



Pós-Graduação em Ciência da Computação

**“Uma Proposta de Métricas para Avaliar Modelos
i*”**

Por

Emanuel Batista dos Santos

Dissertação de Mestrado



Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE, JUNHO/2008



Universidade Federal de Pernambuco

CENTRO DE INFORMÁTICA

PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Emanuel Batista dos Santos

“Uma Proposta de Métricas para Avaliar Modelos i^* ”

Este trabalho foi apresentado à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

ORIENTADOR: Jaelson Freire Brelaz de Castro

RECIFE, JUNHO/2008

Santos, Emanuel Batista dos
Uma proposta de métricas para avaliar modelos
i* / Emanuel Batista dos Santos. – Recife: O Autor,
2008.

xi, 117 folhas : il., fig., tab.

Dissertação (mestrado) – Universidade Federal
de Pernambuco. Cln. Ciência da Computação, 2008.

Inclui bibliografia, apêndices e anexo.

1. Engenharia de software. I. Título.

005.1 CDD (22.ed.) MEI2008-081

Agradecimentos

Agradeço a Deus pela oportunidade. Aos meus pais Neves e Deodato pelo apoio dado sem o qual esse trabalho não seria possível. Ao professor Jaelson Castro, meu orientador, por seus sábios conselhos que contribuíram decisivamente para o resultado desse trabalho. Agradeço também aos meus colegas do Grupo LER pelo apoio, em especial a Ricardo Ramos e Márcia Lucena pela participação em atividades desse projeto. E finalmente, ao Conselho Nacional de Pesquisa (CNPq) pelo apoio financeiro que permitiu a realização desse trabalho.

Resumo

A engenharia de requisitos orientada a metas (em inglês, *Goal-Oriented Requirement Engineering*) tem se mostrado uma forma promissora de descrever sistemas de software. Ela provê uma forma natural de estruturar documentos de requisitos complexos através de metas (em inglês, *Goal*) que fornecem um mecanismo para justificar a existência dos requisitos e facilitar a administração de conflitos entre requisitos. Neste contexto surgiram diversas abordagens que utilizam metas como abstração entre elas KAOS, NFR, V-Graph e i^* . A demanda por software de qualidade exige que todos os artefatos produzidos ao longo do processo de construção do *software* também sejam de qualidade. Artefatos como documentos de requisitos, código e executáveis do sistema devem estar livre de erros e de falhas, pois segundo pesquisas quanto mais cedo problemas são identificados mais barato é a correção desses. Assim identificar problemas ainda na fase de requisitos reduz a necessidade e os custos de correção no futuro. De forma similar às técnicas tradicionais, as técnicas orientadas a metas também necessitam de mecanismos para garantir a qualidade de seus artefatos. Nesta dissertação são apresentadas métricas para avaliar a qualidade de modelos i^* , que são utilizados nas fases iniciais de requisitos. As métricas procuram relacionar qualidades desejadas de documentos de requisitos com construções básicas da técnica i^* , de forma a fornecer mecanismos eficazes para identificar problemas nos modelos i^* . As métricas estão agrupadas para tratar de: erros típicos, nível de detalhamento, ambigüidade e complexidade. Para facilitar a interpretação dos valores obtidos pelo uso das métricas é apresentada como elas poderão ser usadas com o método GQM (*Goal-Question-Metric*).

Palavras-chave Avaliação de Qualidade, Engenharia de Requisitos, Métricas para modelos i^*

Abstract

The Goal-Oriented Requirement Engineering has shown a promising way to describe software systems. It provides a natural mechanism to structure complex requirement documents through goals providing a mechanism to justify the existence of the requirements and facilitate the management of conflicting requirements. In this context, different approaches that use Goal abstraction have emerged, such as KAOS, NFR, V-Graph and i*. The demand for quality software requires that all artifacts produced during the software development should also be of high quality. Artifacts and documents of requirements, executable code and the system must be free of errors and failures, because problems identified in early stage are cheaper to correct. Thus, identify problems in requirements stage reduces the need and costs of correction in future stages. Similar to traditional techniques, Goal-Oriented approaches also demand mechanisms to insure the quality of their artifacts. In this dissertation we present metrics to assess the quality of i* models used for early requirement stage. The metrics matches qualities that we desired of requirement documents to basic i* constructions. In doing so we provide some effective mechanisms to identify problems in i* models. The metrics are grouped to address the following issues: typical errors, level of detail, ambiguity and complexity. In order to facilitate interpretation of values obtained by the use of metrics we presented how they can be used in connection with the GQM Method (Goal-Question-Metric).

Keywords Quality Evaluation, Requirements Engineering, Metrics for i* Models

Sumário

Resumo	ii
Abstract	iii
Sumário	iv
Lista de Figuras	viii
Lista de Tabelas	ix
1 Introdução	1
1.1 Contextualização	2
1.2 Motivação	3
1.3 Trabalhos Relacionados	3
1.4 Objetivos	5
1.5 Escopo e Limitações	6
1.6 Estrutura do Documento	6
2 Engenharia de Requisitos e a Técnica de Modelagem i*	8
2.1 Engenharia de Requisitos - Uma visão Geral	9
2.1.1 Processo de Engenharia de Requisitos	10
2.2 Engenharia de Requisitos Orientada a Metas	11
2.3 Técnica de Modelagem i*	13
2.3.1 Modelo SD	14
2.3.2 Modelo SR	17

2.3.3	Conceitos Avançados de Análise	19
2.4	Conclusão	21
3	Medição de Software	22
3.1	Introdução	23
3.2	Métricas de Software	24
3.2.1	Métricas para Engenharia de Requisitos	26
3.2.1.1	Métricas de Tamanho	26
3.2.1.2	Métricas para Rastreabilidade	26
3.2.1.3	Métricas de Volatilidade de Requisitos	27
3.2.1.4	Métricas de Completude de Requisitos	27
3.3	Métricas para Complexidade	28
3.4	O Método GQM	29
3.5	Conclusão	30
4	Métricas para i*	31
4.1	Introdução	32
4.2	Métrica para Erros Típicos	35
4.3	Métricas para Nível de Detalhamento	39
4.3.1	Taxa de Metas sem ligação com Rotinas	40
4.3.2	Taxa de Dependências sem Ligação a Elementos Internos no Ator	43
4.3.3	Taxa de <i>Softgoals</i> que não são tratados por Contribuição	46
4.4	Métricas para Ambigüidade	49
4.4.1	Número de Palavras Vagas	49
4.4.2	Taxa de Dependências Ambíguas	51
4.5	Métricas para Complexidade	56
4.5.1	Complexidade Ciclomática	57

4.5.2	Métricas de Halstead	63
4.5.2.1	Vocabulário	63
4.5.2.2	Comprimento	67
4.5.2.3	Volume	68
4.5.2.4	Dificuldade	69
4.5.2.5	Esforço	70
4.5.3	Entropia	71
4.6	Resumo das Métricas	74
4.7	Conclusão	75
5	Exemplo de Aplicação	76
5.1	Introdução	77
5.2	Descrição do Exemplo - Informa Turma	77
5.3	Procedimento de Avaliação	79
5.4	Coleta de Dados	82
5.5	Análise dos Resultados	88
5.6	Validação das Métricas	89
5.7	Conclusão	90
6	Conclusão e Trabalhos Futuros	91
6.1	Revisão do Problema e Objetivo	92
6.2	Principais Contribuições	92
6.3	Comparação com Trabalhos Relacionados	93
6.4	Trabalhos Futuros	95
	Referências	96
	Apêndice A – Catálogo dos Erros mais Frequentes com i*	101

Apêndice B – Coleta Automática de Métricas	109
Anexo A – Templates para documentação de Métricas e Dados (IEEE-1061, 1992)	115

Lista de Figuras

1	Atores e relacionamentos entre atores	15
2	Tipos de relacionamento de dependência entre atores no i^*	16
3	Operações sobre elementos nos Modelos SR	18
4	Exemplo de Rotina	20
5	Medição segundo Humphrey	24
6	Método GQM (Goal Question Metric)	30
7	Exemplo do Media Shop	34
8	Exemplo de Metas se ligação com Rotinas	40
9	Exemplo de Dependências sem ligação a Elementos Internos no Ator	43
10	Exemplo D_3	46
11	Separação em Dependência	51
12	Junção em Dependência	51
13	Exemplo do crescimento explosivo de um modelo i^*	56
14	Modelo SR do exemplo Informa Turma	78
15	Modelo SR do Time de Qualidade	111
16	Modelo SR do Coletor Automático de Métricas	113

Lista de Tabelas

1	Entry 01	36
2	Métrica NErros	37
3	Dado Erros Típicos	38
4	Métrica TMR	41
5	Dado NMR	42
6	Métrica TDLEIA	44
7	Dado NDLEIA	45
8	Dado NDA	45
9	Métrica TSC	47
10	Dado NSC	48
11	Dado NSA	48
12	Métrica NPV	50
13	Dado NPV	50
14	Métrica TDA	52
15	Dado NDRD1	53
16	Dado NDRD2	54
17	Dado ND	55
18	Métrica CC	59
19	Dado NL	60
20	Dado NE	61
21	Dado NCC	62
22	Métrica Vocabulário	64

23	Dado NLD	65
24	Dado NED	66
25	Métrica Comprimento	67
26	Métrica Volume	68
27	Métrica Dificuldade	69
28	Métrica Esforço	70
29	Métrica Entropia para Modelos i^*	72
30	Dado Probabilidade Referencial	73
31	Resumo das Métricas	74
32	Intervalo de Aceitação	81
33	Dados para a Métrica Entropia do Exemplo Informa Turma	85
33	Dados para a Métrica Entropia do Exemplo Informa Turma (Continuação)	86
34	Dados para o Exemplo Informa Turma	87
35	Métricas para o Exemplo Informa Turma	87
36	Comparação com Trabalhos Relacionados	93
37	Entry 01	101
38	Entry 02	102
39	Entry 03	102
40	Entry 04	103
41	Entry 05	103
42	Entry 06	104
43	Entry 07	105
44	Entry 08	105
45	Entry 09	106
46	Entry 10	107
47	Entry 11	107

48	Entry 12	108
49	Entry 13	108
50	Estrutura para documentação de Métricas IEEE-1061	115
51	Estrutura para documentação de Dados IEEE-1061	116

1 Introdução

Este capítulo introdutório apresenta as principais motivações para realizar este trabalho. São apresentados os trabalhos relacionados a ele. Também são definidos os objetivos, o escopo e as limitações do mesmo. Além da estrutura como o documento está organizado.

1.1 Contextualização

A Engenharia de Requisitos busca entender e representar condições impostas para a construção de software. Neste contexto, os documentos de requisitos representam a formalização das necessidades dos *stakeholders*¹ e das restrições as quais o sistema deve obedecer (KOTONYA; SOMMERVILLE, 1998). Requisitos, no entanto, podem apresentar alguns problemas tais como:

- Requisitos não refletem as reais necessidades dos consumidores do sistema;
- Requisitos são inconsistentes e/ou incompletos;
- Alterações nos requisitos depois de concordados são caras de fazer;
- Mal-entendidos existem entre consumidores, aqueles que desenvolvem os requisitos do sistema e os engenheiros de software que desenvolvem ou mantêm o sistema.

A engenharia de requisitos orientada a metas ² (em inglês, *Goal-Oriented Requirement Engineering*) tem crescido como uma forma promissora de descrever sistemas de software. Ela provê uma forma natural para estruturar documentos de requisitos complexos através de metas que fornecem um mecanismo para justificar a existência dos requisitos e facilitar a administração de conflitos entre requisitos (LAMSWEERDE, 2001). No contexto da engenharia de requisitos orientada a metas surgiram diversas abordagens que utilizam essa abstração entre elas KAOS (DARDENNE; FICKAS; LAMSWEERDE, 1991), NFR (CHUNG et al., 2000), V-Graph (YU; LEITE; MYLOPOULOS, 2004) e i* (YU, 1995). Em particular, o i* tem sido usado pela engenharia de requisitos para descrever sistemas de software baseado nas intenções de seus *stakeholders* (YU, 1997; GRAU et al., 2008).

Essa dissertação está focada na técnica de modelagem i*, essa técnica é usada em abordagens de diversas áreas não somente na engenharia de requisitos como na modelagem de processos e a reengenharia de software. O desenvolvimento de abordagens baseadas em i* tem gerado a demanda pela avaliação dos modelos. Tanto com o propósito de identificar possíveis problemas como para verificar a eficácia das abordagens. Em especial o grupo de estudo LER (Laboratório de Engenharia de Requisitos) (LER, 2008) usa o i* em diversas atividades relacionadas à engenharia de requisitos o que gera uma constante demanda pela avaliação de modelos i*.

¹Todos aqueles que têm algum interesse no sistema

²O termo **meta** vem da palavra inglesa *Goal* e também pode ser traduzido como **objetivo**

1.2 Motivação

O i^* é uma técnica de modelagem que usa uma notação gráfica para representar requisitos em termos de atores e suas metas. Alguns problemas que podem afetar a qualidade de modelos i^* podem ser apontados:

- Inexistência de padrão, muitas variantes são apresentadas, mas ainda não há uma versão padronizada (LUCENA et al., 2008);
- Crescimento explosivo do tamanho dos diagramas (ALENCAR et al., 2006);
- Erros causados pelo mau uso da notação do i^* (WEBSTER; AMARAL; CYSNEIROS, 2005).

Além dos problemas específicos da técnica, os modelos i^* também podem ser afetados por problemas comuns na engenharia de requisitos, uma vez que é usado para modelar requisitos. Um dos problemas com requisitos é que nem sempre são compreensíveis ou coerentes. Muitas vezes a variedade de *stakeholders* e as diferentes perspectivas que eles têm sobre os requisitos pode prejudicar a criação dos documentos de requisitos (DAVIS et al., 1993). Problemas como esses podem gerar um grande custo se sua correção for feita em fases finais do processo de desenvolvimento (DAVIS, 1993). A identificação de problemas na fase de requisitos reduz os custos para correção de defeitos (BOEHM, 1984; KOTONYA; SOMMERVILLE, 1998). O custo da correção de erros na fase de projeto é cerca de três a seis vezes mais alto do que na fase de definição de requisitos (PRESSMAN, 2002).

Existem diversas técnicas para encontrar potenciais problemas em documentos de requisitos tais como auditoria, revisão, inspeção (IEEE-1028, 1998), e técnicas de leitura (BASILI et al., 1996). Infelizmente essas técnicas não provêm regras objetivas de como avaliar potenciais problemas em documentos de requisitos. Métricas podem ser usadas para avaliar documentos de requisitos de forma menos subjetiva possibilitando a automação. O uso de métricas reduz a subjetividade na avaliação e controle da qualidade do software provendo uma base quantitativa para tomada de decisões (IEEE-1061, 1992).

1.3 Trabalhos Relacionados

Em Reis (2004) é descrita uma metodologia para avaliar aplicações web na fase de requisitos, nela é utilizada a engenharia de domínio para analisar atributos de aplicações

web e atribuir notas à eles obtendo assim uma nota global para a aplicação avaliada. Apesar de fazer uso de métricas para analisar documentos de requisitos essa abordagem não trata o i^* .

Em Cabral et al. (2008) é apresentada uma comparação experimental de técnicas de inspeção baseadas em leitura (*checklists* vs. *Perspective-Based Reading*). Ele utilizou como objeto do experimento documentos de requisitos que foram avaliados por leitores em diferentes perspectivas para comparar a eficiência das técnicas em encontrar erros. O trabalho não faz a definição de uma metodologia própria por se basear em trabalhos já existentes, no entanto o i^* é abordado de forma genérica podendo ser considerado como um documento de requisitos avaliado em busca de erros (CABRAL et al., 2008).

Em Ramos et al. (2006) é definida uma metodologia para avaliar e melhorar documentos de requisitos, a AIRDoc (*Approach to Improve Requirements Documents*). Esta metodologia é baseada no GQM (*Goal-Question-Metric*) (BASILI; CALDIERA; ROMBACH, 1994). Ela usa métricas para identificar possíveis problemas em documentos de requisitos e de acordo com o problema identificado propõe melhorias nos documentos. As melhorias podem ser feitas através da aplicação de padrões ou refatoramentos. Esta metodologia é genérica e se propõe a ser aplicada a diferentes tipos de documentos de requisitos. O i^* no entanto é abordado de forma genérica pela metodologia não levando em consideração os detalhes da técnica de modelagem, em (RAMOS et al., 2007) há um exemplo da aplicação da metodologia na avaliação em modelos i^* .

Em Franch, Grau e Quer (2004), Franch, Grau e Quer (2005), Franch (2006) e Franch e Grau (2008) é apresentada uma estrutura (em inglês, *framework*) para definição de métricas para avaliação de modelos i^* . A estrutura é usada para definir métricas para modelos SD (*Strategic Dependency*), permitindo criar métricas tanto globais como locais para Atores e Dependências. Estas métricas são utilizadas para prever o comportamento de um modelo segundo algum atributo de qualidade (e.g., previsibilidade, segurança, adaptatividade, etc.). Para interpretar as métricas são usados pesos previamente definidos. Como resultado uma nota é obtida para o modelo segundo o atributo de qualidade que está sendo avaliado. A principal aplicação desta estrutura é a definição de métrica para seleção de arquiteturas para software de prateleira, ou COTS (*Commercial Off-The-Shelf*). Esta abordagem está limitada à modelos SD e não leva em consideração possíveis problemas com o modelo antes que seja avaliado, ou seja, os modelos já devem ter sido escritos corretamente antes de serem avaliados para um atributo de qualidade. Apesar de ser feita para definir métricas para modelos i^* a abordagem apresenta apenas uma

métrica instanciada (FRANCH, 2006), caso o usuário da abordagem necessite de métricas para avaliar seus modelos ele deve instanciar as próprias métricas.

Os trabalhos apresentados são em parte limitados quando se referem ao i^* , em alguns casos são definidas metodologias como em Reis (2004) e Ramos et al. (2006) entretanto elas não tratam da uso de métricas para modelos i^* de forma específica. No caso de Cabral (2007) e Ramos et al. (2007) o i^* é abordado de forma genérica para na procura por problemas, mas novamente as abordagens não levam em consideração as construções básicas do i^* . Em Franch (2006) e Franch e Grau (2008) o i^* é tratado de forma específica, mas o tratamento está restrito aos modelos SD, além da abordagem apresentada não ser aplicada a procura por possíveis problemas.

1.4 Objetivos

Requisitos são modelados para simplificar a análise ou documentação, mas devido à características da linguagem de modelagem ou do próprio modelador, os modelos podem não estar bem escritos. Este problema afeta também modelos i^* e podem prejudicar a qualidade dos modelos. Como apresentado anteriormente, algumas abordagens que usadas na avaliação de documentos de requisitos apresentam limitações quando se trata da avaliação de modelos i^* . Em particular, elas não se preocupam com características específicas do i^* , ou quando se preocupam limitam essa avaliação aos modelos SD como é o caso de Franch (2006).

Esta dissertação tem como objetivo principal propor um conjunto de métricas para avaliar quantitativamente modelos i^* na fase de requisitos. As métricas propostas devem considerar tanto modelos SD como modelos SR (*Strategic Rationale*). A avaliação se dará em termos de:

- presença de erros, o mau uso do i^* pode gerar erros que prejudicam a qualidade dos modelos, será avaliada a presença de erros nos modelos;
- nível de detalhamento, quando finalizados os modelos podem apresentar diferentes níveis de detalhe, para avaliar isso serão usadas métricas que vão relacionar construções básicas do i^* com o detalhamento atingido pelo modelo;
- presença de ambigüidade, algumas construções podem apresentar problemas como ambigüidades onde diversas interpretações podem ser geradas devido a problemas de entendimento;

- complexidade, os modelos i^* tem um problema com o crescimento explosivo da complexidade, as métricas devem permitir avaliar quanto complexo está um modelo.

Um conjunto de objetivos específicos está associado ao objetivo principal, entre eles :

- a revisão de conceitos da engenharia de requisitos;
- o estudo detalhado da técnica de modelagem i^* ;
- a revisão dos conceitos de métricas e da avaliação de qualidade para requisitos.

1.5 Escopo e Limitações

As métricas definidas estão direcionadas a artefatos de software que neste caso são específicos da técnica i^* . Por este motivo são exclusivas para avaliação de modelos i^* , não sendo aplicadas à outras abordagens. Elas permitem avaliar os modelos i^* e avaliar possíveis problemas antes que se propagem para fases posteriores.

O i^* não possui um padrão para o processo de criação dos modelos, assim métodos diferentes podem ser usados na modelagem para se chegar a um modelo i^* . A ausência de um processo padronizado dificulta fazer inferências sobre os passos seguidos para obter o modelo, essa dissertação não trata de métricas relacionadas aos processos de criação dos mesmos. Assim as métricas propostas tratam dos modelos como produtos terminados não havendo inferências sobre a forma como eles foram obtidos.

Uma limitação existente na coleta das métricas é que o responsável pela coleta deve ter conhecimento prévio da técnica i^* . Isso se deve ao fato de algumas métricas serem baseadas em construções específicas da técnica e o responsável pela coleta deve estar preparado para reconhecer a ocorrência de problemas com essas construções.

1.6 Estrutura do Documento

A dissertação está organizada na seguinte estrutura de capítulos:

O Capítulo 2 – Engenharia de Requisitos e a Técnica de Modelagem i^* , apresenta conceitos de engenharia de requisitos e de engenharia de requisitos orientada a metas. Neste capítulo também é apresentada a técnica i^* e suas principais construções.

O Capítulo 3 – Medição de Software, apresenta conceitos de medição de software, métricas gerais e específicas para engenharia de requisitos.

O Capítulo 4 – Métricas para i^* , apresenta um conjunto de métricas definidas para avaliar quantitativamente documentos de requisitos modelados com i^* .

O Capítulo 5 – Exemplo de Aplicação, apresenta a aplicação das métricas para i^* a um exemplo.

O Capítulo 6 – Conclusão e Trabalhos Futuros, apresenta as conclusões e trabalhos futuros da dissertação.

O Apêndice A – Catálogo de Erros Típicos, apresenta um catálogo com os principais erros na modelagem com i^* encontrados na revisão da literatura.

O Apêndice B – Coleta Automática de Métricas, apresenta uma proposta de coletor automático de métricas em i^* com sugestões do que seria necessário para criar um coletor.

O Anexo A – *Template* de Métricas do IEEE 1061, apresenta os templates para métricas e dados do padrão IEEE-1061 (1992) usados na dissertação para documentar as métricas apresentadas na dissertação.

*2 Engenharia de Requisitos e a Técnica de Modelagem i^**

Esse capítulo apresenta conceitos básicos de engenharia de requisitos, Engenharia de Requisitos Orientada a Metas (em inglês, GORE - Goal Oriented Requirements Engineering) e a técnica de modelagem i^* . Na seção 2.1 são abordadas definições de Engenharia de Requisitos e Requisitos. Na seção 2.2 são apresentados os conceitos referentes à Engenharia de Requisitos Orientada a Metas e algumas abordagens que são baseadas em metas. Ao final do capítulo, na seção 2.3, é apresentado o i^* e suas principais construções.

2.1 Engenharia de Requisitos - Uma visão Geral

A Engenharia de Software tem como principal objetivo resolver problemas do mundo real através de software. Mas para isso é necessário entender o problema e determinar quais necessidades e restrições devem ser satisfeitas. A Engenharia de Requisitos tenta alcançar estes propósitos através da identificação de *stakeholders*¹ e suas possíveis necessidades. Na literatura é possível encontrar diversas definições para Engenharia de Requisitos. A seguir é feita uma revisão das principais definições.

Segundo Lamsweerde (2000), a Engenharia de Requisitos é a fase do desenvolvimento de sistemas de software responsável pela identificação dos objetivos do sistema pretendido; pela operacionalização de tais objetivos em serviços e restrições; e pela atribuição da responsabilidade dos requisitos resultantes para agentes como: humanos, hardware, e software. Ao longo dessa dissertação será adotada a definição de engenharia de requisitos apresentada por Lamsweerde (2000).

Uma outra definição de Engenharia de Requisitos é da por Zave (1997), na qual Engenharia de Requisitos é o ramo da engenharia de software preocupada com metas do mundo real para funções e restrições em sistemas de software. Ela também está preocupada com o relacionamento destes fatores para uma especificação precisa do comportamento do software e sua evolução ao longo do tempo e através de famílias de software.

Segundo Davis (1993), a Engenharia de Requisitos corresponde à atividade de entendimento das necessidades do usuário no contexto do problema a ser resolvido, bem como das limitações impostas na solução.

Da mesma forma, várias são as definições de requisito. Um requisito pode ser visto como algo que a ser realizado, transformado, produzido ou provido. Dessa forma a tarefa da engenharia de requisitos é descobrir o que deve ser realizado, transformado, produzido ou provido para atender às necessidades dos clientes, além de entender o contexto e domínio do problema.

Mais formalmente um requisito pode ser tratado como (IEEE-610, 1994):

1. Uma condição ou capacidade necessária para um usuário resolver um problema;
2. Uma condição ou capacidade que deve ser encontrada por um sistema ou um componente de sistema para satisfazer um contrato, padrão ou especificação, ou outro documento formalmente imposto;

¹Todos aqueles que interagem de alguma forma com o sistema sendo desenvolvido

3. uma representação documentada de uma condição ou capacidade, conforme os itens (1) e (2).

Davis (1993) sugere que um requisito é uma necessidade do usuário ou uma característica, função ou atributo necessário do sistema que pode ser percebido de uma posição externa daquele sistema.

Segundo Kotonya e Sommerville (1998), um requisito pode descrever:

- uma facilidade no nível do usuário;
- uma propriedade muito geral do sistema;
- uma restrição específica no sistema;
- uma restrição no desenvolvimento do sistema.

Para compreender e modelar requisitos é necessário estabelecer os tipos dos requisitos. Os requisitos podem ser classificados como requisitos funcionais, não-funcionais, ou organizacionais conforme apresentado por Kotonya e Sommerville (1998).

Requisitos funcionais são as declarações das funções que o sistema deve oferecer, como ele se comporta com entradas particulares e como o sistema deve se comportar em situações específicas. O termo “função” é usado no sentido genérico da operação que pode ser realizada pelo sistema, seja por meio de comandos dos usuários, seja pela ocorrência de eventos internos ou externos ao sistema. Em alguns casos, os requisitos funcionais podem também, explicitamente, definir o que o sistema não deve fazer.

Requisitos não-funcionais (RNFs) são as restrições nas funções oferecidas pelo sistema. Incluem restrições de tempo, restrições no processo de desenvolvimento, padrões e qualidades globais de um software: como custo, performance, confiabilidade, manutenibilidade, portabilidade, usabilidade, desempenho, dentre outras.

Requisitos organizacionais dizem respeito às metas da empresa, suas políticas estratégicas adotadas, os relacionamentos entre os seus atores junto com seus respectivos objetivos.

2.1.1 Processo de Engenharia de Requisitos

Um processo de engenharia de requisitos é um conjunto estruturado de atividades que são seguidas para entregar, validar e manter os documentos de requisitos de um sistema.

Atividades de um processo incluem elicitação de requisitos, análise e negociação de requisitos, documentação de requisitos, validação de requisitos e gerenciamento de requisitos (KOTONYA; SOMMERVILLE, 1998). A descrição completa de um processo deve incluir quais atividades são realizadas, a estruturação e agendamento das atividades, quem serão os responsáveis por cada atividade, as entradas e saídas das atividades e ferramentas a serem usadas. Essas atividades não são necessariamente sequenciais, Tipicamente o processo começa com a elicitação de requisitos, passando à análise e negociação. Posteriormente, os requisitos são documentados e validados. O gerenciamento de requisitos ocorre ao longo de todas as outras atividades, ele tem como objetivo controlar a evolução e mudanças nos requisitos.

Entre os artefatos gerados na engenharia de requisitos estão os documentos de requisitos que representam a formalização dos requisitos, segundo Kotonya e Sommerville (1998) um documento de requisitos é uma declaração oficial dos requisitos do sistema para consumidores, usuários finais e desenvolvedores do sistema. Os documentos de requisitos podem conter desde descrições em linguagem natural até modelos do sistema. Os modelos são uma representação abstrata de objetos reais onde detalhes são simplificados para facilitar o entendimento (BEZIVIN, 2005), na engenharia de software eles são usados para facilitar a análise e a documentação de requisitos. Essa dissertação está especialmente interessada em modelos de requisitos como será apresentado nas seções seguintes.

A proposta apresentada nessa dissertação não considera a criação de métricas para o processo de engenharia de requisitos. Essa decisão se deve ao fato de não haver um processo padronizado para a criação de modelos i^* , pois dessa forma seria difícil garantir que métricas seriam abrangentes o suficiente para avaliar qualquer os processos para criar modelos i^* . Por esse motivo as métricas propostas tratam dos modelos como artefatos concluídos, não fazendo inferência sobre a forma como foram obtidos.

2.2 Engenharia de Requisitos Orientada a Metas

Engenharia de Requisitos Orientada a Metas (em inglês, Goal-Oriented Requirements Engineering) está preocupada com o uso de metas para elicitar, elaborar, estruturar, especificar, analisar, negociar, documentar e modificar requisitos (LAMSWEERDE, 2001). Seu uso baseia-se em modelos multi-visão que mostram como as metas, objetos, agentes, cenários, operações, e propriedades de domínio são inter-relacionados em sistemas. São analisados os estados atuais, onde é feita a análise do contexto no qual o sistema está

inserido ou no qual será implantado, e o estado futuro, onde é feita a análise dos impactos da implantação do sistema.

