Soraia Guardado Freitas

Licenciada em Engenharia Informática

Suporte à Modelação de Valor de Negócio

Dissertação para obtenção do Grau de
Mestre em Engenharia Informática

Orientadora: Ana Moreira,
Professora Associada com Agregação,
Universidade Nova de Lisboa

Co-orientador: Eric Souza, Doutorando,
Universidade Nova de Lisboa

Júri:

Presidente:	João Miguel da Costa Magalhães
Arguente:	José Alberto Rodrigues Pereira Sardinha
Vogal:	Ana Maria Diniz Moreira

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Dezembro, 2018

Suporte à Modelação de Valor de Negócio

A todos os que me apoiaram

Agradecimentos

Para começar quero agradecer à minha orientadora, Prof. Ana Moreira, por me ter dado a oportunidade de trabalhar neste projeto, por todo o suporte, conselhos e compreensão. Do mesmo modo, um sincero obrigada ao meu co-orientador, Eric Souza, por me ter recebido no seu projeto, por todo o acompanhamento e esclarecimentos.

Tenho a agradecer à Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, em particular ao Departamento de Informática, por todos os conhecimentos transmitidos e oportunidades proporcionadas.

Agradeço também ao Cristiano de Faveri por todo o apoio prestado. Ao Tiago Silva um muito obrigada pela motivação, e persistência.

A todas as pessoas que participaram na avaliação da ferramenta obrigada pela ajuda, e tempo disponibilizado.

Não menos importante, agradeço à minha família que sem eles não estaria onde estou hoje. Obrigada por toda a paciência, motivação e suporte ao longo destes anos.

Obrigada a todos que de algum modo contribuíram para eu me ter tornado na pessoa que sou hoje.

RESUMO

Num mercado global, é cada vez mais importante para as empresas conseguirem modelar o seu negócio e perceber as trocas de valor com a rede de parceiros com quem negoceiam. Assim, torna-se essencial modelar essas trocas de valor económico, através de um modelo de negócios que seja fácil de compreender e utilizar quer pelo utilizador, quer pela equipa de desenvolvimento de software. Contudo, existem poucos modelos (e métodos) que permitem modelar trocas de valor. Adicionalmente, não existem ferramentas de desenho que façam o alinhamento entre a perspetiva de negócio e a perspetiva de software.

Assim, este trabalho de mestrado tem por objetivo (i) estudar os métodos existentes para representar valor de negócio, identificando as suas limitações, (ii) encontrar uma solução que facilite o desenvolvimento de software baseado em valor, e (iii) avaliar a solução proposta usando as boas práticas da engenharia de software baseada em evidências. Para atingir estes objetivos começámos por executar um estudo sistemático para garantir uma visão abrangente de todos os métodos que existem. Em seguida, selecionámos um ambiente tecnológico onde desenvolvemos uma ferramenta de suporte ao método DVD (*Dynamic Value Description*). Este método permite modelar os conceitos principais associados a uma troca de valor e fazer o alinhamento entre a perspetiva de negócio e a perspetiva de software. A ferramenta desenvolvida, DVDTool, permite a construção de um modelo de valor DVD, a geração de uma arquitetura de serviços em SoaML, e ainda a geração de modelos de requisitos expressos numa extensão do modelo KAOS4Services, que usa um subconjunto da linguagem KAOS e onde foram incluídos conceitos de valor. Finalmente, fizemos um quase-experimento para avaliar a eficácia percebida e a intenção de uso da nossa proposta, onde os resultados mostram que a ferramenta foi considerada fácil de usar, útil, e com intenção de uso no futuro.

Palavras-chave: Modelo de negócio, Modelo de valor, Troca de valor, DSL, MDD

ABSTRACT

In a global market, it is increasingly important for companies to be able to shape their business and realize value exchanges with the network of partners they deal with. Thus, it is essential to model these exchanges of economic value through a business model that is easy to understand and use both by the user and by the software development team. However, there are few models (and methods) that allow modeling value exchanges. In addition, there are no drawing tools that align the business perspective with the software perspective.

Thus, this MSc work aims at (i) studying the existing methods to represent business value, identifying their limitations, (ii) finding a solution that facilitates the development of software based on value, and (iii) evaluating the solution using evidence-based software engineering best practices. To achieve these objectives, we began by carrying out a systematic study to ensure a comprehensive view of all the existing *value-driven* methods. Next, we selected a technological environment where we developed a tool to support the DVD method (textit Dynamic Value Description. This method allows modeling the main concepts associated with a value exchange and aligning the business perspective with the software perspective. The developed tool, DVDTool, allows the construction of a DVD value model, the generation of a service architecture described using SoaML, as well as the generation of requirements models expressed in an extension of the KAOS4Services model. Finally, we performed a quasi-experiment to evaluate the perceived efficacy and intention of use of our proposal. The results show that the tool was considered easy to use, useful, and the participants registered intention of using in the future.

Keywords: Business model, value model, value exchange, DSL, MDD

Conteúdo

Lista de Figuras

LISTA DE TABELAS

CAPÍTULO 1

INTRODUÇÃO

O trabalho desenvolvido para esta dissertação de mestrado foca a modelação de valor de negócio e a consequente geração de modelos de requisitos e de serviços. Este capítulo contextualiza e motiva o tópico estudado, descreve o problema que pretendemos resolver, define os objetivos a atingir e ainda introduz a metodologia utilizada.

1.1 Contexto

As arquiteturas orientadas a serviços (SOA) são uma prática comercial muito utilizada para o desenvolvimento de sistemas de informações empresariais [58]. Um dos principais objetivos desta arquitetura é fazer a ligação entre o produto de software e o negócio, facilitando a interação entre o desenvolvimento tecnológico e a área de negócio de uma empresa ou organização. Um serviço em Tecnologia da Informação (TI) diz respeito a alguma funcionalidade do negócio cujas principais características são ser auto-contido (baixo acoplamento) e sem estado [35]. Devido ao baixo acoplamento de serviços, as arquiteturas SOA facilitam a evolução dos sistemas de informação das empresas mais dinâmicas e que exigem mudanças constantes [35].

Faz parte dos objetivos dos métodos de análise e desenho de serviços visar a identificação dos serviços e organizá-los de uma maneira estruturada de forma a apoiar as atividades de uma empresa. Porém, existem diferentes técnicas de identificação de serviços utilizadas por estes métodos. Alguns deles seguem uma estratégia descendente, ou *top-down*, onde se decompõe o problema até atingir o nível do serviço, ou seja, o nível de refinamento onde os elementos são auto-contidos, fracamente ligados e sem estado próprio. Outros seguem uma abordagem ascendente, ou *bottom-up*, onde serviços mais elementares são compostos em serviços mais complexos e de maior nível de granularidade, até se obterem níveis de abstração idênticos aos dos processos empresariais [35]. Embora ambos os métodos forneçam uma maneira útil de identificar serviços, estes consideram o

desenho do serviço sobretudo como um problema de engenharia de software, o que não é suficiente para um alinhamento adequado entre negócios e TI [67].

Assim é necessário criar métodos que para além de permitirem a perspetiva de engenharia de software, também permitam a perspetiva de negócio visando desenvolver um alinhamento adequado entre negócios e tecnologia de informação [67]. Por exemplo, qualquer que seja a finalidade de uma empresa, esta tem a necessidade de efetuar uma gestão económica, da qual faz parte a troca de bens e serviços. Geralmente, estas trocas comerciais resultam do relacionamento com fornecedores e clientes. Numa perspetiva de negócio, os serviços fornecem *valor* aos seus clientes, que podem interagir como prestadores ou consumidores de serviços [44]. O valor é o motivo da troca entre empresas e pessoas (*troca de valor*), fornecendo um bem ou serviço com o intuito de obter algo em troca. Deste modo, o *modelo de valor* representa um modelo de negócios a partir de uma perspetiva económica, indicando o valor económico trocado entre os intervenientes [30]. Assim, é fundamental o alinhamento do desenvolvimento de software com o valor de negócio, para a identificação de serviços de software que considerem a satisfação dos clientes.

1.2 Motivação

As empresas têm a necessidade de representar as trocas de valor que efetuam com outras entidades do mercado em que vivem. Todavia, os métodos mais comuns que permitem realizar esta descrição de comportamento não permitem modelar todos os conceitos que uma transação de valor pode envolver. A variedade empresarial requer configurações e escolhas para cada domínio comercial, tornando complicada a implementação do modelo de negócios [49]. Em simultâneo, surge a necessidade das modelações serem de fácil compreensão para os especialistas, mas também para qualquer outra parte interessada que não possua conhecimentos de engenharia ou tecnológicos. De facto, têm vindo a ser desenvolvidas diversas abordagens de desenvolvimento de arquiteturas orientadas a serviços. No entanto, a generalidade dessas abordagens, apesar das contínuas melhorias, possuem como principal perspetiva a do software, deixando em falta explorar a perspetiva do negócio.

Com base num estudo sistemático que executámos (Secção 3.2.3), identificámos alguns modelos de negócio, mas apenas os modelos e3value e *Dynamic Value Description* (DVD) são suportados por métodos que guiam a construção de um sistema de informação alinhado com os valores de negócio. Estudos empíricos [59] que comparam estes dois métodos indicam que o DVD é mais eficaz e eficiente do que o e3value. Assim, escolhemos para esta dissertação de mestrado desenvolver uma ferramenta para o DVD que facilite o alinhamento entre os valores de negócio e e a sua representação em sistemas de informação.

1.3 Problema

São poucos os métodos existentes para a representação das transações que envolvem valor económico por parte das empresas e que oferecem modelos que permitem fazer o alinhamento entre tecnologias de informação e valores de negócio. Por outro lado, métodos que têm vindo a ser desenvolvidos apresentam problemas de complexidade e outras limitações como, por exemplo, a falta de ferramentas de apoio. O elevado grau de complexidade, dificulta a compreensão e utilização tanto por parte de quem desenvolve os modelos, como por parte de quem os analisa e interpreta. As limitações impostas, condicionam a representação das interações entre as entidades envolvidas. A maior parte dos métodos não tem uma ferramenta de apoio à construção e validação de modelos de valor [39], e também não há suporte ferramental integrado que auxilie os *stakeholders* durante a aplicação dos métodos que fazem a ligação entre as áreas de negócio e de software (mesmo para o caso dos métodos com suporte de alguma ferramenta de desenho dos modelos de valor) [3, 50].

O método DVD *Dynamic Value Description* oferece um processo para o alinhamento de um sistema de informação com os valores de negócio, de forma simples e de fácil compreensão para os utilizadores. No entanto, este método não fornece um suporte ferramental para ajudar os utilizadores na construção e validação do modelo de valor, nem na obtenção de modelos que permitam efetuar a ligação entre a área de negócios e de software.

1.4 Objetivo da dissertação

Os objetivos desta dissertação são:

- Estudar os métodos orientados a valor de negócio existentes e identificar as suas limitações
- Propor uma solução tecnológica para facilitar o desenvolvimento de software baseado em valor, endereçando algumas das limitações identificadas
- Avaliar a solução proposta usando as boas práticas da engenharia de software baseada em evidências

Para atingir estes objetivos, conduzimos um mapeamento sistemático da literatura com o intuito de perceber quais os métodos que existem para especificar valor de negócio e analisarmos suas limitações. A partir dessa análise, escolhemos tratar as limitações do método que consideramos mais promissor para o desenvolvimento de uma ferramenta de suporte que permita a construção de modelos de valor, e o alinhamento entre a área de negócios e tecnologias de informação. Assim, decidimos criar uma ferramenta para o método DVD por duas razões: (i) o modelo criado pelo método contém os conceitos essenciais de troca de valor, (ii) e o método proporciona uma fácil compreensão e utilização

para os seus utilizadores, facilitando a comunicação entre o utilizador e a equipa de desenvolvimento de software. Para desenvolver a ferramenta, pretendemos utilizar técnicas de desenvolvimento orientado a modelos, para a criação de linguagens de domínio especifico (DSL).

1.5 Metodologia de pesquisa

Com vista a encontrar resposta para o problema identificado (Secção 1.3) recorremos a uma metodologia de pesquisa tecnológica [55], o que incita à criação de uma solução para uma necessidade específica. Os principais passos desta metodologia são: **análise do problema**, **inovação** e **validação** [11, 38, 51, 68].

O objetivo da **análise do problema** é identificar as limitações dos métodos orientados a valor que auxiliam no alinhamento entre as áreas de negócio e tecnologia da informação Com vista a um melhor entendimento do problema é essencial compreender o domínio desse problema, através do levantamento do corpo de conhecimento existente nesse domínio. Isto permitirá identificar conceitos, técnicas e soluções consolidadas. De modo a tomar conhecimento das soluções existentes, e a maneira como estas respondem ao problema encontrado, fizemos um estudo sistemático da literatura (Secção 2.2). Um estudo sistemático é um método que permite identificar, avaliar e interpretar os trabalhos relevantes disponíveis para uma determinada questão de investigação, área de investigação, ou fenómeno de interesse [38]. Os passos principais de um estudo sistemático, incluem a identificação das questões de pesquisa, a delineação da estratégia a usar na pesquisa, a escolha de critérios de inclusão e exclusão, a extração da informação, e finalmente, a análise dos resultados obtidos face às perguntas de pesquisa iniciais.

Durante o processo de **inovação** investigam-se respostas que satisfaçam as necessidades identificadas na análise do problema (Secção 2.2.2) e ainda os recursos para a criação da solução. No nosso caso, o objetivo final é desenvolver uma ferramenta para o método DVD. Assim, é fundamental selecionar tecnologias a comparar para suportar o método DVD. Após este processo, definem-se os critérios de comparação com base nos quais se selecionam os ambientes de desenvolvimento e tecnologias associadas. Por fim, com os resultados obtidos, efetua-se a análise, concluindo, com a escolha da tecnologia a usar no o desenvolvimento da ferramenta.

Finalmente, na fase de **validação** demonstram-se e avaliam-se os resultados obtidos, avaliando se a solução criada responde às necessidades identificadas.

1.6 Estrutura do documento

Esta dissertação de mestrado está estruturada em oito capítulos, sendo o primeiro esta Introdução, e termina com uma lista de referências. O Capítulo 2, *Desenvolvimento Orientado ao Valor*, introduz algumas noções sobre modelos de negócio, discute a execução e os resultados de um estudo sistemático que fizemos sobre representação de valor de negócio,

escolhe e apresenta dois métodos dirigidos por valor que melhor se alinham com os nossos objetivos, escolhendo o método DVD para o desenvolvimento de uma ferramenta.

O Capítulo 3, *Desenvolvimento Orientado a Modelos*, apresenta uma consolidação de conhecimentos sobre desenvolvimento orientado a modelos, faz uma análise comparativa, com base num conjunto selecionado de critérios, de duas ferramentas que darão suporte ao desenvolvimento da ferramenta de apoio ao método DVD, e termina com uma dos resultados obtidos.

O Capítulo 4, *O Método DVD: Conceitos, Técnicas e Modelos*, descreve o método DVD e os seus conceitos através de uma sintaxe abstrata. Discute ainda uma extensão à linguagem orientada por objetivos KAOS, o KAOS4Services, juntamente com uma transformação que permite gerar modelos KAOS a partir de modelos DVD. O processo idêntico é seguido para discutir a geração de uma arquitetura de referência representada em SoaML a partir de um modelo DVD. Ambas as transformações e modelos alvo farão parte da ferramenta a desenvolver DVDTool.

O Capítulo 5, *DVDTool: Uma Ferramenta para o método DVD*, apresenta a ferramenta DVDTool que desenvolvemos, da qual fazem parte três editores de modelos (um para o DVD, outro para o KAOS4Services e outro ainda para o SoaML) e duas transformações M2M.

O Capítulo 6, *Caso de estudo: Loja Virtual*, demonstra todo o processo necessário para a construção de um modelo de valor DVD, bem como a geração de transformações de modelos para modelos.

O Capítulo 7, *Avaliação*, avalia a proposta desenvolvida através de um quase-experimento, quanto à facilidade de uso, utilidade e intenção de uso no futuro.

Finalmente, o Capítulo 8, *Conclusões*, resumo do trabalho desenvolvido nesta dissertação e propõe trabalhos futuros.

CAPÍTULO 2

DESENVOLVIMENTO ORIENTADO AO VALOR

De forma a fazer a ligação entre os níveis de negócio e de sistemas de informação é necessário compreendermos o modo como o mundo empresarial elabora a representação de um negócio. De acordo com a revisão da literatura apresentada por Kundisch e John [39], existem doze modelos de representação de valor. Como esta revisão sistemática da literatura se realizou em 2011, executámos um mapeamento sistemático de forma a garantir que a lista dos métodos existentes está atualizada. Posteriormente, analisámos os modelos de valor utilizados pelos métodos orientados pelo valor com o intuito de perceber quais os métodos que existem para especificar valor de negócio e quais as suas características. Dos métodos relacionados com o desenvolvimento de sistemas de informação, focámo-nos nos dois métodos que representam os conceitos básicos encontrados num modelos de valor, escolhendo o mais promissor para o desenvolvimento de uma ferramenta de suporte.

2.1 Modelos de Negócio

Um *modelo de negócio* possui uma base em ciência empresarial, engenharia de requisitos e técnicas de modelação conceptual, com vista a modelar ideias empresariais. A análise financeira de um modelo de negócio, por norma, refere-se à análise de fluxos de receita e estrutura de custos [72], com o objetivo de projetar um modelo que produza um balanço positivo [71]. O objetivo principal de um modelo de negócio é que as partes interessadas cheguem a um acordo, de modo a responder à questão: "Quem é que oferece determinado valor e quem espera um dado valor de retorno?". Portanto, o conceito primordial de um modelo de negócio é o conceito de *valor*[32].

Um *modelo de valor* é um modelo de negócios que representa como um *valor* de negócio é criado e trocado através de uma rede organizacional, facilitando a descoberta de oportunidades de negócio [31]. O valor é o motivo de troca entre as entidades constituintes de uma organização empresarial. Deste modo, um modelo de valor representa um modelo

de negócios através de uma perspetiva económica, determinando o valor económico permutado entre os intervenientes [30]. A operacionalização de uma *troca de um valor* económico é considerado um serviço de negócios que, ao nível de abstração do modelo, está subentendido na representação da troca de valor [59].

O artigo de Kundisch e John [39], apresenta os resultados de uma revisão da literatura, onde não se encontrou um conceito comum de modelo de negócio. Dada a ambiguidade nos conceitos, representação de modelo de negócio foi entendida no artigo de forma a integrar modelos de domínio específico e representações gráficas qualificadas como modelos de valor (*business model representation*, ou BMR). Este trabalho, publicado em 2012, descreve doze métodos diferentes para modelação de valor de negócio. O nosso contributo foi complementar este estudo para o período 2011-2017.

2.2 Estudo sistemático para modelação de valor de negócio

O nosso objetivo inicial era identificar quais dos métodos existentes são usados para criar modelos que representam trocas de valor de negócios. Encontrámos uma revisão de literatura publicada em 2012 que mostra doze métodos para modelar negócios, cuja maioria considera a representação do conceito de valor nos modelos criados [39]. Como esta revisão de literatura foi realizada em 2011 [39], nós fizemos uma atualização através da realização de um estudo de mapeamento sistemático para catalogar os métodos publicados na literatura entre 2011 e 2017, oferecendo uma visão geral atualizada da prática atual para a modelação de valor do negócio.

2.2.1 Método de pesquisa

Realizámos as três fases de um mapeamento sistemático (do inglês, *systematic mapping study*) [38] para oferecer uma visão alargada dos métodos existentes orientados pelo valor (do inglês *value-driven*) para representar valor de negócios e de troca de valor: Planeamento, Condução e Divulgação. A fase de **planeamento** visa definir e avaliar as questões de pesquisa e o protocolo de pesquisa que o estudo deve cumprir. A fase de **condução** é realizar a pesquisa dos estudos primários relevantes e extrair e sintetizar os dados encontrados de acordo com o protocolo. Por fim, a fase **divulgação** tem por objetivo disseminar os resultados obtidos.

Como encontrámos o estudo sistemático de Kundisch e John [39] com objetivos semelhantes ao nosso, concentrámos a nossa pesquisa no período 2011-2017, período não coberto por aquele estudo. O trabalho que desenvolvemos não é uma replicação exata do estudo original, porque aquele artigo nem sempre apresenta os detalhes suficientes.

Questões de pesquisa A definição do objetivo do nosso estudo segue o objetivo do estudo original. Em prole disso, a principal questão de pesquisa é:
Que métodos existem para especificar modelos de valor de negócio e quais as suas características?

Esta questão de pesquisa principal foi decomposta em cinco sub-questões, como descrito abaixo:

Sub-questão 1: O método foi criado para resolver um problema de qual área?

Sub-questão 2: Quais são os principais conceitos?

Sub-questão 3: Qual é o objetivo principal do método?

Sub-questão 4: Existe uma ferramenta de suporte?

Sub-questão 5: O método usa uma notação gráfica ou textual?

Em relação à sub-questão 1, a modelação de negócios abrange várias áreas diferentes, como sistemas de informação, gestão de negócios, tecnologia da informação, economia, estratégia de negócios e *e-commerce* [53]. Assim, o objetivo desta questão de pesquisa é identificar qual é a origem dos métodos selecionados. Em relação à sub-questão 2, como os modelos abrangem várias áreas diferentes, podem representar conceitos diferentes para abordar diferentes propósitos. Por isso, o objetivo desta questão de pesquisa é identificar quais são os conceitos descritos nos modelos criados pelos métodos. Em relação à sub-questão 3, o objetivo desta questão de pesquisa é identificar qual é o principal motivo pelo qual o método foi criado. Em relação à sub-questão 4, uma ferramenta de suporte é essencial para o método a ser utilizado na prática. Assim, o objetivo desta questão de pesquisa é identificar se o método possui uma ferramenta de suporte. Finalmente, em relação à sub-questão 5, uma notação gráfica pode tornar a comunicação e a compreensão de conceitos mais simples e rápida (por exemplo, sinais de trânsito). Assim, o objetivo desta questão de pesquisa é identificar o tipo de notação utilizada pelo método.

Estratégia de pesquisa A estratégia de pesquisa usada foi a *pesquisa automática*. A pesquisa automática usa os termos de pesquisa para encontrar estudos primários em bibliotecas digitais. Os termos de pesquisa e as bibliotecas digitais usadas são descritos de seguida.

Sequência de pesquisa Selecionamos um conjunto de palavras-chaves para criar a frase de consulta usada, para pesquisar estudos nas bibliotecas digitais. As palavras-chaves que usámos foram as seguintes: *business model*, *business modeling*, *business modelling*, *value chain*, *value delivery*, *value model*, *value modeling*, *value modelling* e *value network*. Com estas palavras-chaves, criámos uma frase de consulta usando os operadores lógicos *AND* e *OR*. A frase de consulta final utilizada nas bibliotecas digitais foi: "((*'value model'* OR *'value network'* OR *'value delivery'* OR *'value modeling'* OR *'value modelling'* OR *'value chain'*) AND (*'business model'* OR *'business modeling'* OR *'business modelling'*))".

Fontes de pesquisa Introduzimos a frase de consulta em mecanismos de pesquisa de bibliotecas digitais para obter estudos que respondam às questões de pesquisa. As bibliotecas digitais escolhidas foram: ACM digital library [2], IEEEXplore [34], e Science Direct [18].

Seleção dos estudos Foram definidos critérios de inclusão e exclusão para auxiliar na seleção dos estudos para análise. Incluímos todos os estudos selecionados até 2011, listados em [39] (I1). Além disso, incluímos artigos (revistos por pares) de revistas, conferências e workshops que apresentassem métodos de modelação de valor de negócios publicados desde 2012 (I2) e artigos recomendados por especialistas (I3). Adicionalmente, usámos os filtros disponíveis na biblioteca digital *Science Direct* para nos focarmos nos resultados iniciais obtidos, tendo em vista o objetivo da nossa pesquisa. Em outras palavras, incluímos estudos que pertenciam a filtros: *business model*, *business*, *service*, *value chain*, *value*, *company*, e *customer*. Por outro lado, excluímos literatura informal (apresentações de slides, revisões de conferências, relatórios informais), estudos secundários e terciários (revisões, pesquisas) e estudos de conferências, workshops e jornais sem revisão por pares (E1), estudos duplicados ou estudos com o mesmo conteúdo (E2), estudos que não responderam à questão de pesquisa (E3), e estudos que não foram escritos em inglês (E4). Nos casos de estudos que complementam trabalhos anteriores, apenas os mais recentes foram selecionados, excluindo o estudo mais antigo como duplicado (E2).

A aplicação desses critérios de exclusão ocorreu em duas fases. Na primeira fase, analisámos todos os estudos candidatos através da leitura de títulos e resumos, eliminando alguns estudos. De seguida, com os restantes estudos candidatos, realizámos a segunda fase através da leitura do estudo completo.

2.2.2 Resultados da pesquisa

A execução da pesquisa nas três bibliotecas digitais resultou num total de 1.438 estudos candidatos, que foram obtidos e importados para uma folha de cálculo. O passo seguinte foi selecionar os estudos através da aplicação dos critérios de inclusão e exclusão, o que diminuiu o número de artigos para 19 estudos relevantes. A Tabela 2.1 resume este processo.

Os estudos selecionados foram lidos integralmente e os dados relevantes foram extraídos e adicionados a uma folha de cálculo (MS excel) previamente estruturada como um formulário. A lista de todos os trabalhos selecionados pode ser encontrada na Tabela 2.2 e uma síntese dos dados extraídos desses 19 artigos é descrita na Tabela 2.3.

Tabela 2.1: Aplicação dos critérios de filtragem na pesquisa de modelação de negócios

Critérios	**de [39]**	**IEEE**	**ACM**	**Science Direct**
I1	+12	+0	+0	+0
I2	+0	+54	+21	+1363
I3	+0	+1	+0	+0
E1	-0	-3	-0	-42
E2	-0	-7	-4	-111
E3	-0	-44	-15	-1199
E4	-0	-0	-0	-7
Total	**12**	**1**	**2**	**4**

I1 - Incluído os artigos selecionados por [39].
I2 - Incluído os estudos relevantes citados pelos autores.
I3 - Incluído os estudos sugeridos por especialistas.
E1 - Excluído as literaturas informais e estudos secundários e terciários.
E2 - Excluídos os estudos duplicados ou com o mesmo conteúdo.
E3 - Excluído os estudos que não respondem às perguntas de pesquisas ou não estavam disponíveis para download.
E4 - Excluído os estudos que não eram escritos em inglês.

Tabela 2.2: Estudos selecionados para o estudo de mapeamento em modelação de negócios.

#	Artigo
S1	Porter, Michael E. "What is strategy."Harvard Business Review, 1996, pp. 61-78.
S2	Peinel, G., Jarke, M., and Rose, T., Business models for eGovernment services, Electronic Government, an International Journal, 2010, pp. 380-401.
S3	Osterwalder, A., The business model ontology: A proposition in a design science approach, University of Lausanne, 2004.
S4	Casadesus-Masanell, R. and Ricart, J. E., From strategy to business models and onto tactics, Long Range Planning, 2010, pp. 195-215.
S5	Gordijn, J. and Akkermans, H. M., Value-based requirements engineering: Exploring innovative e-commerce ideas, Requirements Engineering, 2003, pp.114-134.
S6	Weill, P. and Vitale, M. R., Place to space: Migrating to ebusiness models, HBS Press, 2001.
S7	Eriksson, H. E. and Penker, M., Business modeling with UML, Wiley, 2000.
S8	McCarthy, W. E., The REA accounting model: A generalized framework for accounting systems in a shared data environment, The Accounting Review, 1982, pp. 554-578.
S9	Samavi, R., Yu, E., and Topaloglou, T., Strategic reasoning about business models: A conceptual modeling approach, Information Systems and E-Business Management, 2009, pp. 171-198.
S10	Tapscott, D., Lowy, A., and Ticoll, D., Digital capital: Harnessing the power of business webs, HBS Press, 2000.

Continua na próxima página

Tabela 2.2 – *Continuação da página anterior*

#	Artigo
S11	Parolini, C., The value net: A tool for competitive strategy, Wiley, 1999.
S12	Pynnonen, M., Hallikas, J., and Savolainen, P., Mapping business: Value stream-based analysis of business models and resources in information and communications technology service business, International Journal of Business and Systems Research, 2008, pp. 305-323.
S13	Handoyo, Eko, Slinger Jansen, and Sjaak Brinkkemper. "Software ecosystem modeling: the value chains."Proceedings of the Fifth International Conference on Management of Emergent Digital Ecosystems. ACM, 2013.
S14	Agbabiaka, Olusegun, and Gbenga Adebusuyi. "Delivering eGovernment services through the eTrade distribution network."Proceedings of the 5th International Conference on Theory and Practice of Electronic Governance. ACM, 2011
S15	Walravens, Nils. "Qualitative indicators for smart city business models: The case of mobile services and applications."Telecommunications Policy 39.3-4 (2015): 218-240.
S16	Seidenstricker, Sven, Erwin Rauch, and Cinzia Battistella. "Business model engineering for distributed manufacturing systems."Procedia CIRP 62 (2017): 135-140.
S17	Ongena, Guido, Erik Huizer, and Lidwien van de Wijngaert. "Threats and opportunities for new audiovisual cultural heritage archive services: The Dutch case."Telematics and informatics 29.2 (2012): 156-165.
S18	Kolsch, P., et al. "A novel concept for the development of availability-oriented business models."Procedia CIRP 64 (2017): 340-344.
S19	Souza, E., et al. "An approach to align business and IT perspectives during the SOA services identification". Computational Science and Its Applications (ICCSA), 2017 17th International Conference on. IEEE, 2017..

Tabela 2.3: Síntese dos dados extraídos dos artigos sobre modelação de negócios.

