Universidade Nova de Lisboa
Faculdade de Ciências e Tecnologia
Departamento de Informática

Uma Linguagem de Domínio Específico para a Framework i*

Carlos Miguel Nunes

Dissertação apresentada para obtenção do Grau de Mestre em Informática, pela Universidade Nova de Lisboa, Faculdade de Ciências e Tecnologia

Lisboa
(2009)

Esta dissertação foi preparada com a supervisão dos professores João Araújo e Vasco Amaral da Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa.

Nº do aluno: 26861
Nome: Carlos Miguel Marques Martins Simões Nunes

Título da dissertação:
Uma Linguagem de Domínio Específico para a Framework i*

Palavras-Chave:

- Framework i*
- Linguagens de Domínio Especifico
- Metamodelação
- Engenharia de Requisitos

Keywords:

- i* Framework
- Domain Specific Language
- Metamodeling
- Requirements Engineering

Resumo

A framework i* é uma framework orientada aos objectivos muito popular na comunidade de engenharia de requisitos, tendo começado a ser aplicada na indústria. É uma abordagem sistemática usada para descobrir e estruturar requisitos a um nível organizacional, onde requisitos não funcionais e as suas relações são especificados. No entanto ainda há muito por fazer em termos de investigação sobre este framework. Por exemplo, a definição e a relação dos seus elementos através de metamodelos continua a conter algumas ambiguidades, o que leva a que as ferramentas existentes não consigam implementar de modo eficaz todas as potencialidades disponibilizadas por esta linguagem permitindo a criação de modelos inconsistentes, assim como os modelos criados nessas ferramentas continuam a apresentar sérios problemas de gestão de escalabilidade dos mesmos. Assim um dos objectivos desta tese consiste no estudo do metamodelo do i* e das ferramentas que o implementam, com o objectivo de identificar as suas limitações. O estudo desse metamodelo vai possibilitar a resolução das referidas ambiguidades e introduzir mecanismos que permitam gerir a escalabilidade dos modelos criados, sendo a escalabilidade um problema típico quando se pretende modelar sistemas reais com uma complexidade aceitável. Para que isto se torne possível, uma Linguagem de Domínio Especifico (LDE) vai ser especificada.

Uma LDE tem como propósito especificar e modelar conceitos num determinado domínio, tendo várias vantagens em relação às linguagens de domínio geral, tais como permitir expressar a solução de um problema na linguagem desejada e ao nível de abstracção desejado. Para se poder criar uma LDE com sucesso, normalmente é necessário começar por especificar a sua sintaxe recorrendo a um metamodelo que será dado como input para os workbenches da linguagem que vão gerar o editor correspondente. Com um editor apropriado para a linguagem podemos especificar modelos com a notação proposta.

Esta tese pretende então desenhar e desenvolver uma LDE para a framework i*, com o propósito de gerir a complexidade e escalabilidade dos seus modelos concretos, introduzindo para esse efeito algumas inovações nesse metamodelo tais como mecanismos que ajudem a gerir a escalabilidade dos modelos produzidos.

Abstract

The i* framework proposes a goal-oriented analysis method for requirements engineering. It is a systematic approach to discover and structure requirements at organizational level where non-functional requirements and their relations are specified. But there is still many investigation work to be done about this framework. For example, the definition and relation of it's elements in metamodels continues to have some ambiguaties, leading this way to the existence of several tools that aren't able to implement the totl functionalities that this framework has and allowing the creation of inconcistent models with scability problems. So one of this thesis objectives is to study the i* metamodel and several tools tha implement this framework, with the purpose to identify it's limitations. This study will allow to solve those ambiguities and introduce ways to solve the models complexity. For that to happen a DSL (Domain Specific Language) will be created.

A Domain Specific Language (DSL) has the purpose to specify and model concepts in some domain, having several advantages in relation to general purpose languages, such as allow expressing a solution in the desired language and at the desired abstraction level. In order to create such a DSL, normally it is necessary to start by specifying its syntax by means of a metamodel to be given as input to the language workbenches that generate the corresponding editors for it. With a proper editor for the language we can specify models with the proposed notation.

This thesis presents a DSL for the i* framework, with the purpose to handle complexity and scalability of its concrete models by introducing some inovations in the i* framework metamodel like mechanisms that will help to manage the models scalability.

Índice

1. Introdução 1

1.1 Contexto 1

1.2 Motivação 2

1.3 Objectivos da Tese 3

1.4 Principais Contribuições Previstas 4

1.5 Organização do documento 5

2. Análise de Requisitos Orientadas a Objectivos 7

2.1 Modelos em Análise 7

2.2 O Modelo SD 8

2.3 O Modelo SR 11

2.4 Frameworks relacionados com i* 13

2.4.1 Tropos 13

2.4.2 NFR 14

2.4.3 GRL 14

2.5 Vantagens do i* 15

2.6 Limitações do i* 15

2.7 Ferramentas existentes e seus atributos 16

2.8 Áreas de Aplicação 20

2.9 Framework KAOS 21

2.10 Sumário 23

3. Linguagens de Domínio Especifico 24

3.1 Ferramentas para Linguagens de Domínio Especifico 26

3.1.1 GME 26
3.1.2 DSL Tools 27
3.1.3 EMF/GMF 27
3.2 Sumário 29
4. Trabalho realizado 30
4.1 Metamodelação da Framework i* 30
4.1.1 Metamodelo do Modelo SD 31
4.1.2 Metamodelo do Modelo SR 32
4.2 Especificação da LDE para a framework i* 35
4.2.1 Criação do Modelo Ecore 35
4.2.2 GMFGraph e GMFTool 38
4.2.3 GMFMap 39
4.2.4 Editor da LDE 41
4.3 Sumário 42
5. Avaliação da LDE 43
5.1 Caso de Estudo 43
5.2 Âmbito da Avaliação 54
5.3 Hipótese que se Pretende Testar 55
5.4 Testes 56
5.5 Questionário 56
5.6 Avaliação dos Dados Recolhidos 57
5.6.1 Dados Gerais e suas Conclusões 57
5.6.2 Análise Individual por Questão 58
5.6.3 Opiniões dos Utilizadores 62
5.6.4 Discussão dos Resultados 63
6. Conclusões 64

6.1 Trabalhos futuros 65

7. Bibliografia 66

Anexo 1: Questionário 69

Anexo 2: Manual de Utilizador 77

Índice de Figuras

Figura 2.1.....19

Figura 2.2.....21

Figura 2.3.....32

Figura 4.1.....41

Figura 4.2.....43

Figura 4.3.....47

Figura 4.4.....48

Figura 4.5.....49

Figura 4.6.....50

Figura 4.7.....51

Figura 4.8.....52

Figura 5.1.....56

Figura 5.2.....57

Figura 5.3.....60

Figura 5.4.....61

Figura 5.5.....63

Figura 5.6.....69

Figura 5.7.....70

Figura 5.8.....71

Figura 5.9.....72

Figura 5.10.....73

Índice de Tabelas

Tabela 2.1.....28

Tabela 2.2.....29

Tabela 2.3.....29

1. Introdução

1.1 Contexto

Em engenharia de software é muito importante identificar os requisitos que realmente reflectem as necessidades dos utilizadores. Caso esses requisitos sejam mal identificados ou ignorados, o sucesso do sistema em desenvolvimento pode ficar seriamente comprometido.
Um grande obstáculo na identificação correcta dos requisitos reside no facto de se ter uma compreensão pouco profunda sobre o domínio da aplicação em análise.

Ainda durante uma fase inicial da engenharia de requisitos é muitas vezes necessário ajudar os utilizadores a identificar quais as várias alternativas e quais dessas alternativas vão melhor servir à solução que se pretende alcançar e com isso conseguir formalizar ideias abstractas.

A framework em análise nesta tese é referente à framework criada por Eric Yu chamada de i* [10,11], tendo como principal objectivo de descobrir e estruturar requisitos a um nível organizacional. Os sistemas e os seus ambientes são especificados em termos de relações intencionais entre actores estratégicos. Os actores no sistema são intencionais uma vez que têm desejos e necessidades, e são estratégicos uma vez que se preocupam com oportunidades e vulnerabilidades no sistema. Cada actor é responsável por executar determinadas tarefas e fornecer determinados recursos estando muitas vezes dependente de recursos ou tarefas realizadas por outros actores para a realização das suas tarefas e cumprimento dos seus objectivos.

Para conseguir atingir o objectivo descrito acima a framework propõe dois modelos, o Modelo de Dependência Estratégica (do Inglês, "Strategic Dependency Model" e abreviado como SD) que será um dos modelos em estudo, tendo como propósito descrever uma configuração de relações dependentes entre vários actores organizacionais. O segundo modelo utilizado, o Modelo de Razão Estratégica (do Inglês, "Strategic Rationale Model" e abreviado como SR), tem por objectivo descrever as relações entre os vários componentes internos constituintes de um determinado actor, de modo a poder especificar-se as várias possibilidades em termos de configurações que esse actor poderá adoptar.

Assim pretende-se que esta framework dê algumas respostas ao nível de modelação a questões sobre o comportamento do sistema que se está a modelar. Ou seja faça questões de "como" e "porquê" relativamente aos actores e respectivas relações entre actores, consiga dar resposta em termos de modelação às perguntas feitas e às necessidades encontradas, consiga justificar convenientemente as soluções encontradas para a resolução das várias necessidades apresentadas e consiga arranjar alternativas a soluções já apresentadas [10,11].

1.2 Motivação

Sendo o i* uma linguagem muito conhecida e utilizada na comunidade de engenharia de requisitos, tem havido muito trabalho realizado sobre esta framework [10,11], desde a criação de várias ferramentas que implementam a framework, passando pela adaptação da framework a várias áreas distintas, até à tentativa de alterar e melhorar a respectiva framework. No entanto esta framework continua a ser uma linguagem que não está especificada rigorosamente, logo, ao utilizá-la, isso vai ser repercutido no rigor dos modelos criados, que devido a ambiguidades existentes poderão não estar totalmente correctos podendo ainda ser interpretados de diferentes maneiras.

Além disso, demonstra incapacidade desta de endereçar efectivamente a escalabilidade dos modelos criados tornando-se bastante complexos para serem compreendidos e analisados, especialmente se esta framework for aplicada a um sistema real que contenha uma complexidade suficientemente grande para evidenciar esse problema. Devido a todas as razões apresentadas acima as implementações existentes desta linguagem acabam por não serem consistentes e não oferecem mecanismos suficientes para lidar com os problemas mencionados e mesmo que ofereçam alguns desses mecanismos, isso não é reflectido no seu metamodelo, sendo um metamodelo uma descrição formal de uma linguagem ou o modelo da própria linguagem. Isto quer dizer que a sintaxe da linguagem não está especificada suficientemente o que leva a que os editores existentes permitam construções sintácticas incorrectas e inconsistentes nos modelos criados.

Outra grande motivação presente para a elaboração desta tese, prende-se com a complexidade dos modelos criados ao usar-se esta framework. Uma vez que não existe qualquer mecanismo de controlo de escalabilidade específico presente na framework, à medida que os modelos crescem vão ficando cada vez mais imperceptíveis e difíceis de

analisar, o que levanta grandes problemas aos Engenheiros de Requisitos. Deste modo esta tese propõe uma introdução na framework i* de mecanismos que possam gerir e assim diminuir a escalabilidade dos modelos construídos, de modo a que a complexidade dos modelos seja menor e a sua análise por parte dos Engenheiros de Requisitos seja facilitada.

Devido a todas as razões enumeradas uma grande motivação desta tese prende-se com o estudo da framework i*, assim como o estudo do seu metamodelo de modo a que se possa criar um metamodelo específico que possa ser usado e validado. Pretende-se ainda introduzir nesse metamodelo novas características que permitam gerir a escalabilidade dos modelos criados.

Essa validação pode ser conseguida construindo uma Linguagem de Domínio Específico (LDE) usando como base o metamodelo construído. Assim ao validar-se o metamodelo recorrendo à construção de uma LDE, consegue-se para além da sua validação, verificar se este é rigoroso e consistente, de modo a que não existam ambiguidades na LDE que se pretende criar.

1.3 Objectivos da Tese

Nesta tese pretende-se fazer um estudo aprofundado da framework i* [10,11], quais as suas componentes e como se relacionam entre si, com o objectivo de definir rigorosamente a linguagem em termos de metamodelo. Ir-se-á realizar um estudo de todas as ferramentas que implementam esta framework, verificando quais as suas funcionalidades, objectivos e falhas.

Para se construir uma ferramenta para implementar esta framework e que consiga gerir a complexidade dos seus modelos, vai ser usada uma Linguagem de Domínio Especifico (LDE), ou Domain Specific Language (DSL) [16,20], em inglês. O objectivo de uma LDE é especificar e modelar conceitos de um determinado domínio [16,20], em que neste caso concreto o domínio é a framework do i*. Logo o propósito desta ferramenta é criar um editor gráfico para os modelos baseados no i* e contribuir com algumas funcionalidades novas, tais como mecanismos para gerir a complexidade dos modelos.

Para criar uma LDE com sucesso é necessário especificar um metamodelo representativo da linguagem que represente de um modo rigoroso a sintaxe da própria linguagem. Esse metamodelo será criado com base nos metamodelos existentes da framework i* [1,4,6] e adaptado para uso na LDE, permitindo assim a criação do editor

gráfico do modelo e contribuir com algumas inovações, tais como mecanismos que permitam gerir a complexidade visual dos modelos.

Portanto com base no estudo tanto da framework como das ferramentas de implementação, será construída uma LDE desenhando um metamodelo específico da framework i*. Uma vez que a LDE tem como base um metamodelo do domínio que se pretende implementar é fundamental ter um conhecimento aprofundado da framework i*, de modo a conseguir-se criar um metamodelo completo e correcto o suficiente para se conseguir criar uma ferramenta fiável que consiga implementar correctamente a framework em estudo e consiga adicionar algumas novas funcionalidades em comparação com as outras ferramentas estudadas. Estas novas funcionalidades como já foi referido acima prendem-se principalmente com a introdução de mecanismos, que permitam a gestão da complexidade e escalabilidade dos modelos construídos usando o metamodelo criado especialmente para a resolução destes problemas apresentados pela framework i*.

Em resumo, os objectivos desta tese prendem-se com o estudo da framework i*, o estudo de outras linguagens orientadas a objectivos, assim como o estudo das LDEs, das ferramentas que permitem a sua criação e aplicação destas à framework em causa, e ter um cuidado especial com a introdução de novos mecanismos que permitam gerir a escalabilidade dos modelos criados através da framework i*.

1.4 Principais Contribuições Previstas

As principais contribuições que se esperam atingir com esta tese consistem em, ter um maior conhecimento da framework estudada, assim como esta é constituída de modo a que se consiga formalizar a sua sintaxe criando metamodelos que a representem correctamente. Com esses metamodelos pretende-se derivar um editor de LDE que implemente todas as possibilidades de modelação que esta framework disponibiliza e que adicione novos mecanismos de gestão da complexidade e escalabilidade dos modelos criados.

O propósito de se querer criar uma LDE para esta framework deve-se ao facto de após se ter estudado as principais ferramentas que a implementam, se querer introduzir algumas novas funcionalidades, tais como a possibilidade de se poder encapsular elementos da framework melhorando assim a legibilidade e compreensão dos modelos construídos. Outra funcionalidade consiste na criação de agentes expansíveis dinamicamente, havendo assim a

possibilidade de se passar dinamicamente para o modelo SD ou SR, através da expansibilidade ou retracção dos vários agentes no modelo, tal como proposto em [2,3]. Espera-se com estas funcionalidades melhorar a framework, tanto na sua utilização como ao nível do seu aspecto gráfico e ainda permitir uma gestão eficaz da complexidade dos modelos criados.

1.5 Organização do documento

A organização desta tese é a seguinte:

- O Capítulo 2 mostra em pormenor os modelos SD e SR da framework em estudo, os seus pontos mais positivos e menos positivos, uma análise comparativa de várias ferramentas que implementam a framework, uma explicação sobre as LDEs e quais as ferramentas que as implementam.

- O Capítulo 3 mostra uma explicação sobre as LDEs e qual a abordagem a ser adotada e as ferramentas para as implementar.

- O Capítulo 4 mostra a especificação em pormenor da LDE proposta, sendo esta analisada passo a passo, desde a criação do modelo Ecore (metamodelo da linguagem para a plataforma eclipse) até ao desenvolvimento do Editor, será também mostrado nesta secção um caso de estudo para ilustração do editor desenvolvido para a LDE construída, sendo esta comparada com outra ferramenta de nome OME, que também implementa a framework em análise. Finalmente serão mostrados qual o trabalho a desenvolver no futuro.

- O Capítulo 5 mostra os métodos usados para avaliar a LDE criada, assim como os resultados tirados dessa mesma avaliação.

- O Capítulo 6 mostra a análise dos dados que foram obtidos no capítulo 5 e apresenta as conclusões para cada caso em estudo.

- O Capítulo 7 mostra as conclusões a que se chegou nesta tese e que trabalho futuro é que se poderá desenvolver usando esta tese como ponto de partida.

- O Anexo 1 mostra o questionário que foi feito para se obter os dados para realizar a análise e comparação da ferramenta desenvolvida em relação às ferramentas existentes.

- O Anexo 2 mostra o manual de utilizador criado para a ferramenta desenvolvida.

2. Análise de Requisitos Orientadas a Objectivos

Neste capítulo serão analisadas duas frameworks orientadas a objectivos onde a ênfase vai ser dada ao framework i*.

Em relação ao i* será feito um estudo detalhado sobre a sua metodologia e respectivos componentes. Através desse estudo e do estudo da descrição formal da sua sintaxe será criado um metamodelo refinado com o propósito de tentar resolver algumas ambiguidades presentes nesta framework, sendo esse metamodelo validado através da construção de uma LDE.

Serão ainda estudadas outras frameworks que usam o i* como base, nomeadamente o Tropos e GRL, que ferramentas implementam estas frameworks e quais as suas propriedades e limitações. Finalmente, vai-se estudar quais as vantagens e limitações da própria framework i*.

Relativamente à framework KAOS apenas será estudada de modo global, sem grande detalhe visto não ser o objectivo desta tese estudar essa framework mas sim identificar quais os pontos comuns entre a framework KAOS e i* de modo a que no futuro se possa realizar transformações que permitam a passagem de uma linguagem para a outra.

2.1 Modelos em Análise

A framework i* [10,11] tem como principal objectivo articular uma noção de distributividade intencional, ou seja construir modelos em que os actores se relacionem entre si através de relações entre os actores e elementos constituintes do sistema. Para conseguir atingir este objectivo são usados dois modelos, sendo o nome do primeiro modelo "Strategic Dependency Model" ou SD e tem como propósito descrever uma configuração de relações dependentes entre vários actores organizacionais. Relativamente ao segundo modelo utilizado que dá pelo nome de "Strategic Rationale Model" ou SR, que tem por objectivo descrever as relações internas entre os vários componentes internos constituintes de um determinado actor, de modo a poder especificar-se as várias possibilidades em termos de configurações que esse actor poderá adoptar.

Através dos modelos SD e SR que esta framework disponibiliza pretende-se fazer questões relativamente aos actores e respectivas relações entre actores. Usando essas questões pretende-se dar resposta em termos de modelação às perguntas feitas e às necessidades encontradas. Pretende-se ainda com estes modelos conseguir justificar convenientemente as soluções encontradas, para a resolução das várias necessidades apresentadas e que seja possível arranjar alternativas a soluções já apresentadas.

De seguida será dada uma pequena explicação relativamente a estes dois modelos.

2.2 O Modelo SD

No modelo SD o objectivo centra-se em fazer uma modelação usando as interacções dos vários actores constituintes do processo. Pretende-se assim conseguir perceber quais as motivações e objectivos que levam à realização de uma determinada funcionalidade dentro de um processo através dos actores intervenientes nessa funcionalidade. Com esta abordagem, o modelo SD oferece um método de análise que permite aprofundar mais os pormenores de um determinado processo em relação aos modelos de análise convencionais, que não suportam uma modelação intencional. Sendo assim este modelo ajuda na identificação de Stakeholders, no descobrimento de vulnerabilidades e oportunidades em relação ao sistema em análise, e no reconhecimento de relações entre os actores participantes auxiliando assim na obtenção de soluções para as vulnerabilidades detectadas na análise do sistema.

Um modelo SD consiste num conjunto de actores, objectivos, tarefas, recursos, *softgoals* e dependências entre estes elementos aqui referidos. Não sendo obrigatório a existência de todos estes elementos num modelo SD, assim como pode haver mais de um elemento do mesmo tipo como mostrado na figura 2.1. Esta figura mostra uma parte do modelo SD para o caso de estudo Health Watcher, um sistema sobre informações de saúde e sanitárias, e como se pode ver existem dois actores, o "User" que representa os utilizadores do sistema e o "Health Watcher" que representa o sistema em si, nesta figura está-se a modelar as funcionalidades de apresentar uma queixa e de consultar uma queixa, para esse efeito usam-se vários elementos nas dependências entre os actores para representar esses objectivos, assim como para representar os recursos e as tarefas que são necessárias executar para atingir esses objectivos.

Um actor é um elemento que tem como objectivo realizar tarefas de modo a atingir os seus objectivos. Quando existe uma dependência entre dois actores, ao actor que está dependente do

outro para levar a cabo alguma acção chama-se "*depender*", ao actor que está a ser dependido chama-se "*dependee*" e quanto ao elemento no centro da dependência chama-se "*dependum*", não podendo este elemento ser um actor, podendo no entanto ser outro elemento qualquer. Um actor ao depender de outro actor através de uma dependência acaba por ficar vulnerável, pois caso o actor do qual está a depender por alguma razão não conseguir devolver os dados necessários que o actor precisa, não conseguirá realizar as suas tarefas e atingir os seus objectivos, assim são detectadas vulnerabilidades no sistema.

Os actores ainda têm uma relação especial chamada "*IsA*" que permite a especificação de um determinado actor, herdando este as características retidas pelo actor mais geral do qual proveio dando assim a noção de hereditariedade entre actores.

Um objectivo é um elemento que tem o propósito de representar um estado do sistema, que um determinado actor pretende atingir. Numa dependência envolvendo um objectivo o actor "*depender*" depende do actor "*dependee*" para assegurar o estado do sistema representado pelo objectivo, tendo a liberdade de definir como deve atingir esse estado. O actor "*depender*" pode assumir que esse objectivo está disponível pelo actor "*dependee*" ficando, no entanto dependente do facto se esse objectivo é ou não disponibilizado.

Uma relação que envolva um objectivo tem de ter sempre um actor que precisa desse objectivo e um actor que disponibiliza esse objectivo como mostrado na figura 2.1.

Uma tarefa é um elemento que tem como propósito representar uma determinada acção que vai ser executada por um actor. Numa dependência envolvendo uma tarefa o actor "*depender*" depende do actor "*dependee*" para assegurar que uma determinada actividade é executada, sendo especificado como essa actividade é executada mas não o porquê de ser executada, sendo esse ponto apenas definido no modelo SR. O actor "dependee" pode não realizar a tarefa quer por impossibilidade, quer por opção, caso tenha algo mais prioritário para resolver. Deste modo o actor "depender" fica vulnerável uma vez que existe a possibilidade da tarefa que necessita não chegar a ser realizada. Uma relação que envolva uma tarefa tem de ter sempre um actor que precisa dessa tarefa e um actor que disponibiliza essa tarefa como mostrado na figura 2.1.

Um recurso é um elemento que tem como propósito a representação de uma determinada entidade física que esteja disponível no sistema. Numa dependência envolvendo um recurso o actor "*depender*" depende do actor "*dependee*" para assegurar a disponibilidade desse mesmo recurso, ganhando assim a possibilidade de usar este recurso conforme as suas necessidades, ficando no entanto vulnerável caso o actor "*dependee*" não o disponibilize.

Uma relação que envolva um recurso tem de ter sempre um actor que precisa desse recurso e um actor que disponibiliza esse recurso como mostrado na figura 2.1.

Um *softgoal* é um elemento que tem como propósito representar um meio que serve para atingir um determinado objectivo. Estão relacionados com os requisitos não funcionais (e.g. segurança, desempenho, fiabilidade). Numa dependência envolvendo um *softgoal* o actor "*depender*" depende do actor "*dependee*" para assegurar que uma determinada tarefa é feita de modo a assegurar a realização do *softgoal*, não sendo no entanto especifico que tarefas se devem realizar ou por que ordem, podendo sempre tal como foi dito anteriormente o actor "*depender*" ficar vulnerável caso o actor "*dependee*" não realize as tarefas necessárias para assegurar o *softgoal*. Uma relação que envolva um *softgoal* tem de ter sempre um actor que precisa desse *softgoal* e um actor que disponibiliza esse *softgoal* como mostrado na figura 2.1.

O modelo SD para além dos elementos anteriormente apresentados e respectivas dependências entre esses elementos, oferece a possibilidade de estabelecer vários níveis de força numa determinada dependência, níveis esses que demonstram qual a importância que existe na realização dos objectivos ou tarefas por parte do actor "*dependee*" de modo a conseguir disponibilizar esses resultados ao actor "*depender*". Os três níveis de força numa dependência da menos importante para a mais importante são a dependência aberta, a dependência comprometida e a dependência crítica.

Numa dependência aberta caso o actor "*dependee*" não disponibilize os resultados pretendidos pelo actor "*depender*" na dependência entre eles, os objectivos do "*depender*" vão ser afectados não sendo no entanto algo de consequências graves.

Numa dependência comprometida caso o actor "*dependee*" não disponibilize os resultados pretendidos pelo actor "*depender*" na dependência entre eles, os objectivos do "*depender*" vão ser comprometidos havendo já alguma gravidade na não realização desses mesmos objectivos, pois podem haver outros actores dependentes da obtenção destes resultados.

Numa dependência crítica caso o actor "*dependee*" não disponibilize os resultados pretendidos pelo actor "*depender*" na dependência entre eles, os objectivos do "*depender*" vão ser comprometidos havendo muita gravidade na não realização desses mesmos objectivos, pois muitos outros objectivos e actores que estavam dependentes destes resultados não poderão ser realizados comprometendo assim todo o sistema.