Metas podem ser organizadas em estruturas que capturam como elas são e como podem ser refinadas. Elas podem cobrir desde conceitos em alto nível, como metas estratégicas, até prescrições técnicas que podem ser designadas como responsabilidades de agentes. Essas últimas podem ser requisitos do sistema ou expectativas em seu ambiente. Metas podem se referir a interesses funcionais ou à atributos de qualidade. Uma meta funcional tipicamente captura um conjunto de cenários desejados, ela pode ser estabelecida em um sentido claro. Já as metas de qualidade, geralmente representam requisitos não-funcionais e capturam alguns comportamentos que não são capturados por metas funcionais, são usados para comparar alternativas e impor restrições à operacionalização.

A medida que o uso de Metas cresceu na Engenharia de Requisitos, várias técnicas que usam *Goals* como abstração surgiram, entre elas *KAOS* (DARDENNE; FICKAS; LAMSWEERDE, 1991), *NFR Framework* (CHUNG et al., 2000), *V-Graph* (YU; LEITE; MYLOPOULOS, 2004) e *i** (YU, 1995).

O *KAOS* (*Knowledge Acquisition in autOmated Specification*) é uma estrutura conceitual que define abstrações, como entidades, relacionamentos e agente, como extensões de objeto. Uma entidade é um objeto autônomo independente de outros objetos. Um relacionamento é um objeto subordinado. Um agente é um objeto que tem escolha e comportamento. *KAOS* define crenças, metas e ações. As metas podem ser refinadas em uma árvore através de decomposição em metas e estados. *KAOS* utiliza lógica temporal e refinamento de técnicas de inteligência artificial para que metas e estados sejam consistente e rigorosamente definidos. A principal ênfase do *KAOS* é na prova formal de que os requisitos encontram as metas que foram definidas para visualizar o sistema.

O *NFR Framework* (*Non-Functional Requirements Framework*) está baseado em metas usadas para modelar e analisar requisitos não-funcionais. Ele introduz a noção de *softgoals*² que são metas onde os critérios de aceitação não são claramente definidos, elas são passíveis de negociação e interpretação. Um *softgoal* pode interagir com outros *softgoals* e não precisam ser completamente satisfeitos, podendo haver uma satisfação parcial. O *NFR Framework* pode modelar requisitos não-funcionais como segurança, performance, acurácia entre outros. Esses requisitos são modelados através do *SIG* (*Softgoal Interdependency Graph*) que é um grafo onde as *softgoals* são decompostas em *softgoals* mais refinados e é feito o relacionamento entre eles.

²não existe tradução aceitável para português, por esse motivo o termo será mantido do original

V-graph é um tipo de modelo de metas, proposto em Yu, Leite e Mylopoulos (2004) como uma simplificação do modelo *NFR Framework* (CHUNG et al., 2000) para demonstrar uma abordagem de identificação de candidatos a aspectos em requisitos. O nome *V-graph* originou-se da disposição entre os seus três elementos constituintes, metas, *softgoals* e tarefas, no qual para cada tarefa são associados metas e *softgoals* relacionados.

No *i** vários tipos de atores são definidos para modelar situações onde um ator depende de outro para que suas metas sejam alcançadas, tarefas sejam executadas ou recursos sejam fornecidos. O *i** não modela somente as metas do sistema, mas também os da organização que existem em torno do sistema (YU, 1997).

O *i** não apresenta um arcabouço formal como o KAOS, por isso é mais simples de entender e usar. O *i** também possui mecanismos como dependências, atores e estruturação de atores que não são cobertos pelo KAOS, esses mecanismos permitem fazer análise mais detalhadas dos impactos do software na organização. Apesar de modelar *softgoals* o NFR não trata de requisitos funcionais de forma explícita e não existe a noção de responsabilidade que é dada pela abstração do ator. O *i** também modela *softgoals* integrando eles aos processos das organizações modeladas além de estabelecer os responsáveis por atendê-los. O *i** cobre as abstrações de tarefas, metas e *softgoals* que também estão presentes no *V-graph*, além disso o *i** apresenta abstrações como atores e dependências que permitem analisar os impactos do software na organização. O *i** é o foco deste trabalho ele é detalhado na seção 2.3.

2.3 Técnica de Modelagem *i**

O *i** (YU, 1995) é uma técnica de modelagem organizacional desenvolvida para modelagem e análise. É usado para representar relacionamentos entre atores estratégicos em uma rede social. Isso é feito sob uma visão estratégica e intencional de processos que envolvem vários participantes, ou seja, uma modelagem organizacional. O *i** possui diversas áreas de aplicação além da especificação de requisitos, tais como reengenharia de processos de negócio; desenvolvimento orientado a agentes; análise de impactos organizacionais; e modelagem de processos de software (YU, 1995).

O *i** apresenta diversas variantes como apresentado em (LUCENA et al., 2008), cada variante pode apresentar diferenças sintáticas e semânticas. Para evitar problemas com o entendimento a versão do *i** utilizada nesse trabalho é a apresentada em (GRAU et al., 2008). Essa versão é uma evolução da versão original apresentada em (YU, 1995) e

conserva os principais elementos do i^* original.

Para realizar a análise organizacional, os engenheiros de requisitos modelam os *stakeholders* como atores e suas intenções como metas. Os atores são entidades que possuem metas e podem depender de outros atores para conseguir alcançá-las (YU, 1995). O conjunto de dependências criadas pelos atores formam uma rede social que representa o ambiente do sistema e o sistema em si. A estrutura conceitual do i^* é utilizada para obter uma compreensão mais apurada sobre os relacionamentos da organização, de acordo com os diversos atores do sistema. Além disso, o i^* permite compreender as razões envolvidas nos processos de decisões.

O i^* é formado por dois modelos: o modelo SD, *Strategic Dependency model*,³ e o modelo SR, *Strategic Rationale model*⁴. O modelo SD fornece uma descrição intencional de um processo em termos de uma rede de relacionamentos de dependência entre atores relevantes, ou seja, as relações de dependências externas entre os atores da organização. O modelo SR apresenta uma descrição estratégica do processo, em termos de elementos do processo e das razões que motivam a existência deles.

2.3.1 Modelo SD

No modelo SD, são capturadas as motivações e os desejos dos atores que fazem parte da organização além de apresentar a rede de relacionamentos do ator. Dado um modelo SD, pode-se perguntar que novos relacionamentos entre os atores são possíveis; identificar a viabilidade ou não das dependências; ou ainda relacionar os desejos de um agente com as habilidades do agente do qual depende, para poder explorar as oportunidades disponíveis a esses atores. Modelos SD descrevem as dependências entre os atores, mas não as razões que as motivam.

O termo “Ator” é utilizado para referenciar genericamente a qualquer unidade para a qual dependências intencionais possam ser atribuídas. Atores podem ser considerados: intencionais, por possuírem motivações, desejos e razões por trás de suas ações; e estratégicos, quando não focam apenas o seu objetivo imediato, mas quando se preocupam com as implicações de seu relacionamento estrutural com outros atores.

Os atores podem ser diferenciados em três tipos especializados de atores: agente, papel e posição. Representação gráfica na figura 1.

³em português, Modelo de Dependência Estratégica

⁴em português, Modelo de Razão Estratégica

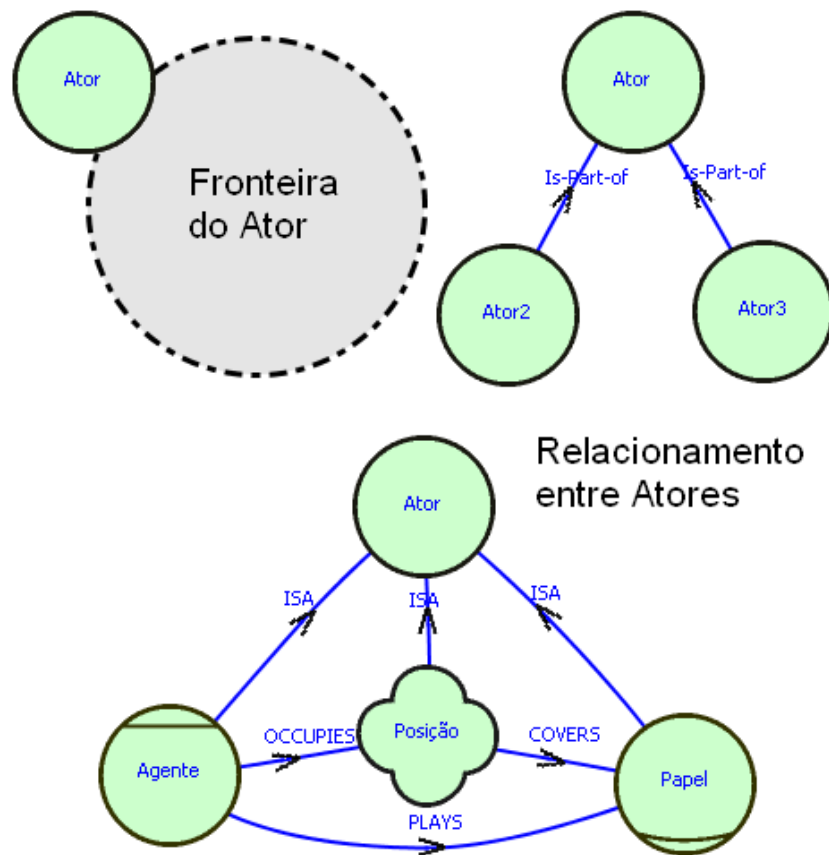


Figura 1: Atores e relacionamentos entre atores

Agente é um ator que possui manifestações físicas concretas. Tanto pode referir-se a humanos quanto a agentes artificiais de software ou hardware;

Papel representa a caracterização abstrata do comportamento social de um ator dentro de determinados contextos sociais ou domínio de informação; apresenta as funções que podem ser exercidas por um agente dentro da organização;

Posição representa uma entidade intermediária entre um agente e um papel. É o conjunto de papéis tipicamente ocupados por um agente, ou seja, representa uma posição dentro da organização onde o agente pode desempenhar várias funções.

Os atores possuem relacionamentos que permitem organizar os mapeamentos desejáveis entre eles. Os relacionamentos são de especialização (*Is-a*), agregação (*Is-part-of*), cobertura (*Covers*), atuação (*Plays*) e ocupação (*Occupies*). Quando essa distinção é feita, a análise torna-se mais detalhada e exige uma atenção maior. Os mecanismos de estruturação de atores são muito complexos e ainda não estão completamente bem definidos, por esse motivo não serão abordados especificamente na dissertação. Maiores detalhes sobre esses mecanismos podem ser encontrados em Leite et al. (2007) e Clotet

et al. (2007).

Cada dependência em i^* é um relacionamento intencional, ou seja, um acordo chamado *dependum*, entre dois atores: o *dependor* e o *dependee*. O *dependor* é o ator que depende de um outro ator (*dependee*) para que o acordo (*dependum*) possa ser realizado. O *dependor* tem metas que não sabe como alcançá-las, não tem recursos para, ou simplesmente não quer realizar, e repassa essa necessidade para outro (o ator *dependee*) que tem a habilidade ou os recursos necessários para isso.

Os relacionamentos de dependência usados em i^* descrevem a natureza do acordo e podem ser de quatro tipos: tarefas, recursos, metas ou *softgoals* ⁵.

Um ator *dependee* satisfaz uma dependência quando disponibilizar o recurso necessário, executar a tarefa que foi requisitada, atender a uma meta, ou satisfazer um *softgoal* do ator *dependor*. A Figura 2 apresenta os tipos de ligações de dependência do i^* .

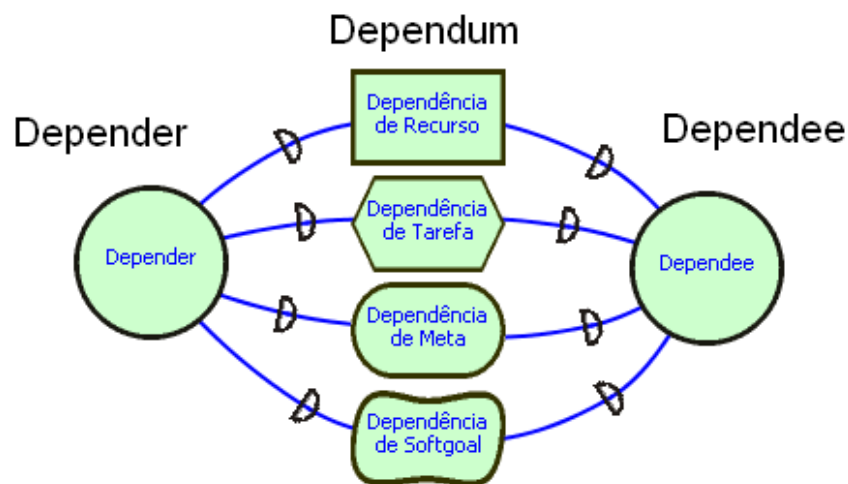


Figura 2: Tipos de relacionamento de dependência entre atores no i^*

Na dependência de tarefa, o *dependee* é requisitado a executar uma dada atividade, sendo informado sobre o que deve ser feito. Contudo, a descrição de uma tarefa em i^* não tem por intenção ser uma completa especificação dos passos necessários à execução dessa tarefa, nem há preocupação em se informar o “porquê” da solicitação de sua realização. Uma tarefa especifica uma forma particular de se fazer algo, elas podem ser vistas como soluções que provêem operações, processos, representações de dados, estruturação, restrições e meios para atender às necessidades estabelecidas nas metas e *softgoals*.

Na dependência de recurso, o ator depende da disponibilidade de uma entidade física

⁵não existe tradução aceitável para português, por esse motivo o termo será mantido do original

ou de uma informação. Por recurso entendemos o produto final de alguma ação, em um processo, que estará ou não disponível para o ator dependente. Nesse tipo de dependência assume-se que não haja aspectos pendentes a serem tratados ou decisões a serem tomadas.

Na dependência de meta, um ator depende de outro para que uma determinada alcançar alguma condição ou estado do mundo. O *dependee* é livre para tomar as decisões necessárias para satisfazer uma meta, não importando a maneira como haverá satisfação.

Um *softgoal* especifica uma condição ou estado do mundo que um ator gostaria de alcançar, mas que a satisfação não pode ser definida a priori como verdadeira ou falsa, de forma que é o objeto de interpretação ou negociação por parte dos *stakeholders*. Uma dependência de *softgoal* representa uma qualidade que está associada a uma meta, assim ela pode estar relacionada aos requisitos não-funcionais que se deseja que o sistema atenda.

O analista pode escolher de acordo com a necessidade um destes tipos de dependência para modelar o contexto do projeto. Cada caso tem um propósito e interpretação. As dependências de tarefa restringem o curso de ação que os atores podem seguir, por outro lado às dependências de metas e *softgoals* dão liberdade aos atores sobre como agir para satisfazer as dependências.

2.3.2 Modelo SR

Enquanto o modelo SD trata apenas dos relacionamentos externos entre os atores, o modelo SR é utilizado para descrever os interesses, preocupações e motivações dos atores participantes de um processo. Ele possibilita a avaliação das possíveis alternativas de definição do processo, investigando mais detalhadamente as razões existentes por trás das dependências entre os atores.

As fronteiras dos atores indicam fronteiras intencionais de um ator. Todos os elementos dentro da fronteira de um ator são explicitamente desejados pelo ator. Para alcançar estas metas, frequentemente um ator pode depender de intenções de outros atores, representado por uma ligação de dependência atravessando a fronteiras de atores (GRAU et al., 2008). A representação gráfica da fronteira está na figura 1 no lado superior esquerdo.

O modelo SR também é composto por nós e ligações que, juntos fornecem uma estrutura para expressar as razões envolvidas no processo. Ele utiliza quatro tipos de nós, que se baseiam nos tipos de dependências do modelo SD: recurso, tarefa, meta e *softgoal*. Nesse modelo, são apresentados outros tipos de relacionamentos que interconectam os nós: as ligações de *Means-Ends* (em português seria meios-fins), as ligações de decomposição

de tarefas, e as contribuições para *softgoals*.

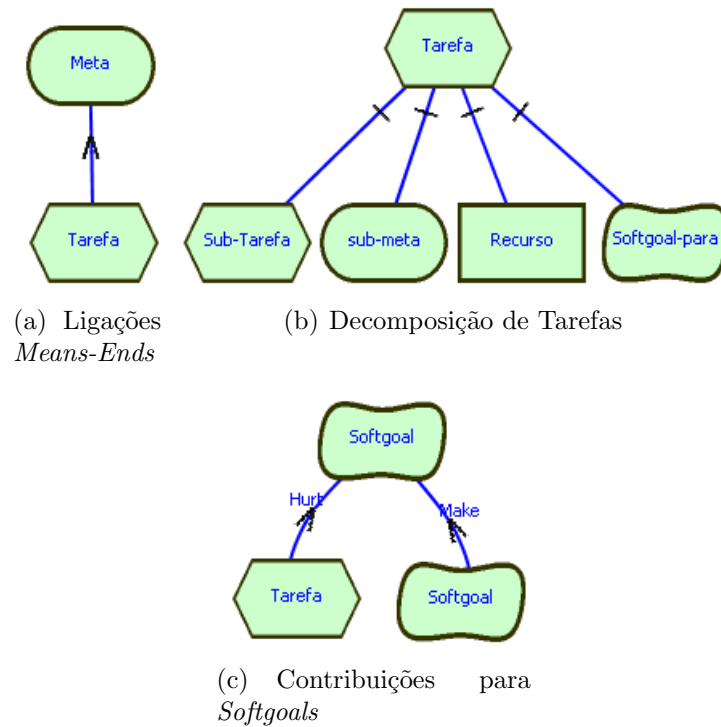


Figura 3: Operações sobre elementos nos Modelos SR

Uma ligação de *Means-Ends* indica um relacionamento entre um nó fim (*End*) - que pode constituir uma meta a ser atingido e um meio (*Mean*) para isso (Figura 3(a)). Esse tipo de ligação sugere alternativas existentes para se alcançar um determinado fim. Os meios são expressos em forma de tarefas, e os fins em forma de metas.

Já as ligações de decomposição de tarefas ligam um nó de tarefa a seus nós componentes, que podem ser outras tarefas, recursos, metas ou *softgoals* (Figura 3(b)). Estes elementos podem ser parte de ligações de dependências em modelos de Dependência Estratégica. O significado da decomposição de tarefa é resumido a seguir (GRAU et al., 2008):

- Decomposição Tarefa-Meta: Sub-meta. Neste tipo de decomposição não é especificado como uma é alcançada, permitindo considerar alternativas;
- Decomposição Tarefa-Tarefa: Sub-tarefa. Quando uma tarefa é especificada como um subcomponente de uma tarefa maior, ela restringe a maior em um curso particular de ação;
- Decomposição Tarefa-Recurso: recurso-para. A entidade representada por um recurso não é considerada problemática por um ator. A principal preocupação é se

estará disponível;

- **Decomposição Tarefa-Softgoal:** *Softgoal* para. Quando uma *Softgoal* é componente em uma decomposição de tarefa, ela serve como um meta de qualidade para a tarefa que guia a seleção de alternativas em uma futura decomposição.

Os *Softgoals* podem ter uma satisfação parcial e serem influenciadas por outros elementos. Para representar a satisfação parcial é utilizada a contribuição para *softgoals* (Figura 3(c)), através dela as tarefas e *softgoals* podem influenciar positiva ou negativamente para a satisfação de um *softgoal*. As ligações de contribuição são representadas por ligações com rótulos que indicam o quanto um elemento ajuda ou prejudica a satisfação de uma *softgoal*. Os rótulos podem ser: *Make*, *Help*, *Some+*, *Some-*, *Hurt* e *Break* (GRAU et al., 2008). Esse rótulos representam uma contribuição fortemente positiva (*Make*), parcialmente positiva (*Some+* e *Help*), parcialmente negativa (*Hurt* e *Some-*) e fortemente negativa (*Break*).

Esses mecanismos de detalhamento (*Means-Ends*, decomposição, e contribuição) são considerados na proposta de métricas para modelos i*. Como pode ser visto no capítulo 4, eles serão avaliados em diversas situações pelas métricas propostas.

2.3.3 Conceitos Avançados de Análise

Além das construções básicas os modelos SD e SR apresentam propriedades que são consideradas durante a fase de análise. Algumas métricas propostas são inspiradas nessas propriedades por isso serão apresentadas a seguir.

As propriedades descritas em (YU, 1995) são: *ability*⁶, *workability*, *committment* e *viability*.

A propriedade *ability* está relacionada à capacidade do ator em realizar algo, por exemplo, se o ator possui uma rotina para realizar uma meta então ele tem a habilidade de realizar aquela meta. A propriedade *workability* indica que o ator acredita que uma determinada rotina irá executar mesmo que não esteja completamente especificada ou conhecida.

Como exemplo é possível ver na figura 4 a meta **Pedidos pela Internet Manipulados** sendo operacionalizada pela tarefa **Usar Carro de Compras**, a rotina formada vai

⁶os termos referentes as propriedades foram mantidos do original por não haver consenso na tradução para português

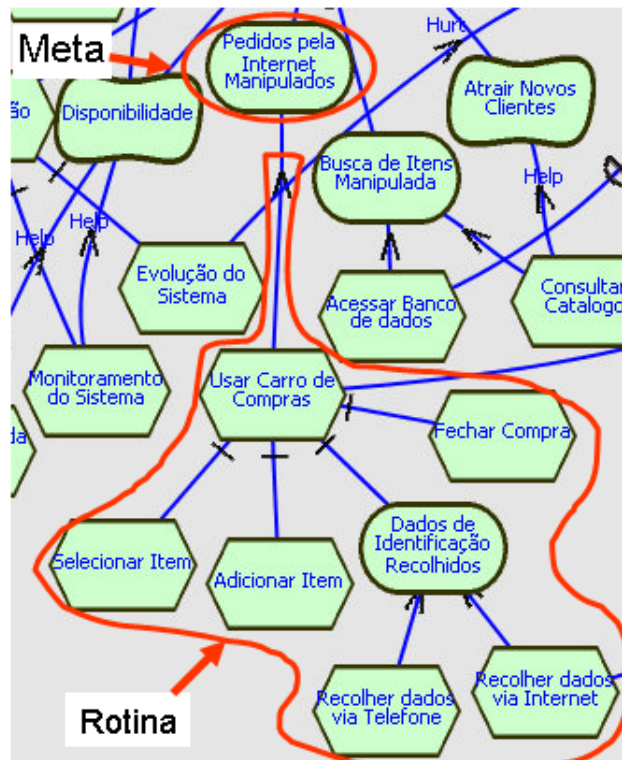


Figura 4: Exemplo de Rotina

até os últimos elementos na árvore de decomposição da tarefa que fez a ligação *Means-Ends*.

A propriedade *commitment* está relacionada à delegação e dependência externa do ator. Quando um ator precisa de algo que não é capaz de fazer sozinho ele pode depender de outros atores para que isso seja feito. O ator *dependee* se compromete, em algum grau, a fazer com que a dependência seja satisfeita.

A propriedade *viability* fornece informação sobre o grau em que uma rotina vai executar de acordo com algum *softgoal* da rotina. Quando os *softgoals* de uma rotina são satisfeitos significa que a rotina é viável para aqueles *softgoals*.

Um modelo que atende a essas propriedades é considerado mais detalhado que modelos que não as atende. Para definir métricas para avaliar nível de detalhamento, partimos da hipótese que as análises em relação aos elementos do i* foram realizadas, assim um modelo mais detalhado apresentaria as seguintes características:

- As metas que estão dentro da fronteira de um ator, estão ligados a rotinas indicando como serão realizados;
- As dependências estão ligadas à elementos internos nos atores, para indicar o “porquê”

de sua existência e “como” serão satisfeitas;

- Os *softgoals* que estão dentro da fronteira dos atores devem receber contribuições que indicam se serão satisfeitos ou não.

As propriedades apresentadas serviram de inspiração para a criação de métricas, mas não serão tratadas detalhadamente nesse trabalho. Para verificar se um modelo atende às propriedades *ability*, *workability* e outras é necessária uma avaliação cuidadosa com algoritmos destinados a esse fim, em Horkoff (2006) é apresentado um algoritmo para avaliação de modelos i^* que pode ser usado para analisar essas propriedades.

2.4 Conclusão

Nesse capítulo foram apresentadas definições de Requisitos e Engenharia de Requisitos que são usadas para compreender a dissertação. Foram identificadas as principais abordagens da engenharia de requisitos orientada a metas e delimitadas as razões pela qual o i^* é escolhido como objeto de pesquisa no contexto dessa dissertação. Além disso, foram apresentadas as principais construções do i^* , que são usadas para definir as métricas.

3 Medição de Software

Este capítulo apresenta uma visão geral dos conceitos de medição de software motivando o uso de métricas para avaliação de qualidade. São apresentadas métricas para requisitos e os conceitos de métricas para complexidade que serão usados ao longo da dissertação. Além disso é feita uma breve introdução do método GQM (BASILI; CALDIERA; ROMBACH, 1994).

3.1 Introdução

A medição de software é uma prática que não é independente das demais e interfere com todos os sub-processos do desenvolvimento de software (RIFKIN; COX, 1991). Devido à má interpretação de conceitos de métricas e a dificuldades em coletar, armazenar, reportar e analisar os resultados, a implantação de um programa de métricas pode se algo custoso e controverso.

A utilização de métricas reduz a subjetividade (IEEE-1061, 1992) na avaliação da qualidade de software através do fortalecimento de informações quantitativas a respeito do produto e do processo, além de tornar a qualidade do software mais visível. A engenharia de software começa a evoluir para um caminho em que medições fazem parte dos processos básicos para se construir software.

Informações quantitativas são utilizadas para orientar a tomada de decisões, e para validar ou refutar novas propostas de processos ou novas tecnologias, entre outros aspectos mensuráveis nas organizações de software (IEEE-1061, 1992). Trabalhando assim, as organizações passam a não serem guiadas apenas por opiniões, idéias ou suposições, e sim por dados objetivos e fatos devidamente embasados.

Medições de software fornecem informações objetivas para dar suporte aos gerentes de projetos principalmente nos seguintes aspectos (MCGARRY et al., 2002):

- Comunicação efetiva através das informações objetivas;
- Acompanhamento dos objetivos dos projetos através da medição dos processos e produtos;
- Identificação e correção dos problemas de forma antecipada, uma vez que medições viabilizam uma estratégia de gerência pró-ativa;
- Suporte na tomada de decisões críticas;
- Justificativas para a tomada de decisões.

Os custos para a definição e implantação de um programa de medição são elevados, e precisam ser respaldados por ganhos reais e visíveis para então serem apoiados pela alta gerência. Segundo Humphrey (1989) as principais motivações para medição de software são descritas a seguir, essa visão também é compartilhada por outros autores como pode ser visto em Park, Goethert e Florac (1996).

Na Figura 5 estão apresentadas as principais motivações para a medição de software, segundo Humphrey:

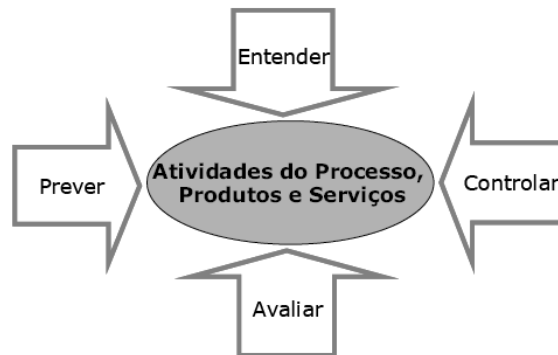


Figura 5: Principais Razões para medir qualidade segundo Humphrey(HUMPHREY, 1989)

Entender – métricas de software podem ser utilizadas para se adquirir aprendizado sobre algum aspecto do produto ou atividade do processo, e estabelecer base para comparação com avaliações futuras;

Avaliar – métricas de software podem ser utilizadas para se avaliar produtos e processos, verificando se os mesmos atendem a seus critérios de aceitação. Medidas são os sensores que permitem saber quando projetos e processos estão se desviando de seus objetivos. Também se pode avaliar para identificar se os objetivos de qualidade estão sendo alcançados, e para identificar o impacto de melhorias em processos e tecnologias;

Controlar – dados podem ser utilizados para controlar o desenvolvimento de software, assim como em diversas outras áreas como engenharia e manufatura, por exemplo.

Prever – métricas de software podem ser utilizadas como base para a elaboração de estimativas, na construção de médias e tendências. Usar métricas para previsão envolve ganhar conhecimento sobre os relacionamentos entre um determinado produto ou processo, e criar modelos para esse relacionamentos então estabelecer valores a partir da observação desse valores.

3.2 Métricas de Software

Medição é o processo através do qual símbolos e números são designados a atributos de entidades do mundo real de uma maneira que os mesmos possam ser descritos através

de regras claramente definidas (FENTON; PFLEEGER, 1996). Uma entidade é uma pessoa, lugar, evento, período de tempo ou um outro objeto que será caracterizado pela medição (ISO-15939; ISO/IEC, 2002). Um atributo é uma característica ou propriedade de uma entidade (MCGARRY et al., 2002; ISO-15939; ISO/IEC, 2002).

Os benefícios de aplicar métricas envolvem:

- identificar metas de qualidade e aumentar o conhecimento das metas;
- prover um rápido feedback dos problemas de qualidade do processo de desenvolvimento;
- aumentar a satisfação dos clientes por quantificar a qualidade do software antes de entregar;
- prover uma base quantitativa para tomada de decisão sobre a qualidade do software;
- reduzir o custo do ciclo de vida do software por melhorar a eficiência do processo.

