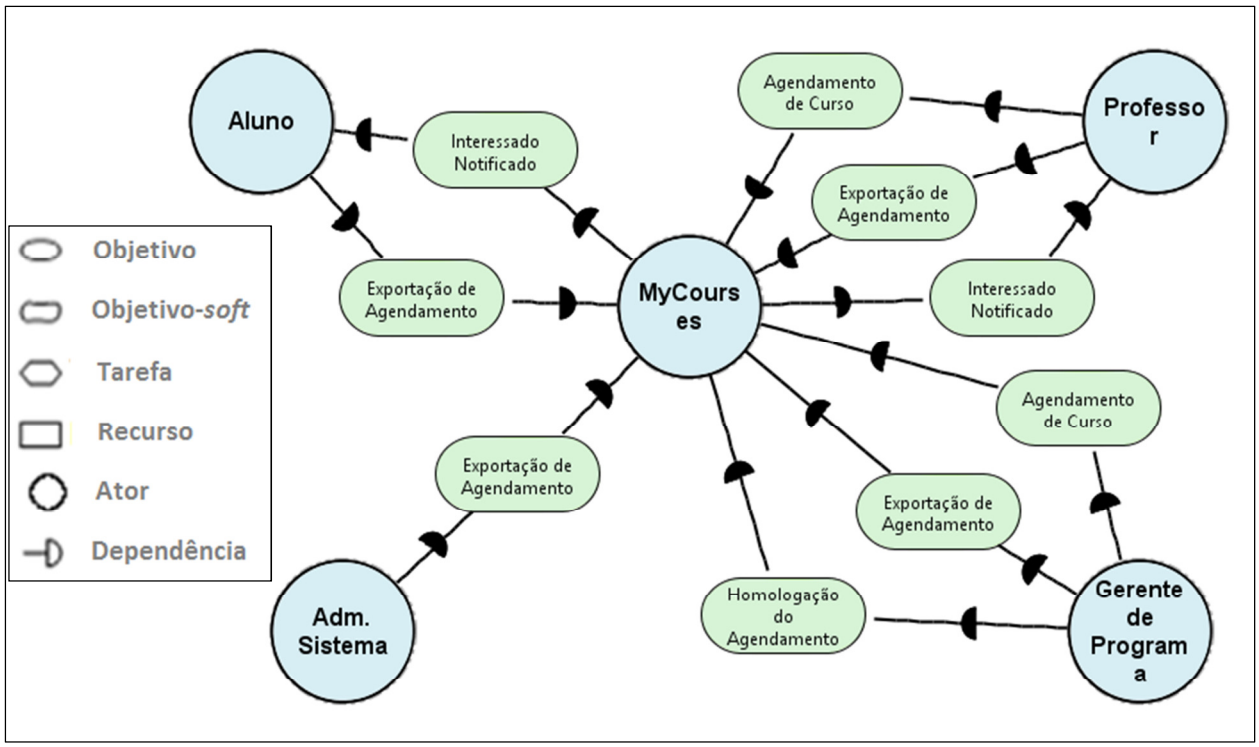
Utilizaremos as descrições sobre o *MyCourses* disponíveis na seção anterior para a modelagem dos artefatos.

4.2.1. Criação do Modelo SR

Utilizaremos as diretrizes apresentadas na seção 2.2.2 para elaborar o modelo de objetivos do framework *i** para o *MyCourses*. Inicialmente, será elaborado o Modelo SD (Figura 89) e, baseado neste modelo, será construído o Modelo SR.

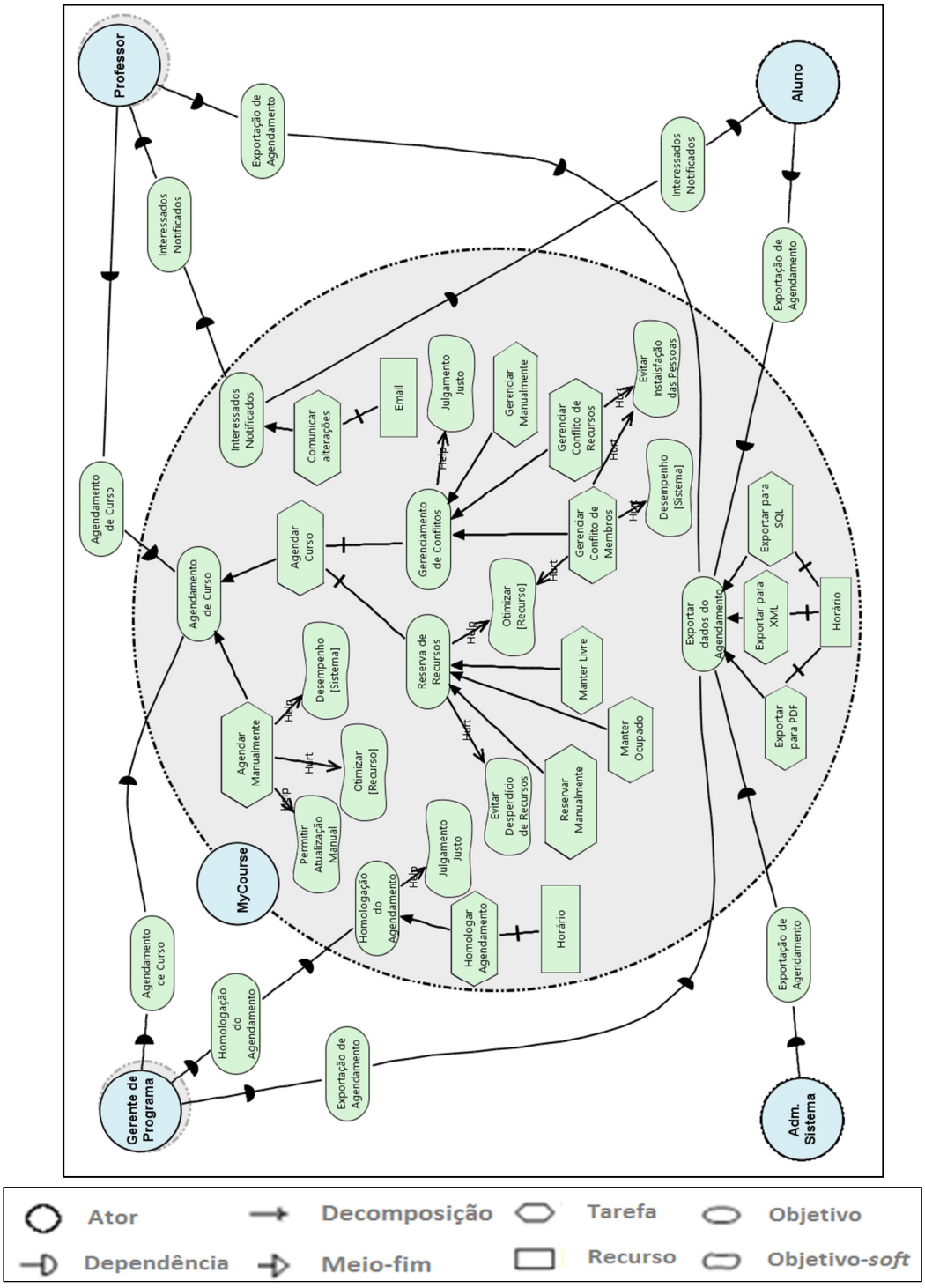
A partir da documentação do *MyCourses* apresentada na seção 4.1 identificamos cinco atores (representados, no modelo SD, por círculos): Aluno, Professor, Administrador do Sistema, Gerente de Programa e o próprio sistema, *MyCourses*. Em seguida, foram modeladas as dependências entre os atores. Observando a Figura 89 percebemos que o ator **Administrador do Sistema** depende do ator ***MyCourses*** para que o seu objetivo ***Exportação de Agendamento*** seja satisfeito. O **Aluno** utiliza o ***MyCourses*** para ***Exportação de Agendamento***. Por sua vez, o ***MyCourses*** notifica os atores **Aluno** e **Professor** quando há alguma modificação ou homologação nos cursos em que eles estão inseridos (***Interessado Notificado***). Os atores **Professor** e **Gerente de Programa** dependem do ator ***MyCourses*** para que os objetivos ***Exportação de Agendamento*** e ***Agendamento de Curso*** aconteçam. O **Gerente de Programa** utiliza o ***MyCourses*** para validar os agendamentos solicitados (***Homologação do Agendamento***).

Figura 89 Modelo SD do MyCourses



Após a conclusão do modelo SD, expande-se as fronteiras dos atores para analisar como as dependências são atendidas internamente. Como forma de facilitar o entendimento, apresentamos na Figura 90 apenas a expansão da fronteira do ator ***MyCourses*** representado por um círculo tracejado.

Figura 90 Modelo SR do MyCourses



Como podemos ver na Figura 90 o objetivo principal do ***MyCourses*** é o **Agendamento de Cursos**, que vai atender à dependência ***Agendamento de Cursos*** dos atores **Professor** e **Gerente de Programa**. Estas dependências podem ser realizadas manualmente (através da execução da tarefa ***Agendar Manualmente***) ou através do sistema (através da execução da tarefa ***Agendar Curso***), pois as mesmas estão ligadas ao objetivo ***Agendamento de Curso*** por uma relação do tipo meio-fim. A tarefa ***Agendar Manualmente*** contribui positivamente para que os objetivos-*soft* ***Permitir atualização manual*** e ***Desempenho[Sistema]*** sejam satisfeitos e, contribui negativamente para o objetivo-*soft* ***Otimização[Recurso]***. A realização da tarefa ***Agendar Curso***, por sua vez, é decomposta nos objetivos ***Gerenciamento de conflitos*** e ***Reserva de recursos***. O objetivo ***Gerenciamento de conflitos*** ajuda a satisfazer o objetivo-*soft* ***Julgamento justo***. Este objetivo é realizado através da execução das tarefas ***Gerenciar manualmente, Gerenciar Conflito de Membros*** (que contribui negativamente para ***Desempenho[Sistema]***, ***Otimizar[Recurso]*** e ***Evitar insatisfação das pessoas***) e ***Gerenciar Conflito de Recursos*** (que contribui negativamente para ***Evitar insatisfação das pessoas***). O objetivo ***Reserva de recurso*** é atingido através da execução das tarefas ***Reservar manualmente***, ***Manter ocupado*** e ***Manter livre*** e, ajuda a satisfazer o objetivo-*soft* ***Otimizar[Recurso]***, mas, contribui negativamente para o objetivo-*soft* ***Evitar desperdício de recurso***. O ***MyCourses*** satisfaz o objetivo ***Interessados notificados*** por meio da realização da tarefa ***Comunicar alterações*** utilizando como recurso o ***Email***. A dependência ***Exportação de agendamento*** é satisfeita através do objetivo ***Exportar dados do agendamento***, que é operacionalizado através das tarefas ***Exportar para PDF***, ***Exportar para XML*** e ***Exportar para SQL*** gerando como recurso o ***Horário***. A dependência ***Homologação do agendamento*** ajuda ao cumprimento do objetivo-*soft* ***Julgamento justo*** através da realização da tarefa ***Homologar agendamento*** utilizando o recurso ***Horário***.

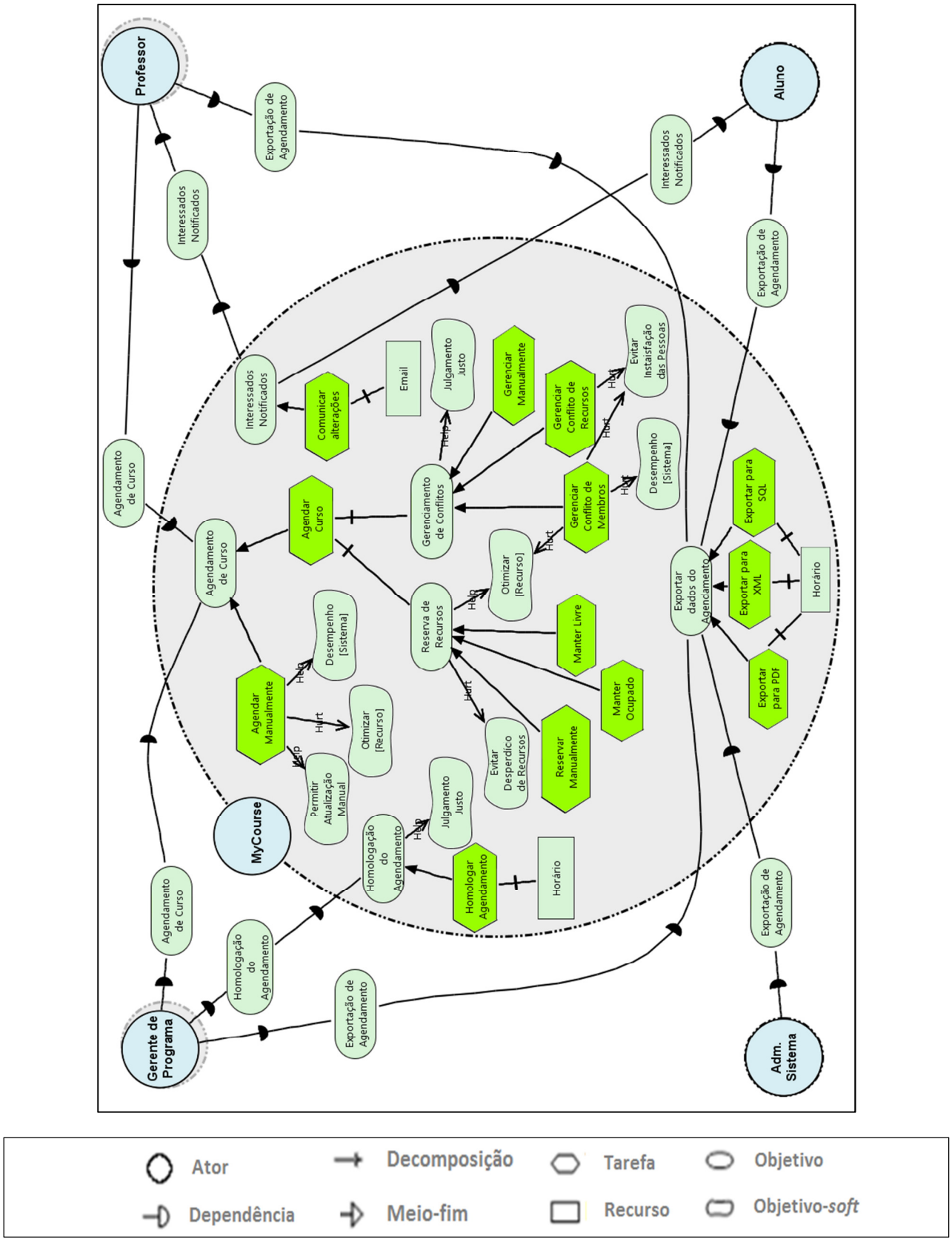
A próxima atividade tem como objetivo destacar no modelo SR, elaborado nesta atividade, os elementos que podem originar *features*.

4.2.2. Identificação dos elementos candidatos a *features*

Para elaboração do modelo SR com os elementos candidatos a *features* destacados desta atividade utilizaremos as heurísticas definidas na seção 2.2.2. Analisando o modelo da Figura 90 percebemos que não há dependência do tipo recurso ou tarefa, mas há várias tarefas como elementos internos do ator que representa o *MyCourses* e todas elas devem ser destacadas como possíveis *features*. A Figura 91 mostra o modelo SR do *MyCourses* após a aplicação dessas heurísticas.

A atividade a seguir tem como objetivo adicionar cardinalidade ao modelo SR elaborado nesta atividade para que se possa capturar informação relativa à presença opcional e obrigatória de algumas *features* em um produto da LPS.

Figura 91 Modelo SR com os elementos candidatos a *features* destacados do *MyCourses*

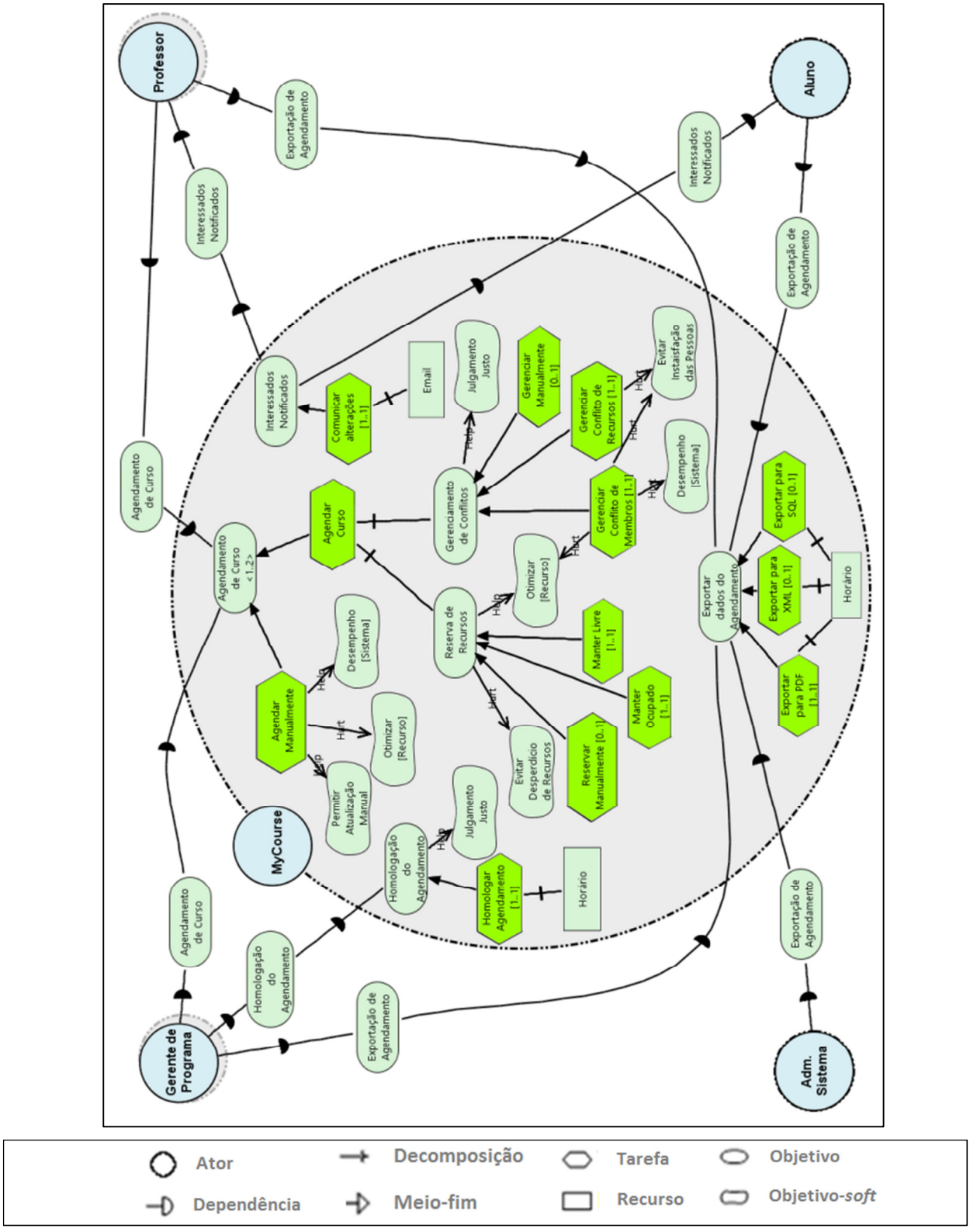


4.2.3. Reengenharia do modelo SR

Adicionaremos cardinalidade ao modelo SR elaborado na atividade anterior, seguindo as heurísticas da seção 2.2.2. Aplicando as heurísticas ao *MyCourses*, através de **H3** descobre-se que todas as tarefas destacadas irão conter cardinalidade. Por **H4**, tem-se que a cardinalidade de **Agendar Manualmente** e **Agendar Curso** serão agrupadas e a cardinalidade pertencerá ao relacionamento. A cardinalidade do grupo formado por **Agendar Manualmente** e **Agendar Curso** é <1..2>, pois deve existir ao menos um tipo de agendamento de curso, mas pode ser qualquer uma das duas opções ou até mesmo ambas. Ainda de acordo com **H4**, **Homologar Agendamento** e **Comunicar alterações** terão cardinalidade [1..1]. De acordo com a heurística **H4**, as tarefas relacionadas por ligação meio-fim aos objetivos **Reserva de recursos**, **Gerenciamento de conflitos** e **Exportar dados do agendamento** poderiam ser agrupadas e ter a cardinalidade posta no relacionamento. Porém, nesses agrupamentos existem tarefas que são obrigatórias e outras que são opcionais. Por isso, também conforme a heurística **H4**, optou-se por colocar a cardinalidade nas tarefas, ao invés de colocá-las no relacionamento. Sendo assim, as tarefas **Manter Livre**, **Manter Ocupado**, **Gerenciar Conflito de Membros**, **Gerenciar Conflito de Recursos** e **Exportar para PDF** são obrigatórias e possuirão cardinalidade [1..1] e, **Gerenciar Manualmente**, **Reservar Manualmente**, **Exportar para XML** e **Exportar para SQL** irão conter cardinalidade [0..1], pois são opcionais.

