



**Unioeste - Universidade Estadual do Oeste do Paraná**  
**CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS**  
Colegiado de Ciência da Computação  
*Curso de Bacharelado em Ciência da Computação*



**Desenvolvendo e Integrando um Editor i\* ao JGOOSE**

*Leonardo Pereira Merlin*

**CASCADEL**  
**2013**

**Leonardo Pereira Merlin**

**Desenvolvendo e Integrando um Editor i\* ao JGOOSE**

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação, do Centro de Ciências Exatas e Tecnológicas da Universidade Estadual do Oeste do Paraná - Campus de Cascavel

Orientador: Prof. Victor Francisco Araya Santander

CASCADEL  
2013

**Leonardo Pereira Merlin**

## **Desenvolvendo e Integrando um Editor i\* ao JGOOSE**

Monografia apresentada como requisito parcial para obtenção do Título de Bacharel em Ciência da Computação, pela Universidade Estadual do Oeste do Paraná, Campus de Cascavel, aprovada pela Comissão formada pelos professores:

---

Prof. Victor Francisco Araya Santander  
Colegiado de Ciência da Computação,  
UNIOESTE

---

Prof. Ivonei Freitas da Silva  
Colegiado de Ciência da Computação,  
UNIOESTE

---

Prof. Elder Elisandro Schemberger  
Colegiado de Ciência da Computação,  
UNIOESTE

Cascavel, 31 de maio de 2013

*Oscar José Merlin Júnior, dedico este trabalho a  
você, meu irmão e minha referência.*

*"Se você pensa que pode ou se pensa que não pode,  
de qualquer forma você está certo."  
Henry Ford*

## AGRADECIMENTOS

Em primeiro lugar, eu gostaria de agradecer ao meu orientador, professor Victor Francisco Araya Santander, pelos conselhos, além do apoio, disponibilidade e interesse no meu trabalho. Apresentou-me oportunidades que eu jamais acreditava que teria novamente. E, além de acreditar nos resultados, sempre me inspirou coragem e ânimo para cumprir meus objetivos.

Também gostaria de agradecer aos outros membros da banca examinadora, os professores Ivonei Freitas da Silva e Elder Elisandro Schemberger. Obrigado por analisar e avaliar cuidadosamente o meu trabalho. Durante as apresentações, contribuíram significativamente com questionamentos e sugestões, de fato, relevantes. E não distante desses, meus sinceros agradecimentos a todos os acadêmicos integrantes do grupo de estudo do Laboratório de Engenharia de Software (LES). Dentre esses, um agradecimento especial aos companheiros Diego Peliser, Leonardo Zannotto Baggio e Rodrigo Trage. Obrigado pelas apresentações e discussões sobre temas de destaque na área de Engenharia de Software. Espero poder retribuir e contribuir para com o grupo.

Quanto aos colegas de moradia, Lucas Inácio, Bruno Belorte, Marcos Schmitt, Nicolas Zaro e Julian Ruiz Diaz, obrigado pela tolerância e bons momentos.

Não posso deixar de agradecer os amigos Fernando Dal Bello, Amadeu Paixão e Felipe Carminati pelas experiências profissionais, oportunidades de um trabalho em equipe bem realizado e um amadurecimento pessoal memorável (nada que uma boa trilha sonora não resolva). Em especial, agradeço ao Carlos Henrique de França, que me ajudou a esclarecer e aguçar minhas pesquisas, além das críticas em numerosas versões de documentos técnicos.

A minha família, pelos conselhos (não ignorados), pelo auxílio e suporte de diversas maneiras. E, Lah (Larissa Torquato de Oliveira e família), este lugar também é de vocês.

A minha companheira, Jamile Merlin, por participar de mais esta fase da minha vida.

À todos, meu sincero agradecimento!

# Lista de Figuras

2.1	Exemplo de relações entre atores . . . . .	8
2.2	Notação de Ator (a), Agente (b), Posição (c) e Papel (d). . . . .	9
2.3	Tipos de associações entre atores (AACHEN, 2013). . . . .	10
2.4	Exemplo de Relação de Dependência ( <i>Dependder -&gt; Dependum -&gt; Dependee</i> ) .	11
2.5	Exemplos de ligação de dependência. . . . .	12
2.6	Exemplos de fronteira do ator. . . . .	13
2.7	Exemplo de ligação meio-fim. . . . .	13
2.8	Exemplo de ligação de decomposição. . . . .	14
2.9	Ligações de contribuição (AACHEN, 2013). . . . .	14
2.10	Exemplo de modelagem com a ferramenta OME3 (arquivo "Meeting-Schedule.tel") . . . . .	18
3.1	Arquitetura e Fluxo de processamento da JGOOSE. . . . .	20

# Lista de Abreviaturas e Siglas


API	<i>Application Programming Interface</i>
DFD	<i>Diagrama de Fluxo de Dados</i>
EMF	<i>Eclipse Modelling Framework</i>
ES	<b>Enngenharia de Software</b>
GOOSE	<i>Goal Into Object Oriented Standard Extension</i>
GUI	<i>Graphical User Interface</i>
IDE	<i>Integrated Development Environment</i>
ITU	<i>International Telecommunication Union</i>
<b>JGOSOE</b>	<i>Java Goal Into Object Oriented Standard Extension</i>
OME	<i>Organization Modelling Environment</i>
ORM	<b>Object/Relational Mapping</b>
POM	<i>Project Object Model</i>
POO	<i>Paradigma Orientado a Objetos</i>
SD	<i>Modelo de Dependências Estratégicas</i>
SR	<i>Modelo de Razões Estratégicas</i>
UML	<i>Unified Modeling Language</i>
XMI	<i>Xml Metadata Interchange</i>
XML	<i>eXtensible Markup Language</i>





# Sumário

<b>Lista de Figuras</b>	<b>vii</b>
<b>Lista de Abreviaturas e Siglas</b>	<b>viii</b>
<b>Sumário</b>	<b>ix</b>
<b>Resumo</b>	<b>xi</b>
<b>1 Introdução</b>	<b>1</b>
1.1 Contexto . . . . .	1
1.2 Motivação . . . . .	3
1.3 Proposta . . . . .	4
1.4 Contribuições Esperadas . . . . .	4
1.5 Estrutura do Trabalho . . . . .	5
<b>2 Framework i*, Variações e Ferramentas</b>	<b>6</b>
2.1 O Framework i* . . . . .	6
2.1.1 Modelo de Dependências Estratégicas . . . . .	8
2.1.2 Modelos de Razões Estratégicas (SR) . . . . .	12
2.1.3 iStarML . . . . .	15
2.2 Variações Baseadas no Framework i* . . . . .	16
2.2.1 i* Wiki . . . . .	16
2.2.2 <i>Goal-Oriented Requirements Language</i> (GRL) . . . . .	16
2.2.3 Tropos . . . . .	16
2.3 Ferramenta OME . . . . .	17
2.4 Considerações Finais do Capítulo . . . . .	18
<b>3 JGOOSE</b>	<b>19</b>
3.1 Visão Geral . . . . .	19

3.2	Histórico de Versões . . . . .	19
3.3	Projeto e Arquitetura . . . . .	20
3.4	Considerações Finais do Capítulo . . . . .	21
<b>4</b>	<b>Proposta</b>	<b>22</b>
4.1	Visão Geral . . . . .	22
4.1.1	Conceitos e Considerações Iniciais . . . . .	22
4.2	Projeto e Arquitetura . . . . .	22
4.2.1	Modelagem i* . . . . .	22
4.2.2	Arquitetura Modular . . . . .	22
4.2.3	Maven . . . . .	22
4.3	Desenvolvimento . . . . .	22
4.4	Recursos . . . . .	22
4.4.1	Internacionalização (i18n) . . . . .	22
4.4.2	UndoManager . . . . .	22
4.4.3	API iStarML . . . . .	22
4.5	Testes e Validação . . . . .	22
4.5.1	Testes Unitários . . . . .	22
4.6	Considerações Finais do Capítulo . . . . .	22
<b>5</b>	<b>Estudo de Caso</b> 	<b>23</b>
5.1	Resultados e Análises . . . . .	23
5.2	Considerações Finais do Capítulo . . . . .	23
<b>6</b>	<b>Considerações Finais</b>	<b>24</b>
6.1	Contribuições . . . . .	24
6.2	Trabalhos Futuros . . . . .	24
<b>A</b>	<b>iStarML API</b>	<b>25</b>
A.1	Introdução . . . . .	25
A.2	Projeto e Arquitetura . . . . .	25
A.3	Documentação . . . . .	25
	<b>Referências Bibliográficas</b>	<b>26</b>

# Resumo

Em discussões pertinentes à engenharia de requisitos, tanto em âmbito acadêmico quanto industrial, ressalta-se que um dos principais desafios da área consiste na integração de modelos organizacionais às demais etapas do processo de engenharia de requisitos.