Em (FEITOSA, 2004) é possível encontrar um levantamento dos principais conceitos sobre métricas, que são resumidos a seguir.

Métrica básica é a medição de um único atributo de uma entidade através de um sistema de mapeamento. A mesma é independente de outras medidas e captura informação a respeito de um único atributo.

Método de medição é uma seqüência lógica de operações utilizadas na quantificação de um determinado atributo em relação a um determinado valor.

Escala de medição é um conjunto ordenado de valores, contínuo ou discreto, ou um conjunto de categorias, às quais os atributos são mapeados.

Unidade de medida é uma quantidade definida ou adotada por convenção, através da qual outras quantidades do mesmo tipo podem ser comparadas com o objetivo de expressar sua magnitude em relação à quantidade sendo mensurada.

Métrica Derivada é a medição definida como função de duas ou mais métricas básicas ou derivadas. A função para caracterizar uma métrica derivada deve ser uma função matemática entre duas ou mais métricas básicas.

Indicador é a métrica que fornece informações adicionais de estimativas ou avaliações de um determinado atributo. Ele representa a base para as atividades de análise e para a tomada de decisões. É definido à partir de um modelo de análise que define a relação entre duas ou mais métricas básicas e derivadas. Um indicador compara a métrica com um resultado esperado. Os indicadores permitem a tomada de decisões através da visão da real situação dos aspectos de um projeto.

3.2.1 Métricas para Engenharia de Requisitos

A engenharia de requisitos lida com elicitación, análise, comunicação e validação de requisitos. Quanto mais cedo os erros forem verificados e corrigidos mais fácil é corrigir se comparado com uma identificação tardia. Muitos dos problemas são causados devido a mudanças nos requisitos. Para eliminar problemas é necessário medi-los. No caso da engenharia de requisitos existem métricas já conhecidas que podem ser categorizadas (ALI, 2006; COSTELLO; LIU, 1995) em: métricas de tamanho, rastreabilidade de requisitos, volatilidade de requisitos, completude de requisitos e complexidade.

3.2.1.1 Métricas de Tamanho

O tamanho é uma métrica importante é usada para medir requisitos. Da mesma forma como Número de Linhas de Código medem o tamanho do software a contagem de requisitos pode ser usada para medir documentos de requisitos.

Casos de Uso também podem ser considerados como uma medida de tamanho quando usado para descrever requisitos. Por exemplo, o número de casos de uso, número de funções cobertas, etc. Bernardez, Duran e Genero (2004) apresenta um conjunto de métricas para casos de uso, elas envolvem número de atores, número de passos no fluxo principal do sistema e alternativos do sistema. Essas métricas são relacionadas a atributos de qualidade como complexidade, completude e facilidade de entendimento.

3.2.1.2 Métricas para Rastreabilidade

Rastreabilidade é a habilidade de rastrear requisitos em uma especificação da origem para um nível mais baixo ou mais alto em um conjunto de ligações no documento. As métricas para rastreabilidade fornecem informação que ajuda em determinar se todos os relacionamentos e dependências estão presentes. Elas ajudam a prevenir erros de interpretação de outras métricas (ALI, 2006). Essas métricas podem coletar informações

sobre o número de níveis que podem ser rastreados a partir de um requisito, o número de requisitos que tem ligações inconsistentes ou número de requisitos que não tem rastro nem acima nem abaixo.

3.2.1.3 Métricas de Volatilidade de Requisitos

Métricas para Volatilidade de Requisitos provêem uma forma rápida de determinar se o grau e as razões para as mudanças nos requisitos são consistentes com o processo de desenvolvimento corrente. O grau com que os requisitos mudam ao longo do tempo pode ser chamado de volatilidade de requisitos (COSTELLO; LIU, 1995). Essas métricas são usadas para encontrar razões para a mudança nos requisitos ao longo do tempo. Essas métricas consistem em contar as mudanças ocorrem ao longo do ciclo de vida dos requisitos, e classificá-las pela razão da mudança. Elas indicam mudanças como a adição, remoção e modificação de requisitos. Isso pode ajudar a rastrear futuras volatilidades de requisitos, projeto e código. Volatilidade pode ser alta nas fases iniciais de desenvolvimento de software, mas é reduzido com o progresso do projeto até que o desenvolvimento não seja mais afetado.

3.2.1.4 Métricas de Completude de Requisitos

Métricas de Completude de Requisitos são usadas para verificar se um requisito está no nível errado de hierarquia ou muito complexo. São procuradas evidências de que a especificação de requisitos ainda não concluída para todos os requisitos. Uma especificação de requisitos é dita completa se os requisitos estão todos contemplados por suas especificações, se nenhuma especificação precisa ser detalhada e se todas as necessidades dos usuários foram alcançadas (DAVIS et al., 1993).

Essas métricas podem ser classificadas em (ALI, 2006):

- Métricas para Decomposição de Requisitos - Essas métricas são usadas para saber se os requisitos especificados foram decompostos o suficiente para a próxima fase. Assim uma funcionalidade complexa pode ter muitos níveis enquanto uma funcionalidade simples terá poucos;
- Métricas para Desenvolvimento da Especificação de Requisitos - Essas métricas são usadas para prover informação sobre se o trabalho de especificação está completo o se ainda há trabalho pendente;

- Métricas para Especificação de Requisitos - Essas métricas provêm informação sobre a quantidade de itens para os quais ainda é necessário uma especificação.

3.3 Métricas para Complexidade

Algumas abordagens são utilizadas para medir a complexidade em modelos. Entre elas a complexidade ciclomática de McCabe (MCCABE, 1976), as medidas de complexidade de Halstead (HALSTEAD, 1977) e uma medida de complexidade baseada em Entropia (KIM; SHIN; WU, 1995).

A complexidade ciclomática de McCabe (MCCABE, 1976) utiliza conceitos da teoria dos grafos para avaliar a complexidade de programas. Ele mede o código do programa em termos dos caminhos possíveis através das estruturas de decisão. As estruturas de decisão podem ser estruturas de controle como os laços (e.g., *while* ou *until*) ou estruturas condicionais (i.e., *if*). Cada módulo do programa é tratado como um vértice no grafo e a passagem entre os módulos são tratadas como arestas no grafo. Além dos vértices (N na equação 3.1) e arestas (E , na equação 3.1) também são contados o número de componentes conexos. Os componentes conexos (C , na equação 3.1) representam os sub-grafos do grafo principal que não estão ligados entre si.

$$CC = E - N + 2C \quad (3.1)$$

As métricas de complexidade de Halstead (HALSTEAD, 1977) oferecem estimativas de complexidade segundo o tamanho e o vocabulário utilizado para descrever determinado módulo. Originalmente proposta para medir código-fonte de aplicações ela conta o tamanho segundo a quantidade de operadores e operandos. O vocabulário é a medida do número distinto de operadores e operandos. Combinando tamanho e vocabulário é possível obter uma estimativa do volume, dificuldade e esforço para manter determinado módulo.

Além das métricas já apresentadas uma outra medida para complexidade é a medida de Entropia (KIM; SHIN; WU, 1995), que é uma medida quantitativa sobre incerteza em um conjunto de dados. Em teoria da informação a incerteza na informação é geralmente calculada pela entropia da informação, frequentemente chamada Entropia de Shannon (SHANNON, 1948). Uma seqüência de símbolos desenhada para um alfabeto pode ser considerada uma mensagem. O campo da teoria da informação lida com a medida da

quantidade de informação contida em uma mensagem. A quantidade de informação se refere não somente à mensagem como um todo, mas ao que está contido em cada símbolo da mensagem. Os símbolos individualmente vão conter informação que é inversamente proporcional à probabilidade de o símbolo aparecer em uma mensagem. Quanto mais provável é o aparecimento de um símbolo menor é a quantidade de informação que ele passa. A informação também é aditiva, isto é, a quantidade total de informação coberta por dois símbolos é a soma de seu conteúdo individual.

Em (KIM; SHIN; WU, 1995) é proposta uma métrica para medição da complexidade baseada na probabilidade referencial. Essa probabilidade é calculada a partir da quantidade de referências que determinada unidade emite ou recebe. Eles apresentam exemplos da utilização desta métrica para código fonte e para ligação entre módulos de um sistema (KIM; SHIN; WU, 1995). No caso do código-fonte são contadas as referências feitas às variáveis e funções, cada manipulação de variável ou chamada de função é computada como referência a unidade. Para calcular a probabilidade são contados o número de elementos. Com base na probabilidade é calculada a medida de entropia para o modelo avaliado através da equação:

$$H = - \sum_{i=1}^q p(x_i) \times \log_2 p(x_i) \quad (3.2)$$

3.4 O Método GQM

Para a definição e avaliação das métricas é recomendado adotar alguma abordagem para esse propósito. Para a análise dos resultados das métricas esse trabalho utiliza o método GQM (BASILI; CALDIERA; ROMBACH, 1994).

O método GQM (Goal Question Metric) foi criado por Basili, Caldiera e Rombach (1994) com o objetivo de suportar as organizações na institucionalização de processos de medições, especificamente na identificação de objetivos que serão traduzidos em medições quantitativas. O GQM define orientações para:

- Definir os principais objetivos que serão tratados no programa de medições (Goal);
- Identificar um conjunto de questões que ajudem o atendimento dos objetivos (Question);
- Definição e recuperação de dados que respondam as questões identificadas (Metric).

O modelo GQM possui uma estrutura hierárquica composta por três níveis, conforme ilustrado na Figura 6. O nível 1 é considerado o nível conceitual, o nível dos objetivos. Objetivos são estabelecidos para as diversas entidades de uma organização, como por exemplo: seus projetos, produtos e processos. O nível 2 é considerado o nível operacional, o nível das questões. As questões são identificadas para caracterizar o alcance de um objetivo específico. O nível 3 é considerado o nível quantitativo, o nível das métricas. Um conjunto de dados é associado a uma determinada questão com o objetivo de respondê-la de maneira quantitativa.

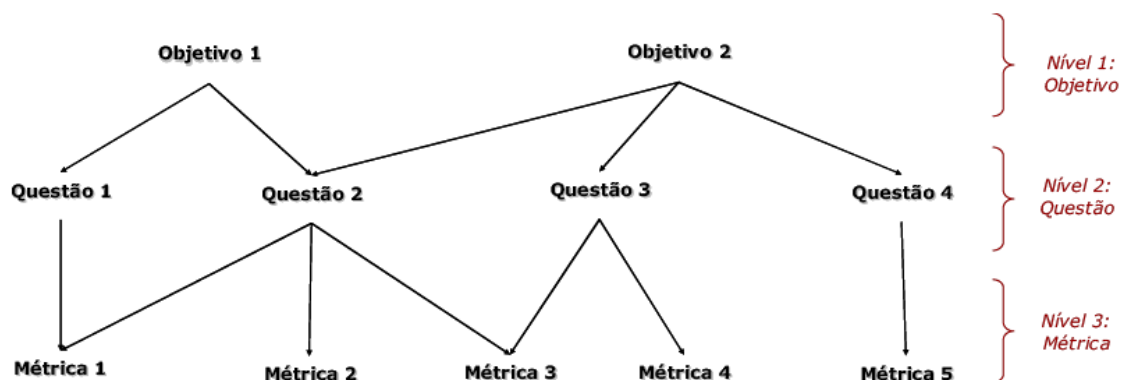


Figura 6: Método GQM (Goal Question Metric)

O objetivo é refinado em diversas questões. Por sua vez, cada questão é refinada em métricas, podendo ser elas objetivas ou subjetivas. Questões podem ser identificadas para mais de um objetivo, e métricas também podem pertencer a mais de uma questão e conseqüentemente endereçar vários objetivos. As métricas devem endereçar os diferentes pontos de vista de cada objetivo. O método GQM se baseia na análise hierárquica dos atributos de qualidade com o intuito de definir como cada atributo vai ser analisado através de perguntas que são respondidas com as métricas (PARK; GOETHERT; FLORAC, 1996).

3.5 Conclusão

Nesse capítulo foi feita uma revisão dos principais conceitos da medição de software e de métricas na engenharia de requisitos. Foram apresentadas as principais motivações para o uso de métricas como a prevenção, avaliação, controle e entendimento (HUMPHREY, 1989). Além disso, o uso de métricas reduz a subjetividade na tomada de decisão (IEEE-1061, 1992) baseando as decisões em dados e não mais em opiniões.

Além de motivar o uso de métricas foram apresentados conceitos que serão usados ao longo da dissertação como as métricas para complexidade de McCabe (1976), Halstead (1977) e Kim, Shin e Wu (1995). O método GQM foi introduzido por servir como base do exemplo de aplicação a ser apresentado no Capítulo 5.

*4 Métricas para i^**

Nesse capítulo serão definidas métricas para avaliar a qualidade de modelos i^* . Na seção 4.2 é definida uma métrica para Erros Típicos, na seção 4.3 são apresentadas métricas para o nível de detalhamento, na seção 4.4 são apresentadas métricas para ambigüidade e na seção 4.5 são apresentados mapeamentos para métricas de complexidade.

4.1 Introdução

A definição das métricas para i^* segue o princípio de tornar o uso das métricas o mais objetivos possível na tentativa de reduzir a necessidade de especialistas para coleta das mesmas (IEEE-1061, 1992).

Fontes da literatura foram avaliadas para identificar o que poderia ser importante medir no i^* , entre outros problemas que poderiam ser identificados com o uso de métricas estão: a presença de erros nos modelos; o nível de detalhamento; a presença de ambigüidades; e a complexidade dos modelos.

A presença de erros típicos se deve ao mau uso da notação do i^* como pode ser visto em Webster, Amaral e Cysneiros (2005), Horkoff (2006) e Grau et al. (2008).

O nível de detalhamento é uma propriedade investigada para permitir a comparação de modelos i^* . O nível de detalhamento se mostra importante por permitir determinar se um modelo está mais detalhado que outro, além de oferecer indícios da ausência das construções básicas do i^* (e.g., decomposição, contribuição, *means-ends*, etc.).

Da mesma forma que outras formas de representação de requisitos o i^* também apresenta problemas com a presença de ambigüidade. Esse problema se deve tanto a natureza ambígua da linguagem natural, que é utilizada para descrever os elementos intencionais (i.e., metas, tarefas, recursos e *softgoals*), quanto ao uso inadequado de construções do próprio i^* .

Outro problema já relatado nos modelos i^* é a grande complexidade atingida pelos modelos à medida que são analisados (ALENCAR et al., 2006). Os modelos i^* podem atingir um nível de complexidade tão grande que prejudique a representação dos requisitos e o entendimento deles.

Outras propriedades também poderiam ser estudadas para definição de métricas como a consistência, completude, rastreabilidade e volatilidade de requisitos. No entanto, elas não foram consideradas para o escopo dessa dissertação. Métricas que dependem da experiência do avaliador para serem coletadas não foram tratadas por ferir o princípio de redução da subjetividade apresentado em IEEE-1061 (1992). Além disso, o i^* não possui um processo bem padronizado o que impede o avaliador de fazer inferências sobre a forma como os modelos foram obtidos. Por esses motivos algumas propriedades que dependem fortemente do conhecimento do avaliador como a consistência e completude dos modelos não foram consideradas. Da mesma forma não foram consideradas métricas

que dependem do processo como rastreabilidade e volatilidade.

Em alguns casos as propriedades estudadas já foram largamente abordadas na literatura de métricas, já tendo sido aceitas e validadas em outras oportunidades. Por isso optou-se por reusar métricas já existentes sempre que a abstração se adequa aos modelos i^* . Essas métricas foram adaptadas ao i^* através de mapeamentos como pode ser visto no caso das métricas para complexidade (vide Seção 4.5).

Para documentar as métricas foi adotado o *template* de métricas do IEEE-1061 (1992). O *template* oferece uma estrutura para as métricas (vide Tabela 50, no Anexo A) e outra para os dados (vide Tabela 51, no Anexo A). Para cada instância de métrica definida nessa dissertação serão apresentadas ao menos duas tabelas: uma para a métrica e uma (ou mais) para os dados necessários. Esse *template* faz a divisão entre métrica e dados para organizar as informações. Na estrutura para métricas são apresentados itens que descrevem as informações para entender a métrica, tais como o nome da métrica, os dados necessários para obter a métrica e os cálculos. Na estrutura para dados estão descritos itens referentes aos dados como o nome do dado, quais métricas precisam deles e as formas de coleta.

Serão apresentadas 13 métricas, sendo uma para erros típicos (E_1), três para nível de detalhamento (D_1 , D_2 e D_3), duas para ambigüidade (A_1 e A_2) e sete para complexidade (C_1 , C_2 , C_3 , C_4 , C_5 , C_6 , C_7). As métricas estão agrupadas de acordo com o que medem, elas podem ser métricas: erros típicos; nível de detalhamento; ambigüidade; e complexidade. A seção 4.2 apresenta uma descrição dos principais erros e maus usos encontrados na literatura sobre i^* e a métrica para documentar esses erros. A seção 4.3 apresenta um conjunto de métricas para avaliar o nível de detalhamento dos modelos i^* . A seção 4.4 apresenta métricas para avaliar os modelos segundo a possibilidade de existência de ambigüidade. A seção 4.5 apresenta métricas adaptadas para medir modelos i^* segundo a complexidade gráfica dos mesmos.

À medida que as métricas forem instanciadas serão apresentadas ilustrações de como elas podem ser obtidas a partir dos modelos. As figuras criadas para esse propósito são baseadas no exemplo *Media Shop* adaptado do original (CASTRO; KOLP; MYLOPOULOS, 2002), vide Figura 7. *Media Shop* é uma loja de venda de diferentes tipos de item como livros, jornais, revistas, CDs e semelhantes. Os Clientes do *Media Shop* podem usar para fazer pedidos um catálogo periodicamente atualizado descrevendo os itens de mídia disponíveis. *Media Shop* é abastecido com os mais novos lançamentos de um produtor de mídia e através de um catálogo pelo Fornecedor de Media. Para aumentar a participação

no mercado, *Media Shop* decidiu abrir um B2C (*Business to Customer*) para proporcionar vendas na Internet. Com uma nova opção, um consumidor pode pedir itens pessoalmente, por telefone, ou através da Internet. O sistema foi chamado *Medi@* e está disponível na Internet. O objetivo básico do sistema é permitir ao Cliente examinar o catálogo on-line e fazer pedidos. Os Clientes em potencial podem pesquisar a loja on-line navegando no catálogo ou através de consultas na base de dados de itens.

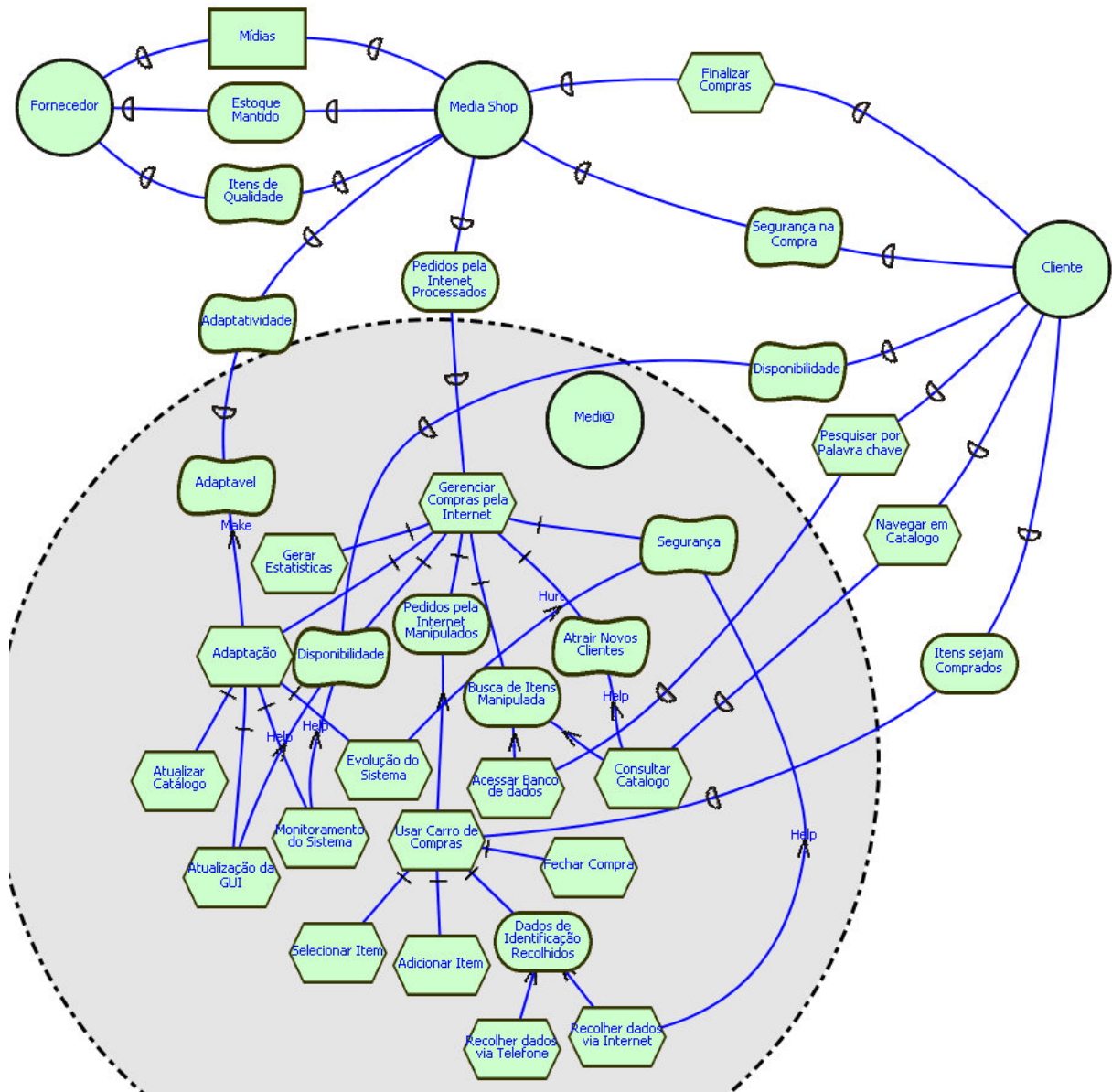


Figura 7: Exemplo do Media Shop

4.2 Métrica para Erros Típicos

O i^* possui uma descrição da sintaxe da linguagem (YU, 1995), mas as construções presentes nessa linguagem podem conter erros (WEBSTER; AMARAL; CYSNEIROS, 2005). Isso ocorre devido ao mau uso do i^* , que pode ser causado por falta de experiência ou não entendimento dos conceitos representados pelas construções.

Modelos com erros podem prejudicar o entendimento do leitor impedindo a compreensão do que está sendo modelado. Nesse sentido é necessário garantir que os modelos não contem erros, para isso foram revisadas fontes da literatura que tratam de erros em modelos i^* como (GRAU et al., 2008), (HORKOFF, 2006) e (WEBSTER; AMARAL; CYSNEIROS, 2005). Atualmente foi criado um catálogo com os principais erros e maus usos, em um total de 13 apresentados no Apêndice A. Os principais erros e maus usos na notação do i^* estão agrupados de acordo com as seguintes categorias:

1. Atores e relacionamentos entre atores
 - (a) Atores sem ligação
 - (b) Atores dentro da fronteira de outros atores
 - (c) Uso de nomes inadequados em atores
 - (d) Ligar Atores com ligação de dependência sem usar um *dependum*
2. Dependências
 - (a) Uso de outras ligações em vez das ligações de dependências
 - (b) Uso inadequado de *dependums* (*softgoals* como metas, tarefas como *softgoals*, etc.)
 - (c) Formação de ciclos entre *dependums* de atores diferentes
3. Elementos internos e relacionamentos entre elementos internos
 - (a) Deixar elementos sem ligações
 - (b) Uso de ligações em situações irregulares (ligação de dependência dentro da fronteira do ator)
 - (c) Elementos do SR fora de fronteira do ator correspondente
 - (d) Decompor metas em sub-metas ou sub-tarefas
 - (e) Decompor *softgoals* em *sub-softgoals* ou sub-tarefas

- (f) Contribuição para metas, tarefas ou recursos
- (g) *Means-Ends* onde uma meta é um meio
- (h) Ligação direta entre elementos internos de dois atores diferentes

Para facilitar a identificação de erros em modelos i* foi organizado um novo catálogo com os principais erros (ver Apêndice A). Assim, seu uso torna mais simples a leitura de modelos em busca de erros reduzindo a necessidade de ler várias fontes com a mesma finalidade. O catálogo constitui um recurso de inspeção que visa guiar a leitura dos modelos aumentando as chances de identificar erros, nele foram resumidos os erros mais comuns para que se tornassem evidentes. As entradas do catálogo apresentam os erros com uma descrição, exemplos de como seria a representação com e sem o erro, e perguntas para guiar a identificação do erro. Essas entradas funcionam como padrões na identificação, sempre que um deles for verificado no modelo é sinal que existe um erro. No exemplo da Tabela 1 é apresentado o erro de existirem atores sem ligação com outros atores, um ator que não se conecta a outros atores não contribui com nenhum tipo de informação para o modelo. Para identificar esse erro em um modelo podem ser usadas perguntas como *Existem atores se nenhum tipo de dependência e sem ligações de estruturação com outros atores (i.e., Is-A, Is-part-of, Covers, Occupies, Plays)?*. O exemplo de uso incorreto mostra um ator no modelo Media Shop sem nenhuma ligação, o que não aparece no exemplo de uso correto.

Tabela 1: Dangling Actor

01 - Dangling Actor	Actor without link to another Actor
Details	An Actor without link to an other Actor does not contribute with information to model
Questions	Are there Actors without dependencies or structural actor's links (i.e., Is-A, Is-part-of, Covers, Occupies, Plays)?
Correct	Wrong

Nessa dissertação é proposta uma métrica para tratar de erros típicos: a métrica

Número de Erros (E_1), vide tabela 2. É usada para documentar a presença de erros em modelos i^* e permite obter a informação sobre a quantidade de erros que ocorrem em um determinado modelo. A métrica Número de Erros apresenta impacto sobre a interpretação de outras métricas. Modelos com erros podem apresentar resultados falsos para outras métricas já que não há garantia que a informação coletada corresponde ao que é esperado devido a presença de erros. O valor ideal para essa métrica é zero o que indica que o modelo não apresenta erros típicos. Assim, a presença de erros pode indicar a necessidade de revisar o modelo para corrigi-los. Para calcular a métrica é necessário o dado Erros Típicos (vide Tabela 3), o valor da métrica corresponde exatamente ao valor do dado. Para usar essa métrica é necessário conhecer a técnica i^* para que o avaliador consiga identificar os erros típicos.

Tabela 2: Informações sobre a métrica Número de Erros

Item	Descrição
Nome	Número de Erros (E_1)
Impacto	A métrica Número de Erros é decisiva na identificação de quando revisar os modelos i^* , modelos com muitos erros de notação não serão compreensíveis podendo até mesmo prejudicar a interpretação de outras métricas.
Valor Alvo	O valor ideal para a métrica é 0 que indica que não há erros básicos no uso da técnica i^* .
Ferramentas	Uma planilha pode ser utilizada para armazenar os valores das métricas
Aplicação	Permite identificar quantos erros no uso na técnica i^* foram encontrados
Dados	Número de Erros (NErros)
Cálculos	Equivale ao valor do dado Erros Típicos
Interpretação	A presença de erros indica os modelos que devem ser revisados para que sejam corrigidos
Considerações	O valor de aceitação da métrica pode variar dependendo da tolerância da equipe de avaliadores. No caso padrão deve-se usar o valor alvo definido como zero.
Treinamento Requerido	Conhecimento na técnica i^*

O dado Erros Típicos (vide Tabela 3) corresponde ao número de erros típicos encontrados nos modelos. O procedimento para coleta é feito através da inspeção dos modelos em busca dos erros. Um forma de fazer isso é se guiar pelo catálogo dos erros (vide Apêndice A), com apresentado anteriormente. Para cada erro encontrado no modelo o valor do dado Erros Típicos é incrementado em um.

Tabela 3: Informações sobre o dado Erros Típicos

Item	Descrição
Nome	Erros Típicos
Métrica	Número de Erros (E_1)
Definição	Corresponde ao número erros encontrados nos modelos avaliados.
Fonte	Diagramas SD e SR dos modelos i^* .
Coletor	Avaliador do modelo
Procedimentos	Inspecionar modelos em busca de erros ou maus usos de notação nas construções, para cada erro encontrado o valor do dado é incrementado em um.
Armazenagem	Planilha eletrônica ou tabela com relação das métricas
Representação	Valor numérico inteiro maior ou igual a zero.

4.3 Métricas para Nível de Detalhamento

Para avaliar o nível de detalhamento de modelos i^* são propostas métricas baseadas em construções básicas do i^* . Elas são inspiradas em propriedades de análise descritas em (YU, 1995), e têm como objetivo oferecer indícios que o modelo avaliado está pouco detalhado em alguma das construções do i^* . As propriedades apresentadas no capítulo 2 são a *ability*, *commitment* e *viability*¹. Essas propriedades não serão avaliadas exaustivamente, o que está sendo procurado com as métricas são evidências que não os modelos não foram detalhados o suficiente para atingi-las. Para maiores detalhes ver outras fontes como Yu (1995) e Horkoff (2006).

