UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E INFORMÁTICA INDUSTRIAL

PAULO JOSÉ DANTAS NOVAES

MÉTODO E LINGUAGEM PARA MODELAGEM GRÁFICA DE REQUISITOS DE *SOFTWARE* E SISTEMAS

DISSERTAÇÃO DE MESTRADO

CURITIBA
2019

PAULO JOSÉ DANTAS NOVAES

MÉTODO E LINGUAGEM PARA MODELAGEM GRÁFICA DE REQUISITOS DE *SOFTWARE* E SISTEMAS

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial da Universidade Tecnológica Federal do Paraná, como requisito parcial para a obtenção do grau de Mestre em Ciências.

Orientador: Prof. Dr. Paulo Cézar Stadzisz

Co-Orientador: Prof. Dr. Jean Marcelo Simão

CURITIBA

2019

Dados Internacionais de Catalogação na Publicação

Novaes, Paulo José Dantas
Método e linguagem para modelagem gráfica de requisitos de software e sistemas [recurso eletrônico] / Paulo José Dantas Novaes.-- 2019.
1 arquivo texto (347 f.): PDF; 12,5 MB.

Modo de acesso: World Wide Web
Título extraído da tela de título (visualizado em 20 maio 2019)
Texto em português com resumo em inglês
Dissertação (Mestrado) - Universidade Tecnológica Federal do Paraná. Programa de Pós-graduação em Engenharia Elétrica e Informática Industrial, Curitiba, 2019
Bibliografia: f. 223-234

1. Engenharia elétrica - Dissertações. 2. Engenharia de requisitos. 3. Engenharia de software. 4. Modelagem (Computação). 5. Modelagem de dados. 6. Gerenciamento de configurações de software. 7. Modelagem de sistemas. I. Stadzisz, Paulo Cézar. II. Simão, Jean Marcelo. III. Universidade Tecnológica Federal do Paraná. Programa de Pós-graduação em Engenharia Elétrica e Informática Industrial. IV. Título.

CDD: Ed. 23 – 621.3

Biblioteca Central da UTFPR, Câmpus Curitiba
Bibliotecário: Adriano Lopes CRB-9/1429

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Diretoria de Pesquisa e Pós-Graduação

CAMPUS CURITIBA

TERMO DE APROVAÇÃO DE DISSERTAÇÃO Nº 828

A Dissertação de Mestrado intitulada **"Método e Linguagem Para Modelagem Gráfica de Requisitos de Software e Sistemas"** defendida em sessão pública pelo(a) candidato(a) **Paulo José Dantas Novaes**, no dia **09 de maio de 2019**, foi julgada para a obtenção do título de Mestre em Ciências, área de concentração **Engenharia De Computação**, e aprovada em sua forma final, pelo Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial.

BANCA EXAMINADORA:

Prof(a). Dr(a). Paulo Cézar Stadzisz - Presidente (UTFPR)

Prof(a). Dr(a). Robson Ribeiro Linhares - (UTFPR)

Prof(a). Dr(a). Andrey Ricardo Pimentel - (UFPR)

A via original deste documento encontra-se arquivada na Secretaria do Programa, contendo a assinatura da Coordenação após a entrega da versão corrigida do trabalho.

Curitiba, 09 de maio de 2019.

DEDICATÓRIA

Para meus pais José Paulo e Maria Lurdes,

pelo amor e por sempre acreditarem que eu podia conseguir.

Para Bárbara, pelo apoio,

e por ser o amor da minha vida.

Para Sophia, a minha menina,

que a cada dia me dá mais orgulho como filha.

AGRADECIMENTOS

Ao Prof. Dr. Paulo Cézar Stadzisz pela orientação e ao Prof. Dr. Jean Marcelo Simão pela co-orientação. Este trabalho só se tornou possível com os inestimáveis ensinamentos e conselhos destes excelentes pesquisadores e profissionais.

Aos membros da banca examinadora, Prof. Dr. Andrey Ricardo Pimentel, Prof. Dr. Robson Ribeiro Linhares e Prof. Dr. Paulo Cézar Stadzisz pela paciência de ler e avaliar um trabalho considerado denso e extenso em conteúdo. Suas pertinentes e valiosas observações contribuíram muito para o enriquecimento deste trabalho.

Ao Prof. Dr. Gilson Yukio Sato um agradecimento especial pelos conselhos sobre pesquisa e pós-graduação e por sua grande influência em minha formação pessoal e profissional.

Ao Prof. Dr. Hugo Vieira Neto pela excelente disciplina sobre Metodologia Científica, além dos conselhos e ensinamentos sobre processamento de imagens, usados pontualmente nesta dissertação.

Aos colegas e amigos que fiz durante a pós-graduação, pelo companheirismo e amizade e pelas contribuições realizadas.

À UTFPR, minha *Alma-Mater*, pela infraestrutura física e pelo suporte a todas as atividades acadêmicas realizadas.

À CAPES/CNPQ, pelo apoio financeiro fornecido por meio de bolsa de pesquisa e por meio do PROAP (Programa de Apoio à Pós-Graduação), para que fosse possível publicar o artigo relacionado a este trabalho em congresso da área.

Por fim, a todos aqueles que contribuíram direta ou indiretamente para este trabalho, o meu muito obrigado!

RESUMO

NOVAES, Paulo José Dantas. **Método e Linguagem para Modelagem Gráfica de Requisitos de *Software* e Sistemas**, 2019. 348f. Dissertação de Mestrado – Programa de Pós-graduação em Engenharia Elétrica e Informática Industrial, Universidade Tecnológica Federal do Paraná. Curitiba, 2019.

Esta dissertação propõe um método e uma linguagem de modelagem gráfica de requisitos de *software* e sistemas sob o nome de RIMON (*Requirements and Interdependencies MOdeling Notation*). Esta linguagem permite representar requisitos e suas interdependências de forma sistemática, precisa e expressiva, visando contribuir para a melhoria da qualidade da especificação de requisitos de *softwares* e sistemas. Originada a partir de conceitos da abordagem RON (Requisitos Orientados a Notificações), a RIMON foi criada para ser visualmente atrativa em possíveis usos comerciais. Essa atratividade decorre de um desenvolvimento fundamentado na teoria de física das notações (*Physics of Notations – PoN*), que fornece princípios destinados a produzir notações visuais eficientes em termos de comunicação. Adicionalmente, a linguagem é definida por meio de uma sintaxe abstrata (metamodelo) e de uma sintaxe concreta (sintaxe visual) complementada por um mapeamento semântico preciso. Em relação ao método proposto, este consiste em um ciclo iterativo de atividades para identificação e análise dos dados de entrada e modelagem gráfica dos requisitos e suas interdependências. A RIMON oferece características tais como suporte a modelagem de requisitos funcionais e não-funcionais, entidades, atributos, pré-condições, pós-condições e identificação de conflitos. Como diferencial, a modelagem de pré-condições possibilita representar restrições relativas a requisitos não-funcionais por meio de atributos contendo condições lógico-matemáticas. Por fim, este trabalho apresenta três experimentos de modelagem usando RIMON, tanto de requisitos de sistemas (extraídos da literatura) quanto de requisitos de *software* (projeto real), destinados a demonstrar e verificar as capacidades da linguagem.

Palavras-chave: engenharia de requisitos; engenharia de sistemas; engenharia de *software*; linguagem de modelagem de requisitos; método de modelagem de requisitos; modelagem de requisitos e interdependências; modelagem gráfica de requisitos; modelo gráfico de requisitos; notação de modelagem de requisitos; notação visual de requisitos.

ABSTRACT

NOVAES, Paulo José Dantas. **Method and Language for Graphical Modeling of Requirements for Software and Systems**. 2019. 348p. M.Sc. Thesis. Graduation Program in Electrical Engineering and Computer Engineering (CPGEI), Federal University of Technology. Curitiba, 2019.

This dissertation proposes a method and a language for graphic modeling of software and systems requirements, under the name of RIMON (Requirements and Interdependencies MOdeling Notation). This language allows to represent requirements and their interdependencies in a systematic, precise and expressive way, aiming to contribute to the quality improvement of software and systems requirements specifications. Sourced in concepts of RON (*Requirements Oriented to Notifications*) approach, RIMON was designed to be visually attractive in its possible commercial usage. This attractiveness results from a development grounded in the Physics of Notations (PoN), which provides principles for creating efficient visual notations in terms of communication. In addition, the language is defined by an abstract syntax (metamodel) and a concrete syntax (visual syntax), complemented by a precise semantic mapping. Regarding the proposed method, it includes a set of steps for identification and analysis of input data, graphical requirements and interdependencies modeling. RIMON offers features such as support for modeling functional and non-functional requirements, entities, attributes, preconditions, postconditions and conflict identification. As a notable characteristic, the modeling of preconditions allows the representation of constraints related to non-functional requirements by means of attributes containing logical-mathematical conditions. At last, this work presents three modeling experiments using RIMON, including system requirements (extracted from the literature) and software requirements (real project) environments, designed to demonstrate and verify the capabilities of the language.

Keywords: graphical requirements model; graphical requirements modeling; requirements engineering; requirements and interdependencies modeling; requirements modeling language; requirements modeling method; requirements modeling notation; requirements visual notation; software engineering; systems engineering;

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS, ACRÔNIMOS E ABREVIATURAS

Sigla	Em inglês / original	Em português / tradução
ACRE	*Approach to Context-Based Requirements Engineering*	Abordagem de Engenharia de Requisitos baseada em contexto
ACM	*Association for Computing Machinery*	Associação para máquinas de computação (sic)
AGORA	*Attributed Goal Oriented Requirements Analysis*	Análise de requisitos orientada a objetivos com atributos
AHP	*Analytic Hierarchy Process*	Processo de hierarquia analítica
AMORE	*Advanced Multimedia Organizer for Requirements Elicitation*	Organizador multimídia avançado para elicitação de requisitos
AoUCM	*Aspects oriented Use Case Maps*	Mapas de caso de uso orientados a aspectos
AoURN	*Aspects oriented User Requirements Notation*	Notação de requisitos de usuários orientada a aspectos
ATHENA	*Advanced Technologies for Interoperability of Heterogeneous Enterprise Networks and their Applications*	Tecnologias avançadas para interoperabilidade de redes empresariais heterogêneas e suas aplicações
ASM	*Abstract System Machines*	Máquinas de sistema abstratas
AWARE	*Analysis of Web Applications Requirements Engineering*	Engenharia de requisitos e análise de aplicações *web*
BABOK	Business Analysis Body of Knowledge	Guia para o corpo de conhecimento em análise de negócios.
BDD	*Block Definition Diagram*	Diagrama de definição em blocos
BPD	*Business Process Diagram*	Diagrama de processo de negócio
BPM	*Business Process Modeling*	Modelagem de processos de negócio
BPMN	*Business Process Modeling Notation*	Notação de modelagem de processos de negócio
CAPES	-	Coordenação de Aperfeiçoamento de Pessoal de Nível Superior
CBSOFT	-	Congresso Brasileiro de *Software*
CCSE	Computing and Communications Systems Engineering	Engenharia de sistemas de computação e comunicações
CIBSE	-	Congresso Ibero-Americano de Engenharia de *Software*
CDD	*Concern Driven Development*	Desenvolvimento orientado a preocupações (interesses)
CI-nets	*Conditional Importance Networks*	Redes de importância condicional
CIBSE	-	Congresso Ibero-Americano em Engenharia de *Software*
CN	*Customer Need*	Necessidade do Cliente
COCA-MDA	*Context Oriented Component Based Application for Model Driven Architecture*	Aplicação baseada em componentes e contexto para arquitetura dirigida a modelos

Sigla	Em inglês / original	Em português / tradução
CP	*Customer Problem*	Problema do cliente
CORAMOD	*Checklist-Oriented Requirements Analysis MODelling*	Modelagem e análise de requisitos orientada a listas de checagem
DHARMA	*Discovering Hybrid Architectures by Modelling Actors*	Descoberta de arquiteturas híbridas pela modelagem de atores
DON	*Development Oriented to Notifications*	Desenvolvimento Orientado a Notificações
e.g.	*Exempli gratia*	por exemplo
EBSE	*Empirical-Based Software Engineering*	Engenharia de *Software* baseada em experimentos (empirismo)
EA	*Enterprise Architect Tool*	Ferramenta de arquitetura empresarial
ER	-	Engenharia de Requisitos
ES	-	Engenharia de Sistemas
FAUST	*Formal Analysis Using Specification Tools*	Análise formal utilizando ferramentas de especificação
FBE	*Fact Base Element*	Elemento da base de fatos
FR	*Functional Requirement*	Requisito funcional
GAM	*Goal Argumentation Model*	Modelo de argumentação orientado a objetivos
GBRAM	*Goal-Based Requirements Analysis Method*	Método de análise de requisitos orientado a objetivos
GATO	*Goal to Architecture TOol*	Ferramenta de objetivos para arquitetura
GOOD	*Goal Object Oriented Development*	Desenvolvimento orientado a objetos e objetivos
GORE	*Goal Oriented Requirements Engineering*	Engenharia de Requisitos orientada a objetivos
GORO	*Gore Oriented Requirements Ontology*	Ontologia para requisitos orientados a objetivos
GRL	*Goal Oriented Requirements Language*	Linguagem de requisitos orientada a objetivos
GRM	*Graphical Requirements Modeling*	Modelagem Gráfica de Requisitos
HUCRE	*Human-Centred Requirements Engineering*	Engenharia de Requisitos centrada em humanos
HTN	*Hierarchical Tasks Network*	Redes de tarefas hierárquicas
IBD	*Internal Block Diagram*	Diagrama de blocos internos
IBICT	-	Instituto Brasileiro de Informação em Ciência e Tecnologia
ITU-T	*International Telecommunications Union*	União Internacional de Telecomunicações
ICSE	*International Conference on Software Engineering*	Conferência Internacional em Engenharia de *Software*
ICSEng	*International Conference on Systems Engineering*	Conferência Internacional em Engenharia de Sistemas