Alguns trabalhos relacionados já apresentaram técnicas para a realização dessa integração. Nesse sentido, em 2006 foi desenvolvida uma solução computacional denominada JGOOSE (do inglês Java Goal Into Object Oriented Standard Extension), uma ferramenta de auxílio a engenharia de requisitos que proporciona a automatização do mapeamento de modelos e diagramas do framework *i\** para casos de uso UML.

Apesar das atualizações e melhorias realizada por outros pesquisadores e membros do grupo do Laboratório de Engenharia de Software (LES) da Universidade Estadual do Oeste do Paraná (UNIOESTE - Campus Cascavel/PR), a ferramenta ainda possui uma dependência crítica quanto a elaboração dos dados de entrada. Ou seja, nas condições atuais é necessário instalar alguma ferramenta específica para produzir o arquivo no formato TELOS.

Desta forma, este trabalho consiste no projeto e desenvolvimento de um editor gráfico de modelos organizacionais *i\** integrado à ferramenta JGOOSE, visando melhorar suas funcionalidades e diminuir a necessidade de outros softwares para este fim. Uma característica importante desse novo editor é o suporte à especificação iStarML - um formato de arquivo baseado em XML para representação de modelos *i\**.

**Palavras-chave:** JGOOSE, iStarML, *i\** Framework, Modelagem Organizacional, Engenharia de Requisitos.

# Capítulo 1

## Introdução

Este primeiro capítulo tem como objetivo a apresentação geral do trabalho. É realizada a contextualização e delimitação da pesquisa ao escopo da Engenharia de Software, bem como são destacados os principais objetivos da proposta. Apresenta-se inicialmente, na seção 1.1, o contexto sobre as ferramentas de modelagem organizacional e suas contribuições na área da Engenharia de Requisitos, destacando a influência dessas ferramentas no desenvolvimento de produtos de qualidade. Na seção 1.2, são apresentadas as principais influências e motivações para a realização do trabalho. Em seguida, na seção 1.3, é apresentada a proposta sob uma visão geral e os objetivos norteadores da pesquisa. Na seção 1.4, descreve-se as contribuições esperadas após a finalização deste trabalho. Por fim, na seção 1.5, é apresentada a estrutura geral e a organização do restante desta monografia.

### 1.1 Contexto

Muitas são as opções de ferramentas e técnicas para engenheiros de requisitos que visam auxiliar na construção de modelos organizacionais i\* (AACHEN, 2013) (GRAU *et al.*, 2006). Essas ferramentas, normalmente do tipo CASE (do inglês, *Computer-Aided Software Engineer*)<sup>1</sup> (CASE, 1985) têm por objetivo tanto o aumento da produtividade nos processos da Engenharia de Software quanto a melhoria da qualidade final dos softwares.

A área de Engenharia de Requisitos (ER), subárea da Engenharia de Software e responsável por diversas atividades que abrangem os processos de análise, elicitação, especificação, avaliação, ajuste, documentação e evolução dos requisitos de um sistema computacional, é vista como

---

<sup>1</sup>Ferramentas CASE, é toda e qualquer ferramenta baseada em computador que auxilie nas atividades de desenvolvimento de software.

uma das mais críticas para o sucesso e qualidade de um projeto de software (KOTONYA; SOMMERVILLE, 1998). Pesquisas pertinentes à ER, tanto em âmbito acadêmico quanto industrial, apontam a falta de um entendimento adequado da organização por parte dos responsáveis pela elaboração do documento de requisitos como sendo uma das principais falhas no processo de especificação dos requisitos (LAMSWEERDE, 2007).

Para tentar diminuir os problemas relacionados as fases iniciais do projeto, pesquisas recentes mostram que a comunidade tem buscado estabelecer e utilizar padrões de técnicas, métodos e ferramentas para tratar especificamente da fase inicial de desenvolvimento de software (SCHNEIDER; BERENBACH, 2013). Pensando nisso, têm-se investido esforços no processo de modelagem organizacional. Este tipo de modelagem visa prover recursos que permitam modelar as intenções, relacionamentos e motivações entre membros de uma organização (MASON, 1997). Dentre as técnicas de modelagem organizacional, destaca-se a *i\**, proposto por (YU, 1993), uma técnica que utiliza a orientação a agentes (YU, 2001) com enfoque tanto nos desejos e intenções desses agentes, quanto suas dependências (AACHEN, 2013) (YU, 1997). Mais detalhes sobre esta técnica e suas variações serão discutidos no capítulo 2.

Pensando em auxiliar no processo de desenvolvimento de software, alguns trabalhos foram propostos com o intuito de realizar o mapeamento de modelos do *framework i\** para diagramas da UML (do inglês *Unified Modeling Language*). Dentre esses trabalhos, destaca-se o trabalho de Santander (SANTANDER, 2002), que propõe a derivação em casos de uso UML a partir de modelos do *framework i\**. Para apoiar esse processo de derivação, foi desenvolvida a ferramenta JGOOSE (Java Goal Into Object Oriented Standard Extension). Inicialmente apresentada como GOOSE em (PEDROZA *et al.*, 2004) e (BRISCHKE, 2005), em seguida, melhorada e apresentada como JGOOSE por (VICENTE *et al.*, 2009), passou também por melhorias com (BRISCHKE *et al.*, 2012) e, atualmente, está sendo aprimorada por (PELISER, 2013). Consiste numa ferramenta que, seguindo algumas diretrizes e passos, mapeia de forma automática os diagramas *i\** para casos de uso UML. Ou seja, dado como entrada os modelos *i\**, no formato de arquivo TELOS (MYLOPOULOS *et al.*, 1990) (KOUBARAKIS *et al.*, 1989), a ferramenta consegue gerar conforme o template proposto em (COCKBURN, 2001) os casos de uso UML com um bom nível de detalhamento.

Porém, a ferramenta JGOOSE ainda não possui funcionalidades para a produção dos ar-

Motivação ?

quívos de entrada da ferramenta. Ainda existe a dependência da ferramenta OME ou, mais especificamente, ao formato de arquivo TELOS. Nesse contexto, percebe-se a necessidade de se desenvolver um editor de modelos i\* integrado à ferramenta JGOOSE, bem como implementar o suporte à especificação do formato de arquivo iStarML (CARES *et al.*, 2007), uma formato em XML para representação de modelos i\* com o propósito de servir como um intercâmbio entre os outros meta-modelos existentes (COLOMER *et al.*, 2011).

## 1.2 Motivação

Referenciar!

A área de Engenharia de Requisitos **está em crescimento e destaque por impactar de forma tão significativa nos resultados finais de um projeto de software.** Desta forma, é válido o investimento de esforços para a melhoria de métodos, técnicas ou ferramentas que auxiliem os profissionais da área a aprimorar seu trabalho de forma eficiente. O i\* é a base da GRL (*Goal-oriented Requirements Language* ou Linguagem de Requisitos Orientada a Objetivos), que junto à UCM<sup>2</sup> constituíram a URN<sup>3</sup>, que passou a ser adotada como um padrão internacional, em novembro de 2008, pela ITU (*International Telecommunication Union*) (AMYOT, 2003) (AMYOT; MUSSBACHER, 2003). Além de se tratar de um padrão internacional, é a técnica de modelagem já justificada pela JGOOSE e seus usuários já estão familiarizados com os conceitos do *mework*.

Motivação Acadêmica! (e a indústria?)

Além disso, a ferramenta de que trata este trabalho é frequentemente usada por acadêmicos do curso de Ciência da Computação da Universidade Estadual do Oeste do Paraná - Campus Cascavel. Todos os anos, alunos se debatem com problemas apresentados por outros softwares de modelagem i\*. Os questionamentos mais comuns estão relacionados à usabilidade e integridade das ferramentas. Isso acarreta em oportunidades para novas soluções computacionais se apresentarem.

ADICIONAR: motivação da indústria!

~~Outro fator, não menos importante, é o gosto pessoal pela área de projeto e desenvolvimento de software. Isto ajudou na tomada de decisão quanto ao foco da pesquisa, bem como resultou na implementação de uma API para o iStarML (Veja o apêndice A).~~

<sup>2</sup>UCM - *Use Case Maps*, em português: "Mapas de Caso de Uso". Uma técnica de engenharia de software baseada em cenários para descrever relacionamentos entre um ou mais casos de uso.

<sup>3</sup>URN - *User Requirements Notation*, em português: "Notação Requisitos de Usuário". Notação destinada a elicitação, análise, especificação e validação de requisitos.