Um modelo que atende as propriedades de análise é considerado mais detalhado que modelos que não as atendem. Para definir métricas para avaliar nível de detalhamento, partimos da hipótese que as análises em relação aos elementos do i^* foram realizadas, assim um modelo mais detalhado apresentaria as seguintes características:

- as metas que estão dentro da fronteira de um ator, estão ligados a rotinas indicando como serão realizados;
- as dependências estão ligadas à elementos internos nos atores, para indicar o “porquê” de sua existência e “como” serão satisfeitas;
- os *softgoals* que estão dentro da fronteira dos atores devem receber contribuições que indicam se serão satisfeitos ou não.

Possíveis falhas no detalhamento dos modelos podem ser apontadas com base nas métricas apresentadas a seguir. São apresentadas três métricas: Taxa de Metas sem ligação com Rotinas (D_1); Taxa de Dependências sem Ligação com Elementos Internos no Ator (D_2); e Taxa de *Softgoals* sem Contribuição (D_3).

¹em tradução livre seria habilidade, comprometimento e viabilidade

4.3.1 Taxa de Metas sem ligação com Rotinas

A métrica Taxa de Metas sem ligação com Rotinas (vide Tabela 4), relaciona o percentual de metas no modelo SR que estão ligadas a rotinas através de uma ligação *Means-Ends*. Quando uma meta está ligada a uma rotina indica que há ao menos uma alternativa para se alcançar a meta. Nesse caso serão procurados indícios da existência da rotina.

A figura 8 apresenta um exemplo de meta sem ligação com rotinas. Nesse caso a meta *Dados de Identificação Recolhidos* não possui uma ligação com rotina ela poderia ser capturada pela métrica Taxa de Metas sem ligação Com Rotinas.

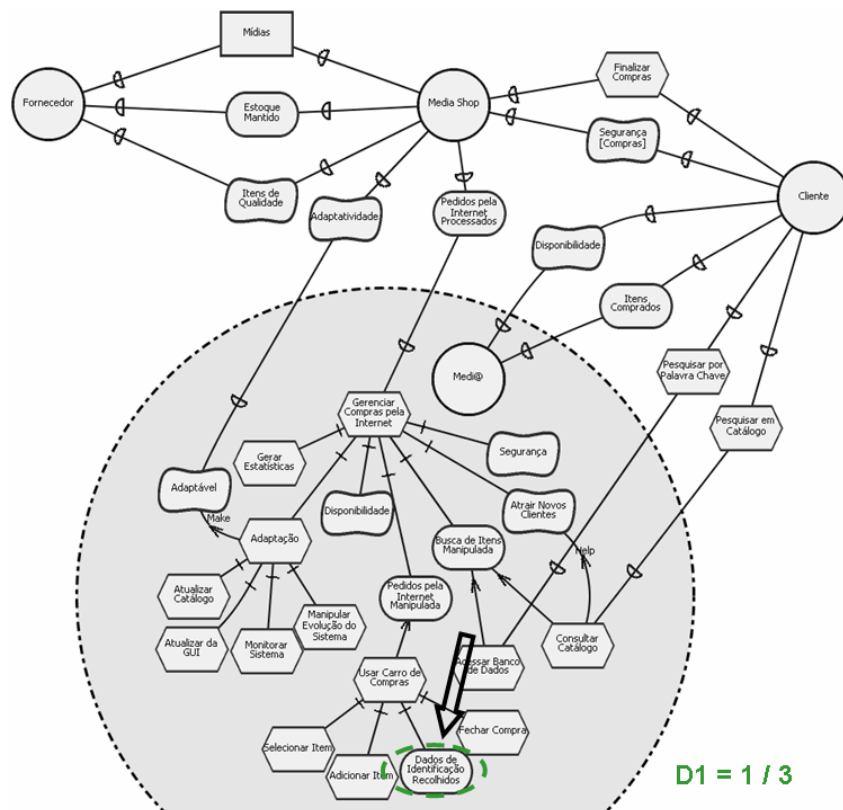


Figura 8: Exemplo de Metas se ligação com Rotinas

A tabela 4 apresenta a métrica Taxa de Metas sem ligação com Rotinas. Essa métrica permite comparar modelos i^* quanto ao detalhamento. O valor da métrica é obtido através da contagem do total de metas com rotinas, dividido pelo total de metas na fronteira do ator, esse valor sempre será um valor entre zero e um. O valor ideal seria zero onde nenhuma das metas presentes no modelo ficaram sem ligação com as rotinas. Valores próximos de um indicam que o modelo apresenta deficiências na análise de alternativas para as metas, isto é, não foram consideradas durante a análise quais são as alternativas de se alcançar as metas.

Tabela 4: Informações sobre a métrica Taxa de Metas sem ligação com Rotinas

Item	Descrição
Nome	Taxa de Metas sem ligação com Rotinas (D_1).
Impacto	Permite comparar modelos quanto ao detalhamento
Valor Alvo	O valor ideal para a métrica é 0, que indica que todas as metas foram mapeados em rotinas que os tornam operacionais.
Ferramentas	Planilha eletrônica ou tabela com relação das métricas
Dados	Número de Metas sem ligação com Rotinas (NMR) Número de Metas (NM)
Cálculos	$\frac{NMR}{NM} \quad (4.1)$
Interpretação	O valor igual a 0 indica que todos as metas tem uma rotina correspondente, valores próximos a 1 indicam que o modelo tem deficiência na análise.
Considerações	O uso dessa métrica não garante que a propriedade de análise na qual foi inspirada seja satisfeita, ela apenas apresenta indícios de que há problemas para satisfazer a propriedade.
Treinamento Requerido	Conhecimento da linguagem i^*

Na tabela 5 é apresentado o dado Número de Metas sem ligação com Rotinas (NMR), que é necessário para calcular a métrica D_1 . O procedimento de coleta consiste em contar o número de metas que estão na fronteira dos atores que estão abertas no modelo SR, e entre elas observar quais estão sem ligação com rotinas. O valor obtido é um valor numérico maior ou igual a zero ($NMR \geq 0$).

Tabela 5: Informações sobre o dado Número de Metas sem Rotina

Item	Descrição
Nome	Número de Número de Metas sem Rotina (NMR)
Métrica	Taxa de Número de Metas sem Rotina (D_1).
Definição	Corresponde ao número de metas que estão nos modelos SR e que não estão ligadas a uma rotina através de uma ligação <i>Means-Ends</i> .
Fonte	Diagramas SR dos modelos i^* .
Coletor	Avaliador do modelo
Procedimentos	Contagem manual.
Armazenagem	Planilha eletrônica ou tabela com relação das métricas
Representação	Valor numérico inteiro maior ou igual a zero.

4.3.2 Taxa de Dependências sem Ligação a Elementos Internos no Ator

Quando um ator está ligado a outros através de relacionamentos de dependência, os atores *dependee* passam a se comprometer, em algum grau, com o ator *depender* para que a dependência seja realizada. Em modelos SR onde a fronteira dos atores é aberta, eles passam a ligar os *dependums* a elementos internos explicando o “porquê” da existência da dependência. Para obter evidências que um ator *dependee* está apto a realizar as dependências está sendo proposta a métrica Taxa de Dependências sem Ligação a Elementos Internos no Ator (D_2). Essa métrica oferece informação de quantas dependências ainda não estão ligadas a elementos internos.

A figura 9 mostra a situação em que duas dependências entram na fronteira do ator **Medi@**. A dependência de Meta **Itens Comprados** e a dependência de *softgoal* **Disponibilidade** estão ligadas ao ator *Medi@* sem se ligar à nenhum elemento internos desse ator. Esse caso poderia ser capturado pela métrica Taxa de Dependências sem Ligação a Elementos Internos no Ator.

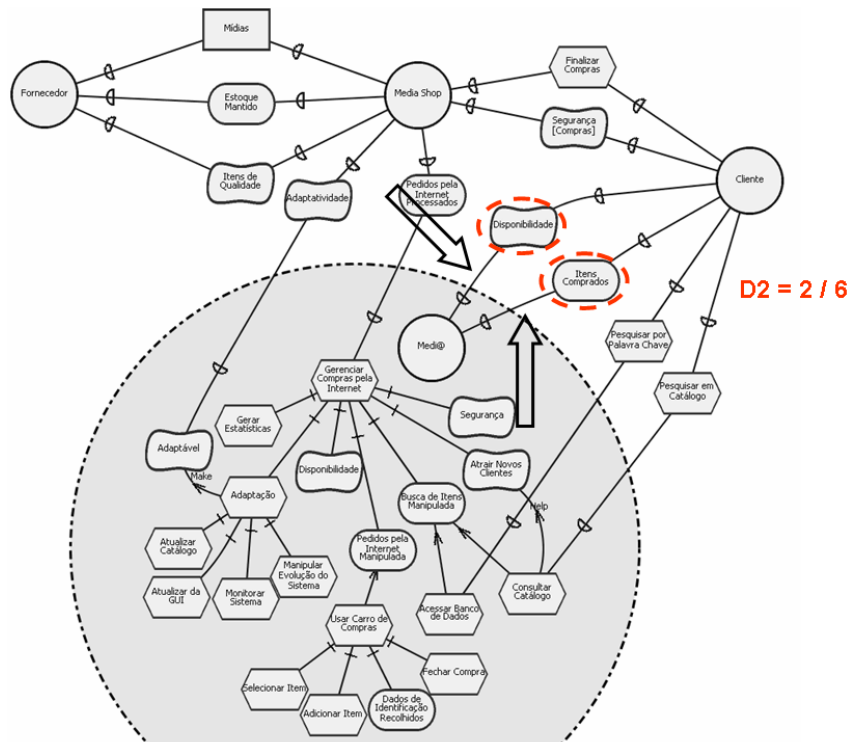


Figura 9: Exemplo de Dependências sem ligação a Elementos Internos no Ator

Tabela 6: Informações sobre a métrica Taxa de Dependências sem Ligação a Elementos Internos no Ator

Item	Descrição
Nome	Taxa de Dependências sem Ligação a Elementos Internos no Ator (D_2)
Impacto	Permite comparar modelos quanto ao detalhamento
Valor Alvo	O valor ideal para a métrica é 0 que indica que todas as dependências de um ator estão ligadas a elementos internos.
Ferramentas	Uma planilha pode ser utilizada para armazenar os valores das métricas
Dados	Número de Dependências sem Ligação à Elemento Interno no Ator (NDLEIA) Número de Dependências do Ator (NDA)
Cálculos	$\frac{NDLEIA}{NDA} \quad (4.2)$
Interpretação	Valores próximos de 0 indicam que a maior parte das dependências estão ligadas a elementos internos no ator.
Considerações	O uso dessa métrica não garante que a propriedade de análise na qual foi inspirada seja satisfeita, ela apenas apresenta indícios de que há problemas para satisfazer a propriedade.
Treinamento Requerido	Conhecimento na técnica i^*

A tabela 6 apresenta a métrica Taxa de Dependências sem Ligação a Elemento Interno no Ator. Ela representa o percentual de dependências que não estão ligadas a elementos na fronteira dos atores. O valor dessa métrica pode variar de 0 quando todas as dependências estiverem ligadas a elementos internos nos atores, até 1 quando todas as dependências estiverem sem ligação a elementos internos na fronteira dos atores. Essa métrica permite comparar modelos quanto ao detalhamento e apontar possíveis deficiências na análise do modelo.

Para calcular o valor de D_2 são necessários o número de dependências sem ligação a elementos internos no ator (NDLEIA, tabela 7) e números de dependências do ator (NDA, tabela 8). O valor assumido por NDLIEA pode variar de 0 quando não há dependências ligadas a elementos internos na fronteira do ator, até o valor de NDA, quando todas as dependências que serão satisfeitas pelo ator estão ligadas a elementos internos. O valor assumido por NDA pode variar de 0, quando não há dependências que o ator deva satisfazer até o número total de dependências direcionadas para ator.

Tabela 7: Informações sobre o dado Número de Dependências sem Ligação a Elementos Internos do Ator

Item	Descrição
Nome	Número de Dependências sem Ligação a Elementos Internos do Ator (NDLEIA)
Métrica	Taxa de Dependências sem Ligação a Elementos Internos do Ator (D_2)
Definição	Corresponde ao Número de Dependências que não estão ligadas a elementos internos na fronteira do ator
Fonte	Diagramas SR dos modelos i^* .
Coletor	Avaliador do modelo
Procedimentos	Contar entre as dependências do ator em foco quais não estão ligadas a um elemento interno.
Armazenagem	Planilha eletrônica ou tabela com relação das métricas
Representação	Valor numérico inteiro maior ou igual a zero.

Tabela 8: Informações sobre o dado Número de Dependências do Ator

Item	Descrição
Nome	Número de Dependências do Ator (NDA)
Métrica	Taxa de Dependências sem Ligação a Elementos Internos do Ator (D_2)
Definição	Corresponde ao número de dependências que estão entrando na fronteira do ator em foco
Fonte	Diagramas SR dos modelos i^* .
Coletor	Avaliador do modelo
Procedimentos	Contar dependências que estão entrando nas fronteiras dos atores
Armazenagem	Planilha eletrônica ou tabela com relação das métricas
Representação	Valor numérico inteiro maior ou igual a zero.

4.3.3 Taxa de *Softgoals* que não são tratados por Contribuição

Nos modelos do i^* os requisitos não-funcionais são representados por *softgoals*. Portanto, quando capturadas em modelos SR eles podem ser analisados através de contribuições (vide capítulo 2) para indicar se serão ou não satisfeitos e em que grau isso pode ocorrer. Os *softgoals* podem receber contribuições positivas ou negativas vindas de outros *softgoals* ou de tarefas. As contribuições também podem ser parciais permitindo que os *softgoals* sejam analisados em diferentes níveis de satisfação.

Quando presentes em rotinas os *softgoals* podem ser usados para analisar a viabilidade daquela rotina nos atores. Contudo, se não houver contribuições para os *softgoals* é impossível fazer esta análise. Para procurar indícios que esta análise foi feita é proposta a métrica Taxa de *Softgoals* tratados por Contribuição (D_3).

A figura 10 apresenta um exemplo de *softgoals* que não recebem contribuições. O *softgoal* *Segurança* e o *softgoal* *Disponibilidade* não são tratados através de contribuições. Esse caso seria capturado pela métrica Taxa de *Softgoals* que não são tratados por Contribuição.

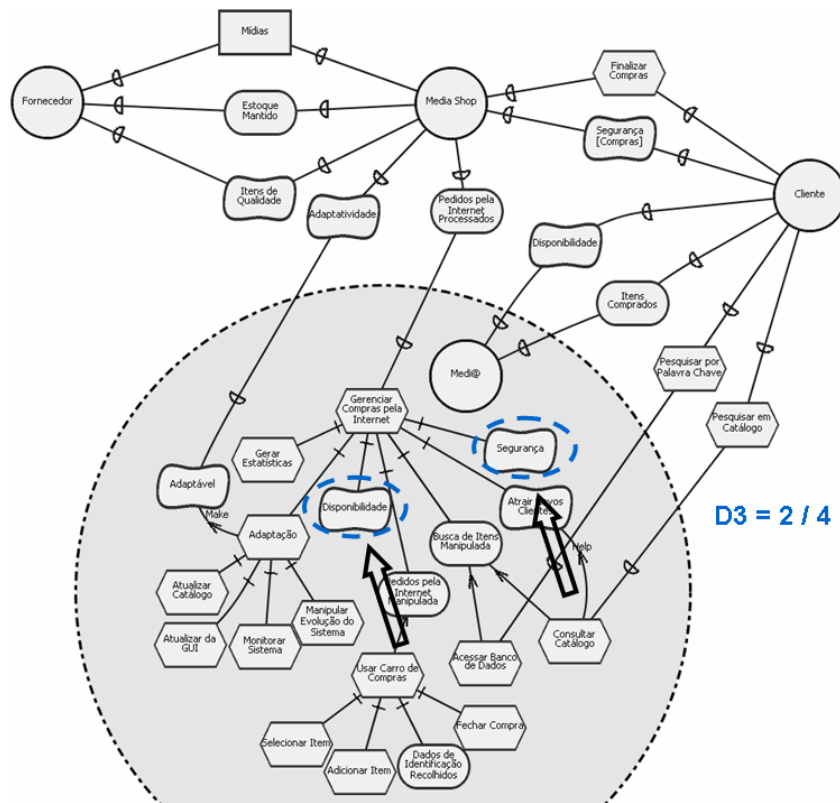


Figura 10: Exemplo de *Softgoals* que não são tratados por Contribuição

A tabela 9 apresenta a métrica Taxa de *Softgoals* que não são tratados por Contribuição. Essa métrica se aplica a comparação de modelos quanto ao detalhamento, além de indicar problemas na análise de *softgoals* dos modelos. O valor da métrica pode variar de 0, quando todos os *softgoals* são tratados por contribuição, até 1 quando nenhum deles é tratado por contribuição. Portanto, valores próximos de 1 indicam pode haver problemas na análise dos *softgoals* dos modelos. A ausência de *softgoals* nos modelos podem impede o cálculo da métrica.

Tabela 9: Informações sobre a métrica Taxa de *Softgoals* que não são tratados por Contribuição

Item	Descrição
Nome	Taxa de <i>Softgoals</i> tratados por Contribuição (D_3)
Impacto	Permite compara modelos quanto ao detalhamento
Valor Alvo	O valor ideal para a métrica é 0 que indica que os <i>softgoals</i> de um ator podem são tratadas através de contribuições.
Ferramentas	Uma planilha pode ser utilizada para armazenar os valores das métricas
Dados	Número de <i>Softgoals</i> não tratados por Contribuição (NSC) Número de <i>Softgoals</i> no Ator (NSA)
Cálculos	$\frac{NSC}{NSA} \quad (4.3)$
Interpretação	Valores próximos de 0 indicam que maior parte dos <i>softgoals</i> são tratados a através de contribuição. Valores próximos de 1 indicam que maior parte dos <i>softgoals</i> não são tratados por contribuições, o que pode indicar uma deficiência na análise do modelo
Considerações	O uso dessa métrica não garante que a propriedade de análise na qual foi inspirada seja satisfeita, ela apenas apresenta indícios de que há problemas para satisfazer a propriedade.
Treinamento Requerido	Conhecimento na técnica i*

Para calcular a métrica D_3 são necessários o Número de *Softgoals* não tratados por Contribuição (NSC) e o Número de *Softgoals* no Ator (NSA). A tabela 10 apresenta o Número de *Softgoals* não tratados por Contribuição (NSC) esse dado é obtido a partir da contagem dos *softgoals* presentes na fronteira de um ator e que não recebem contribuições. O valor de NSC é um inteiro maior ou igual a zero ($NSC \geq 0$).

Tabela 10: Informações sobre o dado Número de *Softgoals* não tratados por Contribuição

Item	Descrição
Nome	Número de <i>Softgoals</i> tratados por Contribuição (NSC)
Métrica	Taxa de <i>Softgoals</i> tratados por Contribuição (D_3)
Definição	Corresponde ao número de <i>softgoals</i> que estão na fronteira aberta de um ator e não são tratados por contribuição.
Fonte	Diagramas SR dos modelos i^* .
Coletor	Avaliador do modelo
Procedimentos	Contagem dos <i>softgoals</i> na fronteira do ator e que não são tratados por contribuição, isto é, que não recebem nenhuma ligação de contribuição.
Armazenagem	Planilha eletrônica ou tabela com relação das métricas
Representação	Valor numérico inteiro maior ou igual a zero.

A tabela 11 apresenta o dado Número de *Softgoals* no Ator, esse dado corresponde ao número total de *softgoals* presentes nas fronteiras dos atores em modelos SR. O valor de NSA é um número inteiro maior ou igual a zero ($NSA \geq 0$), quando esse valor atingir zero a métrica D_3 não poderá ser calculada.

Tabela 11: Informações sobre o dado Número de *Softgoals* no Ator

Item	Descrição
Nome	Número de <i>Softgoals</i> no Ator (NSA)
Métrica	Taxa de <i>Softgoals</i> tratados por Contribuição (D_3)
Definição	Corresponde ao número de <i>softgoals</i> que estão na fronteira aberta de um ator
Fonte	Diagramas SR dos modelos i^* .
Coletor	Avaliador do modelo
Procedimentos	Contagem dos <i>softgoals</i> que estão presentes na fronteira do ator em foco.
Armazenagem	Planilha eletrônica ou tabela com relação das métricas
Representação	Valor numérico inteiro maior ou igual a zero.

4.4 Métricas para Ambigüidade

O uso de linguagem natural pode gerar ambigüidade e mal entendimento em documentos de requisitos (BERRY; KAMSTIES; KRIEGER, 2003), este problema também ocorre com modelos i^* . No caso do i^* , a linguagem natural é utilizada para descrever os rótulos dos elementos intencionais como Tarefas, Recursos, Metas e *SoftGoals*, isso pode levar a de mal-entendimentos e ambigüidades em modelos i^* .

Este problema é comum e pode ser reduzido adotando convenções para os rótulos dos elementos. Como pode ser visto em (GRAU et al., 2005) são usados nome padronizados para os elementos i^* . Verbos na voz ativa são usados para nomear Tarefas, verbos na voz passiva são usados para nomear Metas e substantivos são usados para nomear Recursos. No caso de *softgoals* podem ser usados verbos na voz passiva acompanhados de advérbios, ou substantivos acompanhados de adjetivos. Além disso, os *softgoals* também podem usar substantivos que indicam alguma qualidade como *segurança* ou *performace*. Recomendação semelhante pode ser encontrada em Grau et al. (2008). Outra forma de lidar com a ambigüidade é evitar que ocorra. Para isso, adotando metodologias que desde o início da construção de modelos i^* reduzam a ambigüidade como apresentado em Oliveira et al. (2007).

Nessa dissertação são propostas duas métricas para ambigüidade a métrica Número de Palavras Vagas (A_1) e Taxa de Dependências Ambíguas (A_2) apresentadas a seguir.

4.4.1 Número de Palavras Vagas

Para procurar indícios de problemas com ambigüidade, palavras que podem causar ambigüidade são procuradas nos rótulos dos elementos através da métrica Número de Palavras Vagas (NPV, vide tabela 12). Elas são palavras que adicionam falta de precisão e generalidade aos nomes dos elementos. Entre as classes de palavras problemáticas temos: operadores lógicos como “e”, “ou”, “não”; pronomes como “eles”, “deles”; e quantificadores como “cada” “todo” ou “qualquer”. Essa métrica é usada para obter indícios que os rótulos dos elementos i^* foram mal formados.

A tabela 12 apresenta a métrica Número de Palavras Vagas (A_1). Essa métrica corresponde ao número de palavras pertencentes as classes de palavras problemáticas que estão presentes no rótulos dos elementos intencionais de elementos i^* . Ela pode servir como parâmetro para determinar quando o modelo dever ser revisado para corrigir os possíveis problemas. O valor ideal da métrica é zero, quando não existem palavras que possam

causar problemas com ambigüidade. O avaliador, no entanto pode determinar outro valor de acordo com a tolerância ao uso de palavras como essa.

Tabela 12: Informações sobre a métrica Número de Palavras Vagas

Item	Descrição
Nome	Número de Palavras Vagas (A_1)
Impacto	Pode servir como parâmetro para determinar quando o modelo i^* que está sendo avaliado deve ser revisado
Valor Alvo	O valor ideal para a métrica é 0, que indica que não há palavras que podem gerar problemas interpretação.
Fatores de Qualidade	A métrica está relacionada à Ambigüidade dos modelos i^* .
Ferramentas	Uma planilha pode ser utilizada para armazenar os valores das métricas
Dados	Número de Palavras Vagas (NPV)
Cálculos	Corresponde ao valor de NPV
Interpretação	Valores maiores que 0 indicam presença palavras podem gerar ambigüidade
Treinamento Requerido	Conhecimento na técnica i^*

A tabela 13 apresenta o dado Número de Palavras Vagas (NPV). Esse dado pode é coletado através da contagem das palavras que podem gerar problemas de interpretação como os operadores lógicos (e, ou, não), quantificadores (cada, todos, qualquer) e pronomes (eles, dele, seu/sua). Para cada palavra encontrada é incrementado o valor do dado que é um número inteiro maior ou igual a zero ($NPV \geq 0$).

Tabela 13: Informações sobre o dado Número de Palavras Vagas

Item	Descrição
Nome	Palavras Vagas
Métrica	Número de Palavras Vagas (NPV)
Definição	Corresponde ao número palavras vagas encontradas nos elementos intencionais (Tarefas, Recursos, Metas e <i>Softgoals</i>)
Fonte	Diagramas SR dos modelos i^* .
Coletor	Avaliador do modelo
Procedimentos	Contar o número palavras que podem gerar problemas de interpretação como operadores lógicos (e, ou, não), quantificadores (cada, todos, qualquer) ou pronomes (eles, dele, seu/sua)
Armazenagem	Planilha eletrônica ou tabela com relação das métricas
Representação	Valor numérico inteiro maior ou igual a zero.

4.4.2 Taxa de Dependências Ambíguas

Outra fonte de ambigüidade que pode estar presente em modelos i^* é o uso de construções sintáticas do próprio i^* que permitem múltiplas interpretações. O caso considerado nessa dissertação é o de dependências com múltiplas ligações entre atores e um mesmo *dependum*. Isso pode ocorrer devido a adição de mais de um ator como *dependee* na mesma relação de dependência ou de mais de um ator como *dependor* na relação de dependência. Na figura 11 é apresentado um exemplo onde um *dependum* dividido entre dois atores que serão responsáveis por satisfazer a dependência. A ambigüidade nesse caso está no fato de não haver explicação de como os atores vão satisfazer a dependência, se é necessário que os dois o façam ou se apenas um é o suficiente. Na figura 12 está o caso oposto onde dois atores *dependor* compartilham um mesmo *dependum*. Nesse caso não se sabe qual dos dois teve a intenção no relacionamento da dependência.

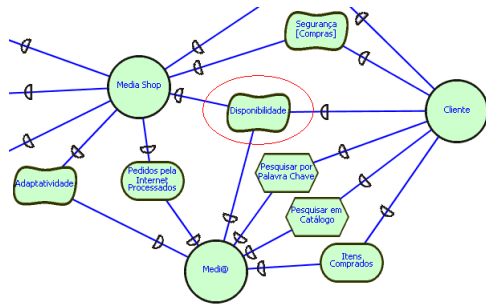


Figura 11: Separação de uma dependência entre dois atores.

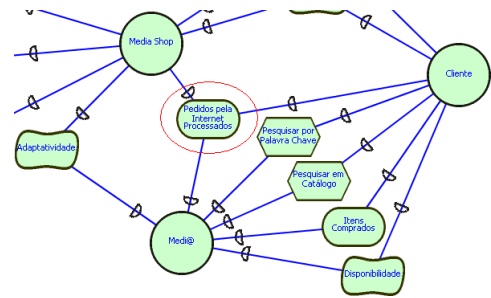


Figura 12: Junção em uma dependência entre dois atores.

As duas situações apresentadas em Grau et al. (2008) são referentes ao caso de um *dependum* ser compartilhado por vários atores *dependor*, ou por vários atores *dependee*.

Para representar a informação desse tipo de ambigüidade é usada a métrica Taxa de Dependências Ambíguas (A_2). A tabela 14 apresenta a métrica A_2 , que representa o percentual de dependências que estão sendo compartilhadas por vários atores. Ela pode servir como parâmetro para determinar quando o modelo i^* deve ser revisado. O valor da métrica varia entre 0, quando não houverem situações de dependências compartilhadas, e 1 quando todas as dependências forem compartilhadas.

Tabela 14: Informações sobre a métrica Taxa de Dependências Ambíguas

Item	Descrição
Nome	Taxa de Dependências Ambíguas (A_2)
Impacto	Pode servir como parâmetro para determinar quando o modelo i^* que está sendo avaliado deve ser revisado.
Valor Alvo	O valor ideal para a métrica é 0, que indica que não há dependências que possam estar gerando dupla interpretação.
Fatores de Qualidade	A métrica está relacionada à Ambigüidade dos modelos i^* .
Ferramentas	Uma planilha pode ser utilizada para armazenar os valores das métricas
Dados	Número de Dependências Relacionadas a mais de um Depende (NDRD1) Número de Dependências Relacionadas a mais de um Depend (NDRD2) Número de Dependências (ND)
Cálculos	$\frac{NDRD1 + NDRD2}{ND} \quad (4.4)$
Interpretação	Valores maiores que 0 indicam que existe dependências que podem estar gerando ambigüidade
Treinamento Requerido	Conhecimento na técnica i^*

Para obter o valor da métrica A_2 é necessário coletar o Número de Dependências que existem no modelo (ND, vide tabela 17), o Número de Dependências que estão Relacionadas a mais de um ator Depende (NDRD1, vide tabela 15) e o Número de Dependências Relacionadas a mais de um ator Depend (NDRD2, vide tabela 16).

A tabela 15 apresenta o dado Número de Dependências Relacionadas a mais de um *Depende*. Esse dado conta o número de dependências que tem mais de um ator do lado *dependee*. O número considera a quantidade de *dependums* envolvidos, não importa a quantidade de vezes que uma dependência é bifurcada o número só será incrementado em um. O valor resultante será um número inteiro maior ou igual a zero ($NDRD1 \geq 0$).