Sigla	Em inglês / original	Em português / tradução
IEEE	*Institute of Electrical and Electronics Engineers*	Instituto de Engenheiros Eletricistas e Eletrônicos
i.e.	*Id est*	isto é, ou seja
IHM	-	Interface Humano Máquina
I*	*I star model*	Modelo i estrela (sic)
INCOSE	*International Council on Systems Engineering*	Conselho Internacional de Engenharia de Sistemas
IRD	*Interelement Requirements Diagram*	Diagrama de inter-elementos de requisitos (sic)
IREB	*International Requirements Engineering Board*	Conselho Internacional de Engenharia de Requisitos
JGOOSE	*Java Goal into Object Oriented Standard Extension*	Objetivos Java em Extensão padrão de Orientação a Objetos
KAOS	*Knowledge Acquisition in autOmated System or Keep All Objects Satisfied*	Aquisição automática de conhecimento em sistemas ou mantenha todos os objetos satisfeitos
KBRE	*Knowledge Based Requirements Engineering*	Engenharia de requisitos baseada em conhecimento
LQN	*Layered Queueing Networks*	Redes enfileiradas por camadas
MBRE	*Model-Based Requirements Engineering*	Engenharia de Requisitos baseada em modelos
MBSE	*Model-Based Systems Engineering*	Engenharia de sistemas baseada em modelos
MDA	*Model Driven Architecture*	Arquitetura direcionada a modelos
MDEReq	*Model Driven Engineering for Requirements Management*	Engenharia direcionada a modelos para gerenciamento de requisitos
MRC	*Multimedia Reusable Components*	Componentes multimídia reutilizáveis
NFR	*Non-Functional Requirement*	Requisito não-funcional
NOM	*Notification Oriented Modeling*	Modelagem Orientada a Notificações
NOP	*Notification Oriented Paradigm*	Paradigma Orientado a Notificações
OCL	*Object Constraint Language*	Linguagem de restrição de objetos
OMG	*Object Management Group*	Grupo de gerenciamento de objetos
OMT	*Object Modeling Technique*	Técnica de modelagem de objetos
OOSE	*Object Oriented Software Engineering*	Engenharia de *Software* Orientada a Objetos
OPD	*Object Process Diagram*	Diagrama de processos e objetos
OPM	*Object Process Methodology*	Metodologia orientada a processos e objetos
PD	-	Paradigma Declarativo
PD	*Parametric Diagram (*SysML*)*	Diagrama paramétrico (SysML)

Sigla	Em inglês / original	Em português / tradução
PI	-	Paradigma Imperativo
PDDL	*Planning Domain Definition Language*	Linguagem de definição para planejamento de domínios
PF	-	Paradigma Funcional
PF	*Problem Frames*	Quadros de problemas
PICOC	*(Population, Intervention, Comparison, Outcome, Context*)	(População, Intervenção, Comparação, Resultado, Contexto)
PL	-	Paradigma Lógico
PP	-	Paradigma Procedimental
PoN	*Physics of Notations*	Física da Notações
PON	*Notification Oriented Paradigm*	Paradigma Orientado a Notificações
POO	-	Paradigma Orientado a Objetos
pUML	*Precise Unified Modeling Language*	Linguagem de modelagem unificada precisa
RCAT	*Requirements Capture Tool*	Ferramenta de captura de requisitos
RD	*Requirements Diagram*	Diagrama de Requisitos
RDG	*Requirements Dependencies Graph*	Grafo de dependências de requisitos
RDN	*Requirements Dependency Network*	Rede de dependências de requisitos
RE	*International Requirements Engineering Conference*	Conferência Internacional sobre engenharia de requisitos
RE	*Requirements Engineering*	Engenharia de requisitos
REBoK	*Requirements Engineering Body of Knowledge*	Guia para o corpo de conhecimento em engenharia de requisitos
REQB	*Requirements Engineering Qualification Board*	Quadro (grupo) de qualificação em engenharia de requisitos.
RIG	*Requirements Interaction Graph*	Grafo de interação de requisitos
ReCVisu	*Requirements Clustering Visualization Tool*	Ferramenta para visualização de requisitos em *clusters*
RON	*Requirements Oriented to Notifications*	Requisitos Orientados a Notificações
RPL	*Requirements Pattern Language*	Linguagem de padrões de requisitos
RTM	*Requirements Traceability Matrix*	Matriz de rastreabilidade de requisitos
RUP	*Rational Unified Process*	Processo unificado da Rational
SBC	-	Sociedade Brasileira de Computação
SBES	-	Simpósio Brasileiro de Engenharia de *Software*
SCR	*Software Cost Reduction method*	Redução de custos de *software (método)*

Sigla	Em inglês / original	Em português / tradução
SD	*Strategic Dependency Model*	Modelo de dependência estratégica
SE	*Systems Engineering*	Engenharia de Sistemas
SEBOK	*Guide to System Engineering Body of Knowledge*	Guia para corpo de conhecimento engenharia de sistemas
SEMP	*Systems Engineering Management Plan*	Plano de gerenciamento de engenharia de sistemas
SERC	*Systems Engineering Research Center*	Centro de pesquisas em engenharia de sistemas
SG	*Software Glance*	Vislumbre do *Software*
SIG	*Softgoal Interdependency Graph*	Grafo de interdependência de objetivos suaves (sic)
Si*	*Secure i star*	I estrela para segurança (sic)
SIRA	*Systematic Integration between Requirements and Architecture*	Integração sistemática entre requisitos e arquitetura
SLM	*Systematic Literature Mapping*	Mapeamento sistemático da literatura
SLR	*Systematic Literature Review*	Revisão sistemática da literatura
SoS	*Systems of Systems*	Sistemas de sistemas
SR	*Strategic Rationale Model*	Modelo de raciocínio estratégico
SRS	*Software Requirements Specification*	Especificação de requisitos de s*oftware*
SSIM	*Structural Similarity Index*	Índice de similaridade estrutural
SV	*Software Vision*	Visão do *Software*
SWE	*Software Engineering*	Engenharia de *Software*
SWEBOK	*Guide to Software Engineering Body of Knowledge*	Guia para corpo de conhecimentos sobre Engenharia de *Software*
SysML	*Systems Modeling Language*	Linguagem de modelagem de sistemas
TEMP	*Test and evaluation management plan*	Plano de gerenciamento de testes e avaliação
TUCM	*Timed Use Case Maps*	Mapas de casos de uso temporizados
TransML	*Model Transformation Language*	Linguagem de transformação de modelos
TROPOS4AS	*Tropos for Adaptative Systems*	Tropos para sistemas adaptativos
UC	*Use Cases*	Casos de uso
UCM	*Use Case Maps*	Mapas de casos de uso
UEML	*Unified Enterprise Modeling Language*	Linguagem unificada para modelagem empresarial
UML	*Unified Modeling Language*	Linguagem de modelagem unificada
URN	*User Requirements Notation*	Notação de requisitos de usuários

Sigla	Em inglês / original	Em português / tradução
URML	*Unified Requirements Modeling Language*	Linguagem unificada de modelagem de requisitos
UTFPR	-	Universidade Tecnológica Federal do Paraná
VA	*Visual Analytics*	Análise visual (ciência)
v.g.	*Verbi gratia*	por exemplo
VHDL	*VHSIC Hardware Description Language*	Linguagem de descrição de hardware VHSIC
VLML	*Very Lightweight Modeling Language*	Linguagem de modelagem muito leve
VHSIC	*Very High Speed Integrated Circuit*	Circuito integrado de velocidade muito alta
VRDL	*Visual Requirements Description Language*	Linguagem visual de descrição de requisitos
XP	*eXtreme Programming*	Programação Extrema
WebUCM	*Web Use Case Maps*	Mapas de casos de uso para redes
WER	-	*Workshop* em Engenharia de Requisitos

SUMÁRIO

1 INTRODUÇÃO

Este capítulo tem por objetivo apresentar a visão geral da dissertação, descrevendo o tema do trabalho e as motivações que levaram à sua elaboração. São apresentados os objetivos da dissertação, os métodos e ferramentas propostos e utilizados, bem como a estrutura da dissertação. Este capítulo contém as seguintes subseções: a subseção 1.1 apresenta o contexto e a motivação da dissertação; a subseção 1.2 apresenta os objetivos gerais e específicos deste trabalho; a subseção 1.3 apresenta os métodos e ferramentas utilizados; a subseção 1.4 apresenta a estrutura do documento da dissertação.

1.1 MOTIVAÇÃO

A Engenharia de Requisitos (*Requirements Engineering* – RE) é a disciplina de engenharia que busca entender e expressar “o que” deverá ser feito, antes que se defina “como” será feito. A medida primária do sucesso de um *software* ou sistema corresponde ao quanto ele atinge o “propósito” para o qual foi concebido. De modo geral, a RE consiste no processo de identificar este propósito, por meio da identificação das partes interessadas (*stakeholders*) e suas necessidades, bem como definir e documentar os requisitos em uma forma que seja passível de análise, comunicação e posterior implementação (NUSEIBEH e EASTERBROOK, 2000). Uma definição formal de RE é dada como segue: “Engenharia de requisitos é a ciência e disciplina preocupada com análise e documentação de requisitos” (ISO/IEC/IEEE 24765, 2010).

“A Engenharia de *Software* (*Software Engineering* – SWE) pode ser definida como a aplicação sistemática de conhecimento científico e tecnológico, métodos e experiência para o projeto, implementação, teste e documentação de *software*” (ISO/IEC/IEEE 24765, 2010). A RE surgiu como uma subdisciplina de SWE (ZAVE, 1997). A importância da RE e seu impacto no resultado de projetos de *software* é bem conhecida, como é possível observar nos seguintes trechos da literatura de SWE e RE:

a) “É um fato razoavelmente bem documentado que a definição de requisitos de *software* tem um grande impacto na qualidade do produto final” (STANDISH GROUP, 1994);

b) “Requisitos deficientes são a principal razão individual para a falha de projetos de *software*” (HOFMANN e LEHNER, 2001);
c) “Um dos principais motivos para projetos de *software* falharem consiste em requisitos de sistema mal definidos” (CHARETTE, 2005);
d) “A falta de precisão na elicitação de requisitos de um sistema é uma das maiores razões de falhas em 90% dos grandes projetos de *software*” (DAVIS et al., 2006);
e) “Requisitos instáveis estão entre as causas mais comuns para a falha em projetos de *software*, por levar a mudanças no escopo do projeto” (VERNER et al., 2008);
f) Dentre as principais causas para falhas de projetos de *software*, foi possível verificar que causas relacionadas a “*Sales & Requirements*” correspondem diretamente a 19,5% das causas de falhas, além de influenciar ou derivar falhas ocorridas em implementação e testes” (LEHTINEN et al., 2014);
g) “Falta de comunicação, falta de tempo para melhorias e testes apropriados, **requisitos inadequados** e especificação de funcionalidades incorretos são as falhas mais relatadas em processos de desenvolvimento de *software*” (MARQUES et al., 2017).

“A Engenharia de Sistemas (*Systems Engineering* – SE) é uma abordagem interdisciplinar que governa todo o esforço técnico e gerencial necessário para transformar um conjunto de necessidades, expectativas e restrições do cliente em uma solução bem como suportar toda a vida dessa solução” (ISO/IEC/IEEE 24765, 2010). De forma similar à área de SWE, é grande a importância da definição de requisitos para a área de SE: “o desenvolvimento de requisitos é muitas vezes venerado como um dos aspectos mais críticos, se não o mais crítico, segundo o paradigma de SE tradicional” (KEATING et al., 2008).

A representação de requisitos pode ser realizada de diversas formas, como por exemplo: texto, tabelas, modelos gráficos, linguagens formais de especificação de requisitos e até mesmo formulações matemáticas. Algumas destas representações estão restritas ao meio acadêmico ou a indústrias muito especificas. Na maioria das indústrias, a especificação de requisitos é

tradicionalmente realizada de forma sentencial, ou seja, feita em linguagem textual ou por meio de tabelas contendo o texto (tabular).

Nesse sentido, há normas específicas que determinam como os requisitos devem ser escritos em formato texto, como a "*Systems and software engineering - Life cycle processes - Requirements engineering*" (ISO/IEC/IEEE 29148, 2011), que contém uma série de recomendações de escrita de requisitos textuais. Geralmente estas recomendações são aplicadas sobre o documento em formato texto de "Especificação de Requisitos de *Software*" (*Software Requirements Specification* - SRS), que desempenha um papel importante em estabelecer a base sobre o que o *software* se destina a fazer (BOURQUE e FAIRLEY, 2014). Até mesmo a linguagem de modelagem de sistemas SysML (OMG, 2007), que possui um forte apelo visual por meio do uso de diagramas diversos, representa requisitos em um diagrama obrigatoriamente vinculado a uma representação textual equivalente (tabular).

De forma geral, a questão de interesse que surgiu no presente trabalho é: "Seria possível representar requisitos de forma visual (diagramática) em contraposição, ou mesmo em complementação, à abordagem tradicional textual utilizada para especificação de requisitos, de forma a contribuir para a melhoria da qualidade das especificações de *software* e sistemas?"

"Uma imagem vale mais do que mil palavras" é uma expressão popular atribuída ao filósofo chinês Confúcio (que viveu entre 552 a.C e 479 a.C), cujo significado remete a ideia de que a comunicação por meio de **representações visuais** tende a ser mais eficaz em comparação com a escrita. Representações visuais podem ser eficazes em diversos contextos porque exploram as capacidades do sistema visual humano, que é poderoso e altamente paralelo. Humanos gostam de receber informações em forma visual e podem processá-las de forma muito eficiente: cerca de 25% do cérebro é dedicado à visão, mais do que todos os outros sentidos combinados (KOSSLYN et al., 1985).

No campo de ciência cognitiva, uma ideia parecida ao ditado popular atribuído a Confúcio foi apresentada no artigo "*Why a Diagram is (Sometimes) Worth Ten Thousand Words*". Segundo os autores deste artigo, **representações diagramáticas** preservam explicitamente informações sobre as relações topológicas e geométricas entre componentes de um problema, enquanto

representações sentenciais não o fazem (LARKIN e SIMON, 1987). Diagramas podem transmitir informações de forma mais concisa e precisa do que a linguagem natural (DEMARCO, 1979). A informação representada visualmente é mais provável de ser lembrada devido ao efeito de superioridade da imagem (GOOLKASIAN, 2000), (LINDWELL et al. 2003).

Pesquisas envolvendo diagramas em RE se iniciaram na década de 1990, tais como os modelos de representação NFR (MYLOPOULOS et al., 1992), KAOS (DARDENNE et al., 1993), UCM (BUHR e CASSELMAN, 1995) e i* (YU, 1995). Inicialmente, tais modelos tiveram pouco impacto na representação de requisitos em termos industriais. A introdução de diagramas de RE como parte de normas internacionais de engenharia e telecomunicações está lentamente modificando este panorama sendo possível salientar, por exemplo, as normas SysML (OMG, 2007) e a URN Z.151 (ITU-T, 2008).

Uma das principais vantagens dos diagramas consiste na maior efetividade perceptiva dos relacionamentos entre seus elementos componentes (LARKIN e SIMON, 1987). Entender os requisitos e seus relacionamentos para sistemas complexos sempre foi um fator crítico de sucesso para projetos de *software* e sistemas (HELMING et al., 2010). Assim, as **dependências** entre requisitos individuais têm importante influência sobre atividades de SWE como, por exemplo: planejamento de projeto, projeto de arquitetura e análise de impacto de mudanças (ZHANG et al., 2014).