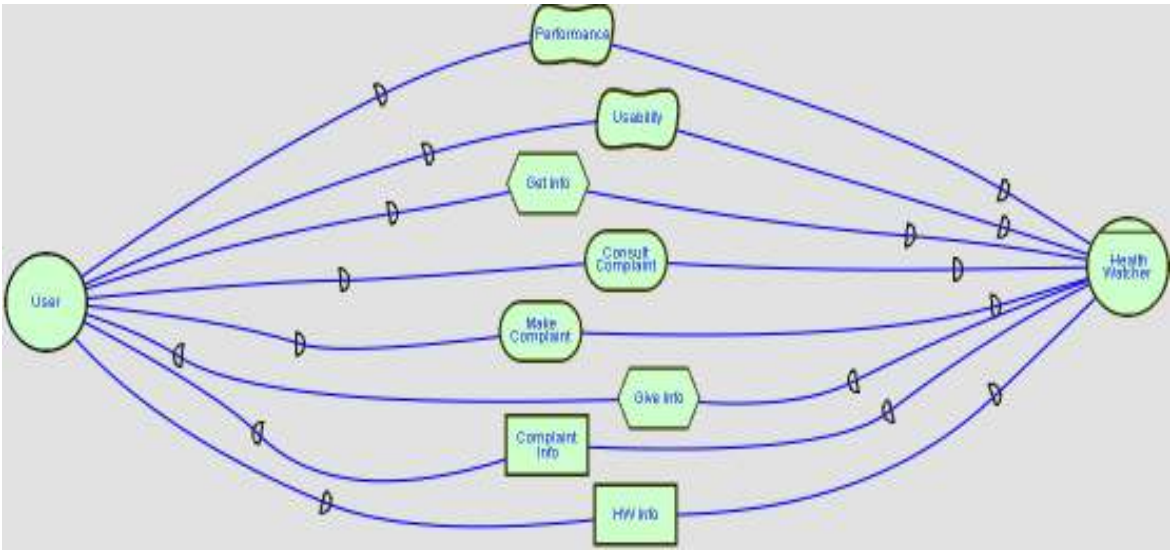

Figura 2.1: Parte do modelo SD para o sistema Health Watcher

2.3 O Modelo SR

Um modelo SR contêm todas as características contempladas no modelo SD mais a possibilidade de se expandir actores e fazer a sua modelação interna, factor que não era permitido no modelo SD. Ao expandir-se um actor há a possibilidade de se ter um conjunto de objectivos, tarefas, recursos, *softgoal* e ligações entre estes elementos, ligações estas que podem ser de decomposição de um determinado elemento em vários sub-elementos constituintes desse elemento, ou em ligações para atingir um determinado fim de modo a especificar como se atinge um determinado objectivo ou se realiza uma determinada tarefa usando um conjunto de elementos. Um actor expandido pode ter um número indeterminado de elementos não sendo obrigatório todos os elementos estarem presentes, podendo mesmo não existir qualquer elemento e pode também conter um número indeterminado de ligações de decomposição e ligações para atingir fins, não sendo também obrigatório a existência destas tal como mostrado na figura 2.2.

Uma ligação de decomposição é um tipo de ligação exclusivo das tarefas não sendo assim possível decompor outro elemento com este tipo de ligação sem ser uma tarefa. Uma tarefa pode-se decompor em objectivos, tarefas, recursos ou *softgoal*, havendo assim um tipo de ligação de decomposição para cada um destes casos. Uma tarefa pode conter várias ligações de decomposição independentemente do tipo da ligação de decomposição ou pode não ter nenhuma ligação de decomposição tal com é mostrado na figura 2.2. Esta figura mostra um

exemplo de um actor expandido. Neste exemplo o actor expandido foi o actor "User" e no seu interior pode-se visualizar vários elementos internos assim como as suas dependências que constituem este actor. Neste exemplo em particular os elementos internos representam o caso em que um utilizador quer registar uma queixa no sistema Health Watcher.

Uma vez que na secção 2.2 já foi explicado para que serve cada um destes elementos, estes não voltarão a ser explicados nesta secção.

Uma ligação para atingir fins (*means-end*) é um tipo de ligação em que o fim a atingir pode ser um objectivo, uma tarefa, um recurso ou um *softgoal* e o meio para atingir esse fim é geralmente representado por uma tarefa.

Numa ligação objectivo-tarefa o fim a atingir é um objectivo e o meio para ajudar a atingir esse fim é uma tarefa.

Numa ligação recurso-tarefa o fim a atingir é um recurso e o meio para ajudar a atingir esse fim é uma tarefa.

Numa ligação *softgoal*-tarefa o fim a atingir é um objectivo leve e o meio para ajudar a atingir esse fim é uma tarefa. Nesta ligação é possível definir qual a contribuição (se positiva, se negativa) da tarefa para o objectivo leve.

Numa ligação *softgoal-softgoal* o fim a atingir é um *softgoal* e o meio para ajudar a atingir esse fim é outro *softgoal*, deste modo é possível estabelecer uma hierarquia entre *softgoals*. Nesta ligação também possível definir qual a contribuição dos objectivos leves na sua ligação.

Numa ligação tarefa-tarefa o fim a atingir é uma tarefa e o meio para ajudar a atingir esse fim é outra tarefa, deste modo pode-se dar um factor de opção para a realização de uma determinada tarefa através do uso de várias tarefas diferentes para atingir a tarefa pretendida.

Numa ligação objectivo-objectivo o fim a atingir é um objectivo e o meio para ajudar a atingir esse fim é outro objectivo, deste modo é possível numa determinada tarefa especificar como essa tarefa é realizada através do uso de um objectivo e de seguida especificar esse objectivo usando vários sub-objectivos.

Uma ligação de contribuição (*contribution link*) é um tipo de ligação em que se mostra como os *softgoals* contribuem positivamente ou negativamente para um determinado elemento. Este tipo de ligação ajuda a escolher caso existam que alternativas são melhores na modelação interna de um actor.

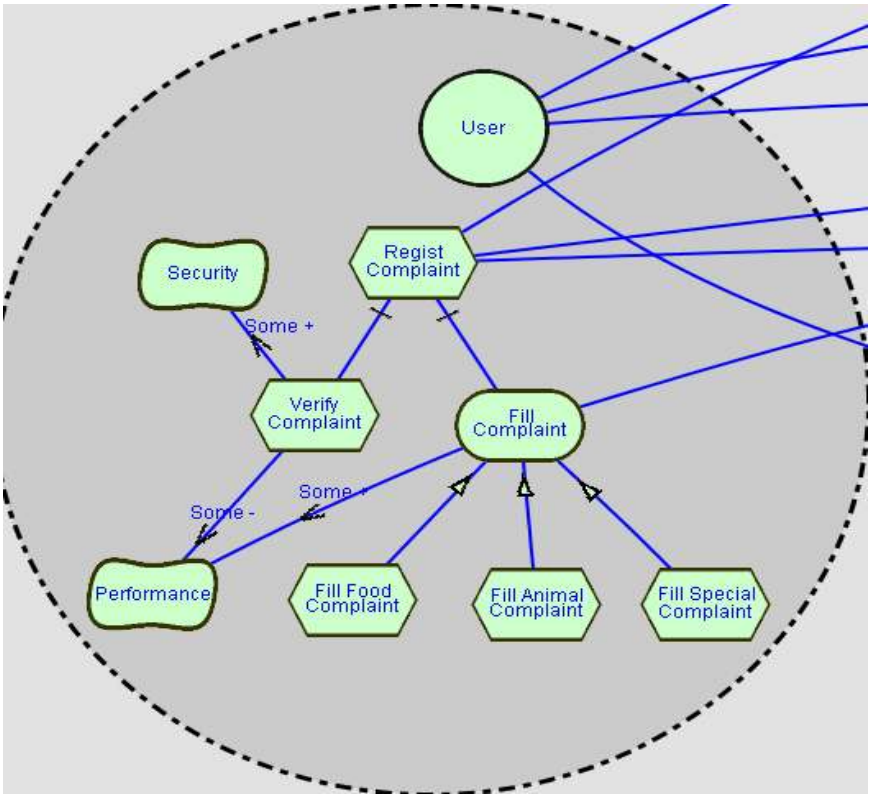

Figura 2.2: Exemplo da expansão do actor User no modelo SR para o sistema Health Watcher

2.4 Frameworks relacionados com i*

Nesta secção serão vistas frameworks que usam o i* como sua base, de modo a definirem a sua própria metodologia. Serão analisadas tanto a framework Tropos [4], como a framework GRL [4].

2.4.1 Tropos

O Tropos [4] é um projecto que tem como objectivo definir um método de desenvolvimento de software orientado aos agentes, usando uma variante da metodologia i* como linguagem de modelação.

O Tropos suporta todas as etapas necessárias ao desenvolvimento de software, desde a fase de levantamento de requisitos, até à fase de implementação do software. Em cada uma destas fases o Tropos adopta vários conceitos definidos no i* (actores, elementos, relações) e adopta cada um desses conceitos de acordo com a fase de análise que está a ser feita, de modo a criar

um modelo que melhor se adapte e represente cada uma das fases de desenvolvimento de software.

Por exemplo na fase de análise de requisitos, os Actores são usados para modelar os *stakeholders* do domínio e representar o sistema a ser construído. Sendo assim as dependências vão representar relações entre os *stakeholders* e o sistema. Já na fase de desenho os actores vão representar componentes constituintes do sistema e que agentes devem ser implementados. As relações nesta fase representam o fluxo de dados entre os componentes e os agentes.

2.4.2 NFR

A Linguagem de Requisitos Não Funcionais ou Non-Functional Requirements (NFR) [17] é uma linguagem usada na modelação orientada a objectivos.

A relação existente entre o NFR e o i* consiste em que ambas estas frameworks possibilitam uma hierarquização e atribuição de pesos positivos e negativos aos *softgoals*. Deste modo é possível definir que *softgoals* contribuem positivamente ou negativamente para outros *softgoals*, assim como permite uma melhor identificação e resolução de conflitos entre *softgoals*.

Esta framework funciona da seguinte maneira, começa-se por identificar os *softgoals* que representam os requisitos não funcionais. De seguida os *softgoals* identificados são geralmente refinados em outros *softgoals* e objectivos de forma a criar uma estrutura em árvore. Uma vez criada essa estrutura é preciso verificar que *softgoals* interferem com outros *softgoals* na estrutura. Finalmente escolhe-se uma *softgoal* que seja uma folha na estrutura de modo a que todos os *softgoals* que sejam raízes possam ser satisfeitos.

2.4.3 GRL

A Linguagem de Requisitos Orientada a Objectivos ou *Goal-Oriented Requirement Language* (GRL) [4] é uma linguagem usada na modelação orientada a agentes e objectivos e lida com requisitos não funcionais. O GRL é fortemente influenciado pelo i* e pela NFR para representar requisitos não funcionais.

Esta linguagem distingue três conceitos principais tal como o i*, sendo esses conceitos elementos intencionais, relações intencionais e actores, embora nestes últimos a sua especialização não seja possível ao contrário do i*.

Este método pode ser representado de três maneiras diferentes, através de uma representação gráfica dos modelos, uma representação textual dos modelos ou ainda uma

representação dos modelos em XML. Devido à possibilidade de representação destas três formas distintas torna-se impossível validar os modelos criados.

As principais diferenças entre o i* e o GRL são a possibilidade oferecida pelo GRL de se usar construtores para fazer ligações com elementos externos ao modelo do sistema que está a ser construído, elementos extra de argumentação e/ou contextualização oferecidos pelo GRL tais como crenças, correlações, tipos de contribuição e etiquetas de avaliação. Estes novos elementos permitem especificar estados de satisfação, estendendo deste modo os tipos das relações intencionais presentes no i*.

2.5 Vantagens do i*

Sendo o i* uma framework bastante versátil os seus principais pontos fortes prendem-se no facto de permitir uma análise tanto ao nível do sistema como ao nível da interacção com o sistema, por parte tanto dos vários utilizadores do sistema como de outros sistemas externos que seja necessário a existência de uma interacção com o sistema a ser desenvolvido.

Tal como o ponto anterior devido à flexibilidade do i*, este permite a sua aplicação em várias áreas distintas [10].

Ao contrário de muitas outras frameworks de análise e modelação, o i* permite especificar o modo de funcionamento dos vários componentes integrantes de um determinado sistema, assim como explicita o "porquê" dessas diversas funcionalidades terem aquele modo específico de funcionamento.

O i* possibilita ainda a identificação dos vários requisitos e preocupações expressas pelos principais interessados num determinado sistema, fazendo deste modo uma melhor modelação, integrando nesta os requisitos funcionais e não funcionais expressos pelos stakeholders [10,11].

Oferece ainda várias possibilidades de modelação de um aspecto específico do sistema, dando a hipótese de analisar as várias opções e escolher a mais correcta para aquele caso específico, através do uso de um sistema de contribuições positivas e negativas [10].

2.6 Limitações do i*

Sendo o i* uma framework ainda relativamente recente que se encontra em fase crescente tanto de utilização como de desenvolvimento, é natural que apresente algumas falhas tanto a nível estrutural como ao nível de visualização.

Aqui serão apresentados os principais pontos fracos desta framework.

Primeiramente, a existência de várias ferramentas que modelam a framework do i* e que por sua vez contêm o seu próprio formato de armazenamento torna muito difícil que haja uma partilha de modelos entre os vários utilizadores, limitando a disseminação desta framework.

Embora exista um método para uniformizar os vários modelos dos vários programas existentes chamado iML [3], que usa Extended Markup Language (XML) [3] para descrição e uniformização dos vários modelos construídos nas várias ferramentas. Deste modo com a existência de um modelo uniformizado torna-se simples a partilha dos modelos entre ferramentas, pois deixam de existir incompatibilidades entre os mesmos. No entanto este projecto encontra-se numa fase inicial pelo que não se sabe quando estará disponível e se realmente será viável.

Outra grande limitação dão i* reside no facto de não se oferecer um mecanismo de controlo da complexidade dos modelos, tornando a sua legibilidade e análise bastante complicada devido à sua vasta extensão e impossibilidade de análise individual de cada componente do modelo. Por exemplo no modelo presente na figura 2.1 pode-se ver muitas dependências entre o actor *User* e o agente *Health Watcher,* aumentando assim a complexidade e tamanho desse modelo i*.

Para tentar dar uma resposta a este problema têm-se estudado um meio de conseguir fragmentar um modelo um vários sub-modelos que representariam as várias componentes do modelo principal [5]. No entanto este estudo ainda não foi posto em prática pelo que não se sabe se realmente será viável e aceite pela comunidade de utilizadores do i*.

Para resolver esse problema nesta tese é proposta uma LDE baseada na framework i* que define compartimentos que agrupam e encapsulam elementos, de modo a que as dependências entre actores sejam reduzidas e cada compartimento possa ser analisado individualmente. Os actores terão a mesma possibilidade de ser expandidos e retraídos para que possam também ser analisados individualmente. Usando esta abordagem espera-se reduzir a complexidade visual dos modelos e ajudar na sua análise.

2.7 Ferramentas existentes e seus atributos

Existem várias ferramentas com o intuito de suportar uma modelação do método de análise i* sendo a maior parte destas ferramentas de âmbito de investigação, no entanto nesta tese apenas

se irá focar nas quatro ferramentas mais conhecidas e usadas, sendo elas o OpenOME [9,13], OME [9,13], J-Prim [9,13] e Taome4e [9,13].

A ferramenta OpenOME que se encontra presentemente na versão 0.13 foi criada pelo grupo de investigação da Universidade de Toronto. O principal objectivo do desenvolvimento desta ferramenta deve-se ao facto deste grupo de investigação querer criar uma ferramenta orientada aos objectivos e aos agentes com o framework do i* proposto por Eric Yu como base. Esta ferramenta é compatível tanto em Windows como em Linux e pode ser executada por si própria ou pode ser adicionada como um plugin no Eclipse [13] tendo sido criada em Java. Esta ferramenta pode ser usada tanto para fazer modelos i* como para desenvolvimento dos mesmos. É uma ferramenta ainda com alguns problemas tanto a nível de apresentação como de usabilidade estando no entanto a serem criadas novas versões com o objectivo de eliminar estas falhas.

A ferramenta OME que se encontra presentemente na versão 3.13 também foi criada pelo grupo de investigação da Universidade de Toronto, tendo como os mesmos objectivos que foram referidos na ferramenta anterior. Esta ferramenta é compatível apenas em Windows podendo ser executada por si própria e tendo sido criada em Java. Esta ferramenta pode ser apenas usada para fazer modelos i*. É uma ferramenta ainda com alguns problemas a nível de usabilidade e alguns bugs que podem invalidar os modelos criados estando no entanto a serem criadas novas versões com o objectivo de eliminar estas falhas.

A ferramenta J-Prim que se encontra presentemente na versão 1.0 foi criada pelo grupo de investigação UPC em Barcelona. O principal objectivo do desenvolvimento desta ferramenta prende-se com o facto deste grupo de investigação pretender criar uma ferramenta em que fosse possível a criação e análise de modelos i* de um ponto de vista orientado para o processo de reengenharia usando o framework proposto por Eric Yu como base. Sendo assim esta ferramenta permite a análise de um determinado sistema de informação e representá-lo numa hierarquia de elementos i*. Uma vez tendo sido modelado o sistema várias alternativas para o sistema podem ser modeladas usando vários modelos i* para cada alternativa, podendo esses modelos ser analisados para determinar qual a melhor alternativa para o sistema em análise. Esta ferramenta é compatível apenas em Windows sendo ainda necessário a existência do JRM 5.0, MySQL Server 5.0 e Eclipse para se executada. Esta ferramenta foi construída em Java. Com esta ferramenta é possível fazer modelos i* como desenvolver os mesmos. È uma

ferramenta ainda com alguns problemas a nível de usabilidade com poucos exemplos presentes na mesma estando no entanto a serem criadas novas versões com o objectivo de eliminar estas falhas.

A ferramenta Taome4e que se encontra presentemente na versão 0.3 foi criada pelo grupo de investigação de Engenharia de Software na IRST em Itália. O principal objectivo do desenvolvimento desta ferramenta deve-se ao facto deste grupo de investigação querer criar uma ferramenta que suportasse a metodologia Tropos e fosse uma ferramenta orientada aos agentes no domínio de desenvolvimento de software. Esta ferramenta é compatível tanto em Windows como em Linux e para ser executada necessita do JDK, Eclipse e dos plugins EMF e GMF, tendo sido criada em Java. Esta ferramenta pode ser usada tanto para fazer modelos i* como para desenvolvimento dos mesmos. Esta ferramenta apresenta os mesmos problemas da ferramenta anterior, com a adição de apresentar também alguns problemas a nível da funcionalidade. No entanto a serem criadas novas versões com o objectivo de eliminar estas falhas.

De seguida serão apresentadas três tabelas comparativas relativamente às funcionalidades, usabilidade e maturidade das ferramentas. Os parâmetros de comparação destas ferramentas serão as funcionalidades que cada uma disponibiliza, a sua usabilidade e em que situações cada uma destas ferramentas já foi testada.

Tabela 2.1: Tabela referente às funcionalidades das ferramentas

	OpenOME	**OME**	**J-Prim**	**Taome4e**
Modelo SD	Sim	Sim	**Não**	Sim
Modelo SR	Sim	Sim	Sim	Sim
Trabalhar Modelo SD e SR	Sim	Sim	Sim	Sim
Modelos Gráficos	Sim	Sim	Sim	Sim
Modelos Textuais	Sim	Sim	**Não**	Sim
Organização dos Modelos	Sim	**Não**	Sim	**Não**
Verificação dos Modelos SD	Sim	**Não**	**Não**	Sim
Verificação dos Modelos SR	Sim	**Não**	Sim	**Não**
Validação entre os Modelos SD e SR	Sim	**Não**	Sim	Sim
Trabalhar com vários Modelos	Sim	**Não**	Sim	**Não**
Agrupamento de modelos em	Sim	Sim	Sim	Sim

Projectos				
Trabalhar com vários Projectos	Sim	Sim	Sim	Sim

Tabela 2.2: Tabela referente à usabilidade das ferramentas

	OpenOME	**OME**	**J-Prim**	**Taome4e**
Usabilidade da Ferramenta	Média	Média	Média	Média
Qualidade do Manual	Compreensível	Compreensível	**Pouco Compreensível**	**Pouco Compreensível**
Ensina como usar o modelo i*	**Não**	**Não**	Sim	**Não**
Contem Exemplos	Sim	Sim	**Não**	Sim
Dificuldade de instalação	Fácil	Fácil	Fácil	Fácil

Tabela 2.3: Tabela referente à maturidade das ferramentas

	OpenOME	**OME**	**J-Prim**	**Taome4e**
Maturidade da Ferramenta	Usada em Publico	Ainda não foi usada em Publico	Ainda não foi usada em Publico	Ainda não foi usada em Publico
Usada em Casos de Estudo	Sim	Sim	Sim	Sim
Usados Modelos Extensos	Sim	Sim	Sim	**Não**
Problemas em Modelos Extensos	Sim, fica lenta	Sim, fica lenta	**Não**	**Não**
Tamanho Máximo de um Modelos	20 actores, 132 elementos e 237 ligações	Cerca de 100 elementos	30 actores e 500 dependências	Cerca de 100 elementos

Na Tabela 2.1, referente às funcionalidades que as ferramentas em análise suportam, pode-se verificar que a ferramenta OpenOme é a mais completa pois implementa todas as funcionalidades em análise. Em relação à ferramenta OME está é a que implementa menos funcionalidades, não suportando modelos textuais, validação de modelos e sua organização. Relativamente à ferramenta J-Prim, esta só suporta modelos textuais não permitindo a sua organização, no entanto todas as restantes funcionalidades em análise são suportadas.

Finalmente em relação à ferramenta Taome4e esta não suporta modelos textuais, suporta apenas a validação do modelo SR e não suporta validação entre modelos SD e SR, suportando no entanto as restantes funcionalidades presentes na tabela 2.1.

Na tabela 2.2, referente à usabilidade das ferramentas verifica-se que a usabilidade destas é aceitável, embora o suporte oferecido pelo manual das ferramentas J-Prim e Taome4e seja fraco. A ferramenta J-Prim por ouro lado é a unica que ensina a metodologia do i*, embora não contenha exemplos práticos de modelos ao contrário das restantes ferramentas.
Finalmente todas as ferramentas em análise são bastante fáceis de instalar.

Na tabela 2.3, referente à maturidade das ferramentas pode-se constatar que apenas a ferramenta OpenOme foi usada em domínios públicos, tendo no entanto todas as ferramentas já sido usadas em casos de estudo especificos. A ferramenta Taome4e foi a única que ainda não foi usada em modelos extensos não se sabendo muito bem portanto quais os seus limites.

Em relação aos limites que se pensa que cada ferramenta tenha estes diferem um pouco de ferramenta para ferramenta, mas tanto a ferramenta OpenOME como a ferramenta OME têm tendencia para ficarem mais lentas com modelos mais pesados.

2.8 Áreas de Aplicação

A framework do i* foi desenvolvida com o intuito de poder ser aplicada a várias áreas dentro da computação. Devido à enorme abrangência de áreas desta framework apenas serão analisadas as áreas mais relevantes que usam o i* [10], que são a engenharia de requisitos, reengenharia do processo de negócios, análise de impacto organizacional e processo de modelação de software.

A contribuição do i* para a área de engenharia de requisitos está na capacidade do i* permitir identificar explicitamente motivações, ou seja o que leva um determinado agente a realizar determinada tarefa ou objectivo e explicar essas mesmas motivações num modelo de requisitos ajudando deste modo a compreender melhor qual o âmbito do problema, assim como a identificar mais facilmente quais as preocupações dos utilizadores e quais os requisitos que estes desejam ver realizados no sistema. O i* também facilita o processo de desenvolvimento do sistema mesmo que aconteçam modificações de requisitos ao longo deste, uma vez que permite facilmente identificar e modelar alternativas para os novos requisitos que entretanto surgem durante a fase de desenvolvimento do sistema.

A contribuição do i* para a reengenharia do processo de negócios está na capacidade de facilmente conseguir identificar motivações e a razão por detrás dessas motivações de modo a conseguir responder às questões "porquê?" feitas no processo de reengenharia. Ajuda também no processo de análise e identificação de componentes desactualizadas no processo de negócios permitindo ainda a modelação de várias alternativas de substituição desse processo, justificando convenientemente as razões por detrás dessas alternativas.

A contribuição que o i* pode dar ao processo de análise de impacto organizacional consiste na possibilidade de modelar o ambiente organizacional da empresa e não só o sistema em si, assim como as relações entre os vários empregados percebendo assim melhor quais as sua capacidades e características, podendo deste modo modelar-se alternativas e verificar quais destas alternativas teriam um melhor impacto a nível social dentro da empresa.

A contribuição que se espera da aplicação do i* ao processo de modelação de software consiste na possibilidade de capturar as motivações por detrás das actividades num determinado projecto de software, dando ainda resposta aos "porquês" das diversas motivações que se encontrem nas possíveis alternativas de software.

2.9 Framework KAOS

Sendo a framework Knowledge Aquisition in autOmated Specification ou Keep All Objects Satisfied (KAOS) [21,22] uma linguagem tal como o i* muito popular e utilizada na comunidade de engenharia de requisitos.

O KAOS é uma linguagem orientada aos objectivos que suporta todo o processo de engenharia de requisitos, desde a definição dos objectivos de alto nível que se quer ver concretizados, até aos vários requisitos, objectos e operações que são atribuídos aos vários agentes intervenientes no sistema.

Geralmente cada modelação usando a linguagem KAOS é dividida em dois níveis estruturais, o nível que contém a camada gráfica onde os conceitos são expressos juntamente com os seus atributos e relações para outros conceitos, e existe o nível que contem a camada formal que serve para se definir formalmente os conceitos do modelo que se está a especificar.

O KAOS é uma linguagem que permite a detecção de conceitos que apareçam no processo de elaboração de requisitos. Os conceitos mais importantes presentes nesta framework são os conceitos de objectivos, objectos, operações, agentes, requisitos, entre outros.

A elaboração de um modelo usando a linguagem KAOS consiste nos seguintes passos.

- Elaboração dos objectivos, em que se define um grafo AND/OR definindo objectivos e quais os seus refinamentos de modo a atingir-se os requisitos que tenham sido assinalados.
- A captura de objectos, em que se identifica quais são os objectos que estão envolvidos na formulação dos objectivos, definindo as suas relações e propriedades.
- A captura de operações, em que se define quais os objectos responsáveis por transições de estado no sistema que se está a modelar, vão ser relevantes para a obtenção dos objectivos.
- Operacionalização, em que na derivação das pré e pós condições das várias operações, assegura-se que todos os requisitos necessários à realização dessas operações são cumpridos.
- A designação de responsabilidades, em que se identifica responsabilidades alternativas para os vários requisitos.