Tabela 15: Informações sobre dado Número de Dependências Relacionadas a mais de um Depende

Item	Descrição
Nome	Número de Dependências Relacionadas a mais de um Depende (NDRD1)
Métrica	Taxa de Dependências Ambíguas (A_2)
Definição	Corresponde ao número de dependências que do lado Depende possui mais de uma ligação.
Fonte	Diagramas SD dos modelos i^* .
Coletor	Avaliador do modelo
Procedimentos	Contar no modelo as dependências bifurcadas no lado do Depende. Considerar o número de dependums envolvidos.
Armazenagem	Planilha eletrônica ou tabela com relação das métricas
Representação	Valor numérico inteiro maior ou igual a zero.

A tabela 16 apresenta o dado Número de Dependências Relacionadas a mais de um *Depender*. Esse dado conta o número de dependências que tem mais de um ator do lado *dependor*. O número considera a quantidade de *dependums* envolvidos, não importa a quantidade de vezes que uma dependência é bifurcada o número só será incrementado em um. O valor resultante será um número inteiro maior ou igual a zero ($NDRD2 \geq 0$).

Tabela 16: Informações sobre dado Número de Dependências Relacionadas a mais de um *Depender*

Item	Descrição
Nome	Número de Dependências Relacionadas a mais de um <i>Depender</i> (NDRD2)
Métrica	Taxa de Dependências Ambíguas (A_2)
Definição	Corresponde ao número de dependências que do lado <i>Depender</i> possui mais de uma ligação.
Fonte	Diagramas SD dos modelos i^* .
Coletor	Avaliador do modelo
Procedimentos	Contar no modelo as dependências bifurcadas no lado do <i>Depender</i> . Considerar o número de <i>dependums</i> envolvidos.
Armazenagem	Planilha eletrônica ou tabela com relação das métricas
Representação	Valor numérico inteiro maior ou igual a zero.

Para obter a métrica A_2 também é necessário contar o Número de Dependências nos modelos (ND). A tabela 17 apresenta o dado Número de Dependências, que consiste no número total de dependências presentes no modelo. Da mesma forma que os dados anteriores, a contagem dos dados deve considerar o número de *dependums* no modelo. Se por algum motivo as dependências forem bifurcadas apenas o número de *dependums* de cada dependência serão considerados. O valor obtido é um valor maior ou igual a zero ($ND \geq 0$).

Tabela 17: Informações sobre o dado Número de Dependências

Item	Descrição
Nome	Número de Dependências (ND)
Métrica	Taxa de Dependências Ambíguas (A_2)
Definição	Corresponde ao número de dependências no modelo.
Fonte	Diagramas SD e SR dos modelos i^* .
Coletor	Avaliador do modelo
Procedimentos	Contar todas as dependências. Considerar o número de <i>dependums</i>
Armazenagem	Planilha eletrônica ou tabela com relação das métricas
Representação	Valor numérico inteiro maior ou igual a zero.

4.5 Métricas para Complexidade

Um problema já relatado com modelos i^* é o crescimento explosivo dos modelos (GRAU et al., 2008). Algumas abordagens têm sido desenvolvidas para lidar com esse crescimento como o uso de aspectos (ALENCAR et al., 2006) ou o uso de visões (YOU, 2004). No entanto, antes de usar qualquer abordagem é necessário conhecer os modelos para identificar se está muito complexo. A figura 13 apresenta um exemplo de como o crescimento no número de elementos pode tornar um diagrama complexo.

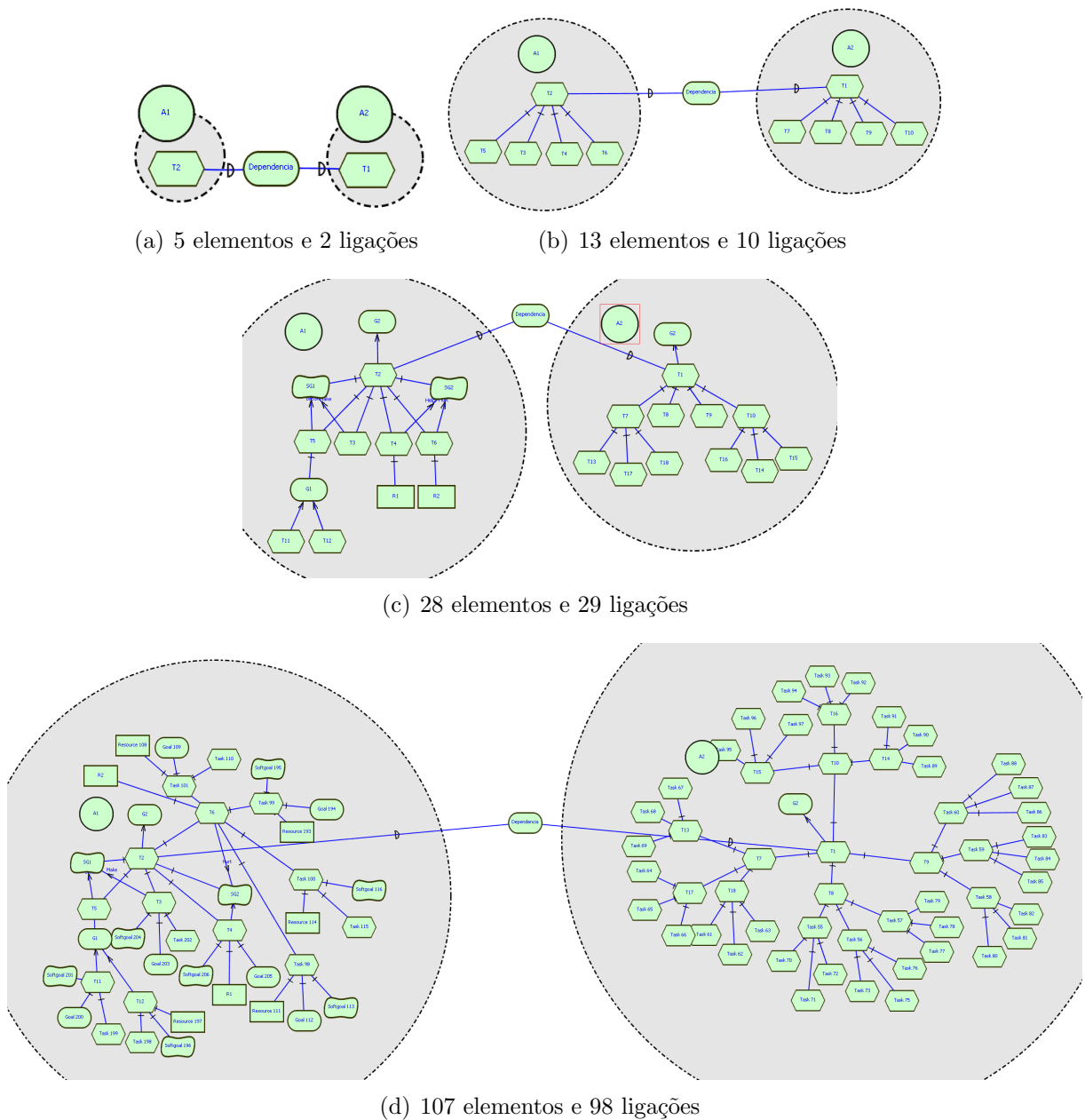


Figura 13: Exemplo do crescimento explosivo de um modelo i^*

Como a complexidade é uma propriedade estudada em outros paradigmas já existem muitas métricas para medir a complexidade de modelos. Para não tornar esse trabalho repetitivo ao criar métricas semelhantes as que já existem, adotou-se a opção de reusar as métricas existentes para avaliar modelos i^* . Isso foi feito através do mapeamento de métricas de complexidade sempre que a abstração usada se adeque ao i^* .

Algumas abordagens já utilizadas em outros paradigmas para medir a complexidade foram adaptadas para medir modelos i^* . Entre elas foram consideradas a complexidade ciclomática de McCabe (MCCABE, 1976); as medidas de complexidade de Halstead (HALSTEAD, 1977); e uma medida de complexidade baseada em Entropia (KIM; SHIN; WU, 1995). Para cada uma dessas métricas será usado um conjunto de regras de mapeamento descrita a seguir. Nessa dissertação serão apresentadas as métricas Complexidade Ciclomática (C_1), Vocabulário (C_2), Comprimento (C_3), Volume (C_4), Dificuldade (C_5), Esforço (C_6) e Entropia (C_7).

4.5.1 Complexidade Ciclométrica

O i^* é por natureza um grafo com vértices e arestas (YU, 1995), onde os vértices são representados por atores (i.e., Ator, Papel, Agente e Posição) ou elementos intencionais (i.e., Metas, *Softgoals*, Recursos e Tarefas); e as arestas são representados pelas ligações que conectam os atores (e.g., ligação Is-A, Is-part-of, etc.) e ligações que conectam os elementos intencionais (dependência, *Means-Ends*, Decomposição e Contribuição).

A Complexidade Ciclométrica de McCabe (1976) oferece uma medida de complexidade baseada na dimensão de um grafo dirigido, para isso são contados o número de vértices, arestas e componentes conexos no grafo. Por sua natureza de grafo (YU, 1995), o mapeamento dessa métrica para o i^* é natural. Para isso basta abstrair os tipos de elementos e ligações usados nos diagramas e considerá-los como vértices e arestas, respectivamente.

A Complexidade Ciclométrica pode ser interpretada como o número de caminhos independentes ou regiões no grafo, considerando divisões nos grafos onde é feita a tomada de decisão. No caso do código isso é feito em estruturas de controle como laços (e.g., *while*) ou estruturas condicionais (e.g., *if-then-else*). As estruturas do i^* como a decomposição e o *means-ends* expressam alternativas de forma semelhante às estruturas condicionais, além disso, ciclos podem se formar em dependências ou através de contribuições (HORKOFF, 2006). Portanto, o mapeamento de diagramas i^* para um grafo dirigido é adequado por preservar as características dos grafos que são usados pela complexidade ciclométrica. Esse mapeamento usa como base a própria definição do i^* , onde os modelos são considerados

grafos dirigidos com nós e ligações (YU, 1995).

1. Os elementos intencionais (i.e., Tarefas, Recursos, Metas e *Softgoals*) são mapeados para vértices;
2. Os atores (i.e., Ator, Agente, Posição e Papel) são mapeados para vértices;
3. As ligações entre atores (i.e., *Is-A*, *Is-part-of*, *INS*, *Plays*, *Covers* e *Occupies*), são mapeadas para arestas;
4. As ligações entre elementos intencionais (i.e., Contribuição, Decomposição, *Means-Ends* e Dependência), são mapeadas para arestas;
5. As ligações entre atores e elementos intencionais (i.e., Dependência) são mapeadas para arestas;
6. O número de componentes conexos é obtido através da contagem do número de sub-grafos independentes.

Um detalhe importante a ser considerado na métrica da complexidade ciclomática é a presença das fronteiras abertas dos atores. Quando os atores tem suas fronteiras abertas, as dependências que existiam com os atores passam a ser feitas com elementos internos. Em muitos casos os atores ficam sem conexões levando a contar o ator como um componente conexo no modelo. Para evitar distorções na contagem dos componentes conexos, sempre que um ator com a fronteira aberta estiver sem conexões no modelo SR e seus elementos internos estiverem conectados, ele não será contado como um componente conexo. Assim as ligações que chegam aos elementos internos de um ator, mesmo que não estejam ligadas ao próprio ator, serão consideradas como ligações para o ator.

Com base nos mapeamentos a equação 3.1 é reescrita como:

$$CC = NL - NE + 2 \times NCC \quad (4.5)$$

, onde NL (Número de ligações) corresponde ao E da equação 3.1 (Número de vértices). NE (Número de Elementos) corresponde ao L da equação 3.1 (Número de arestas) e o C da equação 3.1 (componentes conexos) corresponde ao NCC (Número de Componentes Conexos)

A tabela 18 apresenta a métrica Complexidade Ciclomática adaptada para i^* . A métrica pode servir como parâmetro para compara modelos ou determinar quando um

modelo i^* que está complexo deve ser revisado. O valor alvo definido pela métrica é qualquer valor menor que dez $CC < 10$, esse valor é comumente usado para designar um valor de baixa complexidade (MCCABE, 1976). Os dados necessários para obter o valor de CC são: o Número de Elementos (NE), o Número de Ligações (NL) e o Número de Componentes Conexos (NCC).

Tabela 18: Informações sobre a métrica Complexidade Ciclomática

Item	Descrição
Nome	Complexidade Ciclomática (C_1)
Impacto	Pode servir como parâmetro para comparar modelos ou determinar quando o modelo i^* deve ser revisado.
Valor Alvo	$CC \leq 10$ indica que o diagrama é de baixa complexidade
Fatores de Qualidade	Complexidade
Ferramentas	Uma planilha pode ser utilizada para armazenar os valores das métricas
Dados	Número de Ligações (NL) Número de Elementos (NE) Número de Componentes Conexos (NCC)
Cálculos	$CC = NL - NE + 2 \times NCC \quad (4.6)$
Interpretação	Representa uma estimativa para o número de caminhos linearmente independentes no grafo. Quanto mais alto o valor mais complexo é o modelo avaliado, $CC \leq 10$ indica que o diagrama é de baixa complexidade para $15 \geq CC > 10$ indica que o diagrama tem complexidade intermediária, valores maiores que 20 indica que o diagrama é muito complexo
Considerações	A métrica é apropriada para avaliar modelos i^*
Treinamento Requerido	Não é requerido treinamento específico

A tabela 19 apresenta o dado Número de Ligações. Esse dado é utilizado na métrica Complexidade Ciclomática e corresponde ao número total de ligações usadas para conectar os elementos nos diagramas i^* . As ligações contadas nesse dado podem ser ligações entre atores (i.e., *Is-A*, *Is-part-of*, *Covers*, *Plays*, *Occupies*, *INS*); entre atores e elementos intencionais (i.e., dependências); ou entre elementos intencionais (i.e., decomposição, contribuição e *Means-ends*). Para cada ligação encontrada o dado é incrementado em um. Ele corresponde a um número inteiro maior ou igual a zero. Esse dado pode ser coletado em diagramas SD e SR.

Tabela 19: Informações sobre o dado Número de Ligações

Item	Descrição
Nome	Número de Ligações nos Diagramas (NL)
Métricas	Complexidade Ciclomática (C_1)
Definição	Corresponde às ligações entre elementos dos i^*
Fonte	Diagramas SD e SR
Coletor	Avaliador do modelo
Procedimentos	Contar ligações entre elementos nos diagramas i^* . Considerar todas as ligações independente dos tipos. As ligações podem ser entre atores (i.e., <i>Is-A</i> , <i>Is-part-of</i> , <i>Covers</i> , <i>Plays</i> , <i>Occupies</i> , <i>INS</i>); entre atores e elementos intencionais (i.e., dependências); ou entre elementos intencionais (i.e., decomposição, contribuição e <i>Means-ends</i>).
Armazenagem	Planilha eletrônica ou tabela com relação das métricas
Representação	Valor numérico inteiro maior ou igual a zero.

A tabela 20 apresenta o dado Número de Elementos. Esse dado é utilizado na métrica Complexidade Ciclomática e corresponde ao número total de elementos presentes nos diagramas i^* . Os elementos contados são atores (i.e., Ator, Posição, Papel e Posição) e elementos intencionais (i.e., Tarefa, Recurso, Meta e *Softgoal*). Para cada elemento encontrado o dado é incrementado em um. Ele corresponde a um número inteiro maior ou igual a zero. Esse dado pode ser coletado em diagramas SD e SR.

Tabela 20: Informações sobre o dado Número de Elementos

Item	Descrição
Nome	Número de Elementos no Diagramas (NE)
Métricas	Complexidade Ciclomática (C_1)
Definição	Corresponde ao número total de elementos nos diagramas i^*
Fonte	Diagramas SD e SR
Coletor	Avaliador do modelo
Procedimentos	Contar elementos nos diagramas i^* , os elementos considerados são atores (i.e., Ator, Posição, Papel e Posição) e elementos intencionais (i.e., Tarefa, Recurso, Meta e <i>Softgoal</i>).
Armazenagem	Planilha eletrônica ou tabela com relação das métricas
Representação	Valor numérico inteiro maior ou igual a zero.

A tabela 21 apresenta o dado Número de Componentes Conexos (NCC), que é usado no cálculo da métrica Complexidade Ciclométrica. Para obter esse dado são contados os sub-grafos que não estão conectados entre si no diagrama SD ou SR. Caso o diagrama não apresente sub-grafos independentes, quando todos os elementos estão ligados direta ou indiretamente, o valor do dado será um. Ele é um valor inteiro maior ou igual a 1 e pode ser coletado em diagramas SD e SR.

Tabela 21: Informações sobre o dado Número de Componentes Conexos

Item	Descrição
Nome	Número de Componente Conexos (NCC)
Métricas	Complexidade Ciclométrica (C_1) Comprimento (C_3)
Definição	Corresponde ao número de total sub-grafos independentes
Fonte	Diagramas SD e SR
Coletor	Avaliador do modelo
Procedimentos	Contar o número de sub-grafos independentes, isto é, sub-grafos pertencentes ao grafo principal que não estão ligados entre si.
Armazenagem	Planilha eletrônica ou tabela com relação das métricas
Representação	Valor numérico inteiro maior ou igual a um.

4.5.2 Métricas de Halstead

O i^* é uma técnica de modelagem onde são permitidas operações sobre os elementos presentes na linguagem. Nela os atores podem ser estruturados através dos relacionamentos entre atores como *Is-A*, *Is-part-of*, etc. Operações também podem ser feitas sobre elementos intencionais como a decomposição de tarefas, a contribuição para *softgoals*, ou as alternativas para as metas, através da ligação *Means-ends*. Como o i^* possui uma linguagem onde as operações são bem definidas e se diferenciam claramente dos operandos, considerar as ligações como operações e os elementos como operandos é uma abstração válida. Assim as métricas de Halstead (1977) podem ser mapeadas para i^* . Originalmente definidas para avaliar complexidade de código-fonte através da medição do tamanho e vocabulário usados na descrição dos programas.

Os elementos no i^* são mapeados para operandos. Os elementos são os atores (i.e., Ator, Papel, Posição e Agente) e os elementos intencionais (i.e., Recurso, Tarefa, Meta, e *Softgoal*) e serão considerados como operandos por serem usados em operações através das ligações nos modelos i^* .

As ligações entre elementos são mapeadas para operadores. A de estruturação de atores (e.g., *Is-a*, *Is-part-of*), dependência e operações sobre elementos internos (i.e., decomposição, contribuição e *Means-Ends*) são mapeadas para operadores por serem os mecanismos usados para realizar operações nos elementos dos modelos.

Além do mapeamento para operadores e operandos é necessário definir quando eles são distintos. Será considerada distinta apenas a primeira ocorrência de cada elemento ou ligação. Caso se repitam no modelo as repetições não serão contadas. Para o caso das ligações são contados os tipos de ligações diferentes que aparecem sem considerar possíveis repetições. Por exemplo, quando uma ligação de dependência aparece é considerada apenas uma vez mesmo que se repita em vários locais. Os elementos são considerados distintos se ocorrem em um tipo (e.g., Tarefa, Recurso, etc.) e um rótulo específico, assim elementos com mesmo tipo e rótulos diferentes são considerados distintos. Da mesma forma, elementos com rótulos iguais e tipos diferentes também são considerados distintos. No entanto, elementos com mesmo tipo e mesmo rótulo só serão contados uma vez.

4.5.2.1 Vocabulário

O Vocabulário é o subconjunto mínimo da linguagem usado para descrever o modelo. Ele conta quantos elementos e ligações diferentes são necessários para construir o modelo,

para isso são contados apenas as ligações e elementos distintos, sem considerar possíveis repetições. O tamanho do vocabulário é diretamente proporcional à complexidade do modelo, quanto maior o tamanho do vocabulário mais complexo será o modelo.

A tabela 22 apresenta a métrica Vocabulário (C_2). Essa métrica conta os operadores e operandos distintos usados para descrever um modelo. Ela pode servir como parâmetro para decidir quando um modelo deve ser revisado. A métrica não apresenta um valor alvo pré-definido, mas quanto maior for o tamanho do vocabulário maior será a complexidade do modelo. Para calcular o valor da métrica são necessários os dados Número de Ligações Distintas (NLD) e Número de Elementos Distintos (NED).

Tabela 22: Informações sobre a métrica Vocabulário para modelos i^*

Item	Descrição
Nome	Vocabulário (C_2)
Impacto	Pode servir como parâmetro para comparar modelos ou determinar quando o modelo i^* deve ser revisado
Fatores de Qualidade	Complexidade
Ferramentas	Uma planilha pode ser utilizada para armazenar os valores das métricas
Aplicação	Avaliar Complexidade nos modelos
Dados	Número de Ligações Distintas (NLD) Número de Elementos Distintos(NED)
Cálculos	$C_2 = NLD + NED \quad (4.7)$
Interpretação	Quando maior o tamanho do vocabulário mais complexo será o modelo

A tabela 23 apresenta o dado Número de Ligações Distintas (NLD). Ele é usado para o cálculo das métricas Vocabulário (C_2) e Dificuldade (C_5). Para a contagem são considerados tanto modelos SD como SR. O procedimento é feito através da contagem dos tipos de ligações existentes sem considerar as repetições. Quando um conector estiver presente no modelo o valor de NLD é incrementado, isso acontece apenas uma vez independente de quantas vezes a ligação é usada no modelo. No caso das ligações de Contribuição onde existem diferentes tipos (e.g., *Break*, *Hurt*, *Help*, etc.) cada um deles deve ser contado apenas uma vez caso esteja presente no modelo. O valor de NLD é um inteiro maior que zero.

Tabela 23: Informações sobre o dado Número de Ligações Distintas

Item	Descrição
Nome	Número de Ligações nos Diagramas (NLD)
Métricas	Vocabulário (C_2) Dificuldade (C_5)
Definição	Corresponde às ligações distintas entre elementos dos Diagramas
Fonte	Diagramas SD e SR
Coletor	Avaliador do modelo
Procedimentos	Contar os tipos de ligações usadas nos modelos. Incrementar o valor de NLD sempre que um tipo de ligação for encontrado. Repetições não devem ser consideradas
Armazenagem	Planilha eletrônica ou tabela com relação das métricas
Representação	Valor numérico inteiro maior ou igual a zero.

A tabela 24 apresenta o dado Número de Elementos Distintos (NED). Ele é usado para o cálculo das métricas Vocabulário (C_2) e Dificuldade (C_5). O valor de NED corresponde ao número de elementos no modelo que são únicos. Os elementos considerados podem ser atores (i.e., Ator, Papel, Posição e Agente) ou elemento Intencional (i.e., Tarefa, Recurso, Meta e *Softgoal*). Sempre que tipos e rótulos não forem iguais eles devem ser contados, no caso de um elemento apresentar um repetição ele deve ser contado apenas uma vez.

Tabela 24: Informações sobre o dado Número de Elementos Distintos

Item	Descrição
Nome	Número de Elementos Distintos (NED)
Métricas	Vocabulário (C_2) Dificuldade (C_5)
Definição	Corresponde ao número de elementos distintos
Fonte	Diagramas SD e SR
Coletor	Avaliador do modelo
Procedimentos	Contar elementos sem considerar as repetições de elementos com os mesmo nomes e do mesmo tipo. Incrementar o valor de NED sempre que um tipo de elemento for encontrado.
Armazenagem	Planilha eletrônica ou tabela com relação das métricas
Representação	Valor numérico inteiro maior ou igual a zero.
Treinamento Re-querido	Não é requerido treinamento específico

4.5.2.2 Comprimento

O Comprimento consiste em uma estimativa do tamanho total do modelo. Ela é obtida contando tanto ligações como elementos. Ao contrário da métrica Vocabulário onde as repetições não são consideradas na métrica Comprimento todas as ocorrências de ligações e elementos devem ser consideradas.

A tabela 25 apresenta a métrica Comprimento (C_3). A métrica C_3 é calculada através da soma dos dados Número de Ligações (NL) e Número de Elementos (NE). Ela pode servir como parâmetro para comparar modelos ou determinar quando o modelo i^* deve ser revisado. O valor obtido é diretamente proporcional a complexidade, quanto maior o valor da métrica maior será a complexidade.

Tabela 25: Informações sobre a métrica Comprimento para modelos i^*

Item	Descrição
Nome	Comprimento (C_3)
Impacto	Pode servir como parâmetro para comparar modelos ou determinar quando o modelo i^* deve ser revisado
Fatores de Qualidade	Complexidade
Ferramentas	Uma planilha pode ser utilizada para armazenar os valores das métricas
Aplicação	Avaliar complexidade
Dados	Número de Ligações (NL) Número de Elementos (NE)
Cálculos	$C_3 = NL + NE \quad (4.8)$
Interpretação	Representa o número total de elementos e ligações no modelo, quanto maior o valor mais complexo é o modelo.
Treinamento Requerido	Não é requerido treinamento específico

4.5.2.3 Volume

O Volume (C_4) é uma métrica derivada das métricas Vocabulário (C_2) e Comprimento (C_3). Ela é uma estimativa do volume de informação no modelo que considera tanto o total de elementos e ligações quanto o tamanho do vocabulário que foi usado para descrever o modelo.

A tabela 26 apresenta a métrica Volume (C_4), ela é baseada nas métricas Vocabulário e Comprimento. Ela está relacionada com fatores de qualidade que levam em consideração a compreensão dos modelos como Facilidade de Entendimento e Manutenibilidade. Para obter essa métrica não é necessário treinamento específico em i^* .

Tabela 26: Informações sobre a métrica Volume para modelos i^*

Item	Descrição
Nome	Volume (C_4)
Impacto	Pode servir como parâmetro para comparar modelos ou determinar quando o modelo i^* deve ser revisado
Fatores de Qualidade	Complexidade e fatores relacionados ao entendimento como Facilidade de Entendimento, Manutenibilidade
Ferramentas	Uma planilha pode ser utilizada para armazenar os valores das métricas
Aplicação	Avaliar Complexidade dos modelos
Dados	Comprimento (C_3) Vocabulário (C_2)
Cálculos	$C_4 = C_3 \times \log_2(C_2) \quad (4.9)$
Interpretação	Estimativa de volume de informação do modelo, quanto maior o valor mais complexo é o modelo
Treinamento Requerido	Não é requerido treinamento específico

4.5.2.4 Dificuldade

A métrica Dificuldade (C_5) é uma estimativa da dificuldade de manutenção ou entendimento (MISEK-FALKOFF, 1982). Ela se baseia na quantidade de informação que é necessária para entender devido ao número de operadores ($\frac{NLD}{2}$, primeiro termo da equação 4.10) e ao número médio de operandos que é usado ($\frac{NE}{NED}$, segundo termo da equação 4.10). Essa estimativa é diretamente proporcional à complexidade do modelo, quanto maior o valor da métrica maior é a complexidade.

A tabela 27 apresenta as informações sobre a métrica Dificuldade. Ela pode servir de parâmetro para comparar modelos i^* ou para determinar se deve ser revisado. Está relacionada a fatores de qualidade como a facilidade de Entendimento e Manutenabilidade. Ela é obtida a partir dos dados Número de Ligações Distintas (NLD), Número de Elementos (NE) e Número de Elementos Distintos (NED).

Tabela 27: Informações sobre a métrica Dificuldade para modelos i^*

Item	Descrição
Nome	Dificuldade (C_5)
Impacto	Pode servir como parâmetro para comparar modelos ou determinar quando o modelo i^* deve ser revisado
Fatores de Qualidade	Complexidade e fatores relacionados ao entendimento como Facilidade de Entendimento, Manutenabilidade
Ferramentas	Uma planilha pode ser utilizada para armazenar os valores das métricas
Aplicação	Avaliar Facilidade de entendimento e escalabilidade
Dados	Número de Ligações Distintas (NLD) Número de Elementos (NE) Número de Elementos Distintos (NED)
Cálculos	$C_5 = \frac{NLD}{2} \times \frac{NE}{NED} \quad (4.10)$
Interpretação	Estimativa para o grau de legibilidade do modelo, quanto maior o valor mais complexo é o modelo
Treinamento Requerido	Não é requerido treinamento específico

4.5.2.5 Esforço

A métrica Esforço (C_6) está relacionada com o esforço para manter ou entender um modelo. Ela se baseia nas métricas Volume (C_4) e Dificuldade (C_5). O valor dela é diretamente proporcional ao Volume ou a Dificuldade, pois quanto maior o Volume mais Esforço será necessário para manter o modelo, da mesma forma, quanto maior a Dificuldade maior o Esforço para manter um modelo.

A tabela 28 apresenta as informações da métrica Esforço. Ela está relacionada a fatores de qualidade como Facilidade de Entendimento e Manutenabilidade. Essa métrica é derivada dos valores das métricas Volume e Dificuldade. Não necessita um treinamento específico para ser obtida. Além disso, ela é diretamente proporcional à complexidade do modelo, quanto maior for o valor da métrica maior será a complexidade do mesmo.

Tabela 28: Informações sobre a métrica Esforço para modelos i^*

Item	Descrição
Nome	Esforço (C_6)
Impacto	Pode servir como parâmetro para comparar modelos ou determinar quando o modelo i^* deve ser revisado
Fatores de Qualidade	Complexidade e fatores relacionados ao entendimento como Facilidade de Entendimento, Manutenabilidade
Ferramentas	Uma planilha pode ser utilizada para armazenar os valores das métricas
Aplicação	Avaliar Complexidade
Dados	Dificuldade (C_5) Volume (C_4)
Cálculos	$C_6 = C_4 \times C_5 \quad (4.11)$
Interpretação	Estimativa de esforço, quanto maior o valor mais complexo é o modelo avaliado
Considerações	A métrica é apropriada para avaliar modelos i^*
Treinamento Requerido	Não é requerido treinamento específico

4.5.3 Entropia

Além das métricas já apresentadas uma outra medida para complexidade poderia ser a Entropia (KIM; SHIN; WU, 1995), que é uma medida quantitativa sobre incerteza em um conjunto de dados. Em teoria da informação a incerteza na informação é geralmente calculada pela entropia da informação, freqüentemente chamada Entropia de Shannon (SHANNON, 1948).