#	Nome do método	Origem (área)	Principais conceitos	Principal objetivo	Tem ferramenta de suporte?	Tem notação gráfica?
S1	Activity system map	Estratégia de negócio	Tema estratégico, atividade	Facilitar a compreensão e criação de estratégias de negócios	Não	Sim
S2	BMeG	E-governo	Parceiro, objetos, troca de objetos, (des)vantagem	Apoiar o planeamento de modelos de negócios para serviços de e-Governo	Sim	Não
S3	BMO	Negócio eletrónico	Blocos de construção inter-relacionados, valor, configuração de valor	Propor um modelo conceptual de modelos de negócios	Sim	Não
S4	Causal loop diagram	Estratégia de negócio	Escolha, consequência	Propor uma separação entre tática e estratégia usando a teoria da causalidade	Não	Sim
S5	e3value	Sistemas de informação	Ator, segmento de mercado, valor, troca de valor	Desenvolver um sistema de informações intensivo para comércio eletrónico	Sim	Sim
S6	E-business model schematics	E-business	Ator, valor, fluxo, relação	Migração de negócios para e-business	Não	Sim

Continua na próxima página

Tabela 2.3 – *Continuação da página anterior*

#	Nome do método	Origem (área)	Principais conceitos	Principal objetivo	Tem ferramenta de suporte?	Tem notação gráfica?
S7	Eriksson-Penker business extensions of the Unified Modeling Language	Sistemas de informação	Ator, interação, objetivo, regra	Conhecer todos os casos de uso (ou os corretos) que melhor suportam o negócio no qual o sistema opera	Sim	Sim
S8	REA	Contabilidade de negócios	Recurso, evento, agente, contrato económico	Propor uma estrutura de contabilidade generalizada projetada para ser usada num ambiente de dados compartilhado	Sim	Sim
S9	SBMO	Estratégia de negócio	Ator, objetivo	Ajudar a entender e analisar as metas, intenções, papéis e a lógica por trás das ações estratégicas em um ambiente de negócios	Sim	Não
S10	Value map	Estratégia de negócio	Ator, valor, troca de valor	Acompanhar o fluxo de receita, o fluxo de conhecimento e intangíveis	Não	Sim
S11	Value net	Estratégia de negócio	Ator, atividade, fluxo	Ajudar a criar uma nova perspetiva na análise estratégica	Não	Não
S12	Value stream map	Estratégia de negócio	Ator, fluxo de valor	Mapear os fluxos de valor entre os atores para entender o negócio	Não	Sim

Continua na próxima página

Tabela 2.3 – *Continuação da página anterior*

#	Nome do método	Origem (área)	Principais conceitos	Principal objetivo	Tem ferramenta de suporte?	Tem notação gráfica?
S13	SECO	Ecossistemas	Atores, fluxo	Facilitar a compreensão dos modelos de negócios elementares e cadeias de valor de ecossistema de software	Não	Sim
S14	eTrade	E-governo	Revendedor, distribuidor, grossista, retalhista, cidadão	Apresentar o conceito da rede de distribuição eTrade como um modelo para melhorar o acesso a serviços e produtos de eGovernment em um país em desenvolvimento	Sim	Não
S15	N/A	E-governo	Controle, valor	Analisar modelos de negócios que envolvem atores públicos, e governos municipais em particular, na rede de valor	Não	Não
S16	Bussiness Model Engineering	Estratégia de negócio	Produto / Serviço, ganhos, perdas, tempo	Alcançar excelência de produção em cada unidade de produção e garantir a probabilidade estratégica para aprimorar os sistemas de manufatura distribuída	Não	Não

Continua na próxima página

Tabela 2.3 – *Continuação da página anterior*

#	Nome do método	Origem (área)	Principais conceitos	Principal objetivo	Tem ferramenta de suporte?	Tem notação gráfica?
S17	STOF	Estratégia de negócio	Segmento de mercado, funcionalidade, estrutura de custos, potencial de lucro, estrutura da rede de valor	Analisar o mercado *business-to-consumer* para serviços de arquivo audiovisual digital	Não	Não
S18	N/A	PSS[1]	Fluxo monetário, comunicação, dados, saída	Desenvolver modelos de negócios orientados para a disponibilidade	Não	Sim
S19	DVD	Sistemas de informação	Atores, troca de valor, objeto de valor, nível de valor acordado	Desenvolver um modelo de negócio com rigor suficiente para criar softwares baseados nos valores do negócio	Não	Sim

2.2.3 Resposta às questões da pesquisa

Esta secção discute os resultados, respondendo a cada questão de pesquisa.

Sub-questão 1: O método foi criado para resolver um problema de qual área? Oito dos 19 estudos (42,1%) foram criados para solucionar algum problema relacionado com a área de estratégia de negócios [S1, S4, S9, S10, S11, S12, S16, S17]. Outras áreas com mais artigos são o e-governo e sistemas de informação [S5, S7, S19] com 3 estudos (15,7%) cada, seguido de e-business [S3, S6] com 2 estudos (10,5%). As áreas com menos estudos são contabilidade de negócios [S8], ecossistemas [S13] e sistemas de produtos e serviços [S18] com 5,2% cada (1 de 19 estudos cada).

[1] PSS são modelos de negócios para fornecer uma entrega coesa de produtos e serviços.

Sub-questão 2: Quais são os principais conceitos? Embora os estudos compartilhem um conjunto de conceitos comuns (mesmo que às vezes com nomes diferentes), como atores (parceiro, agente, revendedor, distribuidor, grossista, retalhista ou cidadão) [S2, S5, S6, S7, S8, S9, S10, S11, S12, S13, S19], recursos (valor, objeto, produto, serviço ou dados) [S2, S4, S5, S6, S8, S10, S12, S15, S16, S17, S18, S19] e transferência de recursos (troca de valor, troca de objeto, evento, fluxo, fluxo de valor ou fluxo monetário) entre os atores [S2, S5, S6, S8, S10, S11, S12, S13, S18, S19], também são caracterizados por algumas diferenças , como *configuração de valor* em BMO (macro-processo necessário para criar valor para o cliente), *contrato económico* em REA (agregação de compromissos económicos) e *segmento de mercado* em e3value (grupo de atores semelhantes). Em relação aos métodos relacionados com o desenvolvimento de sistemas de informação, apenas os métodos DVD [S19] e e3value [S5] representam os conceitos básicos encontrados em um modelo de valor [5].

Sub-questão 3: Qual é o objetivo principal do método? Para uma melhor estruturação da resposta, iremos agregar os objetivos dos estudos de acordo com as áreas a partir das quais eles foram criados. Os estudos com foco em estratégias de negócios, destinam-se a facilitar a compreensão [S1, S4, S9, S10, S12] e análise de negócios [S1, S9, S11, S16, S17] para melhorar ou garantir estratégias de negócios. Em relação aos estudos com foco em e-governo, estes pretendem apoiar [S2], analisar [S15] e melhorar [S14] serviços e produtos governamentais. Os estudos de e-business concentram-se na migração de negócios para o e-business [S6] e tentam padronizar os modelos de e-business com a criação de modelos conceptuais [S3]. Os estudos da área de sistemas de informação focam-se na fase de engenharia de requisitos de desenvolvimento de software, mais especificamente, concentram-se na criação de uma especificação de requisitos para desenvolver sistemas de informação que suportem melhor o negócio [S5, S7, S19]. Os outros estudos focam-se numa melhor compreensão da contabilidade empresarial [S8] e do ecossistema [S13], bem como no desenvolvimento de modelos de negócios orientados para a disponibilidade [S18].

Sub-questão 4: Existe uma ferramenta de suporte? A maioria dos estudos não possui nenhum tipo de ferramentas de suporte (12 de 19 estudos — 63,1 %) [S1, S4, S6, S10, S11, S12, S13, S15, S16, S17, S18, S19]. Sete estudos (36.8%) têm ferramentas para apoiar os seus métodos [S2, S3, S5, S7, S8, S9, S14]. Destes, destacamos dois que possuem uma ferramenta para criar um modelo de valor de negócio [S5, S7].

Sub-questão 5: O método usa uma notação gráfica ou textual? 11 estudos (57.8%) usam a notação gráfica para representar os seus principais conceitos num modelo de negócio [S1, S4, S5, S6, S7, S8, S10, S12, S13, S18, S19]. Em contraste, 8 estudos (42,1 %) usam apenas a notação textual [S2, S3, S9, S11, S14, S15, S16, S17]. Embora haja uma diferença no número de estudos que usam uma notação gráfica e uma notação textual, essa diferença não é

estatisticamente significativa. Considerando os três estudos de sistemas de informação [S7, S5, S19], todos usam uma notação gráfica para representar os conceitos de valor. No entanto, o [S7] usa uma abordagem mais centrada na tecnologia para criar modelos de valor (cria um perfil UML, tornando-o menos atraente para a comunidade de negócios [19]). [S5] usa uma abordagem mais voltada para a comunidade de negócios (criando um novo modelo, mas sem o rigor necessário para uma especificação de requisitos de software). Finalmente, [S19] estrutura o modelo com base em mapas mentais, o que facilita o seu uso tanto pela comunidade de negócios como pela comunidade de tecnologia [59].

2.2.4 Principais Constatações

A maioria dos estudos foram criados para solucionar algum problema relacionado com a estratégia de negócios. No entanto, encontrámos três estudos de sistemas de informação que se concentram na criação de modelos de negócios com o objetivo de melhorar o alinhamento entre as áreas de negócio e tecnologia [S5, S7, S19].

Os estudos partilham um conjunto de conceitos comuns, como atores, valor e troca de valores, com o objetivo de facilitar, geralmente, a compreensão e a análise de negócios para melhorar as estratégias de negócios. Considerando apenas os métodos relacionados com o desenvolvimento de sistemas de informação, apenas os métodos DVD [S19] e e3value [S5] representam os conceitos básicos encontrados num modelo de valor [5]. A maioria dos estudos não possui uma ferramenta de suporte. Entre os três estudos relacionados ao desenvolvimento de sistemas de informação, apenas o DVD [S19] não possui uma ferramenta de suporte. Isso prejudica a sua utilização.

Por fim, à uma quantidade maior de estudos que focam na notação gráfica para representar os conceitos usados nos modelos de negócio, porém a diferença na quantidade de estudos com notação gráfica e textual não é significativamente diferente. Considerando os três estudos relacionados com os sistemas de informação, todos usam uma notação gráfica, porém, à exceção do DVD [S19], os estudos não fornecem o rigor necessário para o desenvolvimento de software (e3value [S5]) ou são difíceis de serem usados por especialistas em negócios (Ericsson-Penker [S7]).

Assim, de entre todos os métodos estudados, escolhemos o e3value [67] e o DVD [60] para uma discussão mais detalhada. O método e3value é muito conhecido para modelação de negócios [39, 53]. O DVD, por outro lado, foi desenvolvido para facilitar a comunicação entre as partes interessadas do negócio e de TI e usar os valores do negócio para impulsionar o desenvolvimento de um sistema de informação. Ambos os métodos se destinam a facilitar o alinhamento entre áreas de negócios e desenvolvimento de software [30, 57] e representam os conceitos básicos encontrados num modelo de valor.

2.3 Os Métodos e3Value e DVD

A descrição dos métodos e3value e DVD centrar-se-á no modelo que cada um propõe para a especificação de trocas de valor, assim como o respetivo processo de construção.

2.3.1 Método e3value

O método e3Value permite representar trocas de valor entre atores, numa perspetiva de negócio, permitindo a avaliação da viabilidade e rentabilidade dos modelos de valor [67]. Este método usa um conjunto de conceitos, oferece uma técnica de construção do modelo e3Value, e fornece um processo próprio que ajuda a construção do modelo.

Modelo e3value. O modelo e3Value usa os seguintes conceitos [67]:

- **Ator:** uma entidade economicamente independente, e muitas vezes legal (por exemplo, uma empresa ou um consumidor final).
- **Segmento de mercado:** um conjunto de atores.
- **Atividade:** atividades operacionais atribuídas como um todo a um ator.
- **Objeto de valor:** recurso trocado entre dois atores, com um valor económico para pelo menos um dos atores (serviço, produto ou experiência).
- **Porta de valor:** indica se um objeto de valor flui para dentro ou para fora de um ator (por exemplo, para dentro, receber mercadoria, para fora, fazer um pagamento).
- **Interface de valor:** uma porta de entrada de valor e uma porta de saída de valor, pertencentes ao mesmo ator. Usadas para representar reciprocidade económica.
- **Troca de valor:** uma potencial troca de objetos de valor entre duas portas de valor.

O modelo e3value permite representar cenários de trocas de valor, através dos seguintes elementos [59]:

- **Estimulo de inicialização:** evento que desencadeia o início do cenário.
- **Estimulo de finalização:** evento que termina o cenário.
- **Elemento de conexão:** representa o caminho do cenário.
- **Elemento AND:** divide um caminho em dois ou mais sub-caminhos.
- **Elemento OR:** combina dois ou mais sub-caminhos em um.

A Figura 2.1 apresenta um exemplo de um modelo de valor e3value para um sistema equilibrado do mercado de eletricidade.

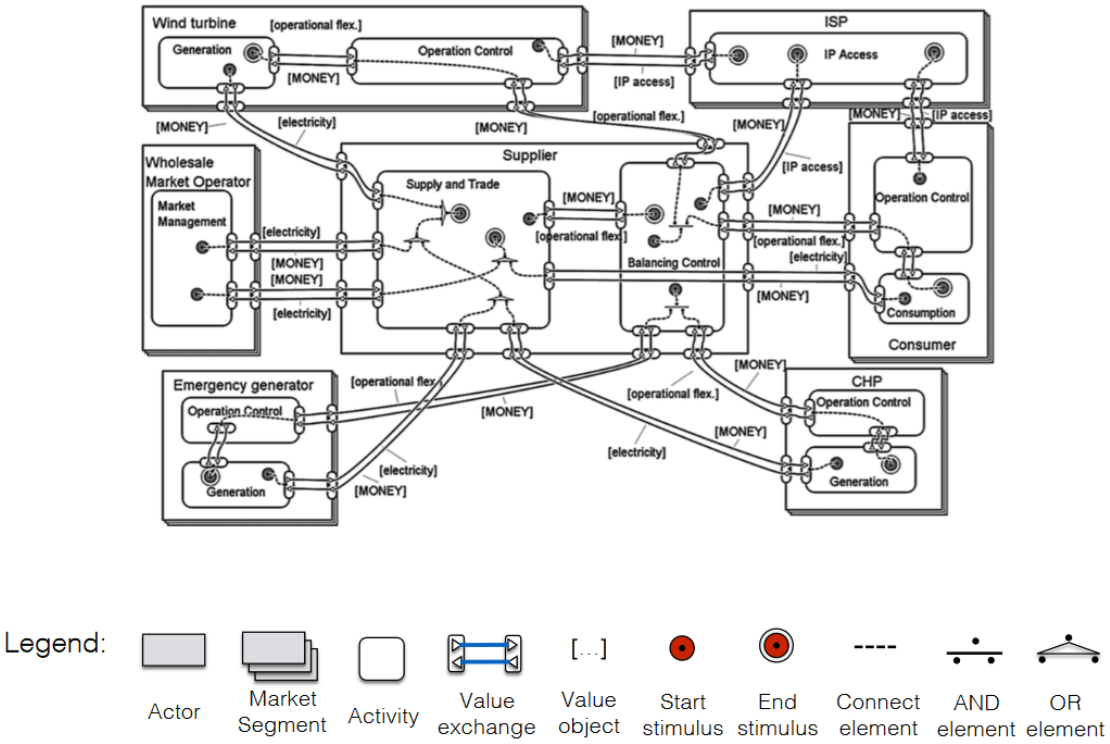


Figura 2.1: Modelo de valor e3value para um sistema de mercado de eletricidade [52]

Processo e3value. Para a construção de um modelo de valor e3value começa-se com a criação de uma lista de cenários, e a identificação dos atores. De seguida cria-se o modelo e3value inicial, através dos produtos e serviços mencionados na lista de cenários, e dos atores identificados. Por último, é feita a introdução dos elementos de modo a representar os caminhos de todos os cenários. A Figura 2.2 apresenta as etapas deste processo de criação do modelo de valor e3value [59].

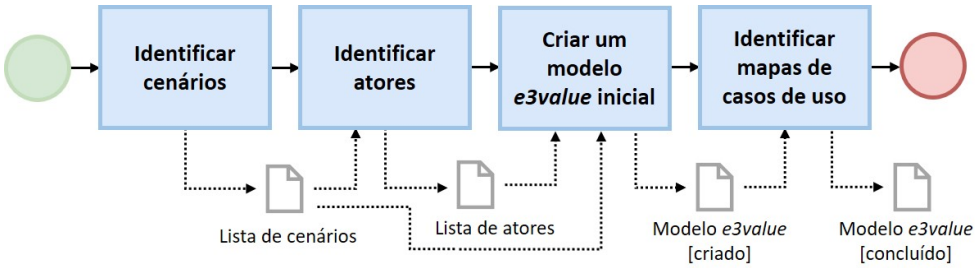


Figura 2.2: Processo de criação de um modelo e3value [59]

Adicionalmente, Weigand usa o modelo e3value e propõe o método *Service Design* para identificar serviços [67]. Este método inicia com a criação de um modelo de valor, onde são representadas as atividades comerciais a partir de uma perspetiva económica. No modelo e3value os serviços ocorrem como trocas de valor. Nesta primeira fase do

método *Service Design* é feita uma abstração dos serviços (principais, complementares e de apoio), tal como as características de qualidade associadas a esses serviços. Na fase seguinte, são identificados serviços comerciais (de coordenação), para além dos identificados anteriormente. Esta identificação inclui a especificação das regras e políticas de negócio que regem os serviços. Na fase posterior, são identificados serviços de software, ao nível da informação e de infraestrutura. Por último, é desenhado uma estrutura de suporte para os serviços informativos [67].

2.3.2 Método *Dynamic Value Description*

Dynamic Value Description (DVD) é um método de engenharia de software baseado em valor que deriva uma arquitetura de serviços de negócio e modelos de requisitos alinhados com os valores económicos de um negócio. Este método usa um conjunto de conceitos, oferece uma técnica para construção do modelo DVD e para geração automática de modelos SoaML e KAOS4Services usando técnicas de engenharia orientada a modelos (*model-driven engineering*). Além disso, o método DVD também fornece um processo próprio e um conjunto de heurísticas que facilitam o seu uso por utilizadores menos experientes [60].

Modelo DVD. Este modelo permite analisar e representar as trocas de valor económico de uma empresa, fornecendo um ambiente em que as partes interessadas podem partilhar as suas visões económicas [58]. Os principais conceitos deste modelo são [59]:

- **Ator:** uma entidade economicamente independente. Podendo ser de dois tipos: ator principal (o foco do modelo) ou ator secundário (ou de ambiente).
- **Objeto de valor:** recurso trocado entre dois atores, com um valor económico para pelo menos um dos atores (dinheiro, bens ou serviços).
- **Porta de valor:** indica se um objeto de valor flui para dentro ou para fora de um ator.
- **Troca de valor:** uma transferência de objetos de valor, através de duas portas de valor.
- **Quem inicia a troca de valor:** indica o ator que inicia uma troca de valor.
- **Nível do valor acordado** (*Value Level Agreement*, ou VLA): regra mínima do negócio acordado entre os atores envolvidos.

O foco do analista de negócios define o ator principal. No entanto, o foco muda ao longo do processo de especificação. Consoante as mudanças de foco, novos modelos são construídos, e consequentemente, novos atores e trocas de valor poderão ser identificados. A cada novo ator principal é identificada a sua relação com os atores secundários, dando origem a uma rede interorganizacional [58]. A Figura 2.3 apresenta um exemplo de um modelo de valor DVD para o mesmo exemplo de um sistema equilibrado para o mercado de eletricidade.

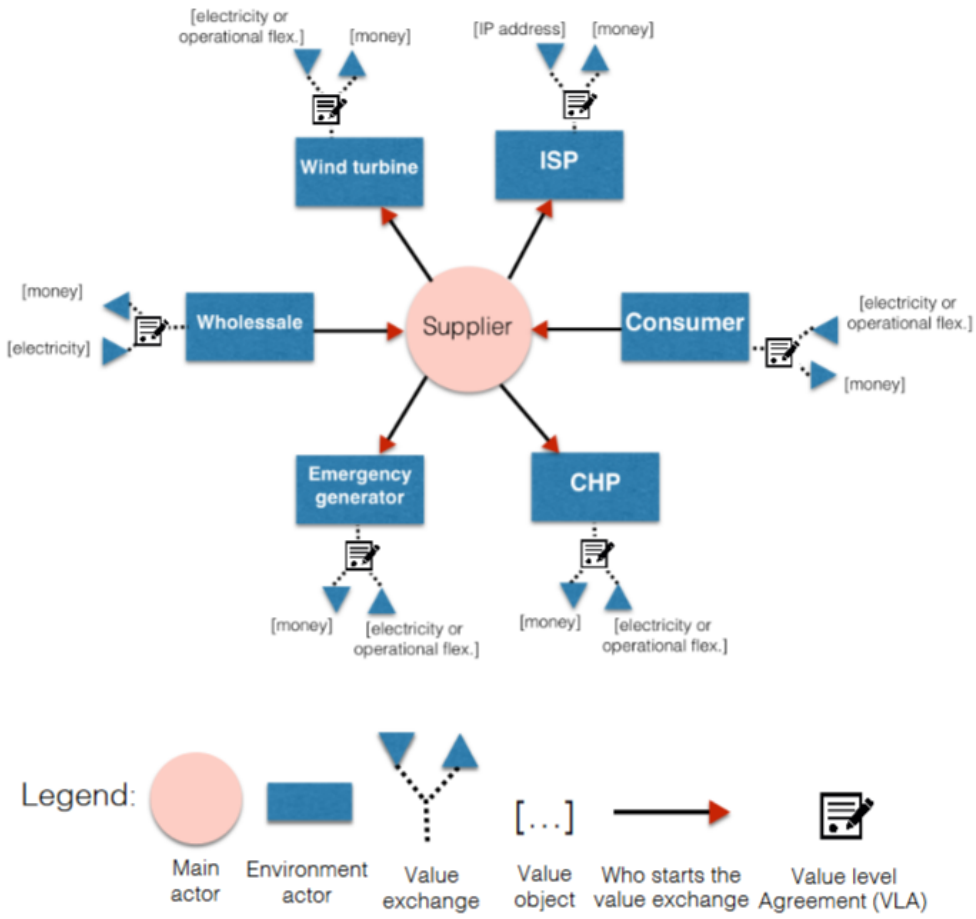


Figura 2.3: Modelo de valor DVD para um sistema de mercado de eletricidade

Processo DVD. O processo de criação de um modelo DVD inicia com o desenho do ator principal e os seus atores secundários. (É de salientar que devido ao foco no ator principal, é necessário criar tantos modelos quantos os necessários para representar todo o modelo de negócio.) De seguida, são adicionadas as trocas de valor ao modelo, indicando o objeto de valor para cada porta. Posteriormente é indicado qual o ator que inicia a troca de valor, verificando se os objetos de valor estão indicados nas portas de valor corretas. Por último, são definidos os critérios necessários para a troca de valor, crucial para entender as restrições de negócios relacionadas a cada troca de valor. A Figura 2.4 apresenta as etapas deste processo de criação do modelo de valor DVD [59].

O método DVD propõe uma abstração de serviços de negócio, através do modelo de valor DVD. Permitindo gerar as entradas (ou *inputs*) necessárias aos métodos de serviços de informação [56]. Este método começa com a criação de um modelo de valor DVD. Depois escolhe-se um modelo *goal-oriented* alvo para a transformação de modelo para modelo. Faz-se a análise dos conceitos desse modelo escolhido. Isto resulta num mapeamento de conceitos, e num conjunto de heurísticas, entre o modelo DVD e o modelo *goal-oriented* escolhido. Por último, é criado o esquema de transformação de modelo para

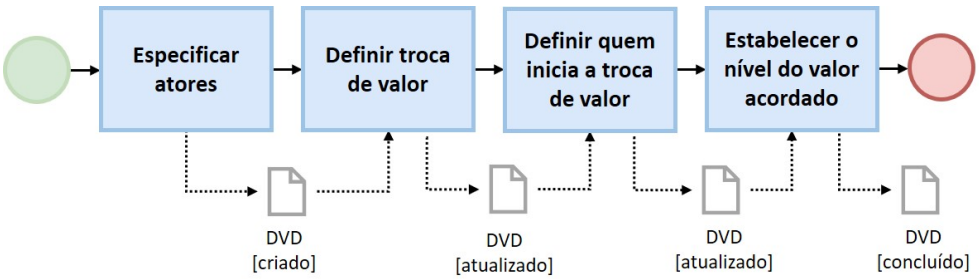


Figura 2.4: Processo de criação de um modelo DVD [59]

modelo (M2M), respeitando o conjunto de heurísticas [56].

2.4 Escolha do Método para a Ferramenta a Desenvolver

Os métodos e3value e DVD foram criados com objetivos semelhantes, formar uma ligação entre modelos de valor e sistemas de informação. Foi executada uma experiência controlada com o intuito de comparar os modelos DVD e e3value em relação à eficiência, eficácia, facilidade de utilização, utilidade, e intenção de usar [59]. Os resultados da análise dos dados obtidos indicaram que, o tempo médio para a criação do modelo (eficiência) foi maior para representar um modelo e3value do que para representar um modelo DVD, resultando, por isso, numa maior eficiência por parte do DVD. Em relação à exatidão e completude (eficácia) com que foram criados, os dados mostram que o DVD é mais eficaz. Dados que se justificam com o facto de o modelo e3value se tornar mais complexo com a junção de conceitos estáticos (por exemplo, objetos) e dinâmicos (por exemplo, cenários). Os resultados representam uma perceção da facilidade de utilização maior para o DVD, em relação ao e3value. Não entanto, em relação à utilidade não é apresentada uma diferença significativa. Este resultado deve-se ao facto de os modelos partilharem o mesmo objetivo e representarem os mesmos conceitos principais. Tendo em conta a intenção de uso, os resultados apresentam valores significativamente maiores para o DVD [59].

Foi também realizada outra experiência (*quasi-experiment*) de forma a avaliar a eficácia, facilidade de utilização e utilidade da criação de um modelo de valor, e a viabilidade de gerar modelos-alvo a partir do modelo DVD (através da utilização de uma ferramenta para provas de conceitos). Os resultados mostram que é uma abordagem promissora para o alinhamento entre perspetivas de negócio e sistemas de informação, SOA [58].

Tanto o método e3value como o método DVD não apresentam uma ferramenta para a identificação de serviços. Contudo, o método e3value tem uma ferramenta para modelar valor, enquanto que o método DVD apresenta apenas uma ferramenta para provas de conceito. Por estas razões, pretende-se desenvolver uma ferramenta de suporte para o método DVD.

2.5 Conclusões

Este capítulo relata um estudo sistemático para modelação de valor de negócio, tendo por base a lista de representações de modelos de Kundisch e John [39]. Para este estudo sistemático, a pergunta de pesquisa questiona quais os métodos existentes para especificar modelos de valor de negócio e quais as suas características. Foi selecionado um conjunto de palavras-chave, de modo a formar uma frase de consulta a inserir nas bibliotecas digitais. Posteriormente, definimos uma estratégia de pesquisa onde foram indicados tanto os critérios de inclusão, como os critérios de exclusão. Os 19 estudos selecionados foram analisados para extração de dados para responder às perguntas de pesquisa deste estudo.

Com base nos resultados obtidos do estudo sistemático comparámos as características dos vários métodos, e concluímos que apenas dois abordavam o conceito de troca de valor, e utilizavam uma notação gráfica. Esses métodos eram o e3value e o DVD. Cada um destes métodos foram discutidos com mais detalhe, apresentando os respetivos conceitos, modelos e processos. Experiências controladas anteriores [58] revelaram que o DVD era um método promissor. Assim, a construção de uma ferramenta de suporte ao método DVD será uma contribuição significativa desta tese de mestrado.

CAPÍTULO 3

DESENVOLVIMENTO ORIENTADO A MODELOS

Um dos objetivos deste trabalho é desenvolver um editor para o método *Dynamic Value Description* que facilite a criação de modelos de valor e suas linguagens de transformação. Para isso, propõe-se uma linguagem de domínio específico (DSL), recorrendo a tecnologias MDD (ou *model-driven development)*. Assim, este capítulo começa por introduzir os conceito base do desenvolvimento orientado a modelos e de seguida faz uma análise comparativa de duas ferramentas de modelação de DSLs. Esta análise inicia com a seleção das ferramentas a comparar, define os critérios utilizados para essa comparação, e após a execução da comparação, demonstra e analisa os resultados obtidos.

3.1 Engenharia Orientada a Modelos

A engenharia orientado a modelos (ou *Model-Driven Engineering (MDE)*) oferece um conjunto de técnicas e boas práticas para auxiliar o desenvolvimento de software centrado em *modelos* em vários níveis de abstração. O seu objetivo é gerar software automaticamente [43]. Os modelos utilizados estão relacionados uns com aos outros através de relações bem definidas que permitem mapeamentos entre eles. Por sua vez, esses modelos estão baseados em *metamodelos* [20]. Numa abordagem orientada a modelos, os metamodelos são utilizados para implementar *transformações* de modelos, e para criar *linguagens de domínio especifico* [43].

Um **modelo** é uma representação abstrata de um sistema (ou uma teoria ou realidade) que permite fazer inferências e previsões sobre o mesmo [40]. Estas representações simplificadas são formuladas através de uma linguagem de modelação. Um **metamodelo** é um modelo para definir modelos [43]. Deste modo, um modelo é uma instância de um metamodelo, e um metamodelo compreende todos os modelos que possam ser expressos através dele [43]. As **transformações de modelos** são feitas tendo em conta os modelos de origem e os modelos alvo. Para efetuar uma transformação é necessário um mapeamento

entre os conceitos dos metamodelos dos respectivos modelos envolvidos na transformação [43]. Isto é, para transformar um modelo A num modelo B é necessário um mapeamento entre os conceitos do metamodelo do modelo A e os conceitos do metamodelo do modelo B. Desta forma, a transformação do modelo A gera um novo modelo B. A Figura 3.1 ilustra o processo de transformação de modelo para modelo (M2M).

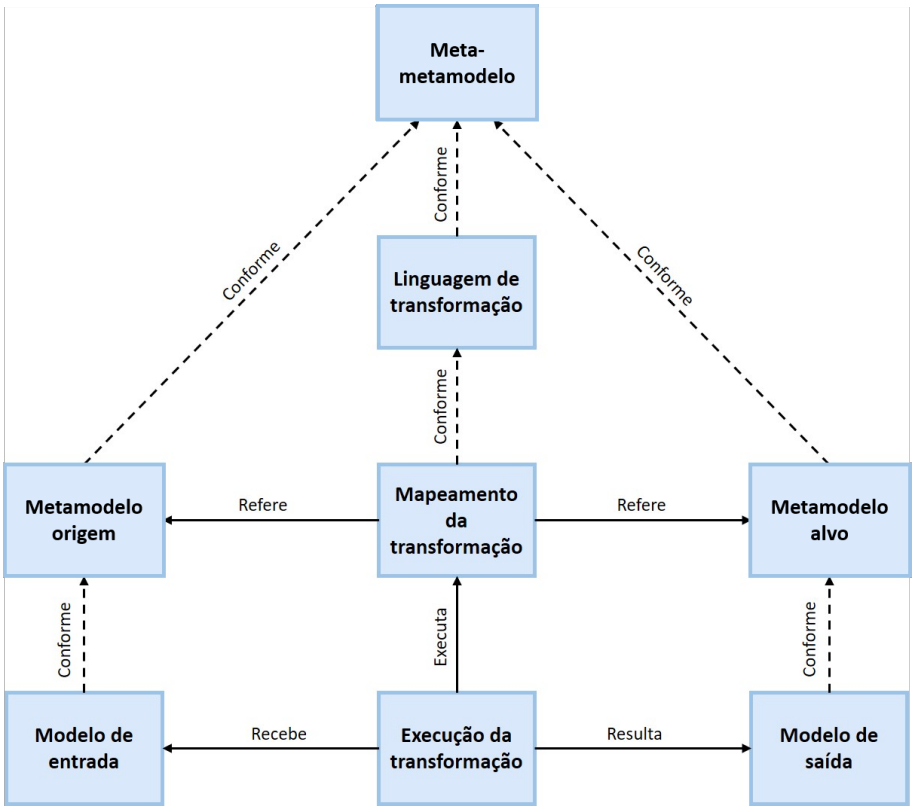


Figura 3.1: Processo de transformação de modelo para modelo (M2M) [22]

As **linguagens de domínio especifico** (do inglês *Domain Specific Languages*, ou DSL) são linguagens não genéricas, especializadas num domínio específico de problemas [65]. Ou seja, uma DSL, ao contrário de uma linguagem de propósito geral, é criada com o intuito de resolver problemas de domínio especifico, para um conjunto especifico de utilizadores, proporcionando uma melhor usabilidade, e consequente aumento de produtividade [65]. Uma DSL é composta por dois tipos de sintaxe, a *sintaxe abstrata* e a *sintaxe concreta*. A **sintaxe abstrata** é uma estrutura de dados, representada por um metamodelo, que define os elementos da linguagem e as regras através das quais estes se relacionam. A **sintaxe concreta** define a notação através da qual os utilizadores podem representar diferentes especificações do domínio [65].