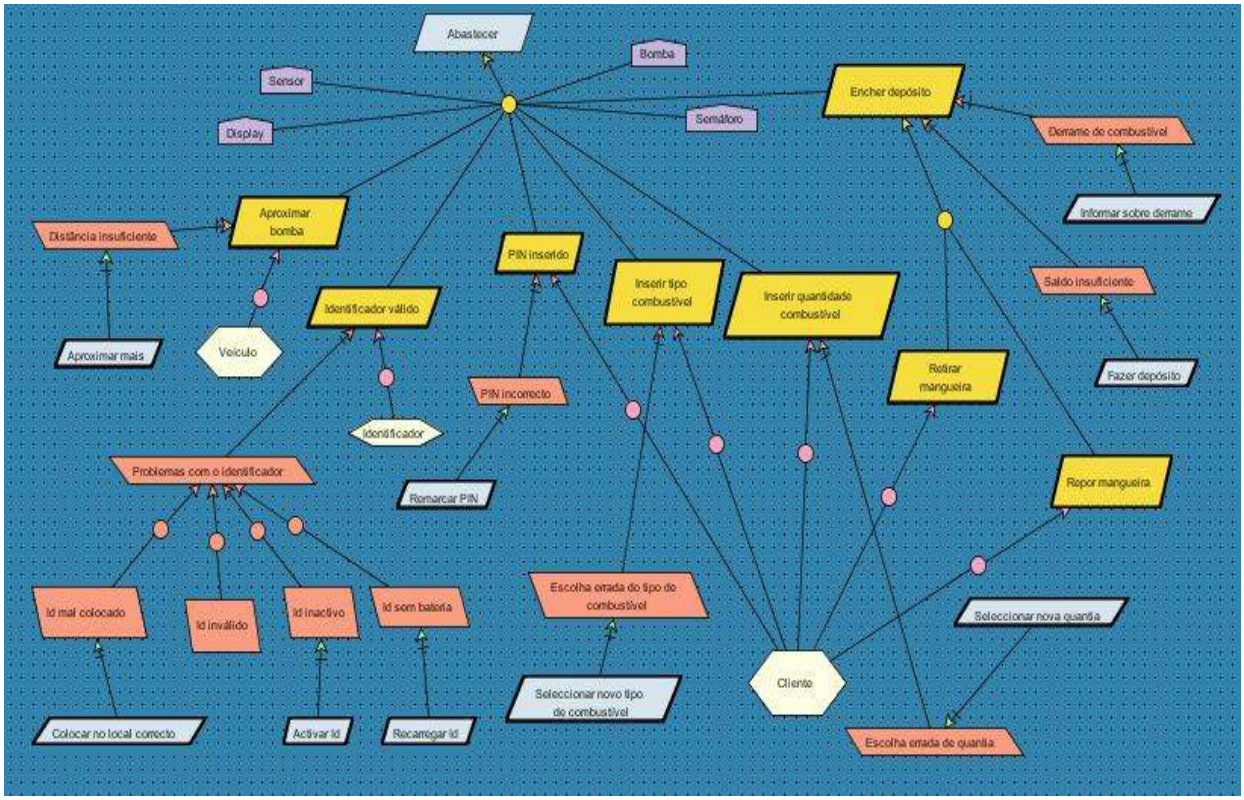

Figura 2.3: Exemplo Abastecer Carro na framework KAOS

2.10 Sumário

Neste capítulo viu-se em pormenor como era constituída a linguagem i*, a que áreas podia ser aplicada, quais as suas vantagens e limitações. Foi ainda mostrado quais as ferramentas mais populares que implementam esta metodologia e foi feito um estudo comparativo entre essas mesmas ferramentas. Finalmente estudou-se outras metodologias que se relacionavam com o i* e outras metodologias de análise de requisitos orientadas a objectivos.

No próximo capitulo vai ser mostrado o que é uma LDE, assim como quais as principais ferramentas que permitem a construção de uma LDE.

3. Linguagens de Domínio Especifico

Uma Linguagem de Domínio Especifico (LDE) ou "*Domain Specific Language (DSL)*" [16] tem como objectivo de especificar uma solução de um determinado problema específico num domínio em particular. Para esse efeito usa um Modelo de Domínio Especifico (MDE) ou "Domain Specific Model (DSM)" [20] com o objectivo de especificar a sintaxe do modelo para resolver o problema em questão.

Os MDEs têm ganho uma crescente popularidade porque permitem atingir um novo nível de abstracção na programação ao permitirem a especificação directa usando conceitos do domínio do problema que pretendem resolver. Este nível de abstracção é possível de obter porque tanto a linguagem como os geradores desta apenas precisam de respeitar os requisitos de um domínio em particular, ao contrário das linguagens de domínio geral.

Estando as LDEs directamente ligadas às MDEs estas também têm vindo a ficar cada vez mais populares na área de desenvolvimento de software. AS LDEs têm como objectivo criar uma linguagem de programação ou especificar uma linguagem própria para o domínio de um determinado problema em concreto, assim como para representar uma solução em particular para esse mesmo problema.

Este conceito presente nas LDEs acaba por não ser novo, uma vez que já existem há bastante tempo linguagens de programação para problemas especiais e linguagens de modelação e especificação, embora o termo tenha ficado popular graças à crescente popularidade da modelação de domínios específicos.

O oposto das LDEs são as Linguagens de Domínio Geral, tais como C ou Java [16, 20], uma vez que não existe nenhum domínio específico para o qual foram criadas estando normalmente directamente associadas a conceitos de computação e podendo ser as aplicações criadas nestas linguagens para os mais variados domínios.

A criação de uma LDE pode ser bastante vantajosa se esta estiver no nível certo de abstracção e usar as notações correctas. Ou seja, se conseguir expressar de forma mais correcta e concreta um determinado problema e/ou solução em relação às linguagens já existentes e se

esse determinado problema for levantado vezes suficientes que justifiquem a criação de uma LDE para lhe dar resposta.

Uma LDE como já foi referido anteriormente é criada como o propósito de resolver um problema específico associado a um determinado domínio, sendo muito difícil ou mesmo impossível resolver problemas associados a outros domínios distintos, ao contrário das linguagens de propósito geral que são criadas com o intuito de resolver problemas associados a vários domínios diferentes.

Um domínio consiste num ponto de vista de uma determinada área tal como uma família de produtos ou uma determinado assunto específico. Por exemplo pode-se ter um domínio para simular combates ou pode-se ter um domínio para resolver a secção contabilística de uma empresa.

Uma LDE encontra-se entre uma pequena linguagem de programação e uma linguagem de *Scripting,* sendo muitas vezes usada como uma *library* de um programa.

As LDEs são linguagens com objectivos muito explícitos tanto ao nível de desenho como de implementação e podem ser usadas a qualquer nível de abstracção, ou seja podem ser usadas ao nível de desenho, requisitos ou implementação. Uma LDE tanto pode ser uma linguagem diagramática, como pode ser uma linguagem textual [16].

A linha de separação entre as LDEs e as linguagens de *Scripting* acaba por ser algo difusa, no entanto as LDEs na sua maioria não contêm funções de baixo nível para acesso a ficheiros, não contêm controlo entre processos simultâneos e outras funcionalidades que caracterizam as linguagens de programação gerais e de *Scripting*.

De modo simplista pode-se ver as LDEs como ferramentas muito poderosas dentro dos domínios para as quais foram criadas, acabando por se perder muitas das suas capacidades se forem aplicadas a domínios diferentes para o qual foram desenvolvidas.

As vantagens de usar LDEs consistem na possibilidade de se poder expressar a solução no idioma e ao nível de abstracção do domínio do problema que se pretende resolver. Sendo assim os especialistas desse domínio conseguem compreender, validar e até modificar a LDE.

As LDE permitem a geração de código comentado para o domínio especificado, sendo ainda possível validar uma LDE ao nível de domínio.

Devido a estas características as LDEs aumentam a qualidade, produtividade, fiabilidade, portabilidade e reusabilidade, quando aplicadas a um domínio específico [16,20].

No entanto as LDEs contêm algumas desvantagens, uma vez que por vezes o custo de desenvolvimento e manutenção de uma LDE é bastante elevado. Por vezes o próprio domínio para o qual a LDE foi desenvolvida pode mudar o que pode levar a esta ficar obsoleta e os custos de actualizar essa mesma LDE muitas vezes acabam por não ser justificáveis.

Em comparação com programas desenvolvidos com linguagens de domínio geral pode haver uma diferença notória em comparação com as LDEs, visto estas serem menos eficientes fora do âmbito para as quais foram desenvolvidas inicialmente.

Finalmente é bastante complicado fazer *debuging* de uma LDE o que pode consumir um grande período de tempo, sendo por vezes mesmo impossível. Estas dificuldades devem-se ao modo como uma LDE é criada, sendo o código gerado a partir de um metamodelo que representa um domínio, deste modo o código gerado acaba por ser estranho a quem está a desenvolver a LDE o que dificulta bastante fazer *debuging* desse mesmo código.

Apesar das desvantagens aqui apresentadas, considera-se que as vantagens das LDEs conseguem superar largamente os seus pontos mais fracos, daí estarem cada vez com mais frequência a ser implementadas com sucesso nos mais variados domínios.

3.1 Ferramentas para Linguagens de Domínio Especifico

Aqui serão apresentadas as ferramentas mais relevantes a nível de utilização tanto no mundo empresarial, como no mundo de investigação, sendo explicado o porquê de se ter escolhido uma delas.

3.1.1 GME

O Ambiente de Modelação Genérico ou *Generic Modeling Environment* (GME) é uma ferramenta configurável para a criação e modelação de domínios específicos criada pelo Instituto para Sistemas Integrados de Software ou *Institute for Software Integrated Systems* [12]. A configuração do domínio que se pretende criar é feita através de meta-modelos que vão especificar o domínio da aplicação a desenvolver. Este paradigma vai conter todas as informações referentes à sintaxe, semântica e apresentação de um determinado domínio. Estando ainda especificado que conceitos serão usados para a construção dos modelos, que relações podem existir entre os conceitos, como podem esses conceitos estar organizados e ser visualizados pelo modelo construído.

O paradigma de modelação define a família de modelos que podem ser criados usando o ambiente que foi modelado.

O GME é baseado no diagrama de classes UML com restrições em OCL [18]. Os meta-modelos que especificam o paradigma de modelação são usados para automaticamente representarem o domínio específico alvo. Uma vez gerado o domínio especifico, este para a construção dos modelos de domínio que serão armazenados ou numa base de dados específica ou em formato XML. Estes modelos são usados para gerarem automaticamente aplicações específicas.

3.1.2 DSL Tools

A ferramenta de nome DSL Tools é uma ferramenta tal como todas as outras aqui em análise que permite criar e modelar domínios específicos, tendo sido criada pela Microsoft [14,19].

A DSL Tools é um conjunto de ferramentas que tem como propósito criar, editar e visualizar domínios específicos, com o objectivo de automatizar o processo de desenvolvimento de software.

Estas ferramentas são suportadas por uma framework que facilita a definição dos domínios que se queira criar, assim como a construção da sua componente visual que é suportada pelo Visual Studio.

Este conjunto de ferramentas é constituído por um *Wizard* que permite a definição e criação de um domínio específico passo a passo, assim como permite a sua validação e construção da componente gráfica para o domínio criado.

Permite actualizar domínios que tenham sido construídos previamente.

Contem um formato XML para criar certas definições no domínio em desenvolvimento, em que a partir deste formato XML vai ser criado o código referente ao domínio podendo assim o utilizador criar a sua parte visual no Visual Studio sem ter de fazer qualquer linha de código. Para além de gerar o código do modelo ainda permite a validação desse código.

3.1.3 EMF/GMF

Uma vez que tanto o EMF (Eclipse Modeling Framework) como o GMF (Graphical Modeling Framework) são plugins do Eclipse obrigatóriamente tem de se usar esta ferramenta para se poder usar tanto o EMF, como o GMF [13,19].

O Eclipse é uma framework de programação em Java que suporta um grande número de plugins entre eles o EMF e GMF que são as ferramentas aqui em análise e que se descrevem brevemente em seguida.

O EMF é uma framework de modelação e geração de código para construção de ferramentas e outras aplicações, baseado num modelo de dados estruturado ou meta-modelo Ecore [13,19].

Para além de se poder construir esse meta-modelo usando uma interface gráfica pode-se ainda construir o mesmo modelo recorrendo a um editor textual em XML.

Uma vez construído o meta-modelo do domínio específico que se pretende implementar, o EMF permite a geração e validação do código específico do modelo construído, gerando um conjunto de classes Java, sendo cada classe uma classe ou relação entre classes expresso no meta-modelo.

O GMF usa o Domain Model (ECore) que foi definido na fase relativa ao EMF. O GMF procura disponibilizar componentes e infra-estruturas que permitam o desenvolvimento de editores gráficos baseados no EMF [13,19].

Com estes editores gráficos é possivel definir toda a componente gráfica tanto ao nível dos elementos do modelo como ao nível das ferramentas que permitem criar esses elementos.

É no GMF que se define também quais as ferramentas responsáveis por cada elemento e como é que os elementos se relacionam entre si.

Esta foi a ferramenta escolhida pois é a que em termos de definição de interface visual e em termos de possibilidades de modelação, oferece os melhores métodos tendo um vasto número de opções disponíveis para se fazer uma correcta e completa modelação de um determinado domínio.

Outra vantagem destas ferramentas prende-se com a geração de código, uma vez que ao gerar-se um modelo é também gerado um ficheiro xml com a descrição completa do modelo criado, podendo facilmente retirar-se todas as informações necessárias desse modelo para gerar o código que se pretenda.

Finalmente outra grande vantagem desta ferramenta prende-se com o facto de ser *freeware* e já existir há algum tempo, o que possibilitou a existência de uma comunidade bastante grande em volta desta ferramenta, existindo assim muita informação disponível sobre a ferramenta e

sobre eventuais problemas que possam surgir durante a sua utilização. Existem ainda diversos exemplos práticos e alguns fóruns onde se pode expor problemas que surjam.

Apesar de ter uma linha de aprendizagem algo acentuada, considera-se que os ganhos compensam largamente o tempo que se tem de disponibilizar com a aprendizagem desta ferramenta.

3.2 Sumário

Neste capítulo introduziram-se os conceitos relacionados a LDEs, quais as suas vantagens em relação às linguagens de domínio geral e quando é que é frutífero utilizar-se uma LDE em vez de uma linguagem de domínio geral. Mostraram-se ainda quais as ferramentas mais populares que permitem o desenvolvimento de LDEs e efectuou-se um pequeno estudo comparativo para determinar qual destas ferramentas se iria utilizar para desenvolver a ferramenta para esta tese.

No próximo capítulo será estudado em pormenor o metamodelo da framework i* de modo a poder construir-se a LDE com base no metamodelo criado a partir desse estudo.

4. Trabalho realizado

Neste capítulo será apresentado o trabalho até agora desenvolvido para a realização desta tese, assim como o trabalho que se irá realizar futuramente.

Na secção 4.1 serão apresentados e explicados os metamodelos criados para representar o modelo SD e SR respectivamente.

Na secção 4.2 serão apresentados todos os passos em pormenor que foi preciso dar para criar-se a LDE que permita modelar correctamente a framework i*.

Na secção 4.3 será apresentado um caso de estudo para demonstrar a funcionalidade da LDE desenvolvida na secção anterior, assim como serão demonstradas quais as diferenças e novas funcionalidades que esta LDE trás em relação às ferramentas existentes.

4.1 Metamodelação da Framework i*

No método de análise orientado aos agentes e aos objectivos i*, são dados dois modelos para se fazer a análise de um problema específico.

No primeiro modelo chamado "*Strategic Dependency Model"* mais conhecido como *"SD Model"* o objectivo centra-se em fazer uma modelação usando as interacções dos vários actores constituintes do processo. Pretende-se assim conseguir perceber quais as motivações e objectivos que levam à realização de uma determinada funcionalidade dentro de um processo através dos actores intervenientes nessa funcionalidade.

O modelo SD ajuda na identificação de Stakeholders, no descobrimento de vulnerabilidades e oportunidades em relação ao sistema em análise, reconhecimento de relações entre os actores participantes auxiliando assim na obtenção de soluções para as vulnerabilidades detectadas na análise do sistema
[10,11].

No segundo modelo chamado ""*Strategic Rationale Model*" mais conhecido como "*SR Model*" o objectivo centra-se em obter um nível de detalhe mais elevado através da análise das

operações internas do actores, não apenas as relações externas entre os actores do sistema como acontece no modelo SD. Deste modo o modelo SR consegue descrever as relações internas de cada actor e dar resposta em relação a como se consegue obter tal resultado para uma determinada funcionalidade, ou o porquê de se fazer determinada tarefa de determinada maneira. Para além de dar respostas a estes paradigmas, implementa ainda a noção de alternativa permitindo especificar na modelação interna de cada actor qual o seu comportamento, assim como as alternativas encontradas para esse comportamento. Deste modo é possível para cada actor encontrar novos modelos de processo que melhor correspondam aos seus interesses, necessidades, expectativas e preocupações.

Nas próximas secções 4.1.1 e 4.1.2 serão apresentados os metamodelos do modelo SD e SR e os seus constituintes, assim como uma explicação desses modelos [10,11].

4.1.1 Metamodelo do Modelo SD

Este metamodelo (Figura 4.1) foi construído tendo como base os metamodelos em [1,4,6]. Para o modelo SD foram apenas utilizados os componentes que diziam respeito a este modelo, sendo os restantes ignorados.

Neste metamodelo tem-se como raiz uma classe que representa o modelo SD com um atributo para representar o nome a dar ao modelo. A essa classe vão estar agregadas outras duas classes, sendo uma classe responsável pela representação dos nós do modelo, que vão conter um atributo para representação do nome desse nó e outra classse para representação das relações existentes entre os vários nós. Num modelo SD como já foi referido pode existir um número indeterminado de nós e relações, podendo mesmo não haver nenhum dos dois.

As classes DependableNode e Dependum são generalizações da classe Node e as classes Depender, Dependee ISA Link e IsPartOf Link são generalizações da classe Relashionship.

A classe DependableNode pode relacionar-se com várias classes Depender e Dependee, no entanto tanto a Classe Depender como Dependee apenas relacionam-se com uma classe, DependableNode.

A classe Dependum relaciona-se apenas com uma classe Depender e Dependee, assim como a Classe Depender e Dependee apenas podem relacionar-se com uma classe Dependum.

As classes Goal, SoftGoal, Resource e Task são generalizações da classe Dependum e a classe Actor é uma generalização da classe DependableNode.

A classe Actor relaciona-se com as classes IsA Link e IsPartOf Link, no entanto tanto a Classe IsA Link como IsPartOf Link apenas podem relacionar-se com uma classe Actor.

As classes Agent, Role e Position são generalizações da classe Actor.

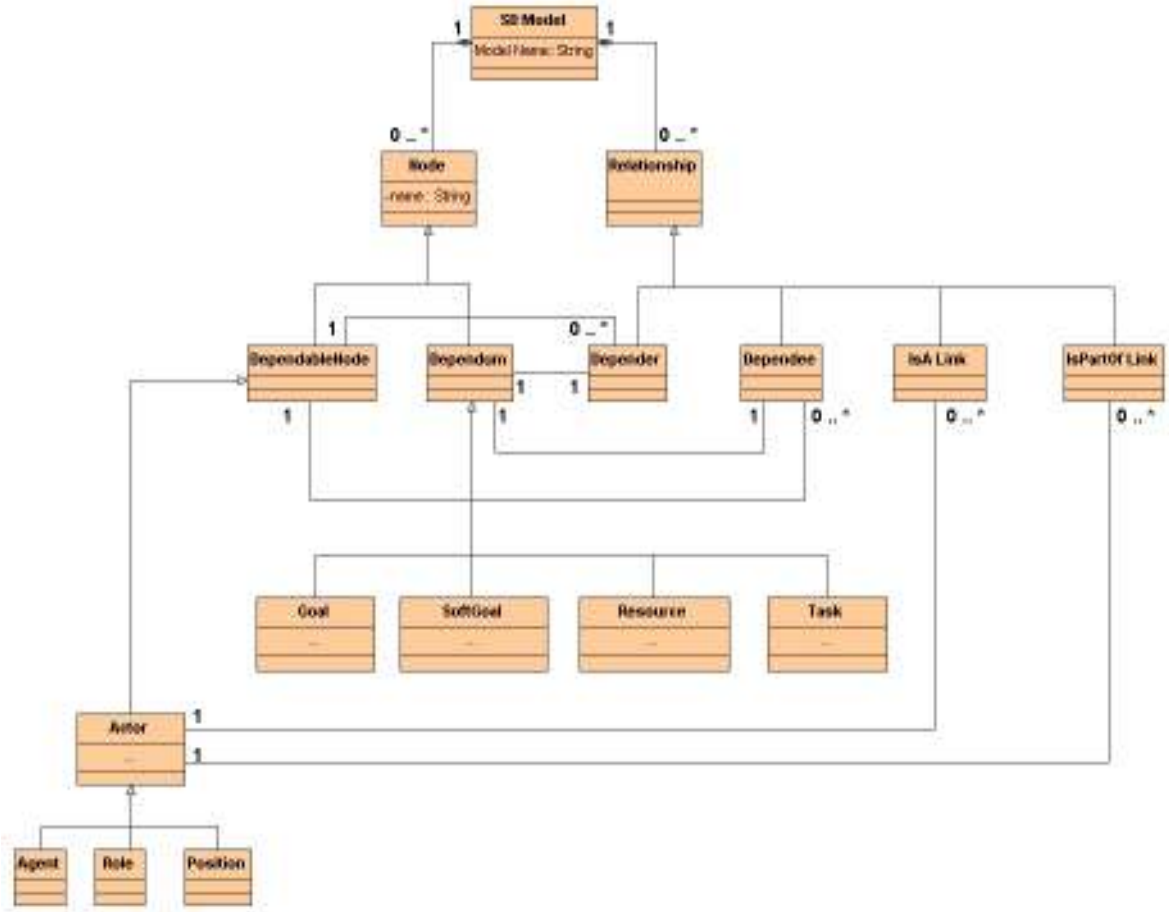

Figura 4.1: Meta Modelo relativo ao Modelo SD

4.1.2 Metamodelo do Modelo SR

Ao contrário do metamodelo construído para o modelo SD em que apenas foram usadas algumas componentes dos metamodelos em [1,4,6], para construir-se o metamodelo que representasse correctamente o modelo SR foi necessário usar quase na sua totalidade os componentes presentes nos metamodelos em [1,4,6], sendo apenas ignoradas as componentes que não pertencessem à framework do i* (Figura 4.2).

Uma vez que muitas componentes constituintes do modelo SR já foram explicadas na secção 4.1.1, apenas serão explicadas as componentes que não faziam parte dessa secção.

No modelo SR ao contrário do modelo SD é permitido expandir os Actores, sendo um actor constituído por vários elementos internos.

A classe Internal Element pode relacionar-se com várias classes Means-End Link, Decomposition Link e Contribution Link, no entanto tanto a Classe Means-End Link,

Decomposition Link e Contribution Link apenas podem relacionar-se com uma classe Internal Element.

As classes Means-End Link, Decomposition Link e Contribution Link são generalizações da classe Relationship apresentada no Modelo SD.

A classe Resource que é uma generalização da classe Internal Element pode relacionar-se com várias classes Decomposition Link, no entanto a classe Decomposition Link só se pode relacionar com uma classe Resource.

A classe SoftGoal que é uma generalização da classe Internal Element pode relacionar-se com várias classes to tipo Contribution Link, no entanto a classe Contribution Link só se pode relacionar com uma classe Softgoal.

As classes Some+, Some-, Help, Hurt, Make, Break e Unknow são generalizações da classe Contribution Link.

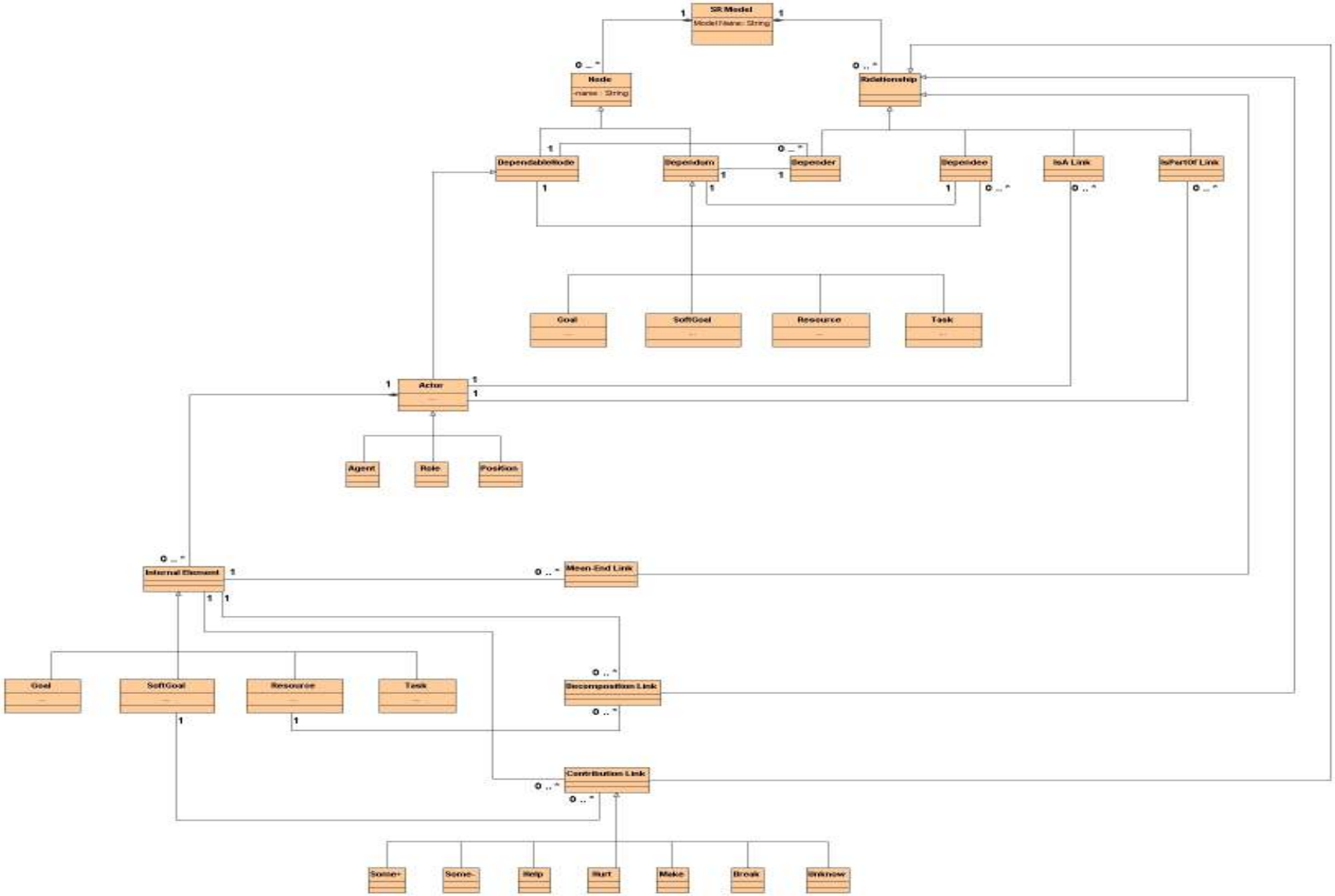

Figura 4.2: Meta Modelo relativo ao Modelo SR

4.2 Especificação da LDE para a framework i*

Foram usadas as ferramentas Eclipse Modeling Framework (EMF) e Graphical Modeling Framework (GMF) [13] para a construção da LDE que implementa o modelo SD. Estas duas ferramentas são plugins do eclipse com o propósito de permitir a criação e modelação de LDEs neste ambiente.