A Figura 92 apresenta o modelo SR com cardinalidade gerado após a aplicação das heurísticas.

Figura 92 Modelo SR com cardinalidade do *MyCourses*



4.2.4. Elaboração do Modelo de *feature*

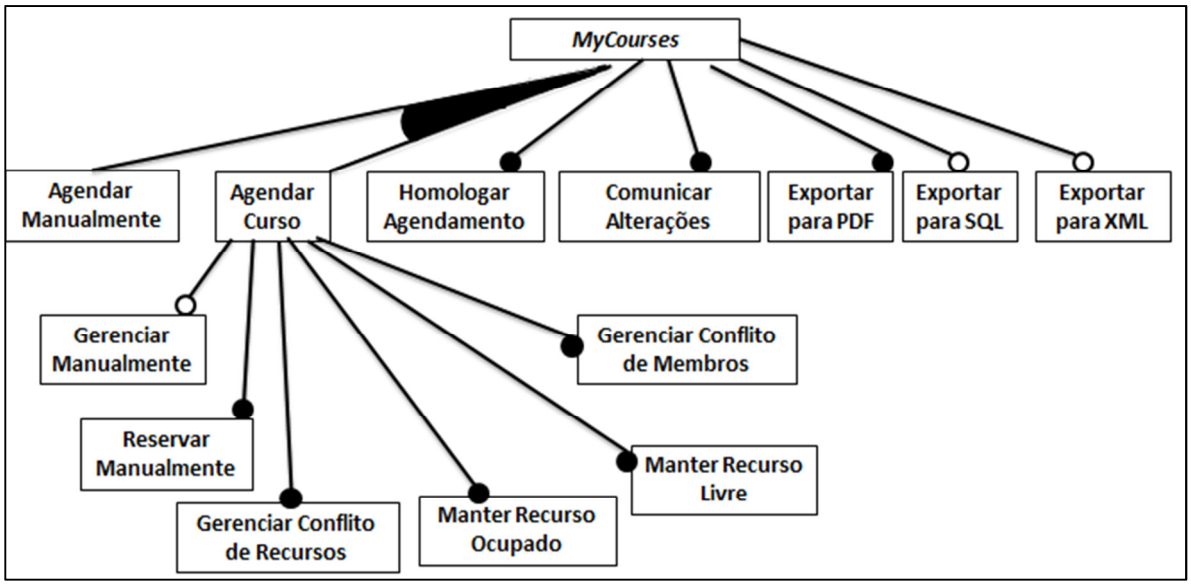
Com base no modelo SR com cardinalidade (Figura 92), utilizaremos das heurísticas definidas na seção 2.2.2 para elaborar o quadro de mapeamento entre elementos e *features*. Através de **H8**, a feature raiz é ***MyCourses***. As heurísticas **H9 e H10** irão guiar o processo de elaboração do quadro de mapeamento (Quadro 24).

Quadro 24 Mapeamento entre elementos e features

Elemento	Cardinalidade		Elemento Pai	*Feature*
	Tipo	Valor		
Agendar Manualmente	Grupo	<1..2>	-	Agendar Manualmente
Agendar Curso	Grupo	<1..2>	-	Agendar Curso
Homologar Agendamento	Elemento	[1..1]	-	Homologar Agendamento
Comunicar Alterações	Elemento	[1..1]	-	Comunicar Alterações
Exportar para PDF	Elemento	[1..1]	-	PDF
Exportar para SQL	Elemento	[0..1]	-	SQL
Exportar para XML	Elemento	[0..1]	-	XML
Reservar Manualmente	Elemento	[0..1]	Agendar Curso	Reservar Manualmente
Gerenciar Manualmente	Elemento	[0..1]	Agendar Curso	Gerenciar Manualmente
Gerenciar Conflito de Recursos	Elemento	[1..1]	Agendar Curso	Recursos
Gerenciar Conflito de Membros	Elemento	[1..1]	Agendar Curso	Membros
Manter Ocupado	Elemento	[1..1]	Agendar Curso	Manter Ocupado
Manter Livre	Elemento	[1..1]	Agendar Curso	Manter Livre

Após a elaboração do Quadro 24, as heurísticas **H11** e **H12** são aplicadas na análise da tabela para a construção do modelo de *features*. A Figura 93 apresenta o modelo de *feature* obtido após a aplicação das heurísticas desta atividade.

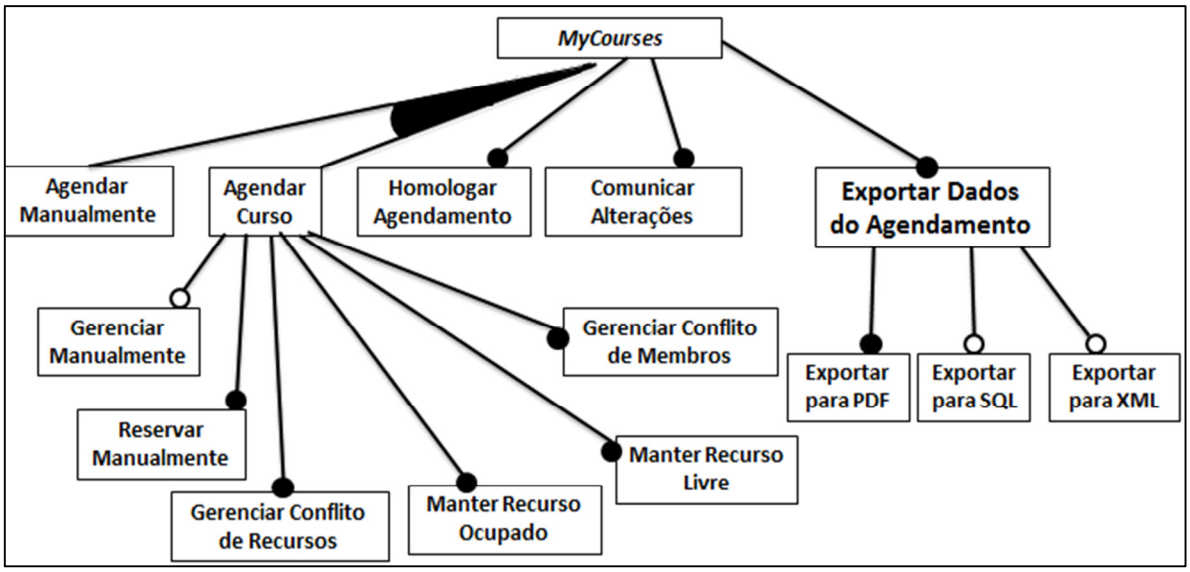
Figura 93 Modelo de *feature* do *MyCourses*



4.2.5. Refinamento do Modelo de *features*

Analisando o modelo de *feature* definido na atividade anterior (Figura 93) percebemos que ele não apresenta inconsistências como, por exemplo, *features* relacionadas a mais de uma feature, *features* repetidas, *features* como a mesma semântica, entre outras. Após discussões entre o autor do trabalho e sua orientadora sobre a estrutura do modelo da Figura 93 resolveu-se reorganizá-lo adicionando a feature **Exportar Dados de Agendamento**. A Figura 94 apresenta o modelo de *feature* refinado.

Figura 94 Modelo de *feature* do *MyCourses* refinado



4.2.6. Elaboração dos Cenários

Para elaborar os cenários, *advices*, *intertype declarations* e o conhecimento de configuração utilizaremos o modelo SR com cardinalidade (Figura 92) elaborado na atividade de Reengenharia do Modelo SR (seção 4.2.3), e o modelo de *feature* elaborado na atividade de Elaboração do Modelo de *feature* (seção 4.2.4) ou na atividade de Refinamento do modelo de *feature* (seção 4.2.5), caso tenha sido realizada. Utilizaremos o modelo de *feature* da Figura 94. Considerando as diretrizes definidas na seção 3.2, a seguir, apresentaremos a execução das diretrizes da atividade.

Diretrizes relacionadas ao elemento ator (Diretrizes 1 - 4)

Aplicando as quatro primeiras diretrizes ao exemplo *MyCourses*, obtemos, pela **Diretriz 1**, os atores *Gerente de Programa, MyCourses, Professor, Aluno e Administrador do Sistema*. Mas, de acordo com a **Diretriz 2**, o ator deve ser externo ao sistema, então o ator *MyCourses* não pode ser mapeado como um ator de caso de uso, restando *Gerente de Programa, Professor, Aluno e Administrador do Sistema*. Analisando a **Diretriz 3**

percebemos que todos os atores que foram mapeados na diretriz anterior possuem dependências com o ator *MyCourses*. A **Diretriz 4** não se aplica ao *MyCourses* pois não há atores que possuem apenas dependência de objetivo-*soft* com o ator *MyCourses*. Assim, os atores *Gerente de Programa, Professor, Aluno e Administrador do Sistema* atendem as diretrizes e podem ser mapeados como atores de caso de uso.

Diretrizes relacionadas a dependências (Diretriz 5)

De acordo com a **Diretriz 5** iremos analisar as dependências entre os atores *Gerente de Programa*, *Professor*, *Aluno* e *Administrador do Sistema* com o ator *MyCourses*. A **Diretriz 5.1** que trata do mapeamento de objetivos no modelo SR para casos de uso será aplicada. Assim, temos as informações contidas no Quadro 25. Como, no modelo SR, não temos dependências do tipo tarefa, recurso e objetivo-*soft*, as diretrizes 5.2, 5.3 e 5.4, respectivamente, não se aplicam ao exemplo *MyCourses*.

Quadro 25 Mapeamento entre atores do *MyCourses* (Diretriz 5.1)

Ator	Dependência	Tipo de Dependência	Diretriz a ser usada
Gerente de Programa	Agendamento de Curso	Objetivo	5.1
Gerente de Programa	Homologação do Agendamento	Objetivo	5.1
Gerente de Programa	Exportação do Agendamento	Objetivo	5.1
Professor	Agendamento de Curso	Objetivo	5.1
Professor	Interessados Notificados	Objetivo	5.1
Professor	Exportação de Agendamento	Objetivo	5.1
Administrador do Sistema	Exportação de Agendamento	Objetivo	5.1
Aluno	Exportação de Agendamento	Objetivo	5.1
Aluno	Interessados Notificados	Objetivo	5.1

Analisando o Quadro 25, os casos de uso gerados nessa atividade são: *Agendamento de Curso*, *Homologação do Agendamento*, *Exportação do Agendamento,* e *Interessados Notificados.*

As diretrizes a seguir visam extrair informações para gerar casos de uso, *advices*, *intertype declaration* e elaborar o conhecimento de configuração.

Diretrizes relacionadas ao mapeamento de cenários, advices, intertype declarations e conhecimento de configuração

A **Diretriz 5.5** trata do *template* para a elaboração de cenários dos casos de uso identificados na diretriz anterior para o *MyCourses*. Os cenários dos casos de uso gerados para *Agendamento de Curso*, *Homologação do Agendamento*, *Exportação do Agendamento* e *Interessados Notificados* são apresentados nas figuras Figura 95, Figura 96, Figura 97 e Figura 98, respectivamente.

Figura 95 Caso de uso Agendamento de Curso

UC01 **Agendamento de Curso** **Ator:** Gerente de Programa, Professor **Cenário Principal 01** **Descrição:** Agendar Curso		
ID	**Ação do Usuário**	**Resposta do Sistema**
CP01.01		

Figura 96 Caso de uso Homologação do Agendamento

UC02 **Homologação do Agendamento** **Ator:** Gerente de Programa **Cenário Principal 01** **Descrição:** Homologar agendamento		
ID	**Ação do Usuário**	**Resposta do Sistema**
CP01.01		

Figura 97 Caso de uso Exportação do Agendamento

UC03 **Exportação do Agendamento** **Ator:** Gerente de Programa, Professor, Administrador de Sistema, Aluno **Cenário Principal 01** **Descrição:** Exportar agendamentos		
ID	**Ação do Usuário**	**Resposta do Sistema**
CP01.01		

Figura 98 Caso de uso Interessados Notificados

UC04 **Interessados Notificados** **Ator:** Professor, Aluno **Cenário Principal 01** **Descrição:** Notificar interessados		
ID	**Ação do Usuário**	**Resposta do Sistema**
CP01.01		

Analisando o modelo SR, percebe-se que o caso de uso derivado do objetivo principal do modelo SR do *MyCourses* é o agendamento de curso. Logo, o caso de uso Agendamento de Curso será mapeado pelas diretrizes **5.6** e **5.7** para o conhecimento de configuração (Quadro 26).

Quadro 26 V1 do Conhecimento de Configuração do MyCourses

Expressão de Feature	Transformação
Agendamento de Curso	*select scenario* UC01.CP

A **Diretriz 6.1** trata do mapeamento de caso de uso a partir de objetivos internos ao ator do sistema que não possuem ligação direta com uma dependência. Aplicando a **Diretriz 6.1** obtêm-se os casos de uso *Reserva de Recursos e Gerenciamento de Conflitos* apresentados nas figuras Figura 99 e Figura 100, respectivamente.

Figura 99 Caso de uso Reserva de Recursos

UC05 **Reserva de Recursos** **Ator:** Gerente de Programa, Professor **Cenário Principal 01** **Descrição:** Reservar recursos		
ID	**Ação do Usuário**	**Resposta do Sistema**
CP01.01		

Figura 100 Caso de uso Gerenciamento de Conflitos

UC06 **Gerenciamento de Conflitos** **Ator:** Gerente de Programa, Professor **Cenário Principal 01** **Descrição:** Gerenciar conflitos		
ID	**Ação do Usuário**	**Resposta do Sistema**
CP01.01		

A **Diretriz 6.2** e suas sub-diretrizes não se aplicam ao exemplo *MyCourses*, pois o seu modelo SR não apresenta elemento com cardinalidade [1..1] que seja sub-elemento de outro elemento com cardinalidade [1..1].

As diretrizes a seguir analisam os sub-elementos com cardinalidade [0..1] ou envolvidos em relacionamentos meio-fim com cardinalidade <*i..j*> e podem ser representados como *advices* ou *intertype declarations.*

Analisando o modelo SR, de acordo com a **Diretriz 6.3.1**, se a cardinalidade estiver na ligação meio-fim, os sub-elementos serão mapeados para *intertype declarations*. Aplicando

as diretrizes 6.3.1 e 6.3.2 ao *MyCourses*, os *intertype declarations* gerados são apresentados na Figura 101 e Figura 102.

Figura 101 *Intertype declaration* Agendar Manualmente

ITD01 **Descrição:** Agendar curso manualmente		
ID	**Ação do Usuário**	**Resposta do Sistema**
ITD01.01		

Figura 102 *Intertype declaration* Agendar Curso

ITD02 **Descrição:** Agendar curso		
ID	**Ação do Usuário**	**Resposta do Sistema**
ITD02.01		

A **Diretriz 6.3.3** trata do mapeamento de *intertype declaration* no conhecimento de configuração, através da aplicação da transformação ***select scenario***. Aplicando as diretrizes **6.3.3** e **5.7** ao *MyCourses*, obtemos o seguinte conhecimento de configuração (Quadro 27).

Quadro 27 V2 Conhecimento de Configuração do *MyCourses*

Expressão de Feature	**Transformação**
Agendamento de Curso	*select scenario* UC01.CP
Agendar Curso Manualmente	*select scenario* ITD01 UC01.CP
Agendar Curso	*select scenario* ITD02 UC01.CP

De acordo com a **Diretriz 6.3.4,** uma ligação meio-fim com mais de um sub-elemento e, cuja cardinalidade [0..1] está no sub-elemento, pode-se mapeá-lo para um *intertype declaration*. Aplicando as diretrizes 6.3.2 e 6.3.4 ao *MyCourses*, os *intertype declarations* gerados são apresentados na Figura 103, Figura 104, Figura 105 e Figura 106.