A revisão sistemática da literatura sobre modelos gráficos em RE (seção 2.3) apresentada nesta dissertação permitiu identificar que muitas das abordagens existentes não apresentam uma semântica precisa e sistemática das interdependências entre requisitos.

Nesse sentido, o presente trabalho procura contribuir para melhoria da qualidade da especificação de requisitos de *software* e sistemas por meio da proposição de uma linguagem e um método de modelagem gráfica que permita **representar requisitos e suas interdependências** de forma sistemática, precisa e expressiva. A linguagem proposta nesta dissertação chama-se **RIMON** (*Requirements and Interdependencies MOdeling Notation*).

A linguagem RIMON foi inspirada e desenvolvida com base nas ideias propostas na abordagem de modelagem RON (Requisitos Orientados a Notificações) introduzida pioneiramente por pesquisadores da UTFPR

(Universidade Tecnológica Federal do Paraná), notadamente destacando-se os professores Jean Marcelo Simão, Paulo Cezar Stadzisz e colaboradores (SIMAO et al., 2016). A RIMON é composta por uma notação visual e por um método de modelagem especificamente desenvolvido para uso desta notação, e possui as seguintes características:

a) Foi desenvolvida com base em princípios de física das notações (PoN);
b) Possui boa expressividade visual (símbolos significativos);
c) Prima pela economia gráfica (poucos símbolos);
d) Se ajusta cognitivamente a diferentes públicos (uso de dialetos visuais);
e) É precisa e visualmente expressiva na modelagem de requisitos funcionais, de não-funcionais e de suas interdependências;
f) Permite modelar restrições relativas a requisitos não-funcionais por meio de atributos contendo condições lógico-matemáticas;
g) Permite realizar a identificação de conflitos entre requisitos.

Maiores explicações e detalhamento sobre as características da linguagem RIMON apresentadas acima serão fornecidas no texto da presente dissertação.

1.2 OBJETIVOS

Esta subseção apresenta o objetivo geral da dissertação, bem como os objetivos específicos.

1.2.1 Objetivo geral

O objetivo geral deste trabalho de pesquisa é contribuir para a melhoria da qualidade da especificação de requisitos de *software* e sistemas por meio da proposição de uma linguagem e um método de modelagem gráfica de requisitos, que permita representar requisitos e suas interdependências de forma sistemática, precisa e expressiva.

1.2.2 Objetivos específicos

Para atingir o objetivo geral, os seguintes objetivos específicos foram propostos:

i. **Estabelecer uma base conceitual para a linguagem de modelagem gráfica a ser proposta**

A **sintaxe** (forma) da linguagem deverá se basear em:

- Princípios para desenvolvimento de notações visuais (física das notações – PoN);
- Princípios e conceitos sobre metamodelagem;
- Conceitos obtidos de abordagens de modelagem gráfica para RE.

A **semântica** (significado) da linguagem deverá se basear em:

- Abordagem de modelagem gráfica RON (SIMAO et al., 2016);
- Conceitos sobre tipos de interdependências entre requisitos;
- Conceitos obtidos de abordagens de modelagem gráfica para RE.

ii. **Estabelecer uma base conceitual para a método de modelagem gráfica a ser proposto**

Deve ser estabelecida uma base conceitual para o desenvolvimento do método de modelagem gráfica, a partir de conceitos básicos existentes na abordagem de modelagem gráfica RON (SIMAO et al., 2016). Esta base conceitual poderá ser ampliada e modificada conforme necessário, de forma a atender aos objetivos e da nova linguagem gráfica.

iii. **Definir um método para desenvolver e verificar a linguagem de modelagem gráfica a ser proposta**

Deve ser estabelecido um método que permita definir, descrever e verificar a linguagem de modelagem gráfica de forma clara.

iv. **Definir a linguagem de modelagem gráfica de requisitos**

A definição da linguagem pode envolver os seguintes passos:

a) Definir diretrizes para desenvolvimento da linguagem;
b) Descrever a sintaxe abstrata e a sintaxe concreta da linguagem e o correspondente mapeamento semântico;
c) Apresentar a notação visual proposta em um formato unificado para fácil utilização pelos usuários.

v. **Definir o método para modelagem gráfica de requisitos**

Definir um método para modelagem e descrever suas etapas.

vi. **Realizar experimentos de aplicação para verificação da aplicabilidade do método e da notação da linguagem gráfica**

Realizar no mínimo dois experimentos de aplicação, sendo ao menos um originado da área de SE e outro da área de SWE, usando a notação unificada proposta e contemplando o número necessário de variações visuais e semânticas para verificar a aplicabilidade do método e notacão.

vii. **Verificar a expressividade e precisão da linguagem de modelagem gráfica sob a perspectiva de princípios de desenvolvimento de notações visuais**

Verificar se os princípios da teoria de física das notações (PoN), que estabelece boas práticas para a definição de notações visuais, foram atendidos.

viii. **Verificar a expressividade da linguagem de modelagem em relação a tipos de interdependências de requisitos suportados**

Verificar quais tipos de interdependências de requisitos existentes na literatura são suportados pela linguagem proposta.

ix. **Verificar se as diretrizes propostas para a linguagem e método de modelagem gráfica foram atendidas**

Verificar se a linguagem de modelagem gráfica e o método propostos satisfazem as diretrizes propostas para o seu desenvolvimento.

x. **Comparar a linguagem de modelagem proposta em relação a outros sistemas de modelagem gráfica de requisitos**

Comparar as características gerais da linguagem de modelagem gráfica proposta com alternativas de modelagem gráfica de requisitos identificadas na literatura de RE.

1.3 MÉTODOS E FERRAMENTAS

Do ponto de vista de método científico (KOTHARI, 2004), o presente trabalho de pesquisa pode ser caracterizado como uma pesquisa **aplicada, qualitativa e empírica**. Sob a perspectiva de RE (WIERINGA, 2006), ele se classifica como uma **pesquisa de validação**. Assim, este trabalho se caracteriza por ser uma pesquisa:

a) **Aplicada**: visa buscar soluções para problemas existentes em organizações industriais/empresariais da sociedade (KOTHARI, 2004);
b) **Qualitativa**: se concentra em aspectos qualitativos na busca de soluções para os problemas propostos (KOTHARI, 2004);

c) **Empírica**: se baseia em experimentos e observação para validação dos resultados (KOTHARI, 2004);
d) **De validação, segundo proposta de classificação de trabalhos em RE** (WIERINGA et al., 2006): investiga as propriedades de uma proposta de solução que ainda não foi implementada na prática de RE. A investigação usa uma configuração de pesquisa completa e metodologicamente sólida. Utiliza métodos tais como experimentos, análise e validação de resultados (WIERINGA et al., 2006).

Em termos de **ferramentas metodológicas**, este trabalho apoia-se em pesquisa bibliográfica, mapeamento sistemático da literatura (*SLM – Systematic Literature Mapping)*, revisão de trabalhos correlatos do grupo de pesquisa, experimentação, análise e validação de resultados. Também é apresentado na seção 3.1 um novo método para a definição e verificação da notação visual da linguagem de modelagem gráfica, a partir da integração de teorias e técnicas existentes na literatura. Na Figura 1 é mostrado o método de pesquisa adotado neste trabalho, em formato BPMN (*Business Process Modeling Notation*):

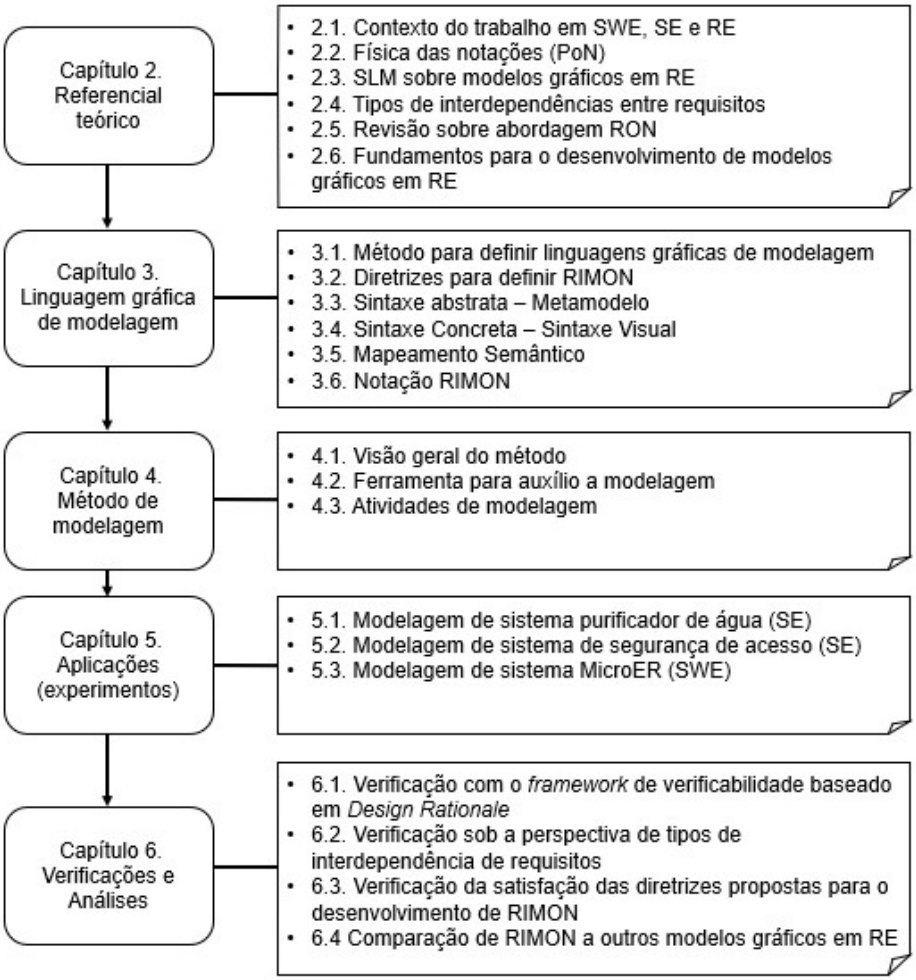


Figura 1 – Método adotado no trabalho de pesquisa
Fonte: o autor

1.3.1 Elaboração do referencial teórico - Capítulo 2

O desenvolvimento do referencial teórico deste trabalho envolveu o uso de diferentes abordagens de pesquisa, que incluíram:

a) Pesquisa bibliográfica sobre SWE, SE e RE (seção 2.1), visando estabelecer o contexto deste trabalho em relação a estas disciplinas e, em particular, nos processos de RE;
b) Pesquisa bibliográfica sobre física das notações e ferramentas relacionadas (*e.g.*: *design rationale*) (seção 2.2);
c) Realização de trabalho de mapeamento sistemático de literatura sobre modelos gráficos em RE (seção 2.3 e apêndices A, B e C), de forma a fundamentar algumas questões relacionadas ao presente trabalho:

- Quais modelos gráficos em RE existentes são potencialmente comparáveis à abordagem proposta? (resposta: seção 2.3)
- Quais são os diferenciais da proposta deste trabalho em relação aos principais modelos gráficos em RE existentes? (resposta: seção 6.4)
- Como a proposta deste trabalho se insere no contexto e processos de RE, em relação aos principais modelos gráficos em RE identificados? (resposta: seção 6.4)
- Quais são os requisitos ou objetivos almejados pelos principais modelos gráficos de RE identificados e que podem ser incorporados pelo presente trabalho? (resposta: seções 2.6.2 e 3.2)

d) Pesquisa bibliográfica sobre "classificação de tipos de interdependências de requisitos" (seção 2.3.8), de forma a fundamentar a verificação da notação sob esta perspectiva (resposta: seção 6.2);
e) Revisão da literatura sobre a abordagem RON (Requisitos Orientados a Notificações) e conceitos relacionados (seção 2.5);
f) Pesquisa bibliográfica visando identificar objetivos e diretrizes para o desenvolvimento de modelos gráficos em RE a serem incorporados pelo presente trabalho (seção 2.6).
g) Pesquisa bibliográfica para estabelecimento de **terminologia,** construída com base em normas internacionais (*e.g.* ISO/IEC/IEEE) e referências bibliográficas. Esta seção ("APÊNDICE D") concentra os termos empregados nesta dissertação sobre as áreas de SWE, SE e RE;

1.3.2 Desenvolvimento de linguagem de modelagem gráfica - Capítulo 3

Neste capítulo, será estabelecido inicialmente um novo método para a definição e verificação da linguagem de modelagem gráfica baseada em teoria de PoN, integrando os seguintes princípios e teorias (seção 3.1):

a) Princípios de metamodelagem aplicada ao desenvolvimento de linguagens de modelagem gráficas;
b) Teoria de física das notações (PoN);
c) *Framework* de verificabilidade de notações visuais baseadas em princípios PoN e de razão de projeto (*design rationale)*.

Na sequência do capítulo, o método citado acima será aplicado para realizar a definição da linguagem de modelagem gráfica RIMON.

1.3.3 Desenvolvimento do método de modelagem gráfica - Capítulo 4

Neste capítulo, será criado o método de modelagem gráfica RIMON. Serão mostradas as atividades do método, suas entradas, saídas e ferramentas auxiliares.

1.3.4 Aplicações e experimentos em SE e SWE - Capítulo 5

Para avaliar a aplicabilidade da notação e do método de modelagem propostos, serão conduzidas experimentações (casos de estudo) com base em especificações de requisitos de *software* (SWE) e de sistemas (SE). As aplicações realizadas apresentarão uma variabilidade que permita uma verificação sólida das diversas características e elementos da notação.

1.3.5 Verificações e Análises - Capítulo 6

As seguintes verificações e análises serão realizadas:

a) Verificação da linguagem de modelagem gráfica proposta neste trabalho sob a perspectiva do *framework* de verificabilidade baseado em *design rationale* para notações visuais definidas segundo os princípios de PoN;
b) Verificação da modelagem proposta neste trabalho sob a perspectiva “tipos de interdependências de requisitos”;
c) Verificação da satisfação da linguagem de modelagem proposta em relação às diretrizes estabelecidas para seu desenvolvimento;
d) Comparação da modelagem proposta neste trabalho com relação a outros modelos gráficos em RE.

1.4 ESTRUTURA DA DISSERTAÇÃO

A estrutura da dissertação e dos principais capítulos será apresentada por meio de diagramas em formato BPMN (*Business Process Modeling Notation*), tal como realizado em (SPIJKERMAN, 2010). Estes diagramas fornecem ao leitor um mapa conceitual e estrutural, que facilita e auxilia no entendimento e exploração do conteúdo deste trabalho, sem a necessidade de

textos longos introduzindo as seções de cada capítulo. Seguindo essa lógica, a estrutura desta dissertação é apresentada na Figura 2.