Para desenvolver uma linguagem de domínio especifico é necessária uma ferramenta de apoio à construção que permita definir a sintaxe abstrata e a sintaxe concreta da linguagem.

3.2 Análise de Ferramentas de Modelação de DSLS

Nesta secção apresentamos uma análise comparativa de duas ferramentas de modelação de DSLs: EuGENia [27] e Sirius [29]. (Esta comparação foi feita em colaboração com o aluno de Mestrado Tiago Silva.)

3.2.1 Ferramentas a Comparar e Critérios de Comparação

Selecionámos as duas ferramentas de modelação mais conhecidas para efetuar a análise comparativa: *Epsilon EuGENia* [27] e *Sirius* [29]. Enquanto o *Epsilon EuGENia* foi utilizada no âmbito do meu percurso académico, a escolha do *Sirius* [29] resulta da análise de estudos que comparam várias ferramentas de modelação [33, 64]. Ambas as ferramentas são fornecidas pela Fundação Eclipse, estando por isso disponíveis no *framework* Eclipse. Dos estudos analisados constavam ferramentas como *Diagen*, *Eugenia*, *GMF*, *Graphiti*, *MetaEdit+*, *Obeo Designer*, *Sirius*, e *Tiger*.

A análise dos dois estudos anteriores [33, 64] ajudou-nos a identificar os critérios de comparação necessários para esta nossa análise. Dos critérios usados nesses estudos, selecionámos os que têm influência na escolha da ferramenta a utilizar para o desenvolvimento dos editores e transformações. A Tabela 3.1 apresenta uma breve descrição de cada um dos critérios selecionados.

Tabela 3.1: Critérios de comparação das ferramentas, subconjunto extraído de [33, 64].

Critérios	Descrição
Documentação Oficial	Quantidade e qualidade da documentação oficial das ferramentas.
Comunidade	Qualidade dos fóruns de discussão e outros recursos, e aceitação dos utilizadores.
Edições	Diferentes versões existentes das ferramentas.
Esforço	Número de passos necessários e complexidade associada, de modo a obter um editor.
Sintaxe Abstrata	Tipo de modelação necessária para a criação de um editor gráfico.
Sintaxe Concerta	Notação na qual o utilizador interage com a linguagem.
Transformações M2M	Suporte da ferramenta para transformações de modelo para modelo.
Transformações M2T	Suporte da ferramenta para transformações de modelo para texto.

3.2.2 Execução

Para que fosse possível comparar as ferramentas de modelação de DSLs, tendo em conta os critérios de comparação selecionados, efetuámos uma recolha de dados com recurso

às páginas das respetivas ferramentas *Web*[1,2]. Desta forma, obtivemos os resultados quantitativos para cada uma das ferramentas. Posteriormente, desenvolvemos uma análise qualitativa dos dados obtidos. Esta análise teve como objetivo avaliar a qualidade da documentação oficial, a dimensão e evolução da comunidade, as versões, bem como as diferenças existentes quanto à sintaxe concreta, sintaxe abstrata, transformações M2M, e transformações M2T. Para esta avaliação procedemos ao desenvolvimento de um editor para a modelação de um pequeno exemplo, o semáforo. Este semáforo é composto por uma luz vermelha, amarela e verde, e uma luz preta para indicar que este está desligado. Assim, modelaram-se as transições entre as luzes do semáforo, tal como a transição de ligado para desligado. A Figura 3.2 mostra o metamodelo correspondente. O editor criado permite também a modelação de um polícia que, ao ser acionado, muda automaticamente a cor do semáforo para vermelho, de acordo com o diagrama da Figura 3.3.

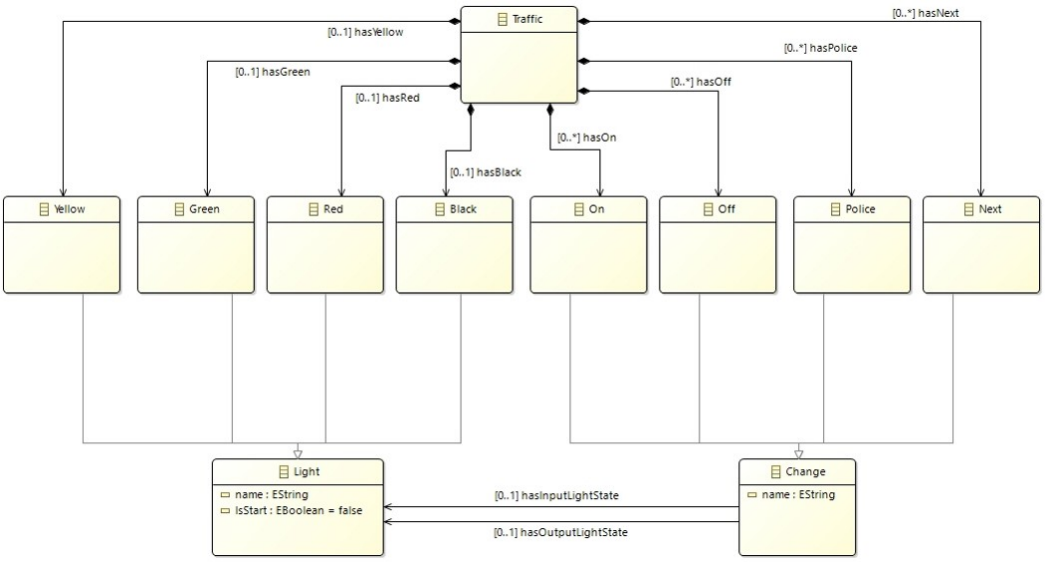


Figura 3.2: Metamodelo do semáforo

[1]http://www.eclipse.org/sirius/overview.html
[2]http://www.eclipse.org/epsilon/doc/eugenia/

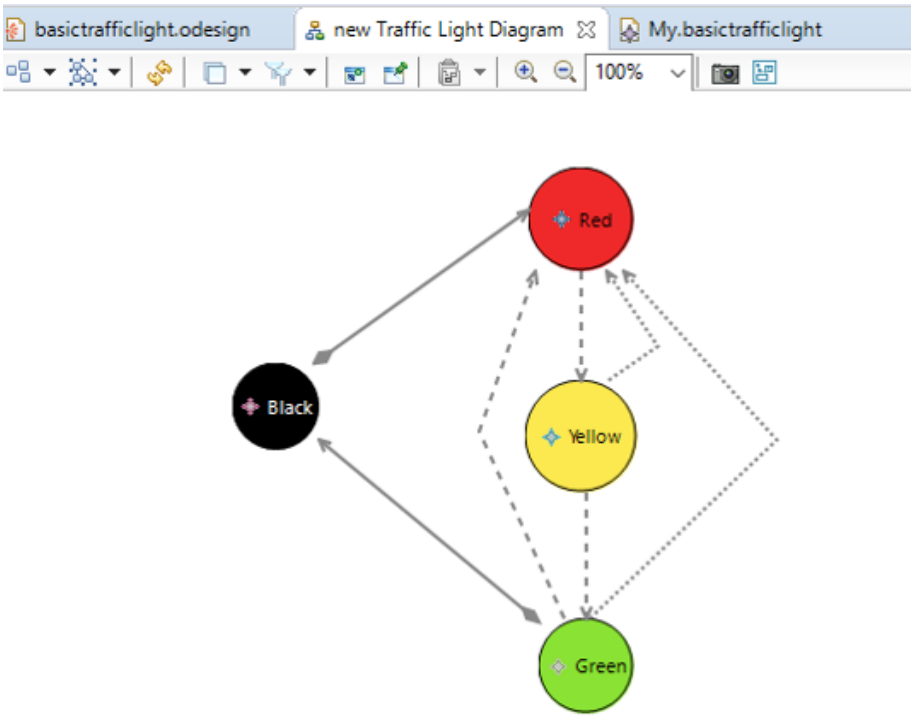


Figura 3.3: Editor de semáforo criado na ferramenta *Sirius*

O procedimento de desenvolvimento do editor do semáforo para a ferramenta *Epsilon EuGENia* a partir do metamodelo foi o seguinte: geração do ficheiro *EMFatic Source*, edição do ficheiro *EMFatic Source*, geração do editor *GMF*, e a exportação dos *plug-ins*. Por outro lado, o procedimento de desenvolvimento do editor semáforo para a ferramenta *Sirius* a partir do metamodelo foi o seguinte: geração do ficheiro *EMFatic Source*, criação do *Sirius Base Model*, e definição da *Viewpoint Specification*

Este processo permitiu medir o esforço para a criação do editor em ambas as ferramentas, *Epsilon EuGENia* e *Sirius*, bem como testar as capacidades destas quanto aos critérios selecionados na Secção 3.2.1. Adicionalmente, o desenvolvimento deste exemplo contribuiu também para compreender o processo de criação de um editor em cada uma das ferramentas.

3.2.3 Análise de Resultados

Nesta secção, resumimos os resultados obtidos e as conclusões retiradas da análise comparativa relativamente a cada critério de comparação. Para cada critério, apresentam-se os resultados encontrados para cada ferramenta e ainda um pequeno parágrafo de conclusão comparativa.

Documentação Oficial

Epsilon EuGENia: Encontrámos um manual de utilizador introduzindo os conceitos teóricos da ferramenta e alguns exemplos práticos; trinta artigos referentes às linguagens utilizadas pela ferramenta; vinte exemplos de partes de código, não contextualizado, sem explicação passo a passo; quinze vídeos no canal *Youtube*.
Sirius: Encontrámos dois manuais (um de utilizador, outro de especificação) com exemplos passo a passo ilustrados com imagens; quatro tutoriais com exemplos passo a passo; vinte vídeos no canal *Youtube*; vários exemplos de projetos.
Comparando: A ferramenta *Sirius* apresenta documentação em maior quantidade e com melhor qualidade. A documentação desta ferramenta revelou-se bastante útil na criação do editor de modelação, mencionado na secção 3.2.2.

Comunidade

Epsilon EuGENia: Encontrámos um fórum oficial com poucos tópicos recentes, em que as perguntas e respostas são colocadas pela comunidade; uma *Wiki* oficial, sendo que a última atualização ocorreu em 2014.
Sirius: Um fórum oficial, com diversos tópicos recentes, em que as perguntas e respostas são colocadas pela comunidade; uma *Wiki* oficial, atualizada; artigos, vídeos e apresentações referentes à conferência anual de modelação gráfica *SiriusCon* [46]; suporte profissional para a indicação de erros a corrigir.
Comparando: A ferramenta *Sirius* apresenta uma comunidade mais ativa, que disponibiliza uma forma de reportar erros, ao contrário da comunidade da ferramenta *Epsilon EuGENia*.

Edições

Epsilon EuGENia: Versão 1.0, com licença pública.
Sirius: Duas versões com licença pública: *Eclipse Sirius* (licença básica) e *Obeo Designer Community*, que adiciona à licença básica um editor de modelo gráfico, mecanismo de comparação de modelos, integração com a equipa do *Eclipse*, e suporte profissional. Adicionalmente, oferece ainda a versão paga *Obeo Designer Team*, que adicionada à licença *Obeo Designer Community* um repositório para trabalho colaborativo.
Comparando: A ferramenta *Sirius* fornece um maior número de versões e funcionalidades, do que a ferramenta *Epsilon EuGENia*. (A licença *Sirius* utilizada para a criação do editor foi a *Obeo Designer Community*.)

Esforço

Epsilon EuGENia e **Sirius**: Comparativamente, o processo de criação do editor na ferramenta *Epsilon EuGENia* é sequencial e com menos número de passos. Contudo, em relação à adição de funcionalidades gráficas, esta ferramenta utiliza notação textual, o que torna o

processo mais complexo, enquanto que a ferramenta *Sirius* utiliza notação gráfica, o que requer um maior número de passos.

Sintaxe Abstrata

Epsilon EuGENia e **Sirius**: Ambas as ferramentas, *Epsilon EuGENia* e *Sirius*, utilizam modelos *Ecore*.

Sintaxe Concreta

Epsilon EuGENia: Utiliza notação textual em todo o processo de criação, edição e validação do editor. O editor é criado e editado a partir do *EMFatic Source*, sendo gerado posteriormente os ficheiros *GMF editor*. Para a construção das regras de validação é usada a *Epsilon Validation Language* (EVL).
Sirius: Utiliza notação gráfica no processo de criação e edição do editor, e notação textual na validação. O editor é criado e editado a partir do *EMFatic Source* e *Viewpoint*. Para a construção das regras de validação é usada a *Acceleo Query Language* (AQL).
Comparando: A aprendizagem da ferramenta *Sirius* tornou-se mais intuitiva, devido ao uso de notação gráfica. Quanto às linguagens de validação, não foi concluído o processo de comparação, pelo que não é possível tirar conclusões.

Transformações M2M

Epsilon EuGENia: Utiliza uma notação textual, com recurso à *Epsilon Transformation Language* (ETL).
Sirius: Utiliza uma notação textual, com recurso a diversas linguagens de transformação M2M (como por exemplo: Atlas Transformation Language (ATL), *Epsilon Transformation Language* (ETL), e Query/View/Transformation (QVT)).
Comparando: Apesar de não termos efetuado uma comparação prática, denotamos que ambas as ferramentas, *Epsilon EuGENia* e *Sirius*, utilizam uma notação textual. Ao contrário do *Epsilon EuGENia*, o *Sirius* não impõe uma linguagem de transformação de modelo para modelo própria.

Transformações M2T

Epsilon EuGENia: Utiliza a *Epsilon Transformation Language* (ETL), ou outras linguagens de transformação de modelos para texto.
Sirius: Não oferece linguagem própria de transformação de modelos para texto.
Comparando: Apesar de não termos efetuado uma comparação prática, denotamos que ambas as ferramentas utilizam uma notação textual. As linguagens de transformação de modelos para texto utilizadas podem ser iguais em ambas as ferramentas, dado não serem impostas linguagens próprias. As linguagens que podem ser utilizadas são, por exemplo:

ATL Transformation Language (ATL) [24], *Acceleo* [23], *Epsilon Generation Language* (EGL) [26], *Xpand* [25], *Xtend* [28].

A Tabela 3.2 resume os resultados encontrados, indicando qual ferramenta de modelação que considerámos ser melhor em cada critério de comparação. Como ambas as ferramentas apresentam resultados iguais quanto à análise dos critérios de sintaxe abstrata e transformações de modelo para modelo (M2M) e modelo para texto (M2T), nenhuma das ferramentas foi considerada superior em relação à outra nesses critérios.

Tabela 3.2: Comparação das ferramentas *Epsilon EuGENia* e *Sirius*

Critérios	***Epsilon EuGENia***	***Sirius***
Documentação Oficial	x	✓
Comunidade	x	✓
Edições	x	✓
Esforço	✓	x
Sintaxe Abstrata	✓	✓
Sintaxe Concerta	x	✓
Transformações M2M	✓	✓
Transformações M2T	✓	✓

De acordo com esta análise, fica claro que o *Sirius* tem vantagens sobre o *Epsilon EuGENia*.

3.3 Conclusões

Após a consolidação de conhecimentos sobre desenvolvimento orientado a modelos (MDD), fizemos uma análise de ferramentas. Selecionámos as ferramentas a comparar, escolhemos os critérios de comparação, e efetuámos uma recolha de dados. Posteriormente, fizemos a comparação das ferramentas *Epsilon EuGENia* e *Sirius* que indica que a ferramenta *Sirius* é melhor em relação aos critérios de documentação oficial, comunidade, edições, e sintaxe concreta. No entanto, consideramos que a ferramenta *Epsilon EuGENia* necessitou de um muito menor esforço para a criação e edição de um editor simples.

Apesar dos conhecimentos adquiridos com a linguagem *Epsilon Validation Language*, no contexto desta dissertação o tempo não permitiu igual investimento na linguagem *Acceleo Query Language*. Por este motivo, não foi possível comparar ambas as linguagens de validação das ferramentas *Epsilon EuGenia* e *Sirius*. Da mesma forma, os conhecimentos adquiridos nas linguagens de transformação que se podem utilizar em ambas as ferramentas não nos permite fazer uma comparação em consciência. No entanto, os resultados obtidos até agora indicam que a ferramenta *Sirius* é a mais adequada para o desenvolvimento do editor. Desta forma, optámos por escolher a ferramenta *Sirius* para prosseguir com o desenvolvimento do trabalho.

CAPÍTULO 4

O MÉTODO DVD: CONCEITOS, TÉCNICAS E MODELOS

Este capítulo discute os conceitos, técnicas e modelos que constituem o método DVD e que serão suportados pela ferramenta DVDTool, ferramenta de apoio ao método DVD. De modo a facilitar o alinhamento entre a área de negócios e a área de tecnologias de informação, propomos a transformação de modelos DVD para modelos KAOS (de requisitos) ou SoaML (de serviços), recorrendo a técnicas MDD. Tendo o modelo DVD, modelo origem, é necessário analisar os modelos alvo e definir os mapeamentos entre os conceitos e relações entre o modelo DVD e os modelos pretendidos, de forma a ser possível gerar um modelo de requisitos orientado a objetivos (Secção 4.2) e um modelo de arquitetura de serviços (Secção 4.2.3).

4.1 O Método DVD

Um modelo de negócios é inspirado na ciência empresarial, na engenharia de requisitos e na modelação de conceitos para modelar ideias empresariais [56]. O conceito primordial de um modelo de negócio é o de *valor*, que explica a criação, a adição e a permuta de valores entre as partes interessadas [32]. Estes conceitos estão descritos com maior detalhe na Secção 2.1. De acordo com a análise dos resultados do estudo sistemático (secções 2.2 a 2.4), o método DVD foi escolhido pois, além da existência de experimentos que avaliam a sua eficácia (facilidade de utilização e utilidade) [57, 59, 60], o método não possui uma ferramenta para dar suporte a modelação de valor.

O método DVD oferece um processo e uma linguagem para identificar e descrever valor e trocas de valor num modelo próprio, o modelo DVD. Assim, a linguagem do DVD tem por objetivo analisar trocas de valor empresariais, permitindo que as partes interessadas

compartilhem os seus pontos de vista sobre essas trocas de valor, num modelo tipo mind-map[1] semi-estruturado. Deste modo, o DVD herda alguns dos benefícios do mind-map, como a organização de ideias e conceitos, bem como a associação entre elementos em ramos e agrupamento de ideias [12]. (Recordamos que os conceitos principais do modelo DVD estão descritos na secção 2.3.2.) Na Figura 4.1 apresentamos um modelo DVD ilustrando os vários conceitos e suas relações.

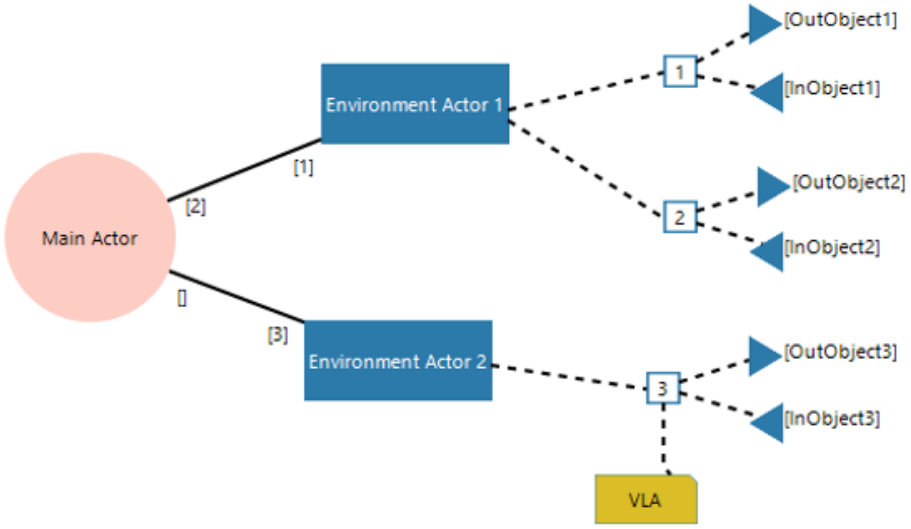


Figura 4.1: Conceitos do modelo DVD

4.1.1 Metamodelo da linguagem DVD

A Figura 4.2 descreve o metamodelo para a linguagem DVD, representando os nós (*Node*) e as relações (*Relation*) necessárias para a construção de um modelo DVD. (Os nós representados a cinzento correspondem a metaclasses abstratas.) Os nós deste metamodelo correspondem aos conceitos principais da linguagem DVD (apresentados na secção 2.1):

- ***Actor:*** cada ator tem um nome. Existem dois tipos de atores: ator principal (***MainActor***) e ator secundário (***EnvironmentActor***). Cada ator tem um atributo único que o identifica, *idMainActor* e *idEnvironmentActor*, respetivamente.
- ***ValueExchange:*** cada troca de valor tem um *id* e uma descrição.
- ***Port:*** existem dois tipos de porta de valor: O objeto de valor de entrada (***InValueObject*** e objeto de valor de saída ***OutValueObject***). Cada porta de valor tem um objeto de valor, uma descrição e um identificador, o *idObject*.
- **Value Level Agreement:** cada nível de valor acordado tem uma descrição e um identificador, o *idVLA*.

[1]Um *mind-map*, ou mapa mental, é um tipo de mapa cognitivo usado para visualizar, classificar e organizar conceitos e gerar novas ideias de maneira direta e intuitiva [12].

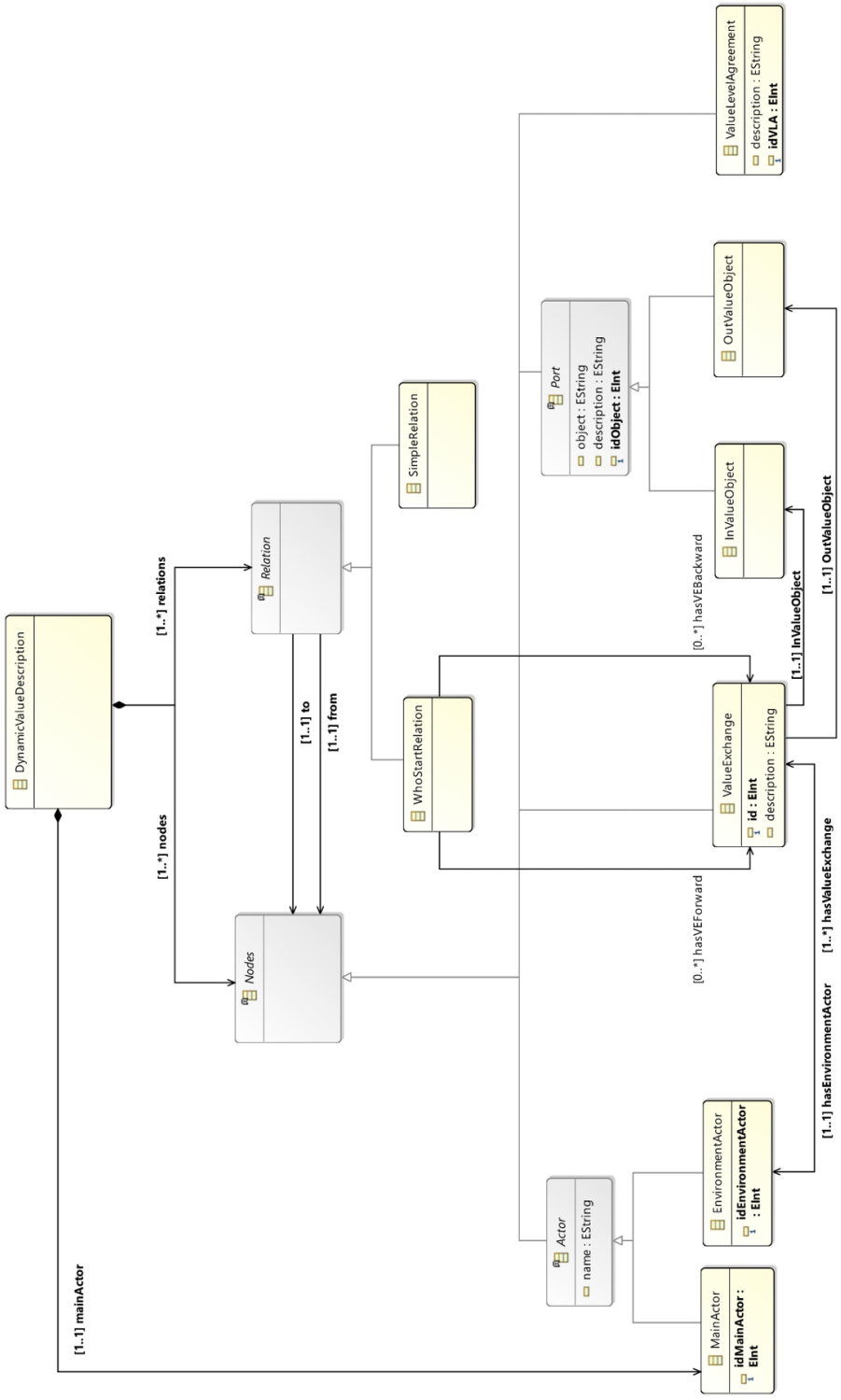


Figura 4.2: Metamodelo da linguagem DVD

As relações apresentadas no metamodelo são de dois tipos: ***WhoStartsRelation*** e ***SimpleRelation***. Cada uma destas relações tem apenas um nó origem (***[1..1] from***) e um nó alvo (***[1..1] to***). A relação entre o ator principal e o ator secundário é feita através da relação *WhoStartsRelation*, tendo como nó origem um *MainActor* e como nó alvo um *EnvironmentActor*. Nesta relação é indicado que trocas de valor (*ValueExchange*) são iniciadas pelo ator principal, através da relação **[0..*]** ***hasVEBackward***, e que trocas de valor são iniciadas pelo ator secundário, através da relação **[0..*]** ***hasVEForward***. Ambas as relações, *hasVEBackward* e *hasVEForward*, obtêm o *id* das trocas de valor (nó *ValueExchange*), cada ator pode não iniciar nenhuma ou todas as trocas de valor com que está relacionado.

Cada ator secundário pode ter mais do que uma troca de valor, mas cada troca de valor apenas pode pertencer a um ator secundário, esta relação é representada pela relação bidirecional **[1..*]** ***hasValueExchange*** e **[1..1]** ***hasEnvironmentActor***, respetivamente. Por sua vez, cada troca de valor tem um objeto de valor de entrada, representado pela relação **[1..1]** ***InvalueObject***, e um objeto de valor de saída, representado pela relação **[1..1]** ***OutValueObject***. Cada troca de valor pode ter relacionado um ou mais níveis de valor acordado, esta ligação é feita através da relação *SimpleRelation*, onde o nó origem é um *ValueExchange* e o nó alvo é um *ValueLevelAgreement*.

4.2 Modelação de Requisitos Orientados a Objetivos

A engenharia de requisitos orientada por objetivos usa o conceito de objetivo (ou *goal*) preocupa-se para elicitar, elaborar, estruturar, especificar, analisar, negociar, documentar e modificar requisitos de software [63]. Um modelo orientado a objetivos utiliza o conceito *objetivo*, (ou *goal*), como meio para raciocinar (*reasoning*) sobre os porquês (*why*) sobre um sistema idealizado [63]. Um engenheiro de requisitos, ao utilizar este tipo de modelação, centra-se numa visão estratégica do sistema, abstraindo-se de detalhe não relevantes para a especificação dos requisitos iniciais, como os detalhes operacionais [30].

Existem várias abordagens orientadas a objetivos, como por exemplo, KAOS [16], EKD [41], BMM [47], i * / Tropos [70], GSN [36], NFR [14], GBRAM [6], Techne [10] e GRL [4]. De entre todas, o KAOS tem sido das linguagens mais citadas na literatura [66] e, quando comparada com as restantes abordagens, é aquela que oferece um maior número de conceitos [56].

4.2.1 KAOS4Services: Conceitos

KAOS4Services é uma parte do método DVD que tem como objetivo final definir transformações entre um modelo (adaptado) KAOS e um modelo de serviços. Este modelo adaptado do KAOS utiliza um subconjunto dos conceitos da linguagem KAOS e é estendido com um conjunto de novos conceitos de valor para facilitar o alinhamento entre as áreas de negócio e de tecnologia de informação. A Figura 4.3 apresenta os vários conceitos do modelo KAOS4Services.