Figura 103 *Intertype declaration* Reservar Manualmente

ITD03 **Descrição:** Reservar manualmente		
ID	**Ação do Usuário**	**Resposta do Sistema**
ITD03.01		

Figura 104 *Intertype declaration* Gerenciar Manualmente

ITD04 **Descrição:** Gerenciar manualmente		
ID	**Ação do Usuário**	**Resposta do Sistema**
ITD04.01		

Figura 105 *Intertype declaration* Exportar para XML

ITD05 **Descrição:** Exportar dados de agendamento para XML		
ID	**Ação do Usuário**	**Resposta do Sistema**
ITD05.01		

Figura 106 *Intertype declaration* Exportar para SQL

ITD06 **Descrição:** Exportar dados de agendamento para SQL		
ID	**Ação do Usuário**	**Resposta do Sistema**
ITD06.01		

O Quadro 28 apresenta uma versão atualizada do conhecimento de configuração do *MyCourses* após a elaboração dos *intertype declarations* acima.

Quadro 28 V3 Conhecimento de Configuração do *MyCourses*

Expressão de Feature	**Transformação**
Agendamento de Curso	*select scenario* UC01.CP.01
Agendar Curso Manualmente	*select scenario* ITD01 UC01.CP
Agendar Curso	*select scenario* ITD02 UC01.CP
Reservar Manualmente	*select scenario* ITD03 UC05.CP
Gerenciar Manualmente	*select scenario* ITD04 UC06.CP
Exportar para XML	*select scenario* ITD05 UC03.CP
Exportar para SQL	*select scenario* ITD06 UC03.CP

A **Diretriz 6.3.5** não se aplica ao *MyCourses*, pois não há ligação meio-fim que possua apenas um sub-elemento com cardinalidade [0..1]. A **Diretriz 6.3.6** não se aplica ao *MyCourses*, pois o seu modelo SR não há ligação de decomposição de tarefa que possua sub-elemento com cardinalidade [0..1].

Como não temos advices no *MyCourses*, conforme a aplicação das diretrizes anteriores, a **Diretriz 6.3.7** que define o *template* do *advice* não se aplica ao *MyCourses*. A **Diretriz 6.3.8** que trata do mapeamento de *advices* no conhecimento de configuração através da transformação *evaluate advice* não se aplica ao exemplo *MyCourses*. As diretrizes **6.3.9** e **6.3.10** que tratam do mapeamento de parâmetros, não se aplicam ao *MyCourses*, pois o seu modelo SR não apresenta cenários que compartilham a mesma sequência de passos e utilizam variáveis diferentes.

A **Diretriz 6.4.1** não se aplica ao *MyCourses*, pois não há, em seu modelo SR, sub-elementos com cardinalidade [1..1] de uma decomposição de tarefa com cardinalidade [0..1] ou alternativa de uma ligação meio-fim com cardinalidade.

Se uma ligação meio-fim tiver mais de um sub-elemento cuja cardinalidade é [1..1], ele será mapeado para um cenário alternativo. Aplicando a **Diretriz 6.4.2** ao *MyCourses*, os cenários alternativos gerados são apresentados nas Figuras 107-114.

Figura 107 Cenário alternativo Manter Recurso Ocupado

UC05 Reserva de Recurso **Ator:** Gerente de Programa, Professor **Cenário Alternativo 01** **Descrição:** Manter recurso ocupado		
ID	**Ação do Usuário**	**Resposta do Sistema**
CA01.01		

Figura 108 Cenário alternativo Manter Recurso Livre

UC05 Reserva de Recurso **Ator:** Gerente de Programa, Professor **Cenário Alternativo 02** **Descrição:** Manter recurso livre		
ID	**Ação do Usuário**	**Resposta do Sistema**
CA02.01		

Figura 109 Cenário alternativo Gerenciar Conflito de Membros

UC06 Gerenciamento de Conflito **Ator:** Gerente de Programa, Professor **Cenário Alternativo 01** **Descrição:** Gerenciar conflito de membros		
ID	**Ação do Usuário**	**Resposta do Sistema**
CA01.01		

Figura 110 Cenário alternativo Gerenciar Conflito de Recurso

UC06 Gerenciamento de Conflito **Ator:** Gerente de Programa, Professor **Cenário Alternativo 02** **Descrição:** Gerenciar conflito de recurso		
ID	**Ação do Usuário**	**Resposta do Sistema**
CA02.01		

Figura 111 Cenário alternativo Exportar Dados para PDF

UC03 Exportação do Agendamento **Ator:** Gerente de Programa, Professor, Administrador do Sistema, Aluno **Cenário Alternativo 01** **Descrição:** Exportar dados para PDF		
ID	**Ação do Usuário**	**Resposta do Sistema**
CA01.01		

Figura 112 Versão 2 do caso de uso Reserva de Recurso

UC05 **Reserva de Recurso** **Ator:** Gerente de Programa, Professor **Cenário Principal** **Descrição:** Reservar recursos		
ID	**Ação do Usuário**	**Resposta do Sistema**
CP.01	[CA.01] [CA.02]	

Figura 113 Versão 2 do caso de uso Gerenciamento de Conflito

UC06 **Gerenciamento de Conflito** **Ator:** Gerente de Programa, Professor **Cenário Principal** **Descrição:** Gerenciar conflito		
ID	**Ação do Usuário**	**Resposta do Sistema**
CP.01	[CA.01] [CA.02]	

Figura 114 Versão 2 do caso de uso Exportação do Agendamento

UC03 **Exportação do Agendamento** **Ator:** Gerente de Programa, Professor, Administrador do Sistema, Aluno **Cenário Principal** **Descrição:** Exportar agendamento		
ID	**Ação do Usuário**	**Resposta do Sistema**
CP.01	[CA.01]	

Se uma ligação meio-fim tiver apenas um sub-elemento e esse sub-elemento tiver cardinalidade [1..1], deve-se aplicar a **Diretriz 6.4.3**. No *MyCourses*, os cenários alternativos gerados são apresentados na Figura 115 e Figura 116.

Figura 115 Cenário alternativo Homologar Agendamento

UC02 Homologação de Agendamento		
Ator: Gerente de Programa **Cenário Alternativo 01** **Descrição:** Homologar agendamento		
ID	**Ação do Usuário**	**Resposta do Sistema**
CA01.01		

Figura 116 Cenário alternativo Comunicar Alterações

UC05 Interessados Notificados **Ator:** Professor, Aluno **Cenário Alternativo 01** **Descrição:** Comunicar alterações		
ID	**Ação do Usuário**	**Resposta do Sistema**
CA01.01		

A Figura 117 e a Figura 118 apresentam versões atualizadas dos cenários principais após a aplicação da diretriz 6.3.4.

Figura 117 Versão 2 do caso de uso Homologação de Agendamento

UC02 Homologação de Agendamento **Ator:** Gerente de Programa **Cenário Principal** **Descrição:** Homologar agendamento		
ID	**Ação do Usuário**	**Resposta do Sistema**
CP.01	[CA.01]	

Figura 118 Versão 2 do caso de uso Interessados Notificados

UC04 **Interessados Notificados** **Ator:** Professor, Aluno **Cenário Principal** **Descrição:** Notificar interessados		
ID	**Ação do Usuário**	**Resposta do Sistema**
CP.01	[CA.01]	

O Quadro 29 apresenta uma versão atualizada do conhecimento de configuração do *MyCourses* após a elaboração dos cenários alternativos acima.

Quadro 29 V4 Conhecimento de Configuração do *MyCourses*

Expressão de Feature	**Transformação**
Agendamento de Curso	*select scenario* UC01.CP
Agendar Curso Manualmente	*select scenario* ITD01 UC01.CP
Agendar Curso	*select scenario* ITD02 UC01.CP
Reservar Manualmente	*select scenario* ITD03 UC05.CP
Gerenciar Manualmente	*select scenario* ITD04 UC06.CP
Exportar para XML	*select scenario* ITD05 UC03.CP
Exportar para SQL	*select scenario* ITD06 UC03.CP
Manter Recurso Ocupado	*select scenario* CA01 UC05.CP
Manter Recurso Livre	*select scenario* CA02 UC05.CP
Gerenciar Conflito de Membros	*select scenario* CA01 UC06.CP
Gerenciar Conflito de Recurso	*select scenario* CA02 UC06.CP
Comunicar Alterações	*select scenario* CA01 UC04.CP
Exportar para PDF	*select scenario* CA01 UC03.CP
Homologar Agendamento	*select scenario* CA01 UC02.CP

A **Diretriz 6.4.4** não se aplica ao exemplo *MyCourses*, pois o seu modelo SR não apresenta uma ligação meio-fim com cardinalidade cujo elemento raiz seja filho de elemento opcional da LPS.

Aplicando as diretrizes **6.5** e **6.5.1** ao *MyCourses*, obtêm-se uma nova versão do cenário previamente identificado, conforme apresentado na Figura 119.

Figura 119 Versão 2 do *ITD* Agendar Curso

ITD02 **Descrição:** Agendar curso		
ID	**Ação do Usuário**	**Resposta do Sistema**
ITD02.01	@incluir UC05 Reserva de Recurso	
ITD02.02	@incluir UC06 Gerenciamento de Conflito	

4.2.7. Refinamento dos cenários

Esta atividade tem como objetivo preencher as lacunas das descrições dos cenários e *advices* elaborados na atividade anterior. As figuras 120-138 ilustram as descrições parciais dos cenários, *intertype declarations* e *advices* após a execução desta atividade.

Figura 120 Caso de uso Agendamento de Curso

UC01 **Agendamento de Curso** **Ator:** Gerente de Programa, Professor **Cenário Principal 01** **Descrição:** Agendar Curso		
ID	**Ação do Usuário**	**Resposta do Sistema**
CP01.01		

Figura 121 Caso de uso Homologação de Agendamento

UC02 Homologação de Agendamento **Ator:** Gerente de Programa **Cenário Principal** **Descrição:** Homologar agendamento		
ID	**Ação do Usuário**	**Resposta do Sistema**
CP.01	[CA.01]	

Figura 122 Caso de uso Reserva de Recurso

UC05 **Reserva de Recurso** **Ator:** Gerente de Programa, Professor **Cenário Principal** **Descrição:** Reservar recursos		
ID	**Ação do Usuário**	**Resposta do Sistema**
CP.01	[CA.01] [CA.02]	

Figura 123 Caso de uso Gerenciamento de Conflito

UC06 **Gerenciamento de Conflito** **Ator:** Gerente de Programa, Professor **Cenário Principal** **Descrição:** Gerenciar conflito		
ID	**Ação do Usuário**	**Resposta do Sistema**
CP.01	[CA.01] [CA.02]	

Figura 124 Caso de uso Interessados Notificados

UC04 **Interessados Notificados** **Ator:** Professor, Aluno **Cenário Principal** **Descrição:** Notificar interessados		
ID	**Ação do Usuário**	**Resposta do Sistema**
CP.01	[CA.01]	

Figura 125 Caso de uso Exportação do Agendamento

UC03 **Exportação do Agendamento** **Ator:** Gerente de Programa, Professor, Administrador do Sistema, Aluno **Cenário Principal** **Descrição:** Exportar agendamento		
ID	**Ação do Usuário**	**Resposta do Sistema**
CP.01	[CA.01]	

Figura 126 *Intertype declaration* Agendar Manualmente

ITD01 **Descrição:** Agendar curso manualmente		
ID	**Ação do Usuário**	**Resposta do Sistema**
ITD01.01	-	Curso atualizado de acordo com os interesses do usuário

Figura 127 *Intertype declaration* Agendar Curso

ITD02 **Descrição:** Agendar curso		
ID	**Ação do Usuário**	**Resposta do Sistema**
ITD02.01	@incluir UC10 Reserva de Recurso	-
ITD02.02	@incluir UC11 Gerenciamento de Conflito	-

Figura 128 *Intertype declaration* Reservar Manualmente

ITD03 **Descrição:** Reservar manualmente		
ID	**Ação do Usuário**	**Resposta do Sistema**
ITD03.01	Selecionar recurso	-
ITD03.02	Inserir informações sobre horários de utilização do recurso	Verificar disponibilidade do recurso para os parâmetros informados

Figura 129 *Intertype declaration* Gerenciar Manualmente

ITD04 **Descrição:** Gerenciar manualmente		
ID	**Ação do Usuário**	**Resposta do Sistema**
ITD04.01	-	Aloca membros e recursos de acordo com os interesses do usuário

Figura 130 *Intertype declaration* Exportar para XML

ITD05 **Descrição:** Exportar dados de agendamento para XML		
ID	**Ação do Usuário**	**Resposta do Sistema**
ITD05.01	Exportar dados do agendamento	Dados exportados para XML

Figura 131 *Intertype declaration* Exportar para SQL

ITD06 **Descrição:** Exportar dados de agendamento para SQL		
ID	**Ação do Usuário**	**Resposta do Sistema**
ITD06.01	Exportar dados do agendamento	Dados exportados para SQL

Figura 132 Cenário alternativo Manter Recurso Ocupado

UC05 Reserva de Recurso **Ator:** Gerente de Programa, Professor **Cenário Alternativo 01** **Descrição:** Manter recurso ocupado		
ID	**Ação do Usuário**	**Resposta do Sistema**
CA01.01	-	Manter os recursos a maior parte do tempo em utilização

Figura 133 Cenário alternativo Manter Recurso Livre

UC05 Reserva de Recurso **Ator:** Gerente de Programa, Professor **Cenário Alternativo 02** **Descrição:** Manter recurso livre		
ID	**Ação do Usuário**	**Resposta do Sistema**
CA02.01	-	Manter os recursos a maior parte do tempo disponíveis

Figura 134 Cenário alternativo Gerenciar Conflito de Membros

UC06 Gerenciamento de Conflito **Ator:** Gerente de Programa, Professor **Cenário Alternativo 01** **Descrição:** Gerenciar conflito de membros		
ID	**Ação do Usuário**	**Resposta do Sistema**
CA01.01	-	Verificar a disponibilidade do membro

Figura 135 Cenário alternativo Gerenciar Conflito de Recurso

UC06 Gerenciamento de Conflito **Ator:** Gerente de Programa, Professor **Cenário Alternativo 02** **Descrição:** Gerenciar conflito de recurso		
ID	**Ação do Usuário**	**Resposta do Sistema**
CA02.01	-	Verificar disponibilidade do recurso

Figura 136 Cenário alternativo Exportar Dados para PDF

UC03 Exportação do Agendamento **Ator:** Gerente de Programa, Professor, Administrador do Sistema, Aluno **Cenário Alternativo 01** **Descrição:** Exportar dados para PDF		
ID	**Ação do Usuário**	**Resposta do Sistema**
CA01.01	-	Exportar dados do agendamento para PDF

Figura 137 Cenário alternativo Homologar Agendamento

UC02 Homologação de Agendamento **Ator:** Gerente de Programa **Cenário Alternativo 01** **Descrição:** Homologar agendamento		
ID	**Ação do Usuário**	**Resposta do Sistema**
CA01.01	Verificar agendamentos	Homologar agendamentos

Figura 138 Cenário alternativo Comunicar Alterações

UC04 Interessados Notificados **Ator:** Professor, Aluno **Cenário Alternativo 01** **Descrição:** Comunicar alterações		
ID	**Ação do Usuário**	**Resposta do Sistema**
CA01.01	-	Verifica se houve homologação ou alteração no curso que o usuário está inscrito
CA01.01	-	Usuário notificado

4.2.8. Configuração do Produto

4.2.8.1. Escolha de Configuração Específica

Esta atividade tem o objetivo de permitir ao *interessado* escolher, de acordo com as suas necessidades, as funcionalidades que farão parte do produto utilizando o modelo SR. É possível que exista mais de uma configuração, por isso, todas as variantes que atendem as necessidades dos interessados devem ser configuradas em modelos SR e classificadas de acordo com as prioridades dos objetivos-*soft*.

Para ilustrar esta atividade, no *MyCourses*, utilizaremos a configuração de um produto realizada por um cliente fictício, pois não tivemos contato com interessados que pudessem auxiliar na realização desta atividade da abordagem.

Suponhamos que o usuário queira executar um agendamento de curso no *MyCourses* que não possua as opção de reservar manualmente, pois este cliente deseja que o *MyCourses* realize todo o trabalho para gerenciar conflitos e recursos. O *MyCourses* deve gerenciar conflitos de recursos. Além de analisar a disponibilidade dos membros evitando conflitos. Os dados devem ser exportados apenas para o formato PDF.

A partir da especificação acima, elaboram-se os modelos SR de cada variante. É possível que haja quatro configurações, a primeira, com a *feature* Reservar Manualmente e sem a *feature* Gerenciar Manualmente; a segunda, com a *feature* Gerenciar Manualmente e sem a *feature* Reservar Manualmente; a terceira, sem as features Gerenciar Manualmente e Reservar Manualmente e, por fim, a quarta variante, contendo as *features* Gerenciar Manualmente e Reservar Manualmente. Os modelos SR das variantes são apresentados nas figuras: Figura 139, Figura 140, Figura 141 e Figura 142.