### 1.3 ~~Proposta~~ ← OBJETIVOS

A ferramenta JGOOSE, no escopo do seu propósito, já atende as principais necessidades do engenheiro de requisitos. Porém, ainda existe a dependência da ferramenta mencionada (OME) para elaborar os modelos organizacionais e exportá-los em arquivo TELOS.

Dessa forma, o objetivo geral deste trabalho consiste em aumentar os recursos e funcionalidades da ferramenta JGOOSE através do desenvolvimento de uma nova interface para edição de modelos do framework i\* integrada ao JGOOSE. Ou seja, prover aos usuários da ferramenta JGOOSE uma interface gráfica rica em recursos que facilitem o trabalho de modelagem organizacional, visando diminuir a necessidade de usar outros softwares para esse fim.

Como objetivos específicos, têm-se:

- Estudar sobre as ferramentas de modelagem organizacional i\* disponíveis à comunidade.
- Estudar sobre o framework i\*, bem como suas variações.
- Estudar o formato de arquivo iStarML e incorporá-lo à ferramenta como o formato de arquivo padrão.
- Realizar um estudo sobre a evolução histórica da ferramenta JGOOSE e analisar sua arquitetura na versão 2013.
- Realizar um estudo de caso, usando a ferramenta proposta, a fim de validar as principais funcionalidades da ferramenta na versão final.

### 1.4 Contribuições Esperadas

Após a finalização deste trabalho, deseja-se uma maior independência para a JGOOSE e, consequentemente, seus usuários, além de aumentar o destaque na comunidade e promover a adoção da ferramenta para fins de modelagem i\*. Por fim, espera-se uma contribuição significativa **diante das ferramentas da comunidade i\***, trazendo um reconhecimento para o grupo de pesquisa do LES e todos os envolvidos no desenvolvimento e <sup>melhorias</sup> ~~progresso~~ da JGOOSE. Além disso, seria uma nova ferramenta no quadro comparativo do i\* Wiki (AACHEN, 2013).

## 1.5 Estrutura do Trabalho

Basicamente, o restante deste trabalho encontra-se organizado da seguinte maneira: Nos capítulos 2 e 3 são apresentados alguns fundamentos teóricos necessários para uma melhor compreensão da área de estudo de que trata este trabalho. No capítulo 2, os conceitos básicos e as características do *Framework i\**, bem como suas variações, são apresentados. Já no capítulo 3, após um estudo sobre a evolução histórica do software JGOOSE, uma análise e discussão detalhada de sua arquitetura, na versão 2013, é realizada. No capítulo 4, a proposta é detalhada através de uma visão geral do projeto e arquitetura da nova interface. Também é apresentada uma discussão sobre os principais recursos disponíveis aos usuários. Em seguida, no capítulo 5 é apresentado um estudo de caso, mostrando e avaliando a aplicação da ferramenta em um domínio específico. Finalmente, o capítulo 6 reúne as análises e considerações finais sobre os resultados, bem como relata sobre os possíveis trabalhos futuros.

Outra composição importante, do ponto de vista técnico-computacional, é o apêndice A: uma documentação sobre a API desenvolvida durante a fase de implementação.



## Capítulo 2

# Framework i\*, Variações e Ferramentas

Neste capítulo são apresentados os conceitos básicos necessários para o entendimento sobre a técnica de modelagem organizacional do framework i\*, bem como os trabalhos baseados nessa técnica. Por fim, discute-se sobre algumas ferramentas de modelagem i\* e/ou variações. Inicialmente, na seção 2.1, são apresentados os conceitos fundamentais do i\* através dos seus dois componentes de modelagem e alguns dos meta-modelos existentes. Na seção 2.2, mostra-se algumas variações ou extensões da proposta inicial do i\* em (YU, 1995). Em seguida, na seção 2.3, comenta-se sobre a **ferramentas OME**, uma ferramenta que **geram** arquivos de entrada para a JGOOSE. Por fim, na seção 2.4, são feitas algumas considerações finais do capítulo.

### 2.1 O Framework i\*

O framework i\* (pronunciado "i-star"<sup>1</sup>), originalmente proposto por Yu (YU, 1995), é um framework de modelagem organizacional conceitual. Ou seja, ajuda no desenvolvimento de modelos que auxiliam a análise de sistemas sob uma visão estratégica e intencional de processos que envolvem vários participantes.

O framework i\* preocupa-se principalmente com a análise do contexto organizacional e social de um sistema. O sistema, nesse caso, não consiste somente em componentes técnicos, mas também de elementos humanos. Como o framework i\* é bastante flexível para representar situações envolvendo interações entre múltiplos participantes, esse framework pode ser utilizado para representar variados contextos organizacionais.

A seguir, têm-se alguns contextos **onde** a modelagem i\* vem sendo aplicada:

---

<sup>1</sup>O nome i\*, pronunciado em inglês "i-star" faz referência ao conceito sobre uma intencionalidade distribuída. No Brasil, são comuns as pronúncias "i-estrela" e "i-star".

**Engenharia de Requisitos:** é uma das áreas de aplicações mais comuns do i\*, principalmente nas fases iniciais do processo de engenharia de requisitos (*Early Requirements*) (YU, 1997) e (MAIDEN *et al.*, 2004);

**Modelagem de Negócio (Business Modeling):** estudos na área apresentaram o uso do i\* para visualização explícita da intencionalidade por trás dos processos de negócios. Isso ajuda a se obter um melhor entendimento sobre o trabalho, além de facilitar seu replanejamento (YU *et al.*, 1996) (KOLP *et al.*, 2003);

**Desenvolvimento Orientado à Objeto:** em (CASTRO *et al.*, 2000) e (CASTRO *et al.*, 2001) utilizou-se da pUML (precise UML) (EVANS; KENT, 1999) e da *Object Constraint Language* (OCL) (WARMER; KLEPPE, 2003) para tratar dos requisitos finais (*Late Requirements*), além de usar o framework i\* para os requisitos iniciais;

**Desenvolvimento Orientado à Agentes:** em (BRESCIANI *et al.*, 2004) apresentou-se o uso de agentes com estrutura BDI (*Believe, Desire and Intention*) (RAO *et al.*, 1995) para realizar análises na fase inicial de requisitos. Já em (BASTOS; CASTRO, 2004), utilizou-se de Sistemas Multi-Agentes (SMA) para especificar a estrutura organizacional;

**Segurança, Confiabilidade e Privacidade:** a modelagem i\* pode ajudar a lidar com elementos de segurança, confiabilidade e privacidade, através do estudo dos conflitos de intenções de diferentes entidades sociais (YU; LIU, 2001);

Segundo (YU *et al.*, 2011), pode-se dizer que o i\* é um framework de modelagem tanto orientado a agentes quanto orientado a objetivos, pois sua essência é realizada na combinação de agentes/atores e objetivos. Ambos os paradigmas, Orientação à Agentes (MAO; YU, 2005) e Orientação à Objetivos (LAMSWEERDE, 2004), têm apresentado bons resultados em contextos de modelagem organizacional, principalmente em modelagens da fase inicial (*Early Requirements*) do processo de engenharia de requisitos.

O i\* é composto por dois components de modelagem: modelo de dependências estratégicas e modelo de razões estratégicas. Esses componentes auxiliam na representação, respectivamente, das dependências entre atores e dos detalhes por trás das dependências de cada ator. É fundamental conhecer as notações e saber aplicar esses conceitos para se construir um bom mo-

delo organizacional (WEBSTER *et al.*, 2005). A seguir, são apresentados os conceitos e notações por trás dos modelos (AACHEN, 2013).

### 2.1.1 Modelo de Dependências Estratégicas

O modelo de Dependência Estratégica (SD)<sup>2</sup>, representa um conjunto de relacionamento estratégicos externos entre os atores organizacionais, formando uma rede de dependências. Fornece uma visão mais abstrata e ampla da organização, sem se preocupar com os detalhes (razões internas) por trás dessas dependências.