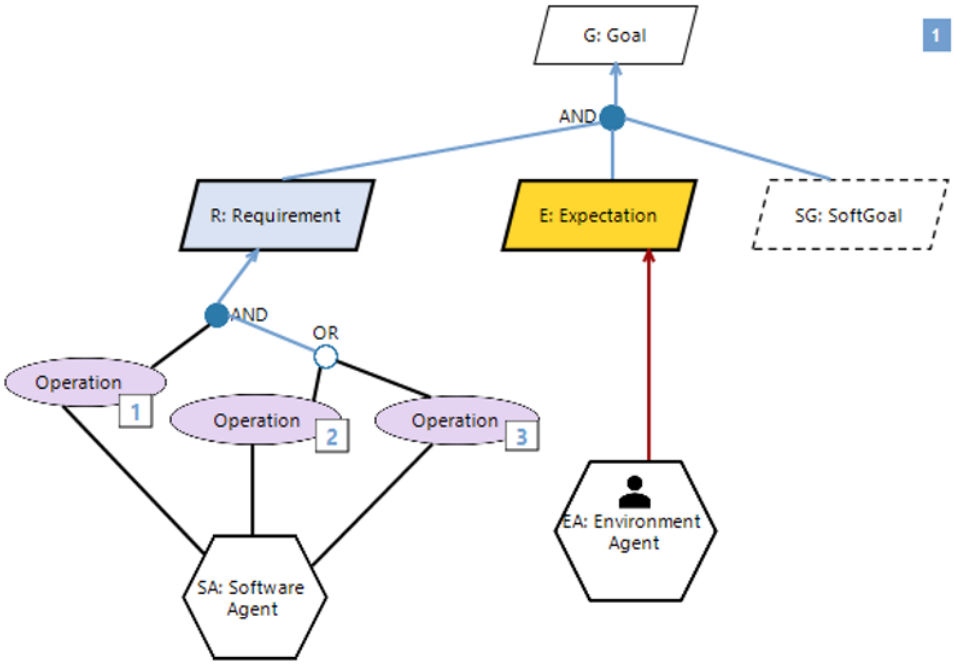


Figura 4.3: Conceitos do modelo KAOS4Services

Os principais conceitos deste modelo são Objetivo (*Goal*), *softgoal*, agentes, expectativas, requisitos, refinamento, operação, ordem e cenário [1]. **Objetivo** é uma meta que o sistema deve atingir (por exemplo, o sistema deve calcular a folha de pagamento da empresa). A satisfação desse objetivo requer a cooperação de alguns agentes. Cada objetivo é tipicamente refinado por uma coleção de *softgoals*, requisitos e expectativas. **Softgoal** é um objetivo para o qual não há critérios para a sua satisfação. Está relacionado com a especificação de atributos e requisitos de qualidade. Por exemplo, no requisito *"o sistema deve ser fácil de compreender"* não há critérios que definam o que é ser fácil de compreender. O conceito de **Agente** pode ser definido como as partes interessadas que interagem com o sistema. Agentes podem ser seres humanos, agentes de ambiente (*environment agents*), ou componentes automatizados, agentes de software (*software agents*) responsáveis pela realização de requisitos e expectativas.

4.2.2 KAOS4Services: Metamodelo

A Figura 4.4 ilustra o metamodelo da linguagem KAOS4Services, onde os nós (*Node*) representam os conceitos principais.

Os elementos do metamodelo são os seguintes:

- ***Agent:*** cada agente tem um nome. Existem dois tipos de agente: ***SoftwareAgent*** e ***EnvironmentAgent***. Cada agente tem um identificador próprio, *DVDelementID*.
- ***Goal:*** o metamodelo apresenta uma abstração que engloba três tipos de objetivos: ***BehavioralGoal*** (um objetivo comportamental que o sistema deve atingir), ***SoftGoal***

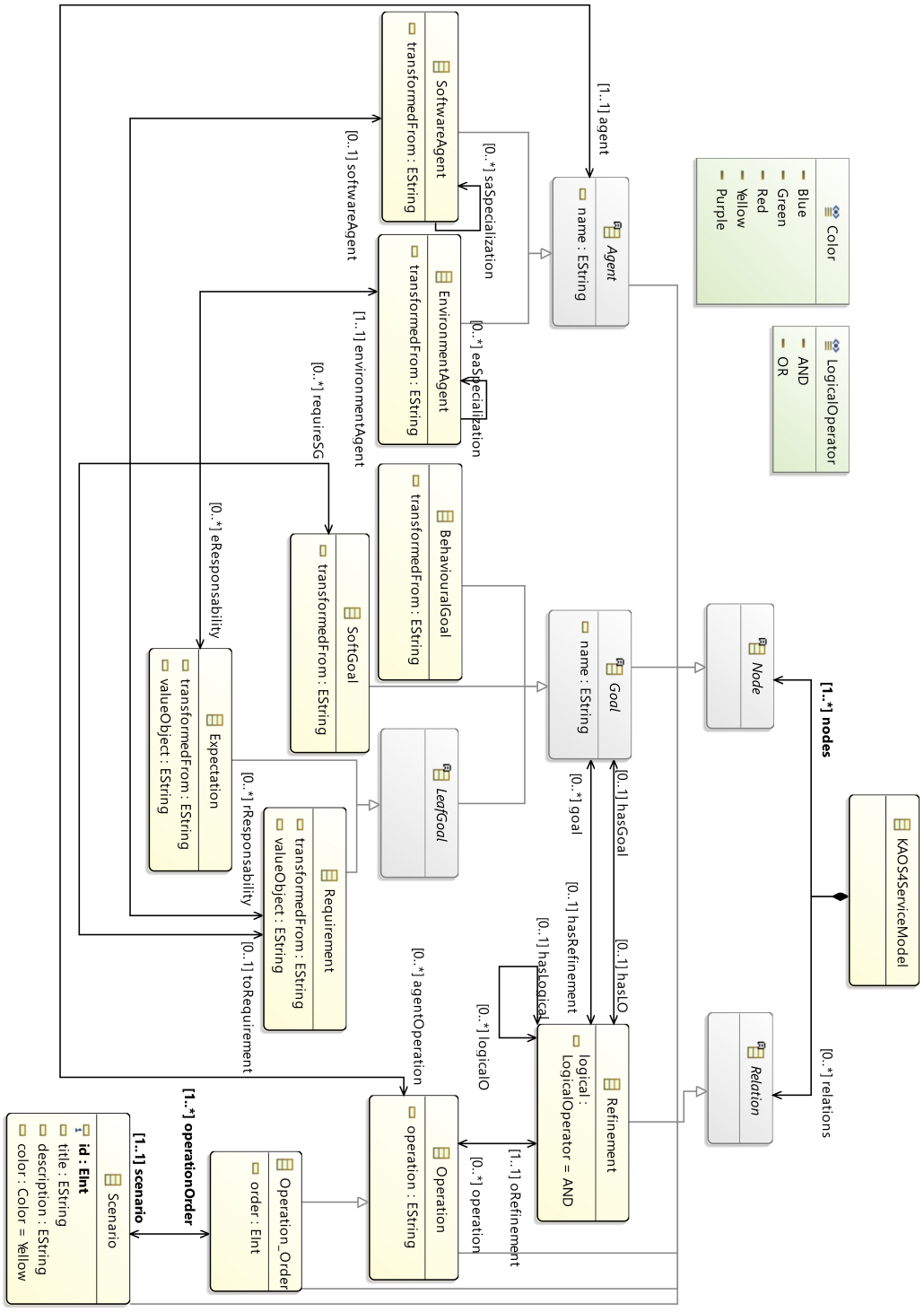


Figura 4.4: Metamodelo da linguagem KAOS4Services

(um objetivo que não há critérios explícitos para sua satisfação) e ***LeafGoal*** (um objetivo criado apenas para generalizar os expectations e requirements). Cada *LeafGoal* tem um *valueObject*. Cada *Goal* tem um nome e um identificador (*DVDelementID*).

- ***Operation:*** cada operação tem um atributo com uma descrição desta.
- ***Operation Order:*** cada operação tem uma ordem. Este nó herda do nó *Operation*, ficando assim atribuído uma ordem a cada operação.
- ***Scenario:*** cada cenário tem um *ID*, um título, uma descrição e uma cor (o enumerado *Color: Blue, Green, Red, Yellow e Purple*).

A relação (*Relation*) representa um ***Refinement***. Este elemento é apresentado como uma relação, visto ter como objetivo conectar dois ou mais elementos como forma de um operador lógico (é um enumerado com os valores AND e OR).

Cada objetivo que o sistema deve atingir (*BehaviouralGoal*) é refinado (através do elemento *Refinement*) para um conjunto de requisitos, expectativas e *softgoals*. Estas relações são representadas pelas relações **[0..1]** ***hasGoal*** e **[0..1]** ***hasRefinement*** (onde um *Refinement* conecta com um *BehaviouralGoal*, e *softgoals*, requisitos e expectativas conectam com um *refinement*). Um requisito pode especificar-se num *softgoal*, sendo que neste caso, um *softgoal* apenas poderá pertencer a um requisito. Esta relação é representada pela relação bidirecional **[0..*]** ***requireSG*** e **[0..1]** ***toRequirement***, respetivamente.

Cada agente pode relacionar-se com requisitos (no caso de ser um agente de software) ou expectativas (no caso de ser um agente de ambiente), ou com operações. Um agente de software pode ter que alcançar vários requisitos, e cada requisito apenas pertence a um agente de software, esta relação é representada pela relação bidirecional **[0..*]** ***RResponsability*** e **[1..1]** ***SoftwareAgent***, respetivamente. Por sua vez, um agente de ambiente pode ter que alcançar várias expectativas, e cada expectativa apenas pertence a um agente de ambiente, esta relação é representada pela relação bidirecional **[0..*]** ***EResponsability*** e **[1..1]** ***EnvironmentAgent***, respetivamente. Por outro lado, qualquer um dos agentes pode ter que cumprir várias operações, mas cada operação apenas pertence a um agente, esta relação é representada pela relação bidirecional **[0..*]** ***Operation*** e **[1..1]** ***Agent***, respetivamente. Cada agente pode também especializar-se em outros agentes do mesmo tipo, isto é, um agente de software pode especializar-se em outros agentes de software, e um agente de ambiente pode especializar-se em outros agentes de ambiente. Estas relações são representadas pela relação **[0..*]** ***Specialization***.

Um conjunto de operações pode relacionar-se por operadores lógicos (através do elemento *Refinement*), essas relações são representadas pela relação **[0..*]** ***ORefinement***, e pela relação **[0..1]** ***hasLogical***, que permite conectar um *refinement* com outro *refinement*. Cada operação tem associado uma ordem, e cada ordem pertence a um só cenário, sendo que cada cenário pode conter um conjunto de ordens de operação. Esta relação é apresentada através da relação bidirecional **[1..1]** ***Scenario*** e **[0..*]** ***Operation***, respetivamente.

4.2.3 Transformação MDD: de Valor a Objetivo

O alinhamento entre modelos de valor e modelos orientados a objetivos é feito através de técnicas de desenvolvimento orientado a modelos (MDD). No nosso caso, este alinhamento é feito entre o modelo DVD e o modelo KAOS4Services. A Figura 4.5 representa este processo, constituído por seis atividades.

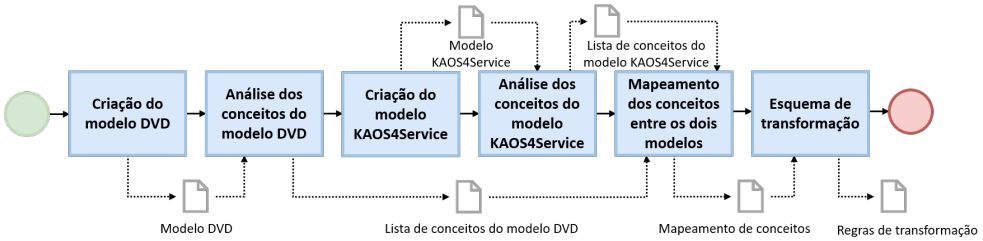


Figura 4.5: Gerando KAOS4Services a partir de um modelo DVD [56]

Criação do modelo DVD Nas transformações de modelo para modelo (M2M), o objetivo é transformar um modelo origem num modelo alvo. Ao criar um modelo DVD obtemos o nosso modelo origem.

Análise dos conceitos do modelo DVD Para ser possível um posterior mapeamento, que permitirá obter um modelo KAOS4Services (modelo alvo), é necessário perceber os conceitos do modelo origem, neste caso do modelo DVD. Nesta atividade, são analisados os conceitos do modelo obtido na atividade anterior. Os conceitos principais do modelo DVD foram apresentados na Secção 2.3.2.

Criação do modelo KAOS4Services Tendo um modelo origem, é necessário um modelo alvo para que a transformação seja possível. Ao criar um modelo KAOS4Services obtemos o nosso modelo alvo.

Análise dos conceitos do modelo KAOS4Services De modo a fazer o alinhamento entre o modelo DVD e o modelo KAOS4Services, é necessário perceber os conceitos de ambos os modelos. Nesta atividade são analisados os conceitos do modelo obtido na atividade anterior. Os conceitos principais do modelo KAOS4Services foram apresentados na secção 4.2.1.

Mapeamento dos conceitos entre os dois modelos Tendo o modelo origem, DVD, e o modelo alvo, KAOS4Services, é necessário fazer um mapeamento de conceitos entre os dois modelos. Esse mapeamento é feito através de uma lista de conceitos obtidos pela análise de ambos os modelos. Desta atividade resulta com conjunto de heurísticas de mapeamento entre o modelo DVD e o modelo KAOS4Services.

Estas heurísticas são as seguintes:

- Heurística 1 (H1): Um ***Value Exchange*** (troca de valor) corresponde a um ***Goal*** (objetivo que o sistema deve atingir). Ambos representam um objetivo a ser alcançado.

- Heurística 2 (H2): Um ***Out Value Object*** (objeto de valor de saída) corresponde a um ***Expectation*** (expectativa). Ambos representam o que um ator deve fornecer para receber algo em troca.

- Heurística 3 (H3): Um ***In Value Object*** (objeto de valor de entrada) corresponde a um ***Requirement*** (requisito). Ambos representam o que um ator deve fornecer para receber algo em troca.

- Heuristica 4 (H4): Um ***Actor*** (ator) corresponde a um ***Agent*** (agente). Mais especificamente, um *environment actor* (ator secundário) corresponde a um *environment agente* (agente de ambiente), e um *main actor* (ator principal) corresponde a um *software agente* (agente de software).

- Heuristica 5 (H5): Um ***Value Level Agreement*** (nível de valor acordado) corresponde a um ***Softgoal*** (objetivo para o qual não há critérios para a sua satisfação). Ambos os conceitos estão relacionados com a especificação de atributos e requisitos de qualidade.

Esquema de transformação Através do mapeamento de conceitos, e respetivas heurísticas, conseguimos criar um esquema de transformação entre os modelos. Nesse esquema é representado qual o conceito do modelo KAOS4Services corresponde a um dado conceito do modelo DVD. A Figura 4.6 ilustra este esquema de transformação usando as heurísticas acima descritas.

4.3 Modelação de uma Arquitetura de Serviços de Referência

A Arquitetura Orientada a Serviços (*Service-Oriented Architecture* (SOA), em inglês) é uma evolução do desenvolvimento baseado em componentes que utiliza serviços como o principal componente de construção [21, 62]. SOA contem um *fornecedor* de serviços que disponibiliza um *serviço* de modo a ser consumido por um *consumidor* [8]. Serviço é uma unidade de lógica de solução [7] que lida com recursos com valor para ambos os participantes (produtores e consumidores); esses serviços são realizados no contexto de trocas económicas que refletem a criação recíproca de valor para os participantes envolvidos. Estas interações entre os participantes definem as trocas de valor entre as entidades. Deste modo, um ou mais serviços de negócio operacionalizam uma troca de valor.

Esta secção tem o objetivo de demonstrar como uma arquitetura de serviços é representada usando a linguagem SoaML. Desta forma, apresentamos os conceitos de um modelo

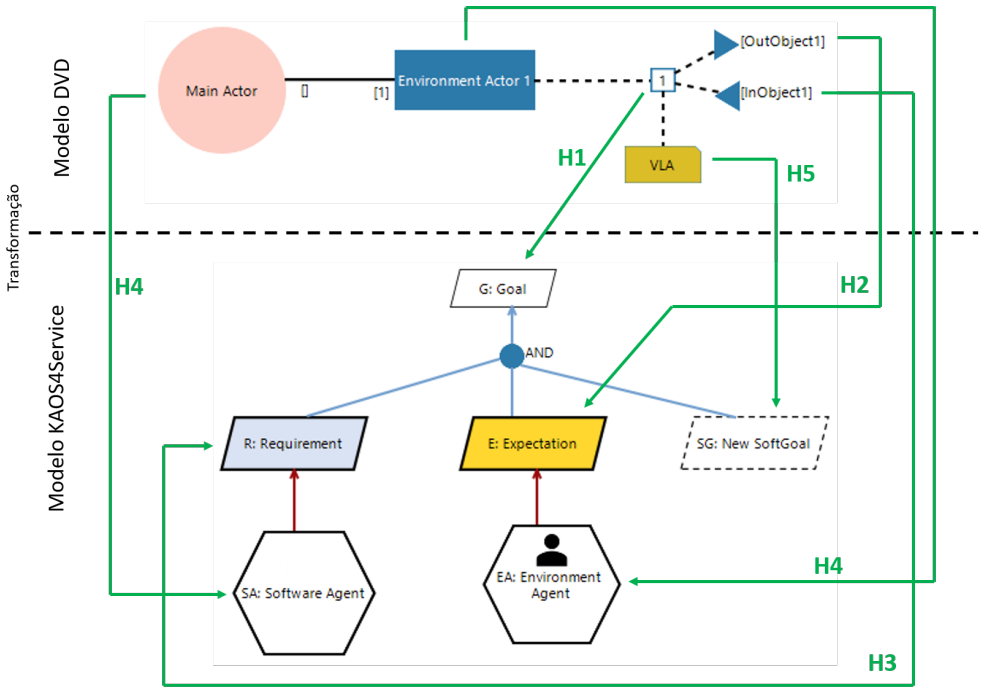


Figura 4.6: Mapeamentos entre elementos do modelo DVD e do modelo KAOS4Services.

SoaML, seu metamodelo e como podemos fazer a geração de um modelo SoaML a partir de um modelo DVD.

4.3.1 Arquitetura de Serviços: SoaML

SoaML (*Service Oriented Architecture Modeling Language*) [48] é o modelo UML padrão para especificar soluções SOA, que abrange as perspetivas de negócios e TI (Tecnologia da Informação). Este modelo suporta diferentes abordagens para a especificação de serviços, mas apenas a abordagem baseada em contratos de serviço, descrevendo as interações entre os participantes, se encaixa na perspetiva de negócios [13]. Os principais conceitos do modelo SoaML são:

- **Service Architecture:** Arquitetura de serviços que engloba todos os participantes e serviços envolvidos.
- **Participant:** Participantes são sujeitos que fornecem ou consomem serviços de negócio (fornecedor ou consumidor, respetivamente).
- **Service Contract:** Contratos de serviços são meta informações sobre o serviço. No modelo SoaML, Service Contract é uma representação de serviços tanto da perspetiva de tecnologia de informação, como da perspetiva de negócio.
- **QoSValue:** Define um conjunto mínimo de regras de negócios acordadas entre os participantes, para um determinado serviço.

A Figura 4.7 ilustra como todos estes conceitos se relacionam.

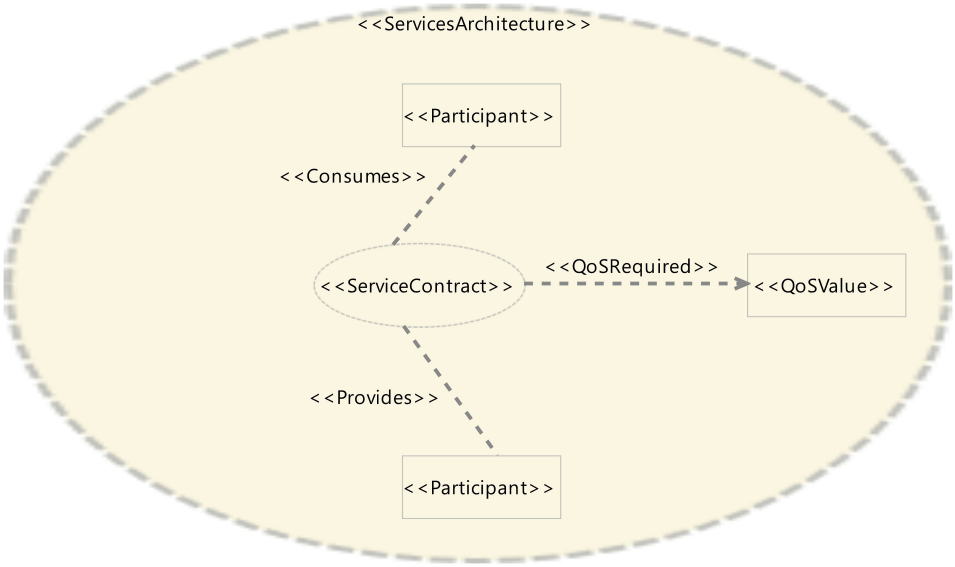


Figura 4.7: Conceitos e relações do modelo SoaML.

4.3.2 SoaML: Metamodelo

A Figura 4.9 descreve os vários conceitos do SoaML, onde os nós (*Node*) representam os conceitos principais do modelo, e também as suas relações.

Os elementos do metamodelo são:

- **Service Architecture:** Uma arquitetura de serviços é identificada por um nome e uma descrição. É composta por participantes (*paticipant*), Serviços de negócio (*service contract*) e QoSvalues.
- **Participant:** Cada elemento participante tem associado um nome.
- **Service Contract:** Cada serviço de negócio tem associado um nome.
- **QoSValue:** Cada elemento QoSValue tem associado um nome.

Cada modelo SoaML é composto por um único elemento *Service Architecture*. Esta composição está representado no metamodelo através da relação de composição **[1..1]** ***servicesArchitecture***. Por sua vez o elemento *ServiceArchitecture* é composto por participantes, serviços de negócio e *QoSValues*. Têm que existir pelo menos dois participantes numa arquitetura de serviços. A ligação entre estes dois elementos é representada pela relação de composição **[2..*]** ***participant***. Existe no mínimo um serviço de negócio numa arquitetura de serviços. Esta composição está representada no metamodelo na relação de composição **[1..*]** ***serviceContract***, que une estes dois elementos.

Cada serviço de negócio tem um participante que é o seu fornecedor, no entanto um participante pode não ser provedor de nenhum serviço ou, por outro lado, pode ser provedor de vários serviços. Esta ligação entre os elementos participante e serviço

de negócios é representada pela relação bidirecional **[1..1]** ***provides*** e **[0..*]** ***hasProvides***, respetivamente. De forma semelhante, cada serviço de negócio tem um participante que é o seu consumidor, porém um participante pode não ser consumidor ou ser consumidor de mais do que um serviço de negócios. Sendo assim, o relacionamento entre os elementos participante e serviços de negócio, é feito por meio da relação bidirecional **[1..1]** ***consumes*** e **[0..*]** ***hasConsumes***, respetivamente.

O elemento arquitetura de serviços (*ServiceArchitecture*) pode ter, não sendo obrigatório, na sua composição elementos *QoSValue*; deste modo, esta composição é representada no metamodelo através da relação de composição **[0..*]** ***qoSValue***. Cada serviço de negócio pode ter ou não associado um *QoSValue*. Todavia, cada *QoSValue* pertence a um único serviço de negócio. A união entre os elementos *QoSValue* e *ServiceContract* é representada através da relação bidirecional **[0..*]** ***qoSValue*** e **[1..1]** ***hasQoSValue***, respetivamente.

4.3.3 Transformação MDD: de Valor a Serviço

De forma semelhante ao alinhamento entre modelos de valor e modelos orientados a objetivos, o alinhamento entre modelos de valor e arquitetura orientada a serviços é feito através de técnicas MDD. Neste caso, o alinhamento é feito entre o modelo de valor DVD e o modelo SoaML. Este processo é composto por seis atividades, que se encontram representadas na Figura 4.8.

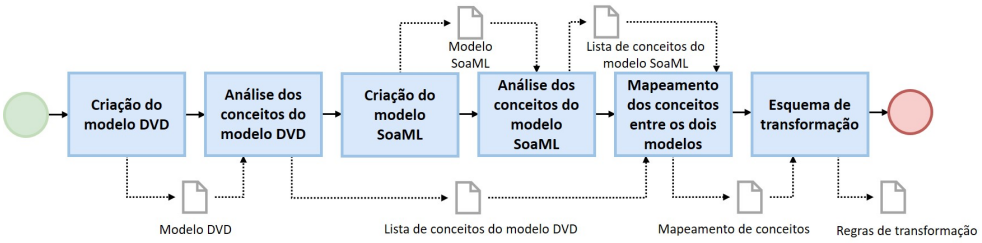


Figura 4.8: Gerando SoaML a partir de um modelo DVD [56]

Criação do modelo DVD Nesta transformação de modelo para modelo (M2M) pretende-se transformar um modelo de valor (modelo origem) num modelo de arquitetura de serviços (modelo alvo). Ao criar um modelo DVD obtemos o nosso modelo origem.

Análise dos conceitos do modelo DVD A compreensão dos conceitos do modelo de origem, permitirá o mapeamento de conceitos entre os dois modelos. Nesta atividade são analisados os conceitos do modelo obtido na atividade anterior. (Os conceitos principais do modelo DVD foram apresentados na Secção 2.3.2.)

Criação do modelo SoaML Tendo o modelo de valor DVD como modelo origem, é necessário um modelo alvo para que a transformação seja possível. Ao criar um modelo SoaML obtemos o nosso modelo alvo.

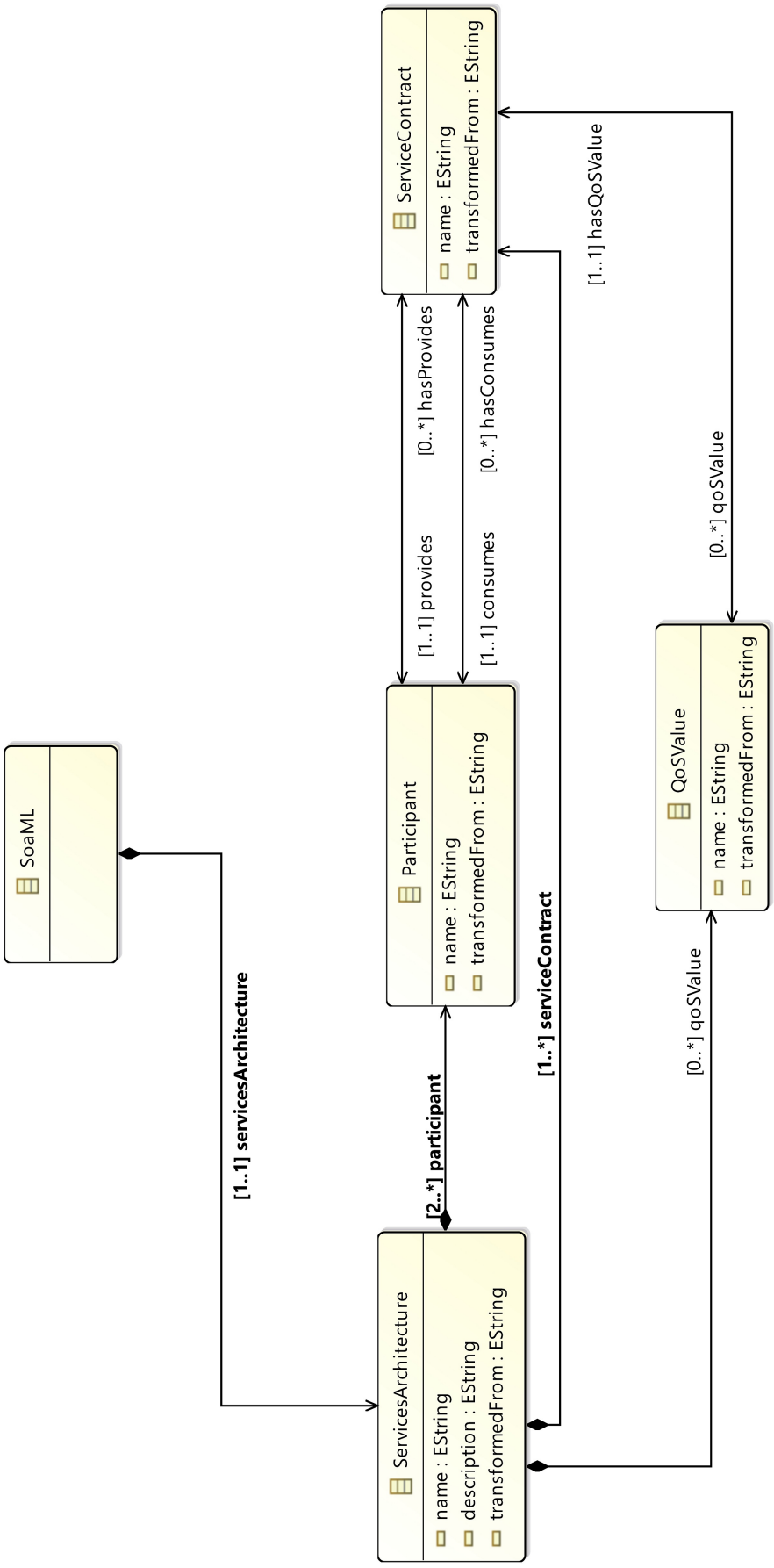


Figura 4.9: Metamodelo da linguagem SoaML

Análise dos conceitos do modelo SoaML Para que o alinhamento entre o modelo DVD e o modelo SoaML seja fazível, é necessário perceber os conceitos de ambos os modelos. Nesta atividade analisamos os conceitos do modelo obtido na atividade anterior. Os conceitos principais do modelo SoaML foram apresentados na secção 4.3.1.

Mapeamento dos conceitos entre os dois modelos É necessário efetuar um mapeamento de conceitos entre o modelo origem DVD e o modelo alvo SoaML. Esse mapeamento é feito através de uma lista de heurísticas pela análise de ambos os modelos. Estas heurísticas são:

- Heurística 1 (H1): Um *Main Actor* (ator principal) corresponde a um *ServiceArchitecture* (arquitetura de serviços). Dado que ambos são o foco do modelo, o ator principal do DVD fornece o nome à arquitetura de serviços em SoaML.

- Heurística 2 (H2): Um *Actor* (ator) corresponde a um *Participant* (participante). Mais especificamente, tanto um *main actor* (ator principal) como um *environment actor* (ator secundário) correspondem a um *participant* (participante). Ambos representam as entidade envolvidas.

- Heurística 3 (H3): Uma *Value Exchange* (troca de valor) corresponde a um *Service Contract* (serviço de negócio). Ambos representam operacionalizações entre as entidades envolvidas.

- Heurística 4 (H4): A relação *WhoStartsRelation* (relação que indica quem inicia uma determinada troca de valor) identifica quem é o fornecedor e quem é o consumidor de um serviço. Sendo que o ator que inicia a troca de valor corresponde ao participante consumidor; de forma complementar, o outro ator envolvido na troca de valor é o fornecedor do serviço.

- Heurística 5 (H5): Um *Value Level Agreement* (nível de valor acordado) corresponde a um *QoSValue* (conjunto mínimo de regras de negócios). Ambos os conceitos estão relacionados com a especificação de atributos e requisitos de qualidade requeridos.

Desta atividade resulta um conjunto de heurísticas de mapeamentos entre o modelo DVD e o modelo SoaML.

Esquema de transformação Através do mapeamento de conceitos, e respetivas heurísticas, criamos um esquema de transformação entre estes dois tipos de modelos. Nesse esquema é representado qual o conceito do modelo SoaML que corresponde a um dado conceito do modelo DVD. A Figura 4.10 apresenta o esquema de transformação usando as heurísticas acima descritas.