Figura 139 Modelo SR da variante (V1) do MyCourses

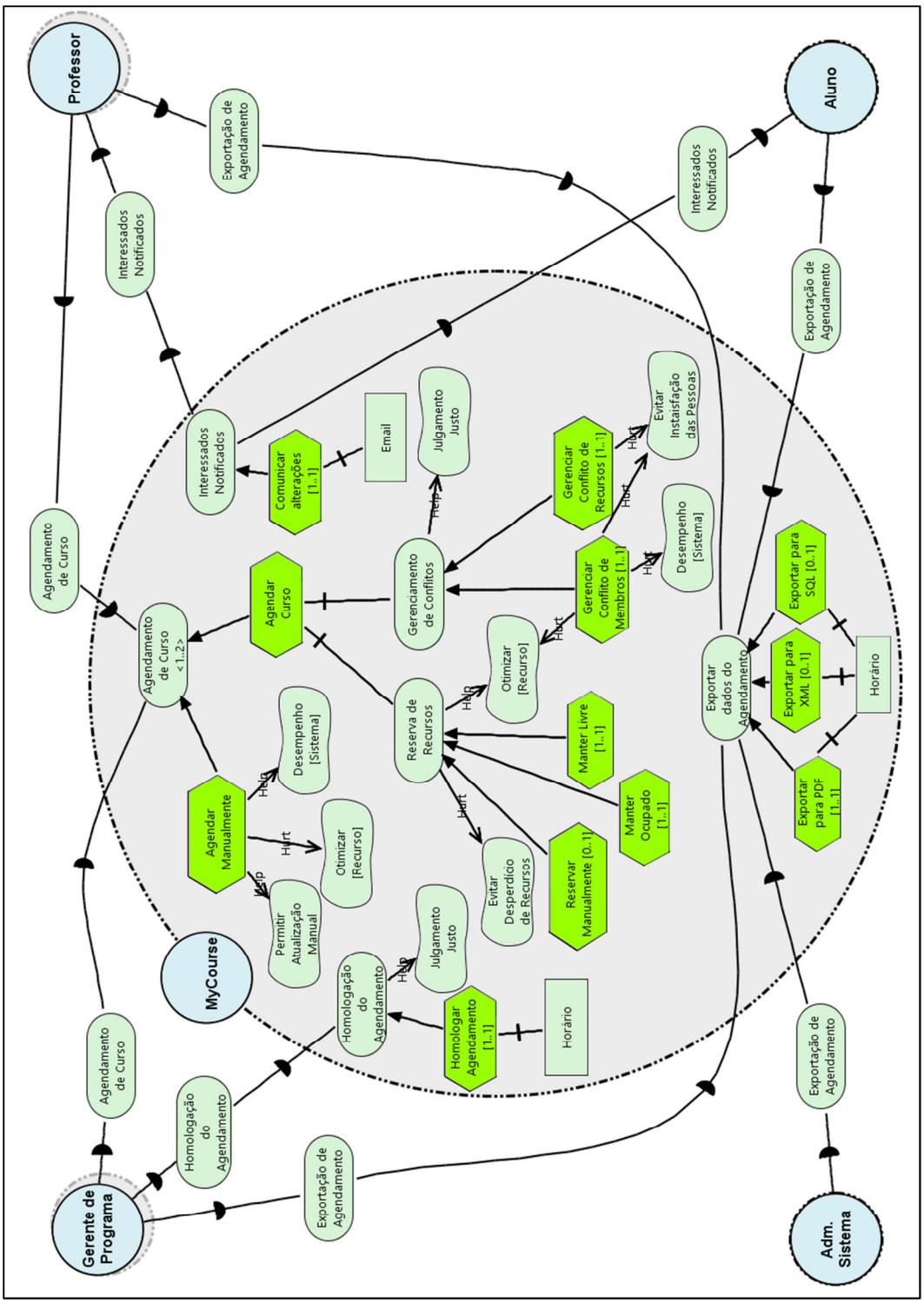


Figura 140 Modelo SR da variante (V2) do MyCourses

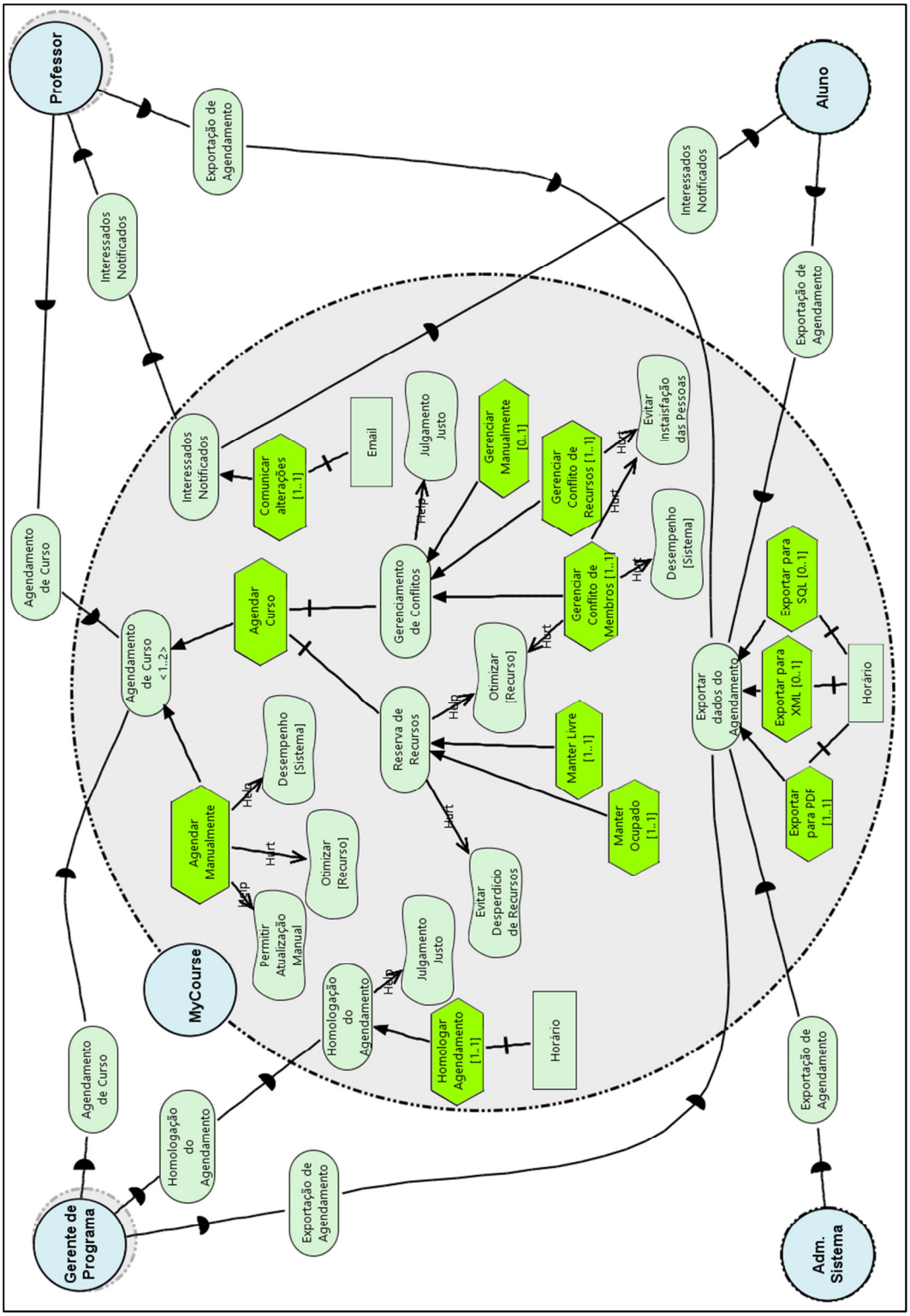


Figura 141 Modelo SR da variante (V3) do MyCourses

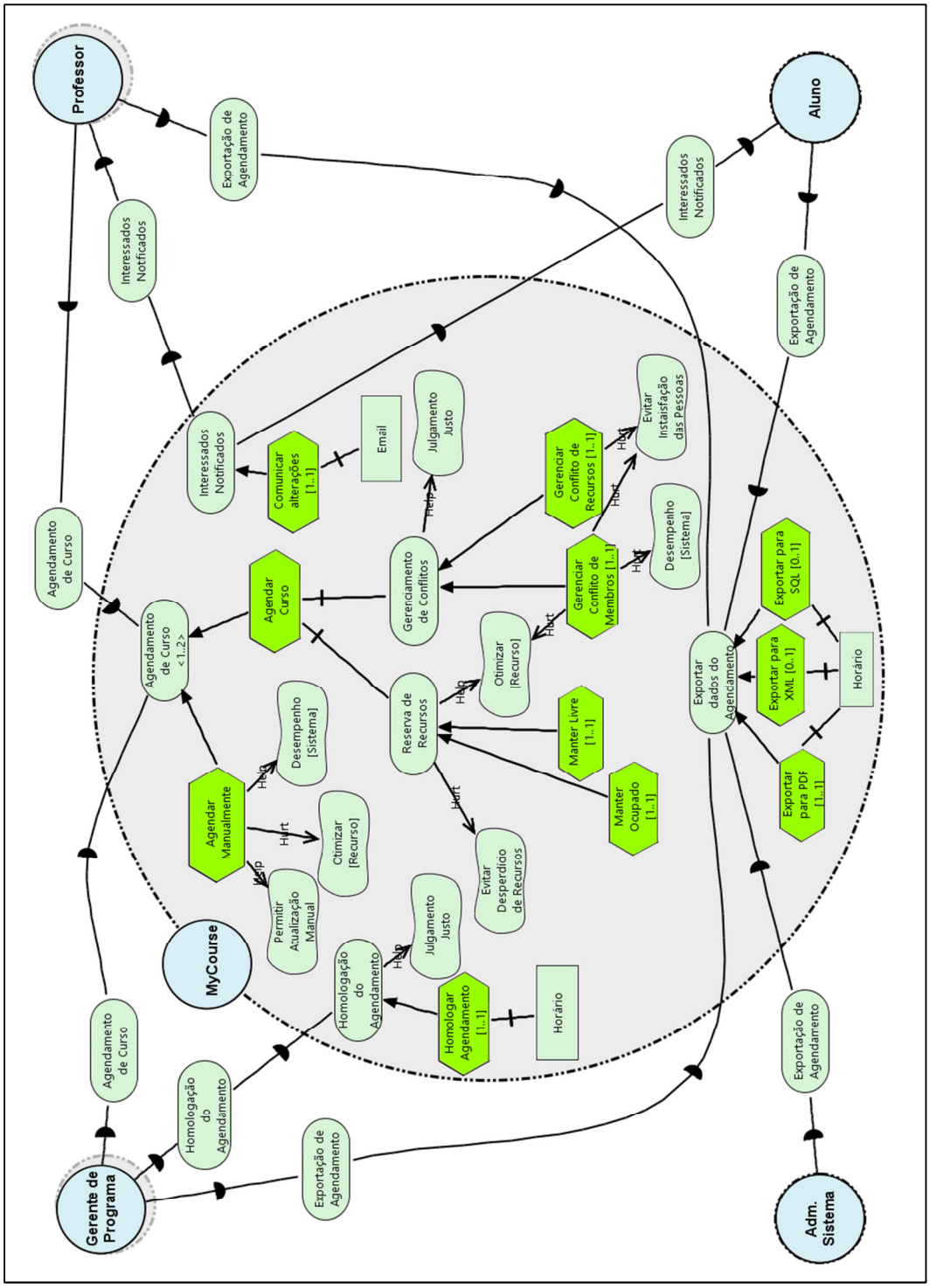
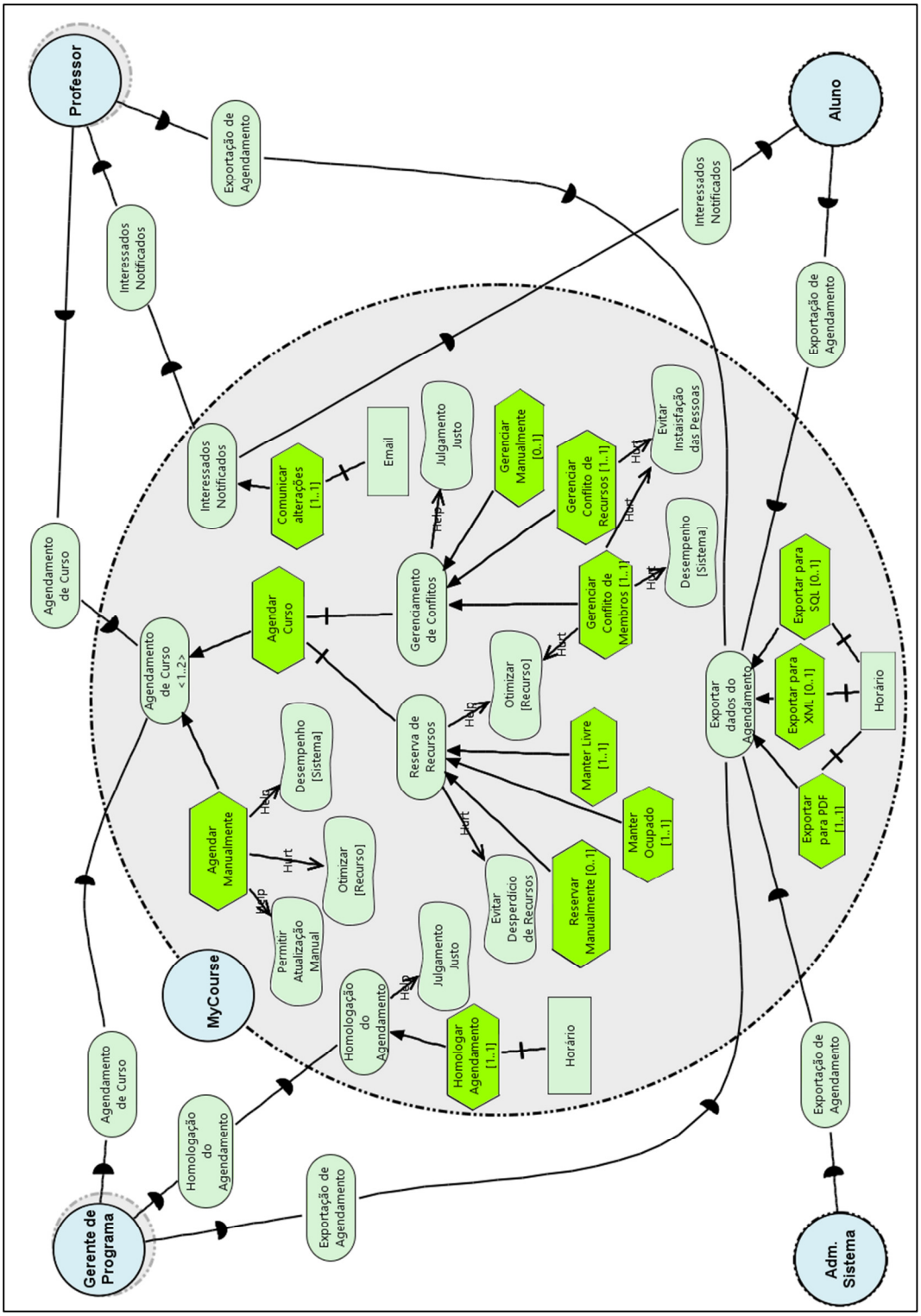


Figura 142 Modelo SR da variante (V4) do MyCourses



4.2.8.2. Priorização de Variantes

O interessado atribui valores para cada objetivo-*soft* de acordo com a prioridade que estes têm para o sistema que está sendo desenvolvido. A prioridade atribuída para cada objetivo-*soft* é mostrada no Quadro 30.

Quadro 30 Prioridade dos objetivos-soft para o TaRGeT

Objetivos-*soft*	**Prioridade [0, 10]**
Desempenho [Sistema] (DES)	10
Otimizar [Recurso] (OTR)	10
Evitar Insatisfação das Pessoas (EIP)	5
Julgamento Justo (JUJ)	5
Evitar Desperdício de Recurso (EDR)	5
Permitir Atualização Manual (PAM)	5

O cálculo da priorização das variantes é o seguinte:

$$\begin{aligned} priority\ (V1) &= [(percentPos(V1,PAM) \times priority(PAM)) \\ &+ 2 \times (percentPos(V1,JUJ) \times priority(JUJ)) \\ &+ 3 \times (percentPos(V1,OTR) \times priority(OTR))] \\ &- [(percentNeg(V1,OTR) \times priority(OTR)) \\ &+ (percentNeg(V1,EDR) \times priority(EDR))] \\ &+ 2 \times (percentPos(V1,EIP) \times priority(EIP)) \\ &+ 3 \times (percentPos(V1,DES) \times priority(DES))] \\ &- [2 \times (percentNeg(V1,DES) \times priority(DES))] \end{aligned}$$

$$\begin{aligned} priority\,(V1) &= [(\tfrac{1}{11} \times 0,75 \times 5) + (2 \times \tfrac{2}{11} \times 0,75 \times 5) + (3 \times \tfrac{3}{11} \times 0,75 \times 10)] - [(\tfrac{1}{4} \times 0,75 \times 10) \\ &+ (\tfrac{1}{4} \times 0,75 \times 5)] + [(2 \times \tfrac{2}{11} \times 0,75 \times 5) + (3 \times \tfrac{3}{11} \times 0,75 \times 10)] - (2 \times \tfrac{2}{4} \times 0,75 \times 10)] \\ &= [7,82 - 5,62 + 7,48 - 7,5] = 2,18 \end{aligned}$$

$$\begin{aligned} priority\ (V2) &= [(percentPos(V2,PAM) \times priority(PAM)) \\ &+ 2 \times (percentPos(V2,JUJ) \times priority(JUJ)) \\ &+ 3 \times (percentPos(V2,OTR) \times priority(OTR))] \\ &- [(percentNeg(V2,OTR) \times priority(OTR)) \\ &+ (percentNeg(V2,EDR) \times priority(EDR)) \\ &+ (percentNeg(V2,EIP) \times priority(EIP))] \\ &+ [2 \times (percentPos(V2,EIP) \times priority(EIP)) \\ &+ 3 \times (percentPos(V2,DES) \times priority(DES))] \\ &- [2 \times (percentNeg(V2,DES) \times priority(DES))] \end{aligned}$$

$$\begin{aligned} priority\,(V2) &= [(\tfrac{1}{11} \times 0,75 \times 5) + (2 \times \tfrac{2}{11} \times 0,75 \times 5) + (3 \times \tfrac{3}{11} \times 0,75 \times 10)] - [(\tfrac{1}{5} \times 0,75 \times 10) \\ &+ (\tfrac{1}{5} \times 0,75 \times 5)] + (\tfrac{1}{5} \times 0,75 \times 5)] + [(2 \times \tfrac{2}{11} \times 0,75 \times 5) + (3 \times \tfrac{3}{11} \times 0,75 \times 10)] - (2 \times \tfrac{2}{5} \times 0,75 \times 10)] \\ &= [7,82 - 3,0 + 7,48 - 6,0] = 6,3 \end{aligned}$$

$$
\begin{aligned}
priority\,(V3) &= [(percentPos(V3,PAM) \times priority(PAM)) \\
&+ 2 \times (percentPos(V3,JUJ) \times priority(JUJ)) \\
&+ 3 \times (percentPos(V3,OTR) \times priority(OTR))] \\
&- [(percentNeg(V3,OTR) \times priority(OTR)) \\
&+ (percentNeg(V3,EDR) \times priority(EDR)) \\
&+ (percentNeg(V3,EIP) \times priority(EIP))] \\
&+ 3 \times (percentPos(V3,DES) \times priority(DES))] \\
&- [2 \times (percentNeg(V3,DES) \times priority(DES))]
\end{aligned}
$$

$$
\begin{aligned}
priority\,(V3) &= [(\frac{1}{11} \times 0,75 \times 5) + (2 \times \frac{2}{11} \times 0,75 \times 5) + (3 \times \frac{3}{11} \times 0,75 \times 10)] - [(\frac{1}{3} \times 0,75 \times 10) \\
&+ (\frac{1}{3} \times 0,75 \times 5)] + [(\frac{2}{11} \times 0,75 \times 5)] + +(3 \times \frac{3}{11} \times 0,75 \times 10)] - (\frac{1}{3} \times 0,75 \times 10)] \\
&= [7,82 - 3,75 + 6,8 - 2,5] = 8,37
\end{aligned}
$$

$$
\begin{aligned}
priority\,(V4) &= [(percentPos(V4,PAM) \times priority(PAM)) \\
&+ 2 \times (percentPos(V4,JUJ) \times priority(JUJ)) \\
&+ 3 \times (percentPos(V4,OTR) \times priority(OTR))] \\
&- [(percentNeg(V4,OTR) \times priority(OTR)) \\
&+ (percentNeg(V4,EDR) \times priority(EDR)) \\
&+ (percentNeg(V4,EIP) \times priority(EIP))] \\
&+ [3 \times (percentPos(V4,DES) \times priority(DES)) \\
&+ (percentPos(V4,EIP) \times priority(EIP))] \\
&- [3 \times (percentNeg(V4,DES) \times priority(DES))]
\end{aligned}
$$