4.2.1 Criação do Modelo Ecore

Para se poder criar uma LDE vai-se usar um modelo Ecore [13,16] para especificar a linguagem que se quer construir. Usou-se um modelo Ecore pois este permite realizar uma especificação gráfica da linguagem, sendo deste modo mais simples de especificar o metamodelo da framework i*.

Um modelo Ecore consiste num metamodelo específico de uma LDE. Através desse modelo Ecore vão ser criadas todas as regras da LDE que se pretende construir, sendo por vezes necessário acrescentar algumas regras sintácticas posteriormente recorrendo à linguagem Object Constraint Language (OCL) [18]. O uso de OCL por vezes é necessário devido à incapacidade de se conseguir expressar todas as regras sintácticas de uma linguagem através do seu metamodelo.

Para criar o metamodelo Ecore referente à framework i* usou-se vários metamodelos específicos dessa mesma framework [1, 4, 6], aproveitando apenas os componentes necessários e adicionando alguns novos componentes para especificar as novas funcionalidades que se pretende implementar na LDE. Através de adaptações desses metamodelos, conseguiu-se criar um modelo Ecore válido para a representação numa LDE da framework em estudo.

Para que fosse possível resolver o problema da complexidade que sempre afectou esta framework criou-se duas novas classes, a classe ElementContainer e a classe SoftGoalContainer. Estas duas novas classes encontram-se ao mesmo nível que o resto dos elementos (*Goal, SoftGoal, Task, Resource*) desta framework e são usadas como compartimentos para estes elementos, deste modo pode-se agrupar vários elementos (*Goal, SoftGoal, Task, Resource*) e *softgoals* dentro de uma destas classes diminuindo assim o numero de ligações entre os actores e os elementos. Estas classes novas têm ainda a possibilidade de serem expandidas ou retraídas, permitindo assim uma análise individual dos elementos constituintes de uma ligação entre dois actores, algo que não era possível realizar usando o metamodelo original da framework.

Outra alteração é a possibilidade de usar os actores como compartimentos, podendo assim estes serem expandidos e retraídos individualmente, esta funcionalidade embora já exista em outras ferramentas nomeadamente a ferramenta OME, nunca foi implementada com grande sucesso. Deste modo é possível fazer a análise individual dos elementos e ligações internas de cada actor, assim como é possível passar dinamicamente do modelo SD para o modelo SR e vice-versa, sem a necessidade de construir outro modelo.

Pode-se visualizar o modelo Ecore com as respectivas inovações descritas na Figura 4.3. Neste modelo Ecore é possível ver estes dois novos elementos e a maneira como estes se relacionam com os restantes elementos do metamodelo. Estas inovações descritas (compartimentos expansíveis) foram as únicas alterações feitas no metamodelo da framework i*.

Com o modelo Ecore criado procedeu-se à geração do código, que transforma o modelo em classes Java. Esse código será usado posteriormente para criar o editor gráfico da LDE.

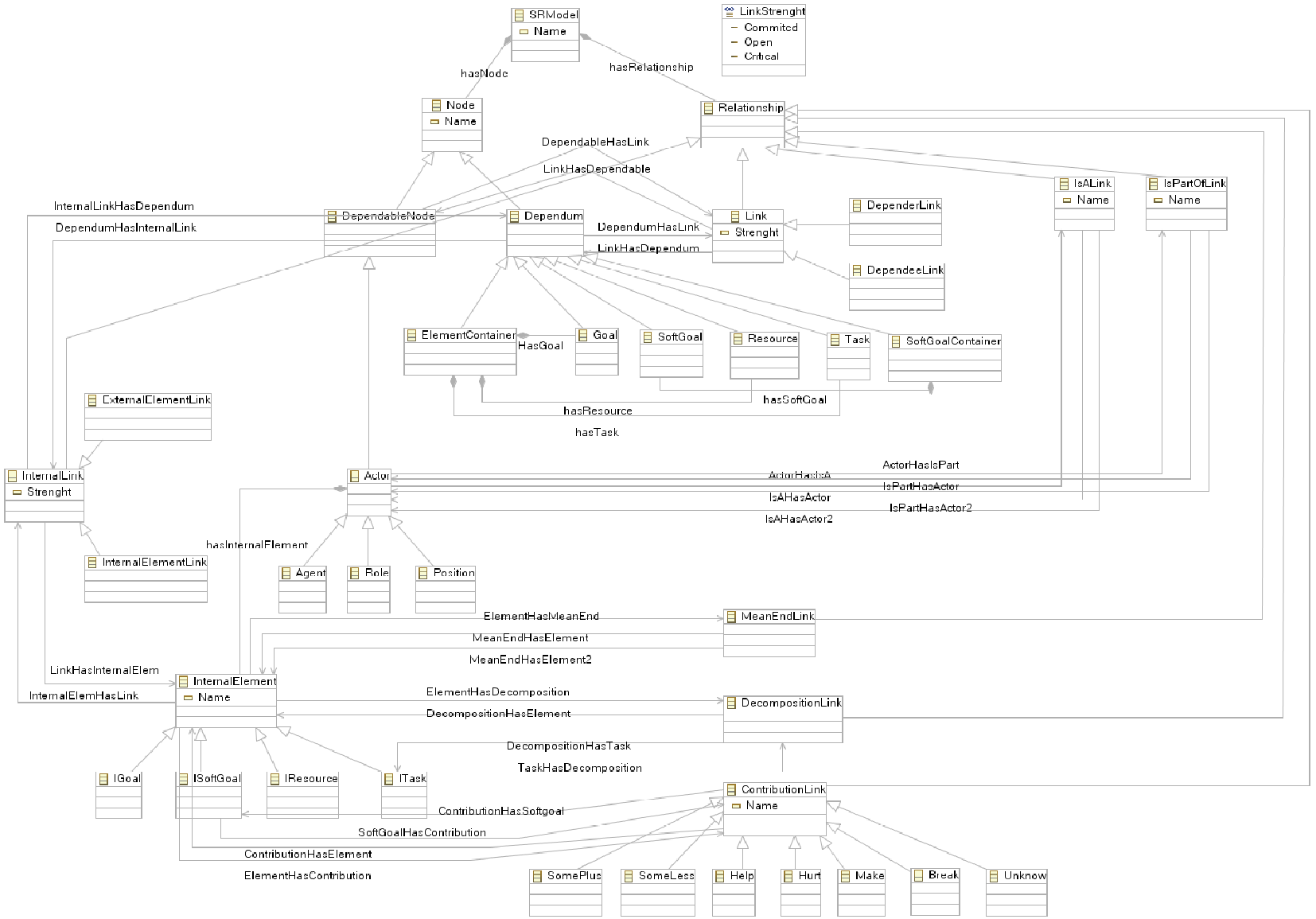

Figura 4.3. Modelo Ecore referente à framework i*

4.2.2 GMFGraph e GMFTool

Sendo o GMFGraph a componente da LDE responsável pela representação gráfica do modelo que se pretende criar e sendo o GMFTool a componente responsável pela apresentação visual da caixa de ferramentas a ser usada no modelo, como se pode ver na Figura 4.8. Uma vez que é nestas componentes que se vai reflectir o retorno visual da ferramenta, é importante que se dê especial atenção a esta fase da modelação da LDE.

Para se conseguir criar tanto a componente GMFGraph como a GMFTool usou-se a ferramenta GMF (figura 4.4).

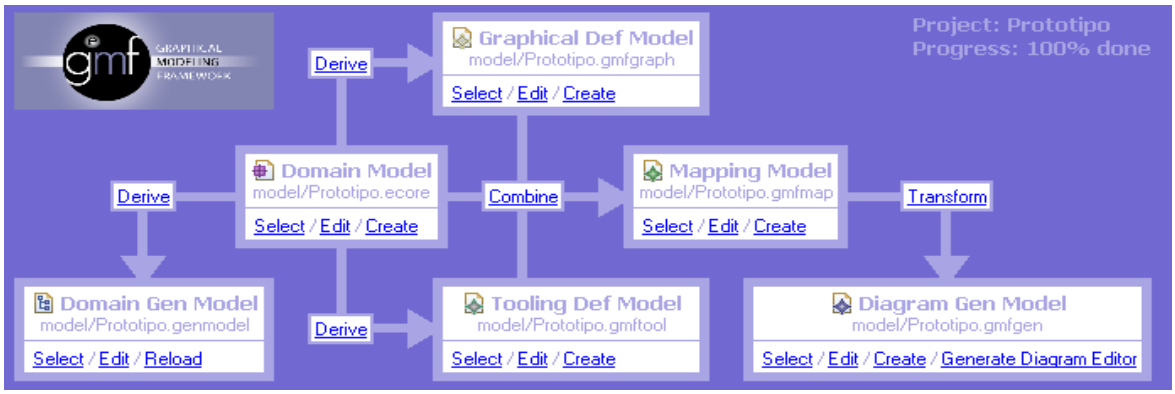

Figura 4.4. Componente de derivação

Como se pode visualizar na Figura 4.4, a partir do Modelo Ecore definido anteriormente, pode-se derivar as componentes GMFGraph e GMFTool. Estando estas duas componentes correctamente criadas, poder-se-á combiná-las, de modo a criar a componente GMFMap. Finalmente com essa componente correctamente criada poder-se-á gerar o editor da LDE que se pretende, sendo neste caso específico um editor para a framework i* (Figura 4.7).

Durante a criação destes dois novos componentes foi necessário definir qual era a classe que representaria o nó raiz, sendo neste caso específico a classe SRModel, como se pode visualizar no modelo Ecore (Figura 4.3). De seguida foi necessário definir quais as classes que representariam nós e ligações entre nós no modelo, as restantes classes que não foram escolhidas nem para nós, nem para ligações foram descartadas.

No final deste processo como era esperado foram identificadas as classes *Actor, Agent, Role, Position, Goal, SoftGoal, Task, Resource, ElementContainer, SoftgoalContainer, Igoal, IsoftGoal, Iresource e ITask* como nós e as classes *DependeeLink, DependerLink, IsALink,*

IsPartOfLink, ExternalElementLink, InternalElementLink, MeanEndLink, DecompositionLink e todas as ligações de *ContributionLink* como ligações.

Gerados estes dois componentes obtêm-se um resultado pré definido, embora tenha sido necessário proceder a algumas alterações, de modo a criar-se o retorno visual pretendido. Estas alterações são necessárias porque o GMFTool cria todos os elementos da caixa de ferramentas num só separador, tendo de se adicionar separadores manualmente. Quanto ao GMPGraph cria por defeito rectângulos para a representação gráfica de todos os elementos, sendo necessário alterar manualmente esses gráficos para os pretendidos para cada elemento.

4.2.3 GMFMap

O GMFMap é a componente responsável pela parte lógica da DSL. É nesta componente que se vai definir logicamente os nós e as ligações do modelo.

Caso seja necessário impor algumas restrições extras ao modelo é nesta fase que essas restrições vão ser incluídas, embora ainda não estejam presentes neste editor.

Para criar esta componente é necessário no GMF (Figura 4.4) combinar as componentes GMFGraph e GMFTool, como referido na Figura 4.4, especificando mais uma vez quais as classes do modelo Ecore que serão nós e ligações. Devido a alguns bugs ainda existentes nesta ferramenta é necessário proceder a uma verificação e correcção manual do GMFMap.

Num nó existem propriedades para definir qual a sua origem no modelo Ecore e quais as relações aos nós criados pelas componentes GMFGraph e GMFTool. Como é mostrado na Figura 4.5.

Property	Value
⊟ Domain meta information	
Element	EClass Resource
⊟ Visual representation	
Appearance Style	
Context Menu	
Diagram Node	Node Resource
Tool	Creation Tool Resource

Figura 4.5. Propriedades de um nó

Como se pode visualizar na Figura 4.5 é preciso indicar qual a classe que representa o elemento que se quer criar, qual o nó definido no GMFGraph que representa o elemento e qual o nó no GMFTool que representa o elemento.

Numa ligação existem propriedades para definir qual a sua origem no modelo Ecore, qual a relação aos nós criados pelas componentes GMFGraph e GMFTool e quais os seus nós de origem e destino. Como é mostrado na Figura 4.6.

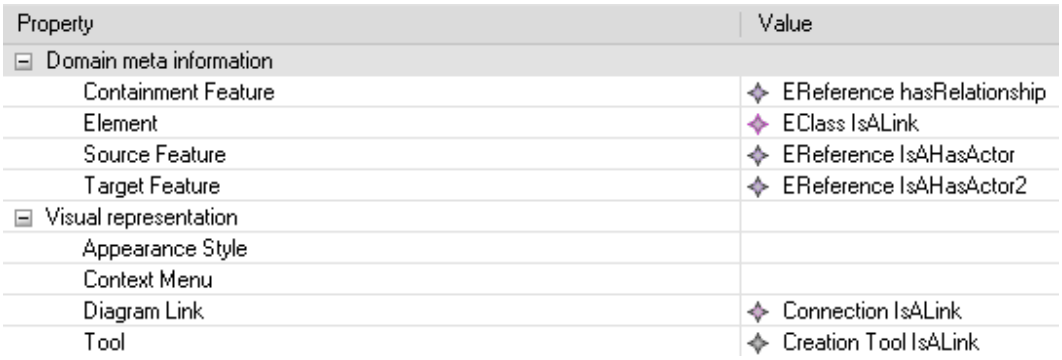

Figura 4.6. Propriedades de uma ligação

Como se pode visualizar na Figura 4.6 é preciso indicar qual a ligação correspondente no modelo Ecore que referencia a ligação que se deseja criar, qual a classe que representa o elemento que se quer criar, quais os nós de origem e destino que a ligação vai conter, qual o nó definido no GMFGraph que representa a ligação e qual o nó no GMFTool que representa a ligação.

Finalmente é no GMFMap que se modela logicamente quais os elementos que vão ser *containers* e quais os elementos que podem ser contidos dentro desses *containers*. Como se pode visualizar na figura 4.7 é necessário antes de tudo definir que o nó vai ser um *compartment,* criando para esse efeito um nó filho dentro do nó que se quer que contenha o *compartment.* De seguida é necessário definir quais os elementos que podem ser contidos dentro do nó *compartment*, sendo necessário criar dentro do nó pai, nós do tipo *ChildReference* que vão definir um nó filho que pode ser contido dentro do compartment. Deste modo consegue-se definir quais os nós que são *compartments* e quais os nós que podem ser contidos dentro desses mesmos *compartments*.

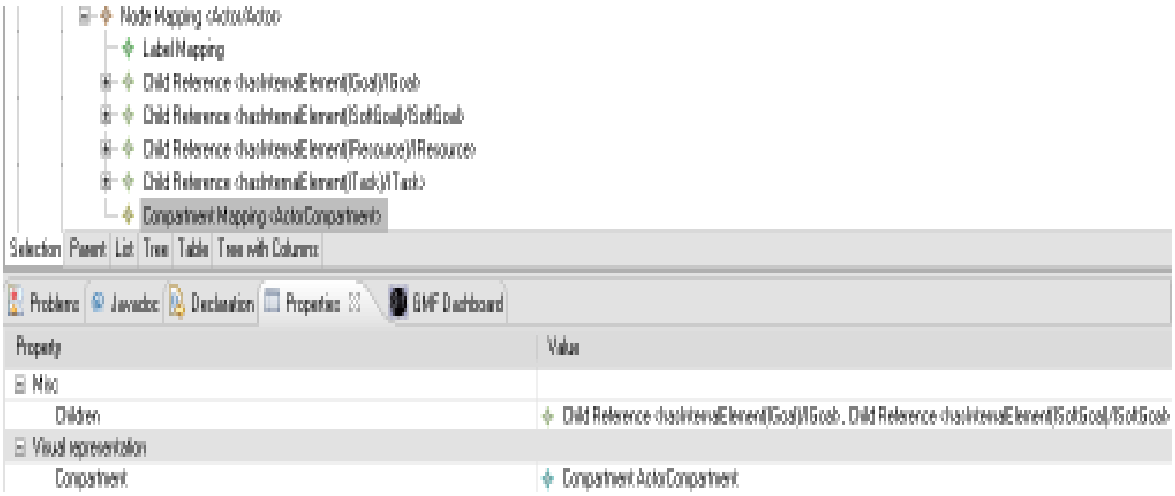

Figura 4.7. Propriedades de um *compartment*

Todas estas propriedades têm de estar bem especificadas para o modelo funcionar correctamente.

4.2.4 Editor da LDE

Estando todos os passos descritos acima concluídos, pode-se finalmente gerar o editor da LDE que se está a construir, neste caso especifico uma LDE baseada na framework i*.

A figura 4.8 mostra o editor da LDE, onde à direita se pode visualizar a palete responsável por conter todos os elementos e ligações que é possível utilizar na LDE e o resto da área disponível serve para editar os modelos que se pretende construir. Esta palete está dividida em cinco partes, na primeira parte estão contidos os actores e as dependências que podem ser feitas entre actores, na segunda parte estão contidos os elementos externos (objectivos, *softgoals*, tarefas e recursos) e os respectivos compartimentos que encapsulam esses elementos, na terceira parte estão contidas as dependências externas que podem ser usadas entre os elementos externos e os actores, na quarta parte estão contidos os elementos internos que podem ser usados no interior de um actor expandido e finalmente na quinta parte estão contidas todas as dependências internas que podem ser usadas entre elementos internos de um actor.

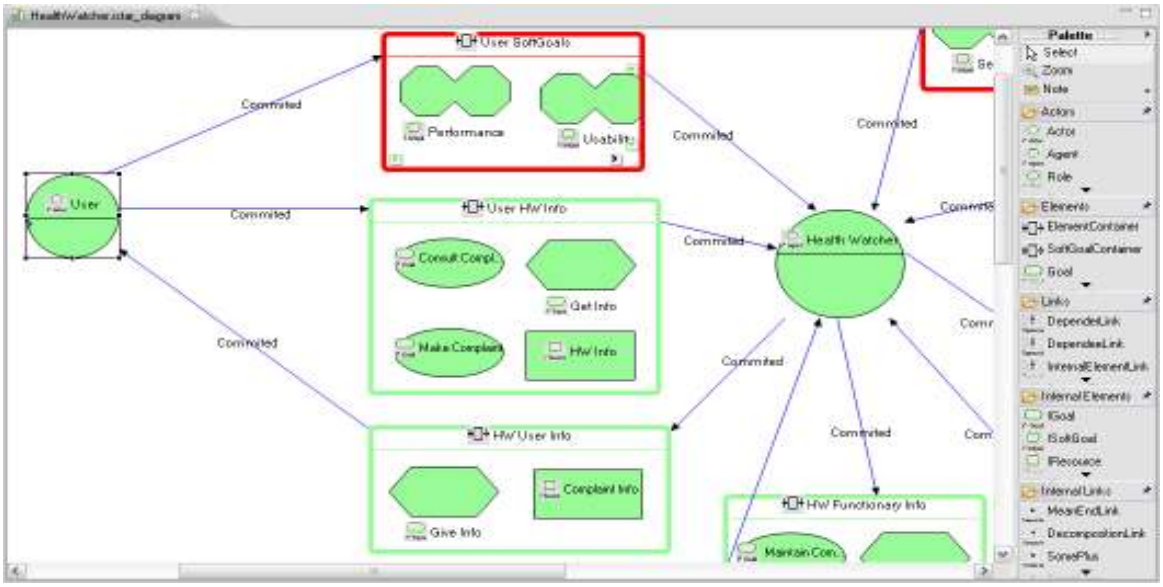

Figura 4.8. Editor da DSL criada para a framework i*

4.3 Sumário

Neste capítulo estudou-se em pormenor o metamodelo da framework i*. Usando este estudo construi-se um metamodelo próprio para modelar a LDE. De seguida foram mostrados todos os passos necessários para se construir com sucesso a LDE.

No próximo capítulo será definido um caso de estudo para se comparar a LDE construída com uma ferramenta muito popular de nome OME, será visto como foi feita a avaliação da ferramenta construída, assim como foi feita a análise dos dados obtidos através dessa avaliação e que conclusões se obtiveram através do estudo desses dados.

5. Avaliação da LDE

Neste capítulo será apresentado um caso de estudo comparativo entre a ferramenta criada e outra ferramenta bastante popular de nome OME, será também apresentado o questionário que foi usado para obter os resultados que serão apresentados nesta avaliação, assim como a análise desses resultados.

Na secção 5.1 será apresentado o caso de estudo e o estudo comparativo entre a LDE criada e a ferramenta OME.

Na secção 5.2 será apresentado o âmbito dos indivíduos que realizaram os testes e o questionário que foi usado para a realização desses mesmos testes.

Na secção 5.3 será apresentada a hipótese que se pretende testar nesta tese, assim como uma justificação para se ter escolhido essa hipótese.

Na secção 5.4 será apresentado como vão ser realizados os testes, assim como em que ambiente e com que objectivos estes vão ser realizados.

Na secção 5.5 será apresentado o questionário que foi usado para a realização dos testes e para recolher a informação necessária durante estes.

5.1 Caso de Estudo

O caso de estudo usado nesta tese é referente ao HealthWatcher [23, 24], que consiste num programa de serviço publico para que se possa fazer vários tipos de queixas que ponham em risco a saúde e segurança pública e também obter informação sobre campanhas de vacinação, essas queixas serão posteriormente analisadas por profissionais designados para o efeito. Usou-se este exemplo para validar a LDE construída. Sem um caso de estudo específico seria impossível demonstrar as funcionalidades desta. Este caso de estudo vai também ser usado para se fazer uma comparação válida com outras ferramentas que implementam a framework i*, através da sua modelação na LDE construída e na ferramenta OME. Ao fazer-se a

modelação do caso de estudo em ambas as ferramentas é possível analisar o resultado da modelação individualmente e detectar quais as suas diferenças e vantagens em relação a cada ferramenta.

Neste caso de estudo o sistema Health Watcher actua como um intermediário entre os cidadãos que têm queixas para apresentar e os funcionários que são responsáveis por analisar e dar resposta a essas queixas. Vai ainda estar dependente do actor "Admin" para realizar algumas actualizações ao sistema.

Para representar os serviços disponibilizados pelo sistema Health Watcher usou-se um agente de nome "Health Watcher".

Em relação ao actor "Citizen" o agente "Health Watcher" depende deste para fazer queixas em relação a problemas que possam afectar a segurança e saúde publica, aqui representadas como o objectivo "Make Complaint" e depende da informação dada por esses mesmos cidadãos por cada queixa feita, aqui representado como a tarefa "Give Info".

Em relação ao actor "Assistant" o agente "Health Watcher" depende deste para manter a informação correcta em relação às queixas previamente feitas, aqui representado como o objectivo "Maintain Complaint", depende da informação dada pelos funcionários para actualizar as queixas feitas, aqui representado como a tarefa "Get Complaint Info", depende dos funcionários para actualizar e criar novas tabelas no sistema, aqui representado como o objectivo "Register Tables" e depende da informação fornecida pelos funcionários para a correcta criação ou actualização das tabelas do sistema, aqui representado como a tarefa "Get Info Tables". Este actor está dependente do agente "Health Watcher" para aceder ao sistema, aqui representado pelo objectivo "Login", depende do sistema para obter a informação correcta relativamente às queixas efectuadas, aqui representado pela tarefa "Get Complaint Info" e pelo recurso "Complaint Info" e depende do sistema para obter a informação contida nas tabelas do sistema, aqui representado pela tarefa "Get Tables Info" e pelo recurso "Tables Info".

Para representar o administrador do sistema usou-se um actor de nome "Admin". Este actor não está dependente do agente "Health Watcher" uma vez que apenas actualiza ou regista nova informação no sistema.

Em relação ao actor "Admin" o agente "Health Watcher" depende deste para manter a informação correcta em relação às várias unidades de saúde presentes no sistema, aqui representado com o objectivo "Update Health Unit", depende da informação dada pelo

administrador para actualizar correctamente as tabelas do sistema, aqui representado como a tarefa "Give Health Unit Info", depende do administrador para regi* ou actualizar novos funcionários, aqui representado pelos objectivos "Update Functionary" e "Register Functionary" e depende da informação fornecida pelo administrador para actualizar ou regi* correctamente um funcionário, aqui representado pela tarefa "Give Functionary Info".

Para representar os cidadãos que interagem com o sistema usou-se um actor de nome "User". Este actor está dependente do agente "Health Watcher" para consultar queixas que tenham sido previamente feitas, aqui representado pelo objectivo "Consult Complaint" e depende da informação fornecida pelo sistema respectivamente às queixas que o cidadão pretende consultar, aqui representado pela tarefa "Get Info" e pelo recurso "Complaint Info".