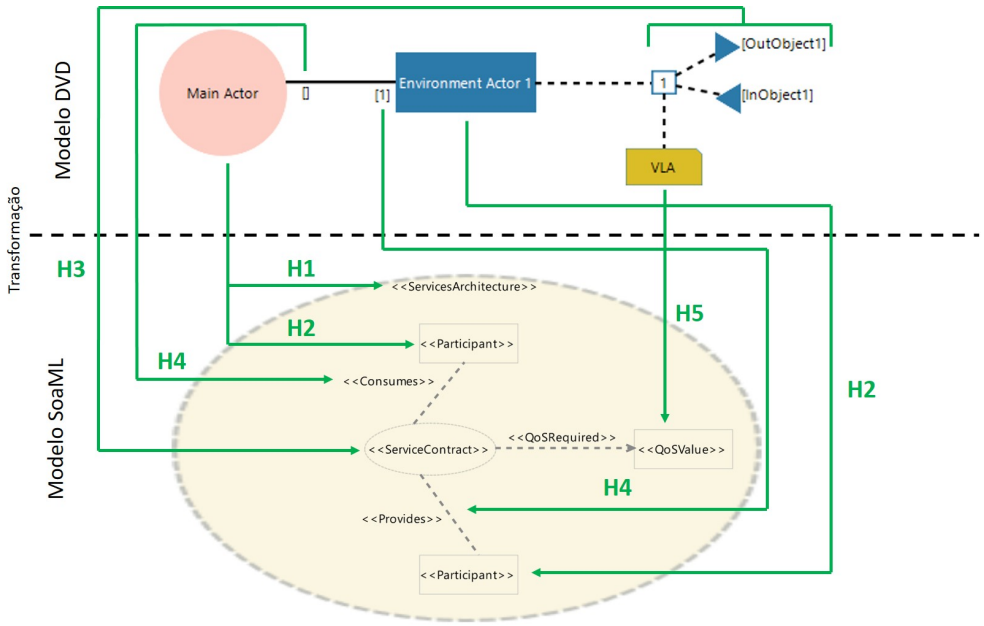


Figura 4.10: Mapeamentos entre elementos do modelo DVD e elementos SoaML.

4.4 Conclusões

Este capítulo analisa a forma como os elementos do modelo DVD se relacionam, e, para obter o alinhamento entre as áreas de negócio e os sistemas de informação, o modelo DVD é usado como modelo origem em transformações M2M para uma extensão do KAOS e SoaML.

Usando técnicas de MDE, geramos um modelo de requisitos orientado a objetivos e um modelo de arquitetura de serviços. Para tal, analisámos os conceitos de requisitos orientados a objetivos. Das diversas abordagens que existem, elegemos o modelo KAOS4Services por ser uma adaptação do modelo de requisitos KAOS (modelo que apresenta um maior número de conceitos no âmbito pretendido). O modelo KAOS4Services contém um subconjunto de conceitos da linguagem KAOS, e um conjunto de conceitos de valor de modo a facilitar a identificação de serviços de software. Analisámos os conceitos deste modelo, e o modo como estes se relacionam. Estudámos o processo de transformação, partindo do modelo de valor DVD até obter um modelo KAOS4Services.

De forma semelhante, analisámos os conceitos de arquitetura de serviços, sendo que o modelo padrão para esta especificação é o modelo SoaML. Após consolidar os conceitos do modelo SoaML, debruçámo-nos sobre o mapeamento de conceitos entre o modelo de origem DVD e o modelo alvo SoaML.

Agora que consolidámos os conceitos e técnicas do método DVD, desde a criação do modelo DVD à obtenção dos modelos KAOS4Services e SoaML, temos os conhecimentos necessários para proceder ao desenvolvimento da ferramenta.

CAPÍTULO 5

DVDTOOL: UMA FERRAMENTA PARA O MÉTODO DVD

Este capítulo apresenta a ferramenta DVDTool, que desenvolvemos para apoiar o desenvolvimento orientado a valor usando o método DVD. Como parte da ferramenta, foram desenvolvidos três editores de modelos, um para o modelo DVD, outro para o modelo KAOS4Services, e outro para o modelo SoaML. Para facilitar o alinhamento entre a área de negócios e a área de sistemas de informação, a DVDTool oferece transformações entre modelos usando técnicas MDD. Através da ferramenta conseguimos obter um modelo KAOS4Services ou um modelo SoaML tendo como modelo fonte um modelo DVD. Esta ferramenta foi desenvolvida para dispositivos com o sistema operativo *Windows*, e está disponível no repositório *Bitbucket*[1].

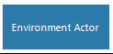
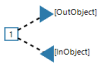
5.1 Editor para o Modelo DVD

A construção do modelo DVD é definida a partir de um metamodelo (Figura 4.2) que representa a linguagem abstrata do nosso domínio específico. A notação com a qual o utilizador interage com a linguagem, isto é, a linguagem concreta, está identificada na Tabela 5.1. Esta tabela apresenta, para cada elemento e relação, uma representação visual, o nome na paleta do editor, e o conceito associado.

No editor de um modelo de valor DVD, os elementos *MainActor* e *EnviromentActor* apresentam como propriedades um nome (*name*) e um identificador (*idMainActor* e *idEnvironmentActor*, respetivamente). Para além destas propriedades, também faz parte do elemento *EnviromentActor* uma propriedade que indica a que trocas de valor é que este ator está associado (*hasValueExchange*). O elemento *ValueExchange* contém uma troca de valor e dois objetos de valor, um que corresponde a um fluxo de entrada e outro a um

[1] https://bitbucket.org/soraiagf/dvd/

Tabela 5.1: Linguagem concreta do DVD

Ícone	Nome na paleta do editor	Conceitos
	Main Actor	Ator Principal
	Environment Actor	Ator Secundário
	Value Exchange	Troca de valor, portas de valor e objectos de valor
	VLA	Valor mínimo acordado
	StartRelation	Indica o ator que inicia a troca de valor
	ExchangeRelation	Indica as trocas de valor de um ator
	VLARelation	Indica os valores acordados para uma troca de valor

fluxo de saída (Figura 5.1, lado direito). Desta forma contribuímos para o fator usabilidade da DVDTool, dado que com um único elemento da paleta, o utilizador cria em simultâneo três elementos e as relações entre esses elementos. Caso contrário, para o utilizador representar uma troca de valor no modelo, seria necessário um elemento na paleta para cada elemento que pretendesse representar (um para a troca de valor, outro para o objeto de entrada e outro para o objeto de saída). Para além dos elementos mencionados, seriam também necessárias mais duas relações, para fazer a ligação entre a troca de valor e o respetivo objeto de valor. Sendo assim, ao fornecermos um único elemento na paleta para a representação deste conjunto de elementos, fazemos com que a construção de um modelo DVD se torne mais ágil.

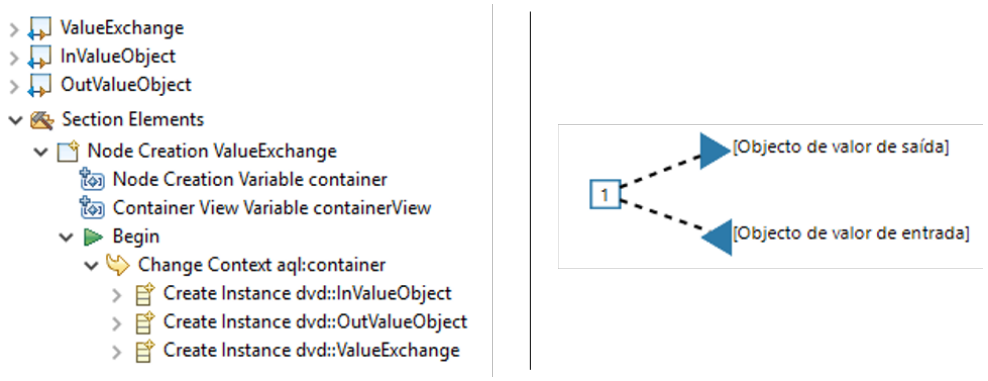


Figura 5.1: Construção do elemento *ValueExchange* e a sua representação

A troca de valor contém no seu conjunto de propriedades ver (Figura 5.2) um identificador (*id*), a indicação do ator secundário a que está associado (*hasEnvironmentActor*), o objeto de valor de saída (*outValueObject*), e o objeto de valor de entrada (*inValueObject*).

Ambos os objetos de valor (*InValueObject* e *OutValueObject*) têm como propriedades um objeto de valor (*object*), uma descrição (*description*) e um identificador (*idObject*). O elemento VLA tem como propriedades uma descrição (*description*) e um identificador (*idVLA*).

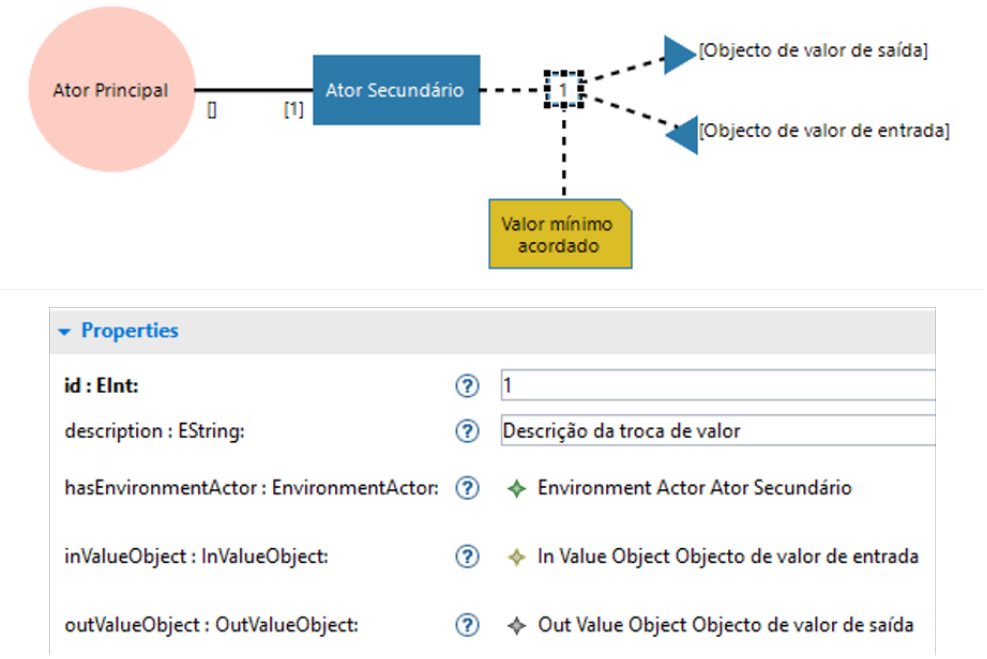


Figura 5.2: Propriedades do elemento *ValueExchange*

A relação *StartRelation*, que une o ator principal ao ator secundário, também tem como função identificar quem inicia cada uma das trocas de valor que ocorrem entre os dois atores envolvidos. Sendo assim, o seu conjunto de propriedades é constituído pela indicação do ator principal (*to*), a indicação do ator secundário (*from*), a indicação das trocas de valor que é o ator secundário a iniciar (*hasVEForward*), e a indicação das trocas de valor que é o ator principal a iniciar (*hasVEBackward*). A relação *ExchangeRelation* tem o mesmo conjunto de propriedades do ator secundário. Por último, a relação *VLARelation* tem como propriedades a indicação do elemento VLA (*to*) e a identificação do elemento *ValueExchange* (*from*).

Todos os identificadores dos elementos têm uma expressão associada que impede a criação de dois elementos com identificadores iguais. A Figura 5.3 apresenta essa expressão para o elemento da troca de valor (*ValueExchange*).

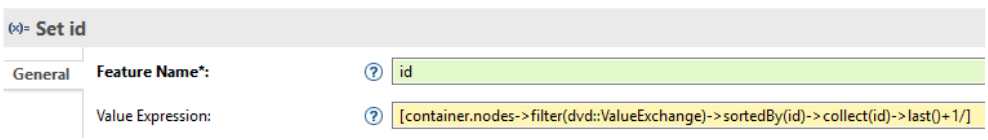


Figura 5.3: Expressão para a criação de identificadores no *ValueExchange*

De modo a melhorar a facilidade de uso da ferramenta, foram adicionadas funcionalidades que permitem editar e eliminar elementos e relações de forma mais simples e rápida para o utilizador. Com a funcionalidade de edição adicionada, permitimos que o utilizador edite a etiqueta dos elementos fazendo apenas duplo clique sobre o elemento pretendido. Desta forma é possível editar diretamente no diagrama, por exemplo, o nome de um ator. A funcionalidade de eliminar permite apagar (Figura 5.4) não só os elementos, bem como as relações do modelo, através da tecla *delete*.

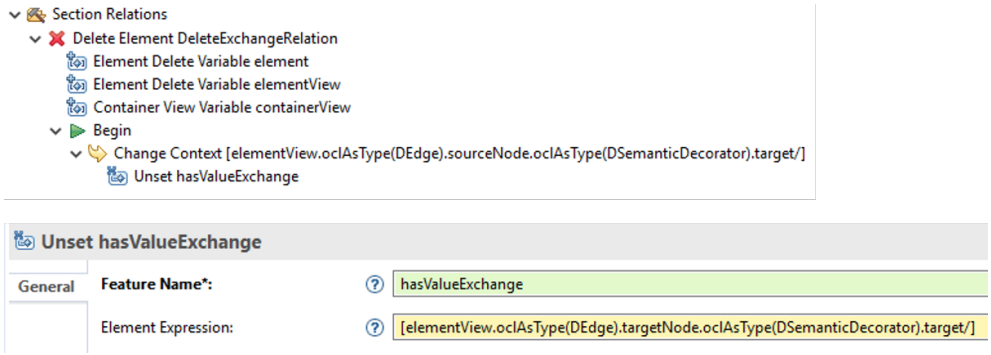


Figura 5.4: Implementação da funcionalidade para eliminar a relação *ExchangeRelation*

De modo a garantir boas práticas para uma boa construção do modelo de valor DVD, para além das que são garantidas a partir do metamodelo, foram escritas regras de validação. Essas regras de validação garantem que todos os elementos representados estão ligados a outro elemento. A Figura 5.5 apresenta a regra de validação que garante que o elemento *EnvironmentActor* está ligado a um elemento *ValueExchange*. Esta regra de validação mostra ao utilizador uma mensagem de erro a indicar que o elemento *EnviromenteActor* não tem uma relação *ExchangeRelation*.

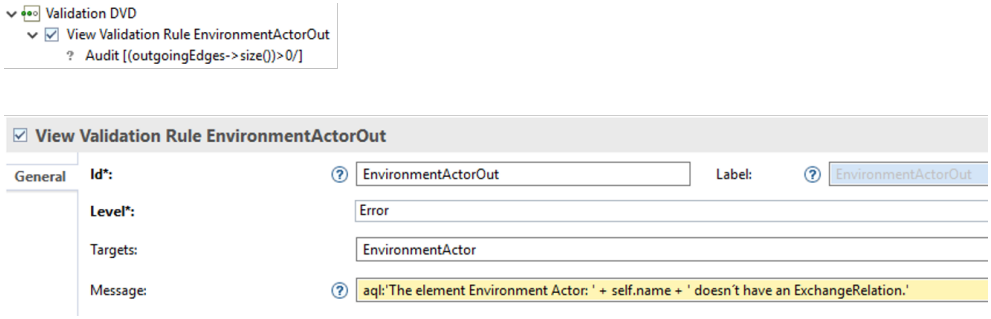


Figura 5.5: Regra de validação e a sua mensagem de erro

5.2 Editor para o Modelo KAOS4Services

A ferramenta desenvolvida permite criar e/ou editar um modelo KAOS4Services gerado a partir de uma transformação. Para tal, antes de criarmos esta transformação M2M, criámos

Tabela 5.2: Elementos da linguagem concreta KAOS4Services

Ícone	Nome do Elemento na paleta	Conceitos
	Goal	Objectivo do sistema
	Requirement	Requisito
	Expectation	Expectativa
	SoftGoal	Objetivo sem critérios
	Software Agent	Agente de software
	Environment Agent	Agente de ambiente
	LogicalOperation	Operador Lógico (AND e OR)
	Operation	Operação
	Scenario	Cenário

Tabela 5.3: Relações da linguagem concreta KAOS4Services

Ícone	Nome da Relação na paleta	Origem	Alvo
	Refinement	LogicalOperation	Goal, Requirement
		Softgoal, Requirement, Expectation	LogicalOperation
		Goal	LogicalOperation
	RequireSG	Requirement	Softgoal
	AResponsability	EnvironmentAgent	Expectation
		SoftwareAgent	Requirement
	Specialization	EnvironmentAgent	EnvironmentAgent
		SoftwareAgent	SoftwareAgent
	OperationLink	EnvironmentAgent, SoftwareAgent	Operation
		Operation	LogicalOperation
	LogicalRefinement	LogicalOperation	LogicalOperation

um editor KAOS4Services. A construção do modelo KAOS4Services é definida a partir do metamodelo da Figura 4.4. Este metamodelo representa a linguagem abstrata para este domínio específico.

O utilizador interage com a linguagem desenvolvida através de uma notação visual, isto é, a linguagem concreta. As tabelas 5.2 e 5.3 apresentam a linguagem concreta a que nos referimos. A Tabela 5.2 apresenta para cada elemento a representação visual, o nome do elemento na paleta do editor, e o conceito associado. A Tabela 5.3 apresenta para cada relação a representação visual, o nome da relação na paleta do editor, os elementos fonte, e os elementos alvo.

No editor do KAOS4Services, os elementos *Goal*, *Requirement*, *Expectation* e *Softgoal*

apresentam como propriedades o nome (*name*), a indicação se existe algum nível de granularidade inferior (*hasLO*), e a indicação se existe algum tipo de granularidade superior (*hasRefinement*). Para além das propriedades mencionadas o elemento *Requirement* tem ainda no seu conjunto de propriedades a indicação se está especificado em algum elemento *Softgoal* (*requireSG*), qual o agente de software responsável pela realização do requisito (*softwareAgent*), e qual o objeto de valor de entrada a partir do qual foi transformado (*valueObject*). Por sua vez, o elemento *Expectation* também tem na sua composição de propriedades a indicação de qual o agente de ambiente responsável pela realização da expectativa (*environmentAgent*), e qual o objeto de valor de saída a partir do qual foi transformado (*valueObject*). O elemento *Softgoal* tem também como propriedade a indicação se é especificação de algum requisito (*toRequirement*).

Os elementos *SoftwareAgent* e *EnvironmentAgent* têm como propriedades o nome (*name*) e a indicação das operações que os agentes devem cumprir de modo a atingirem os seus objetivos (*agentOperation*). Estes elementos podem ser especificados num outro elemento do mesmo tipo, sendo assim o elemento *SoftwareAgent* tem nas suas propriedades a indicação de qual *SoftwareAgent* o especifica (*saSpecification*). Do mesmo modo o elemento *EnvironmentAgent* tem no seu conjunto de propriedades a indicação de qual *EnvironmentAgent* o especifica (*eaSpecification*). Os elementos *SoftwareAgent* e *EnviromentAgent* tem também uma propriedade que indica por qual requisito (*rResponsability*) ou expectativa (*eResponsability*) são responsáveis, respetivamente.

Pertence ainda ao conjunto de elementos acima mencionados (*Goal*, *Requirement*, *Expectation*, *Softgoal*, *SoftwareAgent* e *EnvironmentAgent*) uma propriedade que indica o elemento pertencente ao modelo DVD a partir do qual este foi transformado, na transformação do modelo de valor DVD para o modelo KAOS4Services (*transformedFrom*). A Figura 5.6 apresenta a propriedade mencionada, num exemplo de um *SoftwareAgent* que foi transformado a partir de um *MainActor*.

Figura 5.6: Propriedade presente nos elementos do KAOS4Services transformados

O conjunto de propriedades do elemento *Scenario* é composto por um identificador (*id*), um título (*title*), uma descrição do cenário (*description*), uma cor (*color*) e pela indicação do conjunto de operações que o compõem (*operationOrder*). O identificador deste elemento tem uma expressão associada que impede a criação de dois cenários com identificadores iguais. A propriedade cor deste elemento tem como função facilitar a leitura do modelo, dado que a ordem das operações vão estar identificadas com a mesma cor que o cenário a que pertencem. Disponibilizamos ao utilizador a escolha de cinco cores: azul, verde, vermelho, amarelo ou roxo. Para que esta alteração de cor ocorra, adicionámos personalizações de estilo de modo a efetuar alterações nas propriedades gráficas deste elemento. A Figura 5.7 apresenta as personalizações de estilo desenvolvidas para o elemento *Scenario*.

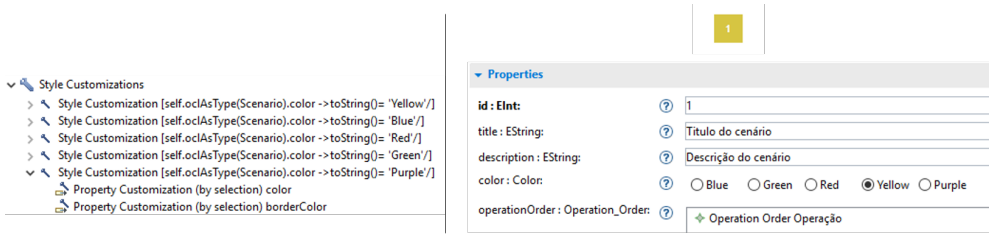


Figura 5.7: Personalizações de estilo do elemento *Scenario*

O elemento *Operation* é composto por dois nós: *operation* e *order*, conforme ilustra a Figura 5.8. Este elemento tem como conjunto de propriedades: a operação (*operation*), o agente que deve cumprir a operação (*agent*), o operador lógico a que está associada (*oRefinement*), a sua ordem (*order*), e qual o cenário a que pertence (*scenario*). O nó *order* deste elemento altera a cor da etiqueta conforme a cor do cenário a que está associado. Esta alteração ocorre devido às personalizações de estilo que adicionámos, tal como podemos observar na Figura 5.9.

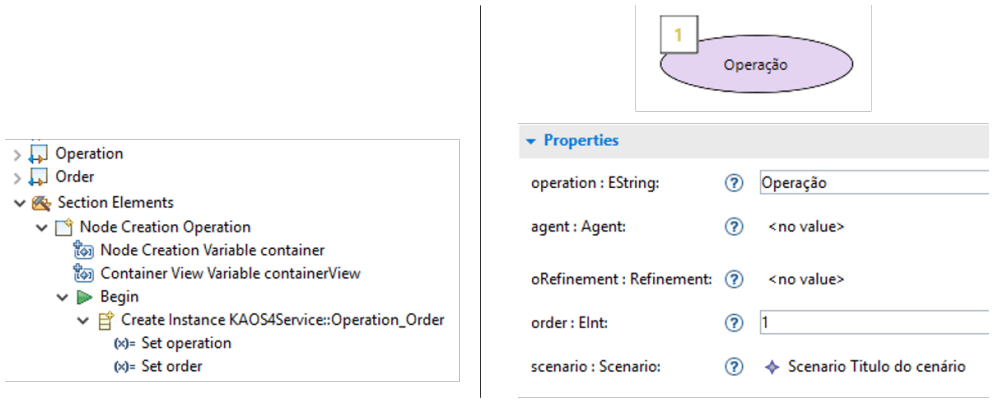


Figura 5.8: Constituintes do elemento *Operation* e suas propriedades

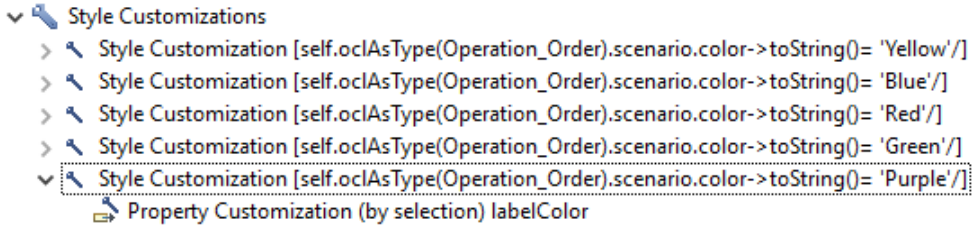


Figura 5.9: Personalizações de estilo do elemento *Order*

O elemento *LogicalOperation* é composto pelas seguintes propriedades: operador lógico (*logical*), pelas operações a que está associado (*operation*), pela indicação de outro operador lógico, com granularidade superior, a que posso estar ligado (*hasLogical*), pela indicação de outro operador lógico, com granularidade inferior, a que possa estar ligado (*logicalO*), pela indicação do objetivo com granularidade superior, a que está associado (*hasGoal*),

e pela indicação dos objetivos com granularidade inferior a que está associado (*goal*). A propriedade *logical* permite ao utilizador escolher entre o operador lógico *AND* e *OR*, evitando assim a criação de um elemento na paleta para cada operador lógico. Par tal, adicionámos uma personalização de estilo de forma a alterar propriedades gráficas neste elemento, Figura 5.10.

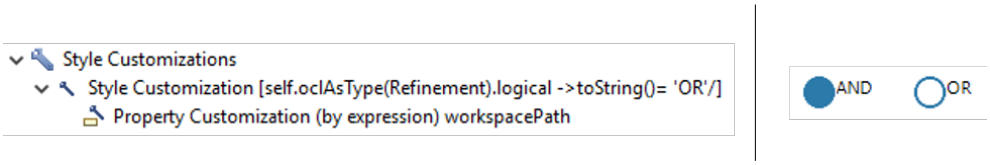


Figura 5.10: Personalização de estilo para escolha entre operador lógico AND e OR

De modo a facilitar a usabilidade da ferramenta, criámos relações que permitem unir mais do que dois elementos. Assim, ao diminuirmos consideravelmente o número de relações representadas na paleta, tornámos a ferramenta mais simples de usar. Como é o caso da relação *Refinement*, que permite fazer a ligação entre o elemento *LogicalOperation* e os elementos *Goal*, *Requirement*, *Expectation* e *SoftGoal*. Esta relação permite baixar o nível de granularidade do elemento *Goal*, dado que faz a ligação entre um *Goal* e um *LogicalOperation*, a ligação entre um *LogicalOperation* e os elementos *Goal* e *Requirement*, e a ligação entre os elementos *Requirement*, *Expectation* e *Softgoal* e o elemento *LogicalOperation*.

A relação *AResponsability* faz a ligação entre os agentes e os requisitos ou expectativas que estes devem alcançar. Sendo assim, esta relação agrupa duas relações, estabelecendo a ligação entre agentes de ambiente e expectativas, e a ligação entre agentes de software e requisitos. O mesmo acontece com a relação *Specialization* que permite que um agente se especialize num outro agente do mesmo tipo. Isto ocorre, pois esta relação agrupa duas relações, uma que permite que um agente de software se especialize num outro agente de software, e outra relação que permite que um agente de ambiente se especialize num outro agente de ambiente. Nesta relação foi introduzida uma expressão que impede que um agente se especialize em si próprio, Figura 5.11.

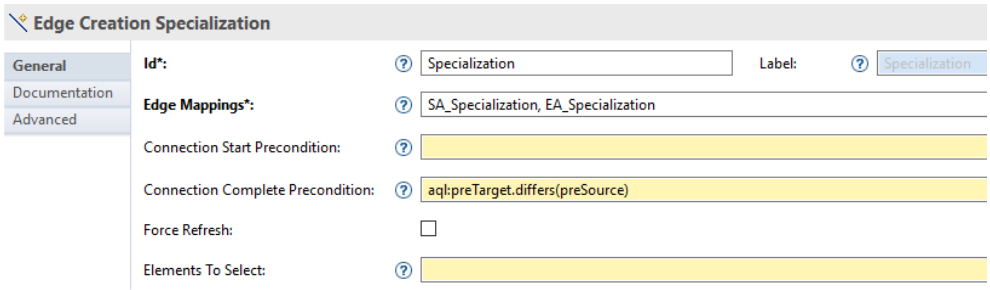


Figura 5.11: Expressão que impede o elemento de criar relações em si próprio

A relação *OperationLink* fornece as relações que unem uma operação com um operador lógico e com um agente. Deste modo, agrupa duas relações uma que estabelece a relação entre um agente e uma operação, e outra que une a operação a um operador lógico. A

relação *LogicalRefinement* permite estabelecer a ligação entre dois operadores lógicos. De modo a que não seja permitido unir um operador lógico a si próprio, adicionámos uma expressão semelhante à expressão adicionada na relação *Specialization*.

Tal como no editor DVD, para o editor KAOS4Services também adicionámos funcionalidades que permitem editar e eliminar elementos e relações de forma mais simples e rápida para o utilizador.

De forma a garantir regras de boa construção do modelo KAOS4Services, para além das regras que são garantidas a partir do metamodelo, foram também escritas regras de validação. As regras de validação que introduzimos garantem que todos os elementos representados estabelecem uma relação com outro elemento. Para além destas validações, adicionámos também uma validação que garante que o elemento *Softgoal* só pode estabelecer relações com mais um elemento. Isto é, o elemento *Softgoal* ou estabelece uma relação com um requisito, ou está associado a um operador lógico. Adicionámos também uma validação que garante que se um requisito foi decomposto em operações (através de um operador lógico), então não pode em simultâneo estar diretamente conectado a um agente, Figura 5.12. Esta regra de validação fornece ao utilizador uma mensagem de erro a indicar que o elemento *Requirement* não pode ter uma relação *AResponsability*.

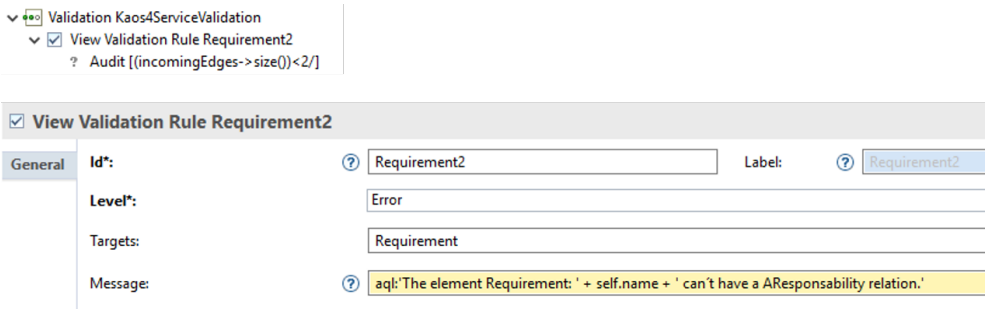


Figura 5.12: Regra de validação e mensagem de erro para o elemento *Requirement*

5.3 Editor para o Modelo SoaML

A ferramenta DVDTool desenvolvida também permite criar e/ou editar um modelo SoaML gerado a partir de uma transformação. Antes de criarmos esta transformação M2M, criámos um editor SoaML. A construção do modelo SoaML é definida a partir do metamodelo da Figura 4.9.

Mais uma vez, o utilizador interage com a linguagem desenvolvida através de uma notação visual, ou seja, através de uma linguagem concreta. A Tabela 5.4 apresenta essa linguagem concreta, mostrando para cada elemento e relação a sua representação visual, o nome do elemento ou relação na paleta do editor e ainda o conceito associado.

No editor SoaML, o elemento *ServicesArchitecture* tem no seu conjunto de propriedades um nome (*name*) e uma descrição (*description*). Este elemento contém os restantes elementos de um modelo SoaML (*Participant*, *ServiceContract*, e *QoSValue*), figura 5.13.