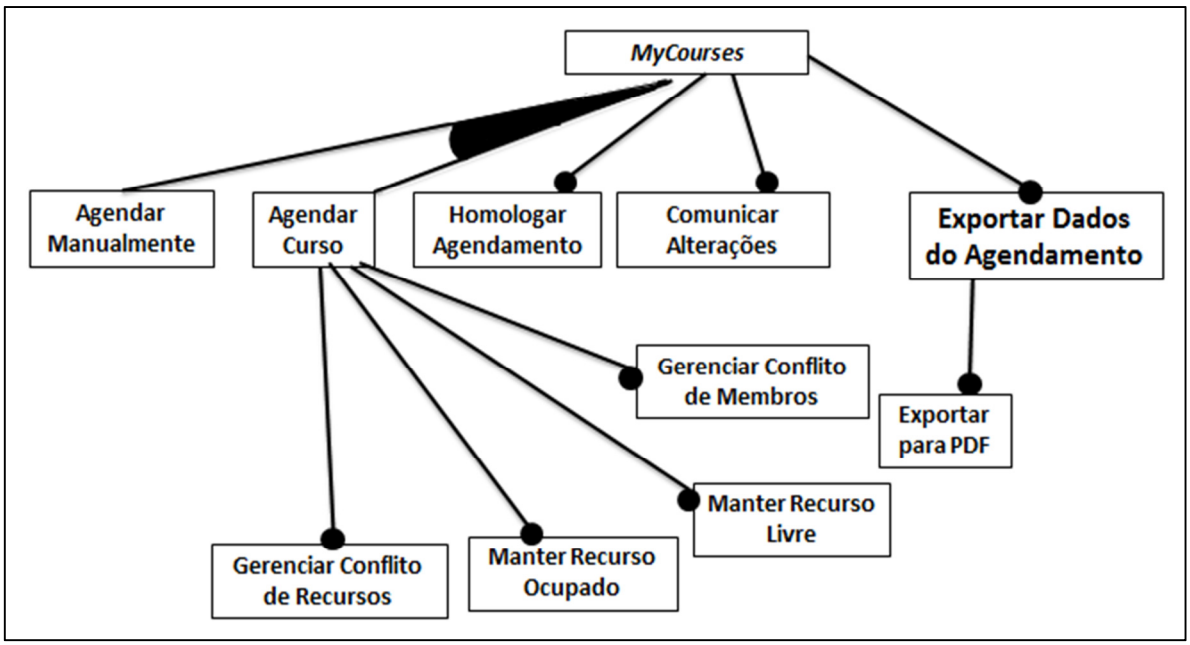
$$
\begin{aligned}
priority\,(V4) &= [(\frac{1}{11} \times 0,75 \times 5) + (2 \times \frac{2}{11} \times 0,75 \times 5) + (3 \times \frac{3}{11} \times 0,75 \times 10)] - [(\frac{1}{6} \times 0,75 \times 10) \\
&+ (\frac{1}{6} \times 0,75 \times 5) + (\frac{1}{6} \times 0,75 \times 5)] + (3 \times \frac{3}{11} \times 0,75 \times 10) + (\frac{2}{11} \times 0,75 \times 5)] - (3 \times \frac{3}{6} \times 0,75 \times 10)] \\
&= [7,82 - 2,49 + 6,8 - 11,25] = 0,88
\end{aligned}
$$

Analisando o resultado dos cálculos de priorização, a variante V3 (Figura 141) com valor 8,37 é a melhor escolha para o interessado que deseja prioridade máxima para os objetivo-*soft* Desempenho [Sistema] e Otimizar [Recurso].

4.2.8.3. Configuração dos Artefatos do Produto

Após a variante ter sido escolhida, o próximo passo é elaborar os artefatos para o produto específico. O primeiro artefato que deve ser elaborado é o modelo de configuração do produto, ou seja, o modelo de *feature*. Deve-se analisar o modelo SR da variante escolhida para extrair as *features*. Para identificar as features associadas ao modelo SR consulta-se o quadro de mapeamento entre elementos e *features* (Quadro 25). A Figura 143 apresenta o modelo de configuração do produto.

Figura 143 Modelo de *feature* do produto do *MyCourses*



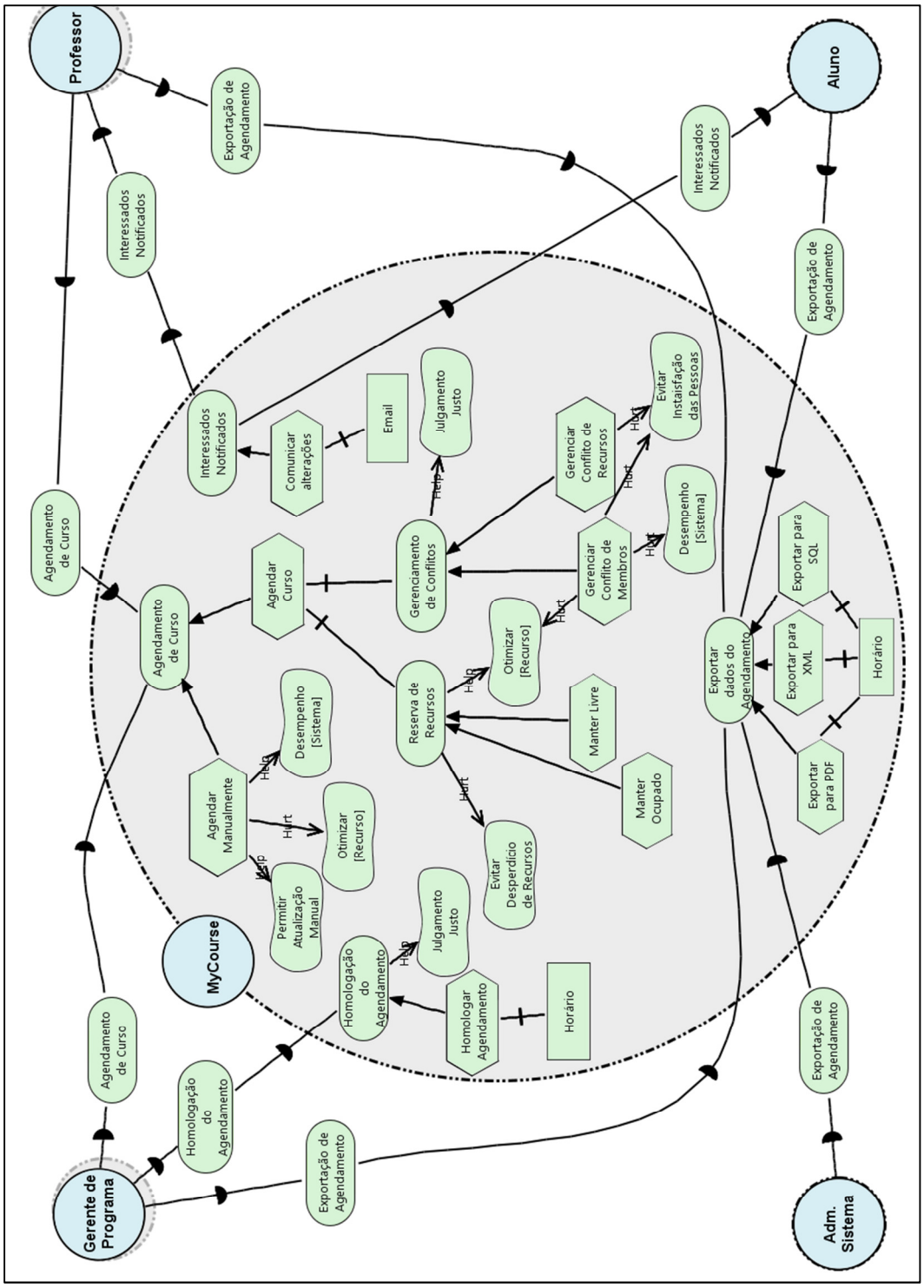
Em seguida, deve-se remover a cardinalidade do modelo de objetivos da variante e os destaques para identificar os elementos que originaram *features*. Assim, será possível manter o rastreamento entre os objetivos dos interessados e os requisitos do produto. A Figura 144 apresenta o modelo SR do produto.

Para elaborar o conhecimento de configuração do produto deve-se analisar o modelo de *feature* para determinar se a *feature* está presente no modelo de configuração do produto. O Quadro 31 apresenta o conhecimento de configuração elaborado para o produto do *MyCourses*.

Quadro 31 Conhecimento de Configuração do produto do *MyCourses*

Expressão de Feature	Transformação
Agendamento de Curso	*select scenario* UC01.CP
Agendar Manualmente	*select scenario* ITD01 UC01.CP
Agendar Curso	*select scenario* ITD02 UC01.CP
Manter Recurso Ocupado	*select scenario* CA01 UC05.CP
Manter Recurso Livre	*select scenario* CA02 UC05.CP
Gerenciar Conflito de Membros	*select scenario* CA01 UC06.CP
Gerenciar Conflito de Recurso	*select scenario* CA02 UC06.CP
Comunicar Alterações	*select scenario* CA01 UC04.CP
Exportar para PDF	*select scenario* CA02 UC03.CP
Homologar Agendamento	*select scenario* CA01 UC02.CP

Figura 144 Modelo SR do produto do MyCourses



Após a elaboração do conhecimento de configuração, deve-se verificar se a expressão de *feature* está relacionada ao cenário, *intertype declaration* ou *advice* visando identificar se o mesmo estará presente na especificação do produto. Os cenários do produto são apresentados nas figuras 145-159.

Figura 145 Caso de uso Agendamento de Curso

UC01 **Agendamento de Curso** **Ator:** Gerente de Programa, Professor **Cenário Principal 01** **Descrição:** Agendar Curso		
ID	**Ação do Usuário**	**Resposta do Sistema**
CP01.01		

Figura 146 *Intertype declaration* Agendar Manualmente

ITD01 **Descrição:** Agendar curso manualmente		
ID	**Ação do Usuário**	**Resposta do Sistema**
ITD01.01	-	Curso atualizado de acordo com os interesses do usuário

Figura 147 *Intertype declaration* Agendar Curso

ITD02 **Descrição:** Agendar curso		
ID	**Ação do Usuário**	**Resposta do Sistema**
ITD02.01	@incluir UC10 Reserva de Recurso	-
ITD02.02	@incluir UC11 Gerenciamento de Conflito	-

Figura 148 Caso de uso Homologação de Agendamento

UC02 Homologação de Agendamento **Ator:** Gerente de Programa **Cenário Principal** **Descrição:** Homologar agendamento		
ID	**Ação do Usuário**	**Resposta do Sistema**
CP.01	[CA.01]	

Figura 149 Caso de uso Reserva de Recurso

UC05 **Reserva de Recurso** **Ator:** Gerente de Programa, Professor **Cenário Principal** **Descrição:** Reservar recursos		
ID	**Ação do Usuário**	**Resposta do Sistema**
CP.01	[CA.01] [CA.02]	

Figura 150 Caso de uso Gerenciamento de Conflito

UC06 **Gerenciamento de Conflito** **Ator:** Gerente de Programa, Professor **Cenário Principal** **Descrição:** Gerenciar conflito		
ID	**Ação do Usuário**	**Resposta do Sistema**
CP.01	[CA.01] [CA.02]	

Figura 151 Caso de uso Interessados Notificados

UC04 **Interessados Notificados** **Ator:** Professor, Aluno **Cenário Principal** **Descrição:** Notificar interessados		
ID	**Ação do Usuário**	**Resposta do Sistema**
CP.01	[CA.01]	

Figura 152 Caso de uso Exportação do Agendamento

UC03 **Exportação do Agendamento** **Ator:** Gerente de Programa, Professor, Administrador do Sistema, Aluno **Cenário Principal** **Descrição:** Exportar agendamento		
ID	**Ação do Usuário**	**Resposta do Sistema**
CP.01	[CA.01]	

Figura 153 Cenário alternativo Manter Recurso Ocupado

UC05 Reserva de Recurso **Ator:** Gerente de Programa, Professor **Cenário Alternativo 01** **Descrição:** Manter recurso ocupado		
ID	**Ação do Usuário**	**Resposta do Sistema**
CA01.01	-	Manter os recursos a maior parte do tempo em utilização

Figura 154 Cenário alternativo Manter Recurso Livre

UC05 Reserva de Recurso **Ator:** Gerente de Programa, Professor **Cenário Alternativo 02** **Descrição:** Manter recurso livre		
ID	**Ação do Usuário**	**Resposta do Sistema**
CA02.01	-	Manter os recursos a maior parte do tempo disponível

Figura 155 Cenário alternativo Gerenciar Conflito de Membros

UC06 Gerenciamento de Conflito **Ator:** Gerente de Programa, Professor **Cenário Alternativo 01** **Descrição:** Gerenciar conflito de membros		
ID	**Ação do Usuário**	**Resposta do Sistema**
CA01.01	-	Verificar a disponibilidade do membro

Figura 156 Cenário alternativo Gerenciar Conflito de Recurso

UC06 Gerenciamento de Conflito **Ator:** Gerente de Programa, Professor **Cenário Alternativo 02** **Descrição:** Gerenciar conflito de recurso		
ID	**Ação do Usuário**	**Resposta do Sistema**
CA02.01	-	Verificar disponibilidade do recurso

Figura 157 Cenário alternativo Exportar Dados para PDF

UC03 Exportação do Agendamento **Ator:** Gerente de Programa, Professor, Administrador do Sistema, Aluno **Cenário Alternativo 01** **Descrição:** Exportar dados para PDF		
ID	**Ação do Usuário**	**Resposta do Sistema**
CA01.01	-	Exportar dados do agendamento para PDF

Figura 158 Cenário alternativo Homologar Agendamento

UC02 Homologação de Agendamento		
Ator: Gerente de Programa **Cenário Alternativo 01** **Descrição:** Homologar agendamento		
ID	**Ação do Usuário**	**Resposta do Sistema**
CA01.01	Verificar agendamentos	Homologar agendamentos

Figura 159 Cenário alternativo Comunicar Alterações

UC04 Interessados Notificados **Ator:** Professor, Aluno **Cenário Alternativo 01** **Descrição:** Comunicar alterações		
ID	**Ação do Usuário**	**Resposta do Sistema**
CA01.01	-	Verifica se houve homologação ou alteração no curso que o usuário está inscrito
CA01.01	-	Usuário notificado

Capítulo 5 Avaliação da Proposta

Este capítulo apresenta a avaliação da abordagem GAS2SPL, na qual foram definidos os objetivos da avaliação e as métricas usadas para avaliar se os objetivos foram alcançados. Assim, os artefatos gerados pela abordagem GS2SPL são comparados com os artefatos gerados pela abordagem definida nesta dissertação com relação as métricas de avaliação consideradas.

5.1. Método de Avaliação

Segundo Wohlin et al. (2000) existem quatro métodos relevantes para condução de experimentos na área de Engenharia de Software: científico, de engenharia, experimental e analítico.

Segundo Travassos (2002), o método científico tenta extrair do mundo algum modelo que possa explicar um fenômeno, e avaliar se o modelo é realmente representativo para o fenômeno que está sob observação.

O método de engenharia observa as soluções existentes, sugere as mais adequadas, desenvolve, mede, analisa e promove melhoria nas soluções, e repete o processo até que nenhuma melhoria adicional seja possível.

O método experimental sugere o modelo, desenvolve o método qualitativo e/ou quantitativo, aplica um experimento, mede e analisa, avalia o modelo e repete o processo. O processo se inicia com o levantamento de um modelo novo, não necessariamente baseado em um modelo já existente, e tenta estudar o efeito do processo ou produto sugerido pelo novo modelo.

O método analítico sugere uma teoria formal, desenvolve a teoria, deriva os resultados e se possível, compara-a com as observações empíricas.

Considerando estes aspectos, este trabalho utilizará o método de engenharia para sua avaliação. Iremos comparar os artefatos gerados pela abordagem GS2SPL e pela nossa abordagem (GAS2SPL) visando determinar se há diferenças significantes nos cenários de caso de uso obtidos com relação à modularidade e expressividade. A comparação será feita do ponto de vista do analista de requisitos, considerando os artefatos gerados para os exemplos *MyCourses* e do *TaRGeT*. Para isto, serão utilizadas métricas para a modularidade de cenários de caso de uso da LPS e métricas para quantificar o detalhamento do conhecimento de configuração (expressividade).

A modularidade de cenários de caso de uso da LPS está interessada em garantir que as *features* estejam encapsuladas em módulos separados (em artefatos de requisitos, os módulos considerados são os cenários) [Alferez et al. 2013]. Nesta avaliação, a modularidade será analisada levando-se em consideração os conceitos da orientação a aspectos, pois, segundo Alferez et al. (2013) a orientação a aspectos reduz o espalhamento e o entrelaçamento das *features* nos cenários da LPS. A expressividade está relacionada a quão fácil uma especificação é escrita e lida, ou entendida [Alferez et al. 2013]. Nesta avaliação, a expressividade do conhecimento de configuração será medida pela quantidade de símbolos necessários para especificar o conhecimento de configuração. Para avaliar a expressividade do conhecimento de configuração na abordagem GS2SPL, analisamos as referências a *features* nos passos dos cenários. Em relação à métrica de expressividade, assim como em Alferez et al. (2013), quanto menos símbolos forem utilizados para especificar o conhecimento de configuração, melhor a expressividade do conhecimento de configuração.