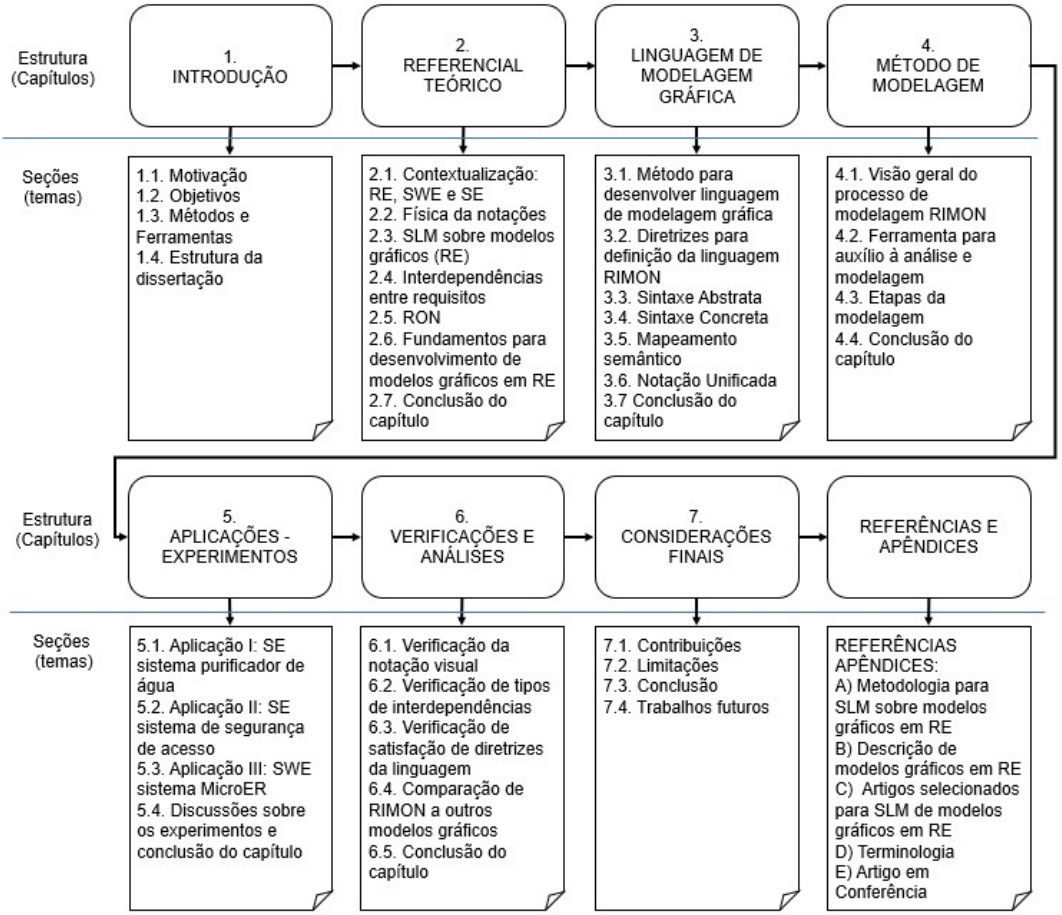


Figura 2 – Mapa conceitual/estrutural: Dissertação
Fonte: o autor

2 REFERENCIAL TEÓRICO

Neste capítulo é apresentado o referencial teórico que fundamenta esta dissertação. Procurou-se manter o máximo de sequência e coerência lógica entre as seções, de forma que cada uma delas propiciasse o embasamento adequado para as seções seguintes (ver mapa conceitual do capítulo em formato BPMN na Figura 3).

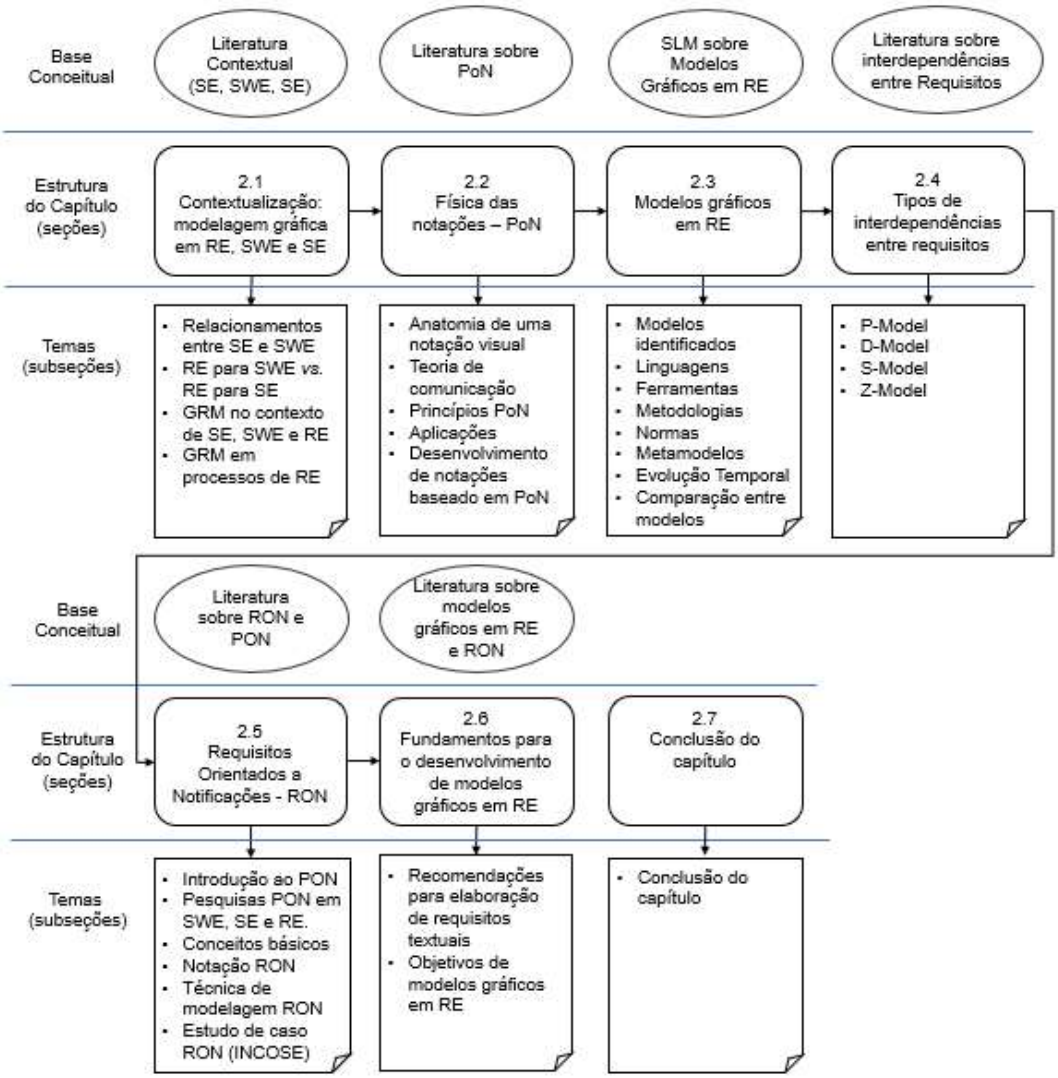


Figura 3 – Mapa conceitual/estrutural: Capítulo 2
Fonte: o autor

2.1 CONTEXTUALIZAÇÃO: MODELAGEM GRÁFICA EM RE, SWE E SE

O objeto de estudo da presente dissertação se refere a uma abordagem de modelagem gráfica de requisitos (*Graphical Requirements Modeling* – GRM), no contexto das disciplinas de Engenharia de Requisitos (*Requirements Engineering* - RE), Engenharia de *Software* (*Software Engineering* - SWE) e Engenharia de Sistemas (*Systems Engineering* - SE). Para entender o relacionamento entre GRM e as áreas de conhecimento de RE, SWE e SE, esta seção se propõe a: ressaltar pontos comuns entre os campos disciplinares de SE e SWE; apresentar o contexto de RE para SWE vs. RE para SE; situar GRM nos supracitados campos disciplinares; e contextualizar modelos gráficos de requisitos em processos de RE.

2.1.1 Relacionamentos entre SE e SWE

A Engenharia de Sistemas (SE) é um campo de engenharia interdisciplinar que foca em como projetar e gerenciar o ciclo de vida em projetos de engenharia complexos e multidisciplinares. Apesar de ter surgido por volta de 1960, a SE só emergiu como disciplina (com periódicos, práticas, normas e formação acadêmica) por volta de 1990 (SHEARD, 2014). Os principais padrões de SE (normas internacionais) foram estabelecidos na década de 90 (SHEARD, 1998), sendo que na atualidade os principais padrões da área são definidos pelo INCOSE (*International Council on Systems Engineering*) (INCOSE, 2015).

A *Engenharia de Software* (SWE) é a aplicação de uma abordagem sistemática, disciplinada e quantificável ao desenvolvimento, operação e manutenção de *software*; *i.e.,* a aplicação da engenharia ao *software* (BOURQUE e FAIRLEY, 2014). A SWE tem suas origens relacionadas a uma conferência da OTAN realizada em 1968, considerada como sendo a primeira sobre SWE (RANDELL, 2018). Em 2018, a 40ª conferência internacional ICSE (*International Conference on Software Engineering*) celebrou em plenário os 50 anos da SWE.

O *software* é proeminente na maioria das arquiteturas de sistemas modernos e é, frequentemente, o principal meio de integração de componentes complexos de sistemas. A SWE e a SE não são meramente disciplinas relacionadas, pois estão intimamente interligadas. Uma boa SE é um fator chave

para permitir uma boa SWE (BOURQUE e FAIRLEY, 2014). O SEBoK (“corpo de conhecimentos em SE”) reconhece e abraça explicitamente o entrelaçamento entre SE e SWE, bem como define a relação entre o SEBoK e o SWEBOK (“corpo de conhecimentos de SWE”) (SERC, 2018).

A unificação de SE e SWE é defendida desde os anos 2000, porém há dificuldades. Segundo Boehm: “Organizações que combinam os processos de SWE e de SE obtém ganhos através desta unificação” (BOEHM, 2000). A parte difícil consiste em alcançar um modelo de referência de processo único e adequado para os ciclos de vida de sistemas e *software*, bem como para a definição de processos e avaliações (MOORE, 2006).

A Figura 4 ilustra a sobreposição e exclusividade em atividades de SE e SWE. Atividades relacionadas a SE estão na esquerda (vermelho), SWE na direita (azul) e de ambas as disciplinas no centro (roxo) (SHEARD, 2014).

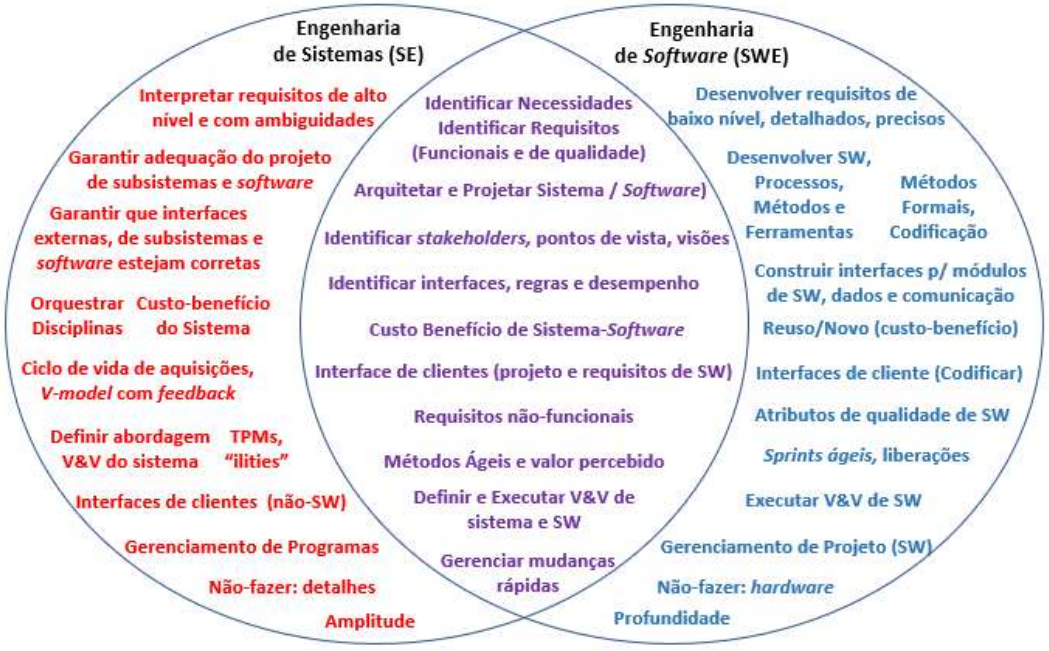


Figura 4 – Atividades de SWE e SE: sobreposição e exclusividade
Fonte: Adaptado (traduzido) de (SHEARD, 2014)

O termo *hardware* aparece apenas na direita, porque os engenheiros de *software* o utilizam para referenciar coisas que não são *software*, enquanto engenheiros de sistemas geralmente não o utilizam. Em vez disso, eles chamam os subsistemas físicos pelo nome (*e.g.*: energia, propulsão, estruturas) ou descrevem disciplinas ou domínios (*e.g.*: engenharia mecânica, de *software*,

análise de órbita). Além disso, responsabilidades de engenheiros de sistemas tendem a ser mais amplas que profundas e, certamente, não são profundas em *software*. Por outro lado, as responsabilidades dos engenheiros de *software* tendem a ser tanto amplas quanto profundas em *software* (SHEARD, 2014).

Regina Gonsales registra que há diferenças culturais entre as áreas de SE e SWE (GONZALES, 2005), conforme mostrado na Tabela 1.

	Engenharia de Software	Engenharia de Sistemas
Crenças e Premissas Fundamentais	Engenheiros podem reduzir problemas ou soluções a um conjunto de formalismos. A especificação e aplicação precisa de métodos de SWE padronizados produzirá resultados iguais, independentemente das pessoas envolvidas. O *software* é o sistema.	Formalismos ajudam, mas a comunicação é mais importante Coletar, misturar e combinar métodos e técnicas, colocando-as em um "*kit*" de ferramentas. Aplicar conforme apropriado, sem qualquer garantia de sucesso. O *software* é um componente do sistema.
Valores compartilhados	Desenvolver formalismos para o problema ou domínio é o passo mais importante para a solução.	Entender as pessoas envolvidas e sua dinâmica interpessoal é muito importante.
Padrões Comportamentais	Formalizar ou estabelecer o problema de forma precisa, mesmo que isso requeira reduzir o escopo. Modelar o problema usando formalismos.	Desenvolver entendimento completo dos aspectos operacionais de uma solução prospectiva e usar uma linguagem natural para estabelecer um acordo.
Normas Comportamentais	Evitar a "bagunça" comum dos sistemas no mundo real, tal como implantação, envelhecimento, ambientes físicos ruins e caos. Ser moderado - às vezes sectário - sobre novos métodos.	Aplicar um equilíbrio de métodos de gerenciamento e engenharia para alcançar o resultado desejado. Ser avesso ao risco.