Ícone	Nome na paleta do editor	Conceitos
	ServicesArchitecture	Serviço de arquitetura
<<Participant>>	Participant	Participante
<<ServiceContract>>	ServiceContract	Serviço de negócio
<<QoSValue>>	QoSValue	Conjunto mínimo de regras de negócio
<<Consumes>>	Consumes	Indica o participante consumidor de um serviço
<<Provides>>	Provides	Indica o participante fornecedor de um serviço
<<QoSRequired>>	QoSRequired	Indica quais os QoSValues associados a um participante

Tabela 5.4: Linguagem concreta do SoaML

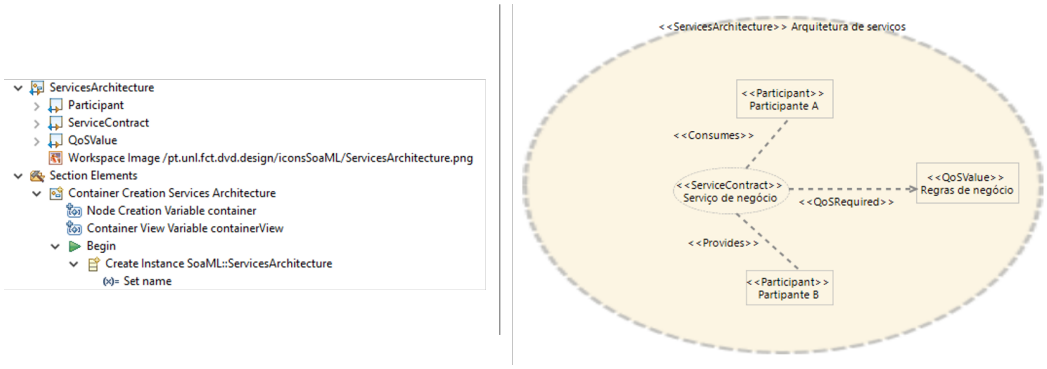


Figura 5.13: Componentes do elemento *ServicesArchitecture* e sua representação

O elemento *Participant* tem como propriedades um nome (*name*), a indicação de quais serviços de negócio que fornece (*hasProvides*), e a indicação de quais os serviços de negócio que consome (*hasConsumes*). O elemento *ServiceContract* tem no seu conjunto de propriedades um nome (*name*), a indicação dos conjuntos mínimos de regras de negócio a que está associado (*qoSValue*), a indicação de qual o ator que o fornece (*provides*), e a indicação de qual o ator que o consome (consumes). As propriedades deste elemento são apresentadas através de um exemplo, na Figura 5.14. O elemento *QoSValue* tem nas suas propriedades um nome (*name*) e a indicação do serviço de negócio a que está associado.

Para além das propriedades acima mencionadas, os elementos *ServicesArchitecture*, *Participant*, *ServiceContract*, e *QoSValue* têm uma propriedade que indica o elemento pertencente ao modelo DVD a partir do qual estes foram transformados do modelo de valor DVD para o modelo SoaML (*transformedFrom*).

As relações do modelo SoaML são relações simples, dado que cada uma tem como função fazer a relação entre dois elementos. As relações *Consumes* e *Provides* estabelecem a ligação do serviço de negócio para os participantes, e a relação *QoSRequired* estabelece a relação do serviço de negócio para o conjunto mínimo de regras. Por este editor de

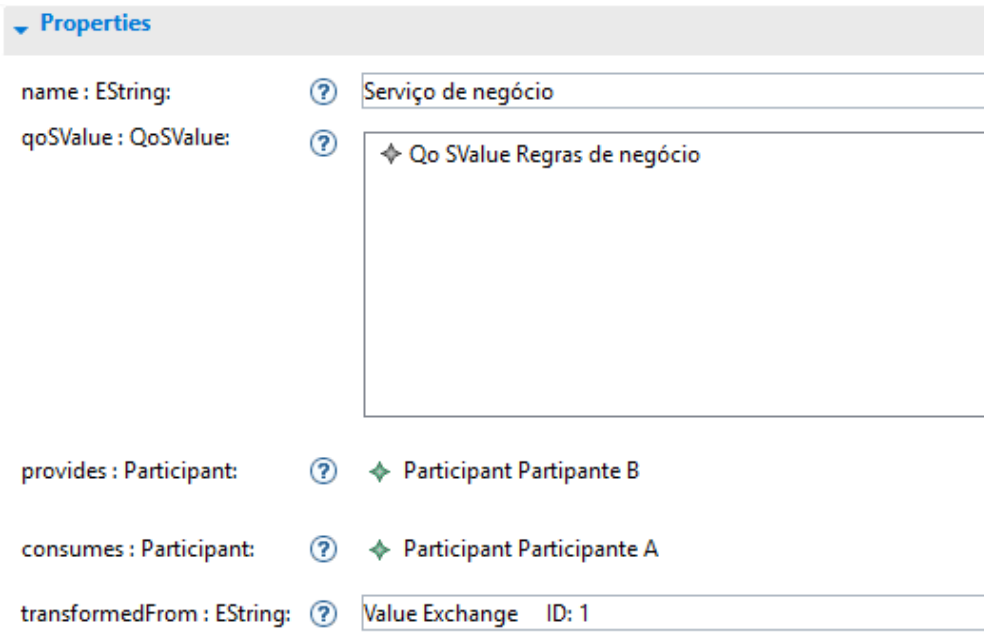


Figura 5.14: Conjunto de propriedades do elemento *ServiceContract*

modelos SoaML não necessitar de um grande número de relações, não foi necessário agrupá-las.

Tal como nos editores do DVD e KAOS4Services, também implementámos no editor SoaML as funcionalidades que permitem editar e eliminar elementos e relações de forma mais simples e rápida para o utilizador.

De forma a garantir as regras na construção do modelo SoaML, para além das que são garantidas a partir do metamodelo, implementámos regras de validação. As regras de validação escritas garantem que os elementos representados estabelecem relações com outos elementos. Os elementos *Participant* e *QoSValue* precisam de pelo menos uma relação, e o elemento *ServiceContract* precisa de pelo menos duas relações, as regras de validação para este elemento apresentam-se na Figura 5.15.

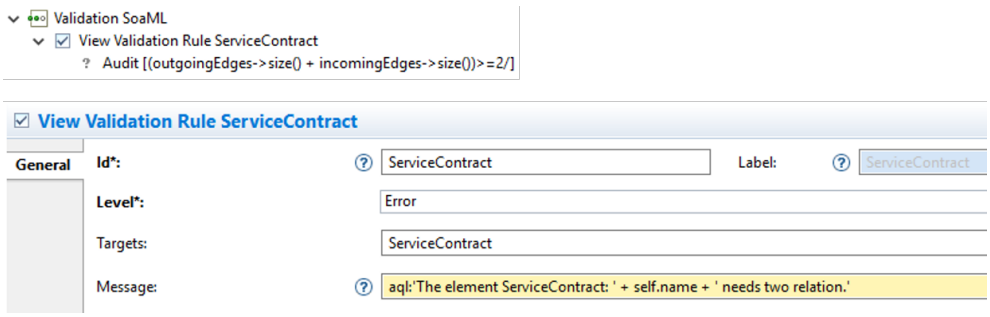


Figura 5.15: Regras de validação para as relações do elemento *ServiceContract*

5.4 Transformações M2M

A DVDTool fornece dois tipos de transformações de modelos para modelos, uma transformação do modelo DVD para o modelo KAOS4Services, e uma transformação do modelo DVD para o modelo SoaML. Para isso criámos ficheiros ETL, cujo código permite gerar as transformações.

5.4.1 Transformação do modelo DVD num modelo KAOS4Services

Nesta transformação começámos por criar um elemento *Goal* inicial e um *LogicalOperation AND*. A união entre estes dois elementos é estabelecida através da relação *hasGoal* (que no editor é representada pela relação *Refinement*).

Por cada troca de valor existente no modelo DVD, é gerado um elemento *Goal*, em que o nome corresponde à descrição da troca de valor, e a propriedade *transformedFrom* é preenchida com a indicação do tipo do elemento (neste caso, *Value Exchange*) e a indicação do seu *id*. Por sua vez, é criada a relação que une o elemento gerado ao operador lógico *AND*, criado no início da transformação, através da relação *hasRefinement* (que no editor é representada pela relação *Refinement*). Quando é gerado um elemento *Goal* a partir de uma troca de valor, é também criado um operador lógico *AND*, e estabelecida a relação entre estes dois elementos.

De seguida é gerado o elemento *Expectation* que corresponde ao objeto de valor de saída da troca de valor. A propriedade *transformedFrom* é preenchida com a indicação do tipo de elemento (neste caso, *OutValueObject*) e o *id* correspondente. A propriedade *valueObject* é correspondida à propriedade *object* do objeto de valor de saída. Na geração do elemento *Expectation* é ainda preenchido o nome deste elemento, que corresponde à descrição do objeto de valor de saída. À posteriori é criada a relação que estabelece a união entre o elemento *Expectation* gerado e o operador lógico criado aquando da geração do elemento *Goal*, proveniente da troca de valor.

Posteriormente é gerado o agente de ambiente que corresponde ao ator secundário associado à troca de valor. Ao elemento *EnvironmentAgent* é atribuído o nome do ator secundário, e a propriedade *transformedFrom* é preenchida com a indicação do tipo do elemento a partir do qual foi gerado (neste caso, *Environment Actor*), o seu nome, e o seu *id*. A relação entre o elemento gerado, e o elemento *Expectation* é criada a partir da relação *eResponsability* (que no editor é representada pela relação *AResponsability*). À propriedade *environmentAgent* do elemento *Expectation* é associado o agente ambiental gerado.

O próximo passo da transformação é gerar o elemento *Requirement*, que corresponde ao objeto de valor de entrada da troca de valor. A este elemento é atribuído o nome que corresponde à descrição do objeto de valor entrada. A propriedade *transformedFrom* é preenchida com a indicação do tipo do elemento a partir do qual foi gerado (neste caso, *InValueObject*), e o *id* correspondente. À propriedade *valueObject* é atribuído a propriedade *object* do objeto de valor de entrada. Para o elemento gerado é criada a relação que faz a

ligação com o operador lógico criado aquando da geração do elemento *Goal*, proveniente da troca de valor.

Por conseguinte é gerado o agente de software que corresponde ao ator principal do modelo DVD. Ao elemento gerado é atribuído o nome que está associado ao ator principal. A propriedade *transformedFrom* é preenchida com a indicação do tipo do elemento a partir do qual foi gerado (neste caso, *Main Actor*), o seu nome, e o *id* correspondente. A relação entre o elemento gerado, e o elemento *Requirement* é criada a partir da relação *rResponsability* (que no editor é representada pela relação *AResponsability*). À propriedade *softwareAgent* do elemento *Requirement* é atribuído o agente de software gerado.

Para concluir, por cada *Value Level Agreement* associado à troca de valor, é gerado um elemento *Softgoal*. A propriedade *transformedFrom* do elemento gerado é preenchida com a indicação do tipo de elemento a partir do qual foi gerado (neste caso, *Value Level Agreement*) e o *id* correspondente. A propriedade nome corresponde à descrição do valor mínimo acordado. Consequentemente, é criada a relação que estabelece a ligação entre o elemento gerado e o operador lógico criado aquando da geração do elemento *Goal*, proveniente da troca de valor.

Ao gerar a transformação do modelo DVD para o modelo KAOS4Services é gerado um ficheiro *.xmi* com a representação textual do modelo. Através do editor criado obtemos uma representação gráfica do modelo gerado. A Figura 5.16 apresenta, através de um exemplo, um modelo DVD fonte e um modelo KAOS4Services gerado pela transformação.

5.4.2 Transformação do modelo DVD num modelo SoaML

Para transformar um modelo DVD num modelo SoaML começámos por criar o elemento *ServicesArchitecture* e um elemento *Participant*. Ambos os elementos gerados proveem do elemento *MainActor* do modelo DVD. Assim, a propriedade *name* dos dois elementos é associada ao nome do ator principal. Na propriedade *transformedFrom* destes elementos é indicado o tipo do elemento a partir do qual foi gerado (neste caso, *MainActor*), o nome e o *id* associado. O elemento participante é associado ao elemento arquitetura de serviços através da relação *participant*.

Por cada ator secundário existente no modelo DVD, é gerado um elemento *Participant*. A este elemento é atribuído o nome que corresponde ao nome do ator secundário, a e propriedade *transformedFrom* é preenchida com o tipo de elemento (neste caso, *EnvironmentActor*), o nome e o *id* associado. O participante gerado é associado ao elemento *ServicesArchitecture* através da relação *participant*. Por cada troca de valor associada ao ator secundário é gerado um elemento *ServiceContract*. Na propriedade *name*, deste elemento, é atribuída a descrição da troca de valor, e a propriedade *transformedFrom* é preenchida com o tipo do elemento a partir do qual foi gerado (*Value Exchange*) e o *id* associado. Este elemento é associado ao serviço de arquitetura através da relação *serviceContract*.

Através da relação *StartRelation*, que une o ator secundário e o ator principal, é verificado em que propriedade (*hasVEForward* ou *hasVEBackward*) está associada a troca de valor.

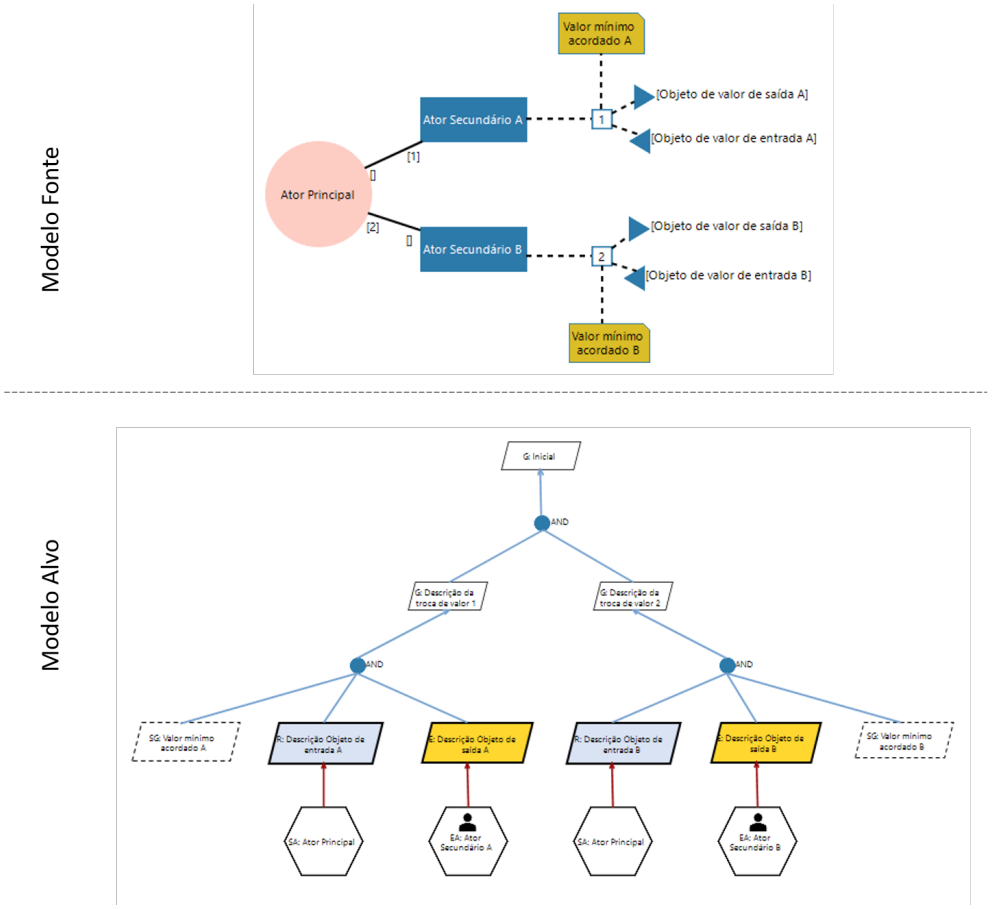


Figura 5.16: Modelo fonte e modelo alvo de uma transformação do modelo DVD para o modelo KAOS4Services

Se a troca de valor estiver associada à propriedade *hasVEForward* da relação *StartRelation*, então o participante proveniente do ator ambiental é associado ao elemento *ServiceContract* através da relação *provides*. Por outro lado, o participante proveniente do ator principal é associado ao elemento *ServiceContract* através da relação *consumes*. No entanto, se a troca de valor estiver associada à propriedade *hasVEForward* da relação *StartRelation*, o participante proveniente do ator ambiental é associado ao elemento *ServiceContract* através da relação *consumes*, e o participante proveniente do ator principal é associado através da relação *provides*.

Para concluir, por cada *Value Level Agreement* associado à troca de valor é gerado um elemento *QoSValue*, em que à propriedade *name* é atribuída a descrição do *VLA*. A propriedade *transformedFrom* do elemento gerado é preenchida com a indicação do tipo de elemento a partir do qual foi criado (neste caso, *Value Level Agreement*), e o *id* associado. Este elemento é associado ao elemento *ServiceContract* através da relação *qoSValue*, e associado ao elemento *Services Architecture* através da relação *qoSValue*.

Ao gerar a transformação do modelo DVD para o modelo SoaML é gerado um ficheiro *.xmi* com a representação textual do modelo. A representação gráfica do modelo gerado

obtém-se através do editor de SoaML que criámos. A Figura 5.17 apresenta, através de um exemplo, um modelo DVD fonte e um modelo SoaML gerado pela transformação.

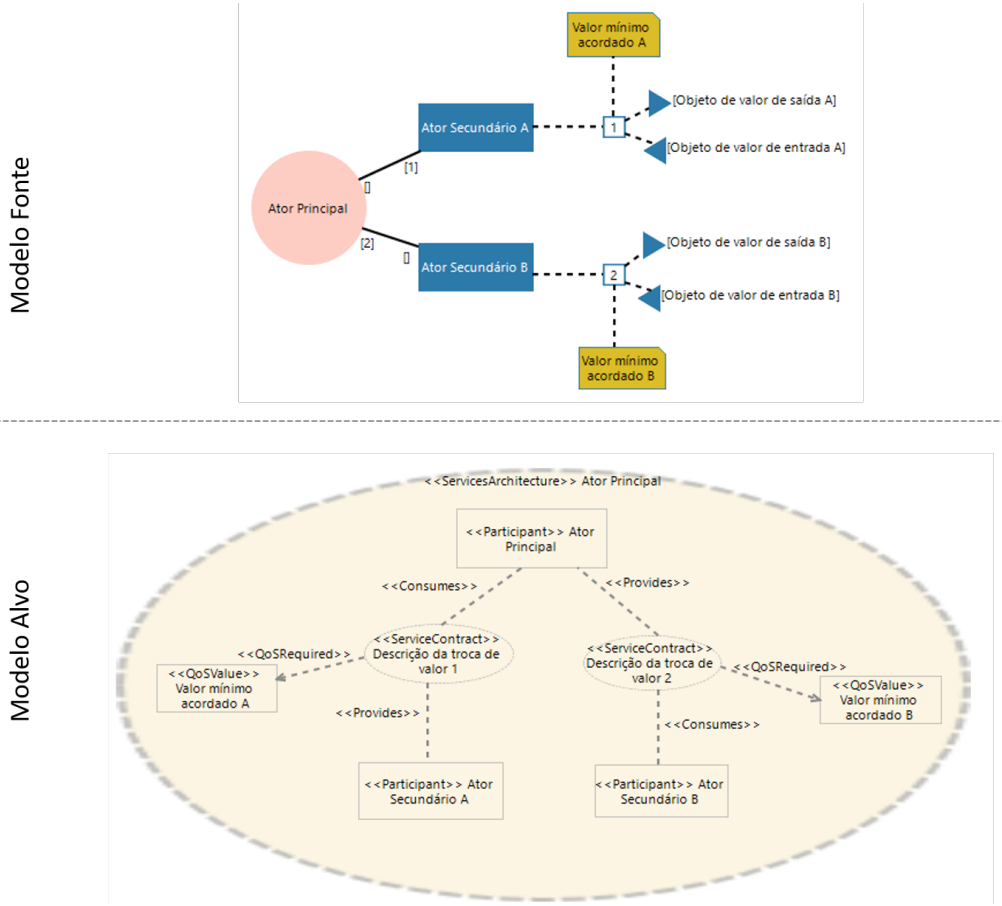


Figura 5.17: Modelo fonte e modelo alvo de uma transformação do modelo DVD para o modelo SoaML

5.5 Conclusões

Este capítulo apresenta a ferramenta que desenvolvemos e que nos permite fazer o alinhamento entre a área de negócios e a área de sistemas de informação usando o método DVD. A DVDTool é uma ferramenta que permite a construção de modelos de valor DVD, a geração de uma arquitetura de serviços em SoaML, e a geração de modelos de requisitos expressos em KAOS4Services. Oferece uma linguagem concreta visual para representar os vários modelos, uma linguagem abstrata com uma semântica bem definida, e ainda transformações de modelos.

Para cada um dos modelos desenvolvemos editores que permitem ao utilizador criar modelos, e editar os modelos gerados através das transformações. De modo a facilitar a usabilidade da ferramenta, alguns dos elementos e relações foram agrupadas, diminuindo o número de elementos apresentados nas paletas e tornando os editores mais intuitivos. No editor DVD, temos o exemplo do elemento *ValueExchange* apresentado na paleta, que

permite a construção em simultâneo de uma troca de valor, os objetos de valor a esta associados (objeto de valor de entrada e objeto de valor de saída) e ainda as relações destes elementos. Desta forma, o utilizador necessita de fazer menos cliques para representar estes elementos, tornando a sua representação mais rápida. Foram também adicionadas, as funcionalidades de editar e de remover, nos três editores. Estas funcionalidades contribuem para a facilidade e rapidez na construção dos modelos.

CAPÍTULO 6

CASO DE ESTUDO: LOJA VIRTUAL

Este capítulo tem por objetivo demonstrar todo o processo necessário para a construção de um modelo de valor DVD e a geração de transformações de modelos para modelos. Para isso, mostra a utilização da ferramenta para modelar um caso de estudo. Começa por explicar como se constrói um modelo de valor DVD, depois mostra como é que se executa uma transformação a partir do modelo DVD para um modelo KAOS4Services, analisa o modelo KAOS4Services obtido, explica como editar esse modelo, e, por último, ilustra uma transformação a partir do modelo DVD para um modelo SoaML.

6.1 Construção do modelo DVD

Para o caso de estudo da loja virtual apresentamos o cenário, que vai servir de base para a construção do modelo DVD:

Cenário: Os clientes compram bens numa loja virtual que lhes garanta a segurança dos seus pagamentos e ainda uma entrega rápida. Para reduzir custos, a loja virtual tem poucos produtos em stock, e por isso compra grande parte da mercadoria que vende a retalhistas que garantam entregas rápidas. Sempre que possível, as mercadorias são adquiridas diretamente aos fabricantes, de modo a aumentar a margem de lucro nos produtos que vende, ou a reduzir o seu preço aos clientes.

Usando a ferramenta desenvolvida, vamos começar por criar o modelo de valor DVD, em que os elementos são apresentados na paleta do editor, Figura 6.1. A paleta está dividida em duas secções:

- Elementos: contém os elementos necessários à construção de modelos de valor DVD;
- Relações: contém as relações que estabelecem a ligação entre os elementos do modelo.

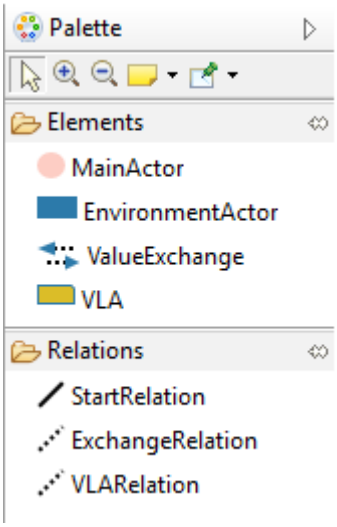


Figura 6.1: Paleta do editor DVD

Analisando o cenário apresentado, identificamos os atores, que neste caso são o cliente, a loja virtual, o retalhista e o fabricante. O passo seguinte é definir o ator principal, que é aquele em que a nossa atenção se foca a cada momento. Os restantes atores são atores secundários. Para cenários diferentes centrados num outro ator do mesmo modelo de negócio, o ator principal muda, resultando num modelo DVD diferente, centrado nesse outro ator.

Para este exemplo, a loja virtual é o ator principal. Este será assim o nó principal deste modelo. A loja virtual é então representada através do elemento *MainActor*, que consta da paleta. Os restantes atores (cliente, retalhista e fabricante) são os atores secundários. Inicialmente vamos focar-nos no ator Cliente, que é representado no modelo através do elemento *EnvironmentActor*. A ligação entre este ator secundário e o ator principal faz-se através da relação *StartRelation*. Neste cenário, o cliente compra um bem numa loja virtual. Esta troca de valor é representada no modelo, através do elemento *ValueExchange*, representado na paleta (que fica identificada por um id), podendo ser possível acrescentar uma descrição a esta troca de valor. Neste caso indicamos na descrição que foi a Venda de um Bem, Figura 6.2.

O elemento *Value Exchange* (neste caso, identificado com o id "1") é composto por dois fluxos da troca de valor (setas triangulares). Neste cenário a troca de valor é feita através do pagamento que o cliente dá à loja, em troca do bem que recebe. O fluxo de saída do *Value Exchange* (seta que sai) representa o pagamento, ao qual se pode adicionar a descrição "Valor monetário do bem", Figura 6.3. Ao fluxo de entrada (seta que entra) adicionamos o bem e a descrição "Entrega do Bem", Figura 6.4. A ligação entre o ator Cliente e a troca de valor é representada através da relação *Exchange Relation*.

Para cada troca de valor, é preciso identificar a sua origem. Para isso serve a relação *StartRelation*, através das propriedades *hasVEForward* para representar as trocas de valor iniciadas pelo ator secundário, e *hasVEBackward* para representar as trocas de valor

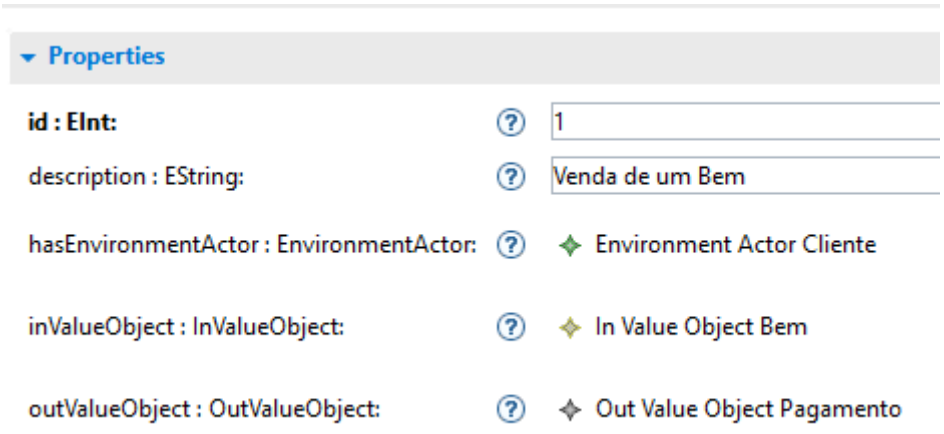


Figura 6.2: Propriedades do elemento *Value Exchange*

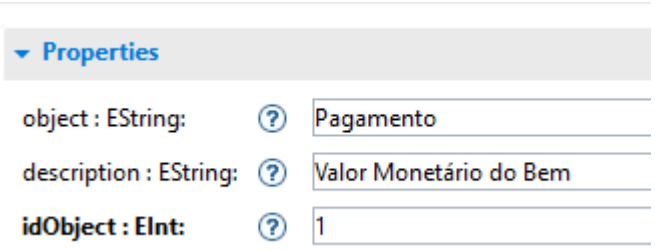


Figura 6.3: Propriedades do *OutValueObject*

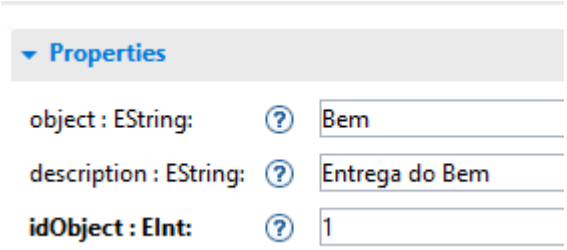


Figura 6.4: Propriedades do *InValueObject*

iniciadas pelo ator principal. Neste caso, o cliente é o responsável por iniciar esta troca de valor concreta, porque efetua o pagamento antes de receber o bem. Então, na relação *StartRelation* que liga estes dois atores, é colocado o identificador da troca de valor na propriedade *hasVEBackward*, Figura 6.5.

Por último definimos os níveis de valor acordados entre os atores. Neste cenário, a troca de valor acontece porque o pagamento é seguro. O nível de valor acordado é representado através do elemento *VLA*, contido na paleta. A ligação entre o elemento *ValueExchange* e *VLA* é feita através da relação *VLA Relation*. As restantes trocas de valor entre a loja virtual, o retalhista e os fabricantes são identificadas e representadas de forma semelhante.

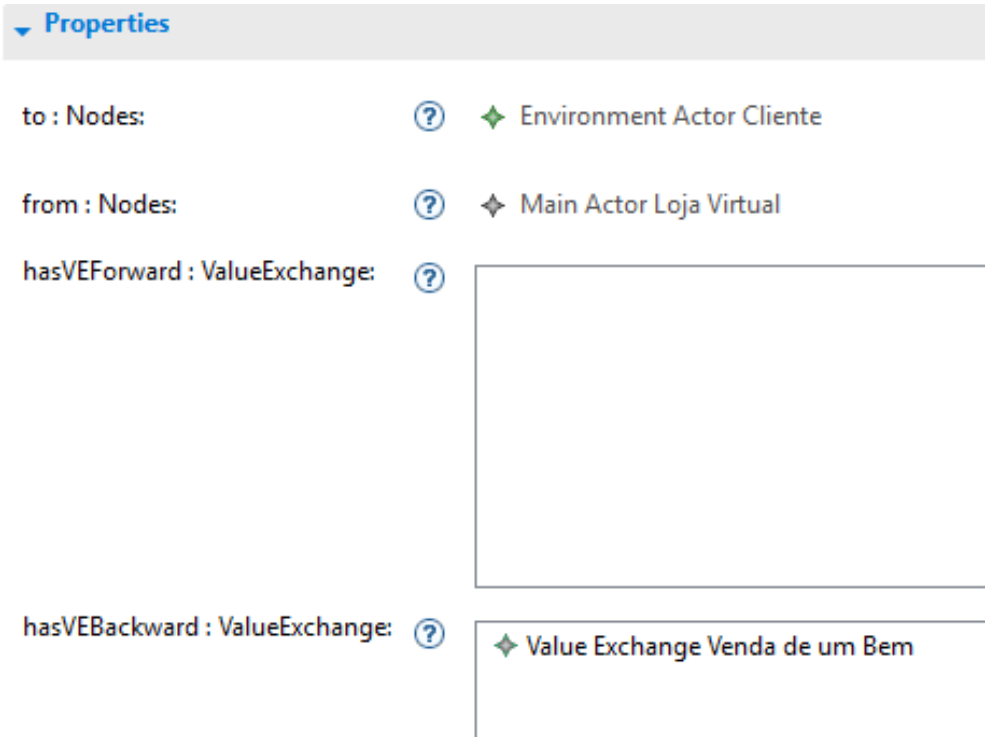


Figura 6.5: Propriedades pertencentes à relação StarRelation

Sendo que a loja virtual compra mercadoria ao retalhista, os valores trocados são mercadoria por um pagamento. Sendo assim, criamos a representação da troca de valor (neste caso, identificada com o id "2"), à qual podemos adicionar a descrição "Compra de Mercadoria ao Retalhista". Nesta troca de valor o fluxo de saída é a "Mercadoria", à qual acrescentamos a descrição "Mercadoria do Retalhista". Por outro lado, o valor do fluxo de entrada é o "Pagamento", em que adicionamos a descrição "Pagamento da Mercadoria ao Retalhista". Quem inicia esta troca é a loja virtual, por efetuar o pagamento antes de receber a mercadoria do retalhista. Consequentemente, o identificador desta troca de valor é colocado na propriedade "hasVEForward" da relação StartRelation. Para esta troca de valor, o nível de valor acordado entre os dois atores é a "Entrega Rápida"das mercadorias.