Alferez et al. (2013) realizou uma comparação entre algumas técnicas de especificação de cenários, dentre elas, o PLUSS e MSVCM. No entanto, os cenários gerados nessas técnicas foram obtidos diretamente dos interessados (do inglês, *stakeholders*). A abordagem MSVCM produziu especificações de cenários mais modulares e um conhecimento de configuração mais expressivo. A nossa abordagem obtém os cenários a partir de modelos de objetivos, assim, precisamos avaliar se a modularidade de MSVCM continua sendo maior nesse caso.

Portanto, para realizar a avaliação, definiram-se as seguintes questões de pesquisa:

Q1 - Os cenários de caso de uso gerados pela abordagem GAS2SPL são mais modularizados do que os criados pela abordagem GS2SPL?

Q2 – A expressividade das especificações gerada pela abordagem GAS2SPL é maior do que as geradas pela abordagem GS2SPL?

Destas questões de pesquisa, elaboraram-se as seguintes hipóteses:

Hipótese nula (H_o1.1): O grau de modularidade dos cenários de caso de uso gerados pela abordagem GAS2SPL é igual ao grau de modularidade dos cenários criados pela abordagem GS2SPL.

Hipótese alternativa (H1): O grau de modularidade dos cenários de caso de uso gerados pela abordagem GAS2SPL é menor que o grau de modularidade dos cenários criados pela abordagem GS2SPL.

Hipótese nula ($H_0$2.1): A expressividade das especificações geradas pela abordagem GAS2SPL é igual do que a das especificações geradas pela abordagem GS2SPL.

Hipótese alternativa (H2): A expressividade das especificações geradas pela abordagem GAS2SPL é menor do que a das especificações geradas pela abordagem GS2SPL

As métricas utilizadas na avaliação da abordagem são apresentadas, de forma resumida, a seguir.

Assim como Almeida (2010), para analisar a modularidade dos artefatos gerados utilizaremos as métricas de espalhamento de features (DoS) e entrelaçamento de cenários (DoT), adaptadas de Eaddy et al. (2007).

De acordo com as fórmulas (6.1) e (6.2), o grau de espalhamento de *features* (do inglês, *degree of scattering of features* - DoS), adaptado de Almeida (2010), quantifica a concentração de uma *feature* sob cada cenário $s \in S$ (sendo S, o conjunto de especificações de cenários). Os valores de DoS estão entre 0 e 1, sendo 0 (completamente localizado/concentrado) e 1 (completamente espalhado). Quanto maior o valor de DoS de uma *feature f*, maior a probabilidade de ter que reorganizar diferentes cenários quando a especificação da *feature* mudar. Na equação (6.1) o $|S|>1$ e significa a cardinalidade do conjunto S, ou seja, em quantos cenários a *feature* está presente.

$$(6.1)\ DoS(f) = 1 - \frac{|S| \sum_{s \epsilon S} (Conc(f,s) - \frac{1}{|S|})^2}{|S| - 1}$$

$$(6.2)\ Conc(f,s) = \frac{número\ de\ passos\ em\ s\ atribuído\ a\ f}{número\ de\ passos\ atribuído\ a\ f}$$

De acordo com as fórmulas (6.3) e (6.4), o grau de entrelaçamento dos cenários (do inglês, *degree of tangling of scenarios* - DoT), adaptado de Almeida (2010), considera quantos passos de um cenário são relacionados a cada *feature* $f \in F$ (sendo F, o conjunto de *features*). Assim como no grau de espalhamento, os valores de DoT estão entre 0 e 1, sendo 0 (completamente focado) e 1 (completamente entrelaçado). Quanto maior o valor de DoT de um cenário *s*, maior a probabilidade de ter que reorganizar um cenário quando uma *feature* relacionada mudar. Quando quisermos analisar os valores da modularidade, utilizaremos o grau de foco (do inglês, *degree of focus* - DoF) que corresponde a 1-DoT.

Portanto, cenários com altos valores de DoF e baixos valores de DoT, correspondem a sistemas bem modularizados [Alferez et al. 2013].

$$(6.3)\quad DoT(s) = 1 - \frac{|F| \sum_{f \epsilon F} (Dedi(s, f) - \frac{1}{|F|})^2}{|F| - 1}$$

$$(6.4)\quad Dedi(s, f) = \frac{número\ de\ passos\ em\ s\ atribuído\ a\ f}{número\ de\ passos\ de\ todas\ as\ features\ em\ s}$$

Para medir a expressividade utilizaremos uma métrica para medir a expressividade do conhecimento de configuração. A expressividade do conhecimento de configuração é realizada através da contagem de símbolos necessários para mapear *features* para outros modelos. Por exemplo, em PLUSS, há referências a *features* nos passos do cenário. No caso do MSVCM, é realizada a contagem dos símbolos utilizados no artefato de conhecimento de configuração [Alferez et al. 2013].

5.2. Comparação entre os artefatos: GAS2SPL x GS2SPL

Utilizaremos os artefatos do TaRGeT gerados no Capítulo 3 elaborados utilizando a abordagem GAS2SPL e, os artefatos do trabalho de Souza (2012) definidos através do processo da abordagem GS2SPL.

Para o *MyCourses*, utilizaremos os artefatos gerados no Capítulo 4 elaborados utilizando a abordagem GAS2SPL e os artefatos elaborados pelo autor deste trabalho a partir das descrições da seção 4.1 utilizando a abordagem GS2SPL (Apêndice A). A seguir são apresentados os resultados da aplicação das métricas aos artefatos.

5.2.1. Modularidade

Para utilizar as métricas de espalhamento de features (DoS) e entrelaçamento dos cenários (DoT), de acordo com as equações (6.2) e (6.4) temos que associar *features* a passos individuais dos cenários.

5.2.1.1. Grau de Espalhamento de Features

Quando estamos tratando dos cenários elaborados com a abordagem GS2SPL analisamos a existência da relação de dependência entre um passo *p* e uma feature *f* se, e somente se, a seleção da feature *f* dispara a configuração de *p*, assim como Almeida (2010). Ou seja, algumas das situações a seguir acontecem:

(i) o passo *p* refere-se a feature *f* como um parâmetro;

(ii) o passo *p* é diretamente relacionado a *feature f*;

(iii) o passo *p* é propriedade de um cenário relacionado a *feature f*.

O Quadro 32 apresenta a quantidade de *features* em cada cenário especificado com GS2SPL. Analisamos os cenários e somamos a quantidade de vezes que a *feature* aparece no cenário. Assim, uma *feature* que está presente em mais de um cenário é considerada dispersa. No Quadro 32, percebe-se que apenas duas *features* apresentam espalhamento, são elas: Checar Caso de Teste Sintaticamente e Checar Caso de Teste Semanticamente. As demais *features* estão completamente localizadas nos seus respectivos cenários.

Quadro 32 Atribuição de *features* a passos dos cenários do TaRGeT em GS2SPL

Cenários / Fatures	UC01	UC02	UC03	UC04	UC05
Fazer *upload* de documento	3	-	-	-	-
Editor de caso de uso	4	-	-	-	-
Checar Caso de Teste Sintaticamente	-	1	1	-	2
Checar Caso de Teste Semanticamente	-	1	1	-	2
Filtrar por propósito de teste	-	1	-	-	-
Filtrar por Requisito	-	1	-	-	-
Filtrar por caso de uso	-	1	-	-	-
Filtrar por caso de uso similar	-	1	-	-	-
Incluir Caso de Teste Permanentemente	-	1	-	-	-
Excluir Caso de Teste Permanentemente	-	1	-	-	-
Parametrizar Teste	-	1	-	-	-
Manter Consistência entre artefatos	-	1	-	-	-
Gerar caso de Teste específico	-	2	-	-	-
Gerar todos os casos de teste	-	-	2	-	-
Detectar mudança nos requisitos	-	-	-	1	-
Atualizar casos de teste	-	-	-	2	-

Incialmente calculamos a equação (6.2). Sendo *f* a feature que está espalhada e *s* o cenário. Fazemos isso para todos os cenários em que a *feature* está presente.

Para o exemplo do TaRGeT, chamaremos a *feature* Checar Caso de Teste Sintaticamente de *f1*. E cada cenário terá a sua respectiva numeração. Assim, de acordo com a

equação (6.2), temos que *f1* está relacionado com 4 passos que estão espalhados em 3 casos de uso:

Conc(*f1*, *UC02*) = $^{1}/_{4}$ = 0,25

Conc(*f1*, *UC03*) = $^{1}/_{4}$ = 0,25

Conc(*f1*, *UC05*) = $^{2}/_{4}$ = 0,5

Em seguida, calculamos o *DoS* da feature *f1* através da equação (6.1). Lembrando que o |*S*| representa a quantidade de cenários em que a *feature* está presente. Assim, de acordo com a equação (6.1):

$$DoS(f1) = 1 - \frac{3 * [(0,25 - \frac{1}{3})^2 + (0,25 - \frac{1}{3})^2 + (0,5 - \frac{1}{3})^2]}{2}$$

Assim, o *DoS* (*f1*) é igual a 0,93.

Como a *feature* Checar Caso de Teste Semanticamente também está presente nos mesmos três cenários e possui os mesmos passos nos cenários, temos que o valor de o *DoS* (*f2*) = 0,93.

Em relação aos cenários elaborados com a abordagem GAS2SPL consideramos o conhecimento de configuração para identificar os passos *p* atribuídos a *feature f*. Assim, verifica-se a existência de uma expressão de *feature* (*e*) no conhecimento de configuração que se refere a *f*, conforme Almeida (2010). A atribuição ocorre quando uma das situações a seguir acontece:

(i) *e* liga um parâmetro *par*, e o passo *p* refere-se a *par*;

(ii) *e* seleciona um cenário *s*, e o passo *p* está incluído nos passos de *s*;

(iii) *e* avalia um *advice a*, e o passo *p* está incluído nos passos de *a*;

(iv) *e* avalia um *intertype declaration itd*, e o passo *p* está incluído nos passos de *itd*;

O Quadro 33 apresenta a quantidade de *feature* em cada passo dos cenários, *intertype declarations* e *advices* elaborados com a abordagem GAS2SPL.

Quadro 33 Atribuição de features a passos dos cenários do TaRGeT em GAS2SPL

Cenários / Fatures	ITD01 UC01	ITD02 UC01	ITD03 UC02	ITD04 UC05	ITD05 UC04	CA01 UC02	CA01 UC04	CA01 UC05	CA02 UC05	CA03 UC05	CA04 UC05	CA05 UC05	CA06 UC05	ADV01
Gerar Suíte de Teste Detalhada	1	-	-	-	-	-	-	-	-	-	-	-	-	-
Gerar Suíte de Teste Diretamente	-	1	-	-	-	-	-	-	-	-	-	-	-	-
Escrever no Editor	-	-	1	-	-	-	-	-	-	-	-	-	-	-
Parametrizar Teste	-	-	-	1	-	-	-	-	-	-	-	-	-	-
Checar caso de teste semanticamente	-	-	-	-	1	-	-	-	-	-	-	-	-	-
Fazer *Upload* de documento	-	-	-	-	-	1	-	-	-	-	-	-	-	-
Checar caso de teste sintaticamente	-	-	-	-	-	-	1	-	-	-	-	-	-	-
Filtrar por propósito de teste	-	-	-	-	-	-	-	1	-	-	-	-	-	-
Incluir Caso de Teste Permanentemente	-	-	-	-	-	-	-	-	1	-	-	-	-	-
Excluir Caso de Teste Permanentemente	-	-	-	-	-	-	-	-	-	1	-	-	-	-
Filtrar por caso de uso similar	-	-	-	-	-	-	-	-	-	-	1	-	-	-
Filtrar por requisito	-	-	-	-	-	-	-	-	-	-	-	1	-	-
Filtrar por caso de uso	-	-	-	-	-	-	-	-	-	-	-	-	1	-
Manter Consistência entre artefatos	-	-	-	-	-	-	-	-	-	-	-	-	-	1

Observando o Quadro 33 percebe-se que não há espalhamento de *features*. Portanto, o *DoS*(*f*) de todas as *features* é 0.

Analisando os quadros 33 e 34 percebemos que há diferença entre a quantidade de *features* mapeadas nos cenários elaborados com GS2SPL em relação à abordagem GAS2SPL. Isto ocorre, pois as *features* Gerar Caso de Teste Específico e Gerar Todos os Casos de Teste foram mapeados, em GAS2SPL, como passos dos *intertype declarations* ITD01 e ITD02, respectivamente. As *features* Detectar Mudança nos Requisitos e Atualizar Casos de Teste foram mapeados como passos do *advice* Manter Consistência entre Artefatos.

O Quadro 34 apresenta a quantidade de *feature* em cada cenário do *MyCourses* elaborado pela abordagem GS2SPL. Analisando-o percebe-se que três *features* apresentam espalhamento, são elas: Interessados Notificados, Reservar Manualmente e Gerenciar Manualmente. As demais *features* estão completamente localizadas nos seus respectivos cenários.

Quadro 34 Atribuição de *features* a passos dos cenários do *MyCourses* em GS2SPL

Cenários / Fatures	UC01	UC02	UC03	UC04	UC05	UC06
Agendar Manualmente	2	-	-	-	-	-
Agendar Curso	2	-	-	-	-	-
Homologar Agendamento	-	3	-	-	-	-
Exportar Dados do Agendamento	-	-	4	-	-	-
Exportar para XML	-	-	2	-	-	-
Exportar para SQL	-	-	2	-	-	-
Exportar para PDF	-	-	3	-	-	-
Interessados Notificados	-	1	-	2	-	-
Reservar Manualmente	1	-	-	-	3	-
Manter Ocupado	-	-	-	-	2	-
Manter Livre	-	-	-	-	2	-
Gerenciar Manualmente	3	-	-	-	-	2
Gerenciar Conflito de Membros	-	-	-	-	-	1
Gerenciar Conflito de Recurso	-	-	-	-	-	1

Aplicando a fórmula (6.1) para as *features* que possuem espalhamento, Interessados Notificados, Reservar Manualmente e Gerenciar Manualmente. O *DoS(f)* dessas *features* é:

Interessados Notificados (*DoS(f)* = 0,89), Reservar Manualmente (*DoS(f)* = 0,75) e Gerenciar Manualmente (*DoS(f)* = 0,96).

O Quadro 35 apresenta a quantidade de *feature* em cada passa dos cenários e *intertype declaration* do *MyCourses* elaborados com a abordagem GAS2SPL.

Quadro 35 Atribuição de features a passos dos cenários do *MyCourses* em GAS2SPL

Cenários / Fatures	ITD01 UC01	ITD02 UC01	ITD03 UC05	ITD04 UC06	ITD05 UC03	ITD06 UC03	CA01 UC05	CA02 UC05	CA01 UC06	CA02 UC06	CA01 UC04	CA01 UC03	CA01 UC02
Agendar Curso Manualmente	1	-	-	-	-	-	-	-	-	-	-	-	-
Agendar Curso	-	1	-	-	-	-	-	-	-	-	-	-	-
Reservar Manualmente	-	-	1	-	-	-	-	-	-	-	-	-	-
Gerenciar Manualmente	-	-	-	1	-	-	-	-	-	-	-	-	-
Exportar XML	-	-	-	-	1	-	-	-	-	-	-	-	-
Exportar SQL	-	-	-	-	-	1	-	-	-	-	-	-	-
Manter Recurso Ocupado	-	-	-	-	-	-	1	-	-	-	-	-	-
Manter Recurso Livre	-	-	-	-	-	-	-	1	-	-	-	-	-
Gerenciar Conflito de Membros	-	-	-	-	-	-	-	-	1	-	-	-	-
Gerenciar Conflito de Recurso	-	-	-	-	-	-	-	-	-	1	-	-	-
Comunicar Alterações	-	-	-	-	-	-	-	-	-	-	1	-	-
Exportar para PDF	-	-	-	-	-	-	-	-	-	-	-	1	-
Homologar Agendamento	-	-	-	-	-	-	-	-	-	-	-	-	1

Assim como nos artefatos do TaRGeT elaborados pela abordagem GAS2SPL, para os artefatos do *MyCourses* não há espalhamento de *features*. Portanto, o *DoS(f)* de todas as *features* apresentadas no Quadro 35 é igual a 0.

5.2.1.2. Grau de Entrelaçamento de Cenários

Utilizaremos o Quadro 32 e o Quadro 34 para quantificar os passos de cada cenário relacionados às *features* obtidas pelas abordagens GS2SPL e GAS2SPL, respectivamente. Devemos identificar quais as *features* que estão presentes em cada cenário e, assim, calcular a relação entre os passos dessas *features* e a quantidade de passos do cenário.

Aos cenários do TaRGeT obtidos com o GS2SPL analisaremos as *features* que estão presentes em cada cenário. Assim, para o cenário *UC01*, apresentado no Quadro 32, de acordo com a equação (6.4), temos:

Para a *feature* Fazer *upload* de documento, o Dedi(*UC01*, *f1*) = $^{3}/_{7}$ = 0,42

Para a *feature* Editor de caso de uso, o Dedi(*UC01*, *f2*) = $^{4}/_{7}$ = 0,57

Em seguida, calculamos o *DoT* do cenário *UC01* através da equação (6.3). Lembrando que o |*T*| representa a quantidade de *features* nos cenários. Assim, de acordo com a equação (6.3):

$$DoT(UC01) = 1 - \frac{2 * [(0,42 - \frac{1}{2})^2 + (0,57 - \frac{1}{2})^2]}{2}$$

Assim, para o GS2SPL aplicado à TaRGeT, o *DoT* (*UC01*) é igual a 0,97. Em seguida, devemos calcular o grau de foco *DoF(s)* = 1-DoT, que será 0,03 para o caso de uso UC01, apresentado no Quadro 36.

Analisando o Quadro 33 e o Quadro 35 referentes a aplicação do GAS2SPL e, aplicando a equação (6.3), os valores de *DoT* do TaRGeT e *MyCourses* são iguais a 0. Como não há entrelaçamento, o grau de foco (*DoF(s)* = 1-DoT) é igual a 1. Os resultados são apresentados no Quadro 36.