Em relação aos requisitos não-funcionais foram identificados os seguintes para este caso de estudo:

- A segurança é um requisito não-funcional bastante importante neste sistema, uma vez que não se pretende que haja acessos não autorizados a queixas feitas, assim como não se pretende que os dados sejam adulterados.
- A Disponibilidade mede a probabilidade de o sistema bancário estar disponível para responder às necessidades dos clientes.
- A Performance é responsável pelo rápido acesso aos dados que se pretende assim como a uma execução em tempo útil das tarefas que se pretende realizar.
- A Usabilidade é responsável por assegurar uma fácil utilização do sistema por parte dos cidadãos.

Com o âmbito do caso de estudo definido para o modelo SD procedeu-se a sua modelação na LDE criada e na ferramenta OME sendo o resultado apresentado nas Figuras 5.1 e 5.2.

Em relação à figura 5.2 pode-se verificar que o aspecto gráfico dos elementos e as suas propriedades se manteve, uma vez que os utilizadores desta framework já estão muito habituados a esta definição gráfica. Em relação à palete de ferramentas tem-se o nome do elemento que essa ferramenta cria, assim como uma pequena imagem a explicitar que elemento é. Esta palete está disponível tanto da parte lateral direita do editor, como na parte de edição.

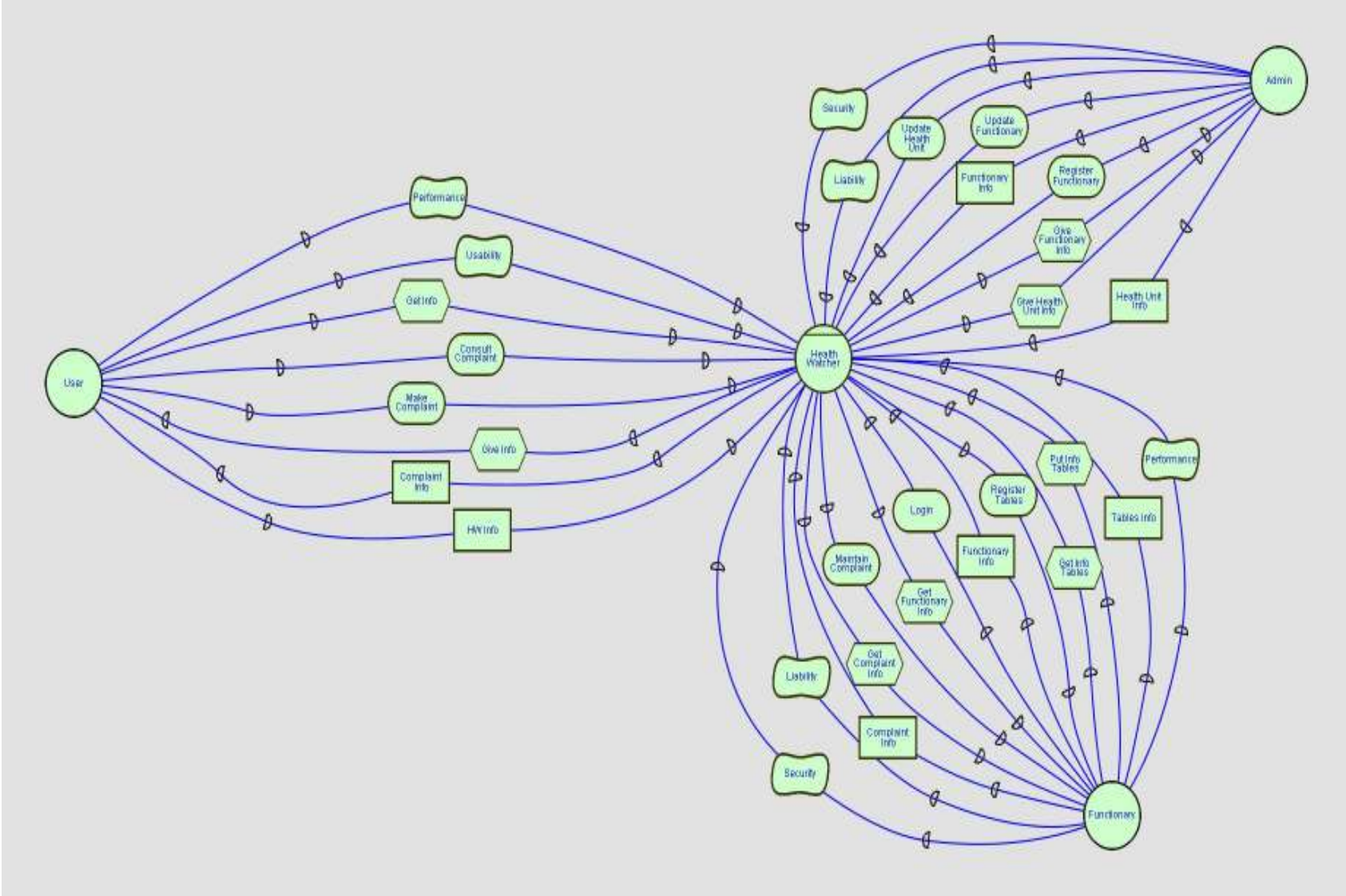

Figura 5.1. SD Model para o caso de estudo Health Watcher usando a ferramenta OME

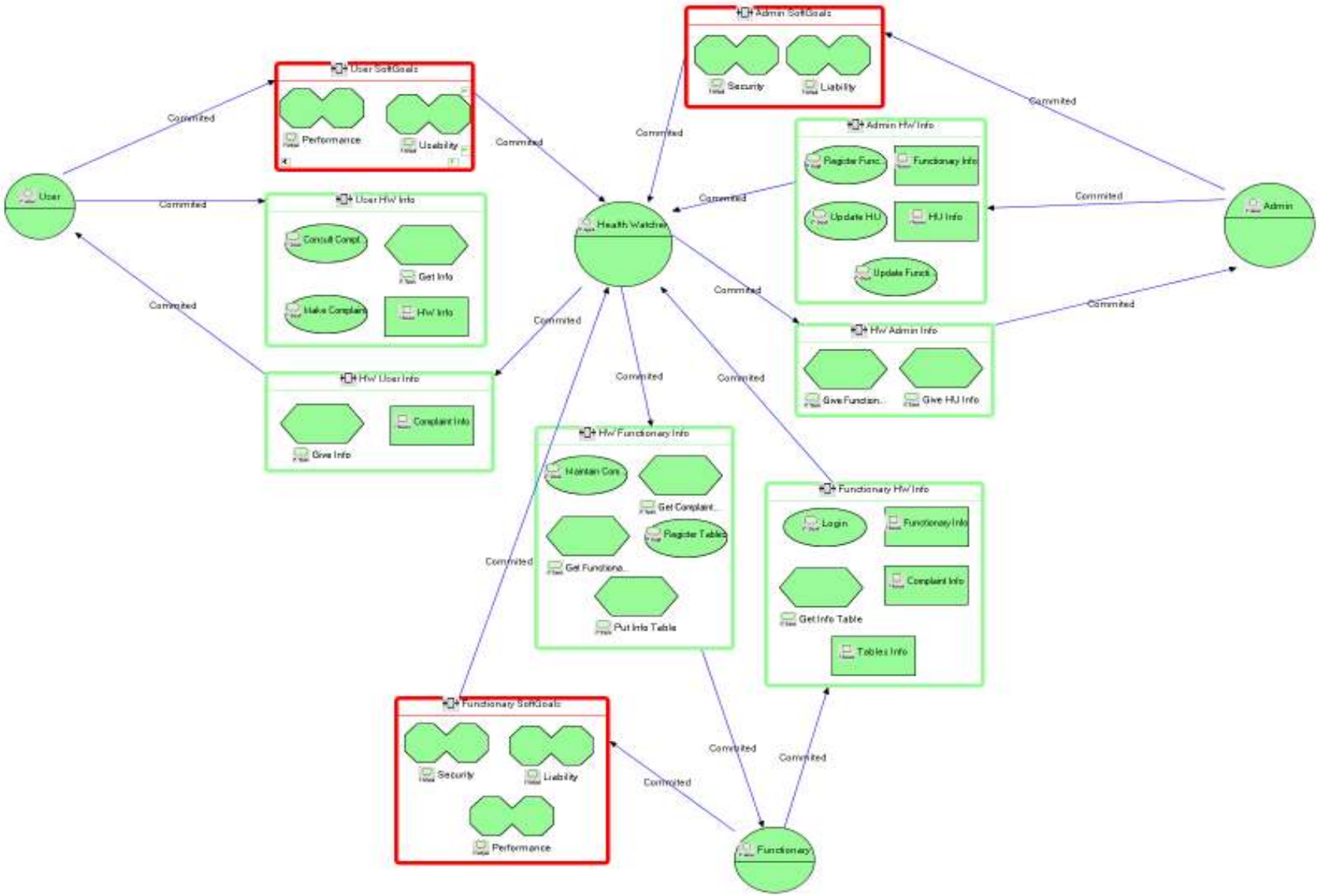

Figura 5.2. SD Model para o caso de estudo Health Watcher usando a ferramenta LDE i*

Em relação ao modelo SR, o caso de estudo vai ser idêntico tendo apenas os actores expandidos para que se possa especificar o seu comportamento interno.

O actor "User" foi expandido de forma a poder-se especificar como um cidadão realiza uma queixa e quais os procedimentos necessários para que essa queixa se realize. Para se especificar a realização de uma queixa usa-se uma tarefa interna no actor "User" de nome "Regist Complaint". Essa tarefa vai ser composta por uma outra tarefa "Verify Complaint" que serve para verificar se a queixa é válida e um objectivo "Fill Complaint" que especifica o preenchimento dos dados da queixa, este objectivo por sua vez pode ter três tipos distintos, aqui representados como os objectivos "Fill Food Complaint", "Fill Animal Complaint" e "Fill Special Complaint". Quanto aos requisitos não funcionais, dois foram relevantes, o requisito não funcional segurança, que contribui positivamente para a verificação das queixas, mas contribui negativamente para o tempo de preenchimento das queixas e o requisito não funcional performance que contribui positivamente para o preenchimento das queixas.

O actor "Assistant" foi expandido de forma a poder-se especificar como um funcionário entra no sistema e como regista novas tabelas no sistema, assim como se especifica que procedimentos internos a este actor são necessários para a realização dessas tarefas. Para se especificar a entrada no sistema por parte de um funcionário usa-se uma tarefa interna no actor "Assistant" de nome "Make Login". Essa tarefa vai ser composta por outras duas tarefas, "Verify Login" que serve para verificar se a informação que o funcionário introduziu é válida para entrar no sistema e "Put Login Info" que serve para especificar a informação necessária que um funcionário tem de colocar para poder entrar no sistema. Para especificar-se o registo de novas tabelas no sistema usa-se uma tarefa interna ao actor "Assistant" de nome "Regist Table". Essa tarefa vai ser composta por outra tarefa de nome "Verify Info Table" que serve para verificar se a informação colocada na tabela é válida e por um objectivo de nome "Put Info Table" que especifica o preenchimento de uma tabela a ser introduzida ou actualizada por parte de um funcionário, este objectivo por sua vez pode ser de três tipos distintos, aqui representados como os objectivos "Put HU Info", "Put Assistant Info" e "Put Disease Info". Quanto aos requisitos não funcionais, apenas um foi considerado relevante, o requisito não funcional segurança, que contribui positivamente para a verificação da informação de login e para a verificação dos dados de uma determinada tabela.

O actor “Admin” foi expandido de forma a poder-se especificar como o administrador actualiza a informação de novas unidades de saúde e como insere ou actualiza um novo funcionário, assim como se especifica que procedimentos internos a este actor são necessários para a realização dessas tarefas. Para se especificar a actualização de uma unidade de saúde usa-se um objectivo internos ao actor “Admin” de nome “New Info HU”. Este objectivo vai ser composto por uma tarefa de nome “Fill Update HU”, que serve para especificar o preenchimento da informação referente à unidade de saúde e por sua vez vai decompor-se numa outra tarefa de nome “Verify HU Info”, que serve para verificar se a informação referente à unidade de saúde é correcta. Para especificar-se como se insere ou actualiza um funcionário usa-se um objectivo interno ao Actor “Admin” de nome “New Info Assistant”. Este objectivo vai ser composto por uma tarefa de nome “Fill New Assistant”, que serve para especificar o preenchimento da informação necessária a um novo funcionário e outra tarefa de nome “Fill Update Assistant”, que serve para especificar o preenchimento da informação necessária à actualização de um funcionário. Por sua vez estas duas tarefas vão se decompor em uma outra tarefa de nome “Verify Assistant Info”, que serve para verificar se a informação preenchida é realmente correcta. Quanto aos requisitos não funcionais, dois foram relevantes, o requisito não funcional segurança, que contribui positivamente para a verificação da informação preenchida e o requisito não funcional fiabilidade que também contribui positivamente para a verificação da informação preenchida.

De seguida serão apresentadas quais as diferenças entre as ferramentas OME e LDE i*, assim como quais as vantagens da ferramenta LDE i*.

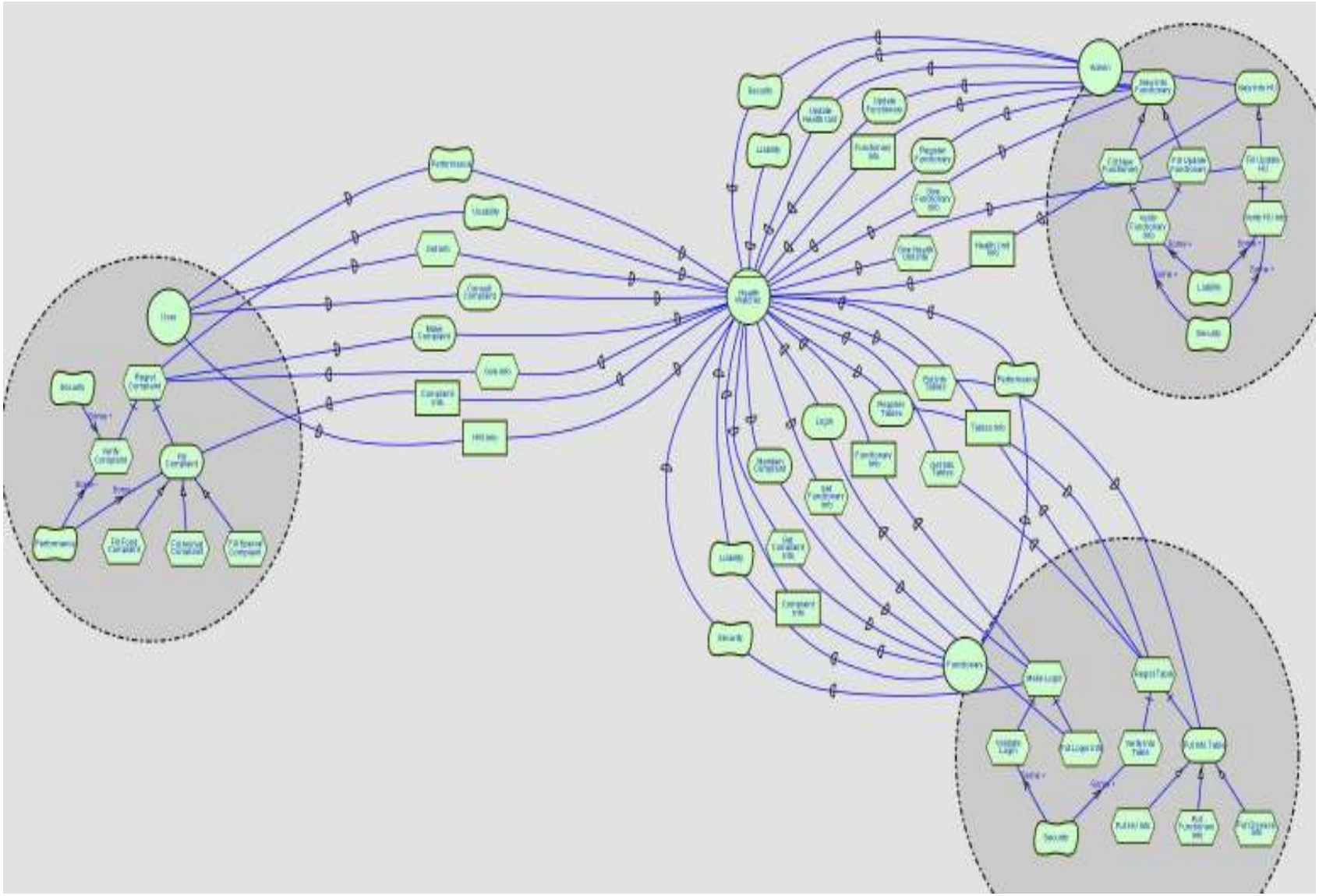

Figura 5.3. SR Model para o caso de estudo Health Watcher usando a ferramenta OME

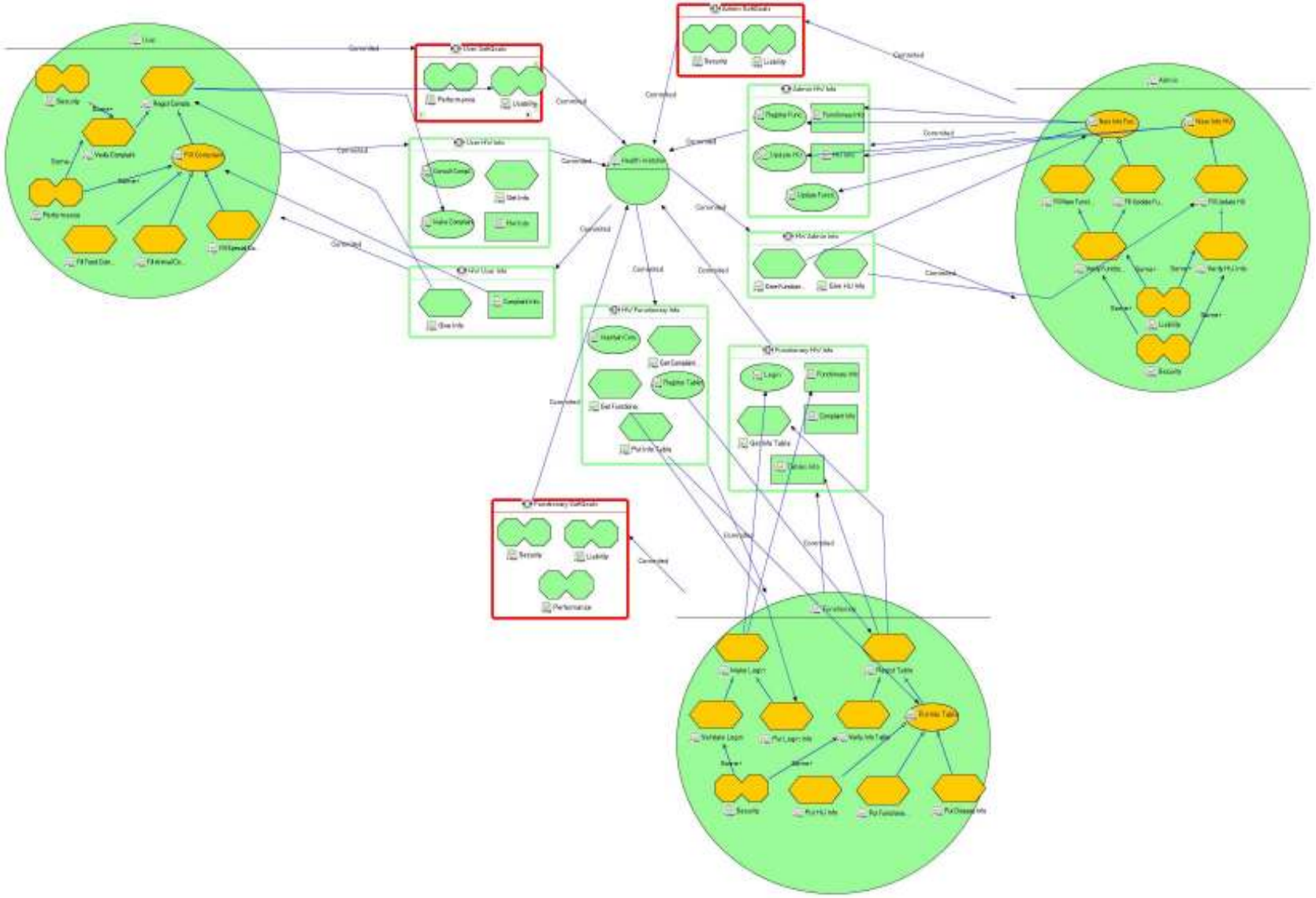

Figura 5.4. SR Model para o caso de estudo Health Watcher usando a ferramenta LDE i*

Na maior parte das ferramentas existentes, caso se queira passar de um modelo SD para um modelo SR é necessário construir um modelo à parte para esse efeito. Sendo assim é necessário a criação de dois modelos para abranger o caso de estudo, sendo uma modelo para representar o modelo SD e outro para representar o modelo SR. No entanto na DSL criada, assim como na versão mais recente da ferramenta OME é possível expandir e retrair os actores, sendo possível passar do modelo SD para o modelo SR e vice-versa. Com esta funcionalidade pode ainda fazer-se uma análise individual do conteúdo de cada actor, o que facilita a análise do conteúdo do modelo em estudo.

A figura 5.3 mostra um modelo SR criado na ferramenta OME para o caso de estudo. Pode-se ver por essa figura que existem várias ligações entre os actores o que dificulta a leitura e compreensão de modelos em grande escala.

A LDE construída implementa todas as funcionalidades vistas anteriormente, desde a implementação de todas as regras da framework i*, até à possibilidade de expandir e retrair os actores, de modo a fazer uma análise individual de cada um ou ainda passar dinamicamente do modelo SD para o modelo SR e vice-versa, retraindo ou expandindo os actores.

A figura 5.4 mostra um modelo SR criado na LDE construída para o caso de estudo. Para além das funcionalidades presentes em outras ferramentas, esta LDE propõem-se a resolver os problemas de escalabilidade apresentados pela framework i*, através dos já referidos compartimentos diminuindo consideravelmente as ligações entre os actores.

Existem dois tipos de compartimento, um apenas para os softgoals existentes entre dois actores e outro compartimento para os restantes elementos (Goals, Tasks and Resources) existentes entre dois actores. Deste modo separa-se eficazmente requisitos funcionais de requisitos não funcionais e consegue-se diminuir a escalabilidade e complexidade dos modelos.

Com os compartimentos é ainda possível fazer uma análise individual por compartimento, de quais os elementos constituintes de uma determinada ligação entre dois actores. Ajudando assim ao melhor entendimento dos modelos em análise, assim como a redução de complexidade dos mesmos.

Na figura 5.5 pode ver-se um exemplo da análise de um conjunto de compartimentos individuais.

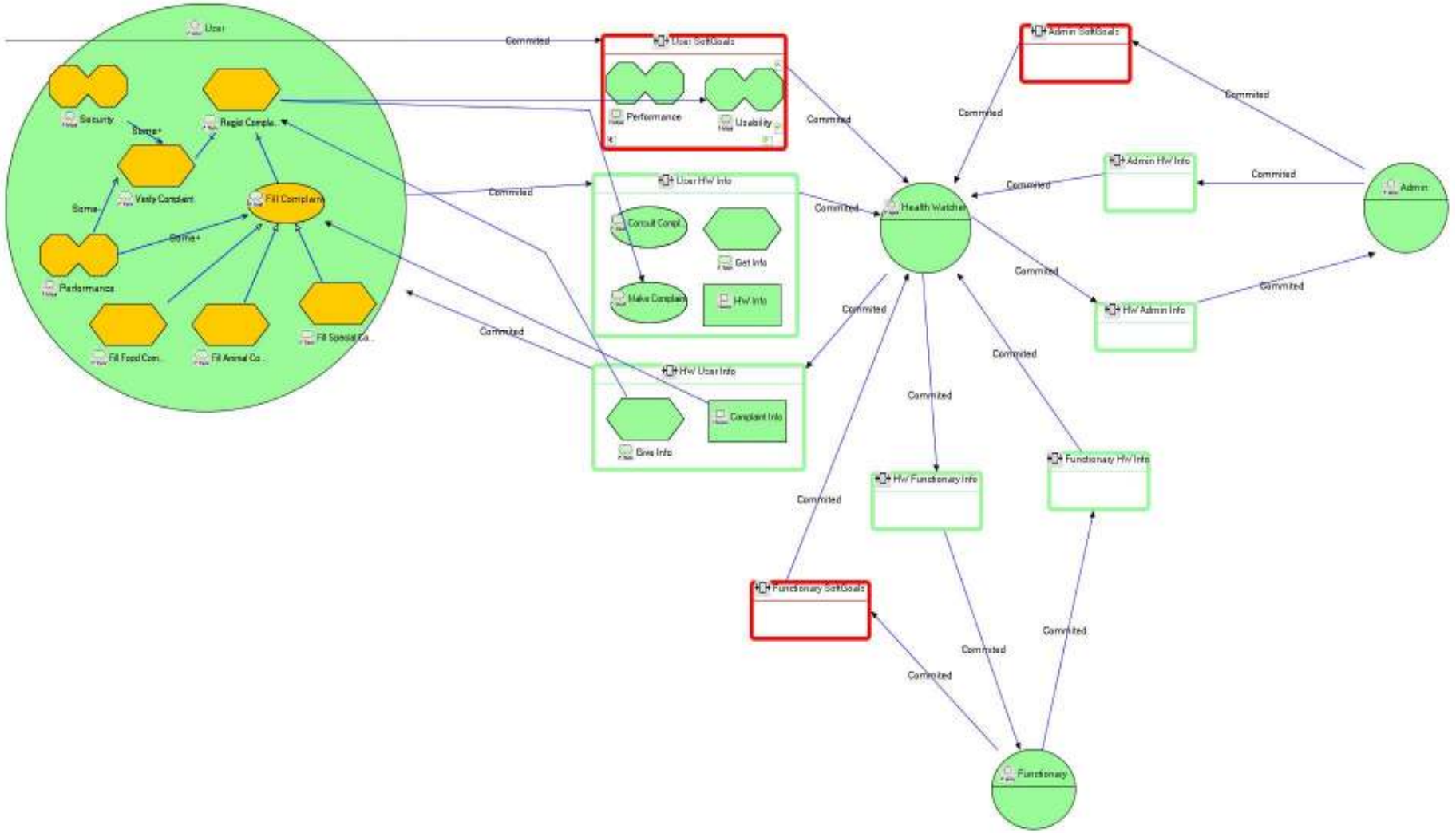

Figura 5.5. Exemplo sobre o uso dos compartimentos no modelo SD

Apesar destas inovações é possível construir modelos sem usar os compartimentos aqui apresentados, recorrendo assim à framework original do i*.