Para obter uma medida de Entropia foi adotado um mapeamento da métrica apresentada em Kim, Shin e Wu (1995). Essa métrica é uma medida para a forma como a informação está distribuída de acordo com as referências feitas à elementos da linguagem. Para a contagem os objetos que serão contados (e.g., código fonte ou modelos) são transformados em grafos dirigidos e a partir deles são obtidas as medidas necessárias para o cálculo da Entropia. O principal dado necessário para o cálculo da Entropia é a probabilidade referencial que conta a proporção de vezes que o objeto medido faz referências ou é referenciado por outro. O número de referências no caso de código-fonte é o número de vezes que uma função, dado ou operador é usado, no caso de diagramas de módulos é o número de vezes que um módulo é associada a outros módulos. Se o número de referências for equilibrado, quando os elementos têm aproximadamente o mesmo número de referências, o valor da entropia será baixo, por outro lado, se um pequeno número de objetos concentrar muitas referências o valor da Entropia será alto.

Como apresentado na seção 4.5.1, o i^* pode ser visto como um grafo dirigido onde as ligações (e.g., dependências, *Means-Ends*, Contribuição, etc.) podem ser vistas como arestas e os elementos (e.g., Atores, Tarefas, Recursos, etc.) podem ser vistos como vértices. Usando essa abstração de grafo dirigido a medida de Entropia pode ser obtida para modelos i^* associando cada elemento à um valor de probabilidade referencial. A probabilidade referencial para modelos i^* será a proporção entre o número de ligações que entram e saem de cada elemento e o número total de ligações nos modelos.

O mapeamento desta métrica para i^* é feito segundo as seguintes regras de mapeamento:

1. Os elementos do i^* como atores (i.e., Ator, Papel, Posição, Agente) ou elementos intencionais (i.e., Recursos, Tarefas, Metas, *Softgoals*) são tratados como vértices de um grafo dirigido;
2. As ligação do i^* como dependências, *Means-Ends*, contribuições e decomposição são tratadas são tratadas como arestas de um grafo dirigido;

3. Cada aresta que entra ou sai de um vértice será considerada uma referência.

A tabela 29 apresenta as informações da métrica Entropia. A métrica está relacionada à complexidade dos modelos i^* , ela é obtida através da computação da probabilidade referencial ($p(x_i)$) de todos os elementos do modelo. Ela não precisa de treinamento específico para ser coletada. O valor obtido pela métrica é diretamente proporcional à complexidade do modelo, quanto maior a Entropia mais complexo será o modelo.

Tabela 29: Entropia para Modelos i^*

Item	Descrição
Nome	Entropia (C_7)
Impacto	Serve de parâmetro para comparar modelo i^*
Fatores de Qualidade	Complexidade
Ferramentas	Uma planilha pode ser utilizada para armazenar os valores das métricas
Aplicação	Avaliar Facilidade de entendimento e escalabilidade
Dados	Probabilidade Referencial ($p(x_i)$)
Cálculos	$C_7 = - \sum_{i=1}^{NE} p(x_i) \times \log_2 p(x_i) \quad (4.12)$
Interpretação	Um valor para a organização da informação no modelo, quanto maior mais desorganizada está a informação e mais complexo será o modelo.
Treinamento Requerido	Não é requerido treinamento específico

A tabela 30 apresenta o dado Probabilidade Referencial. Esse dado é obtido a partir da contagem da proporção de ligações que entram e saem dos elementos i^* . Para calcular a Probabilidade é necessário focar em cada elemento presente no modelo e contar o número de ligações que entram e saem de cada elemento, esse valor é então dividido por duas vezes o número de ligações ($2 \times NL$). O resultado será um valor real entre um e zero.

Tabela 30: Informações sobre o dado Probabilidade Referencial

Item	Descrição
Nome	Probabilidade Referencial ($p(x_i)$)
Métricas	Entropia (C_7)
Definição	Corresponde a proporção do número de ligações do elemento em foco em relação ao total.
Fonte	Diagramas SD e SR
Coletor	Avaliador do modelo
Procedimentos	Para o elemento que está em foco (fator x_i da equação) contar o número de referências que são feitas à ele. Cada referência corresponde à ligações (e.g., dependência, contribuição, decomposição, etc.), são contadas tanto as que saem quanto as que chegam aos elementos. O valor obtido pela contagem do número de ligações é dividido por $2 \times NL$ que corresponde ao total de ligações feitas nos dois sentidos. Com isso se obtém a proporção das ligações feitas do elemento em relação ao total.
Armazenagem	Planilha eletrônica ou tabela com relação das métricas
Representação	Valor real entre um e zero.

4.6 Resumo das Métricas

A tabela 31 mostra as métricas apresentadas nesse capítulo de forma resumida. Ela trás o nome das métricas, um resumo da aplicação das mesmas e o código usado para referenciar tabela onde a métrica foi definida.

Tabela 31: Resumo das Métricas

Métricas	Resumo	Referência
Número de Erros (E_1)	Indica a presença de erros sintáticos nos modelos	Tabela 2
Taxa de Metas sem ligação com Rotinas (D_2)	Aponta falta de detalhamento na análise de metas	Tabela 4
Taxa de Dependências sem Ligação a Elementos Internos no Ator (D_2)	Aponta falta de detalhamento nas ligações de dependências	Tabela 6
Taxa de <i>Softgoals</i> que não são tratados por Contribuição (D_3)	Aponta falta de detalhamento nas contribuições para <i>Softgoals</i>	Tabela 9
Número de Palavras Vagas (A_1)	Indica a presença de palavras que podem gerar ambigüidade	Tabela 12
Taxa de Dependências Ambíguas (A_2)	Indica a presença de problemas sintáticos que podem gerar ambigüidade	Tabela 14
Complexidade Ciclomática (C_1)	Estimativa de complexidade baseada na quantidade de ciclo no modelo	Tabela 18
Vocabulário (C_2)	Conjunto mínimo de ligações e elemento usados para descrever o modelo	Tabela 22
Comprimento (C_3)	Número total de ligações e elementos em um modelo	Tabela 25
Volume (C_4)	Estimativa de tamanho baseada no vocabulário e no comprimento	Tabela 26
Dificuldade (C_5)	Estimativa da dificuldade de entender o modelo	Tabela 27
Esforço (C_6)	Estimativa do esforço para manter e entender o modelo baseado na dificuldade e no esforço	Tabela 28
Entropia (C_7)	Medida de complexidade baseada na organização da informação	Tabela 29

4.7 Conclusão

Nesse capítulo foi apresentado um conjunto de 13 métricas para avaliar modelos i^* . As métricas foram descritas usando o *template* de IEEE-1061 (1992). Para cada métrica foram apresentados exemplos gráficos que ilustram as informações capturadas pelas métricas. As métricas estão agrupadas de acordo com os atributos que se deseja medir: Erros típicos, Detalhamento, ambigüidade e complexidade. Esse conjunto de métricas não cobre todos os tipos de atributos de qualidade, mas são um recurso válido para avaliar a qualidade de modelos i^* .

5 Exemplo de Aplicação

Esse capítulo apresenta um exemplo detalhado de como usar o conjunto de métricas propostas em uma avaliação. Os dados são coletados e as métricas calculadas e analisadas em um exemplo modelado com a técnica i^* .

5.1 Introdução

As métricas apresentadas no capítulo 4 serão aplicadas a um exemplo para demonstrar a utilização das mesmas. O exemplo adotado foi o Informa Turma, um projeto de alunos de graduação da disciplina IF716 Especificação de Requisitos e Validação de Sistemas do Centro de Informática da Universidade Federal de Pernambuco. Ele foi escolhido para servir de exemplo por ter sido feito por iniciantes no uso do i^* e por isso mais propensos a introduzir problemas nos modelos i^* . Além disso, os autores do modelo não tiveram contato com as métricas evitando assim a tendência de direcionar a solução deles para obter uma boa avaliação. O exemplo não possui valor estatístico para ser considerado uma validação experimental, o objetivo é demonstrar como obter o valor das métricas com um passo-a-passo.

5.2 Descrição do Exemplo - Informa Turma

O exemplo Informa Turma descreve os requisitos de um sistema para gerenciar festas de formatura a partir da identificação dos principais problemas encontrados nessa atividade. O público alvo dessa aplicação são formandos e comissões de formatura que usariam o sistema para ter um ambiente de comunicação e tomada de decisão conjunta entre os envolvidos. Os atores descritos nos modelos i^* são:

- O ator *Formando* corresponde aos alunos que estão concluindo o curso e pretende participar dos eventos da formatura;
- O ator *Comissão* corresponde aos alunos que fazem parte da comissão de formatura;
- O ator *Empresa Desenvolvedora* corresponde ao envolvidos no desenvolvimento do sistema em si;
- O ator *Informa Turma* é a representação do sistema em si que vai ser utilizado pelos atores *Formando* e *Comissão* e será responsável por gerenciar as atividades da comissão de formatura.

Os principais requisitos descritos no sistema estão relacionadas às atividades da comissão de formatura, entre elas estão a gerência de votação, a participação em discussões, gerência dos dados das turmas e gerência dos dados dos formandos. Eles foram descritos no modelo SR da Figura 14.

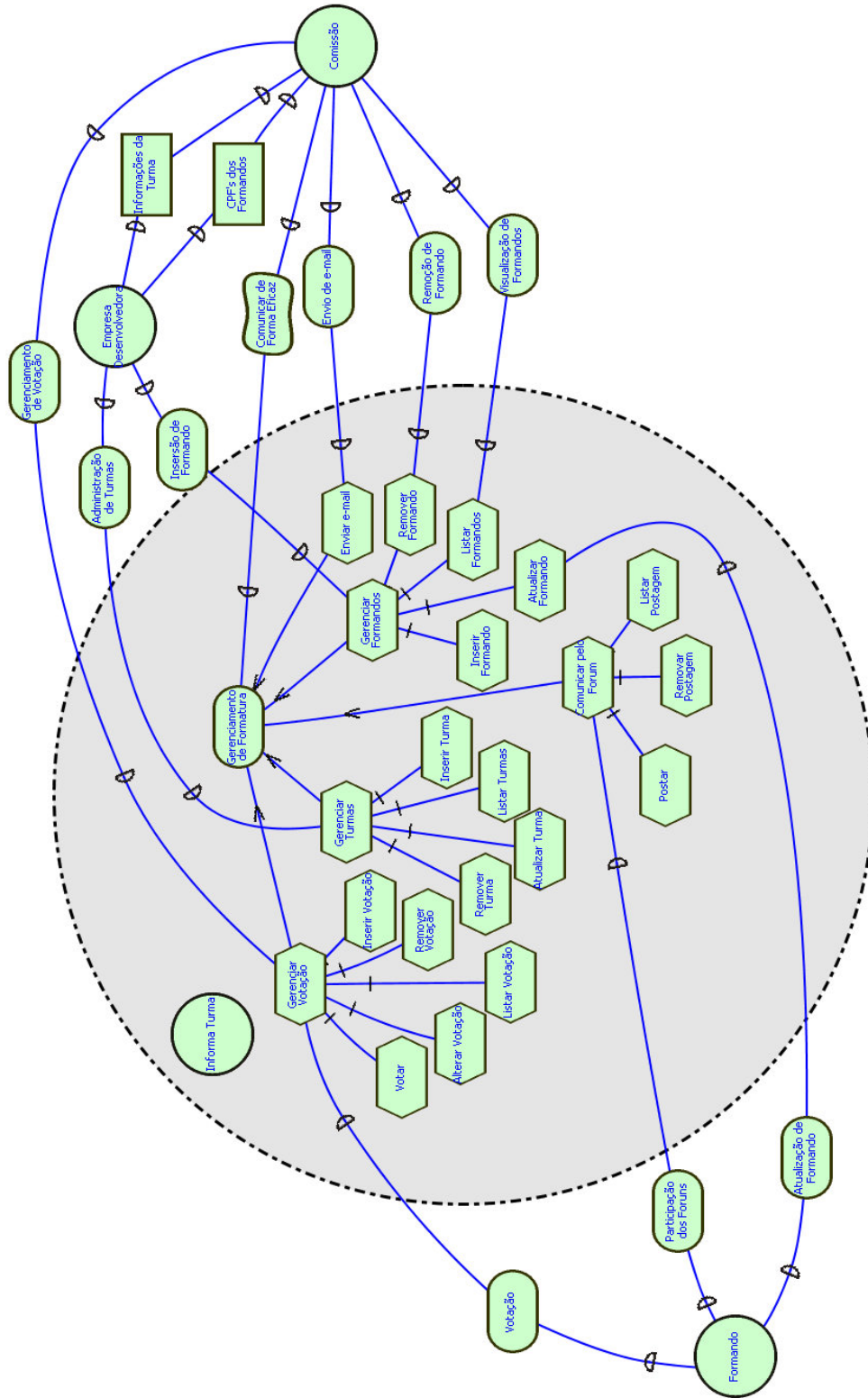


Figura 14: Modelo SR do exemplo Informa Turna

5.3 Procedimento de Avaliação

Para demonstrar o uso das métricas foi escolhido o método GQM (Goal Question Metric) (BASILI; CALDIERA; ROMBACH, 1994). O GQM se baseia na análise hierárquica dos tributos e qualidade com o intuito de definir como cada atributo vai ser analisado através de perguntas que são respondidas por métricas.

O método GQM não foi proposto originalmente para suportar o reuso de métricas apenas para defini-las. Com as métricas já foram definidas no Capítulo 4 a fase de definição das métricas foi substituída pela seleção de métricas dentro do conjunto apresentado nessa dissertação.

O procedimento usado na avaliação consiste em definir o objetivo da avaliação, determinar perguntas que sejam usadas para alcançar esse objetivo e selecionar métricas para responder as perguntas. A partir dos resultados são feitas as análises que respondem às perguntas resultando em uma conclusão sobre o atributo de qualidade que está sendo avaliado.

A seguir é definido o objetivo para avaliar a qualidade do modelo Informa Turma.

Objetivo 1 – **Analisar** Documentos de Requisitos Modelados com i* **com o propósito** de avaliar a qualidade dos modelos i* **com respeito à** legibilidade **sob o ponto de vista** de time de qualidade **no contexto de** projetos acadêmicos. De acordo com esse objetivo é possível identificar o objeto da análise, que é o documento de requisitos modelado com i*; a finalidade, que é avaliar a qualidade; o fator de qualidade, que é a legibilidade; o ponto de vista que é o do time de qualidade; e o contexto no qual a avaliação transcorrerá que é o contexto de projetos acadêmicos.

A seguir são definidas as questões usadas para alcançar o objetivo e são selecionadas as métricas para responder cada questão. Com base nos resultados obtidos pelas métricas as perguntas serão respondidas e será possível determinar se o objetivo vai ser alcançado ou não.

O modelo contém algum erro? - A presença de erros pode tornar o modelo ilegível, por não permitir ao leitor tirar conclusões sobre o modelo. Essa pergunta é respondida pela métrica Número de Erros (E_1).

O modelo apresenta alguma construção ambígua? - Ambigüidade prejudica o entendimento do que está escrito, impedindo o leitor de distinguir qual das interpretações é a correta. Esta pergunta pode ser respondida pelas métricas Número de Palavras Vagas

(A_1) e Taxa de Dependências Ambíguas (A_2) .

O modelo está muito complexo? - Modelos grandes, com o cruzamento de muitas ligações ou presença de muitos elementos gráficos podem dificultar o entendimento dos leitores. Para responder essa pergunta serão utilizadas as métricas Complexidade Ciclomática (C_1) , Vocabulário (C_2) , Comprimento (C_3) , Volume (C_4) , Dificuldade (C_5) , Esforço (C_6) e Entropia (C_7) .

A aceitação das métricas é feita através da interpretação dos resultados das medidas. Cada métrica deve possuir uma faixa de valores que podem ser aceitos para um determinado atributo de qualidade. A maior parte das métricas definidas no capítulo 4 possuem valores alvo definidos. Esses valores, no entanto, representam a situação ideal não oferecendo margem de tolerância para interpretação.

Como pode ser visto em Park, Goethert e Florac (1996) a faixa de valores pode ser classificada como: aceita, quando se encontra o valor alvo definido para a métrica; marginal quando encontra um valor próximo ao ideal que seja aceitável; e rejeitado quando o valor encontrado não pode ser considerado válido. Para permitir a avaliação dos resultados, intervalos de aceitação foram definidos para as métricas usadas no exemplo. As faixas de valores para o exemplo apresentado nessa seção foram definidas de forma *ad hoc* apenas para efeito ilustrativo. Para uma aplicação real elas precisam ser definidas pelos avaliadores e calibradas de acordo com o histórico de uso das métricas.

A tabela 32 apresenta as métricas e os intervalos de aceitação que serão usados no exemplo. Para cada métrica estão definidos os intervalos de aceitação classificados em aceito, marginal e rejeitado.

Tabela 32: Intervalo de Aceitação das Métricas para o Exemplo Informa Turma

Métricas	Intervalos de Aceitação		
	Aceita	Marginal	Rejeitada
Número de Erros (E_1)	$E_1 = 0$	$0 < E_1 \leq 10$	$E_1 > 10$
Taxa de Metas sem ligação com Rotinas (D_1)	$D_1 = 0$	$0 < D_1 \leq 0,3$	$D_1 > 0,3$
Taxa de Dependências sem Ligação a Elementos Internos no Ator (D_2)	$D_2 = 0$	$0 < D_2 \leq 0,3$	$D_2 > 0,3$
Taxa de <i>Softgoals</i> que não são tratados por Contribuição (D_3)	$D_3 = 0$	$0 < D_3 \leq 0,3$	$D_3 > 0,3$
Número de Palavras Vagas (A_1)	$A_1 = 5$	$5 < A_1 \leq 10$	$A_1 > 10$
Taxa de Dependências Ambíguas (A_2)	$A_2 = 0$	$0 < A_2 \leq 5$	$A_2 > 5$
Complexidade Ciclomática (C_1)	$C_1 \leq 10$	$10 < C_1 \leq 15$	$C_1 > 15$
Vocabulário (C_2)	$C_2 \leq 70$	$70 < C_2 \leq 120$	$C_2 > 120$
Comprimento (C_3)	$C_3 \leq 120$	$120 < C_3 \leq 150$	$C_3 > 150$
Volume (C_4)	$C_4 \leq 800$	$800 < C_4 \leq 1000$	$C_4 > 1000$
Dificuldade (C_5)	$C_5 \leq 4$	$4 < C_5 \leq 7$	$C_5 > 7$
Esforço (C_6)	$C_6 \leq 1000$	$1000 < C_6 \leq 1500$	$C_6 > 1500$
Entropia (C_7)	$C_7 \leq 5,0$	$5,0 < C_7 \leq 10,0$	$C_7 > 10,0$

5.4 Coleta de Dados

Para obter o valor de cada métrica primeiro é necessário fazer a coleta dos dados. A seguir será apresentada a coleta dos dados de cada métrica para o exemplo Informa Turma.

A métrica Número de Erros (E_1) depende apenas do dado Número de Erros. A coleta desse dado é feita através da comparação do modelo da figura 14 com as entradas do catálogo de erros típicos (Apêndice A). Cada vez que houver um erro correspondente ao encontrado no catálogo o valor do dado é incrementado em uma unidade. Ao final da busca o valor total será obtido. No caso do exemplo, nenhum dos erros típicos enumerados no catálogo foi encontrado assim o valor do dado e conseqüentemente da métrica foi zero (0).

A métrica Taxa de Metas sem ligação com Rotinas (D_1) é obtida através da contagem dos dados Número de Metas sem ligação com Rotinas (NMR) e Número de Metas (NM). O dado Número de Metas é o número total de metas que aparecem nos modelos SR sem contar as metas que são *dependums*. No exemplo InformaTurma (vide Figura 14) a única meta encontrada nessa situação é a meta *Gerenciamento de Formatura*, portanto o valor de NM é um (1). O dado NMR é obtido através da contagem do número de metas que não são *dependums* e que não estão ligados à rotinas através da ligação *Means-End*, como a única meta que não é um *dependum* está ligada à rotinas o valor do dado NMR é zero (0). Com base nesses dados o valor calculado para a métrica (D_1) será

$$D_1 = \frac{NMR}{NM} = \frac{0}{1} = 0$$

assim o valor da métrica D_1 é zero.

A métrica Taxa de Dependências sem Ligação a Elementos Internos no Ator (D_2) é obtida através da contagem dos dados Número de Dependências sem Ligação à Elemento Interno no Ator (NDLEIA) e Número de Dependências do Ator (NDA). O dado NDA é o total de dependências que chegam aos atores com as fronteiras abertas, no exemplo Informa Turma o único ator com a fronteira aberta é o ator *Informa Turma*. Ao todo são 10 dependências que chegam ao ator *Informa Turma* assim o valor do dado NDA é 10. Todas as dependências estão ligadas a elementos internos do ator *Informa Turma* dessa forma o valor do dado NDLEIA é 0. O valor da métrica (D_2) é calculado através da fórmula

$$D_2 = \frac{NDLEIA}{NDA} = \frac{0}{10} = 0$$

resultando no valor zero (0).

A métrica Taxa de *Softgoals* que não são tratados por Contribuição (D_3) não pode ser calculada para o modelo SR pois não há *softgoals* dentro da fronteira do ator *Informa Turma*.

A métrica Número de Palavras Vagas (A_1) é obtida através da contagem do número de vezes que palavras vagas aparecem nos rótulos dos elementos intencionais. São procuradas palavras como conectores lógicos (i.e., E, OU, Não), quantificadores (Cada, Qualquer, Todo) ou pronomes (Dele, Seu, etc.). No caso do exemplo Informa Turma não há ocorrência de nenhuma palavra vaga o que resulta no valor zero para a métrica A_1 .

A métrica Taxa de Dependências Ambíguas (A_2) é obtida através dos dados Número de Dependências Relacionadas a mais de um Dependee (NDRD1), Número de Dependências Relacionadas a mais de um Dependee (NDRD2) e Número de Dependências (ND). O dado número de Dependências é o número total de dependências no modelo que equivale à 12. Os dados NDRD1 e NDRD2 contam o número de dependências que possuem bifurcações, como nenhum caso foi encontrado no exemplo o valor de ambos os dados é zero (0). O valor da métrica A_2 é então computado pela fórmula

$$A_2 = \frac{NDRD1 + NDRD2}{ND} = \frac{0 + 0}{12} = 0$$

A métrica Complexidade Ciclomática (C_1) é calculada a partir dos dados Número de Elementos (NE), Número de Ligações (NL) e Número de Componentes Conexos (NCC). O dado NE corresponde ao número total de elementos intencionais (i.e., Recursos, Tarefas, Metas e *Softgoals*) e de atores (i.e., Ator, Papel, Posição e Agente) a partir da contagem do elementos no exemplo Informa Turma o valor de NE é 38. O dado Número de Ligações corresponde ao número total de ligações presentes no modelo (e.g., ligações de dependência, contribuição, *Means-end*, decomposição), o exemplo Informa Turma apresenta 45 ligações. O dado número de Componentes Conexos apresenta valor um 1 pois o grafo que representa o modelo i^* não apresenta sub-partes independentes. Com base nos valores dos dados o valor da complexidade ciclomática é calculado através da fórmula

$$C_1 = NL - NE + 2 \times NCC = 45 - 38 + 2 \times 1 = 9$$

, obtendo assim o valor nove (9).

A métrica Vocabulário (C_2) é obtida através da contagem do dados Número de Ele-

mentos Distintos (NED) e Número de Ligações Distintas (NLD). O valor de NED corresponde ao total de elementos sem considerar as repetições, no caso de um elemento ter um mesmo nome que outro todos os elementos com aquele nome só são contados uma vez. No caso do exemplo Informa Turma todos os elementos são distintos pois não há repetições dessa maneira o valor de NED é 38. NLD corresponde ao número de ligações sem contar as repetições, no caso do exemplo Informa Turma apenas três tipos de ligações estão presentes que são a ligação de dependência, que se repete 24 vezes; a ligação de decomposição, que se repete 16 vezes; e a ligação de *Means-End* que se repete 5 vezes, resultando no valor de NLD igual a três. O valor de C_2 é calculado pela fórmula

$$C_2 = NLD + NED = 38 + 3 = 41$$

.

A métrica Comprimento (C_3) é obtida através da contagem do tamanho total dos modelos, sendo necessários os valores de NE e NL já coletado anteriormente. O valor de C_3 é dado por

$$C_3 = NE + NL = 38 + 45 = 83$$

.

A métrica Volume (C_4) é derivada das métricas Vocabulário (C_2) e Comprimento (C_3), no exemplo Informa Turma o valor de C_4 é dado por

$$C_4 = C_3 \times \log_2 C_2 = 83 \times 5,357 = 444,67$$

A métrica Dificuldade (C_5) é calculada através dos dados Número de Elementos (NE), Número de Ligações Distintas (NLD) e Número de Elementos Distintos (NED). Ela é obtida pela fórmula

$$C_5 = \frac{NLD}{2} \times \frac{NE}{NED} = \frac{3}{2} \times \frac{38}{38} = 1,5 \times 1 = 1,5$$

.

A métrica Esforço (C_6) é derivada das métricas Volume (C_4) e Dificuldade (C_5). Ela é obtida pela seguinte fórmula

$$C_6 = C_4 \times C_5 = 444,67 \times 1,5 = 667,01$$

.

A métrica Entropia (C_7) necessita do dado Probabilidade Referencial ($p(x_i)$) que é obtido através da contagem das referências feitas aos elementos no i^* . Considerando um modelo i^* como um grafo dirigido as referências serão o número de ligações que entram e saem de um elemento. Por exemplo, o ator *Comissão* possui cinco ligações saindo e duas entrando, num total sete referências. Quando calculada a proporção dessas referências em relação ao total é obtido o valor 0,078. Então o valor obtido para o ator *Comissão* é $p(\text{Comissão}) = 0,078$. A probabilidade referencial é então calculada para todos os elementos contando quantas vezes as ligações chegam e saem de cada um. Com base neles é calculada a Entropia para o modelo ela é dada por

$$C_7 = \sum_i p(x_i) \times \log_2 p(x_i)$$

, onde x é o elemento em foco e i varia de 1 até o valor de NE (Número de Elementos).

Na tabela 33 são apresentadas as probabilidades referenciais de cada elemento e ao final o valor da Entropia é calculado com base nas probabilidades referenciais. Na primeira coluna estão os nomes dos elementos que correspondem ao x_i . Na segunda coluna está o valor do Referenciado que corresponde a quantas ligações chegaram ao elemento, na terceira coluna está o valor Referencia que corresponde a quantas ligações estão saindo do elemento. Na quarta coluna estão os valores das probabilidades referenciais que é obtido pela soma dos valores das colunas anteriores divididos pelo total do número de referências.

Tabela 33: Dados para a Métrica Entropia do Exemplo Informa Turma

Elemento (x_i)	Ligações	$p(x_i)$	$p(x_i) \times \log_2 p(x_i)$
Votação	2	0,022	-0,122
Participação dos Fóruns	2	0,022	-0,122
Atualização dos Formandos	2	0,022	-0,122
Gerenciar Votação	8	0,089	-0,310
Votar	1	0,011	-0,072
Alterar Votação	1	0,011	-0,072
Listar Votação	1	0,011	-0,072
Remover Votação	1	0,011	-0,072
Inserir Votação	1	0,011	-0,072
Gerenciar Turmas	6	0,067	-0,260
Remover Turma	1	0,011	-0,072
Atualizar Turma	1	0,011	-0,072
Listar Turma	1	0,011	-0,072
Inserir Turma	1	0,011	-0,072

continua na próxima página

Tabela 33: Dados para a Métrica Entropia do Exemplo Informa Turma (Continuação)

Elemento (x_i)	Ligações	$p(x_i)$	$p(x_i) \times \log_2 p(x_i)$
Comunicar Pelo Fórum	5	0,056	-0,232
Postar	5	0,011	-0,072
Remover Postagem	1	0,011	-0,072
Listar Postagem	1	0,011	-0,072
Gerenciar Formando	6	0,067	-0,260
Inserir Formando	1	0,011	-0,072
Atualizar Formando	2	0,022	-0,122
Listar Formando	2	0,022	-0,122
Remover Formando	2	0,022	-0,122
Enviar E-mail	2	0,022	-0,122
Gerenciamento Formatura	6	0,067	-0,260
Informa Turma	0	0,000	0,000
Gerenciamento de Votação	2	0,022	-0,122
Administração de Turmas	2	0,022	-0,122
Inserção de Turmas	2	0,022	-0,122
Empresa Desenvolvedora	4	0,044	-0,200
Informações da Turma	2	0,022	-0,122
CPF's dos Formandos	2	0,022	-0,122
Comunicar de Forma Eficiente	2	0,022	-0,122
Envio de E-mail	2	0,022	-0,122
Remoção de Formando	2	0,022	-0,122
Visualização de Formando	2	0,022	-0,122
Comissão	7	0,078	-0,287
Formando	3	0,033	-0,164
Valor da Entropia (C_7)		3,494	

Os valores obtidos para os dados são apresentados na tabela 34 e as métricas calculadas são apresentadas na tabela 35.