Quadro 36 DoF dos cenários apresentados

	Cenários	GAS2SPL	GS2SPL
TaRGeT	UC01	1,00	0,03
	UC02	1,00	0,01
	UC03	1,00	0,07
	UC04	1,00	0,11
	UC05	1,00	0.00
MyCourses	UC01	1,00	0,05
	UC02	1,00	0,25
	UC03	1,00	0,03
	UC04	1,00	1
	UC05	1,00	0,02
	UC06	1,00	0,07

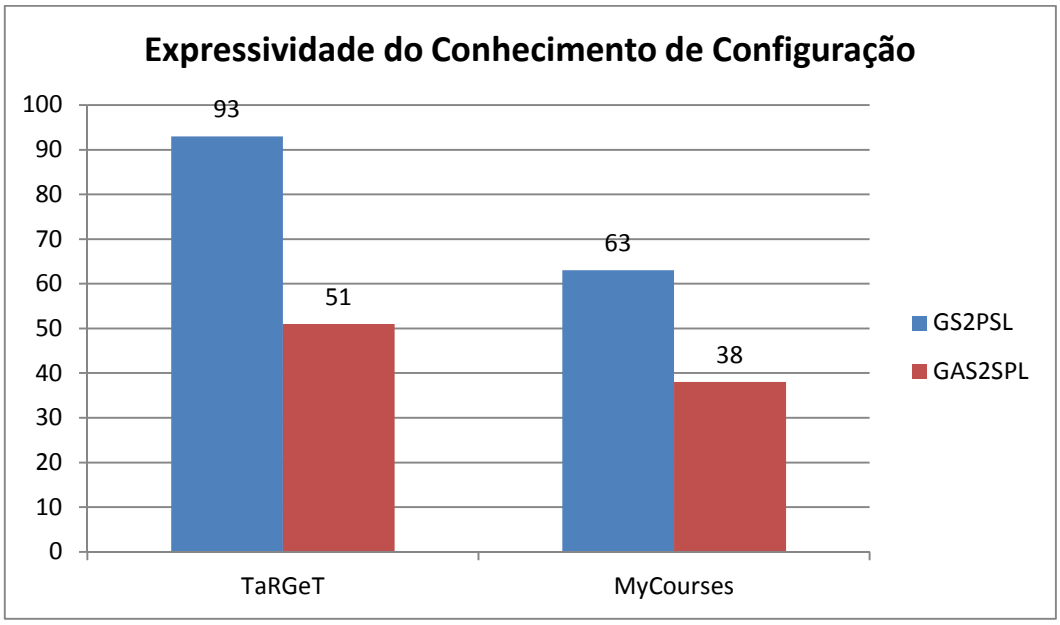
5.2.2. Expressividade

Esta seção apresenta nossa análise de expressividade considerando os estudos de caso TaRGeT e *MyCourses*. Utiliza-se a métrica que quantifica a expressividade do conhecimento de configuração.

5.2.2.1. Expressividade do conhecimento de configuração

A expressividade do conhecimento de configuração mede a quantidade de símbolos necessários para especificar o conhecimento de configuração [Alferez et al. 2013]. Para avaliar a expressividade do conhecimento de configuração na abordagem GS2SPL, analisamos as referências a *features* nos passos dos cenários. Na abordagem GAS2SPL, contamos todos os símbolos utilizados no conhecimento de configuração para especificar a composição nos *advices* ou *intertype declarations* nos cenários base. A Figura 160 ilustra o resultado da aplicação desta métrica nas abordagens para o TaRGeT e o *MyCourses*.

Figura 160 Expressividade do Conhecimento de Configuração



Analisando a Figura 160 percebemos que o conhecimento de configuração elaborado pela abordagem GS2SPL é mais detalhado do que o conhecimento de configuração elaborado pela nossa abordagem. Isto ocorre, pois GS2SPL possui referências a features nos passos dos cenários, associando-as com comportamentos variantes e, GAS2SPL possui um artefato dedicado para descrever todas as composições. À medida que o conhecimento de configuração cresce, esse detalhamento obtido com GS2SPL prejudica o entendimento dos artefatos. Assim, a nossa abordagem tem um desempenho melhor do que a abordagem GS2SPL no aspecto de expressividade do conhecimento de configuração.

5.4. Considerações Finais

Neste capítulo nós apresentamos uma avaliação através da aplicação de métricas de modularidade e expressividade aos artefatos do TaRGeT e *MyCourses* elaborados pelas abordagens GS2SPL e pela nossa abordagem (GAS2SPL). Os resultados foram comparados para avaliar qual abordagem promovia melhor modularidade e expressividade para LPS. A abordagem GAS2SPL apresentou bons resultados em relação à modularidade, eliminando o espalhamento e o entrelaçamento de *features* nos cenários e, uma maior expressividade do conhecimento de configuração, pois utilizam menos símbolos para elaborar este artefato. Assim, conseguimos responder as duas questões de pesquisa, que os artefatos produzidos pela nossa abordagem são mais modularizados e possuem uma maior expressividade do que os artefatos elaborados pela abordagem GS2SPL.

Capítulo 6 Conclusão

Este capítulo apresenta as contribuições e limitações encontradas para definir a abordagem GAS2SPL proposta nesta dissertação, além de sugestões de trabalhos futuros.

6.1. Contribuições do trabalho

Esta dissertação definiu um processo da Engenharia de Requisitos para Linhas de Produto de Software chamado GAS2SPL (*Goals and Aspectual Scenarios to Software Product Lines*) que consiste de oito atividades, sendo as sete primeiras, relacionadas a Engenharia de Domínio e, a última, um subprocesso relacionado a Engenharia de Aplicação de LPS. GAS2SPL permite obter um modelo de *feature*, cenários aspectuais e o conhecimento de configuração a partir do modelo de objetivos de uma LPS. O artefato para representar o conhecimento de configuração (do inglês, *configuration knowledge*), captura o mapeamento entre o modelo de *feature* e os artefatos da LPS. No subprocesso de configuração de produtos, as possíveis variantes da linha são geradas e analisadas com base na prioridade atribuída pelos interessados aos requisitos não funcionais. O objetivo dessa atividade é sugerir a configuração de produto que melhor atenda aos requisitos não funcionais priorizados pelas partes interessadas.

O processo GAS2SPL inclui a definição de heurísticas para permitir a integração das técnicas GS2SPL e MSVCM, de forma que elas façam parte de um único processo para apoiar a engenharia de requisitos de LPS. A abordagem GS2SPL leva em consideração os objetivos dos interessados para identificar as *features* e elaborar os cenários, mas não utiliza recursos da orientação a aspectos nesses artefatos. Assim, a variabilidade e as partes comuns da LPS estão presentes no mesmo cenário de caso de uso, dificultando o entendimento e a evolução da LPS, assim como a configuração dos produtos. Se o engenheiro usar somente o GS2SPL, se beneficiará com a identificação sistemática das *features* e cenários a partir dos modelos de objetivos, e a configuração do produto será dirigida pela satisfação de requisitos não funcionais, mas não terá a separação das *features* nos cenários que é promovida pelo MSVCM. Se o engenheiro usar somente o MSVCM para especificar a LPS, não terá os benefícios providos pelo GS2SPL. Fazendo a integração das duas técnicas é possível somar os benefícios de ambas e poder derivar os artefatos de requisitos da LPS a partir do modelo de objetivos, que é um modelo mais próximo da realidade dos interessados por capturar os seus objetivos. Fez-se necessário, também, estender o MSVCM, pois esta técnica não contemplava as mudanças estruturais nos casos de uso que, apenas, poderiam ser implementadas através da inclusão dos

intertype declarations (ITD). O *ITD* é um recurso da orientação a aspecto, não considerado no MSVCM, mas essencial para promover a modularidade de casos de uso para linhas de produto.

Realizou-se uma avaliação quantitativa através da utilização de métricas para modularidade e expressividades que foram aplicadas aos artefatos dos exemplos TaRGeT e *MyCourses* elaborados pelas abordagens GS2SPL e GAS2SPL. As métricas de modularidade (espalhamento de *features* e entrelaçamento de cenários) visam garantir que as *features* estejam contidas em módulos separados. A métrica utilizada para garantir a facilidade de entendimento de uma especificação foi a expressividade do conhecimento de configuração. Analisando os resultados da avaliação percebe-se que a abordagem GAS2SPL conseguiu reduzir o espalhamento das features nos dois estudos de caso e, o entrelaçamento dos cenários no TaRGeT. Portanto, conseguiu-se validar as duas questões de pesquisas definidas na seção de avaliação. Os cenários elaborados com a abordagem GAS2SPL são mais modularizados e as especificações tem maior expressividade do que as elaboradas com a abordagem GS2SPL.

Assim, percebemos através dos resultados da avaliação realizada que a abordagem GAS2SPL preenche duas lacunas deixadas pela abordagem GS2PSL: a utilização de cenários aspectuais, que favorece a separação de interesses transversais, e a representação do conhecimento de configuração, que relaciona o mapeamento entre o modelo de *features* e os artefatos da LPS. Assim, a obtenção sistemática dos cenários de caso de uso com separação de interesses transversais, facilitará a manutenção e o reuso dos artefatos de requisitos da LPS.

6.2. Limitações da abordagem

Este trabalho apresenta algumas limitações, principalmente em relação à complexidade inicial de utilização da abordagem. De fato, a qualidade e a consistência dos artefatos produzidos possuem uma forte dependência do nível de detalhamento associado às informações descritas nos modelos de requisitos em *i**. Além disso, devido ao processo ainda não ter ferramenta de suporte, o sucesso da aplicação do processo depende da experiência do engenheiro de software em relação aos conceitos relacionados ao *framework i**, orientação a aspectos, abordagem GS2SPL e MSVCM. Portanto, o engenheiro de software deve dedicar um tempo considerável para familiarizar-se com as atividades e seus passos, visto que a realização de um mapeamento incorreto pode resultar é uma LPS inconsistente com a realidade.

Algumas diretrizes não puderam ser aplicadas aos estudos de caso. Assim, a abordagem não foi testada completamente, pois os estudos de caso utilizados não apresentavam elementos que se encaixassem nas restrições das diretrizes.

O processo ainda não foi usado e avaliado com usuários reais, por isso não pudemos

enumerar problemas identificados por outros usuários. É necessária a realização de um experimento como forma de avaliar a usabilidade e a eficácia do processo da abordagem GAS2SPL, aplicando-se a abordagem a um estudo de caso que será desenvolvido por usuários reais. Apesar disso, as abordagens GS2SPL e GAS2SPL foram aplicadas a dois exemplos de LPS - o TaRGeT e o *MyCourses.* Os artefatos gerados foram avaliados usando métricas de modularidade e expressividade e os resultados foram comparados para avaliar qual abordagem promovia melhor modularidade e expressividade para LPS. Os resultados obtidos indicam que a abordagem GAS2SPL produz artefatos mais modulares do que a abordagem GS2SPL.

6.3. Trabalhos futuros

Como trabalhos futuros pretendem-se:

- Realizar um estudo empírico para avaliar o processo da abordagem GAS2SPL do ponto de vista dos interessados e especialistas em Engenharia de Requisitos para identificar se o processo da abordagem e os artefatos gerados estão coerentes com o que se propõe a fazer e podem ser validados.
- Realizar um experimento, em que o processo será utilizado e avaliado qualitativamente por estudantes de graduação.
- Estender a abordagem GAS2SPL para as demais fases do *framework* de Engenharia de Linha de Produtos de Software definido por Pohl, Böckle e Linden (2005), que não seja apenas a fase de requisitos da Engenharia de Domínio e Aplicação.
- Realizar mais estudos de caso utilizando a abordagem GAS2SPL, com o objetivo de identificar o desempenho da abordagem na elaboração dos artefatos para outros cenários.
- Para reduzir o esforço de utilização do processo e evitar que haja inconsistências nos artefatos obtidos, é necessário uma ferramenta capaz de tornar o processo semi-automatizado para gerar o modelo de *feature* e os cenários aspectuais a partir do modelo de objetivos. Esta ferramenta também deve suportar a seleção de *features* (configuração do produto) utilizando objetivos-*soft* e a configuração de cenários utilizando o conhecimento de configuração.

6.4. Considerações finais

Desenvolver este trabalho com temas relevantes que vêm ganhando a atenção da academia e da indústria como Linhas de Produto de Software e Orientação a Aspectos e que está em pleno crescimento foi uma oportunidade de aprendizado e amadurecimento, principalmente por saber que este trabalho poderá contribuir com a comunidade para o processo de obtenção de cenários de caso de uso com separação de interesses transversais a partir dos objetivos dos interessados. Espera-se que esta dissertação possa gerar uma publicação em um simpósio de alcance nacional da área, para que não fique restrita a comunidade acadêmica regional.

O maior desafio enfrentado no decorrer deste trabalho foi o fato de realizar o trabalho individualmente, sem participar de um grupo de pesquisa. Isto iria favorecer debates a respeito da elaboração do processo da abordagem, bem como a troca de experiências entre os membros do grupo de pesquisa. Mas, a possibilidade de contatar a Orientadora sempre que havia algum questionamento e obter *feedback* rapidamente supriu em partes essa dificuldade. Desenvolver a abordagem exigiu conhecimento de diversas áreas que não tinha vivência, das quais se pode citar, Engenharia de Requisitos Orientada a Objetivos, especificação de cenários de caso, *framework i**, além de compreender o processo da abordagem GS2SPL e da técnica MSVCM para apontar melhorias.

Após o desenvolvimento da abordagem e, através da avaliação realizada, podemos afirmar que a abordagem conseguiu atingir o objetivo de modularizar as especificações dos cenários de caso e facilitar a expressividade do conhecimento de configuração.

Referências Bibliográficas

ALFEREZ, M. et al. Evaluating scenario-based spl requirements approaches - The case for modularity, stability and expressiveness. *Requirements Engineering*, p. 1–22, 10 2013.

ALMEIDA, R. B. D. *Modeling software product line variability in use case scenarios: an approach based on crosscutting mechanisms*. Tese (Doutorado) — Universidade Federal de Pernambuco, 2010. Centro de Informática - Universidade Federal de Pernambuco.

ARAUJO, J.; WHITTLE, J.; KIM, D.-K. Modeling and com- posing scenario-based requirements with aspects. p. 58–67, Sept 2004. ISSN 1090-705X.

ASADI, M. e. a. Goal-driven software product line engineering. In: *Proceedings of SAC 2011*. TaiChung, China,: [s.n.], 2011. p. 21–25.

BACHMANN, F.; BASS, L. Managing variability in software architectures. Softw. Eng. Notes 26(3), p. 126–132, 2001.

BACHMANN, F.; BASS, L. J. Managing variability in software architectures. In: *SIGSOFT, Softw. Eng. Notes 26(3)*. [s.n.], 2001. p. 126–132. Disponível em: <http://www.sei.cmu.edu/library/assets/variability.pdf>.

BODKIN, R.; LADDAD, R. Zen and the art of aspect-oriented programming. In: . [S.l.: s.n.], 2004. Linux Magazine, April.

BONIFÁCIO, R.; BORBA, P. Modeling scenario variability as crosscutting mechanisms. Proceedings of the 8th ACM International Conference on Aspect-oriented Software Development. ACM, New York, NY, USA, p. 125–136, 2009. Disponível em: <http://doi.acm.org/10.1145/1509239.1509258>.

BONIFÁCIO, R.; TEIXEIRA, L.; BORBA, P. Hephaestus: a tool for managing spl variabilities. In Simpósio Brasileiro de Componentes, Arquitetura e Reuso. Sessão de Ferramentas. Natal, Brasil., 2009.

BORBA, C. C. Uma abordagem orientada a objetivos para as fases de requisitos de linhas de produto de software. In: Recife, PE, Brasil: Universidade Federal de Pernambuco, 2009. Dissertação de Mestrado.

CLEMENTS, P.; NORTHROP, L. *Software Product Lines: Practices and Patterns*. [S.l.]: Addison Wesley, 2002. (SEI Series in Software Engineering). ISBN 0201703327.

CHITCHYAN, R.; GREENWOOD, P.; SAMPAIO, A.; RASHID, A.; GARCIA, A.; SILVA, L. Semantic vs. syntactic compositions in aspect-oriented requirements engineering: an empirical study. In: K. J. Sullivan (ed.) AOSD 2009, p. 149-160. ACM, Charlottesville, Virginia, USA (2009).

CHUNG L, NIXON BA, YU E, MYLOPOULOS J. Non-Functional Requirements in Software Engineering. Kluwer Academic: Boston, 2000.

COCKBURN, A. *Writing Effective Use Cases*. 1st. ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2000. ISBN 0201702258.

CZARNECKI, K.; EISENECKER, U. *Generative Programming: Methods, Tools, and Applications*. Boston, MA: Addison-Wesley, 2000. ISBN 978- 0-201-30977-5.

CZARNECKI K, HELSEN S, EISENECKER U. Staged Configuration Using Feature Models, *Software Product Lines: Third International Conference SPLC 2004*: Boston MA USA, 2004;266-283

DIJKSTRA, E. W. A Discipline of Programming. Prentice Hall, Englewood Cliffs, NJ, 1976.

EADDY, M; AHO, A; MURPHY, G. Identifying, assigning, and quantifying crosscutting concerns. In First Workshop on Assessment of Contemporary Modularization Techniques (ACOM), Minneapolis, USA, Maio 2007.

ERIKSSON, M.; BÖRSTLER, J.; BORG, K. The pluss approach - domain modeling with *features*, use cases and use case realizations. In: OBBINK, H.; POHL, K. (Ed.). *Software Product Lines*. Springer Berlin Heidelberg, 2005, (Lecture Notes in Computer Science, v. 3714). p. 33–44. ISBN 978-3-540-28936-4. Disponível em: < http://www8.cs.umu.se/magnuse/papers/ErikssonSPLC05.pdf>.

FIGUEIREDO, E. M. L. Uma abordagem quantitativa para desenvolvimento de software orientado a aspectos. Dissertação (mestrado) – Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática. 140 f. Rio de Janeiro, 2006.

GOETTEN, V. J.; WINCK, D. Aspectj programação orientada a aspectos com java. In: . São Paulo: Novatec Editora, 2006.

GRAU, G. et al. Fostering investigation, collaboration, and evaluation: The i* wiki experience. In: *Proc. of the 3rd International i* Workshop*. [S.l.: s.n.], 2008.

GUEDES, G. et al. Gs2spl: Goals and scenarios to software product lines. In: *SEKE - Knowledge Systems Institute Graduate School*. [S.l.: s.n.], 2012. p. 651– 656.

KANG, K. C. et al. *Feature-Oriented Domain Analysis (FODA) Feasibility Study*. [S.l.], 1990.

KICZALES, G.; LAMPING, J.; MENHDHEKAR, A.; MAEDA, C.; LOPES, C.; LOINGTIER, J. M.; IRWIN, J. Aspect-Oriented Programming, Springer-Verlag, 1997, p. 220-242.