5.2 Âmbito da Avaliação

Antes de se poder realizar uma avaliação é necessário definir que tipo de avaliação se vai realizar e qual vai ser o âmbito desta. Vai ser usado o modelo de avaliação usado em [25], sendo o objectivo desta avaliação recolher dados quantitativos que possam corroborar as hipóteses apresentadas nesta tese em relação à LDE criada. Para além dos dados quantitativos também vão ser recolhidos alguns dados qualitativos através do questionário.

O objectivo desta avaliação para além da recolha de informação quer quantitativa, quer qualitativa, consiste na comparação da ferramenta OME com a LDE criada e efectivamente demonstrar que a LDE oferece algumas funcionalidades inovadoras e uma maior facilidade de utilização quando comparada com outras ferramentas.

O grupo escolhido para realizar os testes à LDE e responder ao questionário foi um grupo constituído por 10 indivíduos com conhecimentos em relação à framework i* e com alguma experiência no uso da ferramenta OME. Embora os indivíduos que participaram nestes testes sejam estudantes de mestrado da disciplina de Engenharia de Requisitos e não engenheiros de software, com experiência em casos reais e de grande escala, considerou-se que os seus conhecimentos em relação a framework i* e respectivas ferramentas que a implementam, que foram ganhos ao longo do curso eram suficientes para realizarem estes testes com o rigor e experiência que estes exigem. Considerou-se que o nível de conhecimento em relação à framework i*, ganho no curso era suficiente uma vez que neste estava presente a aprendizagem da framework i* assim como a sua aplicação a casos de estudo reais através da utilização de ferramentas que implementam esta framework inclusive a ferramenta OME.

Seleccionaram-se indivíduos com estas características pois, considerou-se que para os testes e resultados que dai fossem produzidos serem válidos, esses mesmos testes teriam de ser executados por pessoas com um conhecimento prévio da framework i* e da ferramenta OME, de modo a poderem fazer uma comparação válida entre a ferramenta OME e a LDE criada.

5.3 Hipótese que se Pretende Testar

Uma vez definido o grupo de testes que vai realizar a avaliação da ferramenta construída é necessário definir a hipótese que se quer testar de forma a poder-se especificar que dados se pretende recolher para validar essa hipótese.

Deste modo a hipótese desta tese é constituída por:

- Maior facilidade de utilização – pretende-se verificar se a ferramenta construída é mais simples de utilizar quando comparada com outras ferramentas, através da classificação do nível de satisfação dos utilizadores ao utilizarem esta ferramenta. Para esse efeito vão ser usadas questões escalares que permitem ao utilizador dizer com que facilidade realizou as tarefas pedidas e se estas tarefas foram mais simples ou mais difíceis de realizar nesta ferramenta.
- Inovação em relação a outras ferramentas – pretende-se verificar se a ferramenta construída trouxe realmente funcionalidades inovadoras que sejam úteis à comunidade interessada na framework i*. Para esse efeito vão ser usadas questões escalares que permitem aos utilizadores dizer se a framework usada nesta ferramenta tem semelhanças com as framework usadas noutras ferramentas e se realmente consideram que esta ferramenta traz ou não novas funcionalidades úteis.
- Gestão da complexidade dos modelos criados – pretende-se verificar se a ferramenta construída consegue realmente gerir melhor a complexidade dos modelos criados quando comparada a outras ferramentas existentes. Para esse efeito vai ser usada uma questão escalar que permite aos utilizadores dizer se realmente consideram que esta ferramenta melhora ou não a gestão da complexidade dos modelos construídos.

Para além das questões escalares apresentadas vão ser usadas algumas questões de resposta livre de modo a poder saber-se as opiniões de cada indivíduo após a realização do teste. Consideraram-se estas questões de resposta livre importantes pois através delas pretende-se obter informação mais detalhada do que os utilizadores pensaram realmente da ferramenta, informação essa que é impossível obter com questões escalares. Estas questões prendem-se com os aspectos positivos e negativos que consideraram, que inovações realmente acharam relevantes e que mudariam e melhorariam na ferramenta.

Com esta informação extra pretende-se ter um melhor entendimento e uma melhor análise dos dados obtidos, assim como se pretende ter uma justificação mais completa para a hipótese apresentada. Estes dados extra também são importantes para que numa fase futura se consiga corrigir aspectos menos positivos que tenham sido identificados pelos utilizadores durante os testes.

5.4 Testes

O objectivo destes testes é testar e avaliar a ferramenta construída, assim como se pretende ainda comparar essa mesma ferramenta com outras ferramentas que implementam a ferramenta i* como já foi referido anteriormente.

Como já foi referido na secção 5.2 são usados 10 voluntários com experiência prévia com a framework i* e ferramentas que implementam esta metodologia. Este número foi escolhido de forma a poupar-se o máximo tempo possível mas sempre com o objectivo de não prejudicar os resultados da experiência. Assim consideraram-se 10 indivíduos como um número razoável para se obter os dados necessários para esta avaliação.

Estes indivíduos resolveram previamente o problema proposto no questionário usando a ferramenta OME. Uma vez que já tinham essa experiência prévia foi pedido que resolvessem esse mesmo problema mas desta vez usando a LDE desenvolvida, de modo a poderem comparar com rigor ambas as ferramentas.

Cada sessão de teste foi feita individualmente demorando cerca de 30 minutos. Antes de cada sessão era explicada ao indivíduo que inovações tinham sido implementadas na LDE em relação à ferramenta OME e como eram essas funcionalidades utilizadas. Foi ainda explicado a interface gráfica da LDE, assim como foi dada uma rápida explicação sobre como o utilizador devia interagir com a ferramenta. Nesta parte da explicação não foi preciso explicar pormenores, uma vez que a LDE usa uma interface e uma interacção com a mesma muito semelhante à de outras ferramentas, nomeadamente a ferramenta OME, estando assim os utilizadores já familiarizados com este tipo de interface.

5.5 Questionário

Em relação ao questionário este divide-se em duas partes, sendo a primeira para questões escalares e a segunda para questões abertas. Na primeira parte tem-se como objectivo medir o nível de satisfação dos utilizadores. Na segunda parte tenta-se complementar os dados obtidos na primeira parte com opiniões dadas pelos utilizadores.

No anexo 1 pode-se ver o questionário que foi feito aos utilizadores e com o qual foram obtidos os dados para se usar nesta avaliação.

5.6 Avaliação dos Dados Recolhidos

Nesta secção serão apresentados que dados foram recolhidos durante a fase de testes. Esses dados serão analisados e será apresentado o resultado dessa análise assim como as conclusões resultantes dessa mesma análise.

5.6.1 Dados Gerais e suas Conclusões

Para se obter um gráfico comparativo de ambas as ferramentas com os dados gerais usou-se uma média dos resultados obtidos para cada uma das cinco questões quantitativas feitas.

À questão "Com que facilidade realizou as questões?" em média os utilizadores responderam com facilidade ou muita facilidade. Esta média é justificável pois apesar de se terem introduzido inovações em relação ao resto das ferramentas na LDE construída tentou-se ao máximo manter a framework i* original e simplificar o mais possível a interacção entre a LDE e os utilizadores.

À questão "Em comparação com outras ferramentas que usam a metodologia i* achou esta ferramenta mais simples ou mais complicada de utilizar?" em média os utilizadores disseram que esta ferramenta era mais simples de usar. Esta média é justificável porque com já foi dito anteriormente ao construir-se a LDE teve-se sempre o cuidado de tornar a interface da LDE o mais fácil e simples de utilizar possível, mantendo sempre todas as funcionalidades características da framework i* original.

À questão "Considera que esta ferramenta se aproxima da metodologia usada em outras ferramentas em relação a esta questão?" em média os utilizadores disseram que havia bastantes semelhanças mas também algumas diferenças. Esta média é justificável pois foram introduzidas novas funcionalidades relativamente à framework i* tais como os compartimentos, deste modo alterou-se um pouco a framework i* original.

À questão "Considera que esta ferramenta trouxe alguma inovação à metodologia i* em relação a esta questão?" em média os utilizadores responderam que esta ferramenta trouxe alguma inovação ou mesmo bastante inovação. Esta média é justificável porque como já foi mencionado anteriormente foram introduzidas algumas novas funcionalidades na framework i* tais como os compartimentos e uma melhor validação da sintaxe dos modelos.

À questão "Considera que esta ferramenta melhora os métodos de análise em termos de escalabilidade em relação a esta questão?" em média os utilizadores responderam que esta ferramenta ajudava ou ajudava bastante a reduzir e gerir a escalabilidade dos modelos. Esta média é justificável porque como já foi explicado anteriormente as novas funcionalidades introduzidas na LDE tais como os compartimentos e a possibilidade de expansão e retracção

dos actores e compartimentos permite que se analise partes isoladas do modelo construído ajudando assim à redução e gestão da complexidade dos modelos criados.

Na figura 5.6 mostra-se um gráfico com as médias obtidas nas cinco questões aqui visualizadas. O eixo "x" do gráfico representa o número das questões e o eixo "y" representa o nível de satisfação dos utilizadores. O nível de satisfação vai desde o valor 1 que representa um nível de satisfação baixo até ao valor 5 que representa um nível de satisfação elevado. Pode-se ver na figura 5.6 que o nível de satisfação está sempre acima do valor 3,5 mostrando assim que em média os utilizadores ficaram bastante satisfeitos com a LDE construída.

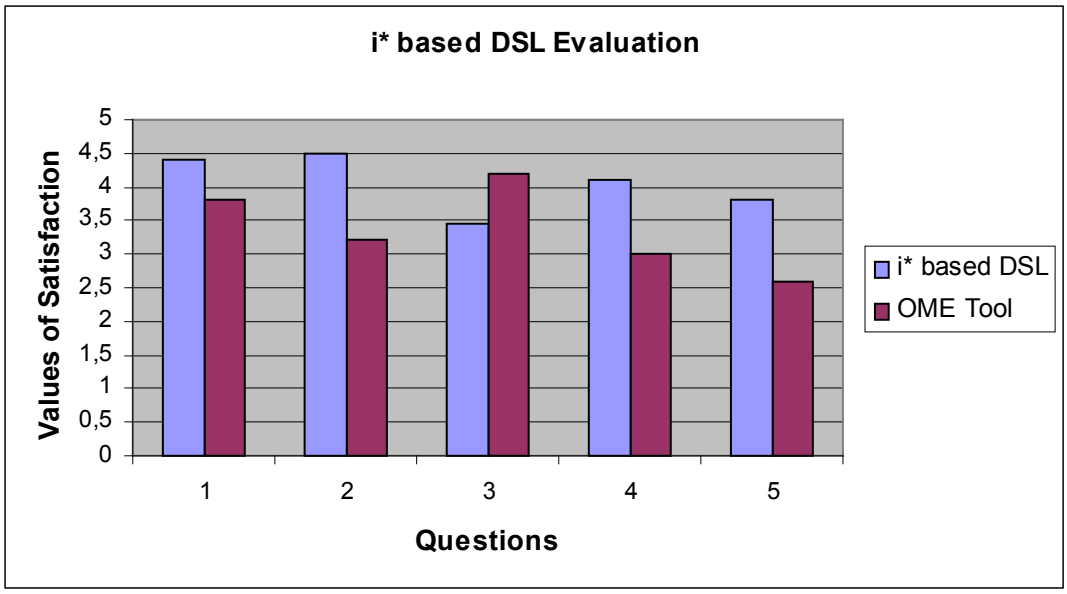

Figura 5.6. Gráfico mostrando as médias obtidas em relação a cada questão

5.6.2 Análise Individual por Questão

Tal como é mostrado no anexo 1 foram propostos quatro problemas para resolução por parte dos utilizadores de modo a poder-se obter dados quantitativos que pudessem ser usados para testar a ferramenta desenvolvida.

Para cada problema foram feitas as cinco questões apresentadas no capitulo 5.6. De seguida serão apresentados os objectivos para cada problema, assim como serão analisados individualmente os dados obtidos em cada problema.

No primeiro problema tinha-se como objectivo mostrar as funcionalidades comuns da framework i* original na ferramenta desenvolvida. Na figura 5.7 mostra-se o gráfico com os resultados da análise dos dados para este primeiro problema.

Os resultados que se obtiveram foram os esperados para este problema uma vez que tanto ao nível da sua resolução como ao nível da facilidade de utilização da ferramenta, os

utilizadores consideraram a LDE desenvolvida mais fácil de usar e por conseguinte conseguiram resolver o problema mais facilmente do que na ferramenta OME. Quanto à metodologia os utilizadores acharam que a ferramenta OME se aproximava ligeiramente mais da framework i* do que a LDE desenvolvida, esta resposta é justificável pois sendo a LDE uma nova ferramenta com algumas modificações tanto ao nível de funcionalidades como no aspecto gráfico (tendo havido no entanto a preocupação de manter esta ferramenta o mais próximo possível da framework i* original) é natural os utilizadores estarem mais familiarizados com a ferramenta OME uma vez que a usam à bastante tempo. Finalmente em relação às funcionalidades testadas e à gestão da complexidade, os utilizadores consideraram ao nível de funcionalidades a LDE mais completa, pois apesar das funcionalidades mais relevantes ainda não terem sido testadas neste primeiro problema existem mecanismos de verificação de sintaxe que não existem na ferramenta OME e quanto à gestão de complexidade os utilizadores consideram que existe uma gestão fraca para ambas as ferramentas neste problema, o que também é aceitável uma vez que os mecanismos de gestão não foram incluídos neste problema.

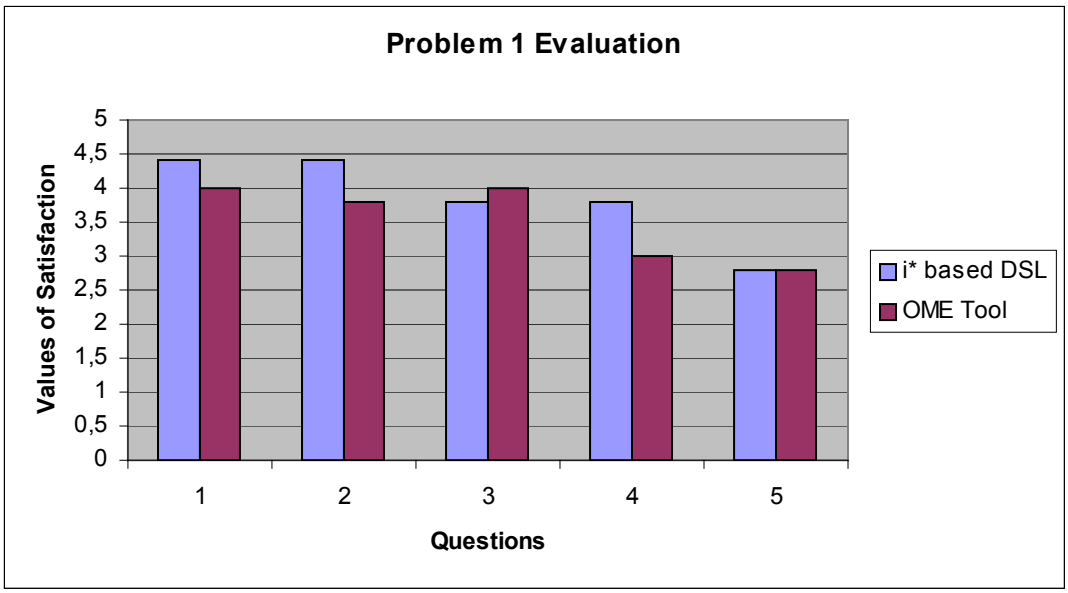

Figura 5.7. Gráfico com a análise dos dados obtidos no problema 1

No segundo problema tinha-se como objectivo mostrar os compartimentos e as suas vantagens de utilização ao agrupar vários elementos. Na figura 5.8 mostra-se o gráfico com os resultados da análise dos dados para este segundo problema.

Os resultados que se obtiveram foram os esperados para este problema uma vez que tanto ao nível da sua resolução como ao nível da facilidade de utilização da ferramenta, os utilizadores consideraram a LDE desenvolvida mais fácil de usar para esta questão devido à

possibilidade de agrupar elementos dentro dos compartimentos introduzidos nesta LDE e por conseguinte conseguiram resolver o problema mais facilmente do que na ferramenta OME que não tem os referidos compartimentos. Quanto à metodologia os utilizadores acharam que a ferramenta OME se aproximava mais da framework i* do que a LDE desenvolvida, esta resposta é justificável pois foram introduzidas novas funcionalidades que não existiam na framework i* original como os compartimentos, deste modo é natural os utilizadores afirmarem que existe algum afastamento da metodologia por parte da LDE desenvolvida. Finalmente em relação às funcionalidades testadas e à gestão da complexidade, os utilizadores consideraram ao nível de funcionalidades a LDE mais completa com a introdução dos compartimentos que permitem agrupar elementos e quanto à gestão de complexidade os utilizadores consideram que existe uma melhor gestão desta por parte da LDE uma vez que ao agrupar-se os elementos em compartimentos fica-se com menos ligações o que tornam os modelos mais fáceis de ler e compreender.

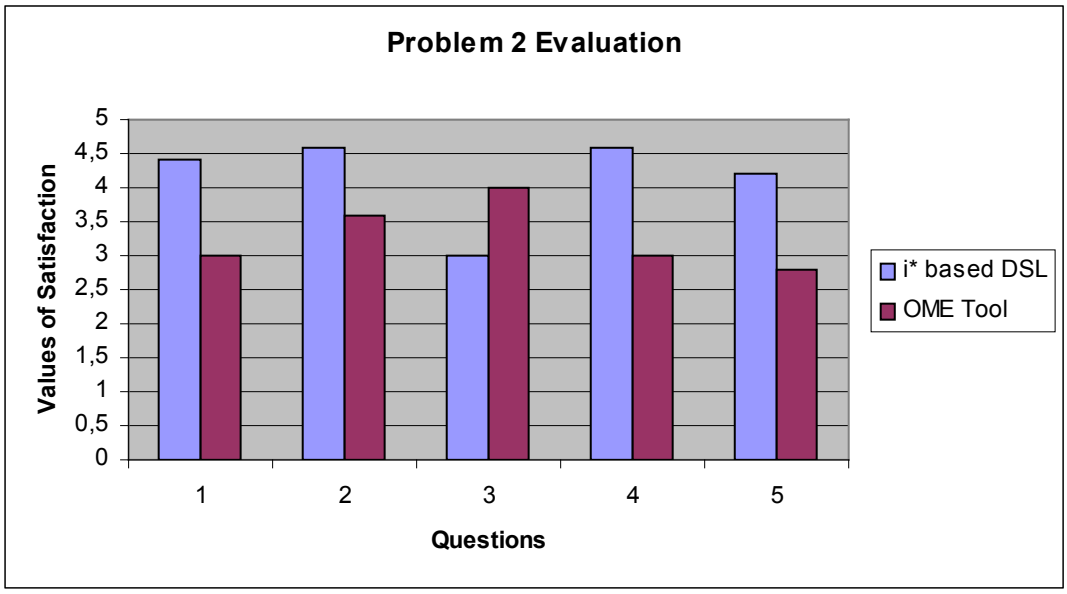

Figura 5.8. Gráfico com a análise dos dados obtidos no problema 2

No terceiro problema tinha-se como objectivo mostrar os elementos e ligações internas aos actores. Na figura 5.9 mostra-se o gráfico com os resultados da análise dos dados para este terceiro problema.

Os resultados que se obtiveram foram os esperados para este problema uma vez que tanto ao nível da sua resolução como ao nível da facilidade de utilização da ferramenta, os utilizadores consideraram a LDE desenvolvida mais fácil de usar para esta questão devido à possibilidade de se poder navegar livremente dentro dos actores e devido ao facto de os elementos internos e ligações internas estarem bem especificadas conseguindo assim resolver o

problema mais facilmente do que na ferramenta OME. Quanto à metodologia os utilizadores acharam que a ferramenta OME se aproximava mais da framework i* do que a LDE desenvolvida, esta resposta é justificável pois sendo a LDE uma nova ferramenta com algumas modificações tanto ao nível de funcionalidades como no aspecto gráfico (tendo havido no entanto a preocupação de manter esta ferramenta o mais próximo possível da framework i* original) é natural os utilizadores estarem mais familiarizados com a ferramenta OME uma vez que a usam há bastante tempo. Finalmente em relação às funcionalidades testadas e à gestão da complexidade, os utilizadores consideraram ao nível de funcionalidades a LDE mais completa devido à possibilidade de se poder navegar livremente dentro do actores e se poder expandir ou retrair os actores livremente e quanto à gestão de complexidade os utilizadores consideram que existe uma melhor gestão desta por parte da LDE uma vez que é possível expandir ou retrair os actores individualmente analisado assim apenas os actores que se quer analisar.

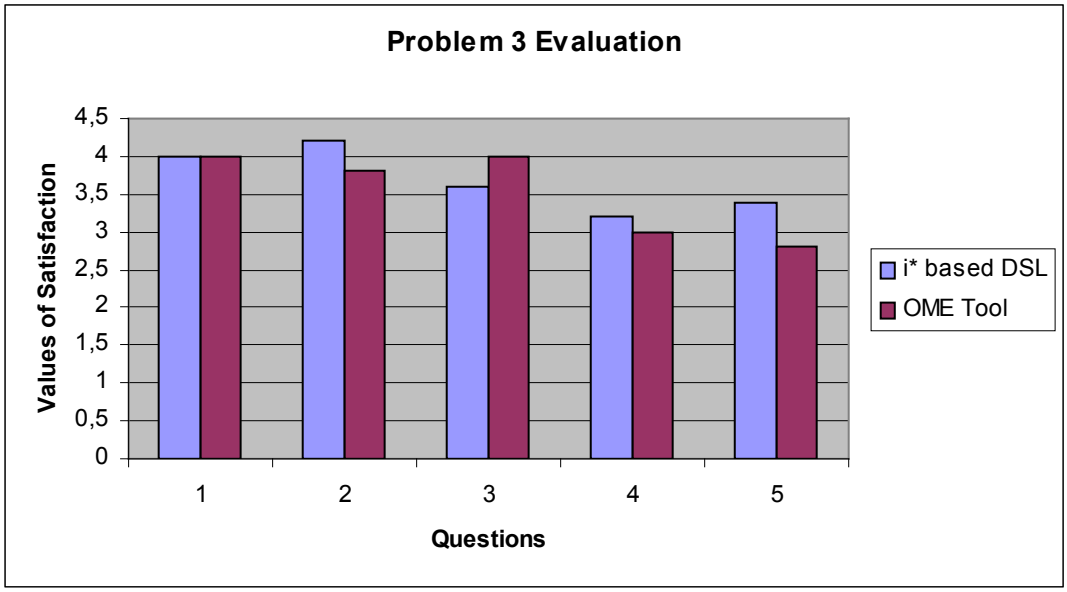

Figura 5.9. Gráfico com a análise dos dados obtidos no problema 3

No quarto problema tinha-se como objectivo mostrar as capacidades de expansão e retracção dos compartimentos e dos actores. Na figura 5.10 mostra-se o gráfico com os resultados da análise dos dados para este quarto problema.

Os resultados que se obtiveram foram os esperados para este problema uma vez que tanto ao nível da sua resolução como ao nível da facilidade de utilização da ferramenta, os utilizadores consideraram a LDE desenvolvida mais fácil de usar para esta questão devido à possibilidade de se poder expandir ou retrair quer os compartimentos quer os actores facilitando assim a resolução deste problema em comparação com a ferramenta OME que não tem os referidos compartimentos. Quanto à metodologia os utilizadores acharam que a

ferramenta OME se aproximava mais da framework i* do que a LDE desenvolvida, esta resposta é justificável pois foram introduzidas novas funcionalidades que não existiam na framework i* original como os compartimentos e a sua capacidade de expandir e retrair, deste modo é natural os utilizadores afirmarem que existe algum afastamento da metodologia por parte da LDE desenvolvida. Finalmente em relação às funcionalidades testadas e à gestão da complexidade, os utilizadores consideraram ao nível de funcionalidades a LDE mais completa com a introdução dos compartimentos e a possibilidade de expandir ou retrair os compartimentos e actores permitindo assim uma análise individual destes e quanto à gestão de complexidade os utilizadores consideram que existe uma melhor gestão desta por parte da LDE uma vez que ao se ter a possibilidade de se expandir ou retrair os compartimentos e actores pode-se analisar estes individualmente o que torna os modelos mais fáceis de ler e compreender.

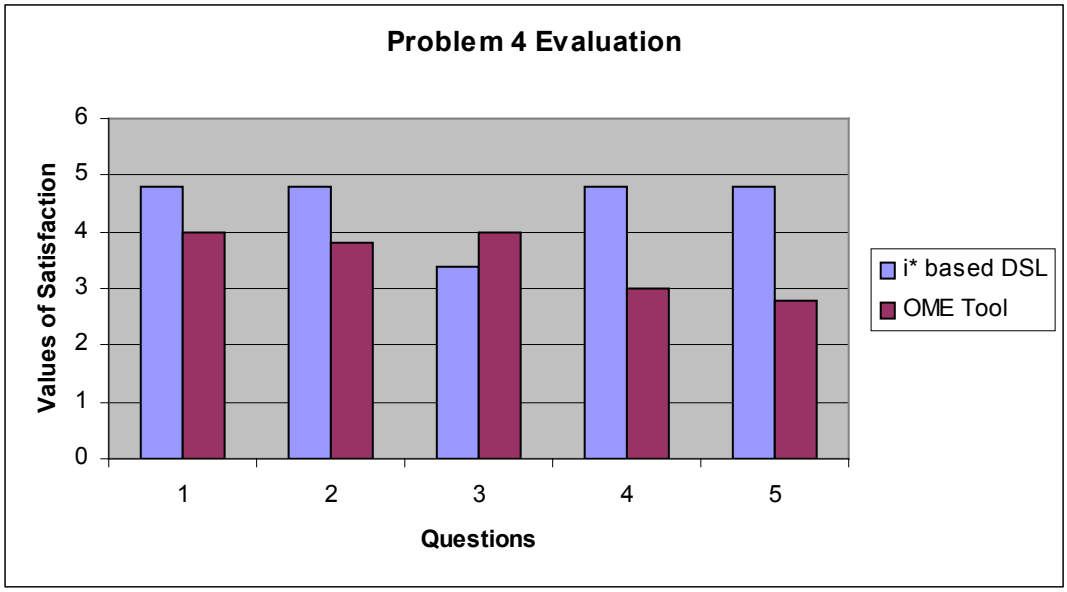

Figura 5.10. Gráfico com a análise dos dados obtidos no problema 4

5.6.3 Opiniões dos Utilizadores

Após os utilizadores terem concluído os quatro problemas apresentados no questionário e terem respondido às questões quantitativas, foram feitas algumas questões para saber quais os sentimentos que os utilizadores tinham ficado em relação à LDE. As opiniões que a seguir se vai mostrar foram obtidas usando as questões de parte aberta presentes no questionário.