	Engenharia de Software	Engenharia de Sistemas
Artefatos	Diagramas UML e cenários do usuário, dentre outros.	Cenários operacionais e documentos de gerenciamento, tais como SEMP (plano de gerenciamento de engenharia de sistemas), TEMP (plano de gerenciamento de testes e avaliação), planos de gerenciamento de riscos, artefatos SysML etc.

Tabela 1 – Cultura de SWE versus cultura de SE
Fonte: Adaptado (traduzido) de (GONZALEZ, 2005)

Apesar do entrelaçamento entre SE e SWE, essas disciplinas evoluíram independentemente e em comunidades "separadas". Segundo Stephanie White, isso se reflete em atitudes encontradas na indústria: "No paradigma disciplinar, SWE e SE não estão bem integrados, sendo que engenheiros de sistemas alocam requisitos de *software* a subsistemas antes ter uma compreensão detalhada dos requisitos do sistema como um todo. Frequentemente, o sistema resultante tem interfaces excessivamente complexas e desempenho ruim. Quando isso ocorre, a arquitetura deve ser alterada após o sistema estar construído, tornando-a difícil de manter" (WHITE, 2014).

2.1.2 RE para SWE *versus* RE para SE

Historicamente, apesar de ser utilizável tanto em SE quanto em SWE, a disciplina de Engenharia de Requisitos (RE) se originou a partir da SWE, como pode ser observado na seguinte definição: "RE é o ramo (subdisciplina) da SWE preocupada com as metas, funções e restrições reais dos sistemas de *software*. Também se preocupa com a relação desses fatores com especificações precisas do comportamento do *software* e com sua evolução ao longo do tempo e entre as famílias de *software*" (ZAVE, 1997).

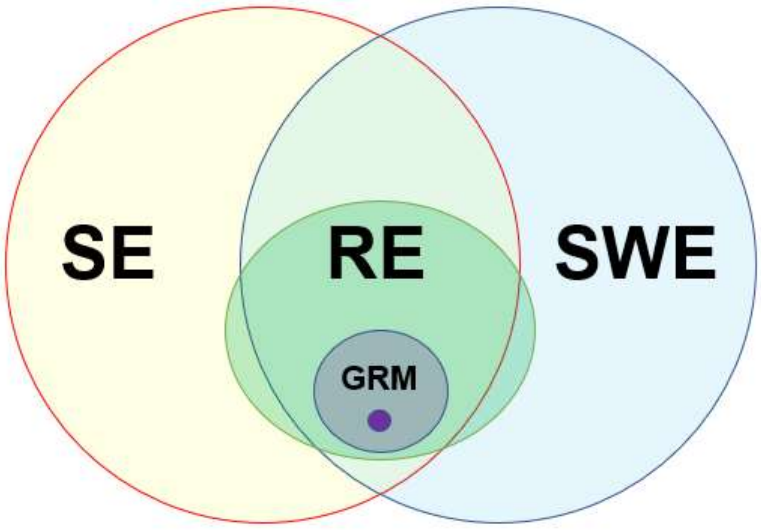
Para identificar diferenças entre RE para SWE e RE para SE deve-se começar por esclarecer as relações entre os conceitos *"software*" (definição na seção 1) e "sistema" (definição na seção 6). Além de *software*, um sistema de informação inclui *hardware*, infraestrutura e pessoas que usam o sistema

(*stakeholders* - definição na seção 13). Assim, RE para SE pode ser vista como sendo mais ampla do que RE para SWE (KAINDL et al., 2012).

Em relação à RE, o que ocorre é que engenheiros de sistemas ainda estão aprendendo a aplicar abordagens mais metodológicas para elicitar e analisar requisitos, enquanto engenheiros de *software* estão aprendendo a ter a visão mais ampla necessária para desenvolver soluções de sistema. Ambas as comunidades estão em um processo de amadurecimento na compreensão da disciplina de RE e estão começando a entender que a captura de requisitos requer a compreensão do problema antes de desenvolver uma solução. A cultura de cada comunidade necessariamente influencia a outra (GONZALES, 2005).

2.1.3 GRM no contexto de SE, SWE e RE

A modelagem gráfica de requisitos (GRM) se refere à prática bem difundida na área de RE de representar requisitos e seus inter-relacionamentos sob o formato de diagramas visuais e de racionalizar sobre estas representações (GILLAIN et al., 2016). Com base nesta definição e nas informações das seções anteriores (2.1.1 e 2.1.2), é possível apontar como se insere o conhecimento que será apresentado nesta dissertação (Figura 5).



Termos utilizados:
Engenharia de Sistemas → *Systems Engineering* (SE)
Engenharia de *Software* → *Software Engineering* (SWE)
Engenharia de Requisitos → Requirements Engineering (RE)
Modelagem Gráfica de Requisitos → Graphical Requirements Modeling (GRM)

Figura 5 – Contextualização de GRM em SE, SWE e RE
Fonte: o autor

A Figura 5 foi construída de forma a determinar o relacionamento entre as áreas, conforme segue:

a) GRM corresponde a um conjunto de práticas pertencentes a RE;
b) RE é uma subdisciplina tradicional da SWE, que tem ganhado espaço e aplicações no campo de SE (seção 2.1.2);
c) SWE e SE são campos disciplinares inter-relacionados e interdependentes e que estão em processo de integração. Além disso, RE possui conhecimentos específicos e exclusivos de cada área (RE exclusiva para SE e RE exclusiva para SWE) (seção 2.1.1);
d) Esta dissertação pode ser situada na Figura 5 (ponto menor dentro do círculo de GRM) sob o contexto predominante de RE e com impactos nas áreas de SWE e SE.

2.1.4 Processos de RE

Um processo de RE visa produzir um documento de especificação que satisfaça as necessidades das partes interessadas (*stakeholders*) no sistema (SOMMERVILLE, 2010). Diferentes autores e organizações subdividem e organizam seus processos de RE de diferentes formas. Há, porém, um conjunto de atividades comuns que geralmente são realizadas, o que permite caracterizar modelos de processo. Uma das proposições de processos para RE, em que as atividades são realizadas usualmente da esquerda para a direita, é mostrado na Figura 6.

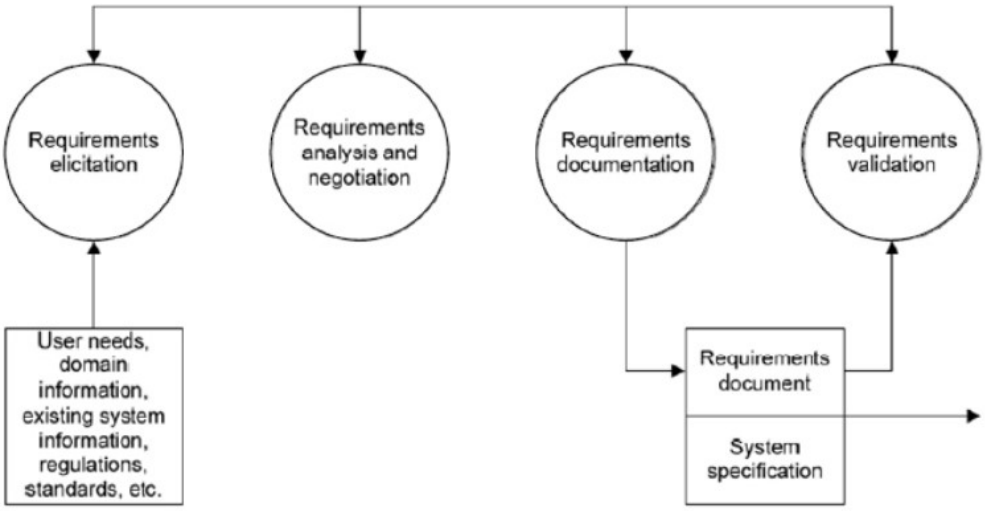


Figura 6 – Modelo de processo de RE – Kotonya e Sommerville
Fonte: (KOTONYA e SOMMERVILLE, 1998)

Este processo contempla (KOTONYA e SOMMERVILLE, 1998):

a) **Elicitação de requisitos:** os requisitos de sistema são “descobertos” através de consulta a *stakeholders*, a partir de documentos do sistema, conhecimento de domínio e estudos de mercado. Outros nomes para este processo são “aquisição de requisitos” e “descoberta de requisitos”;
b) **Análise e negociação de requisitos**: a análise detalhada dos requisitos é realizada, bem como negociações com os *stakeholders*. O objetivo é determinar quais requisitos devem permanecer, de forma a evitar conflitos entre requisitos e, eventualmente, ajustar o escopo;
c) **Documentação de requisitos**: os requisitos acordados são documentados em um nível apropriado de detalhe em um documento de requisitos contendo a especificação dos requisitos do sistema (*system requirements specification)*;
d) **Validação de requisitos**: uma verificação cuidadosa dos requisitos especificados é feita de forma a garantir sua consistência e completude. Uma segunda proposição de processo de RE é mostrada na Figura 7.

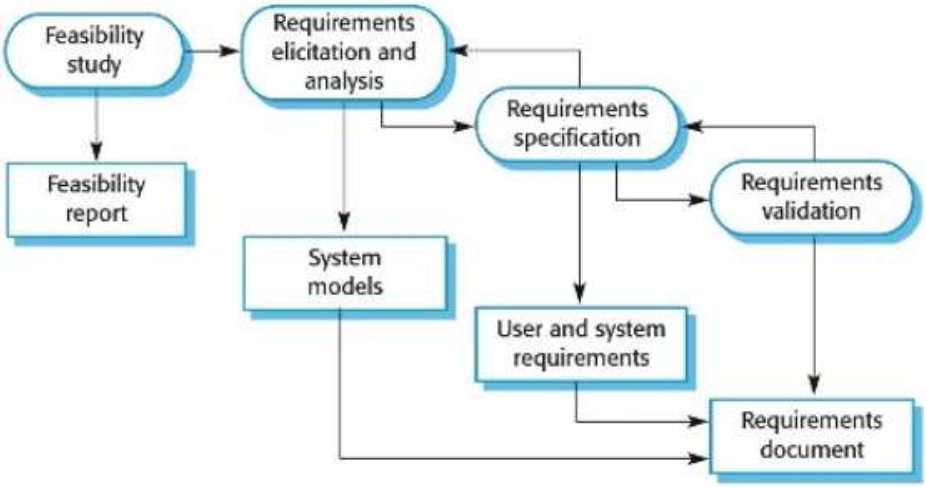


Figura 7 – Modelo de processo de RE – Sommerville
Fonte: (SOMMERVILLE, 2010)

Este processo contempla (SOMMERVILLE, 2010):

a) **Estudo de viabilidade:** procura determinar se um sistema proposto terá suficiente relação custo-benefício e se o projeto deve prosseguir;
b) **Elicitação e análise de requisitos**: processo de obter ou derivar os requisitos esperados para o sistema a partir da observação de sistemas

existentes e conversas com os *stakeholders*, de forma a desenvolver um entendimento do sistema a ser especificado;

c) **Especificação de requisitos**: transformação das informações obtidas anteriormente em um documento com um conjunto de requisitos de usuário (mais abstratos) e/ou de sistema (mais detalhados);
d) **Validação de requisitos**: verificação e correção dos requisitos especificados para garantir seu realismo, consistência e completude.

O processo sugerido pelas normas (ISO/IEC/IEEE 29148, 2011) e (ISO/IEC/IEEE 15288, 2008) contempla as seguintes atividades em relação a requisitos de *stakeholders*:

a) **Elicitação de requisitos**: descobrir as necessidades, desejos, expectativas e restrições esperadas para o sistema e relatadas pelos *stakeholders* legítimos e representá-las em um modelo textual ou formal com o propósito e comportamento esperado para o sistema;
b) **Definição de requisitos**: definir restrições para uma possível solução de sistema, identificando os serviços necessários que correspondam aos cenários operacionais e de apoio, além de possíveis interações entre usuários e sistema e requisitos de qualidade críticos (segurança, ambiente etc.);
c) **Análise e manutenção de requisitos**: analisar o conjunto completo de requisitos definidos, de forma a assegurar a correção, consistência, viabilidade e confiabilidade em relação aos desejos dos *stakeholders*, bem como a posterior rastreabilidade.

Um processo de RE comum e amplamente difundido, segundo (LOUCOPOULOS et al., 2013), consiste em:

a) **Descoberta** (ou **elicitação**) **de requisitos**: compreender o domínio da aplicação e inferir as necessidades específicas de projeto por meio de consultas com *stakeholders* e outras fontes;
b) **Especificação de requisitos**: é o processo pelo qual a equipe de projeto adquire, abstrai e representa os requisitos, que são reunidos em um documento de especificação que representa um acordo entre as partes interessadas e a equipe de projeto;

c) **Negociação de requisitos**: os *stakeholders* concentram-se em soluções alternativas e tentam prever o comportamento potencial do sistema antes de sua implementação;
d) **Validação e verificação:** assegurar que os requisitos são de alta qualidade; atendem às necessidades dos usuários; são apropriados para o esforço de projeto; são consistentes e sem defeitos.

Por fim, outra dimensão que pode ser analisada em relação a processos de requisitos se refere a dicotomia **contextual,** que subdivide o processo nas etapas de *Early Requirements* (requisitos antecipados (sic)) e *Late Requirements* (requisitos atrasados (sic)) (LOUCOPOULOS et al., 2013):

a) ***Early Requirements***: são quase que exclusivamente orientados pela comunicação com *stakeholders* e, normalmente, envolvem questões como: (i) processos de negócios; (b) demanda por produtos ou serviços; (c) nível de serviço esperado; (d) recursos necessários; e (e) compromisso entre níveis de serviço e recursos necessários;
b) ***Late Requirements***: concentram-se nos atributos do sistema desejado em termos de funcionalidade e qualidade.

2.1.5 Aplicação da GRM

A modelagem gráfica de requisitos pode ser realizada em diferentes etapas de um processo de RE ou em diferentes contextos. A Figura 8 apresenta a aplicabilidade dos principais modelos em RE (ver seção 2.3.1), no contexto dos diferentes processos de requisitos elencados na seção anterior. As abordagens (ou modelos gráficos) da figura podem ser contextualizadas da seguinte forma:

- ***Early requirements****:* abordagens que se concentram em etapas iniciais do processo de requisitos (*e.g.:* elicitação) tais como o i* (YU, 1995);
- **Ciclo Completo:** abordagens que contemplam o ciclo completo do processo de requisitos (*early requirements* + *late requirements*), tais como KAOS NFR (MYLOPOULOS et al., 1992) (DARDENNE et al., 1993), URN (AMYOT, 2002) e Tropos (GIORGINI et al., 2004);
- **Late Requirements**: abordagens focadas em etapas finais do processo (e.g.: especificação), tais como UML Use Cases (JACOBSON et al., 1992), UCM (BUHR et al., 1995) e SysML (OMG, 2007).