Tabela 34: Dados para o Exemplo Informa Turma

Dados	Valores
NErros	0
NMR	0
NM	1
NDLEIA	0
NDA	10
NSC	0
NSA	0
NPV	0
NDRD1	0
NDRD2	0
NL	45
NE	38
NCC	1
NLD	3
NED	38
NEC	1

Tabela 35: Métricas para o Exemplo Informa Turma

Métricas	Valores
Número de Erros (E_1)	0
Taxa de Metas sem ligação com Rotinas (D_2)	0
Taxa de Dependências sem Ligação a Elementos Internos no Ator (D_2)	0
Taxa de <i>Softgoals</i> que não são tratados por Contribuição	-
Número de Palavras Vagas (A_1)	0
Taxa de Dependências Ambíguas (A_2)	0
Complexidade Ciclomática (C_1)	9
Vocabulário (C_2)	41
Comprimento (C_3)	83
Volume (C_4)	444,67
Dificuldade (C_5)	1,5
Esforço (C_6)	667,01
Entropia (C_7)	3,494

5.5 Análise dos Resultados

A interpretação das métricas é feita através da resposta às perguntas definidas para cada métrica. De acordo com o valor definido nos intervalos de validação da tabela 32 é possível guiar as respostas para cada pergunta.

A questão *O modelo contém algum erro?* é respondida pela métrica Número de Erros (E_1). Como o valor obtido para a métrica foi zero a resposta da pergunta é *Não há erros sintáticos no modelo Informa Turma*. Com base nessa resposta é possível concluir os erros não interferem na legibilidade do modelo.

A questão *O modelo apresenta alguma construção ambígua?* pode ser respondida pelas métricas Número de Palavras Vagas (A_1) e Taxa de Dependências Ambíguas (A_2). Ambigüidade prejudica o entendimento do que está escrito, impedindo o leitor de distinguir qual das interpretações é a correta. Como o valor de ambas as métricas foi zero elas foram aceitas de acordo com os intervalos de aceitação previstos. Assim a resposta para a pergunta é *Não há construções ambíguas no modelo Informa Turma*. Com base na resposta obtida é possível concluir que não há interferência de problemas com ambigüidade nos modelos.

A questão *O modelo está muito complexo?* é respondida pelas métricas Complexidade Ciclomática (C_1), Vocabulário (C_2), Comprimento (C_3), Volume (C_4), Dificuldade (C_5), Esforço (C_6) e Entropia (C_7). Modelos grandes, com o cruzamento de muitas ligações ou presença de muitos elementos gráficos podem dificultar o entendimento dos leitores. De acordo com os intervalos de aceitação definidos na tabela 32 todas as métricas apresentam valor aceitável para a complexidade. A Complexidade Ciclomática (C_1) apresentou o valor 9 onde o limite aceitável era $C_1 \leq 10$. O Vocabulário (C_2) apresentou o valor 41 onde o limite aceitável era $C_2 \leq 70$. O Comprimento (C_3) apresentou o valor 83 onde o limite aceitável era $C_3 \leq 120$. O Volume (C_4) apresentou o valor 444,67 onde o limite aceitável era $C_4 \leq 800$. A Dificuldade (C_5) apresentou o valor 1,5 onde o limite aceitável era $C_5 \leq 4$. O Esforço (C_6) apresentou o valor 667,01 onde o limite aceitável era $C_6 \leq 1000$. A Entropia (C_7) apresentou o valor 3,494 onde o limite aceitável era $C_7 \leq 5,0$. Com base no resultado dessas métricas a resposta para a pergunta é *O modelo não está muito complexo*. Os resultados das métricas indicam que o modelo Informa Turma não apresentou nenhum problema com complexidade que possa afetar a legibilidade.

O resultado da análise das questões não revelou nenhum problema relevante com o modelo Informa Turma com relação a legibilidade. No entanto, outras análises poderiam

ser feitas levando em consideração outros atributos de qualidade. Um atributo que levasse em consideração as métricas de detalhamento, por exemplo, poderia revelar que o modelo não possui nenhum detalhamento em relação aos *softgoals*. O que seria uma falta grave já que os *softgoals* são usados para modelar requisitos não-funcionais e isso não está presente no modelo Informa Turma.

5.6 Validação das Métricas

Um conjunto de métricas deve ser validada para mostrar que são capazes de representar devidamente os atributos que se destinam. Para isso podem ser aplicados testes estatísticos que permitam analisar a acurácia das métricas (IEEE-1061, 1992). Essa dissertação se limitou a apresentar um exemplo, que como mencionado anteriormente não possui valor estatístico para validar a proposta apresentada. Um experimento foi conduzido para validar as métricas mas não apresentou resultados relevantes devido ao pequeno número de amostras, apenas cinco amostras foram coletadas durante a tentativa de validar as métricas. Por esses motivos a validação das métricas não fez parte da dissertação constando como um trabalho futuro.

A validação não significa uma validação universal das métricas para todas as aplicações, ela se refere ao relacionamento entre as métricas e o atributo de qualidade de uma dada aplicação (IEEE-1061, 1992). Para realizar a validação é necessário identificar entre as métricas quais são métricas diretas e quais são métricas para previsão (IEEE-1061, 1992). Métricas diretas são aquelas que correspondem ao atributo de qualidade avaliado sem a necessidade de interpretação de outras métricas. Um exemplo de métrica direta apresentada nessa dissertação é a métrica Número de Erros (E_1) que corresponde diretamente ao fator de qualidade que está avaliando que é o caso da presença de erros.

As métricas para previsão são usadas para identificar qual o comportamento futuro do atributo de qualidade com base no valor atual, um exemplo apresentado nessa dissertação de métrica para previsão são as métricas de Dificuldade (C_5) e Esforço (C_6), que foram mapeadas das métricas de Halstead (1977).

Seguindo a metodologia de validação sugerida pelo IEEE-1061 (1992) as métricas diretas não precisam ser validadas pois já correspondem diretamente ao atributo de qualidade que deveriam representar. Já as métricas para previsão devem ser validadas estudando a correlação entre elas e as métricas diretas. Para cada atributo de qualidade devem ser escolhidas métricas diretas que o representem, todas as métricas para previsão são

comparadas às métricas diretas para o seu respectivo atributo de qualidade.

Esse estudo de correlação exige a coleta de um número significativo de amostras das quais são coletadas as métricas. As métricas diretas são comparadas as outras em termos de correlação, consistência, previsibilidade, poder de discriminação e confiabilidade (IEEE-1061, 1992). Para cada um desses fatores é estabelecido um limiar no qual as métricas podem variar para serem consideradas válidas.

5.7 Conclusão

Nesse capítulo foi apresentado o exemplo Informa Turma no qual foram aplicadas as métricas definidas no capítulo 4. As métricas foram apresentadas com um passo-a-passo da coleta dos dados, onde cada métrica foi calculada demonstrando como isso pode ser feito. Esse exemplo não possui valor experimental para validar as métricas, no entanto, é uma forma de demonstrar o uso das métricas e como poderia ser feita a análise dos resultados. Além do exemplo foi feita uma breve discussão dos passos para se validar as métricas.

6 Conclusão e Trabalhos Futuros

Este capítulo apresenta as conclusões deste trabalho e trabalhos futuros relacionados a ele. É feita a revisão do problema e do objetivo, são ressaltadas as principais contribuições e é apresentada uma comparação deste trabalho com os trabalhos relacionados à ele.

6.1 Revisão do Problema e Objetivo

A demanda por software de qualidade aumenta a necessidade de obter em todas as fases do desenvolvimento produtos de qualidade. Em particular, os documentos de requisitos apresentam um produto que é a base de todas as outras fases do desenvolvimento do software. Como apresentado anteriormente corrigir erros de documentos nas fases iniciais do desenvolvimento reduz os custos das correções nas fases posteriores. Para identificar possíveis problemas na modelagem de requisitos podem ser utilizadas técnicas de avaliação de qualidade. No caso específico dos documentos de requisitos que utilizam i^* como técnica de modelagem ainda há poucos trabalhos que tentam avaliar a qualidade deste produto. Por esses motivos essa dissertação teve como principal objetivo definir um conjunto de métricas para avaliar modelo i^* .

6.2 Principais Contribuições

Este trabalho apresentou a definição de métricas para avaliação da qualidade de modelos i^* e mostra exemplos de como aplicar as métricas. As métricas procuram relacionar construções básicas usadas pela técnica i^* com qualidades conhecidas de especificação de requisitos. As métricas apresentadas podem ser usadas para ajudar a identificar e diminuir a quantidade de erros nos documentos de requisitos que são representados com a técnica i^* .

As principais contribuições deste trabalho estão relacionadas a seguir:

1. Revisão dos principais conceitos de engenharia de requisitos e da técnica i^* . Os conceitos da dissertação e do i^* usados para entender a dissertação foram aplicados aos exemplos;
2. Revisão de conceitos da avaliação de qualidade para requisitos;
3. Proposta de um conjunto de métricas específicas para avaliação da qualidade de documentos de requisitos modelados com i^* ;
4. Levantamento dos principais erros em modelos i^* . Alguns levantamentos já feitos foram compilados no formato de um catálogo para facilitar a procura de erros nos modelos i^* .

6.3 Comparação com Trabalhos Relacionados

A tabela 36 mostra uma comparação da proposta apresentada nessa dissertação com os trabalhos relacionados apresentados na seção 1.3. Os trabalhos foram comparados em termos de:

- relação com o i^* , as abordagens são avaliadas em termos do nível de detalhes que aborda o i^* . Elas podem não abordar quando não fazem nenhum tipo de referência ao i^* . Abordar de forma genérica quando citam o i^* mas não entram em detalhes nas construções do i^* . Ou abordar de forma específica, quando usam as construções específicas do i^* na avaliação;
- métricas, compara as abordagens segundo o uso de métricas para avaliar a qualidade;
- inspeção, as abordagens podem usar inspeção ou gerar artefatos que dão suporte à inspeção como forma de avaliar qualidade;
- metodologia, algumas abordagens descrevem metodologias para usar na avaliação, outras usam metodologias já existentes.

A tabela 36 apresenta uma comparação dos trabalhos relacionados ao apresentado nessa dissertação, que está referenciado como (SANTOS, 2008). O símbolo $--$ é usado para indicar que a propriedade não é atendida, por exemplo, se uma abordagem não usa métricas ela é marcada como esse símbolo. O símbolo $+/-$ é usado para indicar que uma abordagem atende parcialmente uma propriedade, por exemplo, se uma abordagem não aborda especificamente o i^* é marcada com esse símbolo. Caso uma das abordagens apresente uma propriedade ela é assinalada com o símbolo $++$.

Tabela 36: Comparação com Trabalhos Relacionados

Abordagem	Relação com i^*	Métricas	Inspeção	Metodologia
(CABRAL, 2007)	$+/-$	$--$	$++$	$--$
(REIS, 2004)	$--$	$++$	$--$	$++$
(RAMOS et al., 2006)(RAMOS et al., 2007)	$+/-$	$++$	$+/-$	$++$
(FRANCH; GRAU; QUER, 2004) (FRANCH, 2006) (FRANCH; GRAU, 2008)	$++$	$++$	$--$	$++$
(SANTOS, 2008)	$++$	$++$	$+/-$	$--$

O trabalho apresentado nessa dissertação usa métricas assim como os trabalhos de Reis (2004), Ramos et al. (2006), Ramos et al. (2007), Franch, Grau e Quer (2004) e

Franch (2006). No entanto, é dado um suporte parcial a inspeção é dado através do catálogo de erros típicos, o que se assemelha aos guias apresentados em Cabral (2007).

O trabalho de Franch (2006) difere do apresentado nessa dissertação tanto na estrutura como na finalidade. Apesar de tratar do especificamente do i^* Franch (2006) não apresenta instâncias de métricas, ele apresenta uma forma de definir métricas mas não cria instâncias para as métricas. Além disso a finalidade dessa abordagem é avaliar modelos para definir a arquitetura que será adotada pelo sistema desenvolvido. Franch (2006) parte do pressuposto que não há problemas com os modelos, para usar essa abordagem os modelos sempre devem estar corretos. A abordagem apresentada nessa dissertação, no entanto se aplica à procura por problemas, sendo usada principalmente para identificar problemas.

O trabalho de Cabral (2007) apresenta uma comparação experimental de técnicas de inspeção de documentos de requisitos. Ele tem uma finalidade semelhante ao apresentado nessa dissertação que é a procura por problemas em documentos de requisitos, no entanto difere quanto a forma com isso é feito. Nessa dissertação a procura por problemas é feita através de métricas enquanto em Cabral (2007) são usadas técnicas de leitura.

Reis (2004) apresenta uma metodologia que é baseada em métricas mas não aborda o i^* . Além de não abordar o i^* a abordagem de Reis (2004) tem a finalidade de comparar aplicações e não procurar por problemas.

A abordagem AIRDoc (RAMOS et al., 2007) apresenta uma metodologia para melhor documentos de requisitos que usa métricas para identificar problemas. Ela aborda o i^* de forma genérica onde não há referências à construções específicas do i^* . Abordagem apresentada nessa dissertação não define uma metodologia, mas é específica para modelos i^* . Ambas são voltadas para a procura de problemas.

6.4 Trabalhos Futuros

Durante o desenvolvimento deste trabalho se viu a necessidade de automatizar a coleta das métricas para reduzir o custo de utilizá-las na avaliação da qualidade. As métricas objetivas, que não dependem do conhecimento dos avaliadores, podem ser coletadas de forma automática. Para isso é necessário desenvolver uma ferramenta para coleta dos dados e cálculo das métricas. Fica como trabalho futuro desenvolver a ferramenta para coleta de dados. Além disso, também é possível fazer a integração da ferramenta de coleta às ferramentas de modelagem para obter dados de forma dinâmica.

Como pode ser visto em norma (IEEE-1061, 1992), é importante validar métricas que são utilizadas para previsão. A validação experimental não foi feita nessa dissertação. Para que ela possa ser feita é necessário coletar amostras e avaliar essas amostras para obter as métricas. Com base nas métricas é necessário identificar a correlação entre as métricas e atributo de qualidade que será avaliado.

Como as métricas apresentadas aqui são limitadas à avaliação do modelo como um artefato finalizado elas não lidam com o processo da engenharia de requisitos. Outro trabalho futuro que pode ser apontado é a criação de métricas para outros atributos de qualidade, principalmente atributos de processo como rastreabilidade e volatilidade.

As limitações do trabalho apresentado quanto à metodologia e o suporte a inspeção, vide tabela 36, podem ser resolvidas através da definição de uma metodologia de avaliação e aplicação das métricas, e da definição de artefatos para dar suporte a inspeção. A metodologia AIRDoc (RAMOS et al., 2006) pode utilizar as métricas apresentadas para procurar problemas e propor melhorias aos documentos de requisitos modelados com i*. Uma atividade futura é integrar o uso das métricas apresentadas aqui com a metodologia AIRDoc, resolvendo o problema da falta de uma metodologia. Para o caso de suporte a inspeção, novos artefatos podem ser definidos para guiar a leitura de modelos e a coleta dos dados. Esse suporte já é oferecido para a métrica Número de Erros (E_1) através do Catálogo de Erros Típicos, vide Apêndice A, como trabalho futuro fica a extensão do suporte a inspeção para as outras métricas.

Referências

- ALENCAR, F.; MOREIRA, A. M. D.; ARAUJO, J.; CASTRO, J.; SILVA, C. T. L. L.; MYLOPOULOS, J. Using aspects to simplify i* models. In: *Proceedings of 14th International IEEE Requirements Engineering Conference (RE'06)*. Minneapolis, USA: IEEE Computer Society, 2006. p. 335–336.
- ALI, M. J. *Metrics for Requirements Engineering*. Dissertação (Mestrado) — Umea University, Department of Computing Science, Sweden, 2006.
- BASILI, V. R.; CALDIERA, G.; ROMBACH, H. D. The goal question metric approach. *Encyclopedia of Software Engineering*, p. 528–532, 1994.
- BASILI, V. R.; GREEN, S.; LAITENBERGER, O.; LANUBILE, F.; SHULL, F.; SORUMGARD, S.; ZELKOWITZ, M. V. The empirical investigation of perspective-based reading. *Empirical Software Engineering: An International Journal*, Kluwer Academic Publishers, v. 1, n. 2, p. 133–164, 1996.
- BERNARDEZ, B.; DURAN, A.; GENERO, M. Empirical evaluation and review of a metrics-based approach for use case verification. *Journal of Research and Practice in Information Technology*, v. 36, n. 4, 2004. Disponível em: <<http://www.jrpit.acs.org.au/jrpit/JRPITVolumes/JRPIT36/JRPIT36.4.247.pdf>>.
- BERRY, D.; KAMSTIES, E.; KRIEGER, M. M. *From Contract Drafting to Software Specification: Linguistic Sources of Ambiguity*. [S.l.: s.n.], 2003. Available at: <http://se.uwaterloo.ca/dberry/handbook/ambiguityHandbook.pdf>.
- BEZIVIN, J. On the unification power of models. *Software and System Modeling*, v. 4, n. 2, p. 171–188, 2005.
- BOEHM, B. W. Verifying and validating software requirements and design specification. *IEEE Software*, v. 1, p. 94–103, jan 1984.
- CABRAL, M. S. *Técnicas de Leitura para Inspeção da Especificação Inicial de Requisitos*. Dissertação (Mestrado) — Centro de Informática, Universidade Federal de Pernambuco, 2007.
- CABRAL, M. S.; ALENCAR, F.; CASTRO, J.; PASTOR, O.; SÁNCHEZ, J. Aplicação de técnicas de leitura durante a análise de requisitos. In: *Proceedings of the 11th Workshop on Requirements Engineering - WER 2008*. Barcelona, Spain: [s.n.], 2008. Disponível em: <<http://wer.inf.puc-rio.br/>>.
- CASTRO, J.; KOLP, M.; MYLOPOULOS, J. Towards requirements-driven information systems engineering: the tropos project. *Information Systems*, Elsevier Science Ltd., Oxford, UK, v. 27, n. 6, p. 365–389, 2002. ISSN 0306-4379. Disponível em: <[http://dx.doi.org/10.1016/S0306-4379\(02\)00012-1](http://dx.doi.org/10.1016/S0306-4379(02)00012-1)>.

- CHUNG, L. K.; NIXON, B. A.; YU, E.; MYLOPOULOS, J. *Non-Functional Requirements in Software Engineering*. [S.l.]: Kluwer Publishing, 2000. (International Series in Software Engineering, v. 5).
- CLOTET, R.; FRANCH, X.; LÓPEZ, L.; MARCO, J.; SEYFF, N.; GRÜNBAKER, P. On the meaning of inheritance in i*. In: PERCINI, B.; GULLA, J. A. (Ed.). *The 19th International Conference on Advanced Information Systems Engineering. Proceedings of Workshops and Doctoral Consortium*. [S.l.]: Tapir Academic Press, 2007. v. 2. ISBN:978-82-519-2246-3.
- COSTELLO, R. J.; LIU, D.-B. Metrics for requirements engineering. *Journal of Systems and Software*, Elsevier Science Inc., New York, NY, USA, v. 29, n. 1, p. 39–63, 1995. ISSN 0164-1212.
- DARDENNE, A.; FICKAS, S.; LAMSWEERDE, A. van. Goal-directed concept acquisition in requirements elicitation. In: *IWSSD '91: Proceedings of the 6th international workshop on Software specification and design*. Los Alamitos, CA, USA: IEEE Computer Society Press, 1991. p. 14–21. ISBN 0-8186-2320-9 (PAPER).
- DAVIS, A.; OVERMYER, S.; JORDAN, K.; CARUSO, J.; DANDASHI, F.; DINH, A.; KINCAID, G.; LEDEBOER, G.; REYNOLDS, P.; SITARAM, P.; TA, A.; THEOFANOS, M. Identifying and measuring quality in software requirements specifications. In: *IEEE-CS International Software Metrics Symposium*. Los Alamitos, California: IEEE Computer Society Press, 1993. p. 141–152.
- DAVIS, A. M. *Software Requirements: Objects, Functions and States*. New Jersey, USA: Prentice Hall, 1993.
- FEITOSA, C. M. A. C. *Definição de um Processo de Medição e Análise com base nos Requisitos do CMMI*. Dissertação (Mestrado) — Centro de Informática, Universidade Federal de Pernambuco, Dezembro 2004.
- FENTON, N. E.; PFLEEGER, S. *Software Metrics, A Rigorous Approach*. [S.l.]: Int. Thomson Computer Press, 1996.
- FRANCH, X. On the quantitative analysis of agent-oriented models. In: *Proceedings 18th International Conference in Advanced Information Systems Engineering, CAiSE*. [S.l.]: Springer, 2006. p. 495–509.
- FRANCH, X.; GRAU, G. Towards a catalogue of patterns for defining metrics over i* models. In: *Proceedings of 20th International Conference on Advanced Information Systems Engineering, CAiSE 2008 Montpellier, France, June 16-20, 2008*. [S.l.]: Springer Berlin / Heidelberg, 2008. v. 5074/2008, p. 197–212.
- FRANCH, X.; GRAU, G.; QUER, C. A framework for the definition of metrics for actor-dependency models. In: *RE '04: Proceedings of the Requirements Engineering Conference, 12th IEEE International (RE'04)*. Washington, DC, USA: IEEE Computer Society, 2004. p. 348–349. ISBN 0-7695-2174-6.
- FRANCH, X.; GRAU, G.; QUER, C. Un marco para la definición de métricas sobre modelos de dependencias entre actores. In: *Anais do WER05 - Workshop em Engenharia de Requisitos*. Porto, Portugal: [s.n.], 2005. p. 185–196.

GRAU, G.; FRANCH, X.; MAYOL, E.; AYALA, C.; CARES, C.; HAYA, M.; NAVARRETE, F.; QUER, P. B. C. Risd: A methodology for building i* strategic dependency models. In: *Proceedings of The Seventeenth International Conference on Software Engineering and Knowledge Engineering, SEKE 2005*. [S.l.: s.n.], 2005. p. 259–266.

GRAU, G.; HORKOFF, J.; YU, E.; ABDULHADI, S. *IStar Guide*. March 2008. Web site. [Http://istar.rwth-aachen.de/tiki-index.php?page_ref_id=53](http://istar.rwth-aachen.de/tiki-index.php?page_ref_id=53). Disponível em: <http://istar.rwth-aachen.de/tiki-index.php?page_ref_id=53>. Acesso em: 30 de Março de 2008.

HALSTEAD, M. H. *Elements of Software Science (Operating and programming systems series)*. New York, NY, USA: Elsevier Science Inc., 1977. ISBN 0444002057.

HORKOFF, J. M. *Using i* Models for Evaluation*. Dissertação (Mestrado) — Department of Computer Sciences, University of Toronto, 2006.

HUMPHREY, W. S. *Managing the Software Process*. [S.l.]: Addison Wesley, 1989.

IEEE-1028. *IEEE Standard for Software Reviews - IEEE Std 1028-1997*. New York, NY, USA, Mar 1998.

IEEE-1061. *Standard for a Software Metrics Methodology Std. 1061*. New York, NY, USA, 1992.

IEEE-610. *Standard Glossary of Software Engineering Terminology Std 610.12-190*. New York, NY, USA, 1994.

ISO-15939; ISO/IEC. *International Standard ISO/IEC FDIS 15939, Software Engineering - Software Measurement Process*. Geneva, Switzerland, 2002.

KIM, K.; SHIN, Y.; WU, C. Complexity measures for object-oriented program based on the entropy. In: *Proceedings of the Second Asia Pacific Software Engineering Conference (APSEC)*. Washington, DC: IEEE Computer Society, 1995.

KOTONYA, G.; SOMMERVILLE, I. *Requirements Engineering: Processes and Techniques*. New York, NY, USA: John Wiley & Sons, Inc., 1998. ISBN 0471972088.

LAMSWEERDE, A. van. Requirements engineering in the year 00: a research perspective. In: *ICSE '00: Proceedings of the 22nd international conference on Software engineering*. New York, NY, USA: ACM, 2000. p. 5–19. ISBN 1-58113-206-9.

LAMSWEERDE, A. van. Goal-oriented requirements engineering: A guided tour. In: *RE '01: Proceedings of the 5th IEEE International Symposium on Requirements Engineering*. Washington, DC, USA: IEEE Computer Society, 2001. p. 249. ISBN 0-7695-1125-2.

LEITE, J.; WERNECK, V.; OLIVEIRA, A.; CAPPELLI, C.; CERQUEIRA, A.; CUNHA, H.; GONZALEZ-BAIXAULI, B. Understanding the strategic actor diagram: an exercise of meta modeling. In: *Proceedings of 10th Workshop on Requirements Engineering*. [S.l.: s.n.], 2007.

- LER. *Laboratório de Engenharia de Requisitos*. Junho 2008. Web site.
[Http://www.cin.ufpe.br/~ler](http://www.cin.ufpe.br/~ler). Disponível em: <<http://www.cin.ufpe.br/~ler>>.
 Acesso em: 20 de Junho de 2008.
- LUCENA, M.; SANTOS, E.; SILVA, C.; ALENCAR, F.; SILVA, M. J.; CASTRO, J. Towards a unified metamodel for i*. In: *Proceeding of 2nd IEEE Research Challenges in Information Science - RCIS 2008*. [S.l.: s.n.], 2008.
- MCCABE, T. A. Complexity measure. *IEEE Transactions of Software Engineering, SE*, v. 4, p. 308–320, 1976.
- MCGARRY, J.; CARD, D.; JONES, C.; LAYMAN, B.; CLARK, E.; DEAN, J.; HALL, F. *Practical Software Measurement: Objective Information for Decision Makers*. [S.l.]: Addison Wesley, 2002.
- MISEK-FALKOFF, L. D. A unification of halstead's software science counting rules for programs and english text, and a claim space approach to extensions. *SIGMETRICS Perform. Eval. Rev.*, ACM, New York, NY, USA, v. 11, n. 2, p. 80–114, 1982. ISSN 0163-5999.
- OLIVEIRA, A. P. A.; LEITE, J. C. S. P.; CYSNEIROS, L. M.; CAPPELLI, C. Eliciting multi-agent systems intentionality: from language extended lexicon to i* models. In: *Proceedings of The XXVI International Conference of Chilean Computer Science Society (SCCC 2007)*. Iquique, Chile: [s.n.], 2007.
- PARK, R. E.; GOETHERT, W. B.; FLORAC, W. A. *Goal-Driven Software Measurement – A Guidebook*. Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213, June 1996.
- PRESSMAN, R. S. *Engenharia de Software*. Rio de Janeiro: McGraw-Hill, 2002.
- RAMOS, R. A.; ALENCAR, F.; ARAÚJO, J.; MOREIRA, A.; CASTRO, J.; PENTEADO, R. i* with aspects: Evaluating understandability. In: *Proceeding of 10th Workshop on Requirements Engineering (WER 2007)*. Toronto, Canada: [s.n.], 2007.
- RAMOS, R. A.; ARAÚJO, J.; CASTRO, J. F. B.; MOREIRA, A.; ALENCAR, F.; SILVA, C. Uma abordagem de instanciação de métricas para medir documentos de requisitos orientados a aspectos. In: *Proceeding of III Workshop Brasileiro de Desenvolvimento de Software Orientado a Aspectos (WASP'2006)*. Florianópolis, Santa Catarina - Brasil.: [s.n.], 2006.
- REIS, T. P. C. *REQE – Uma Metodologia para Medição de Qualidade de Aplicações Web na Fase de Requisitos*. Dissertação (Mestrado) — Centro de Informática, Universidade Federal de Pernambuco, Junho 2004.
- RIFKIN, S.; COX, C. *Measurement in Practice*. [S.l.], June 1991.
- SANTOS, E. B. dos. *Uma Proposta de Métricas para Avaliar Modelos i**. Dissertação (Mestrado) — Centro de Informática - Universidade Federal de Pernambuco, Recife, PE, Brasil, Junho 2008.
- SHANNON, C. E. A mathematical theory of communication. *Bell System Technical Journal*, v. 27, p. 379–423, July 1948.

WEBSTER, I.; AMARAL, J.; CYSNEIROS, L. M. A survey of good practices and misuses for modelling with i^* . In: *Proceedings of VIII Workshop on Requirements Engineering (WER 2005)*. Porto, Portugal: [s.n.], 2005.

YOU, Z. *Using Meta-Model-Driven Views to Address Scalability in i^* Models*. Dissertação (Mestrado) — Department of Computer Sciences, University of Toronto, 2004.

YU, E. *Modelling Strategic Relationships for Business Process Reengineering*. Tese (Doutorado) — Dept. of Computer Science, University of Toronto, 1995.

YU, E. S. K. Towards modelling and reasoning support for early-phase requirements engineering. In: *Proceedings of the 3rd IEEE Int. Symp. on Requirements Engineering (RE 97)*. Annapolis, MD, USA: IEEE Computer Society, 1997. p. 226–235. Disponível em: <citeseer.ist.psu.edu/article/yu97towards.html>.

YU, Y.; LEITE, J.; MYLOPOULOS, J. From goals to aspects: discovering aspects from requirements goal models. In: *Proceedings of 12th IEEE International Requirements Engineering Conference, 2004*. Kyoto, Japan: IEEE Computer Society, 2004. p. 38–47.

ZAVE, P. Classification of research efforts in requirements engineering. In: *ACM Computing Surveys*. New York, NY, USA: ACM Press, 1997. v. 29, n. 4, p. 315–321.