#### Atores, Especializações e Fronteira

- i. **Ator** pode ser definido como uma entidade (humana ou computacional) que age sobre o meio que está inserido para conquistar seus objetivos, exercitando seu *know-how* (YU, 1995). Atores podem ser vistos como uma referência genérica a qualquer unidade que se possa atribuir dependências intencionais. Os atores possuem relações de dependências com outros atores para um determinado fim. Quando existe uma necessidade de maiores detalhes sobre um modelo organizacional, atores podem ser diferenciados em três especializações: agentes, posições e papéis. A Figura 2.2 exemplifica os tipos de atores, enquanto a Figura 2.1 apresenta um exemplo dos possíveis relacionamento entre os tipos de atores. A seguir, descreve-se esses tipos:

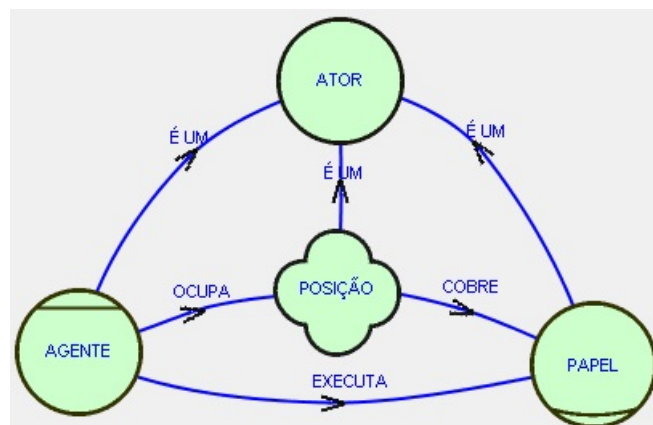


Figura 2.1: Exemplo de relações entre atores

<sup>2</sup>SD, do inglês Strategic Dependency

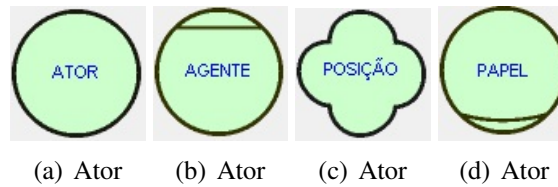


Figura 2.2: Notação de Ator (a), Agente (b), Posição (c) e Papel (d).

- ii. **Agente** é a decomposição de um ator que possui manifestações físicas concretas. Refere-se **tanto a humanos quanto a agentes** de software ou hardware. Um agente possui dependências independentemente do papel que está executando. As características de um agente normalmente não são fáceis de se transferir para outros atores/agentes. São como experiências, habilidades ou, até mesmo, limitações físicas.
- iii. **Papel** é a caracterização abstrata do comportamento de um ator dentro de determinados contextos sociais ou domínio de informação. Essas características devem ser facilmente transferíveis a outro ator social. As dependências associadas a um papel são aplicáveis independentemente do agente que desempenha o papel.
- iv. **Posição** representa uma abstração intermediária entre um agente e um papel. É o conjunto de papéis tipicamente executados por um agente, ou seja, representa uma posição dentro da organização **onde** o agente pode desempenhar várias funções (papéis). Diz-se que um agente ocupa uma posição e uma posição cobre um papel.

**Associações entre Atores:** As associações entre os atores são descritas através de links de associação **(conforme a Figura 2.3)**. Essas associações podem ser de seis tipos:

- i. **IS PART OF** (faz parte de) - Nessa associação cada papel, posição e agente pode ter subpartes. Em *IS PART OF* há dependências intencionais entre o todo e sua parte. Por exemplo, a dependência do todo sobre suas partes para manter a unidade na organização.
- ii. **ISA** (é um) - Essa associação representa uma generalização, com um ator sendo um caso especializado de outro ator. Ambas, *ISA* e *IS PART OF*, podem ser aplicadas entre quaisquer duas instâncias do mesmo tipo de ator.
- iii. **PLAYS** (executa) - A associação plays é usada entre um agente e um papel, com um agente executando um papel. A identidade do agente que executa um papel não deverá ter efeito

algun nas responsabilidades do papel ao qual está associado, e similarmente, aspectos de um agente deverão permanecer inalterados mesmo associados a um papel que este desempenha.

- iv. **COVERS** (cobre) - A associação covers é usada para descrever uma relação entre uma posição e os papéis que esta cobre.
- v. **OCCUPIES** (ocupa) - Esta associação é usada para mostrar que um agente ocupa uma posição, ou seja, o ator executa todos os papéis que são cobertos pela posição que ele ocupa.
- vi. **INS** - Esta associação é usada para representar uma **INST**ância específica de uma entidade mais geral. Por exemplo, quando se deseja representar um agente que é uma instanciação de outro agente.

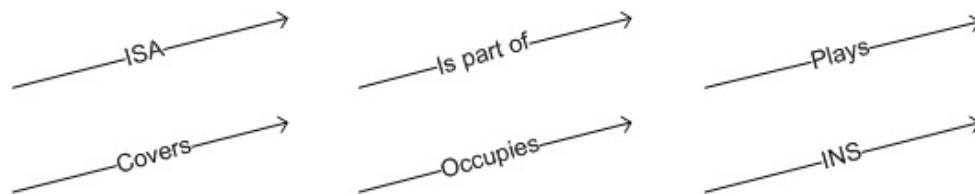


Figura 2.3: Tipos de associações entre atores (AACHEN, 2013).

**Relação de Dependência** Uma relação de dependência pode ser definida como um acordo entre dois atores. Os elementos que compõem uma relação de dependência são:

- i. **Depender**: é o ator dependente, ou seja, o ator que precisa que o acordo (*Dependum*) seja realizado. Esse ator não se importa como o outro ator (*Dependee*) irá satisfazer a necessidade da dependência.
- ii. **Dependum**: é o elemento intermediário, objeto de questionamento e validação, da relação de dependência.
- iii. **Dependee**: é o ator que tem a responsabilidade de satisfazer a relação de dependência.

Dessa forma, pode-se classificar o tipo de uma relação de dependência com base em dos seguintes tipos de *Dependum*:

- i. **Objetivo** (*Goal*) - é uma declaração de afirmação sobre um certo estado do mundo. Deve ser de fácil verificação. O *Dependee* é livre para tomar qualquer decisão para satisfazer o objetivo e é esperado que ele o faça. Não importa para o *Depender* como o *Dependee* irá alcançar esse objetivo.
- ii. **Tarefa** (*Task*) - é uma atividade a ser realizada pelo *Dependee*. Tarefas podem ser vistas com a realização de operações, processos e etc. Porém, não deve ser uma descrição passo-a-passo ou uma especificação completa de execução de uma rotina.
- iii. **Recurso** (*Resource*) - é entidade (física ou informativa) a ser entregue para o *Depender* pelo *Dependee*. Satisfazendo-se esta dependência, o *Depender* está habilitado a usar essa entidade como um recurso.
- iv. **Objetivo-Soft** (*Softgoal*) - é semelhante ao Objetivo, porém os critérios de avaliação e verificação são mais subjetivos. O *Depender* pode decidir sobre o que constitui a realização satisfatória do objetivo.

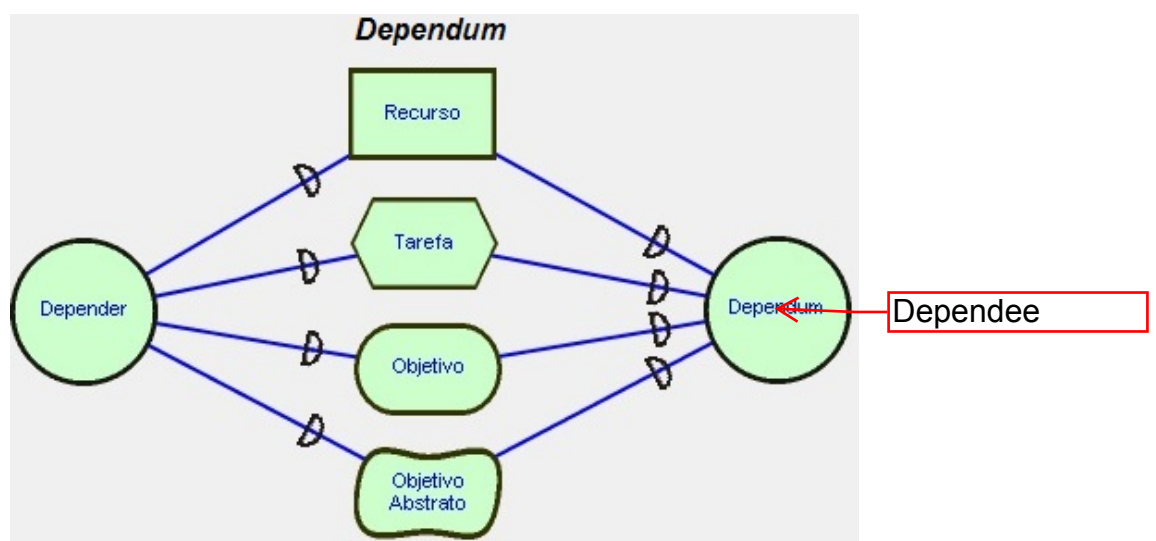


Figura 2.4: Exemplo de **Relação de Dependência** (*Dependder -> Dependum -> Dependee*)

A Figura 2.4 apresenta alguns exemplos de relações de dependências.

**Ligação de dependência** É uma estritamente a conexão entre os elementos de forma direcionada. Assim, pode-se ter somente duas opções de conexão: início no *Depender*, fim no

*Dependum* e início no *Dependum* e fim no *Dependee*. Essa conexão é definida por um segmento contínuo, com a letra "D" sobrescrita, e direcionada da origem para o destino (conforme os exemplos da Figura 2.5).