KIENZLE, J.; GUELFI, N.; MUSTAFIZ, S. Transactions on aspect-oriented software development vii. In: KATZ, S.; MEZINI, M. (Ed.). Berlin, Heidelberg: Springer-Verlag, 2010. cap. Crisis Management Systems: A Case Study for Aspect-oriented Modeling, p. 1–22. ISBN 3-642-16085-9, 978-3-642-16085-1.

KOTONYA, G.; SOMMERVILLE, I. *Requirements Engineering - Processes and Techniques*. [S.l.]: John Wiley & Sons, 1998.

LAMSWEERDE, A. v. Requirements engineering in the year 00: A re- search perspective. In: *Proceedings of the 22Nd International Conference on Software Engineering*. New York, NY, USA: ACM, 2000. (ICSE '00), p. 5–19. ISBN 1-58113-206-9. Disponível em: <http://doi.acm.org/10.1145/337180.337184>.

LAMSWEERDE, A. v. Goal-oriented requirements engineering: A guided tour. Toronto, Canada, p. 249–262, 2001.

LAPOUCHNIAN, A. *Goal-Oriented Requirements Engineering: An Overview of the Current Research*. Toronto, Canada, 2005.

LIMA, C. D. Q. E-spl: uma abordagem para a fase de requisitos na engenharia de domínio e na engenharia de aplicação com modelos de objetivos. In: Recife, PE, Brasil: Universidade Federal de Pernambuco, 2011. p. 199. Dissertação de Mestrado.

MAIDEN, N.; ALEXANDER, I. *Scenarios, stories, use cases: through the systems development life-cycle*. [S.l.]: J. Wiley and sons 1st Edition. ed. [S.l.], 2004.

MobileMedia. MobileMedia, 2011. Disponível em: <http://twiki.cin.ufpe.br/twiki/bin/view/ProjetoProcad/MobileMedia>. Acesso em Junho de 2013.

OLIVEIRA, A. P. A.; CYSNEIROS, L. M.; LEITE, J. C.; DO PRADO, J. C.; FIGUEIREDO, E. M.; LUCENA, C. J. Integrating scenarios, i*, and aspect in the context of multi-agent systems. *Proceedings of the 2006 conference of the Center for Advanced Studies on Collaborative research*, ACM, Toronto, Ontario, Canada, 2006.

OLIVEIRA, C.; ARAÚJO, J.; SILVA, C. T. L. L. Integração de kaos com cenários aspectuais. In: *JISBD - XV Jornadas de Ingenieria de Software y Bases de Datos*. [S.l.: s.n.], 2010. p. 49–60.

POHL, K.; BÖCKLE, G.; LINDEN, F. J. v. d. *Software Product Line Engineering: Foundations, Principles and Techniques*. Secaucus, NJ, USA: Springer- Verlag New York, Inc., 2005. ISBN 3540243720.

POHL, K.; METZGER, A. The e-shop product line online. In: [s.n.], 2006. Disponível em: <http:www.sei.cmu.edu/splc2006/eshop.pdf>.

POHL, K. METZGER, A. Business process modeling notation. In: OMG Available Specification, 2008. (V1.1). Disponível em: <http://www.omg.org/spec/BPMN/1.1/PDF/>.

RASHID, A. et al. Early aspects: A model for aspect-oriented requirements engineering. In: *In Proceedings of the 10th Anniversary IEEE Joint international Conference on Requirements Engineering*. Washington, DC: IEEE Computer Society, 2002. p. 199–202.

ROLLAND, C.; C., S.; ACHOUR, C. B. Software engineering, IEEE transactions. *24(12)*, p. 1055–1071, 1998.

SANTOS, L.; SILVA, L.; BATISTA, T. On the integration of the feature model and PL-AOVGraph. Proceedings of the 2011 International Workshop on Early Aspects, 31-36.

SANTOS, E. B. dos. Uma Proposta de Métricas para Avaliar Modelos i*. 2008. 132 f. Dissertação (Mestrado em Ciência da Computação) – Centro de Informática, Universidade Federal de Pernambuco, Recife, 2008.

SANTANDER, V. F. A. *Integrando Modelagem Organizacional com Modelagem Funcional*. Tese (Doutorado) — Universidade Federal de Pernambuco, Recife, PE, Brasil, 2002. (Tese de Doutorado) Universidade Federal de Pernambuco.

SCORE. Score, 2011. Disponível em: < http://score-contest.org/2011/>. Acesso em Outubro de 2014.

SILVA, C. T. L. L.; BORBA, C.; CASTRO, J. G2spl: Um processo de engenharia de requisitos orientada a objetivos para linhas de produtos de software. In: *Proceedings of Workshop on Requirements Engineering (WER'10 at CibSE'10)*. Cuenca, Equador: [s.n.], 2010. p. 5–16.

SILVA, C. et al. Tailoring an Aspectual Goal Oriented Approach to Model *Features*. In: 20th International Conference on Software Engineering and Knowledge Engineering (SEKE'08). San Francisco Bay, USA: [s.n]. 2008.

SILVA, C. T. L. L.; BORBA, C. C.; CASTRO, J. A goal oriented approach to identify and configure feature models for software product lines. In: *Proceedings of 14th Workshop on Requirements Engineering(WER 2011)*. Rio de Janeiro, RJ, Brasil: [s.n.], 2011. p. 395–406.

SOUZA, G. G. d. *Objetivos e Cenários na Engenharia de Requisitos para Linhas de Produto de Software*. (Dissertação de Mestrado) Centro de Informática, Universidade Federal de Pernambuco.

TARGET. Target product line. In:[s.n.], 2011. Disponível em: <http://twiki.cin.ufpe.br/twiki/bin/view/TestProductLines/TaRGeTProductLine>.

TRAVASSOS, G. H. Introdução à engenharia de software experimental. In: COPPE/UFRJ: [s.n.], 2002. Relatório Técnico RT-ES 590/02.

WHITTLE, J.; ARAÚJO, J. Scenario modeling with aspects. *IEE Proceedings - Software*, v. 4, p. 157–171, 2004.

WHITTLE, J.; JAYARAMAN, P. Mata: A tool for aspect-oriented modeling based on graph transformation. *Workshop on Aspect Oriented Modeling at the International Models Conference*, Nashville, 2007.

WOHLIN, C. et al. Experimentation in software engineering: an introduction. *Kluwer Academic Publishers*, USA, 2000.

YU, E. S.-K. *Modelling Strategic Relationships for Process Reengineering*. Tese (Doutorado) — University of Toronto, Toronto, Ont., Canada, Canada, 1995.

YU, Y.; DO PRADO LEITE, J. C. S.; LAPOUCHNIAN, A.; MYLOPOULOS, J. Configuring *features* with stakeholder goals. Proceedings of the 2008 ACM symposium on Applied computing, 645-649.

YU, E.; MYLOPOULOS, J.; MAIDEN, N.; GIORGINI, P. *Social Modeling for Requirements Engineering*, Massachusetts: MIT Press, 2011. 760 p.

Apêndice A – Artefatos do *MyCourses* elaborados pela abordagem GS2SPL

As figuras a seguir (161-166), apresentam os casos de uso do *MyCourses* elaborados através da abordagem GS2SPL. Estes artefatos foram utilizados para realização da avaliação da abordagem.

Figura 161 Caso de uso Agendamento de Curso, do MyCourses, elaborado com a abordagem GS2SPL

<table>
<tr><td colspan="3">Caso de Uso 1: Agendamento de curso</td></tr>
<tr><td colspan="3">INFORMAÇÃO CARACTERÍSTICA

Ator Primário: Gerente de Programa, Professor
Feature: -
Escopo: TaRGeT
Pré-condições: -
Condição de Sucesso (pós-condições): Curso agendado</td></tr>
<tr><td colspan="3">CENÁRIO PRINCIPAL</td></tr>
<tr><td>Passo</td><td>Ação do Usuário</td><td>Resposta do Sistema</td></tr>
<tr><td>1</td><td>Inserir dados sobre membros, cursos e recursos</td><td>-</td></tr>
<tr><td>2a</td><td>Agendar curso automaticamente [Agendar Curso]</td><td>Curso cadastrado</td></tr>
<tr><td>2b</td><td>Definir alocação manual de membros, recurso e curso [Agendar Manualmente] [Reservar Manualmente] [Gerenciar Manualmente]</td><td>Aloca membro e recurso de acordo com os interesses dos usuários [Gerenciar Manualmente]</td></tr>
<tr><td>3b</td><td>-</td><td>Curso cadastrado de acordo com os interesses do usuário [Agendar Manualmente]</td></tr>
<tr><td colspan="3">CENÁRIOS SECUNDÁRIOS</td></tr>
<tr><td>Passo</td><td>Ação do Usuário</td><td>Resposta do Sistema</td></tr>
<tr><td>2a</td><td>Falha na geração do agendamento de curso [Agendar Curso]</td><td>Informa exceção ocorrida e o curso não é gerado</td></tr>
<tr><td>2b</td><td>Falha no gerenciamento de conflito [Gerenciar Manualmente]</td><td>Informa exceção ocorrida e o curso não é gerado</td></tr>
<tr><td colspan="3">INFORMAÇÃO RELACIONADA (opcional)</td></tr>
<tr><td colspan="3">Caso de uso Pai: -
Casos de Uso Subordinados: Reserva de Recurso, Gerenciamento de Conflito
Requisitos Não funcionais:</td></tr>
</table>

Figura 162 Caso de uso Homologar Agendamento, do MyCourses, elaborado com a abordagem GS2SPL

<table>
<tr><td colspan="3">Caso de Uso 2: Homologar Agendamento</td></tr>
<tr><td colspan="3">INFORMAÇÃO CARACTERÍSTICA

Ator Primário: Gerente de Programa
Feature: -
Escopo: TaRGeT
Pré-condições: -
Condição de Sucesso (pós-condições): Agendamento de curso homologado</td></tr>
<tr><td colspan="3">CENÁRIO PRINCIPAL</td></tr>
<tr><td>Passo</td><td>Ação do Usuário</td><td>Resposta do Sistema</td></tr>
<tr><td>1</td><td>Verifica alocação de membros, recursos</td><td>-</td></tr>
<tr><td>2</td><td>Homologar Agendamento [Homologar Agendamento]</td><td>Agendamento Homologado [Homologar Agendamento] [Notificar Interessados]</td></tr>
<tr><td colspan="3">CENÁRIOS SECUNDÁRIOS</td></tr>
<tr><td>Passo</td><td>Ação do Usuário</td><td>Resposta do Sistema</td></tr>
<tr><td>2</td><td>Agendamento contém inconsistências [Homologar Agendamento]</td><td>Informa exceção ocorrida e o agendamento não é realizado</td></tr>
<tr><td colspan="3">INFORMAÇÃO RELACIONADA (opcional)</td></tr>
<tr><td colspan="3">Caso de uso Pai: -
Casos de Uso Subordinados: Notificar Interessados
Requisitos Não funcionais:</td></tr>
</table>

Figura 163 Caso de uso Exportação de Agendamento, do MyCourses, elaborado com a abordagem GS2SPL

<table>
<tr><td colspan="3">Caso de Uso 3: Exportação de Agendamento</td></tr>
<tr><td colspan="3">INFORMAÇÃO CARACTERÍSTICA

Ator Primário: Gerente de Programa, Professor, Administrador de Sistema, Aluno
Feature: -
Escopo: TaRGeT
Pré-condições: -
Condição de Sucesso (pós-condições): Agendamento exportado</td></tr>
<tr><td colspan="3">CENÁRIO PRINCIPAL</td></tr>
<tr><td>Passo</td><td>Ação do Usuário</td><td>Resposta do Sistema</td></tr>
<tr><td>1</td><td>Escolhe a forma de exportação dos dados do agendamento</td><td>-</td></tr>
<tr><td>2a</td><td>Exportar dados para XML [Exportar dados do agendamento]</td><td>Dados exportados para XML [Exportar para XML]</td></tr>
<tr><td>2b</td><td>Exportar dados para SQL [Exportar dados do agendamento]</td><td>Dados exportados para SQL [Exportar para SQL]</td></tr>
<tr><td>2c</td><td>Exportar dados para PDF [Exportar dados do agendamento]</td><td>-</td></tr>
<tr><td>3c</td><td>Escolher a versão do PDF [Exportar para PDF]</td><td>Dados exportados para PDF [Exportar para PDF]</td></tr>
<tr><td>4</td><td>-</td><td>Dados exportados com sucesso [Exportar dados do agendamento]</td></tr>
<tr><td colspan="3">CENÁRIOS SECUNDÁRIOS</td></tr>
<tr><td>Passo</td><td>Ação do Usuário</td><td>Resposta do Sistema</td></tr>
<tr><td>2a</td><td>Falha na geração da exportação [Exportar para XML]</td><td>Informa exceção ocorrida e a exportação não é realizada</td></tr>
<tr><td>2b</td><td>Falha na geração da exportação [Exportar para SQL]</td><td>Informa exceção ocorrida e a exportação não é realizada</td></tr>
<tr><td>2c</td><td>Falha na geração da exportação [Exportar para PDF]</td><td>Informa exceção ocorrida e a exportação não é realizada</td></tr>
<tr><td>4</td><td>Falha na geração da exportação</td><td>Informa exceção ocorrida e a exportação não é realizada</td></tr>
<tr><td colspan="3">INFORMAÇÃO RELACIONADA (opcional)</td></tr>
<tr><td colspan="3">Caso de uso Pai: -
Casos de Uso Subordinados:
Requisitos Não funcionais:</td></tr>
</table>

Figura 164 Caso de uso Interessados Notificados, do MyCourses, elaborado com a abordagem GS2SPL

<table>
<tr><td colspan="3">Caso de Uso 4: Interessados notificados</td></tr>
<tr><td colspan="3">INFORMAÇÃO CARACTERÍSTICA

Ator Primário: Professor, Aluno
Feature: -
Escopo: TaRGeT
Pré-condições: -
Condição de Sucesso (pós-condições): Interessados notificados</td></tr>
<tr><td colspan="3">CENÁRIO PRINCIPAL</td></tr>
<tr><td>Passo</td><td>Ação do Usuário</td><td>Resposta do Sistema</td></tr>
<tr><td>1</td><td>-</td><td>Verifica se houve homologação ou alteração no curso que o usuário está inscrito [Notificar Interessado]</td></tr>
<tr><td>2</td><td>-</td><td>Usuário notificado com sucesso [Notificar Interessado]</td></tr>
<tr><td colspan="3">CENÁRIOS SECUNDÁRIOS</td></tr>
<tr><td>Passo</td><td>Ação do Usuário</td><td>Resposta do Sistema</td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td colspan="3">INFORMAÇÃO RELACIONADA (opcional)</td></tr>
<tr><td colspan="3">Caso de uso Pai: -
Casos de Uso Subordinados:
Requisitos Não funcionais:</td></tr>
</table>

Figura 165 Caso de uso Reserva de Recurso, do MyCourses, elaborado com a abordagem GS2SPL

<table>
<tr><td colspan="3">Caso de Uso 5: Reserva de recurso</td></tr>
<tr><td colspan="3">INFORMAÇÃO CARACTERÍSTICA

Ator Primário: Gerente de Programa, Professor
Feature: -
Escopo: TaRGeT
Pré-condições: -
Condição de Sucesso (pós-condições): Recurso reservado</td></tr>
<tr><td colspan="3">CENÁRIO PRINCIPAL</td></tr>
<tr><td>Passo</td><td>Ação do Usuário</td><td>Resposta do Sistema</td></tr>
<tr><td>1</td><td>Selecionar recurso</td><td>-</td></tr>
<tr><td>2a</td><td>Inserir informação sobre horário de utilização do recurso [Reservar Manualmente]</td><td>Verifica disponibilidade do recurso para os parâmetros informados [Reservar Manualmente]</td></tr>
<tr><td>3a</td><td>Manter recurso ocupado [Manter Ocupado]</td><td>Manter os recursos a maior parte do tempo em utilização [Manter Ocupado]</td></tr>
<tr><td>3b</td><td>Manter recurso livre [Manter Livre]</td><td>Manter os recursos a maior parte do tempo disponível [Manter Livre]</td></tr>
<tr><td>4</td><td>-</td><td>Reserva de recurso realizado [Reservar Manualmente]</td></tr>
<tr><td colspan="3">CENÁRIOS SECUNDÁRIOS</td></tr>
<tr><td>Passo</td><td colspan="2">Ação do Usuário</td><td>Resposta do Sistema</td></tr>
<tr><td></td><td colspan="2"></td><td></td></tr>
<tr><td colspan="3">INFORMAÇÃO RELACIONADA (opcional)</td></tr>
<tr><td colspan="3">Caso de uso Pai: -
Casos de Uso Subordinados:
Requisitos Não funcionais:</td></tr>
</table>

Figura 166 Caso de uso Gerenciamento de Conflito, do MyCourses, elaborado com a abordagem GS2SPL

<table>
<tr><td colspan="3">Caso de Uso 6: Gerenciamento de Conflito</td></tr>
<tr><td colspan="3">INFORMAÇÃO CARACTERÍSTICA

Ator Primário: Gerente de Programa, Professor
Feature: -
Escopo: TaRGeT
Pré-condições: -
Condição de Sucesso (pós-condições): Conflito gerenciado</td></tr>
<tr><td colspan="3">CENÁRIO PRINCIPAL</td></tr>
<tr><td>Passo</td><td>Ação do Usuário</td><td>Resposta do Sistema</td></tr>
<tr><td>1</td><td>-</td><td>Aloca membros e recursos de acordo com os interesses dos usuários
[Gerenciar Manualmente]</td></tr>
<tr><td>2a</td><td>Gerenciar conflito de membros</td><td>Verifica disponibilidade do membro
[Gerenciar Conflito de Membro]</td></tr>
<tr><td>2b</td><td>Gerenciar conflito de recurso</td><td>Verifica disponibilidade do recurso
[Gerenciar Conflito de Recurso]</td></tr>
<tr><td>3</td><td>-</td><td>Alocação realizada sem conflito
[Gerenciar Manualmente]</td></tr>
<tr><td colspan="3">CENÁRIOS SECUNDÁRIOS</td></tr>
<tr><td>Passo</td><td colspan="2">Ação do Usuário</td><td>Resposta do Sistema</td></tr>
<tr><td></td><td colspan="2"></td><td></td></tr>
<tr><td colspan="3">INFORMAÇÃO RELACIONADA (opcional)</td></tr>
<tr><td colspan="3">Caso de uso Pai: -
Casos de Uso Subordinados:
Requisitos Não funcionais:</td></tr>
</table>