Tabela 38: Internal Actor

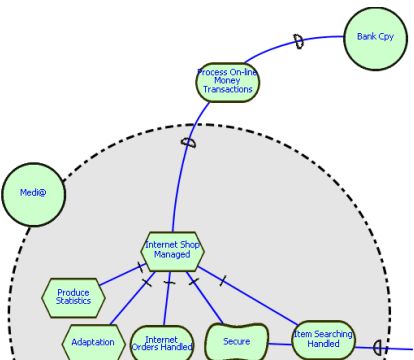
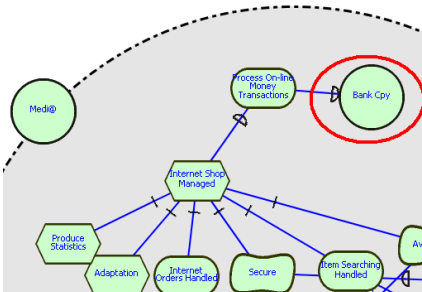
02 - Internal Actor	Do not include an Actor within another Actor
Details	Actors are active and autonomous entities that should be modeled separately. Only intentional elements can be inside an actor (eg. Goal, Softgoal, Task or Resource)
Questions	Is there more than one actor inside the same boundary?
Correct	Wrong
	

Tabela 39: Actor Boundary

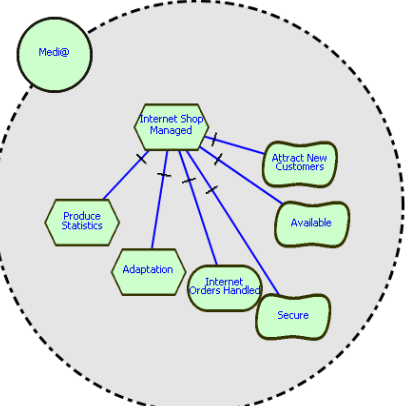
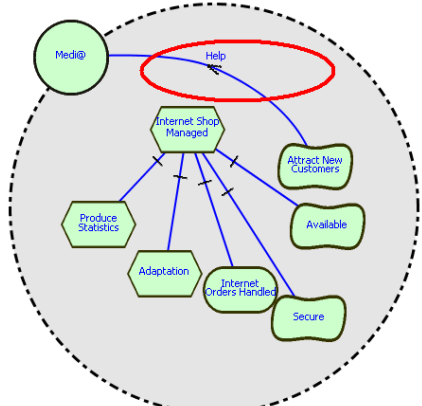
03 - Actor Boundary	Internal element connection to Actor boundary
Details	Do not connect internal elements (goals, tasks, etc.) to the actor boundary
Questions	Is there any internal element connected to the actor boundary?
Correct	Wrong
	

Tabela 40: Actor Dependency

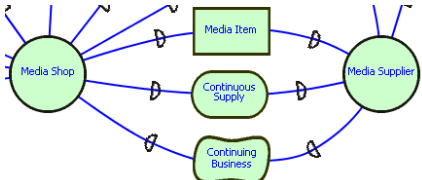
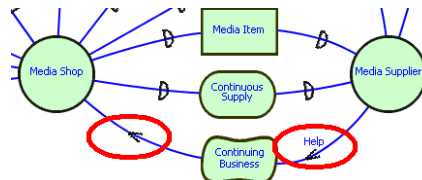
04 - Actor Dependency	Use only the dependency symbol (D) to denote a dependency link
Details	Do not use other types of links (Means-end, Contribution, Decomposition) to denote dependency links.
Questions	Is there any Actor Dependency which does not use the Dependency symbol?
Correct	Wrong
	

Tabela 41: Softgoal

05 - Softgoal	Softgoals are distinct from Goals
Details	Softgoals are special Goals that have no clear cut achievement state. There are different levels of satisfaction (e.g., Satisfied, Partially Satisfied, Denied, etc.) They are used to capture quality issues or constraints such as: security, performance, etc.
Questions	Is there any Goal with adverbs or adjectives in their label? The condition expressed in label could be partially satisfied or not completely achieved?
Correct	Wrong
	

Tabela 42: Goal Refinement

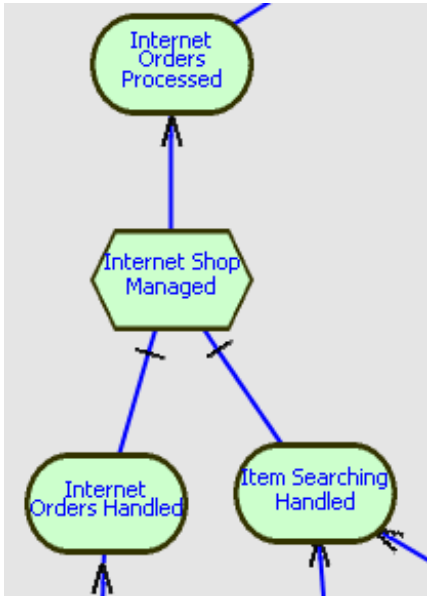
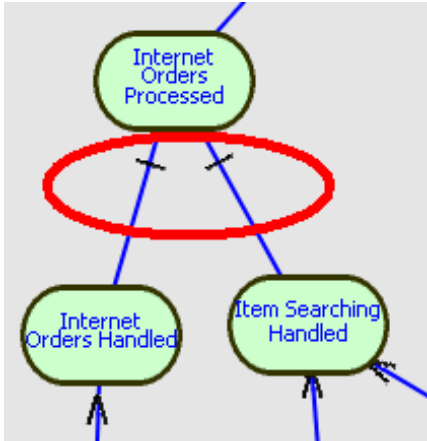
06 - Goal Refinement	Linking of Goals in SR models
Details	In SR models: - Goals cannot be directly linked to other Goals - Goals cannot be present at both ends of Decomposition, Contribution, or Means-Ends links
Questions	Are there Goals linked to other Goals?
Correct	Wrong
	

Tabela 43: Internal Dependency Link

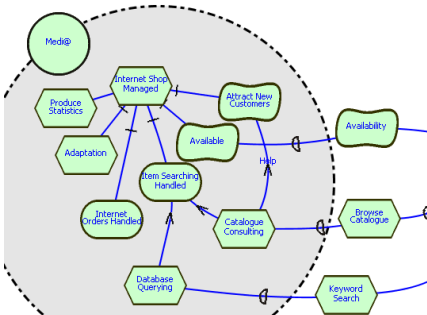
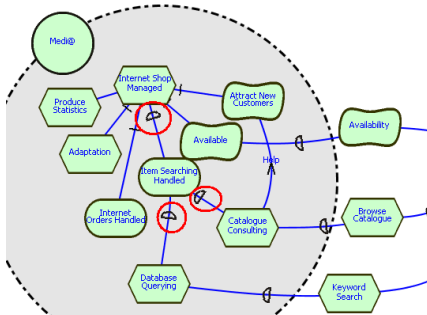
07 - Internal Dependency Link	Do not use dependency links inside an actor
Details	Dependency links cannot be used to connect two intentional elements inside the same actor. It can be used to connect an internal element to a dependum, but not to connect two internal elements.
Questions	Is there any internal element that is connected to an other internal element using a dependency link?
Correct	Wrong
	

Tabela 44: External Dependency Link

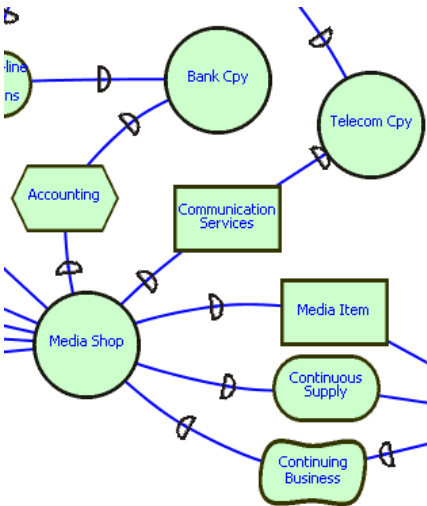
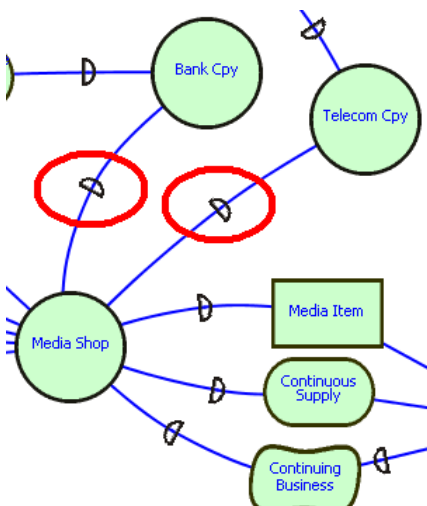
08 - External Dependency Link	Do not use a dependency link between two actors without showing the dependum
Details	The Dependency link should contain a Dependum. Extending the Dependency Link from the Depender to the Dependee without showing the Dependum does not convey what the dependency is about.
Correct	Wrong
	

Tabela 45: Internal Contribution Link

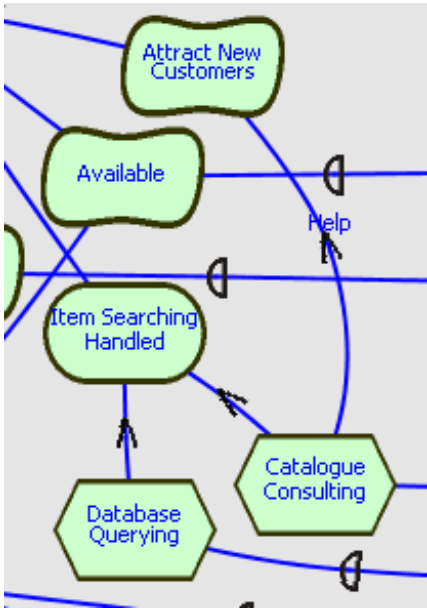
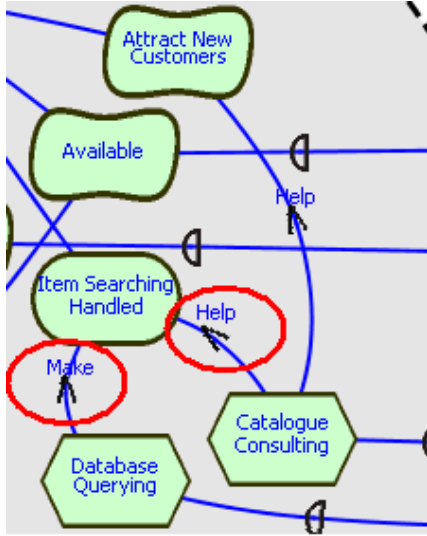
09 - Internal Contribution Link	A contribution link should only be used to connect from an intentional element (of any type) to a softgoal
Details	Contribution links are not allowed from any element to a goal, only to softgoals
Questions	Is there any Goal receiving Contribution links?
Correct	Wrong
	

Tabela 48: External Means-End Link

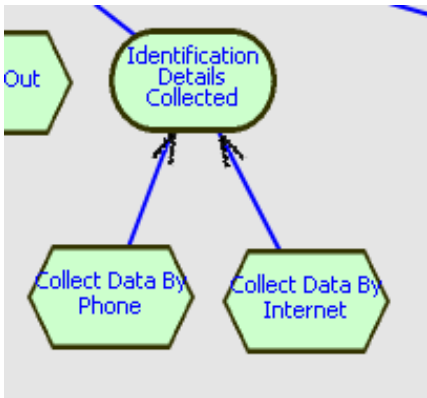
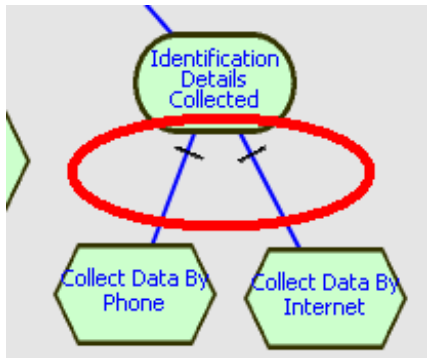
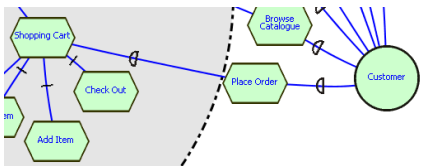
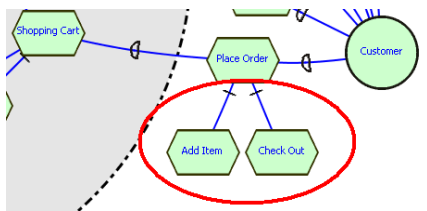
12 - External Means-End Link	Do not extend Means-Ends link beyond the boundaries of actors
Details	Means-Ends Link need to be drawn within an actor and not between actors
Questions	Is there any Means-Ends link outside actor's boundary? Is there any Means-Ends link between an internal element and an element outside actor's boundary?
:browse confirm wa Correct	Wrong
	

Tabela 49: External Decomposition Link

13 - External Decomposition Link	Do not extend Decomposition links beyond the boundaries of actors.
Details	Decomposition Links need be drawn within an actor, and not between actors
Questions	Is there any Decomposition link outside actor's boundary? Is there any Decomposition link between a Task inside actor's boundary and an external element?
Correct	Wrong
	

APÊNDICE B – Coleta Automática de Métricas

Durante a definição e aplicação das métricas a exemplos verificou-se que é custoso e cansativo coletar os dados das métricas de forma manual. Com a motivação de simplificar a coleta dos dados está sendo propostas uma ferramenta para coleta automática dos dados.

Como a ferramenta não faz parte do escopo dessa dissertação, será apresentada a seguir apenas uma proposta da ferramenta com uma descrição de como a ferramenta deveria funcionar e o que é esperado que ela produza como resultados.

Para demonstrar a importância da ferramenta na organização foi utilizando a técnica i*. Como os modelos são muito grandes por questão de apresentação serão descritos apenas dois modelos. O primeiro é um modelo SR que apresenta a organização e as razões pelas quais a coleta automática de dados pode ter vantagens sobre outras técnicas. O segundo modelo apresenta é um modelo SR que apresenta as razões pelas que motivam a adoção de algumas decisões quanto as possíveis soluções que um coletor automático de métricas poderia adotar.

A figura 15 apresenta um modelo SR onde são descritas as razões pelas quais uma ferramenta pode ser importante para a coleta de métricas. São apresentados três atores que fazer parte da organização, *Engenheiro de Requisitos*, *Time de Qualidade* e *Desenvolvedor*, e o ator *Coletor Automático* que representa a ferramenta em si.

O ator *Desenvolvedor* é responsável por desenvolver a ferramenta e espera ter os modelos e documentos necessários para que possa desenvolver o software. O ator *Engenheiro de Requisitos* tem a preocupação de produzir documentos que sejam fáceis de entender e manter, e para isso vai depender do ator *Time de Qualidade* que vai oferecer uma avaliação imparcial e apontar possíveis melhoras nos documentos.

Além disso o ator *Time de Qualidade* está envolvido com restrições nas quais a avaliação deve ser imparcial, objetiva e rápida, representadas respectivamente pelos *softgoals*:

Imparcialidade, Objetividade e Rapidez. Entre as opções para a avaliação o *Time de Qualidade* tem a inspeção e o uso de métricas representados respectivamente pelas rotinas *Inspecionar* e *Usar Métricas*. Entre as possibilidades para o uso de métricas estão a coleta manual e a automática, representadas pelas rotinas *Coletar Manualmente* e *Coletar Automaticamente*. De acordo com a análise sobre os *softgoals* é possível verificar que *Inspecionar* e *Coletar Manualmente* são equivalentes quanto a imparcialidade, objetividade e rapidez. Isso se deve ao fato de ambas alternativas precisarem da intervenção humana o que faz com que os resultados dependam dos conhecimentos do avaliador para serem coletados, tornando o processo incerto e lento se comparado a coleta automática. Assim repassar a coleta das métricas para o ator *Coletor Automático* se torna uma opção mais vantajosa se comparado as outras.

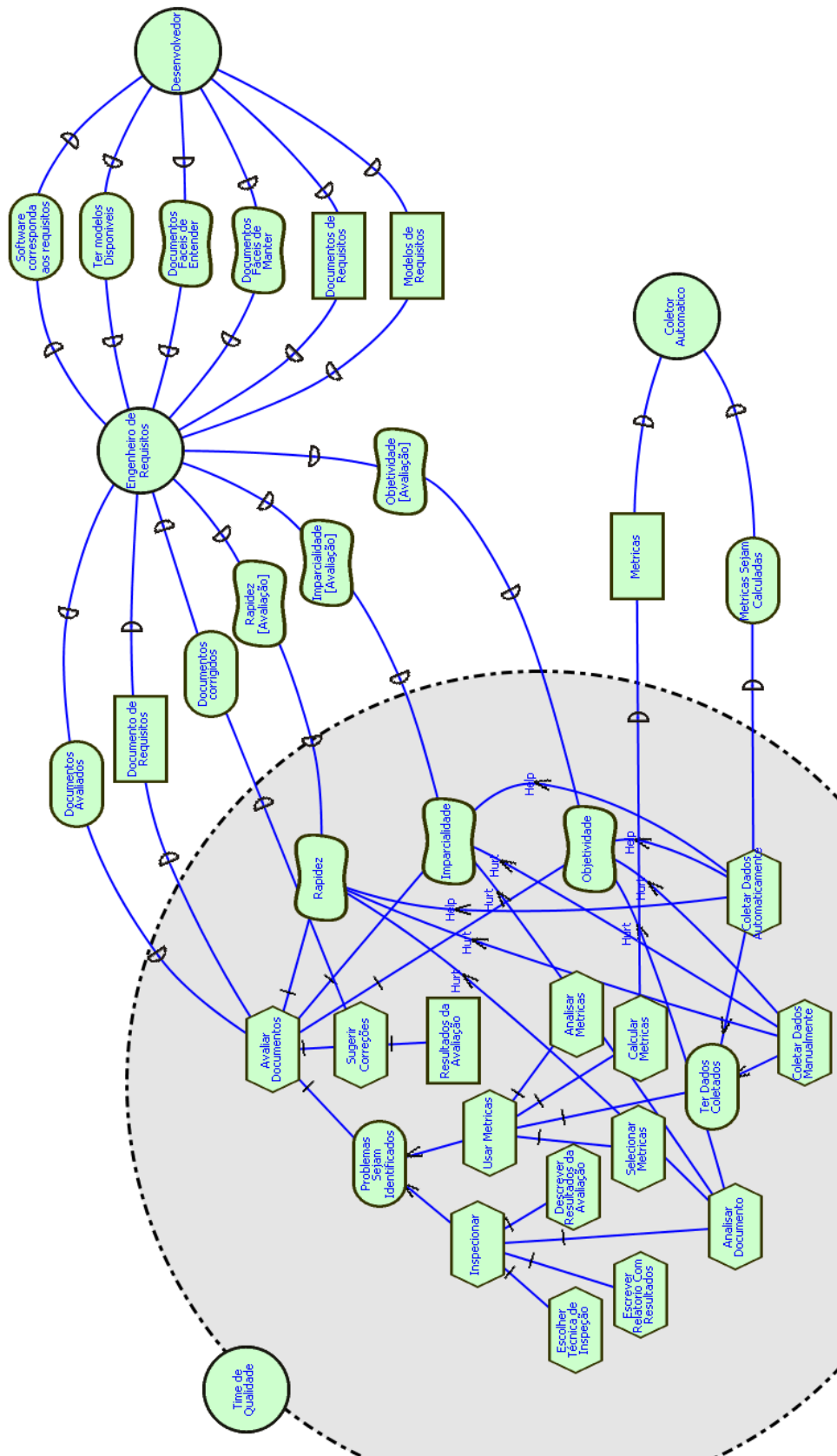


Figura 15: Modelo SR do Time de Qualidade

A figura 16 apresenta um modelos SR com a descrição de como o ator *Coletor Automático* vai satisfazer as necessidades de coleta automática. Para que a meta *Métricas Sejam Obtidas* seja satisfeita é descrita a rotina *Obter Metricas* que envolve a meta *Dados sejam Obtidos*, as tarefas *Calcular Metricas* e *Gerar Relatório com Métricas*, e os *softgoals* *Reusável*, *Portável* e *Desempenho*. Os *softgoals* descritos são restrições que permitem decidir em torno das opções para o coletor de métricas. Entre as alternativas para obter os dados estão a de coletados depois da modelagem e durante a modelagem, descritas pelas rotinas *Obter Dados Pós Modelagem* e *Obter Dados Durante Modelagem*. *Obter Dados Pós Modelagem* envolve o processamento dos documentos gerados por uma ferramenta de modelagem qualquer, a coleta dos dados, e possui três opções para a contagem de erros, uma baseada no uso de regras, outra baseado no uso de máquinas de estado e uma terceira baseada em transformação de modelos. A *Checagem baseada em Regras* corresponde a criar regras para verificar os erros de um tipo específico de documento ela apresenta as desvantagens de prejudicar a portabilidade e reusabilidade por se aplicar a apenas um tipo de documento e precisariam ser reescritas para outros formatos de arquivos. A segunda opção que a *Checagem por Máquina de Estado* é uma opção que favorece o *Desempenho* por resultar em mecanismo mais rápido comumente usado em compiladores, por outro lado ela apresenta as mesmas limitações do uso de regras por se aplicar à um tipo específico de documento. A outra opção é usar *Transformação de Modelos* para checar erros, essa opção permite fazer a contagem de erros em diferentes tipos de arquivos. Para isso os modelos escritos em diferentes formatos são transformados para um formato intermediário onde é feita checagem. Por operar sobre um modelo intermediário essa opção permite ao coletor ser *Reusável* e *Portável*. No entanto, essa perde em *Desempenho* pois é necessário um passo intermediário que não existe nas outras alternativas.

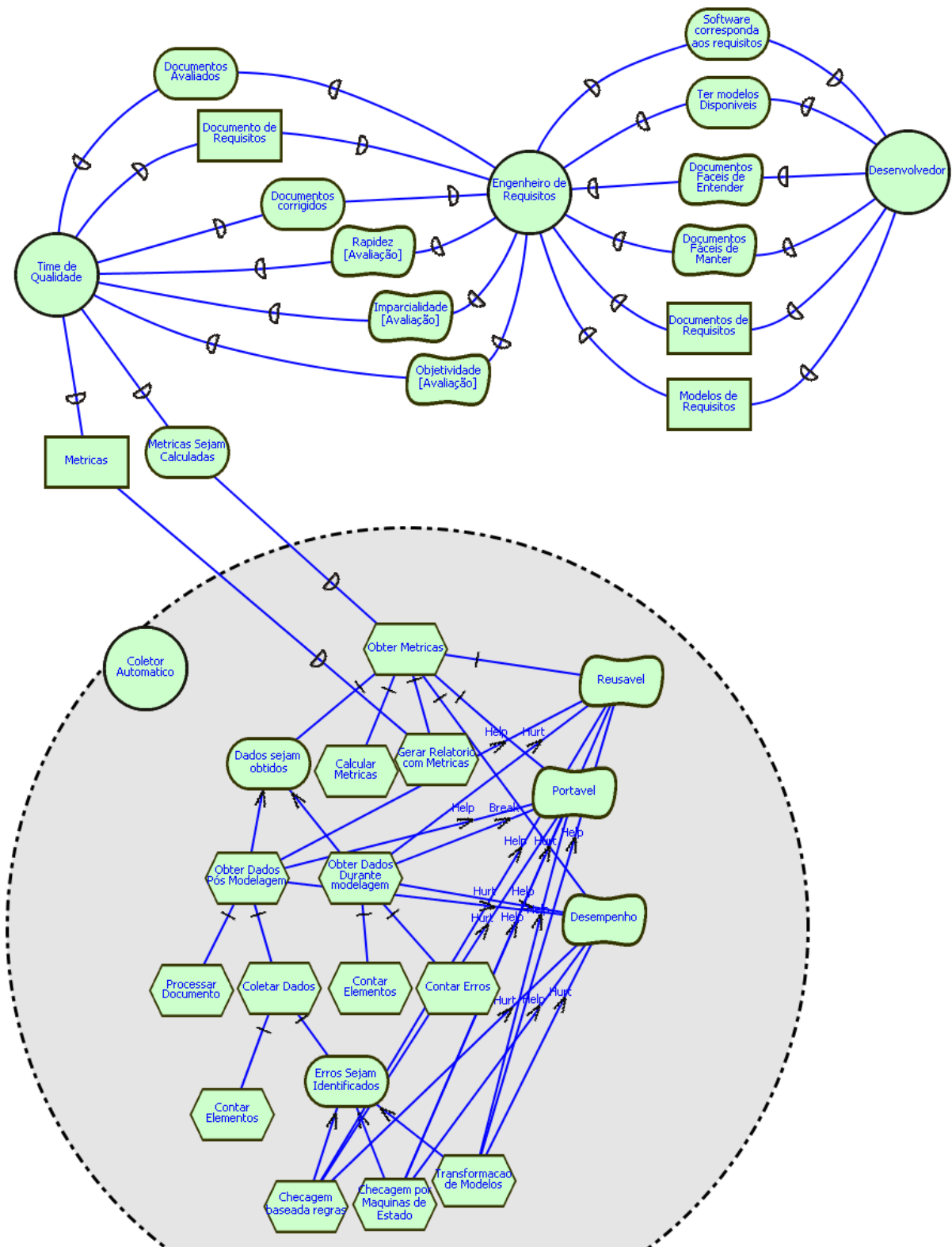


Figura 16: Modelo SR do Coletor Automático de Métricas

Com base nas análises apresentadas a sugestão é que o coletor automático obtenha as métricas a partir da análise de arquivos com os modelos. Essa coleta preferencialmente deve ser feita com o arquivo pós-modelagem, pois devido a grande variedade de ferramentas é mais vantajoso reusar um coletor que interpreta os arquivos produzidos que construir um coletor para cada ferramenta. Além disso, algumas das ferramentas não têm código aberto o que impede a possibilidade de criar um coletor que funcione durante a modelagem. Como forma de contar os erros é recomendado o uso de transformação de modelos, que apesar de perder em desempenho, permite reusar o coletor em diferentes tipos de arquivos.

ANEXO A – Templates para documentação de Métricas e Dados (IEEE-1061, 1992)

Tabela 50: Tabela com a estrutura para documentação de Métrica do padrão IEEE-1061 (1992)

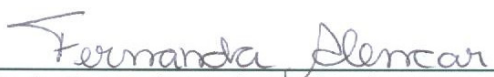
Item	Descrição
Nome	Nome sugerido a métrica
Custo	Custo de usar a métrica
Benefícios	Benefícios em usar a métrica
Impacto	Indicação da possibilidade da métrica ser usada para alterar ou interromper um projeto
Valor Alvo	Valor numérico da métrica que se procura atingir para encontrar o fator de qualidade. Inclui o valor crítico e a faixa de valores da métrica
Fator de Qualidade	Fator de qualidade que está relacionado a esta métrica
Ferramentas	Ferramentas de Software ou hardware que são usadas para obter, armazenar, computar a métrica ou analisar os resultados
Aplicação	Descrição de como a métrica é usada e quais suas áreas de aplicação
Dados	Valores necessários para computar o valor da métrica
Cálculos	Explicação dos passos envolvidos no cálculo da métrica
Interpretação	Interpretação dos resultados da métrica
Considerações	Considerações sobre a adequação da métrica para esta aplicação
Treinamento requerido	Treinamento requerido para implementar ou para usar a métrica
Exemplo	Um exemplo da aplicação da métrica
Histórico de Validação	Nome de projetos que tem usado esta métrica e os critérios de satisfação da métrica.
Referências	Referências, como uma lista de projetos e detalhes de projeto, dando detalhes para futuro entendimento ou implementação de métrica.

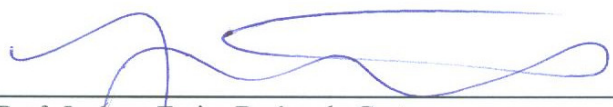
Tabela 51: Tabela com a estrutura para documentação de Dados do padrão IEEE-1061 (1992)

Item	Descrição
Nome	Nome sugerido para o dado
Métricas	Métricas que são associadas com o dado
Definição	Definição do dado
Fonte	Local de onde o dado se origina
Coletor	Entidade responsável por coletar o dado
Tempo	Número de vezes no ciclo de vida em que os dados são coletados (Alguns dados são coletados mais de uma vez)
Procedimentos	Metodologia (automática ou manual) usada para coletar os dados
Armazenamento	Local onde os dados são armazenados
Representação	Maneira em que os dados são representados, ou seja, precisão e formato (inteiro, binário, adimensional, etc.)
Amostra	Método usado par selecionar os dados que serão coletados e o percentual de disponibilidade dos dados a serem coletados
Verificação	Maneira na qual os dados são coletados para checar erros na coleta
Alternativas	Métodos alternativos para coletar os dados (se houver)
Integridade	Pessoas ou organizações autorizadas a alterar os dados e em que condições

Dissertação de Mestrado apresentada por **Emanuel Batista dos Santos** Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título, “**Uma Proposta de Métricas para Avaliar Modelos i***”, orientada pelo **Prof. Jaelson Freire Brelaz de Castro** e aprovada pela Banca Examinadora formada pelos professores:


Prof. Alexandre Marcos Lins de Vasconcelos
Centro de Informática / UFPE


Profa. Fernanda Maria Ribeiro de Alencar
Departamento de Eletrônica e Sistemas / UFPE


Prof. Jaelson Freire Brelaz de Castro
Centro de Informática / UFPE

Visto e permitida a impressão.
Recife, 26 de junho de 2008.


Prof. FRANCISCO DE ASSIS TENÓRIO DE CARVALHO
Coordenador da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.