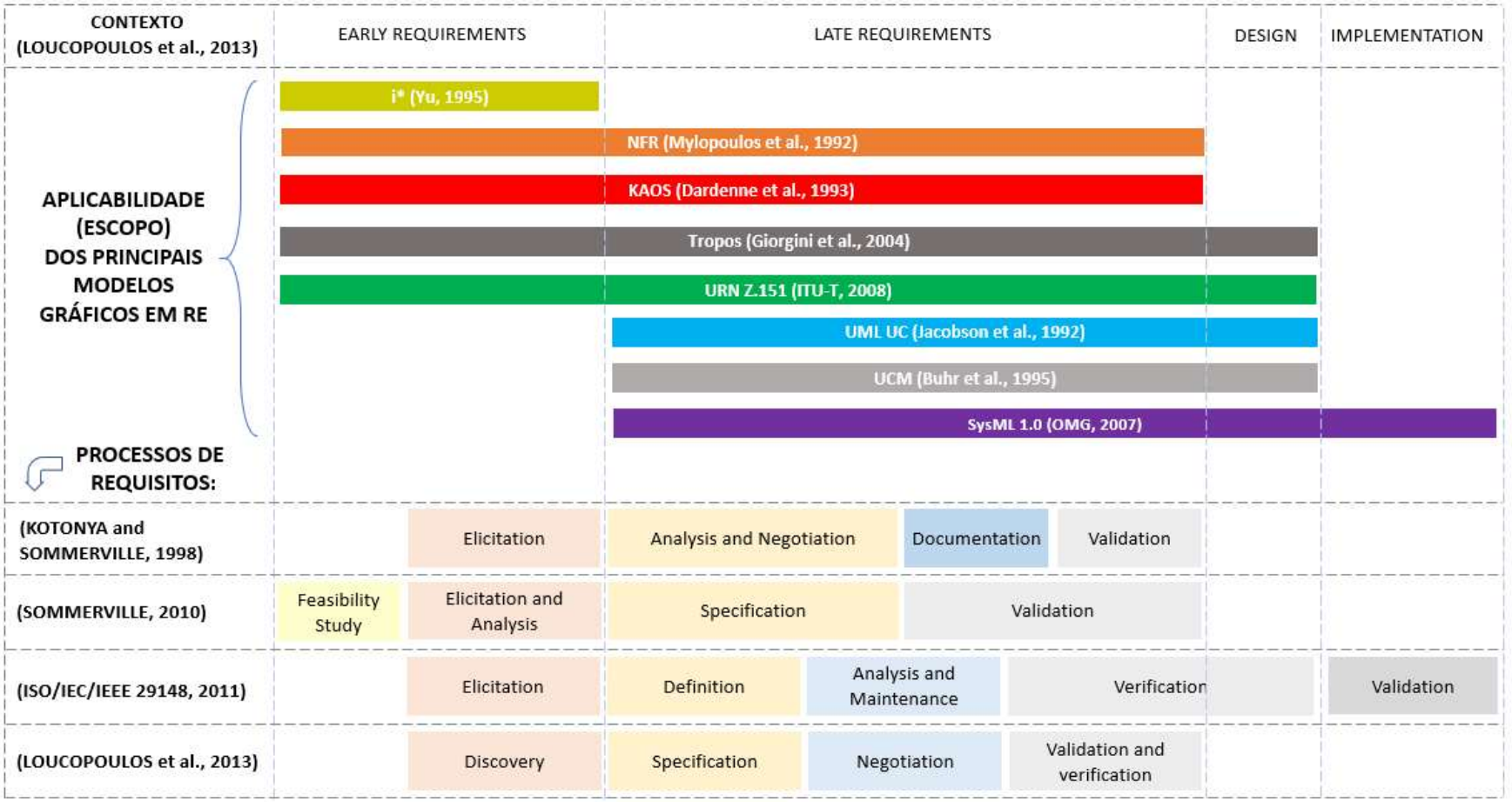


Figura 8 – Aplicabilidade (escopo) dos principais modelos gráficos em RE
Fonte: o autor

2.2 FÍSICA DE NOTAÇÕES – PoN

Apesar de notações visuais serem parte integrante da linguagem de SWE, historicamente pesquisadores desta área e projetistas de notações ignoraram ou subestimaram questões de representação e sintaxe visual. Ao projetar notações, a maior parte do esforço é gasto em estudos semânticos, com convenções gráficas e escolhas de representação visual em grande parte sendo pensadas tardiamente e sem nenhuma justificativa de *design* (MOODY, 2009).

A teoria de "Física de Notações" (*Physics of Notations - PoN*) estabelece um conjunto de princípios de *design* para a criação de notações visuais cognitivamente eficazes e otimizadas para comunicação humana, se concentrando nas propriedades físicas (perceptuais) das notações e não em suas propriedades lógicas (semânticas) (MOODY, 2010).

A Figura 9 contém uma representação genérica de uma notação visual (MOODY et al., 2010), em que a parte da esquerda contém a sintaxe (forma), enquanto que a parte da direita apresenta a semântica (conteúdo). Além disso, a parte superior da Figura 9 apresenta o "nível de tipo", que equivalente a definição da notação de modelagem (ou metamodelo), enquanto que a parte inferior apresenta o "nível de instância", correspondente aos modelos visuais e seus componentes quando instanciados a partir da definição.

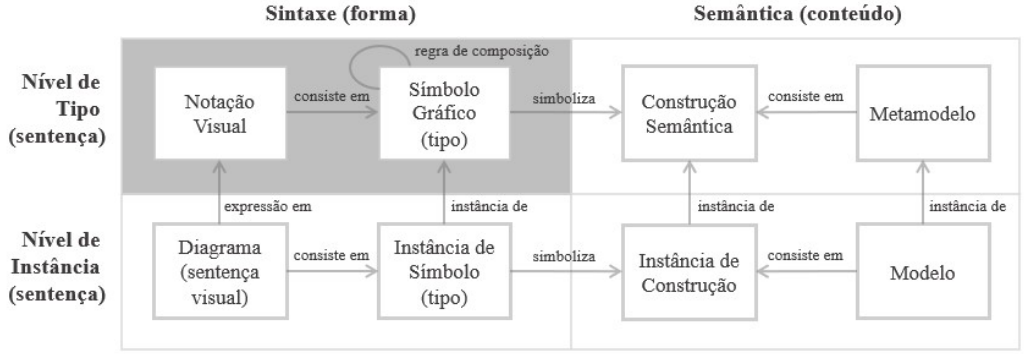


Figura 9 – Escopo da PoN – Quadrante superior direito
Fonte: Adaptado (traduzido) de (MOODY et al., 2010)

A PoN tem como escopo somente a sintaxe visual (*i.e.,* forma), que está situado pelo quadrante superior esquerdo da Figura 9. Nesse quadrante estão representados os conceitos de notação visual (ver seção 2.2.1), que consistem em símbolos gráficos regidos por regras de composição (MOODY, 2009).

2.2.1 Anatomia de uma notação visual

Uma **notação visual** consiste em (MOODY, 2009):

a) **Vocabulário visual**: conjunto de **símbolos gráficos** usados para simbolizar construções semânticas, tipicamente definidas por um metamodelo;
b) **Gramática visual**: conjunto de **regras de composição** que representam como os símbolos gráficos podem se relacionar;
c) **Semântica visual**: definições sobre o **significado** de cada símbolo.

A **sintaxe visual** é composta do vocabulário visual e da gramática visual. Uma expressão válida em uma notação visual é chamada de **sentença visual** ou **diagrama.** Diagramas são compostos de instâncias de símbolos gráficos (*tokens*), arranjados conforme as regras de composição da gramática visual (MOODY, 2009).

2.2.2 Teoria de comunicação de notações visuais em PoN

A teoria de comunicação de PoN corresponde à uma especialização da teoria de comunicação “clássica” (SHANNON e WEAVER, 1963), aplicada ao domínio de notações visuais (MOODY, 2009).

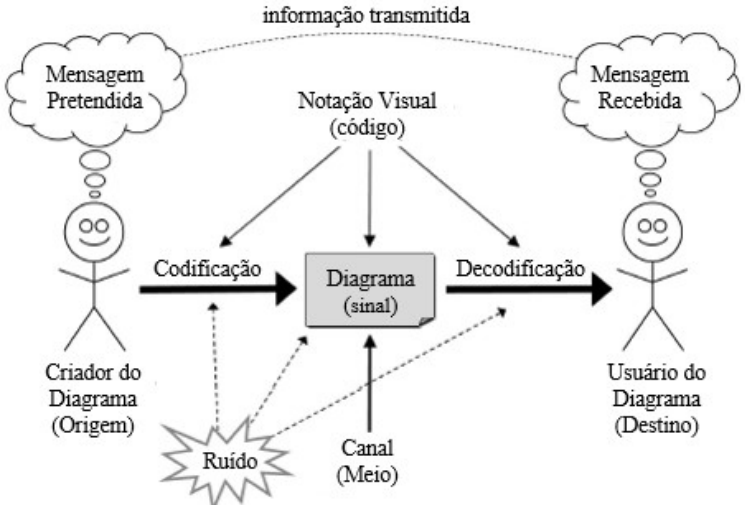


Figura 10 – Teoria da comunicação diagramática de PoN
Fonte: Adaptado (traduzido) de (MOODY, 2009)

Na **teoria de comunicação diagramática** (Figura 10), um criador de diagrama (origem) codifica a mensagem na forma de um diagrama (sinal) e o

usuário do diagrama (destino) decodifica este sinal. O diagrama é codificado usando uma notação visual (código), que define um conjunto de convenções comuns entre criador e usuário. O meio (canal) é a forma física na qual o diagrama é apresentado (*e.g.*: papel, quadro branco e tela do computador). O ruído representa uma variação aleatória no sinal que pode interferir na comunicação. A eficácia da comunicação é medida pela correspondência entre a mensagem pretendida e a mensagem recebida (MOODY, 2009). Nesse contexto, a comunicação consiste em dois processos complementares (MOODY, 2009): a codificação e a decodificação.

A **codificação** (*encoding*), definida pelo espaço de *design* (*design space)*, contém o conjunto de codificações gráficas possíveis para uma dada mensagem (MOODY, 2009). Durante a codificação, há um conjunto de **variáveis visuais** que podem ser utilizadas para codificar a informação graficamente (KOSSLYN et al., 1985). Estas variáveis compõem o chamado **alfabeto visual** (MOODY et. al, 2010), sendo categorizadas em **planares** (as duas dimensões espaciais) e **retinianas** (características da imagem na retina). Esta classificação é mostrada no exemplo na Figura 11.

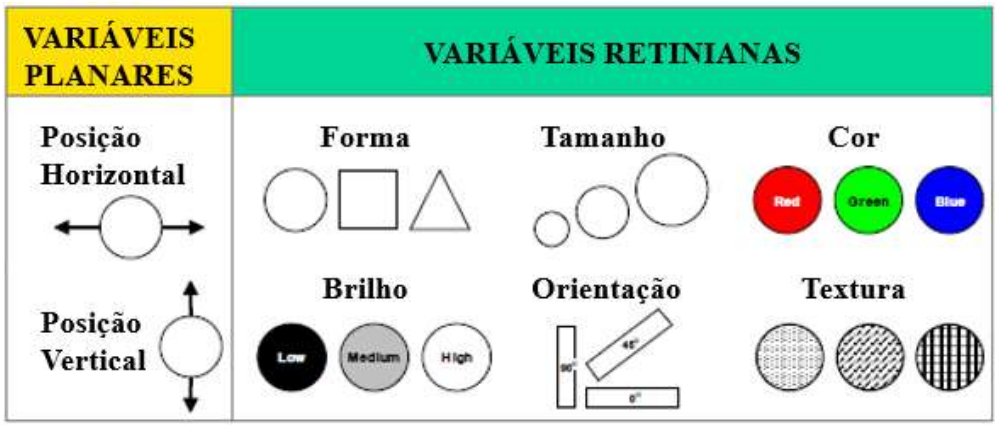


Figura 11 – Variáveis visuais (alfabeto visual)
Fonte: Adaptado (traduzido) de (MOODY et al., 2010)

As variáveis visuais (posição horizontal, posição vertical, forma, tamanho, cor, brilho, orientação e textura) podem ser utilizadas para criar um número praticamente ilimitado de símbolos gráficos para uma determinada notação visual (MOODY et al., 2010).

A **decodificação** (*decoding*), definida pelo espaço de solução (*solution space*), se rege pelos princípios de processamento de informação, percepção e cognição humanas (MOODY, 2009).

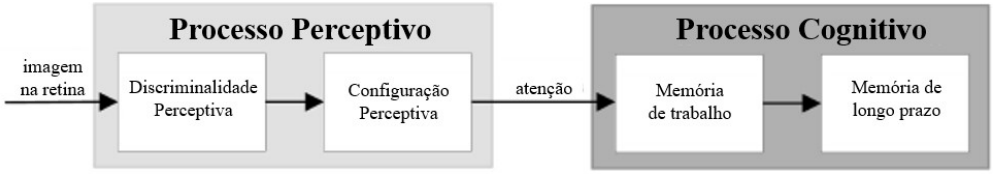


Figura 12 – Modelo de processamento gráfico humano
Fonte: Adaptado (traduzido) de (MOODY, 2009)

A Figura 12 mostra um modelo de processamento de informação gráfica humana, no qual o processamento é dividido em duas fases (MOODY, 2009):

a) **Processo perceptivo** (visão): opera automaticamente, de forma rápida e execução geralmente paralela. A etapa de discriminalidade perceptiva separa os elementos gráficos constituintes da imagem, que então são configurados perceptivamente através da inferência inerente às estruturas e relacionamentos entre elementos;
b) **Processo cognitivo** (compreensão): opera sob o controle consciente da atenção e são relativamente lentos, trabalhosos e sequenciais. A memória de trabalho corresponde à área de "processamento temporário" humano, que reflete o foco corrente de atenção. A cognição das informações de um diagrama visual ocorre quando estas se integram ao conhecimento previamente existente na memória de longo prazo.

2.2.3 Princípios de PoN

A teoria de PoN estabelece um conjunto de princípios (Figura 13) desenvolvidos segundo uma abordagem de síntese das melhores abordagens. É embasada em teorias e evidências empíricas sobre efetividade cognitiva de representações visuais (MOODY, 2009).