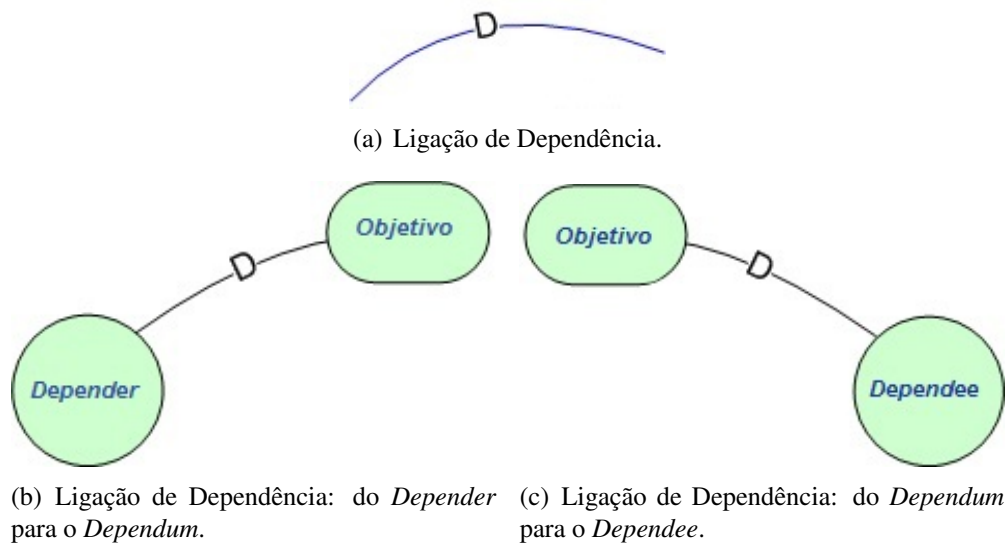


Figura 2.5: Exemplos de ligação de dependência.

### 2.1.2 Modelos de Razões Estratégicas (SR)

Já o modelo de Razões Estratégicas (SR)<sup>3</sup>, representa os detalhes das razões internas que estão por trás das dependências dos atores. Com isso, é possível detalhar os interesses, preocupações e motivações específicas de um ator. Esse tipo de modelo também torna possível a avaliação de alternativas em definições de processos.

O modelo SR, além de poder conter todos os atores e dependências do SD, "abre-se" os Atores para se detalhar as dependências e expressar as razões. Ou seja, para os atores que precisam ser detalhados, é habilitado o limite da fronteira que deve estar visível e com espaço o suficiente para receber os elementos de dependência e/ou ligações internas. Dessa forma, pode-se pensar nos elementos internos ao ator, ou seja, dentro da área de fronteira, como "pertencentes" ao ator. A seguir, são descritos os demais elementos de um modelo SR.

**Fronteira** Uma fronteira indica os limites intencionais de um determinado ator. Todos os elementos dentro dos limites de um ator, são explicitamente desejos ou pretensões desse ator. Uma

<sup>3</sup>SR, do inglês Strategic Rationale

fronteira é representada por um círculo tracejado e o elemento do ator dessa fronteira deve ser sobreposto a ela, ficando acima do tracejado (conforme a Figura 2.6).

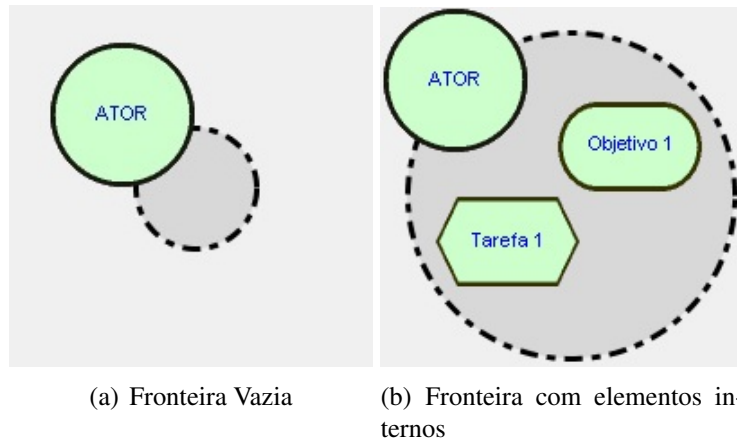


Figura 2.6: Exemplos de fronteira do ator.

**Ligação de meio-fim (*means-end*)** É representada graficamente por uma seta direcionada ao nó fim, significando o meio para atingir um fim (objetivo, recurso, *softgoal*, ou uma tarefa). A Figura 2.7 exemplifica este tipo de ligação.

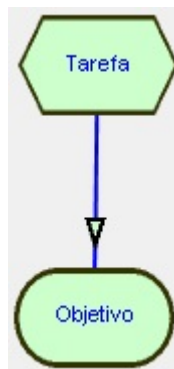


Figura 2.7: Exemplo de ligação meio-fim.

**Ligação de decomposição (*decomposition*):** É responsável por detalhar e expressar da melhor como realizar uma determinada tarefa, através da decomposição em sub-elementos ligados a tarefa principal (superior) através de um segmento de reta cortado. Esses sub-elementos podem ser: metas, tarefas, recursos e objetivos-soft.



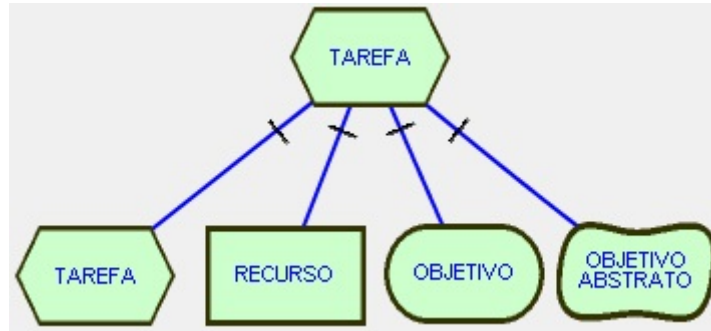


Figura 2.8: Exemplo de ligação de decomposição.

**Ligações de Contribuição (*contribution*):** As ligações de contribuição são para ligar elementos à exclusivamente um objetivo-soft (*softgoal*). Essa ligação ajuda a modelar a forma como os elementos contribuem para a satisfação desse objetivo-soft (*softgoal*). Essas ligações de contribuição, ilustradas na Figura 2.9, podem ser:



Figura 2.9: Ligações de contribuição (AACHEN, 2013).

- i. **Make:** é uma contribuição positiva, suficientemente forte para satisfazer o objetivo-soft.
- ii. **Some +:** é uma contribuição positiva, mas cuja força de influência é desconhecida. Pode equivaler a um *make* ou a um *help*.
- iii. **Help:** é uma contribuição positiva fraca, pois não é suficiente para que ela sozinha satisfaça o objetivo-soft.
- iv. **Unknown:** é uma contribuição cuja influência é desconhecida.
- v. **Hurt:** é uma contribuição negativa fraca, porém não é suficiente para que ela sozinha recuse a satisfação de um objetivo-soft.
- vi. **Some -:** é uma contribuição negativa, mas a força de sua influência é desconhecida. Pode equivaler a um *hurt* ou a um *break*.

- vii. **Break**: é uma contribuição negativa, suficientemente forte para rejeitar a satisfação do objetivo-soft.
- viii. **Or**: é uma contribuição **onde** o objetivo-soft é satisfeito se algum dos descendentes for satisfeitos.
- ix. **And**: é uma contribuição **onde o** objetivo-soft é satisfeito se todos os descendentes forem satisfeitos.

### 2.1.3 iStarML

Muitas ferramentas foram criadas com base nos conceitos do framework i\* ou ~~de~~ variações ~~desse~~ (CARES; FRANCH, 2009) (CARES *et al.*, 2011). Isso acabou gerando modelos específicos para cada ferramenta, dificultando o intercâmbio de modelos entre essas ferramentas. Pensando nisso, foi desenvolvida a iStarML. Um meta-modelo baseado em XML (*Extensible Markup Language*) usado para representar modelos i\* (CARES *et al.*, 2008).

O principal objetivo desse meta-modelo é proporcionar um formato de intercâmbio entre os outros formatos de modelos do i\*. Ou seja, a especificação iStarML deve suportar todas as outras definições e especificações dos modelos já propostos. Com isso, tudo o que se consegue especificar no formato TELOS, por exemplo, deve-se conseguir também no formato iStarML.

Como prova de conceitos, foi realizado um estudo **onde se** aplicou o iStarML estritamente para fazer a interconexão entre duas ferramentas diferentes (COLOMER *et al.*, 2011). Nesse estudo, as ferramentas aplicadas foram jUCMNav (KEALEY *et al.*, 2006) e a HiME (LÓPEZ *et al.*, 2009).