Para as questões "Quais os principais pontos positivos que detectaram ao usar esta ferramenta" e "Que inovações considera que esta ferramenta trouxe" as respostas mais dadas foram:

- Possibilidade de expandir e colapsar compartimentos e actores
- Mais simples de utilizar em comparação com outras ferramentas
- Ferramenta bastante estável

Deste modo chegam-se às conclusões que pretendidas e que já se tinham chegado no capitulo 5.6.2: realmente os utilizadores que testaram esta ferramenta afirmam que esta é realmente mais simples de utilizar, introduz algumas novas funcionalidades bastante relevantes e acaba por ser uma ferramenta mais estável em termos de utilização e fiabilidade comparada com as existentes.

No entanto a LDE desenvolvida ainda contém alguns problemas menores de apresentação e interface pelo que foi também pedido aos utilizadores que dissessem que pontos menos positivos e que melhorias sugeriam para a ferramenta. Para as questões "Quais os principais pontos negativos que detectou ao usar esta ferramenta" e "Que sugere para que esta ferramenta possa ser melhorada" as respostas mais dadas foram:

- Alguma dificuldade na navegação interna dos actores
- Alguns bugs a nível da interface
- Ao retrair ou expandir elementos deviam sempre voltar à forma inicial

Estas sugestões dadas e pontos menos negativos identificados serão corrigidos em versões futuras desta ferramenta.

5.6.4 Discussão dos Resultados

Os pontos fulcrais que se devem reter dos dados obtidos são os seguintes:

- Dados Gerais – As médias gerais obtidas corroboram as hipóteses feitas em relação à ferramenta, nomeadamente a introdução de funcionalidades inovadoras, criação de mecanismos de gestão de complexidade dos modelos e a criação de uma ferramenta mais simples e acessível aos utilizadores.

- Dados Quantitativos – Para cada problema apresentado obtiveram-se dados que vieram corroborar as hipóteses feitas para cada problema em questão. Desde os testes comparativos destinados a verificar a funcionalidade de cada ferramenta, passando pelos testes de verificação da importância das novas funcionalidades introduzidas, até aos testes dos mecanismos de gestão da complexidade dos modelos.

- Opiniões dos utilizadores – A opinião geral dada pelos utilizadores que testaram a ferramenta foi bastante positiva e apesar de alguns pontos mais fracos terem sido detectados a nível geral considera-se que as inovações que esta ferramenta traz e a sua facilidade de utilização compensam as pequenas falhas que ainda existem a nível de interface e apresentação.

6. Conclusões

Esta tese de mestrado focou-se principalmente no estudo da framework i* [10, 11], com o objectivo de identificar as questões mais relevantes ainda em aberto e realizar um estudo comparativo entre as várias ferramentas que suportam esta framework [9, 13]. Para além desse estudo teve-se também como objectivo implementar uma LDE para essa framework após um estudo sobre as LDE e sobre as ferramentas que implementam essa tecnologia, com o propósito de escolher a mais adequada para a construção do protótipo aqui apresentado. A implementação de uma LDE que suporta-se a framework i* foi essencial, pois através dessa LDE formalizaram-se os modelos baseados na framework i* e tentou-se gerir a complexidade desses modelos usando técnicas de engenharia orientadas aos modelos.

Assim foi possível desenhar e desenvolver uma ferramenta que implementasse totalmente a framework i* onde a noção de uso de compartimentos foi introduzida para controlar a visualização das dependências entre actores. Para além disso garante-se que todos os modelos criados usando esta LDE são correctos de acordo com a descrição da sintaxe abstracta correspondente ao metamodelo usado para criar esta LDE.

Foi feito um estudo comparativo entra a LDE construída e a ferramenta OME. Tendo sido mostrado os aspectos em comum que as ferramentas possuem e mostraram-se ainda quais as inovações que a LDE trouxe.

Esta ferramenta foi testada por dez estudantes de mestrado de engenharia informática, de modo a que se pudesse obter dados quantitativos, ajudando assim à compreensão e ao suporte da abordagem tomada e que realmente a LDE construída trouxe benefícios quando comparada com outras ferramentas existentes.

Após a fase de avaliação e depois de se ter analisado os dados obtidos concluiu-se que a LDE construída realmente correspondeu às expectativas. Sendo assim após se ter analisado os dados e após se ter ouvido as opiniões dos indivíduos que realizaram os testes, pode-se concluir que realmente a LDE traz funcionalidades inovadoras, ajuda a reduzir a complexidade dos modelos criados e é mais fácil de utilizar em comparação com as outras ferramentas.

6.1 Trabalhos futuros

Em relação ao trabalho pretende-se investigar outras propostas de modelação orientadas a objectivos, como por exemplo o KAOS [20,21], para identificar que aspectos são comuns entre essas frameworks de modo a poder criar-se uma framework orientada aos objectivos que seja mais eficiente que as metodologias existentes.

7. Bibliografia

[1] Fernanda Alencar, Carla Silva, Márcia Lucena, Jaelson Castro, Emanuel Santos and Ricardo Ramos. Improving the understandability of i* models. Accepted for publication in the 10th International Conference on Enterprise Information Systems (ICEIS'08). 12 - 16, June 2008. Barcelona, Espanha.

[2] Carla Silva, João Araújo, Ana Moreira, Jaelson Castro, Fernanda Alencar, Ricardo Ramos. Organizational Architectural Styles Specification. In: Proceedings of the XI Jornadas de Ingeniería del Software y Bases de Datos (JISBD´06), 2006, Sitges. 2006. p.1 – 11. Barcelona, Espanha.

[3] Carlos Cares, Xavier Franch, Anna Perini and Angelo Susi. i*ML: an XML-based Interchange Format for i* Models. In: Third International i* Workshop (I*'08). February 11-12, 2008. Co-located at the IDEAS'08. Recife, Brasil.

[4] Claudia P. Ayala, Carlos Cares, Juan P. Carvallo, Gemma Grau, Mariela Haya, Guadalupe Salazar, Xavier Franch, Enric Mayol, Carme Quer. A Comparative Analysis of i*-Based Agent-Oriented Modeling Languages. In: Proceedings of SEKE, 2006. San Francisco Bay, USA.

[5] Fernanda Alencar, Jaelson Castro, Cleviton Monteiro, Ricardo Ramos, Emanuel Santos. Towards Aspectual i*. In: Third International i* Workshop (I*'08). February 11-12, 2008. Co-located at the IDEAS'08. Recife, Brasil.

[6] Márcia Lucena, Emanuel Santos, Carla Silva, Fernanda Alencar, Maria J. Silva, Jaelson Castro. Towards a Unified Metamodel for i*. In the Research Challenges in Information Science. June 3 - 6, 2008. Marrakech, Marrocos.

[7] Maria Jocelia Silva, Paulo Roberto Maciel, Rosa Cândida Pinto, Fernanda Alencar, Patrícia Tedesco, Jaelson Castro. Extracting the Best Features of Two Tropos Approaches for the Efficient Design of MAS. In: Proceedings of the 10th Iberoamerican Workshop on Requirements Engineering and Software Environments, 2007. p. 1-10. Isla de Margarita, Venezuela.

[8] Raimundas Matulevicius, Improving the Syntax and Semantics of Goal Modelling Languages. In: Third International i* Workshop (I*'08). Fevereiro 11-12, 2008. Co-located at the IDEAS'08. Recife, Brasil.

[9] Raimundas Matulevicius, Patrick Heymans, and Guttorm Sindre (2006), Comparing Goal-Modelling Tools With The RE-TOOL Evaluation Approach, In: *Information Technology And Control, Kaunas, Technologija,* 2006, Vol. 35A, No. 3, 276 - 284.

[10] Yu, Eric (1995), Modelling Strategic Relationships for Process Reengineering. Ph.D. thesis. Department of Computer Science. University of Toronto.

[11] Yu, Eric, Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering. In: Proceedings of the Third IEEE International Symposium on Requirements Engineering, 1997. Annapolis, MD, U.S.A.

[12] The GME page: http://www.isis.vanderbilt.edu/projects/gme/index.html

[13] The Eclipse/GMF page: http://wiki.eclipse.org/GMF_Documentation

[14] The LDE Tools page: http://msdn.microsoft.com/en-us/library/bb126235.aspx

[15] The i* wiki : http://i*.rwth-aachen.de/tiki-index.php

[16] The LDE wiki : http://en.wikipedia.org/wiki/Domain_Specific_Language

[17] The NFR wiki : http://en.wikipedia.org/wiki/NonFunctional_Requirements_framework_%28NFR%29

[18] Unified Modeling Language Version 2.0 (2005), Object Management Group (OMG). http://www.omg.uml

[19] Jean Bézivin, Guillaume Hillairet, Frédéric Jouault, Ivan Kurtev, William Piers (2005), Bridging the MS/DSL Tools and the Eclipse Modeling Framework, ATLAS Group (INRIA & LINA, University of Nantes), In: Proceedings of the International Workshop on Software Factories at OOPSLA 2005, San Diego, California, USA.

[20] Steven Kelly, Juha-Pekka Tolvanen (2008), Domain Specific Modeling Enabling Full Code Generation, Wiley-IEEE Computer Society Press.

[21] Alexei Lapouchnian (2005), Goal-Oriented Requirements Engineering: An Overview of the Current Research, Department of Computer Science University Of Toronto.

[22] L. Kolos-Mazuryk, P. van Eck, R. Wieringa (2006), A Survey of Requirements Engineering Methods for Pervasive Services, Centre for Telematics and Information Technology (CTIT) University Of Twente.

[23] Tiago Massoni, Sérgio Soares, Paulo Borba (2007), Requirements Health-Watcher version 2.0, In: Early Aspects at ICSE: Workshop in Aspect-Oriented Requirements Engineering and Architecture Design. Minneapolis, USA.

[24] Sérgio Soares, Eduardo Laureano, and Paulo Borba (2002), Implementing Distribution and Persistence Aspects with AspectJ, In: Proceedings of the 17th ACM conference on Object-oriented programming, systems, languages, and applications. Seattle, Washington, USA.

[25] Norman S. Murray, Norman W. Paton, Carole A. Goble, Joanne Bryce (1998), Kaleidoquery--a flow-based visual language and its evaluation, In: Proceedings of the working conference on Advanced visual interfaces. New York, USA.

Anexo 1: Questionário

De seguida serão apresentadas algumas questões a serem resolvidas através da LDE i*. Pede-se que tente resolver as questões apresentadas e que responda às perguntas seguintes com o máximo de veracidade.

Questão 1

1. Crie um actor "cliente" e um agente "banco".
2. Crie um actor "privado" e um actor "empresa" e ligue-os ao actor "cliente" usando uma ligação IsALink.
3. Crie um objectivo "depositar" e ligue-o ao cliente e ao banco, sendo o cliente o depender e o banco o dependee.
4. Crie um softgoal "segurança" e um softgoal "tempo de resposta" ligando-os ao cliente e ao banco, sendo o cliente o depender e o banco o dependee.
5. Crie um recurso "dinheiro" e ligue-o ao cliente e ao banco, sendo o cliente o dependee e o banco o depender.
6. Crie uma tarefa "depositar dinheiro" e ligue-o ao cliente e ao banco, sendo o cliente o depender e o banco o dependee.

Responda às seguintes perguntas em relação à questão 1:

Com que facilidade realizou esta questão?

☐	☐	☐	☐	☐
Pouca		Alguma		Muita

Em comparação com outras ferramentas que usam a metodologia i* achou esta ferramenta mais simples ou mais complicada de utilizar?

☐	☐	☐	☐	☐
Mais complicada		Igual		Mais Simples

Considera que esta ferramenta se aproxima da metodologia usada em outras ferramentas em relação a esta questão?

☐	☐	☐	☐	☐
Nenhuma		Alguma		Bastante

Considera que esta ferramenta trouxe alguma inovação à metodologia i* em relação a esta questão?

☐	☐	☐	☐	☐
Nenhuma		Alguma		Bastante

Considera que esta ferramenta melhora os métodos de análise em termos de escalabilidade em relação a esta questão?

☐	☐	☐	☐	☐
Pouca		Alguma		Muita

Questão 2

1. Elimine todos os elementos excepto os actores e agentes.
2. Crie um compartimento para softgoals de nome “SoftContainer1” e ligue-o ao cliente e ao banco, sendo o cliente o depender e o banco o dependee.
3. Adicione um softgoal “segurança” e um softgoal “tempo de resposta” ao compartimento SoftContainer1.
4. Crie um compartimento para elementos de nome “ElemContainer1” e ligue-o ao cliente e ao banco, sendo o cliente o depender e o banco o dependee.

5. Adicione um objectivo “depositar” ao compartimento ElemContainer1.
6. Adicione uma tarefa “depositar dinheiro” ao compartimento ElemContainer1.
7. Crie um compartimento para elementos de nome “ElemContainer2” e ligue-o ao cliente e ao banco, sendo o cliente o dependee e o banco o depender.
8. Adicione um recurso “dinheiro” ao compartimento ElemContainer2.

Responda às seguintes perguntas em relação à questão 2:

Com que facilidade realizou esta questão?

☐	☐	☐	☐	☐
Pouca		Alguma		Muita

Em comparação com outras ferramentas que usam a metodologia i* achou esta ferramenta mais simples ou mais complicada de utilizar?

☐	☐	☐	☐	☐
Mais complicada		Igual		Mais Simples

Considera que esta ferramenta se aproxima da metodologia usada em outras ferramentas em relação a esta questão?

☐	☐	☐	☐	☐
Nenhuma		Alguma		Bastante

Considera que esta ferramenta trouxe alguma inovação à metodologia i* em relação a esta questão?

☐	☐	☐	☐	☐
Nenhuma		Alguma		Bastante

Considera que esta ferramenta melhora os métodos de análise em termos de escalabilidade em relação a esta questão?

☐	☐	☐	☐	☐
Pouca		Alguma		Muita

Questão 3

1. Crie um objectivo "Aceitar Deposito" dentro do agente Banco.
2. Ligue esse objectivo ao objectivo Depositar contido no ElemContainer1 usando um ExternalLink.
3. Crie uma tarefa "Verificar Dados" dentro do agente Banco e ligue-a ao objectivo Aceitar Deposito usando um MeanEndLink.
4. Crie duas tarefas "Verificar Dados" e "Verificar Cliente" dentro do agente Banco e ligue-os à tarefa Verificar Dados usando um DecompositionLink.
5. Crie um softgoal "Segurança" dentro do agente Banco e ligue-o à tarefa Verificar Dados usando um ContributionLink SomePlus.

Responda às seguintes perguntas em relação à questão 3:

Com que facilidade realizou esta questão?

☐	☐	☐	☐	☐
Pouca		Alguma		Muita

Em comparação com outras ferramentas que usam a metodologia i* achou esta ferramenta mais simples ou mais complicada de utilizar?

☐	☐	☐	☐	☐
Mais complicada		Igual		Mais Simples

Considera que esta ferramenta se aproxima da metodologia usada em outras ferramentas em relação a esta questão?

☐	☐	☐	☐	☐
Nenhuma		Alguma		Bastante

Considera que esta ferramenta trouxe alguma inovação à metodologia i* em relação a esta questão?

☐	☐	☐	☐	☐
Nenhuma		Alguma		Bastante

Considera que esta ferramenta melhora os métodos de análise em termos de escalabilidade em relação a esta questão?

☐	☐	☐	☐	☐
Pouca		Alguma		Muita

Questão 4

1. Passe do modelo SR para o Modelo SD (retraindo o agente Banco) e analise este compartimento a compartimento.
2. Passe do modelo SD para o Modelo SR retraindo os compartimentos analisados anteriormente e expandindo o agente Banco.

Responda às seguintes perguntas em relação à questão 4:

Com que facilidade realizou esta questão?

☐	☐	☐	☐	☐
Pouca		Alguma		Muita

Em comparação com outras ferramentas que usam a metodologia i* achou esta ferramenta mais simples ou mais complicada de utilizar?

☐ ☐ ☐ ☐ ☐

Mais complicada Igual Mais Simples

Considera que esta ferramenta se aproxima da metodologia usada em outras ferramentas em relação a esta questão?

☐ ☐ ☐ ☐ ☐

Nenhuma Alguma Bastante

Considera que esta ferramenta trouxe alguma inovação à metodologia i* em relação a esta questão?

☐ ☐ ☐ ☐ ☐

Nenhuma Alguma Bastante

Considera que esta ferramenta melhora os métodos de análise em termos de escalabilidade em relação a esta questão?

☐ ☐ ☐ ☐ ☐

Pouca Alguma Muita

De seguida serão apresentadas algumas questões para dar a sua opinião, por favor dê cerca de três a quatro exemplos em cada questão.

Quais os principais pontos positivos que detectou ao usar esta ferramenta:

Quais os principais pontos negativos que detectou ao usar esta ferramenta:

Que inovações considera que esta ferramenta trouxe:

Que sugere para que esta ferramenta possa ser melhorada:

Anexo 2: Manual de Utilizador

1. Introdução

Neste manual de utilizador vão ser mostrados todos os passos necessários à correcta utilização da LDE i*.
Serão primeiro apresentados quais os requisitos necessários para instalar esta ferramenta, sendo apresentado de seguida a sua correcta instalação.
Nos capítulos seguintes serão apresentadas todas as acções que é possível fazer com a ferramenta, assim como os vários elementos e ligações constituintes de cada acção, sendo sempre apresentada uma breve descrição e exemplo de cada elemento e ligação, para melhor esclarecimento por parte do utilizador.
Por fim será apresentado um pequeno exemplo ilustrativo do funcionamento real da ferramenta, para melhor compreensão desta por parte do utilizador.

1.1 Requisitos

Para esta ferramenta funcionar correctamente é necessário ter previamente instalado os seguintes componentes:

- Ter instalado ou instalar a ferramenta Eclipse
- Ter instalado ou instalar os plugins do Eclipse EMF e GMF

Pode-se fazer o download da ferramenta eclipse na seguinte página:
http://www.eclipse.org/downloads/

Uma vez instalado o Eclipse ir a Help → Software Updates → Find and Install

Depois:

- Search for new features to install
- Calisto Discovery Site

Finalmente basta escolher instalar os seguintes plugins:

- Eclipse Modeling Framework (EMF)
- Graphical Modeling Framework (GMF)

Não esquecer de aceitar instalar todas as dependências, como mostrado na figura1.

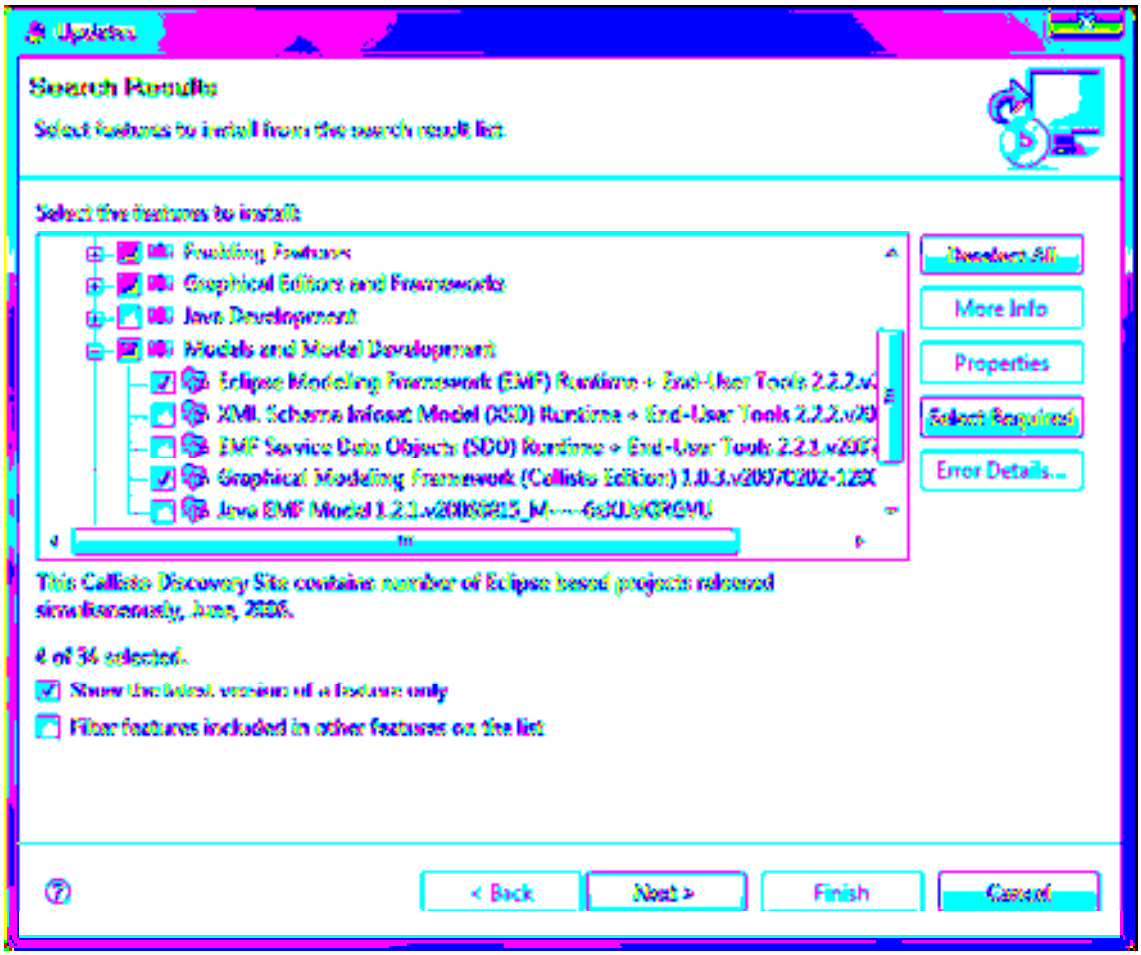

Figura 1: Plugins do Eclipse para instalar o EMF e o GMF

1.2 Instalar e Executar a LDE i*

Uma vez instalados os plugins do Eclipse referentes ao EMF e GMF vai ser necessário instalar a LDE i*. Para esse efeito é preciso no Eclipse ir a
File → Import.
Estando na janela referente ao Import deve-se escolher
General → Existing Projects into Workspace.
Finalmente indica-se o caminho no Browse do ficheiro iStarPrototipe.zip que contêm os ficheiros da LDE i*, selecciona-se todas as pastas correspondentes e carrega-se em Finish como mostrado na figura2.

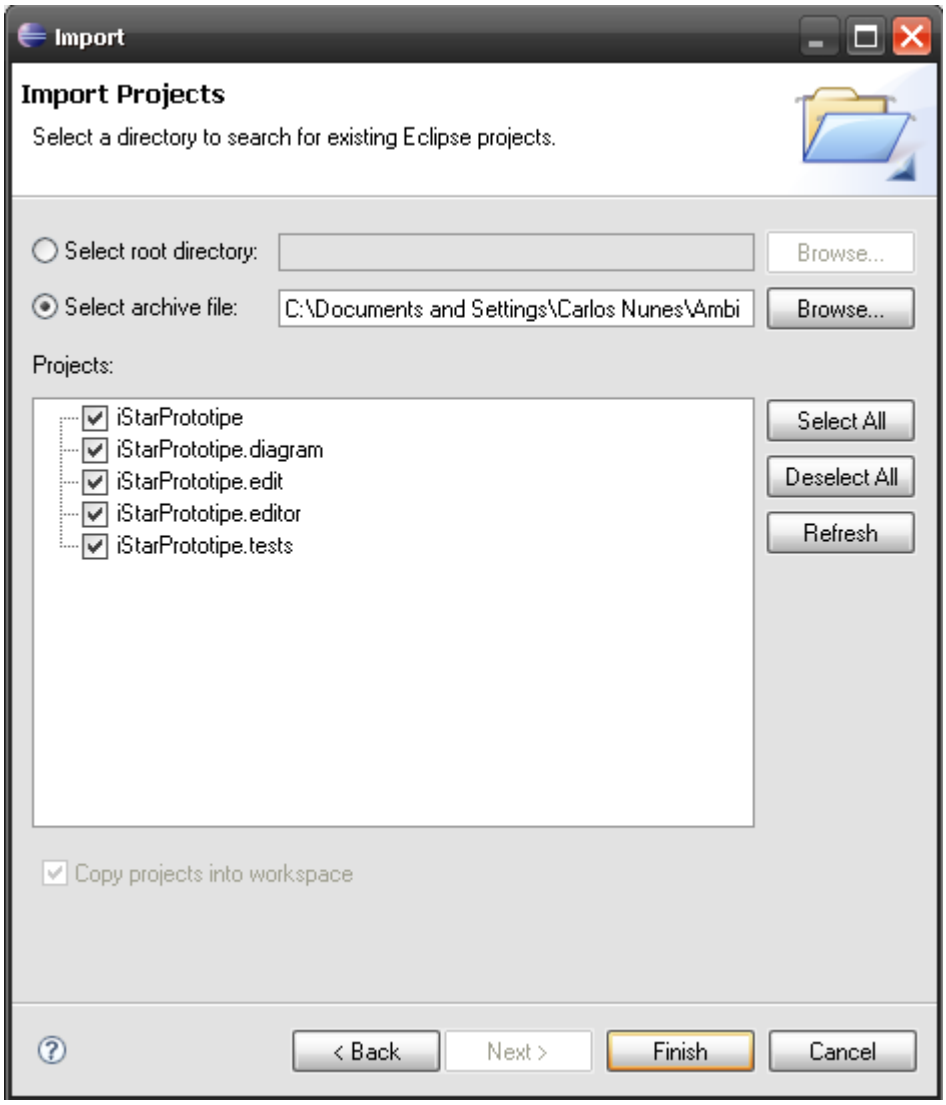

Figura 2: Selecção das pastas necessárias à LDE iStar

Para se executar a LDE i* é necessário no Eclipse ir a
Run → Open Run Dialog.
Por fim no Run Dialog faz-se duplo click em Eclipse Application e faz-se Run.
Na nova extensão do Eclipse vai ser necessário criar um novo projecto. Sendo assim escolhe-se New → Project → General → Project.
Agora neste novo projecto escolhe-se
New → Example → iStarPrototipe Diagram e dá-se o nome desejado, como é mostrado na figura3.