Por último, no caso da troca de valor entre a loja virtual e o fabricante, os valores trocados também são mercadoria por um pagamento. Na representação desta troca de valor (com o identificador "3"), a descrição é a "Compra de Mercadoria ao Fabricante". Sendo que o fluxo de saída é "Mercadoria", em que a descrição é "Mercadoria do Fabricante", e o fluxo de entrada corresponde ao "Pagamento", com a descrição "Pagamento da Mercadoria ao Fabricante". A loja virtual inicia a troca por efetuar o pagamento antes de receber a mercadoria. Então, o identificador da troca de valor é colocado na propriedade *hasVEForward* da relação *StartRelation* que une os dois atores. Por último, o nível de valor acordado para esta troca de valor é a importância das mercadorias terem um "Baixo Custo".

Com a representação das trocas de valor entre o ator principal Loja Virtual e os atores

secundários, Cliente, Retalhista e Fabricante, obtemos o modelo de valor DVD completo, ilustrado na Figura 6.6, para o cenário discutido.

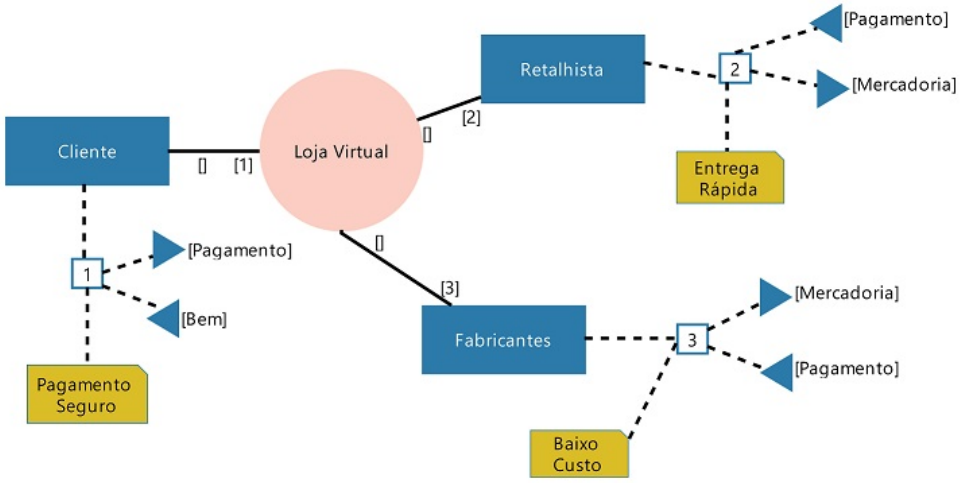


Figura 6.6: Modelo DVD Loja Virtual

6.2 Geração do modelo KAOS4Services

Ao modelo de valor DVD da loja virtual criado na secção anterior, aplicamos uma transformação para gerar um modelo KAOS4Services. Para tal, abrimos as configurações da execução, tal como indicado na Figura 6.7, de modo a criar uma transformação onde definimos o nome da transformação que pretendemos. De seguida, escolhemos o ficheiro ETL correspondente à transformação do modelo de valor DVD para o modelo KAOS4Services (DVD2KAOS4Service.etl), como mostra a Figura 6.8.

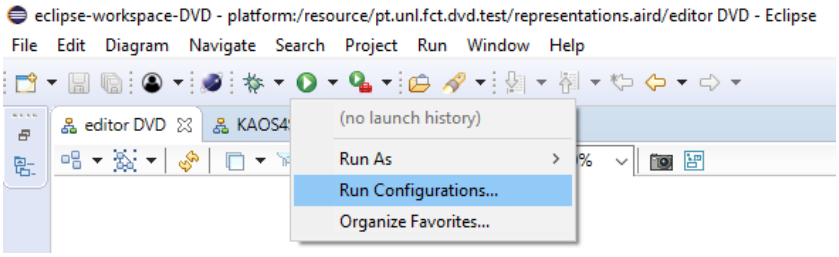


Figura 6.7: Menu para abrir janela de execução da transformação

Posteriormente indicamos o modelo fonte desta transformação M2M. Ou seja, adicionamos um EMF Model, ao qual atribuímos o nome *dvd*, e selecionamos o modelo DVD já criado (main.dvd), figura 6.9. Do mesmo modo, indicamos o modelo alvo desta transformação de modelos. Isto é, adicionamos um EMF Model, ao qual atribuímos no nome *KAOS4Services*, e indicamos o ficheiro para onde pretendemos gerar o modelo KAOS4Services (KAOS4ServiceModelTrans.xmi), Figura 6.9.

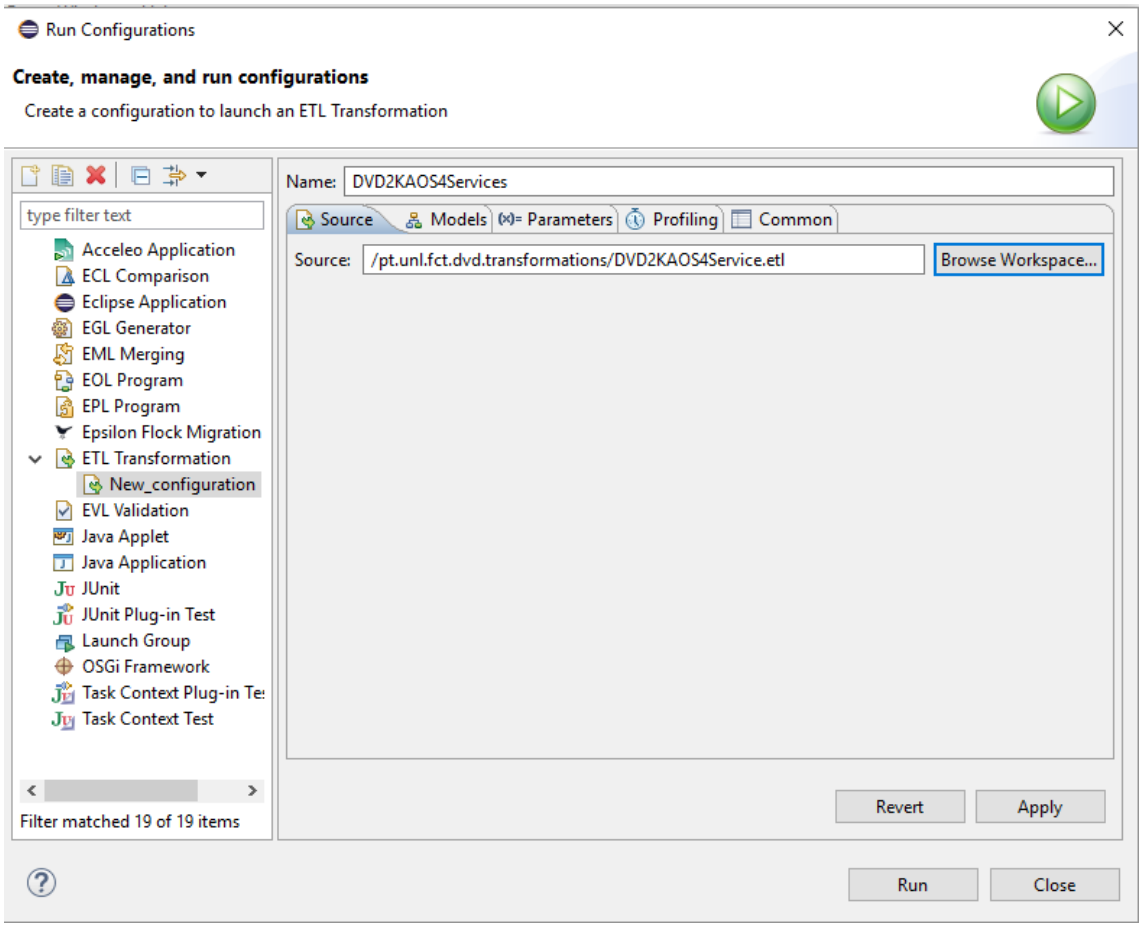


Figura 6.8: Seleção do ficheiro ETL para a transformação do modelo DVD para o modelo KAOS4Services

Para concluir, basta executar a transformação obtendo assim o ficheiro KAOS4Services pretendido, apresentado na Figura 6.10 [1].

6.2.1 Análise do modelo KAOS4Services gerado

A geração do modelo KAOS4Services resulta do mapeamento entre os conceitos do modelo de valor DVD e o modelo KAOS4Services, apresentado na Figura 4.6 da Secção 4.2.3. Analisando o modelo obtido podemos compreender concretamente a relação entre o modelo de valor DVD e o modelo KAOS4Services. Para efetuar esta análise, concentremo-nos na troca de valor entre a loja virtual e o cliente. A transformação do modelo que diz respeito às trocas de valor ocorridas entre a loja virtual e o retalhista e a loja virtual e o fabricante são semelhantes. Cada troca de valor apresentada no modelo DVD representa um objetivo que a loja virtual deve atingir. Esse objetivo, indicado na descrição da troca de

[1] Para facilitar a aprendizagem dos utilizadores, nós disponibilizamos um vídeo no YouTube a demonstrar todo este processo $https://youtu.be/dHlj_SzVVL4?t=2m25s$

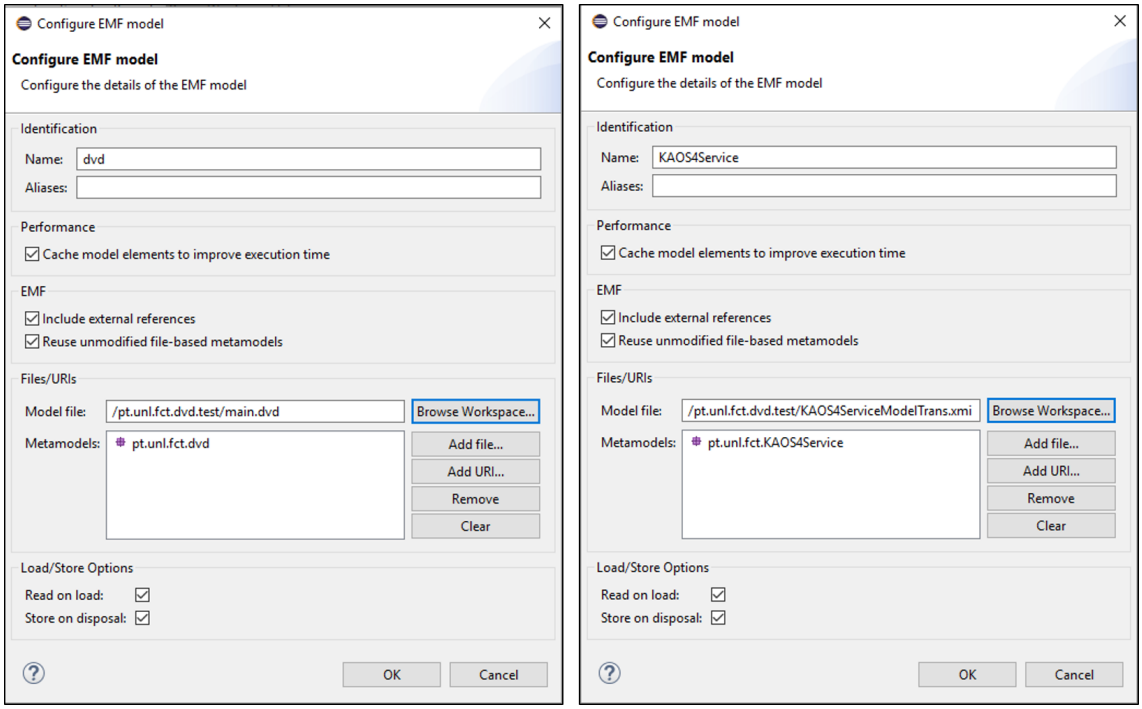


Figura 6.9: Inclusão do modelo fonte DVD e do modelo alvo KAOS4Services para a transformação

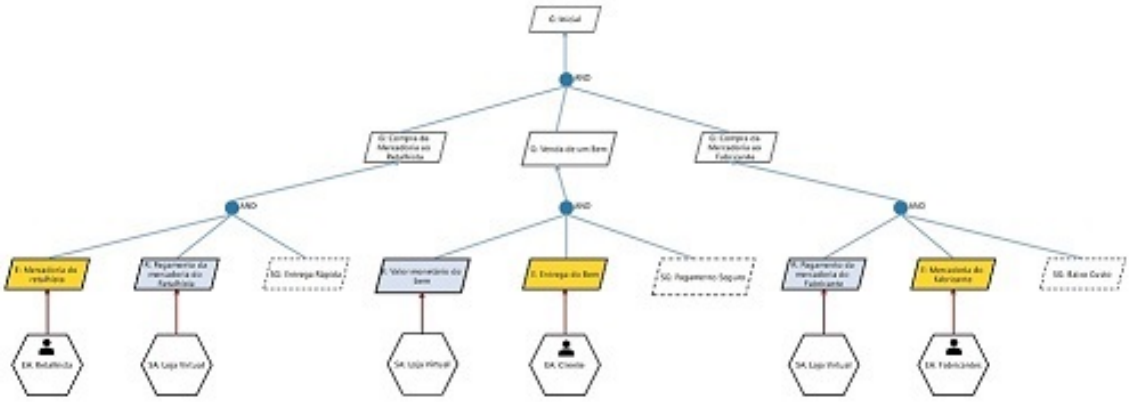


Figura 6.10: Modelo KAOS4Services obtido a partir da transformação do modelo DVD

valor (“Venda de bem”), é apresentado no modelo KAOS4Services como um *Goal*. Cada objetivo (*Goal*) é refinado em expectativas, requisitos e softgoals.

O valor que a loja virtual espera receber do cliente (descrição do fluxo de saída — valor monetário do bem) indica a expetativa da loja virtual em relação ao objetivo. Esse objetivo é representado no modelo gerado como uma *Expectation*. O responsável pela concretização da expetativa é o ator secundário, neste caso o cliente, que no modelo KAOS4Services é representado como um agente secundário. Por outro lado, o valor que a loja tem de fornecer ao cliente (descrição do fluxo de entrada — entrega do bem) indica o requisito

que a loja virtual tem de cumprir para atingir o objetivo. Esse objetivo é representado no modelo KAOS4Services como *Requirement*. O responsável pela concretização do requisito é o ator principal. Neste caso, a loja virtual, que é representada como agente principal.

Por último, o valor mínimo acordado entre a loja virtual e o cliente para a troca de valor, está relacionado com a especificação de atributos e requisitos de qualidade, sendo assim um objetivo para o qual não há critérios para a sua satisfação. Este objetivo indicado no valor mínimo acordado (pagamento seguro) é apresentado no modelo KAOS4Services como um *Softgoal*.

6.2.2 Edição do modelo KAOS4Services

Ao modelo KAOS4Services obtido podemos adicionar um conjunto de elementos, para além dos que já se encontram representados no modelo. Estes elementos são: o cenário, a operação e a sua ordem.

Para percebermos melhor esta edição do modelo KAOS4Services vamos considerar que, para que a loja cumpra o requisito de obter o valor monetário do bem, tem de verificar a disponibilidade do bem em stock, e posteriormente disponibilizar o método de pagamento por PayPal ou MB. Com recurso à paleta disponibilizada na ferramenta (Figura 6.11) adicionamos um cenário ao modelo. O cenário adicionado refere-se ao processo de efetuar uma venda. De seguida, eliminamos a relação que une o agente "Loja virtual" com o requisito "Entrega do bem". Para fazer a representação do cenário, começamos por adicionar a operação "Verificar Stock" que ligamos ao Requisito através de um operador lógico *AND*. A ligação entre o operador lógico e o requisito é feito pela relação *Refinement*, e as ligações entre a operação e o operador lógico, e entre o agente e a operação são feitas através da relação *OperationLink*. De seguida indicamos a que cenário pertence a operação e qual a sua ordem, tal como mostra a Figura 6.12.

Posteriormente, representamos as operações "Pagamento por PayPal" e "Pagamento por MB". Neste caso, ou acontece uma, ou outra, pelo que estas operações são ligadas por um operador lógico *OR*, através da relação *OperationLink*, Figura 6.13. A ligação entre os dois operadores lógicos é feita através da relação *Logical Refinement*. Para concluir, fazemos a ligação entre o agente e as operações através da relação *OperationLink*.

O modelo KAOS4Services editado encontra-se na Figura 6.14. [2]. A edição dos modelos KAOS4Services gerados pelas transformações efetuadas, podem ser realizadas não só através da adição dos elementos demonstrados no cenário acima, como também por elementos que já compõem o modelo gerado. Assim, é possível a edição das propriedades dos elementos representados no modelo, bem como a adição de mais elementos do mesmo tipo (como, por exemplo, agentes, objetivos, requisitos, expectativas, operadores lógicos, e as respetivas relações que unem os elementos).

[2]Para facilitar a aprendizagem dos utilizadores, nós disponibilizamos um vídeo no YouTube a demonstrar todo este processo $https://youtu.be/dHlj_SzVVL4?t = 4m58s$

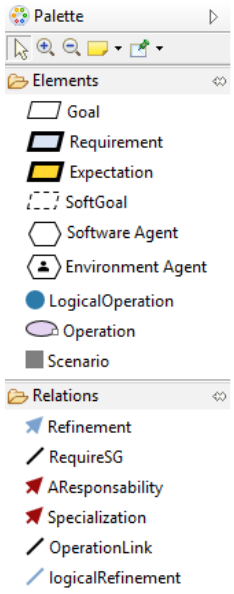


Figura 6.11: Paleta do editor KAOS4Services

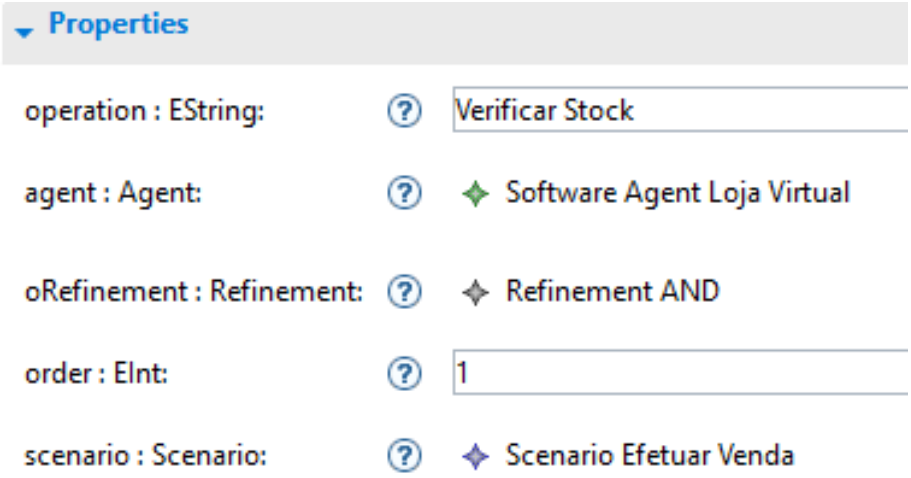


Figura 6.12: Propriedades do elemento Operation

6.3 Geração do modelo SoaML

De forma semelhante à obtenção do modelo KAOS4Services (Secção 6.2), o modelo SoaML é obtido por meio de uma transformação M2M, em que o modelo origem é o modelo DVD da loja virtual representado na Figura 6.6. Assim, começamos por abrir as configurações da execução de modo a criar uma transformação, onde definimos o nome da transformação que pretendemos. De seguida, escolhemos o ficheiro ETL correspondente à transformação do modelo de valor DVD para o modelo SoaML (DVD2SoaML.etl), figura 6.15.

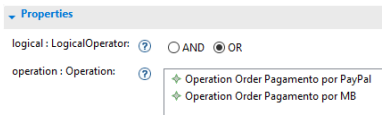


Figura 6.13: Propriedades do operador lógico OR

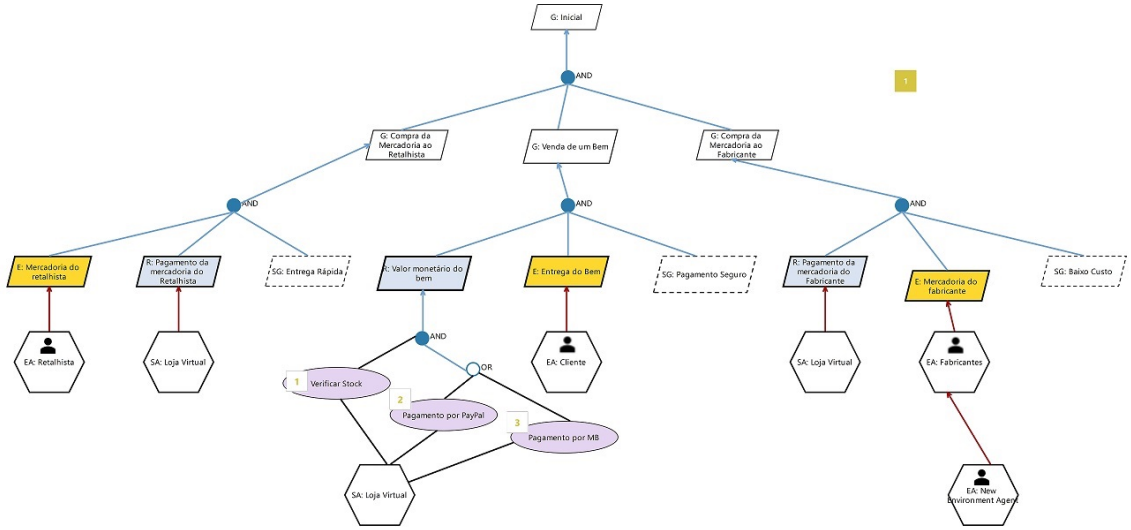


Figura 6.14: Modelo KAOS4Services com elementos adicionados

Posteriormente indicamos o modelo fonte da transformação de modelos, isto é, adicionamos um EMF Model ao qual atribuímos o nome *dvd*, e selecionamos o modelo DVD já criado (main.dvd), tal como ilustrado na Figura 6.16. Da mesma forma indicamos o modelo alvo desta transformação de modelo para modelo. Ou seja, adicionamos um EMF Model, ao qual atribuímos o nome *SoaML*, e indicamos o ficheiro, para onde pretendemos gerar o modelo SoaML (SoaMLTrans.xmi), figura 6.16.

Para finalizar o processo, basta executar a transformação obtendo assim o ficheiro SoaML pretendido, apresentado na Figura 6.17. [3].

6.3.1 Análise do modelo SoaML gerado

A geração do modelo SoaML resulta do mapeamento entre os conceitos do modelo de valor DVD e do modelo SoaML, apresentado na Figura 4.10. Analisemos o modelo da loja virtual que se refere à troca de valor entre a loja virtual e o cliente. (As transformações das restantes subsecções ocorrem de forma semelhante à secção em que nos vamos focar.) O ator principal do modelo DVD loja virtual corresponde a uma arquitetura de serviços em SoaML, visto que são o foco em ambos os modelos. Deste modo, o elemento *Service Acchitecture* do modelo SoaML fica com o nome do ator principal "Loja Virtual". Por

[3]Para facilitar a aprendizagem dos utilizadores, nós disponibilizamos um vídeo no YouTube a demonstrar todo este processo https://youtu.be/70xJjpUiWvM?t=2m42s

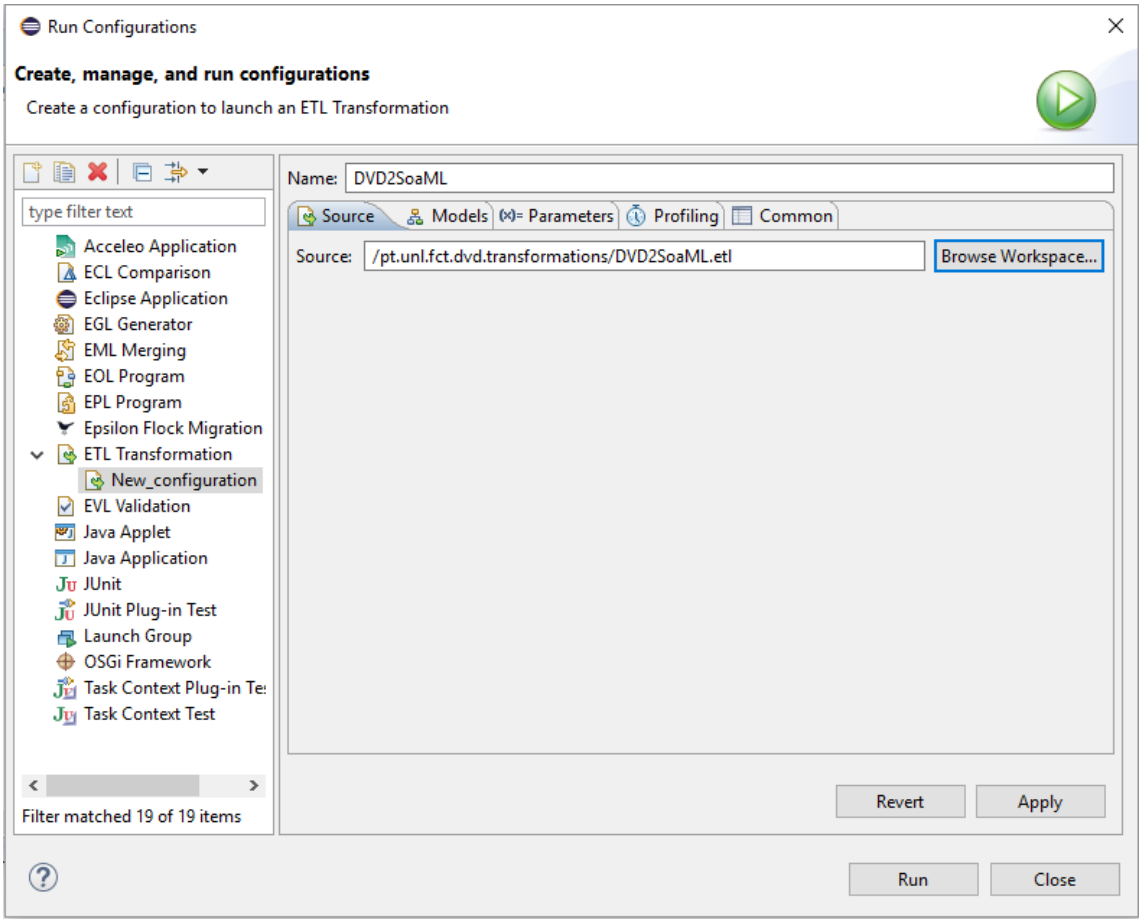


Figura 6.15: Seleção do ficheiro ETL para a transformação do modelo DVD para o modelo SoaML

outro lado, os atores do modelo de valor DVD são transformados em participantes no modelo SoaML, já que os participantes são sujeitos que fornecem (isto é, oferecem) ou consomem serviços de negócios. Deste modo, tanto o ator principal "loja virtual", como o ator secundário "cliente", são representados como participantes.

A operacionalização de uma troca de valor é um serviço de negócio, que está implícito na representação da troca de valor. Este serviço em SoaML é representado pela descrição identificada na troca de valor do DVD. Por exemplo, a descrição da troca de valor representada neste subconjunto é a "Venda de Bem". A informação de quem inicia a troca de valor, serve para identificar em SoaML o participante fornecedor e o participante consumidor desse serviço. Quem inicia a ação consome um serviço que é fornecido por outro participante. Desta forma, quem inicia é o consumidor e o outro é o fornecedor. Neste caso, como o cliente é quem inicia a troca de valor, então o participante cliente vai ser o consumidor, e o participante loja virtual vai ser o fornecedor.