Porém, existe a preocupação sobre a real adoção da especificação iStarML. Como exemplo, tem-se a promessa da ferramenta OpenOME, que diz estar trabalhando para implementar rotinas de importação/exportação em iStarML (HORKOFF *et al.*, 2011) (LAUE; STORCH, 2011).

Conforme já mencionado no Capítulo 1, é objetivo dessa pesquisa a implementação e adoção do iStarML como especificação do formato de arquivo padrão. Portanto, detalhes mais específicos sobre a linguagem iStarML serão feitas no Capítulo 4.

## 2.2 Variações Baseadas no Framework i\*

### 2.2.1 i\* Wiki

O i\* Wiki, é um projeto criado com o intuito de reunir trabalhos relativos ao i\*, de forma colaborativa (AACHEN, 2013) (LEUF; CUNNINGHAM, 2001). Com isso, a comunidade incentiva a colaboração dos usuários do framework, por meio de *feedback* ou mesmo inserção de conteúdo em site oficial. Além disso, esses usuários podem sugerir alternativas ou extensões sintáticas e semânticas em relação a linguagem utilizada.

Apesar da ampla visão que a comunidade pode ter com os trabalhos divulgados no site, a intenção é fornecer e evoluir uma única versão semântica do i\*. Dessa forma, o i\* Wiki funciona sobre duas versões do guia para o i\* (AACHEN, 2013): uma versão estável, servindo de referência para os usuários; outra versão aberta a discussão, acessível aos usuários registrados no site e passível de comentários e sugestões individuais. Além disso, o site reúne um conjunto de Estudos de Casos, Publicações e Eventos relacionados a área de i\*.

### 2.2.2 *Goal-Oriented Requirements Language* (GRL)

A GRL é uma linguagem de apoio à modelagem orientada a agentes e objetivos. Assim como o i\*, a GRL foca na modelagem dos relacionamentos estratégicos entre atores e seus objetivos. Pode-se pensar como uma alternativa que concentra recursos das metodologias NFR (*Non-Functional Requirements*), i\* e Tropos (REGEV; WEGMANN, 2005). Outro ponto interessante é que a GRL é escalável, ~~podendo-se trabalhar~~ com diferentes níveis de granularidade, em múltiplos diagramas ou visões de um mesmo modelo.

Além disso, uma combinação da GRL com a *Use Case Map* (UCM) deu origem a *User Requirement Notation* (URN) - um padrão internacional do *International Telecommunication Union* (ITU) para notação de requisitos de usuário.

### 2.2.3 Tropos

Tropos é um projeto que foi lançado em 2000 (MYLOPOULOS *et al.*, 2001), e visa apoiar a construção de sistemas de software orientados a agentes. O projeto reúne um grupo de autores de diversas Universidades no Brasil, Canadá, Bélgica, Alemanha, Itália, etc. O processo de desenvolvimento segundo esta metodologia inicia com um estudo e elaboração de um modelo

do ambiente no qual o sistema em desenvolvimento irá operar. Este modelo é refinado até que este represente o ambiente com o sistema em seu contexto. Cada modelo é descrito em termos dos atores observados no ambiente em modelagem, seus objetivos e relacionamentos. A metodologia Tropos oferece um framework que engloba as principais fases de desenvolvimento de software, com o apoio das seguintes atividades: Requisitos Iniciais, Requisitos Finais, Projeto Arquitetural e Projeto Detalhado.

## 2.3 Ferramenta OME

Atualmente, existem várias ferramentas de modelagem i\*. Pode-se encontrar mais de 20 ferramentas referenciadas no site do i\* Wiki (AACHEN, 2013). Além disso, alguns trabalhos já realizaram comparações sobre ferramentas do framework i\*, como em (SANTOS, 2008)(Tabela 6) e no site (AACHEN, 2013).

O Ambiente de Modelagem Organizacional, tradução literal de OME - **Organization Modelling Environment**, é um editor gráfico de propósito geral para dar suporte à modelagem orientada a objetivo e/ou orientada a agentes. É uma aplicação Java para *desktop* desenvolvida na Universidade de Toronto (YU; YU, 2013).

A ferramenta possui recursos que auxiliam o usuário no desenvolvimento e manipulação de modelos i\* e NFR (*Non-Functional Requirements*) (CHUNG *et al.*, 2000). Em 2004, o desenvolvimento foi parado e seu código foi portado para a plataforma Eclipse (FOUNDATION, 2013), dando origem a sua versão em código aberto chamada OpenOME (HORKOFF *et al.*, 2011).

Apesar do seu desenvolvimento ter sido finalizado, ainda existem usuários da OME3. Além disso, a ferramenta possui um manual do usuário online (<http://www.cs.toronto.edu/km/ome/docs/manual/manual.html>) e é de fácil utilização. A maioria dos recursos i\*, por exemplo, estão de acordo com (YU, 1995).

Como exemplo, **têm-se** na Figura 2.10 um dos modelos de exemplos já contidos na ferramenta OME3 junto à instalação. Nessa tela, pode-se observar a barra de ferramentas (*toolbar*) com as opções para criação dos elementos i\*: atores, elementos de dependência, links de dependência, links de associação, etc.

Cabe lembrar que essa ferramenta gera os modelos i\* no formato TELOS, atualmente aceitos pela ferramenta JGOOSE. Além disso, considerando que a ferramenta apresenta proble-

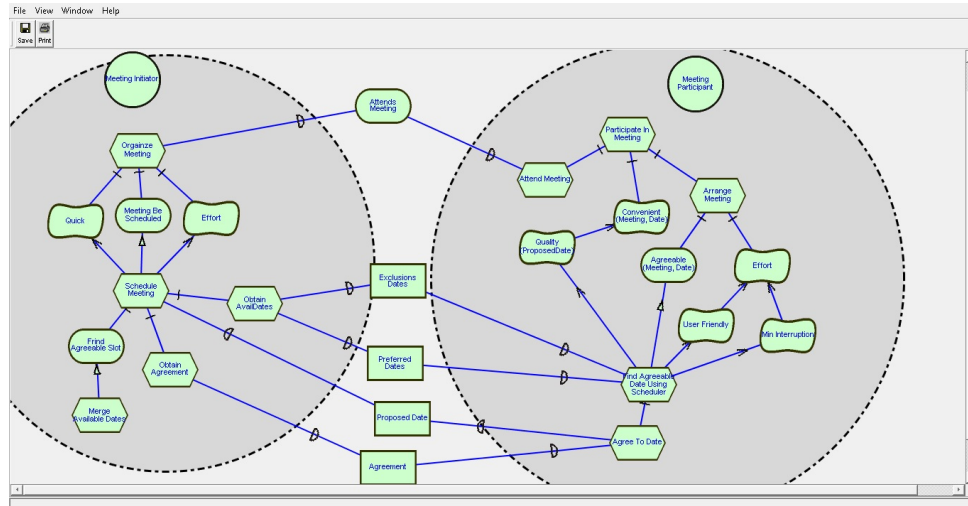


Figura 2.10: Exemplo de modelagem com a ferramenta OME3 (arquivo "Meeting-Schedule.tel")

mas de instabilidade e seu processo de desenvolvimento foi descontinuado, acredita-se que o JGOOSE deva adotar uma nova solução para este fim.

## 2.4 Considerações Finais do Capítulo

Neste Capítulo foram apresentados os conceitos gerais do framework  $i^*$ , bem como as notações da última versão do  $i^*$  Wiki. Além disso, as variações do  $i^*$  e seus respectivos metamodelos, são estudados e analisados. Apesar das variações apresentadas, destaca-se a solução de intercâmbio entre diferentes formatos e ferramentas, iStarML. O iStarML foi adotado pelo editor proposto e será discutido melhor no Capítulo 4.

# Capítulo 3

## JGOOSE

Neste capítulo, é apresentada a ferramenta JGOOSE (*Java Goal into Object Oriented Standard Extension*), bem como uma breve análise das versões ao longo dos anos. Além disso, é analisada a organização arquitetural e o projeto da ferramenta na versão 2013 para posteriormente, no Capítulo 4, realizar a integração com o Editor proposto neste trabalho.

### 3.1 Visão Geral

A ferramenta JGOOSE tem como objetivo geral o mapeamento de modelos organizacionais i\* para casos de uso UML (VICENTE, 2006). A ferramenta trabalha sobre as diretrizes propostas por (SANTANDER, 2002) e é com base nessas diretrizes que a ferramenta interpreta os modelos organizacionais do framework i\* e apresenta os casos de uso mapeados no formato proposto por (COCKBURN, 2001)

### 3.2 Histórico de Versões

A JGOOSE, desde a sua primeira versão ~~feita por~~ (VICENTE, 2006), já passou por várias melhorias e aprimoramentos. A seguir, comenta-se sobre algumas considerações a **respeito** das versões do JGOOSE.