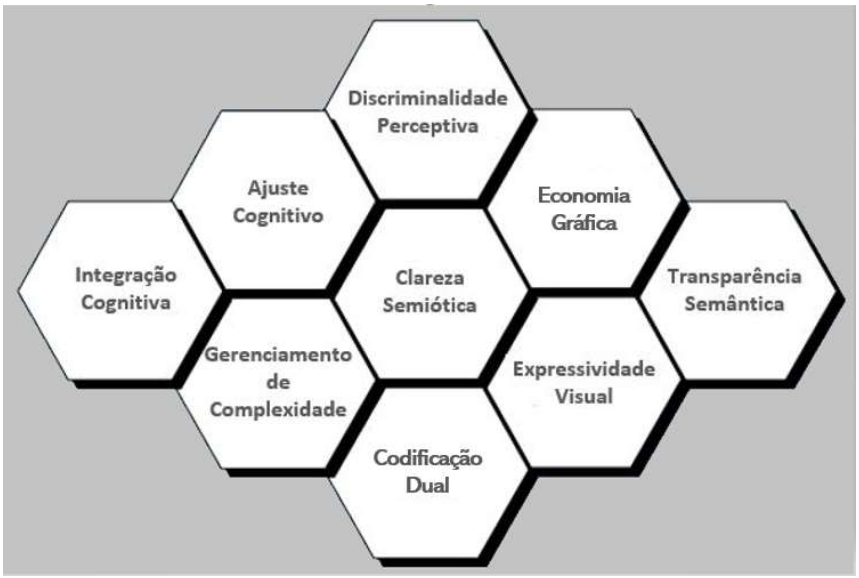


Figura 13 – Princípios de PoN para notações cognitivamente efetivas
Fonte: Adaptado (traduzido) de (MOODY et al., 2009)

Os princípios representados na Figura 13, foram aplicados em diversos estudos sobre notações visuais pré-existentes (em análises de efetividade cognitiva) tais como i*, UML, UCM, BPMN e WebML (ver 2.2.4 sobre trabalhos relacionados). O formato de "favos de mel" simboliza que novos princípios podem ser adicionados com o tempo. Um resumo do significado de cada princípio de PoN é dado na Tabela 2:

Princípio	Descrição (MOODY et al., 2010)
Clareza Semiótica (*Semiotic Clarity*)	Deve haver uma correspondência de 1:1 entre construções semânticas e símbolos gráficos.
Discriminabilidade Perceptiva (*Perceptual Discriminability*)	Símbolos devem ser claramente distinguíveis uns dos outros.
Transparência Semântica (*Semantic Transparency*)	Usar símbolos cuja aparência sugira seu significado.
Gerenciamento de Complexidade (*Complexity Management*)	Incluir mecanismos explícitos para lidar com a complexidade.
Integração Cognitiva (*Cognitive Integration*)	Incluir mecanismos explícitos para integração de informações de diagramas diferentes.
Expressividade Visual (*Visual Expressiveness*)	Usar toda a gama e capacidades de variáveis visuais.

Princípio	Descrição (MOODY et al., 2010)
Codificação Dual (*Dual Coding*)	Usar texto para complementar gráficos.
Economia Gráfica (*Graphic Economy*)	Manter gerenciável cognitivamente o número de diferentes símbolos gráficos.
Ajuste Cognitivo (*Cognitive Fit*)	Usar diferentes dialetos visuais para diferentes tarefas e/ou públicos (audiências).

Tabela 2 – Visão geral dos princípios de física das notações (PoN)
Fonte: Adaptado (traduzido) de (LINDEN et al., 2017)

Uma explicação resumida de cada princípio é dada na sequência (MOODY, 2009) (MOODY et al., 2009) (MOODY et al., 2010):

a) **Clareza Semiótica**: estabelece que deve haver uma correspondência de um para um (1:1) entre construções semânticas e símbolos gráficos usados em uma notação. Quando não houver esta correspondência, podem ocorrer as anomalias mostradas na Figura 14: déficit de símbolos, redundância de símbolos, sobrecarga de símbolos e excesso de símbolos (MOODY et al., 2009).

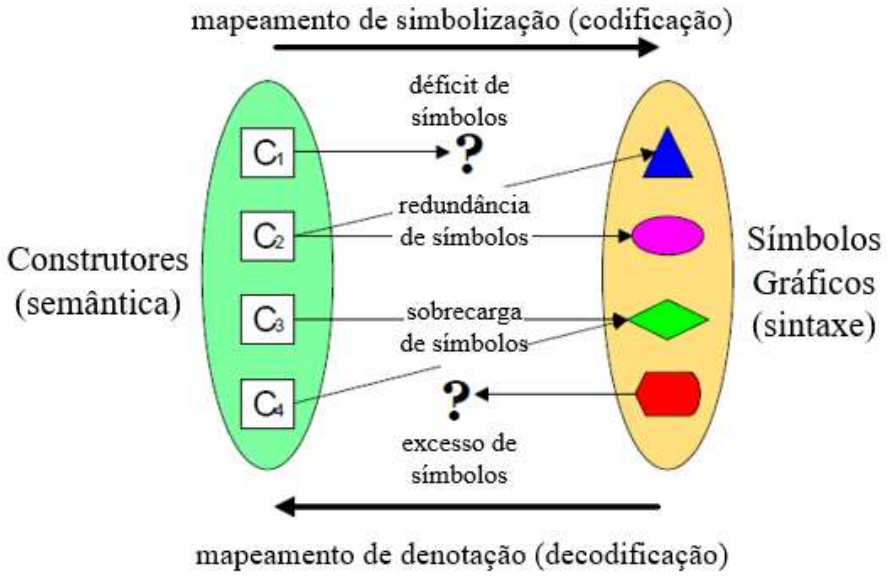


Figura 14 – Princípio PoN da clareza semiótica e suas anomalias
Fonte: Adaptado (traduzido) de (MOODY et al., 2009)

A avaliação da clareza semiótica em uma notação envolve conduzir um mapeamento entre o metamodelo da notação (construtores semânticos)

e o conjunto de símbolos, de forma a verificar a correspondência 1:1 entre estes elementos (MOODY et al., 2009).

b) **Discriminalidade Perceptiva**: refere-se à facilidade e precisão com que os símbolos podem ser diferenciados uns dos outros, o que é um pré-requisito para interpretação precisa de diagramas (WINN, 1990). A discriminação é determinada pela distância visual entre os símbolos, que é medida pelo número de **variáveis visuais** (forma, tamanho, cor, brilho etc.) nas quais os símbolos diferem e o tamanho dessas diferenças. Em geral, quanto maior a distância visual entre os símbolos, mais rapidamente e com mais precisão eles serão reconhecidos (WINN, 1993). Se as diferenças forem muito sutis, erros de interpretação podem resultar (MOODY et al., 2010).
c) **Transparência Semântica**: este princípio requer que os símbolos gráficos da notação forneçam pistas para o seu significado, de forma a melhorar a velocidade e precisão do entendimento, especialmente por usuários iniciantes (MOODY et al., 2010). Símbolos semanticamente transparentes reduzem a carga cognitiva porque possuem mnemônicos embutidos, tornando mais fácil aprender e lembrar o que eles significam (PETRE, 1995). Na Figura 15, é possível observar diferentes graus possíveis de transparência semântica.

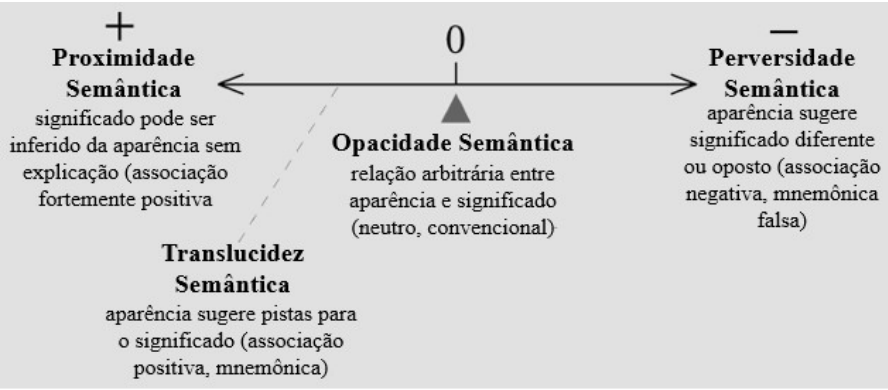


Figura 15 – Associação entre forma e conteúdo (transparência semântica)
Fonte: Adaptado (traduzido) de (MOODY, 2009)

Essa gradação pode variar desde a **proximidade semântica** (forma e conteúdo são próximos, com associação fortemente positiva) até a **perversidade semântica** (forma e conteúdo são opostos, com

associação fortemente negativa), passando por graus intermediários como a **translucidez semântica** (há alguma proximidade entre forma e conteúdo) e a **opacidade semântica** (na qual a relação entre forma e conteúdo é arbitrária ou neutra) (MOODY, 2009).

d) **Gerenciamento de Complexidade**: refere-se à capacidade de uma notação visual representar informações sem sobrecarregar a mente humana. Neste contexto, "complexidade" refere-se à complexidade diagramática, que é medida pelo número de elementos (instâncias de símbolo ou *tokens*) em um diagrama. Embora isso seja aparentemente um problema de nível de diagrama (sentença), ele requer recursos notacionais para resolvê-lo. A complexidade tem um efeito importante sobre a eficácia cognitiva, pois a quantidade de informação que pode ser transmitida de forma eficaz por um único diagrama é limitada pelas capacidades perceptivas e cognitivas humanas (MOODY, 2009).

   Para representar efetivamente situações complexas, notações visuais devem fornecer mecanismos de **modularização** e **estruturação hierárquica**, que correspondem a subsistemas e estruturas de nível na teoria ontológica (WEBER, 1997). Apesar da teoria ontológica definir construções semânticas necessárias para suportar o gerenciamento de complexidade, ela não define uma solução sintática para o problema (MOODY, 2009).

e) **Integração Cognitiva**: este princípio aplica-se somente quando vários diagramas são usados para representar o sistema. Aplica-se igualmente a diagramas do mesmo tipo (integração homogênea) ou diagramas de tipos diferentes (integração heterogênea) (MOODY, 2009).

   De acordo com a teoria de integração cognitiva de diagramas (KIM et al., 2000), para que as representações de vários diagramas de SWE sejam cognitivamente eficazes, elas devem incluir mecanismos explícitos de:

   - **integração conceitual** para permitir que usuários integrem informações de diagramas separados em uma representação mental única do sistema;
   - **integração perceptual** através de pistas perceptivas simplificadoras da navegação e transições entre diagramas. (MOODY et al., 2010).

f) **Expressividade Visual**: é definida pelo número de diferentes variáveis visuais utilizadas e gama de valores utilizada de cada uma (capacidade). O uso de uma maior variedade de variáveis visuais (*e.g.*: posição horizontal, posição vertical, forma, tamanho, cor, brilho, orientação e textura) resulta em uma representação perceptivelmente enriquecida que maximiza a eficácia cognitiva (MOODY et al., 2009).

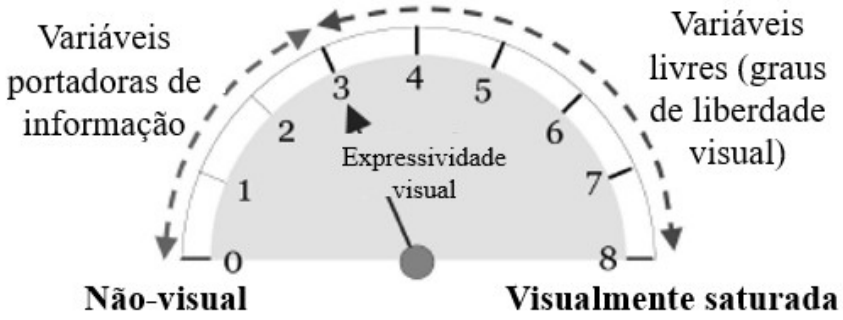


Figura 16 – Expressividade visual em PoN
Fonte: Adaptado (traduzido) de (MOODY, 2009)

Na Figura 16 é mostrada uma representação da escala de utilização das variáveis visuais. O número de variáveis livres é chamado de **graus de liberdade visual** e é o inverso da expressividade visual. Assim:

- Uma notação **não-visual** (ou textual) não possui variáveis visuais portadoras de informação, sendo que sua expressividade visual é igual a zero e possui oito graus de liberdade visual;
- Uma notação **visualmente saturada** usa todas as variáveis visuais sendo que sua expressividade visual é igual a oito e possui zero graus de liberdade visual. (MOODY, 2009).

g) **Codificação Dual**: os princípios de discriminabilidade perceptiva e expressividade visual desaconselham o uso de texto para codificar informações em notações visuais. No entanto, isso não significa que o texto não tenha lugar no design das notações (MOODY et al., 2010).

De acordo com a teoria de codificação dual (PAIVIO, 1990), usar textos e gráficos juntos para transmitir informações é mais eficaz do que usar apenas um deles. Quando informações são simultaneamente apresentadas textual e graficamente, suas representações são codificadas em sistemas separados na memória de trabalho e as conexões referenciais entre as duas são reforçadas (MOODY, 2009).

h) **Economia Gráfica**: complexidade gráfica refere-se ao número de símbolos diferentes usados em uma notação, ou o tamanho do seu vocabulário visual (MOODY et al., 2009). Estudos empíricos mostram que a complexidade gráfica pode reduzir significativamente a compreensão dos diagramas de RE, especialmente por usuários novatos (NORDBOTTEN e CROSBY, 1999). A razão disto é que a capacidade humana de discriminar perceptivamente entre alternativas distintas é de cerca de 6 categorias (extensão de julgamento absoluto) (MILLER, 1994), o que define um limite superior efetivo para complexidade gráfica.

   A economia gráfica pode ser atingida através de três estratégias principais que permite reduzir a complexidade gráfica excessiva (MOODY, 2009):

   - Reduzir ou particionar a complexidade semântica: reduzir a quantidade de símbolos e sua respectiva semântica leva à redução da complexidade gráfica;
   - Introduzir déficit de símbolos: alguns símbolos (e a correspondente semântica) deixam de ser representados graficamente e passam a ser representados em texto;
   - Aumentar a expressividade visual: utilizar diferentes variáveis visuais para aumentar a expressividade visual (ao invés de acrescentar novos símbolos / semântica associada).

i) **Ajuste Cognitivo**: este princípio preconiza que diferentes representações de informações são adequadas para diferentes tarefas e diferentes públicos (VESSEY e GALLETTA, 1992). O desempenho na resolução de problemas (que corresponde aproximadamente à eficácia cognitiva) é determinado por um ajuste de três vias entre a representação do problema, as características da tarefa e as habilidades do solucionador de problemas (MOODY, 2009).

   A maioria das notações visuais de SWE exibe um monolinguismo visual: usam uma única representação visual para todos os propósitos. O princípio de ajuste cognitivo, no entanto, advoga a necessidade de desenvolver diferentes dialetos visuais para atender diferentes audiências (*e.g.*: usuários especialistas x usuários novatos). Uma

proposta de aplicação deste princípio seria por exemplo a criação de pelo menos dois dialetos visuais diferentes em uma notação: um para o especialista ("*pro*") e um para o novato ("*lite*").