O nível de valor acordado define um conjunto mínimo de regras de negócios ou atributos de qualidade acordados entre os atores. Este é representado em SoaML através

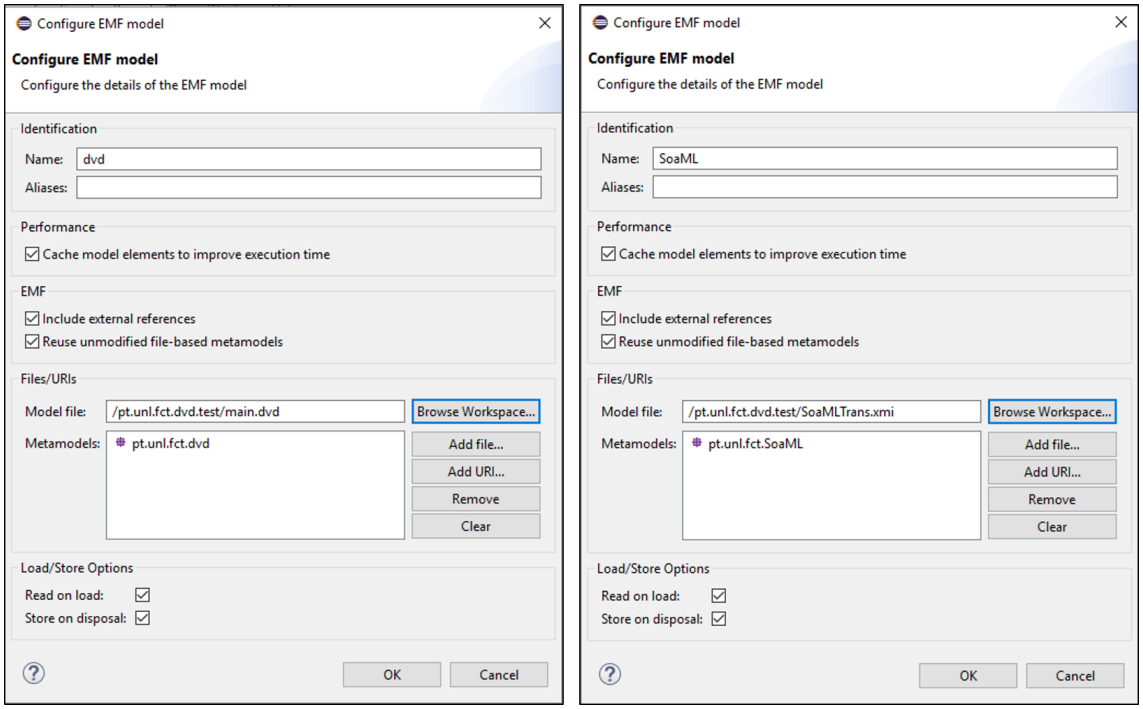


Figura 6.16: Inclusão dos modelos fonte e alvo para a transformação

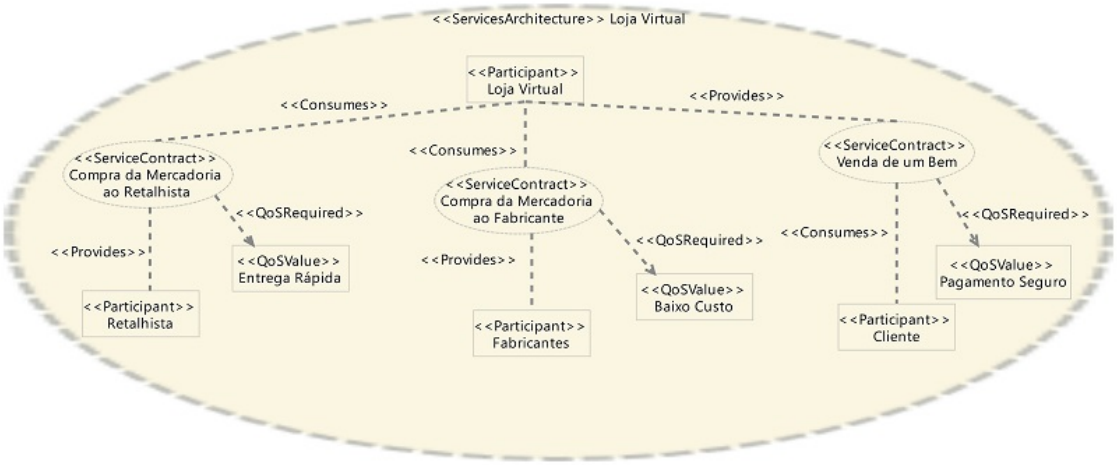


Figura 6.17: Modelo SoaML obtido a partir da transformação do modelo DVD

do elemento *QoSValue*, que mostra os atributos de qualidade requeridos para satisfazer o serviço de negócio. Neste caso, é o atributo "pagamento seguro".

Tal como ocorreu com o modelo KAOS4Services obtido por transformação, o modelo SoaML também pode ser editado através da paleta (Figura 6.18). Neste caso, a edição do modelo é feita através da edição das propriedades dos elementos representados no modelo, bem como a adição de mais elementos do mesmo tipo (como por exemplo, participantes,

serviços, *QoSValue*, e as respetivas relações que unem os elementos).

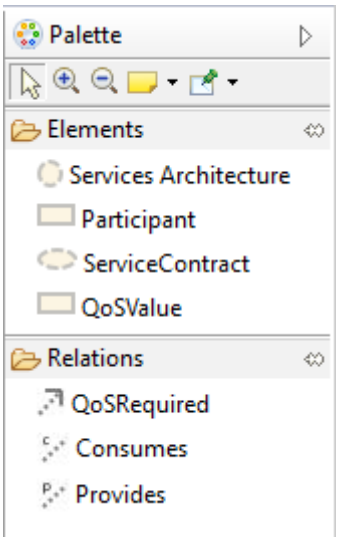


Figura 6.18: Paleta do editor SoaML

6.4 Conclusões

Este capítulo ilustra a utilização da ferramenta DVDTool com um caso de estudo, de modo a mostrar a interação que o utilizador tem com a ferramenta. O cenário que apresentámos foi o de uma loja virtual com as trocas de valor que esta executa com os seus clientes, fornecedores e retalhistas. A partir deste cenário começámos por aplicar o método DVD de criar o modelo de valor loja virtual.

Com o modelo criado, mostrámos como é simples e rápido executar a transformação do modelo DVD para um modelo KAOS4Services. Analisámos o modelo gerado, de modo a clarificar a relação entre o modelo DVD da loja virtual e o modelo gerado pela transformação. Posteriormente, exemplificámos como é que o modelo KAOS4Services da loja virtual pode ser editado. A edição deste modelo pode ser efetuada tanto a partir de modificações nas propriedades dos elementos representados, como da adição de um novo conjunto de elementos.

De forma semelhante, apresentámos o método para gerar uma transformação do modelo DVD para um modelo SoaML. De seguida analisámos o modelo SoaML gerado, de modo a tornar evidente a relação entre o modelo DVD e o modelo SoaML gerado. Por último referimos que é possível editar o modelo obtido através da transformação, alterando as propriedades dos elementos representados ou adicionando novos elementos ao modelo.

CAPÍTULO 7

AVALIAÇÃO

O método experimental é considerado a abordagem mais adequada para experimentação na área de Engenharia de Software [61]. Este método desenvolve uma abordagem qualitativa e/ou quantitativa para medir, analisar e avaliar hipóteses. Neste capítulo, descrevemos um quase-experimento usado para avaliar a nossa proposta.

7.1 Qual a eficácia percebida e a intenção de uso do suporte ferramental para modelação de valor de negócio?

Esta secção descreve um **quase-experimento** para avaliar a eficácia percebida (isto é, a *facilidade de uso percebida* e a *utilidade percebida*) e a intenção de uso da nossa proposta. Assim, as questões de pesquisa para a nossa avaliação são as seguintes:

RQ1 : *A DVDTool é percebida como fácil de usar?*

RQ2 : *A DVDTool é considerada útil?*

RQ3 : *Caso necessário, os participantes têm intenção de usar a DVDTool no futuro?*

7.1.1 Desenho do experimento

Para avaliar as questões de pesquisa, seguimos o processo GQM [9], definindo o objetivo desta experiência como **analisar** o suporte ferramental **com a finalidade de** verificar a eficácia percebida **em relação à** facilidade de uso percebida, à utilidade percebida e intenção de uso, **do ponto de vista de** modeladores com diversos graus de experiência **no contexto da** área de engenharia informática.

Contexto da experiência Focámos a nossa avaliação em modeladores com diversos graus de experiência, já que um dos objetivos do DVD é fornecer uma linguagem fácil e útil que ajudará desde os modeladores menos experientes aos modeladores mais experientes a especificar serviços de negócios. Embora projetado para analistas de negócios, o DVD pode ser também pode ser usado por engenheiros de software. Os participantes do experimento são **5** estudantes em informática e **10** engenheiros informáticos. Todos os participantes eram voluntários e estavam cientes dos propósitos práticos e pedagógicos do experimento, mas não conheciam as hipóteses experimentais nem tinham conhecimento prévio sobre o DVD.

Formulação de hipóteses Formulámos três hipóteses nulas, definidas de uma forma unidimensional, de como queremos analisar o efeito do uso da nossa ferramenta sobre as variáveis. Cada hipótese nula e sua alternativa são apresentadas da seguinte forma:

H1-0: Não há facilidade percebida significativa para o suporte ferramental / **H1-a:** O suporte ferramental é percebido como fácil de usar.

H2-0: Não há utilidade percebida significativa para o suporte ferramental / **H2-a:** O suporte ferramental é percebido como útil.

H3-0: Não há intenção de uso futuro para o suporte ferramental / **H3-a:** Os participantes têm intenção de uso do suporte ferramental, caso necessitem.

Variáveis selecionadas As variáveis dependentes são baseadas na perceção, avaliando as perceções dos participantes sobre o suporte ferramental (DVDTool) após assistirem alguns vídeos explicativos. Estas variáveis são baseados em TAM [17], um modelo teórico amplamente aplicado para analisar a aceitação do utilizador e o comportamento de uso de tecnologias de informação emergentes. Tem suporte empírico através de validações e replicações [37]. A eficácia percebida do método pode ser dividida em três variáveis dependentes subjetivas:

- **Facilidade de Uso Percebida (*Perceived Easy Of Use — PEOU*):** refere-se ao grau em que uma pessoa acredita que aprender e usar o nosso suporte ferramental não exigiria esforço significativo.
- **Utilidade Percebida (*Perceived Usefull — PU*):** refere-se ao grau em que uma pessoa acredita que ao usar o nosso método aumentaria o seu desempenho no trabalho dentro de um contexto organizacional.
- **Intenção de uso (*Intention To Use — ITU*):** refere-se à extensão em que uma pessoa pretende usar o suporte ferramental. Representa um julgamento percetivo da eficácia da ferramenta, ou seja, se é custo-efetivo e é comummente usado para prever a probabilidade de aceitação de um método ou ferramenta na prática.

Desenho do quase-experimento Ao iniciar a avaliação, os participantes preenchiam um formulário de dados demográficos (por exemplo, idade, género, formação académica) e logo em seguida eram disponibilizados quatro vídeos explicativos, um a um. Os quatro vídeos explicativos tinham como objetivo fazer com que os participantes (i) conhecessem o método DVD, (ii) conhecessem o suporte ferramental que criámos para o método DVD, (iii) conhecessem mais a fundo como uma arquitetura de negócio pode ser criada com a nossa ferramenta e (iv) como um modelo de requisitos também pode ser gerado com a ferramenta. Para evitar a possível fadiga dos participantes, a soma do tempo de execução dos quatro vídeos explicativos tem apenas 25 minutos. Após os vídeos explicativos serem visualizados, foi disponibilizado um questionário final. As três variáveis subjetivas foram medidas através das respostas desse questionário final usando uma escala de 5 pontos: 4 perguntas para PEOU, 5 para PU e 2 para ITU [1]. As perguntas foram formuladas usando o formato de declaração oposta. Assim, cada questão contém duas declarações contraditórias representando os valores máximo e mínimo possíveis (5 e 1), onde 3 é considerado uma perceção neutra. O valor agregado é a média aritmética das respostas às questões associadas a cada variável baseada na perceção. Utilizamos o teste Cronbach alpha [45] para avaliar a fiabilidade do questionário.

Procedimento de análise Escolhemos testes estatísticos pela sua robustez e sensibilidade para analisar os dados recolhidos. Decidimos aceitar uma probabilidade de 5% de Erro Tipo-I [69], rejeitando a hipótese nula quando ela é verdadeira. A normalidade da distribuição dos dados foi testada com o teste de Shapiro-Wilk [54]. Usámos o teste T para uma amostra quando os dados assumiram a distribuição normal, para verificar as nossas hipóteses. No entanto, quando os dados não assumiram a distribuição normal, aplicámos o teste de Wilcoxon [15].

7.1.2 Resultados

O uso de várias questões para medir o mesmo construtor requer o exame da fiabilidade do questionário. Usámos o Cronbach alpha e o resultado para o questionário foi de 0,886. Isso significa que o questionário é confiável (o Cronbach alpha é maior que 0,7 [45]). Após, analisámos a estatística descritiva das três variáveis, conforme mostra a Tabela 7.1.

Tabela 7.1: Estatística descritiva para PEOU, PU e ITU

Variável	Mínimo	Máximo	Mediana	Média	Desvio Padrão	valor-p
PEOU	3,25	5	4,5	4.36	0.55	0,140
PU	3	5	4,4	4,32	0.61	0,070
ITU	3	5	4,5	4.36	0.66	0,007

Estes dados, aparentemente, mostram que os participantes perceberam o suporte ferramental como sendo fácil de usar, útil e revelaram intenção de o usar no futuro, pois a

[1] O questionário pode ser encontrado em https://goo.gl/forms/7txK44boQpO4Rw7s2

média e a mediana de cada variável é superior a 3. Porém, devemos verificar este resultado averiguando as hipóteses. Para isso, aplicámos o teste de Shapiro-Wilk [54] para verificar a normalidade da distribuição para todas as variáveis (valor-p da coluna na Tabela 7.1). Os resultados mostram que a ITU não possui distribuição normal (ITU < 0,05) e PEOU e PU possuem distribuição normal (PU e PEOU > 0,05). Para finalizar, aplicámos o teste de Wilcoxon nos dados da ITU e o teste T de uma amostra em dados em PU e PEOU para confirmar se a média é significativamente maior que 3 (PEOU = 0,000, PU = 0,000 e ITU = 0,001). Como os resultados do teste de Wilcoxon e do teste T (de uma amostra) foram menores que 0,05, podemos aceitar que o suporte ferramental foi percebido como fácil de usar e útil, confirmando as hipóteses H1-a e H2-a. Além disso, também podemos aceitar que os participantes têm intenção de usar o suporte ferramental no futuro caso seja necessário, confirmando a hipótese H3-a.

7.2 Ameaças à validade

Certas questões podem ameaçar a validade deste quase-experimento. Sobre a **validade interna**, as principais ameaças são a experiência dos participantes e a compreensibilidade dos vídeos explicativos. A experiência dos participantes não foi um problema, pois nenhum deles teve experiências anteriores com o DVD, muito menos com o suporte ferramental criado para ele. A compreensão dos vídeos explicativos foi aliviada com a realização de um estudo piloto e com a produção dos mesmos em português (o idioma nativo dos participantes).

No que diz respeito à **validade externa**, a principal ameaça é a representatividade dos resultados. A representatividade dos resultados pode ser afetada pelo exemplo ilustrativo usado nos vídeos explicativos. Sobre o exemplo ilustrativo, atenuámos isso optando por escolher um exemplo ilustrativo simples.

As principais ameaças à **validade de construção** estão relacionadas com as medidas aplicadas na análise dos dados e com a fiabilidade do questionário. Mitigámos essas ameaças usando medidas que são geralmente usadas em outros experimentos de engenharia de software, e as variáveis são baseadas em TAM [17]. A fiabilidade do questionário foi testada com o teste de Cronbach alpha [45].

Finalmente, a ameaça da **validade de conclusão** é a validade dos testes estatísticos aplicados. Assim, escolhemos os testes mais comuns utilizados na engenharia de software baseada em evidências, devido à sua robustez e sensibilidade [42].

7.3 Conclusões

Avaliámos a nossa proposta de suporte ferramental para modelação de valor com um quase-experimento, para avaliar a sua facilidade de uso, a utilidade da ferramenta e a intenção de uso dos participantes. Os resultados mostram que a nossa proposta foi considerada fácil de usar e útil (confirmando as hipóteses H1-a e H2-a). Além disso,

também verificámos que os participantes têm intenção de usar o nosso suporte ferramental no futuro, caso tenham a necessidade de desenvolver sistemas baseados em valores de negócio (confirmando a hipótese H3-a).

CAPÍTULO 8

CONCLUSÕES

Esta dissertação inicia com a identificação da escassez de métodos que descrevam transações que envolvam valor económico por parte das empresas e que ofereçam modelos que permitam o alinhamento entre tecnologias de informação e valores de negócio. A maioria destes métodos não fornece uma ferramenta de suporte, que permita a construção e validação de modelos de valor, e o alinhamento entre a área de negócios e tecnologias de informação. Uma revisão sistemática da literatura publicada por Kundisch e John [39] em 2012, identifica 12 métodos orientados para especificar valor de negócio. De forma a completar este estudo, efetuámos um mapeamento sistemático que focasse o período 2011 a 2017. Da análise aos resultados obtidos, selecionámos o método DVD (*Dynamic Value Description*) como o método mais promissor para satisfazer os nossos objetivos, o desenvolvimento de uma ferramenta de suporte.

Para desenvolver a ferramenta de suporte ao método DVD contribuímos com uma linguagem de domínio específico (DSL), com recurso a tecnologias MDD (ou *model-driven development*). Para tal, consolidámos os conhecimentos sobre desenvolvimento orientado a modelos, e efetuámos uma análise comparativa de duas ferramentas de modelação de DSLs. Desta análise concluímos que a ferramenta *Sirius* era a mais adequada para prosseguir com o desenvolvimento da ferramenta. Posteriormente, discutimos os conceitos, técnicas e modelos que constituem o método DVD e que serão suportados pela ferramenta DVDTool. A ferramenta DVDTool integra três editores de modelos, um para o modelo DVD, outro para o modelo KAOS4Services (uma extensão do modelo KAOS que integra conceitos de valor), e outro para o modelo SoaML. A ferramenta fornece transformações entre modelos, através das quais conseguimos gerar um modelo KAOS4Services ou um modelo SoaML tendo como fonte um modelo DVD. Desta forma contribuímos com uma ferramenta que permite a construção de modelos de valor, e ainda estabelecer o alinhamento entre a área de negócios e tecnologias de informação.

Posteriormente, ilustrámos a utilização da ferramenta DVDTool com um caso de estudo, de modo a mostrar a interação que o utilizador tem com a ferramenta. Para tal, apresentámos um cenário a partir do qual construímos um modelo de valor DVD. Tendo por base o modelo construído, apresentámos o processo necessário para obter uma arquitetura de Serviços SoaML, e para obter e editar um modelo de requisitos KAOS4Services. Por fim, efetuámos uma avaliação à DVDTool através de um quase-experimento, com o objetivo de avaliar a sua facilidade de uso, utilidade, e intenção de uso no futuro por parte dos participantes. A análise aos resultados obtidos nesta avaliação revelaram que a nossa proposta foi considerada fácil de usar e útil, e os participantes revelaram intenção de uso no futuro.

Para trabalho futuro, vamos aperfeiçoar a interface gráfica da DVDTool na plataforma eclipse, fazendo com que esta fique mais simples e fácil de usar. Além disso, pretendemos desenvolver uma versão WEB da ferramenta de forma a torná-la disponível na nuvem (*cloud*), por isso, mais acessível para qualquer utilizador (seja ele da área de negócios ou TI), eliminando assim os típicos problemas de compatibilidade das versões *standalone*. Em relação à condução de estudos empíricos, pretendemos realizar uma nova avaliação da DVDTool com pessoas da área de negócios, pois acreditamos que este tipo de estudo vai contribuir para consolidar os resultados já obtidos. Pretendemos também comparar a DVDtool com a ferramenta de modelação do e3value, com o intuito de verificar a eficácia de ambas as ferramentas e oferecer mais dados empíricos para a comunidade académica. Finalmente, um problema mais complexo que exige bastante mais investigação está relacionado com evolução de maneira a garantir que os modelos se encontram sempre todos sincronizados. Por exemplo, a edição de um modelo gerado, causa problemas de sincronização com os modelos fonte. O nosso objetivo é explorar o uso de transformações inversas para resolver problemas de sincronização de modelos.

BIBLIOGRAFIA

[1] "A KAOS Tutorial (v1.0ed.)" Em: *RespectIT* (2007).

[2] ACM. *ACM Digital Library*. Online accessed: 22/Sep/2017. 2017. URL: https://dl.acm.org/.

[3] M. M. Al-Debei e D. Avison. "Developing a unified framework of the business model concept". Em: *European Journal of Information Systems* 19.3 (2010), 359–376.

[4] J. G.D.M. G. Amyot D.and Horko? "A lightweight grl pro?le for i* modeling". Em: *International Conference on Conceptual Modeling* (2009).

[5] B. Andersson, M. Bergholtz, A. Edirisuriya, T. Ilayperuma, P. Johannesson, J. Gordijn, B. Grégoire, M. Schmitt, E. Dubois, S. Abels et al. "Towards a reference ontology for business models". Em: *Conceptual Modeling (ER 2006)* (2006), pp. 482–496.

[6] A. Anton. "Goal-based requirements analysis". Em: *RE'96, IEEE* (1996), 136–144.

[7] Arcitura Education Inc. *ServiceOrientation.com: Service Definition*. Acedido em: 27/Aug/2018. URL: http://serviceorientation.com/soaglossary/service.

[8] A. Arsanjani. "Service-oriented modeling and architecture: How to identify, specify, and realize services for your soa." Em: (2004), pp. 1–15.

[9] V. R. Basili e H. D. Rombach. "The TAME project: Towards improvement-oriented software environments". Em: *IEEE Transactions on software engineering* 14.6 (1988), pp. 758–773.

[10] N. J.I.L. A. Borgida A.and Ernst. "Techne: A (nother) Requirements Modeling Language". Em: *ICSE'09* (2009).

[11] P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner e M. Khalil. "Lessons from applying the systematic literature review process within the software engineering domain". Em: 80.4 (mai. de 2007), pp. 571–583. ISSN: 0164-1212. DOI: 10.1016/j.jss.2006.07.009.

[12] T. Buzan e B. Buzan. "The Mind Map Book". Em: (1996).

[13] C. Casanave. "Enterprise service oriented architecture using the omg soaml standard, a model driven solutions." Em: (2012).

[14] B. Chung L.and Nixon e J. Yu E.S.K.and Mylopoulos. "Non-Functional Requirements in Software Engineering". Em: (1999).

[15] W. J. Conover. *Practical Nonparametric Statistics*. 3rd. Wiley, 2006. ISBN: 8126507756, 9788126507757.

[16] A. Dardenne, A. van Lamsweerde e S Fickas. "Goal-directed requirements acquisition". Em: *Science of Computer Programming 20 (1-2)* (1993), pp. 3–50.

[17] F. D. Davis. "Perceived usefulness, perceived ease of use, and user acceptance of information technology". Em: *MIS quarterly* (1989), pp. 319–340.

[18] S. Direct. *Science Direct Digital Library*. Online accessed: 22/Sep/2017. 2017. URL: http://sciencedirect.com.

[19] B. Elvesæter, A.-J. Berre e A. Sadovykh. "Specifying Services using the Service Oriented Architecture Modeling Language (SoaML)-A Baseline for Specification of Cloud-based Services." Em: *CLOSER*. 2011, pp. 276–285.

[20] H. E. Eriksson, M. Penker, B. Lyons e D. Fado. "UML toolkit." Em: (2004).

[21] T. Erl. "Service-oriented architecture: concepts, technology, and design". Em: (2005).

[22] J. B.I. K. F. Jouault F. Allilaire. "ATL: A model transformation tool". Em: *Science of Computer Programming* 72 (2008), pp. 31–39.

[23] E. Foundation. *Acceleo*. Acedido em: 10 maio 2017. URL: https://www.eclipse.org/acceleo/.

[24] E. Foundation. *ATL*. Acedido em: 10 maio 2017. URL: http://www.eclipse.org/atl/.

[25] E. Foundation. *Eclipse Modeling - M2T - Xpand*. Acedido em: 10 maio 2017. URL: https://www.eclipse.org/modeling/m2t/?project=xpand.

[26] E. Foundation. *Epsilon Transformation Language*. Acedido em: 10 maio 2017. URL: http://www.eclipse.org/epsilon/doc/etl/.

[27] E. Foundation. *EuGENia*. Acedido em: 8 maio 2017. URL: http://www.eclipse.org/epsilon/doc/eugenia/.

[28] E. Foundation. *Xtend - Modernized Java*. Acedido em: 10 maio 2017. URL: http://www.eclipse.org/xtend/.

[29] T. E. Foundation. *Sirius Overview.* Acedido em: 9 maio 2017. URL: http://www.eclipse.org/sirius/overview.html.

[30] J. Gordijn. "Value-based Requirements Engineering: Exploring Innovative e-Commerce Ideas". Tese de doutoramento. Amsterdam, Netherlands: Vrije Universiteit Amsterdam, 2002.

[31] J. Gordijn e H. Akkermans. "Designing and evaluating e-business models". Em: *IEEE Intelligent Systems* 16.4 (2001), pp. 11–17. DOI: http://dx.doi.org/10.1109/5254.941353.

[32] J. Gordijn, H. Akkermans e H. van Vliet. "Business Modelling Is Not Process Modelling". Em: *Conceptual Modeling for E-Business and the Web*. Ed. por S. W. Liddle, H. C. Mayr e B. Thalheim. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 40–51. ISBN: 978-3-540-45394-9.

[33] D. Granada. *Comparing tools to build graphical modeling editors*. Acedido em: 6 maio 2017. URL: http://modeling-languages.com/comparing-tools-build-graphical-modeling-editors/.

[34] IEEE. *IEEE Xplore Digital Library*. Online accessed: 22/Sep/2017. 2017. URL: https://ieeexplore.ieee.org.

[35] N. M. Josuttis. *SOA in practice: the art of distributed system design*. O'Reilly Media, Inc., 2007.

[36] T. Kelly e R. Weaver. "The goal structuring notation–a safety argument notation". Em: *Dependable systems and networks workshop on assurance cases* (2004).

[37] W. R. King e J. He. "A meta-analysis of the technology acceptance model". Em: *Information & management* 43.6 (2006), pp. 740–755.

[38] B. A. Kitchenham. *Guidelines for performing Systematic Literature Reviews in Software Engineering*. Rel. téc. Durham, UK: Department of Computer, Science University of Durham, jul. de 2007.

[39] D. Kundisch e T. John. "Business model representation incorporating real options: An extension of e3-value". Em: *Proceedings of the Annual Hawaii International Conference on System Sciences* (2011), pp. 4456–4465. DOI: 10.1109/HICSS.2012.139.

[40] T. Kühne. "Matters of (Meta-)Modeling". Em: *Software and System Modeling 5.4* (2006), pp. 369–385.

[41] P. Loucopoulos, V. Kavakli, N. Prekas, G. Rolland C.and Grosz e S. Nurcan. "Using the ekd approach: the modelling component". Em: (1997).

[42] K. Maxwell. *Applied statistics for software managers*. Prentice Hall, 2002.

[43] S. J. Mellor, K. Scott, A. Uhl e D. Weise. *MDA distilled: principles of model-driven architecture*. Addison-Wesley Professional, 2004.

[44] N. Nayak, M Linehan, A. Nigam, D. Marston, J.-J. Jeng, F. Y. Wu, D. Boullery, L. White, P. Nandi e J. L. Sanz. "Core business architecture for a service-oriented enterprise". Em: *IBM Systems Journal* 46.4 (2007), pp. 723–742.

[45] J. C. Nunnally e I. H. Bernstein. "Psychometric theory". Em: (1978).

[46] Obeo. Acedido em: 9 maio 2017. URL: http://www.siriuscon.org/.

[47] "OMG: Business Motivation Model (BMM)". Em: (2010).

[48] "OMG: Documents Associated With Service Oriented Architecture Modeling Language (SoaML), Version 1.0.1". Em: (). URL: http://www.omg.org/spec/SoaML/1.0.1/PDF.

[49] A. G. Pateli e G. M. Giaglis. "A research framework for analysing eBusiness models". Em: *European Journal of Information Systems* 13.4 (2004), 302–314.

[50] A. G. Pateli e G. M. Giaglis. "Technology innovation-induced business model change: A contingency approach, Journal of Organizational Change Management". Em: (2005), pp. 167–183.

[51] K. Petersen, S. Vakkalanka e L. Kuzniarz. "Guidelines for conducting systematic mapping studies in software engineering: An update". Em: *Information and Software Technology* 64 (2015), pp. 1–18.

[52] V. Pijpers e J. Gordijn. "Considering software quality requirements as networked business quality requirements". Em: *HEC University, Lausanne Tech. Rep* (2008).

[53] A. Rasiwasia. "Meta Model for Business Model Design: Designing a Meta model for E3 value model based on MOF". Tese de mestrado. Stockholm, Sweden, 2013.

[54] S. Shaphiro e M. Wilk. "An analysis of variance test for normality". Em: *Biometrika* 52.3 (1965), pp. 591–611.

[55] I. Solheim e K. Stølen. "Technology research explained". Em: (2007).

[56] E. Souza e A. Moreira. "Aligning business models with requirements models". Em: *European, Mediterranean and Middle Eastern Conference on Information Systems* (2017).

[57] E. Souza, S. Abrahão, A. Moreira, J. Araújo e E. Insfran. "Comparing Value-Driven Methods: an Experiment Design." Em: *HuFaMo@ MoDELS, Saint Malo, France*. 2016, pp. 19–26.

[58] E. Souza, A. Moreira e C. De Faveri. "An approach to align business and IT perspectives during the SOA services identification". Em: *17th International Conference on Computational Science and Its Applications* (2017).

[59] E. Souza, S. Abrahão, A. Moreira, J. Araujo e E. Insfran. "Evaluating the efficacy of value-driven methods: a controlled experiment". Em: (2017).

[60] E. Souza, A. Moreira, J. Araújo, S. Abrahão, E. Insfran e D. S. da Silveira. "Comparing business value modeling methods: A family of experiments". Em: *Information and Software Technology* (2018).

[61] G. H. Travassos, D. Gurov e E. Amaral. *Introdução à engenharia de software experimental*. UFRJ, 2002.

[62] M. Valipour, B. AmirZafari, Maleki e N. K.N. Daneshpour. "A brief survey of software architecture concepts and service oriented architecture". Em: (2009).

[63] A. Van Lamsweerde. "Goal-oriented requirements engineering: a guided tour". Em: *Requirements Engineering conference - RE'01* (2001).

[64] I. Vasiljevic, G. Milosavljevic, I. Dejanovic e M. Filipovic. "Comparison of Grafical DSL Editors." Em: *6th PSU-UNS International Conference on Engineering and Technology* (2013).

[65] M. Voelter, S. Benz, C. Dietrich, B. Engelmann, M. Helander, L. C. L. Kats, E. Visser e G. Wachsmuth. "DSL Engineering - Designing, Implementing and Using Domain-Specific Languages". Em: (2013), pp. 1–558.

[66] F. Wanderley e J. Araujo. "Generanting goal-oriented models from creative requirements using model driven engineering". Em: *Model-Driven Requirements Engineering (MoDre), International Workshop on. IEEE* (2013).

[67] H. Weigand, P. Johannesson, B. Andersson e M. Bergholtz. "Value-based service modeling and design: Toward a unified view of services". Em: (2009), pp. 410–424.

[68] C. Wohlin. "Writing for synthesis of evidence in empirical software engineering". Em: *the 8th ACM/IEEE International Symposium*. New York, New York, USA: ACM Press, 2014, pp. 1–4. ISBN: 9781450327749. DOI: 10.1145/2652524.2652559.

[69] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell e A. Wesslén. *Experimentation in software engineering*. Springer-Verlag Berlin Heidelberg, 2012. ISBN: 978-3-642-29043-5. DOI: 10.1007/978-3-642-29044-2. URL: http://link.springer.com/10.1007/978-3-642-29044-2.

[70] P. M.N.M. J. Yu E.and Giorgini. "Social modeling for requirements engineering". Em: *Mit Press* (2011).

[71] C. Zott e R. Amit. "The business model as the engine of network-based strategies. In The network challenge: Strategy, profit, and risk in an interlinked world". Em: *Wharton School Publishing* (2009), pp. 259–275.

[72] C. Zott, R. Amit e L. Massa. "The business model: recent developments and future research". Em: *Journal of management* 37.4 (2011), pp. 1019–1042.