- **Antecedentes** - a ferramenta JGOOSE algumas influências da área de mapeamento de modelos i\* para UML (PEDROZA *et al.*, 2004). ~~Porém, com os avanços das tecnologias, os produtos de softwares também precisam ser atualizados. Pensando nisso, foi proposto uma nova ferramenta, a JGOOSE, mas que mantivesse as funcionalidades principais.~~

- **Versão 2006**, por Vicente (VICENTE, 2006): nova implementação em Java; importação de arquivos em TELOS;
- **Versão 2011**, por Bischke (BRISCHKE, 2012): implementação de três diretrizes faltantes; implementação da exportação em XMI; refinamento manual de caso de uso; visualizador de caso de uso;
- **Versão 2013**, por Peliser (PELISER, 2013): correção de *bugs* na aplicação das diretrizes; otimizações de código (algoritmos); melhorias na interface gráfica; melhorias na documentação.

Com base nesse ~~apanhado~~ histórico, ~~percebe-se~~ que a ferramenta caminha em direção ao profissionalismo e qualidade na área de mapeamento de modelos i\* para caso de uso.

### 3.3 Projeto e Arquitetura

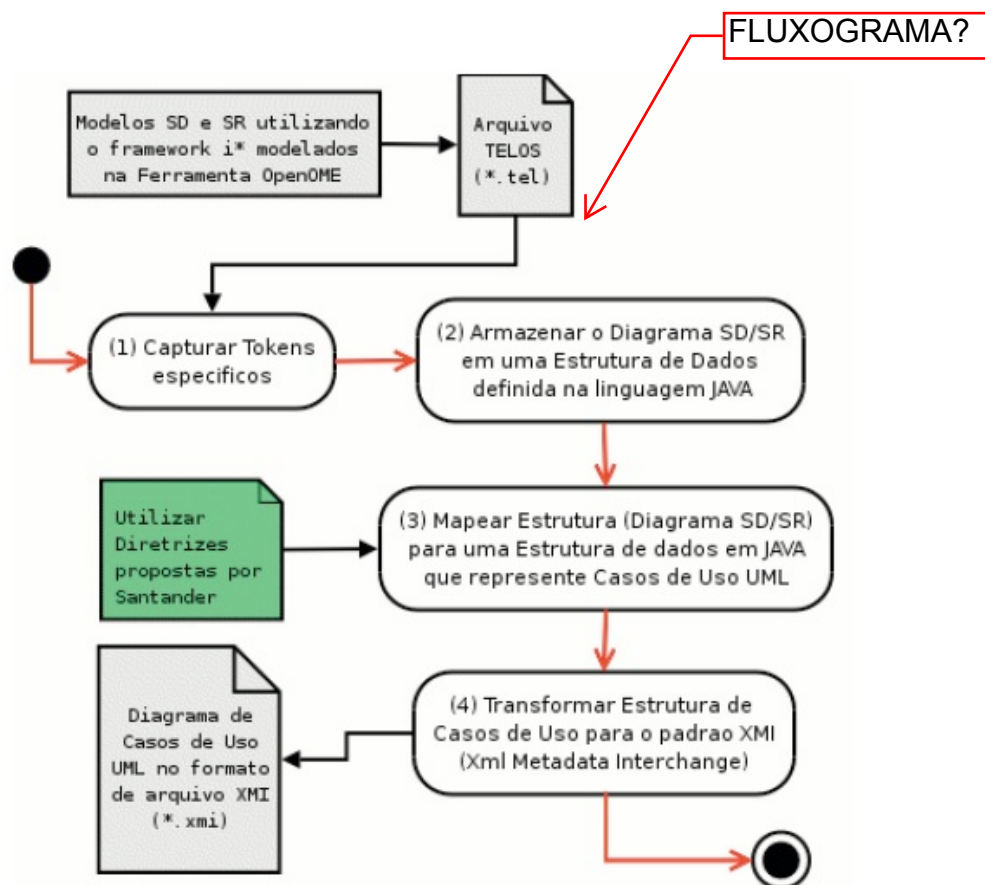


Figura 3.1: Arquitetura e Fluxo de processamento da JGOOSE.

Conforme a seção anterior, a ferramenta passou por várias mudanças. Porém, seu fluxo de processamento manteve-se de acordo com os passos e diretrizes propostos por (SANTANDER, 2002).

### **3.4 Considerações Finais do Capítulo**

O entendimento da ferramenta JGOOSE se faz necessário visto o objetivo de se incorporar um ambiente de edição de modelos organizacionais i\*. Assim, a proposta deste trabalho trata especificamente da etapa (1) "Capturar Tokens Específicos- passando a suportar o formato de arquivo iStarML.



# Capítulo 4

## Proposta

### 4.1 Visão Geral

#### 4.1.1 Conceitos e Considerações Iniciais

### 4.2 Projeto e Arquitetura

#### 4.2.1 Modelagem i\*

#### 4.2.2 Arquitetura Modular

#### 4.2.3 Maven

### 4.3 Desenvolvimento

### 4.4 Recursos

#### 4.4.1 Internacionalização (i18n)

#### 4.4.2 UndoManager

#### 4.4.3 API iStarML

### ~~4.5 Testes e Validação~~

#### 4.5.1 Testes Unitários

### 4.6 Considerações Finais do Capítulo

# Capítulo 5

## **Estudo de Caso**



### **5.1 Resultados e Análises**

### **5.2 Considerações Finais do Capítulo**

## **Capítulo 6**

### **Considerações Finais**

#### **6.1 Contribuições**

#### **6.2 Trabalhos Futuros**

# **Apêndice A**

## **iStarML API**

### **A.1 Introdução**

...

### **A.2 Projeto e Arquitetura**

...

### **A.3 Documentação**

...

# Referências Bibliográficas

AACHEN, R. *i\* Wiki*. 2013. Acessado em: 04 de junho de 2013. Disponível em: <<http://istarwiki.org/>>.

Padronizar Datas

AMYOT, D. Introduction to the user requirements notation: learning by example. *Computer Networks*, Elsevier, v. 42, n. 3, p. 285–301, 2003.

AMYOT, D.; MUSSBACHER, G. *URN: Towards a new standard for the visual description of requirements*. 2003.

BASTOS, L. R.; CASTRO, J. F. Enhancing requirements to derive multi-agent architectures. In: *Proceedings of WER*. [S.l.: s.n.], 2004. p. 127–139.

BRESCIANI, P.; PERINI, A.; GIORGINI, P.; GIUNCHIGLIA, F.; MYLOPOULOS, J. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, Springer, v. 8, n. 3, p. 203–236, 2004.

BRISCHKE, M. Desenvolvimento de uma ferramenta para integrar modelagem organizacional e modelagem funcional na engenharia de requisitos. *Monografia de Graduação*, 2005.

BRISCHKE, M. Melhorando a ferramenta jgoose. *Monografia de Graduação*, Cascavel - PR, Dezembro 2012.

BRISCHKE, M.; SANTANDER, V. F. A.; SILVA, I. F. da. Melhorando a ferramenta jgoose. In: SN. *XV Workshop de Engenharia de Requisitos*. [S.l.], 2012. v. 1.

CARES, C.; FRANCH, X. Towards a framework for improving goal-oriented requirement models quality. 2009.

CARES, C.; FRANCH, X.; PERINI, A.; SUSI, A. *iStarML Reference's Guide*. [S.l.], 2007.