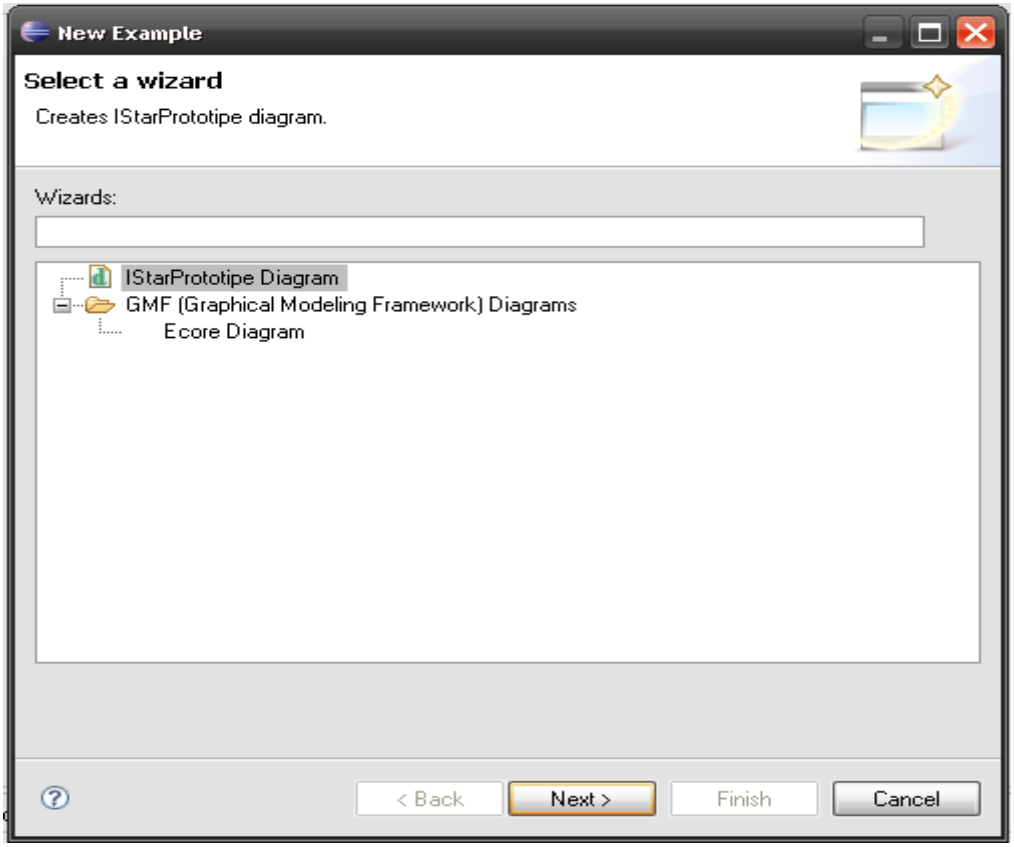

Figura 3: Selecção do projecto iStarPrototipe Diagram

Terminados estes procedimentos será lançado o editor que terá o aspecto apresentado na figura 4.

Figura 4: Editor do iStarPrototipe

2. Menu de Selecção

Nesta secção vai ser apresentado o menu de selecção e como está este dividido. Em cada componente constituinte do menu será dada uma breve explicação de cada componente, assim como, que elementos constituem essa componente.

2.1 Actors

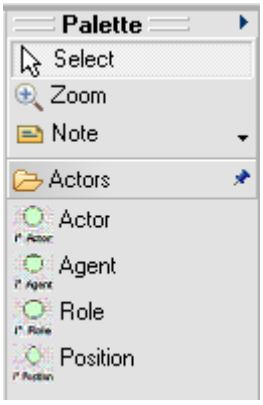

Figura 5: Menu Actores

Nesta componente do menu relativamente aos actores do sistema pode-se encontrar 4 tipos de elementos que podem ser usados.

São eles os actores intervenientes no sistema, os agentes intervenientes no sistema, as posições do sistema e os papéis do sistema.
Na figura 5 pode-se ver a componente do menu responsável por estes elementos.

2.2 Elementos Externos

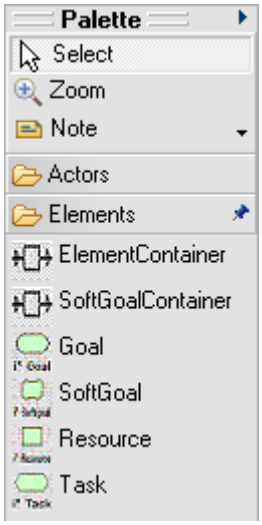

Figura 6: Menu Elementos Externos

Nesta componente do menu relativamente aos elementos externos do sistema pode-se encontrar 6 tipos de elementos que podem ser usados.
São eles os objectivos do sistema, os recursos do sistema, as tarefas do sistema, os compartimentos para guardar os três elementos referidos acima, os requisitos não funcionais do sistema aqui representados como softgoals e os compartimentos para guardar os softgoals.
Na figura 6 pode-se ver a componente do menu responsável por esses elementos.

2.3 Ligações Externas

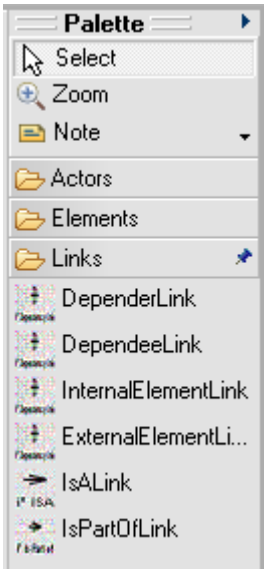

Figura 7: Menu Ligações Externas

Nesta componente do menu relativamente às ligações externas do sistema pode-se encontrar 6 tipos de ligações que podem ser usadas. Estas ligações são usadas para estabelecer ligações entre actores e outros elementos, ou entre actores entre si. São eles as ligações de dependência (DependerLink e DependeeLink), as ligações de especificação entre actores (IsALink e IsPartOfLink) e as ligações de elemntos externos para elementos internos (InternalElementLink e ExternalElementLink).
Na figura 7 pode-se ver a componente do menu responsável por esses elementos.

2.4 Elementos Internos

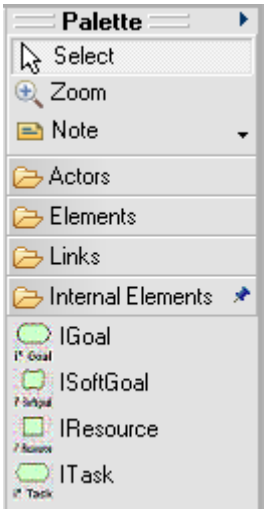

Figura 8: Menu Elementos Internos

Nesta componente do menu relativamente aos elementos internos do sistema, que vão estar contidos no interior dos actores, pode-se encontrar 4 tipos de elementos que podem ser usados.
São eles os objectivos do sistema, os recursos do sistema, as tarefas do sistema e os requisitos não funcionais do sistema aqui representados como softgoals.
Na figura 8 pode-se ver a componente do menu responsável por esses elementos.

2.5 Elementos Internos

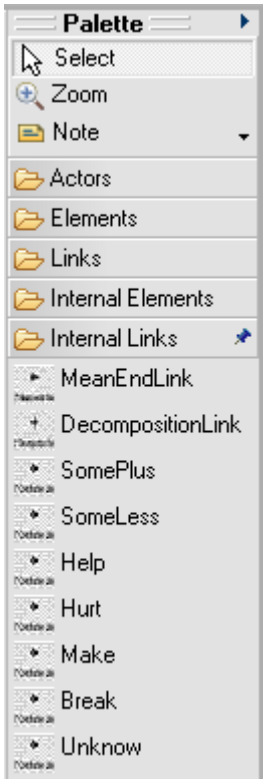

Figura 9: Menu Ligações Internas

Nesta componente do menu relativamente às ligações externas do sistema pode-se encontrar 9 tipos de ligações internas que podem ser usados. Estas ligações são apenas usadas na ligação de elementos contidos nos actores.
Na figura 9 pode-se ver a componente do menu responsável por esses elementos.

3. Utilização dos vários elementos

Nesta secção vai ser explicado para cada elemento a maneira correcta da sua utilização, estas explicações serão acompanhadas de exemplos ilustrativos para uma mais fácil compreensão.

3.1 Actors

Os actores, agentes, papéis e posições vão ser usados de forma idêntica, uma vez que são apenas variantes uns dos os outros conforme a situação que se quer modelar.
Ao criar-se um destes elementos aparece uma caixa de texto, nessa caixa de texto escreve-se o nome que se pretende dar ao actor. Estes elementos podem ser expandidos ou retraídos de modo a ficarem maiores ou menores, esta opção é bastante importante pois estes elementos contêm um repositório que lhes permite armazenar outros elementos e suas ligações, modelando assim o seu comportamento interno.
Na figura 10 pode-se ver um exemplo de um actor com o seu repositório.

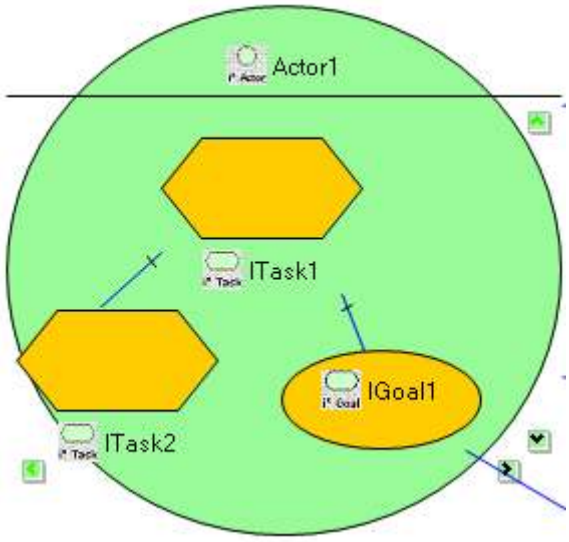

Figura 10: Actor com Repositório

Estes repositórios podem ser escondidos de modo a só se visualizar o seu interior quando necessário, para se efectuar essa acção basta clicar na seta do quanto superior esquerdo do repositório e caso se queira ver o repositório carrega-se de novo nessa mesma seta.
Na figura 11 pode-se ver um exemplo de como esconder o repositório.

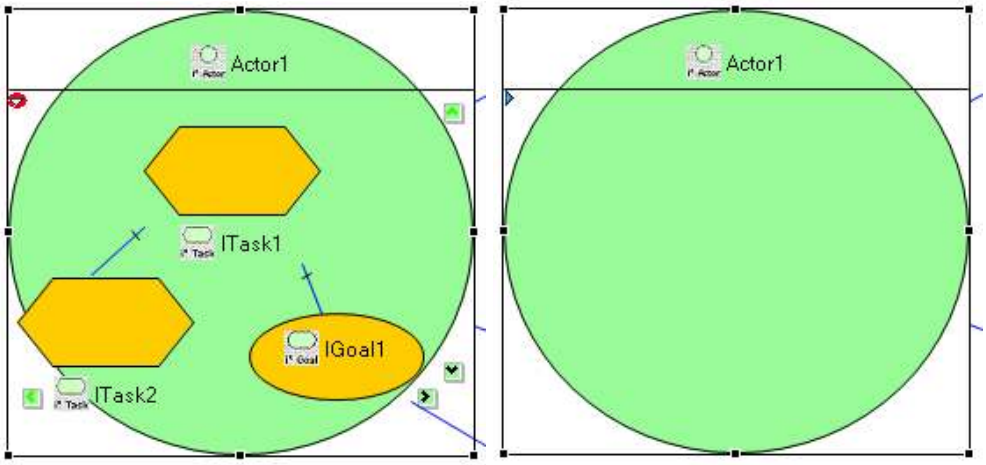

Figura 11: Actor com Repositório aberto e fechado

3.2 Elementos Externos

Os objectivos, softgoals, recursos e tarefas podem ser usados individualmente, assim como podem ser agrupados em repositórios especiais de modo a melhorar a legibilidade dos diagramas construídos.
Ao criar-se um destes elementos aparece uma caixa de texto, nessa caixa de texto escreve-se o nome que se pretende dar ao elemento criado.
Estes elementos não podem ser redimensionados, uma vez que são elementos atómicos e não vão conter outros elementos no seu interior.
Na figura 12 pode-se ver um exemplo de um objectivo ligado entre um agente e um role.

Figura 12: Objectivo ligado entre um agente e um role

Podem ainda ser criados dois repositórios para agrupamento destes elementos de modo a permitir uma melhor organização, leitura e entendimento do diagrama construído.
Existem dois tipos de repositórios, o repositório destinado a apenas softgoals e o repositório destinado aos restantes elemento (objectivos, recursos e tarefas).
Estes repositórios podem ser expandidos, retraídos e escondidos de modo a que se possa visualizar apenas os elementos desejados dentro do repositório.
Estas acções são realizadas da mesma forma que são realizadas nos repositórios dos actores descritos acima.

Na figura 13 pode-se ver um exemplo de um repositório para softgoals e na figura 14 pode-se ver um exemplo de um repositório para os restantes elementos.

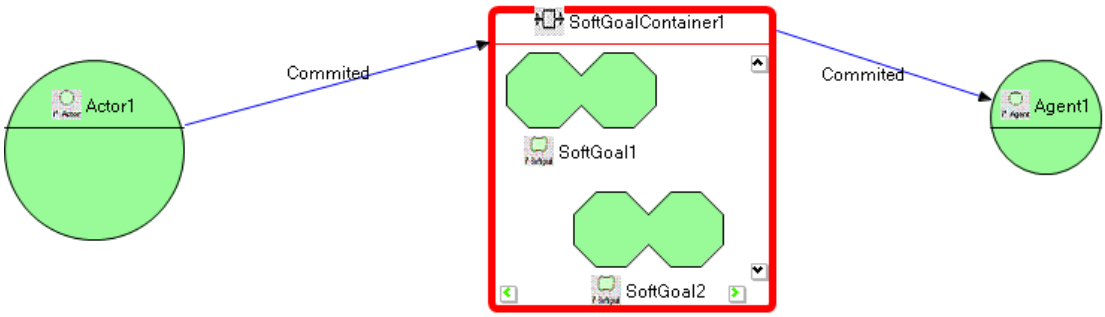

Figura 13: Repositório de softgoals com dois softgoals

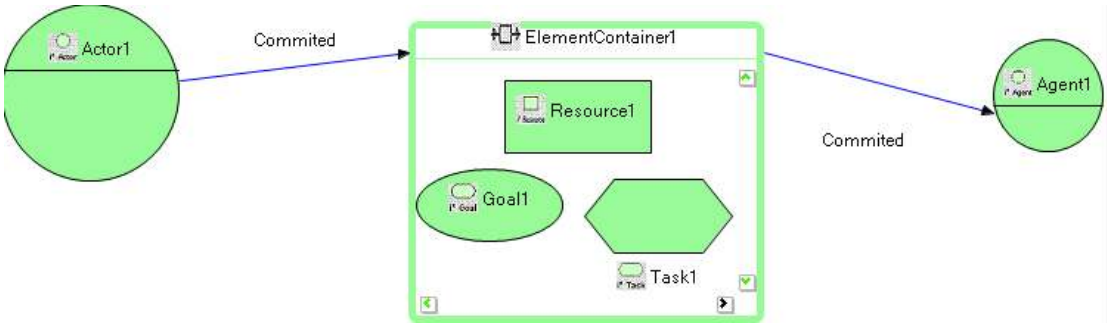

Figura 14: Repositório de elementos com um objectivo, um recurso e uma tarefa

3.3 Ligações Externas

As ligações DependerLink e DependeeLink são usadas para ligar actores a elementos externos.
Na ligação DependerLink o início da ligação tem de ser um actor e o seu final um elemento.
Na ligação DependeeLink o início da ligação tem de ser um elemento e o seu final um actor.
Nestas duas ligações vai ser criado automaticamente uma label a especificar qual o grau de importância da ligação, pode-se mudar esse grau seleccionando a label, nas propriedades na opção strenght pode-se mudar então a importância da ligação.
Na figura 15 pode-se ver um exemplo destas duas ligações e a alteração do grau de importância destas ligações.

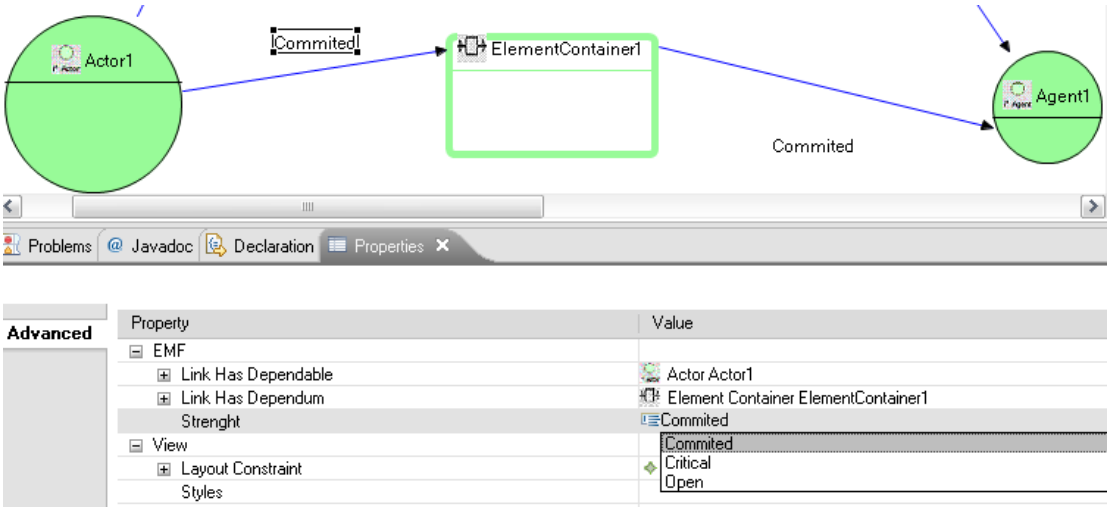

Figura 15: Alteração do grau de importância de uma ligação DependerLink

As ligações IsALink e IsPartOfLink são usadas para especificar actores usando outros actores.
Nestas duas ligações é necessário que o seu início e o seu fim sejam actores, não podendo no entanto o início e o fim serem o mesmo actor.
Na figura 16 pode-se ver um exemplo de uma ligação IsALink entre um agente e um actor.

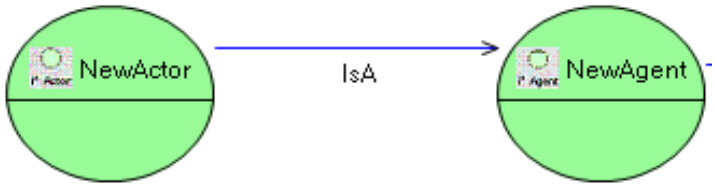

Figura 16: Ligação IsALink

As ligações InternalLink e ExternalLink são usadas para ligar elementos externos a elementos internos e vice-versa.
Na ligação InternalLink é necessário que o início da ligação seja um elemento interno e que o final da ligação seja um elemento externo.
Na ligação ExternalLink é necessário que o início da ligação seja um elemento externo e o final da ligação seja um elemento interno.
Estas ligações ao serem criadas aparecem duplicadas, isto deve-se a um bug do eclipse ainda não solucionado, bastando no entanto apagar uma das ligações.
Na figura 17 pode-se ver um exemplo de uma ligação InternalLink.

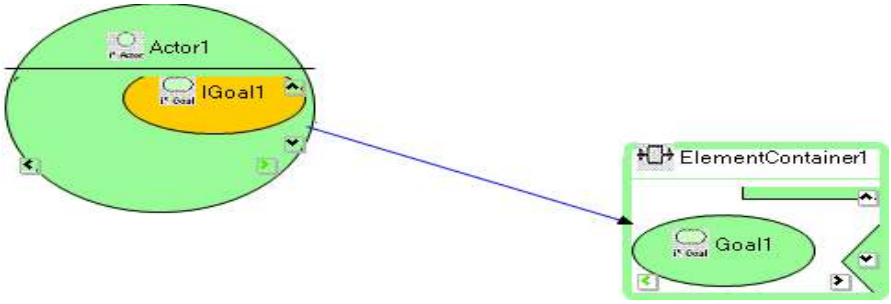

Figura 17: Ligação InternalLink

3.4 Elementos Internos

Os elementos internos sendo eles objectivos, softgoals, recursos e tarefas só podem ser utilizados no interior dos compartimentos dos actores.
Na figura 18 pode-se ver um exemplo de um actor com vários elementos internos dentro dele.

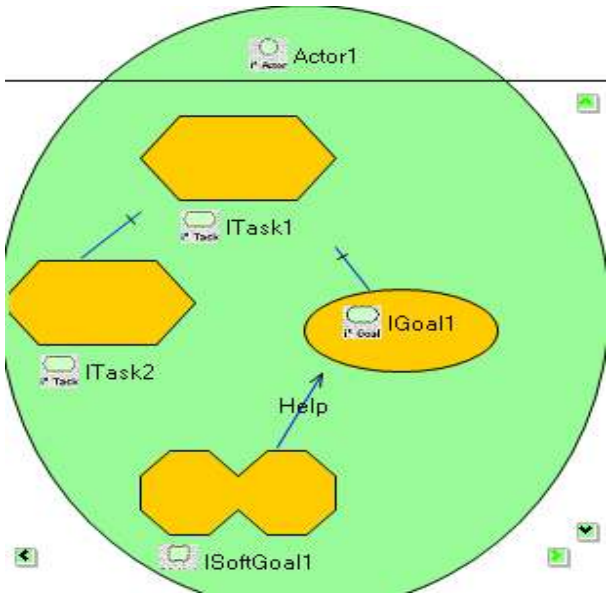

Figura 18: Elementos Internos dentro de um actor

3.5 Ligações Internas

A ligação interna MeanEndLink é utilizada para ligar dois elementos, de modo a descrever como o elemento que é ligado por outro funciona. Para esta ligação ser executada correctamente é necessário que o elemento inicial seja diferente do elemento final.
Na figura 19 pode-se ver um exemplo de uma ligação MeanEndLink entre dois objectivos internos.

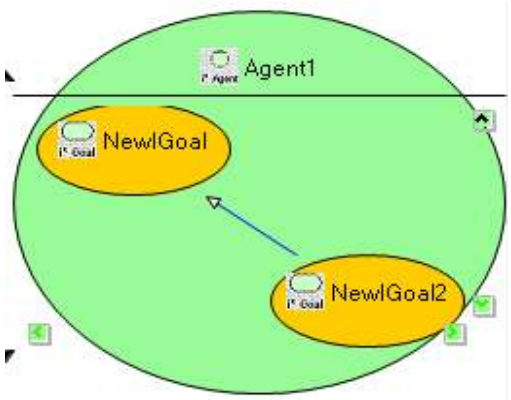

Figura 19: Ligação MeanEndLink

A ligação interna DecompositionLink é utilizada para decompor uma tarefa interna um vários elementos. Para esta ligação ser executada correctamente é necessário que o elemento inicial da ligação seja uma tarefa e o elemento final seja qualquer elemento excepto a tarefa inicial.
Na figura 20 pode-se ver um exemplo de uma ligação DecompositionLink entre uma tarefa e outra tarefa e um objectivo.

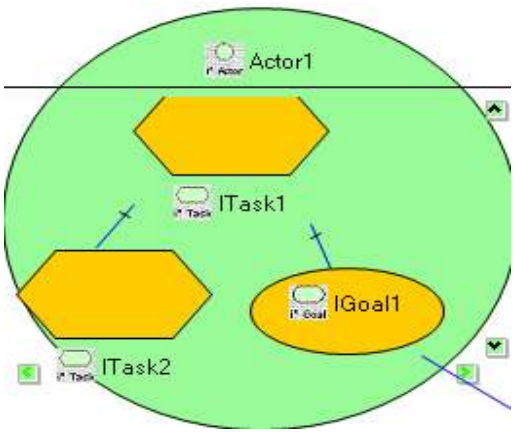

Figura 20: Ligações DecompositionLink

A ligação interna ContributionLink divide-se em sete especificações desta ligação (SomePlus, SomeLess, Help, Hurt, Make, Break, Unknow), estas ligações servem para mostrar as contribuições positivas ou negativas que um determinado softgoal pode fazer a um determinado elemento. Para que estas ligações sejam feitas correctamente é necessário que o elemento inicial da ligação seja um softgoal e que o elemento final seja qualquer elemento excepto o softgoal inicial.
Na figura 21 pode-se ver um exemplo de uma ligação Help entre um softgoal e um objectivo.

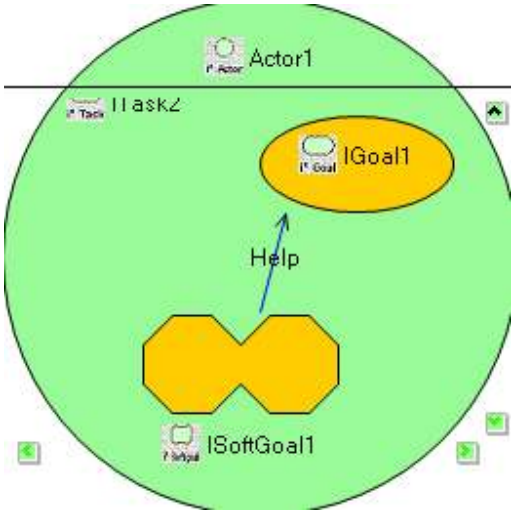

Figura 21: Ligação ContributionLink (Help)

Todas estas ligações apenas podem ser realizadas dentro dos compartimentos dos actores entre elementos internos contidos nesses compartimentos.
Estas ligações ao serem criadas aparecem duplicadas, isto deve-se a um bug do eclipse ainda não solucionado, bastando no entanto apagar uma das ligações.