- CARES, C.; FRANCH, X.; PERINI, A.; SUSI, A. istarml: An xml-based model interchange format for i\*. In: *Proc. 3rd Int. i\* Workshop, Recife, Brazil*. [S.l.: s.n.], 2008. v. 322, p. 13–16.
- CARES, C.; FRANCH, X.; PERINI, A.; SUSI, A. Towards interoperability of i\* models using istarml. *Computer Standards & Interfaces*, Elsevier, v. 33, n. 1, p. 69–79, 2011.
- CASE, A. F. Computer-aided software engineering (case): technology for improving software development productivity. *ACM SIGMIS Database*, ACM, v. 17, n. 1, p. 35–43, 1985.
- CASTRO, J.; ALENCAR, F.; CYSNEIROS, G. Closing the gap between organizational requirements and object oriented modeling. *Journal of the Brazilian Computer Society*, SciELO Brasil, v. 7, n. 1, p. 05–16, 2000.
- CASTRO, J. F.; ALENCAR, F. M.; FILHOL, G.; MYLOPOULOS, J. Integrating organizational requirements and object oriented modeling. In: *IEEE. Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on*. [S.l.], 2001. p. 146–153.
- CHUNG, L.; NIXON, B.; YU, E.; MYLOPOULOS, J. Non-functional requirements. *Software Engineering*, 2000.
- COCKBURN, A. *Writing effective use cases*. [S.l.]: Addison-Wesley Reading, 2001.
- COLOMER, D.; LÓPEZ, L.; CARES, C.; FRANCH, X. Model interchange and tool interoperability in the i\* framework: a proof of concept. In: *Proc. of the 14th Workshop on Requirements Engineering*. [S.l.: s.n.], 2011. p. 369–381.
- EVANS, A.; KENT, S. Core meta-modelling semantics of uml: the puml approach. In: *«UML»'99—The Unified Modeling Language*. [S.l.]: Springer, 1999. p. 140–155.
- FOUNDATION, E. *Eclipse*. 2013. Consultado na INTERNET: <http://www.eclipse.org/>, 2013.
- GRAU, G.; CARES, C.; FRANCH, X.; NAVARRETE, F. J. A comparative analysis of i\* agent-oriented modelling techniques. In: *Proceedings of the Eighteenth International Conference on Software Engineering and Knowledge Engineering, SEKE*. [S.l.: s.n.], 2006. p. 5–7.

- HORKOFF, J.; YU, Y.; YU, E. Openome: an open-source goal and agent-oriented model drawing and analysis tool. In: *CEUR proceedings of the 5th international i\* workshop (iStar 2011)*. [S.l.: s.n.], 2011. p. 154–156.
- KEALEY, J.; KIM, Y.; AMYOT, D.; MUSSBACHER, G. Integrating an eclipse-based scenario modeling environment with a requirements management system. In: IEEE. *Electrical and Computer Engineering, 2006. CCECE'06. Canadian Conference on*. [S.l.], 2006. p. 2432–2435.
- KOLP, M.; GIORGINI, P.; MYLOPOULOS, J. Organizational patterns for early requirements analysis. *Lecture Notes in Computer Science*, Springer, v. 2681, p. 617–632, 2003.
- KOTONYA, G.; SOMMERVILLE, I. *Requirements engineering: processes and techniques*. [S.l.]: J. Wiley, 1998. (Worldwide series in computer science). ISBN 9780471972082.
- KOUBARAKIS, M.; MYLOPOULOS, J.; STANLEY, M.; BORGIDA, A. *Telos: Features and formalization*. [S.l.]: Computer Science Institute, Foundation of Research and Technology, Hellas, 1989.
- LAMSWEERDE, A. V. Requirements engineering in the year 00: A research perspective. In: ACM. *Proceedings of the 22nd international conference on Software engineering*. [S.l.], 2000. p. 5–19.
- LAMSWEERDE, A. van. Goal-oriented requirements engineering: a roundtrip from research to practice [engineering read engineering]. In: IEEE. *Requirements Engineering Conference, 2004. Proceedings. 12th IEEE International*. [S.l.], 2004. p. 4–7.
- LAUE, R.; STORCH, A. Adding functionality to openome for everyone. In: *CEUR Proceedings of the 5th International i\* Workshop (iStar 2011)*. [S.l.: s.n.], 2011. v. 766, p. 169–171.
- LEUF, B.; CUNNINGHAM, W. *The wiki way: quick collaboration on the web*. Addison-Wesley Professional, 2001.
- LÓPEZ, L.; FRANCH, X.; MARCO, J. Hime: hierarchical i\* modeling editor. *Revista de Informática Teórica e Aplicada*, v. 16, n. 2, p. 57–60, 2009.

MAIDEN, N. A.; JONES, S. V.; MANNING, S.; GREENWOOD, J.; RENOU, L. Model-driven requirements engineering: synchronising models in an air traffic management case study. In: SPRINGER. *Advanced Information Systems Engineering*. [S.l.], 2004. p. 368–383.

MAO, X.; YU, E. Organizational and social concepts in agent oriented software engineering. In: *Agent-Oriented Software Engineering V*. [S.l.]: Springer, 2005. p. 1–15.

MASON, G. L. A conceptual basis for organizational modelling. *Systems Research and Behavioral Science*, Wiley Online Library, v. 14, n. 5, p. 331–345, 1997.

MYLOPOULOS, J.; BORGIDA, A.; JARKE, M.; KOUBARAKIS, M. Telos: Representing knowledge about information systems. *ACM Transactions on Information Systems (TOIS)*, ACM, v. 8, n. 4, p. 325–362, 1990.

MYLOPOULOS, J.; KOLP, M.; CASTRO, J. Uml for agent-oriented software development: The tropos proposal. In: *The Unified Modeling Language. Modeling Languages, Concepts, and Tools*. [S.l.]: Springer, 2001. p. 422–441.

PEDROZA, F. P.; ALENCAR, F. M.; CASTRO, J. F.; SILVA, F. R.; SANTANDER, V. F. Ferramentas para suporte do mapeamento da modelagem i\* para a uml: extended good xgood e goose. In: *Proceedings of the VII Workshop on Requirements Engineering-WER*. [S.l.: s.n.], 2004. v. 4, p. 164–175.

Onde ?



PELISER, D. Aprimorando a ferramenta jgoose. Projeto de Iniciação Científica - PIBIC. Abril 2013.

RAO, A. S.; GEORGEFF, M. P. *et al.* Bdi agents: From theory to practice. In: SAN FRANCISCO. *Proceedings of the first international conference on multi-agent systems (ICMAS-95)*. [S.l.], 1995. p. 312–319.

REGEV, G.; WEGMANN, A. Where do goals come from: the underlying principles of goal-oriented requirements engineering. In: IEEE. *Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference on*. [S.l.], 2005. p. 353–362.

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



SANTOS, B. S. *IStar Tool-Uma proposta de ferramenta para modelagem de i\**. Tese (Dissertação de Mestrado) — Centro de Informática, Recife, PE, outubro 2008.

SCHNEIDER, F.; BERENBACH, B. A literature survey on international standards for systems requirements engineering. *Procedia Computer Science*, Elsevier, v. 16, p. 796–805, 2013.

VICENTE, A. A. Jgoose: Uma ferramenta de engenharia de requisitos para integração da modelagem organizacional i\* com a modelagem funcional de casos de uso uml. *Monografia de Graduação*, Cascavel - PR, Dezembro 2006.

VICENTE, A. A.; SANTANDER, V. F.; CASTRO, J. B.; SILVA, I. F. da; MATUS, F. G. R. Jgoose: a requirements engineering tool to integrate i\* organizational modeling with use cases in uml jgoose: Una herramienta de ingeniería de requisitos para la integración del modelado organizacional i\* con el modelado de casos de uso en uml. *Ingeniare. Revista chilena de ingeniería*, SciELO Chile, v. 17, n. 1, p. 6–20, 2009.

WARMER, J.; KLEPPE, A. *The object constraint language: getting your models ready for MDA*. [S.l.]: Addison-Wesley Longman Publishing Co., Inc., 2003.

WEBSTER, I.; AMARAL, J.; CYSNEIROS, L. M.; GERAIS-BRASIL, B. H.-M. A survey of good practices and misuses for modelling with i\* framework. In: *Proc. of the VIII Workshop on Requirements Engineering, WER*. [S.l.: s.n.], 2005. v. 5, p. 148–160.

YU, E. Modeling organizations for information systems requirements engineering. In: *IEEE. Requirements Engineering, 1993., Proceedings of IEEE International Symposium on*. [S.l.], 1993. p. 34–41.

YU, E. Towards modelling and reasoning support for early-phase requirements engineering. In: *IEEE. Requirements Engineering, 1997., Proceedings of the Third IEEE International Symposium on*. [S.l.], 1997. p. 226–235.

YU, E. Agent orientation as a modelling paradigm. *Wirtschaftsinformatik*, Springer, v. 43, n. 2, p. 123–132, 2001.

YU, E.; GIORGINI, P.; MAIDEN, N. *Social modeling for requirements engineering*. [S.l.]: Mit Press, 2011.

YU, E.; LIU, L. Modelling trust for system design using the i\* strategic actors framework. In: *Trust in Cyber-societies*. [S.l.]: Springer, 2001. p. 175–194.

YU, E.; YU, Y. *Organization Modelling Environment*. 2013. Consultado na INTERNET: <http://www.cs.toronto.edu/km/ome/>, 2013.

YU, E. S.; MYLOPOULOS, J.; LESPÉRANCE, Y. AI models for business process reengineering. *IEEE expert*, IEEE, v. 11, n. 4, p. 16–23, 1996.

YU, E. S.-K. *MODELLING STRATEGIC RELATIONSHIPS FOR PROCESS REENGINEERING*. Tese (Doutorado) — University of Toronto, 1995.