2.2.4 Aplicações e trabalhos relacionados a PoN

A teoria de PoN pode ser aplicada a diferentes notações visuais da área de SWE. Desde sua criação, em torno de 70 notações visuais diferentes foram analisadas (em maior ou menor grau) sob o ponto de vista da teoria de PoN (LINDEN e HADAR, 2018), dentre os quais destacam-se alguns estudos sobre efetividade cognitiva em modelos gráficos (notações visuais) de RE:

a) **i*** *(i-star)*: linguagem visual orientada a objetivos, usada para modelagem organizacional e de requisitos (*early requirements*) (MOODY et al., 2009) (MOODY et al., 2010) (GENON e al., 2012) (CAIRE et al., 2013);
b) **UML** (*Unified Modeling Language*): linguagem visual de modelagem de projetos de *software* e padrão OMG (*Object Management Group*) (MOODY e HILLEGERSBERG, 2009);
c) **UCM** (*Use Case Maps*): linguagem visual de modelagem de cenários, parte da notação URN (*User Requirements Notation*), que visa à elicitação, análise, especificação e validação de requisitos (GENON et al., 2010);
d) **BPMN** (*Business Process Modeling Language*): linguagem visual de modelagem de processos de negócio (GENON et al., 2011);
e) **URML** (*Unified Requirements Modeling Language*): linguagem visual proposta segundo princípios da PoN (HELMING et al., 2010) e refinada como uma extensão de profile UML para realizar a modelagem de requisitos industriais (BERENBACH et al., 2012). Posteriormente, foi materializada como um *Add-In* da ferramenta *Enterprise Architect*® (SCHNEIDER et al., 2013).

Maiores referências e informações sobre as notações visuais citadas acima podem ser encontradas na seção 2.3 desta dissertação.

2.2.5 O desenvolvimento de notações visuais baseada em PoN

A teoria de PoN contém princípios prescritivos, que podem ser usados para avaliar, comparar e melhorar as notações visuais existentes, bem como para construir novas. Isso significa que, em vez de considerar infinitas possibilidades ao criar uma notação visual, pode-se optar por aquelas possibilidades que melhor atendem a PoN (LINDEN et al., 2016).

Como críticas a esta teoria, é possível apontar que há poucas diretrizes concretas sobre a operacionalização prática dos princípios de PoN e que sua formulação contém diretrizes informais, embora bem descritas e completas (LINDEN et al., 2016). Questionou-se, também, se os princípios de PoN podem ser verificados de maneira replicável e sistemática, com a argumentação de que eles não são nem precisos nem abrangentes o suficiente para serem aplicados de uma maneira objetiva para analisar notações práticas de SWE (STÖRLLE e FISH, 2013).

Para sistematizar a utilização de PoN e responder as críticas colocadas, alguns estudos foram realizados, tais como PoN-S (PoN *systematized*) (TEIXEIRA et al., 2016) e o *framework* para melhorar a verificabilidade de PoN em notações visuais (LINDEN et al., 2017).

A abordagem PON-S apresenta um processo de *design* sistematizado (PoN-S) que estabelece as atividades a serem realizadas, sua conexão com os princípios de PoN, bem como os critérios para agrupar os princípios que orientam o processo (TEIXEIRA et al., 2016). Apesar de apresentar um processo que realiza o sequenciamento de etapas para aplicar os princípios PoN, não há um detalhamento claro de como aplicar cada princípio individualmente. Por essa razão, essa abordagem não foi aplicada no presente trabalho.

O *framework* de verificabilidade de PoN, por outro lado, fornece uma abordagem sistemática para aplicar individualmente os princípios de PoN, através da proposição de um modelo focado nos requisitos da linguagem e definição da razão do *design* (LINDEN et al., 2017). A **razão do *design*** (*design rationale)* é o processo de documentar as decisões de *design* feitas e as razões pelas quais elas foram feitas, visando permitir rastreabilidade no processo de *design* e ajudar a justificar o *design* final (LEE, 1997). Tal raciocínio é notavelmente ausente no projeto de notações visuais em SWE: convenções

gráficas são tipicamente definidas sem referência à teoria ou evidência empírica, ou justificativas de qualquer tipo (MOODY, 2009).

A Tabela 3 apresenta a proposta de um perfil de captura (*capture profile*) de informações de *design rationale* vinculada a requisitos da linguagem de notação visual e aos princípios de PoN (LINDEN et al., 2017).

Título	Título descritivo
Princípio	Princípio ao qual a escolha do *design* está relacionada = {Clareza Semiótica, Discriminabilidade Perceptiva, Transparência Semântica, Gerenciamento de Complexidade, Integração Cognitiva, Expressividade Visual, Codificação Dual, Economia Gráfica, Ajuste Cognitivo}.
Símbolo gráfico	Construtor visual para o qual a escolha do *design* é feita.
Propriedade visual	Coleção de variáveis visuais do símbolo relacionado a escolha do *design* = {Forma, Cor, Textura, Tamanho, Brilho, Orientação, Posição horizontal (x), Posição vertical (y)}
Escolha de *design*	Descrição textual da escolha de *design* realizada.
Justificativa	Descrição textual da justificativa e evidências que a sustentam.
Requisito	Requisitos explícitos a partir dos quais a escolha de *design* foi feita (se houver).
Evidência	O nível de confiança da evidência que sustenta a lógica da escolha de *design* = {alta, moderada, baixa, muito baixa}.

Tabela 3 – Perfil de captura para fundamentos de escolha de *design*
Fonte: Adaptado (traduzido) de (LINDEN et al., 2017)

Para cada símbolo gráfico da notação visual (ou para todos simultaneamente) em processo de projeto sugere-se a criação do perfil de captura correspondente (Tabela 3), como forma de documentar as justificativas de projeto (*design rationale)* e permitir a verificabilidade posterior (LINDEN et al., 2017). Um exemplo da aplicação deste perfil é dado na Tabela 4.

Título	Representar agentes como James Bond
Princípio	Transparência Semântica
Símbolo gráfico	Agente

Título	Representar agentes como James Bond
Propriedade visual	Forma
Escolha de *design*	Representar o agente como um boneco usando óculos escuros e segurando uma pistola. 
Justificativa	A figura proposta fará as pessoas associarem o símbolo a “James Bond”, um famoso “agente secreto”.
Requisito	A notação deve ser rica (*sic*) a ponto de permitir a comunicação com pessoas externas.
Evidência	Muito baixa (avaliação de especialista), baseando-se em observação do autor

Tabela 4 – Exemplo de aplicação de perfil de captura para um caso específico
Fonte: Adaptado (traduzido) de (LINDEN et al., 2017)

O seguinte método para verificação do modelo baseado em relatórios de verificação (*checklists)* de requisitos da notação visual é sugerido (LINDEN et al., 2017):

a) Verificar requisitos (ou diretrizes) específicos da notação

As seguintes etapas devem ser realizadas:

- Listar os requisitos que impactam a implementação;
- Listar: S = construções semânticas;
  V = construções visuais = (Entidades U Relacionamentos);
  M = mapeamento entre S e V, M: S→V.
- Listar todas as variáveis visuais usadas e seus possíveis valores usados nos elementos visuais V;
- Listar todas as regras de composição R usadas para S e V;
- Verificar a notação segundo o relatório da Tabela 5, que apresenta locais específicos em formato de quadrados (“□”), para marcar os itens a serem verificados (formato *checklist*).

Notação
Assegurar-se de que as seguintes informações sejam relatadas. Requisitos: □ S: □, V: □, M: S→V: □, VisVar: □, R: □

Tabela 5 – Relatório de verificação de requisitos específicos da notação
Fonte: Adaptado (traduzido) de (LINDEN et al., 2017)

b) Verificar requisitos de princípios PoN

Cada princípio de PoN deve ser reportado com base nas escolhas de *design* criadas (modelo Tabela 3) e conforme os relatórios modelo propostos (da Tabela 6 até a Tabela 14).

Clareza Semiótica	
Calcule o valor do símbolo…	
… redundância =	$\lvert\{v \in V \mid \lvert\{s \in S: M(s) = v\}\rvert > 1\}\rvert$ [ ]
... sobrecarga =	$\lvert\{s \in S \mid \lvert\{v \in V: M(s) = v\}\rvert > 1\}\rvert$ [ ]
... excesso =	$\lvert\{v \in V \mid \neg\exists s \in S: M(s) = v\}\rvert$ [ ]
... déficit =	$\lvert\{s \in S \mid \neg\exists v \in V: M(s) = v\}\rvert$ [ ]
Para qualquer valor diferente de zero: fornecer um *design rationale* estruturado (segundo Tabela 3).	

Tabela 6 – Modelo de relatório de verificação PoN: Clareza Semiótica
Fonte: Adaptado (traduzido) de (LINDEN et al., 2017)

Discriminalidade Perceptiva
Escolher uma métrica *Sim* que para cada duas formas *s* e *s'* de construtores visuais *v* e *v'* retorna o seu score de similaridade: [ ]
Escolher um limite de dissimilaridade D. Se *Sim*(*s, s'*) < D → então *s* e *s'* não são semelhantes. [ ]
Fornecer detalhes para formas *s* e s' de diferentes construções visuais *v* e *v'*:
<u>Semelhança de forma</u>:
Fornecer *Sim*(*s, s'*) = [ ]
Se *Sim*(*s, s'*) >= D [ ]:
Fornecer um *design rationale* estruturado: [ ]
<u>Inconsistência de forma</u>:
Se *s* é um subtipo de *s'* e *Sim*(*s,s'*) <D:
Fornecer um *design rationale* estruturado: [ ]
<u>Discriminabilidade dos relacionamentos:</u>
Se *v* e *v'* são relacionamentos e *Sim*(*s,s'*) >= D:
Fornecer um *design rationale* estruturado: [ ]

Tabela 7 – Modelo de relatório de verificação PoN: Discriminalidade Perceptiva
Fonte: Adaptado (traduzido) de (LINDEN et al., 2017)

Transparência Semântica
Para cada símbolo visual *v* ϵ V:
Fornecer a localização de *v* na escala Transparência da Figura 15 e fornecer um *design rationale* estruturado: [ ]

Tabela 8 – Modelo de relatório de verificação PoN: Transparência Semântica
Fonte: Adaptado (traduzido) de (LINDEN et al., 2017)

Gerenciamento de Complexidade
Fornecer o nº. de tipos de diagrama que têm níveis diferentes de abstração. [ ]
Há pelo menos um nível de abstração usado para decomposição recursiva de diagramas (*i.e.*, *black-boxing*, (de)composição para novos diagramas)?
Sim [ ] Não [ ]
Se o número de níveis de abstração for zero ou decomposição recursiva não está disponível: Fornecer um *design rationale* estruturado: [ ]

Tabela 9 – Modelo de relatório de verificação PoN: Gerenciam. de Complexidade
Fonte: Adaptado (traduzido) de (LINDEN et al., 2017)

Integração Cognitiva
Se houver vários tipos de tipos de diagrama (|Tipos| > 1):
Para qualquer caso em que um *s* é mapeado para diferentes *v* em diagramas diferentes dos tipos t1, t2:
Fornecer um *design rationale* estruturado: [ ]
Aplicar a integração cognitiva da teoria de diagramas conforme requisitos:
Fornecer um *design rationale* estruturado: [ ]

Tabela 10 – Modelo de relatório de verificação PoN: Integração Cognitiva
Fonte: Adaptado (traduzido) de (LINDEN et al., 2017)

Expressividade Visual
Fornecer | Variáveis Visuais | = [ ]
A quantidade de | Variáveis Visuais | = 8? Sim [ ] Não [ ]
Para as variáveis visuais selecionadas *x* ϵ | Variáveis Visuais |:
Fornecer um *design rationale* estruturado, abordando até que ponto as variáveis visuais x selecionadas contribuem para:
(i) Discriminalidade Perceptiva (ii) Legibilidade, e (iii) Estética: [ ]

Tabela 11 – Modelo de relatório de verificação PoN: Expressividade Visual
Fonte: Adaptado (traduzido) de (LINDEN et al., 2017)

Codificação Dual
Aplicar a teoria de codificação dupla de acordo com os requisitos da notação. Fornecer um *design rationale* estruturado para a implementação e como os requisitos são endereçados: [ ]

Tabela 12 – Modelo de relatório de verificação PoN: Codificação Dual
Fonte: Adaptado (traduzido) de (LINDEN et al., 2017)

Economia Gráfica
Fornecer | V | = [ ] Se | *Tipos* | = 1 e | V | não é 7 ± 2: Fornecer um *design rationale* estruturado: [ ] Se | *Tipos* | > 1, para cada tipo *t*: Vt = [ ] (em que Vt é o conjunto V usado em diagramas do tipo t) Fornecer | Vt | para cada t ϵ | Tipos |: [ ] Se | Vt | não é 7 ± 2: Fornecer um *design rationale* estruturado: [ ]

Tabela 13 – Modelo de relatório de verificação PoN: Economia Gráfica
Fonte: Adaptado (traduzido) de (LINDEN et al., 2017)

Ajuste Cognitivo
Fornecer dialetos visuais de acordo com os requisitos da notação. Se relevante para os requisitos da notação, fornecer um dialeto para pelo menos diferentes níveis de especialização e diferentes meios de representação. Se nenhum dialeto for fornecido e seus requisitos indicarem usuários com diferentes níveis de especialização, ou se for possível usar em diferentes mídias de representação: Fornecer um *design rationale* estruturado: [ ]

Tabela 14 – Modelo de relatório de verificação PoN: Ajuste Cognitivo
Fonte: Adaptado (traduzido) de (LINDEN et al., 2017)

A presente subseção apresentou um conjunto de relatórios para verificação de notações visuais que sejam desenvolvidas segundo princípios PoN (LINDEN et al., 2017). A seção 3.1 deste trabalho mostra como estes relatórios de verificabilidade podem ser integrados a um método para o desenvolvimento e verificação de notações visuais de requisitos.

2.3 MODELOS GRÁFICOS EM ENGENHARIA DE REQUISITOS

A representação de requisitos para *softwares* e sistemas pode ser realizada de múltiplas formas e com várias notações, tais como linguagem natural, linguagens formais ou semi-formais, diagramas, gráficos e tabelas. A linguagem natural e tabelas de requisitos são formatos usuais para representar requisitos na indústria, apesar das desvantagens em termos de precisão, compreensão e visualização. Para superar estes problemas, modelos gráficos têm sido frequentemente propostos por pesquisadores da área de RE.

Os resultados apresentados nesta seção contêm uma síntese de informações relativas a modelagens gráficas propostas por diferentes pesquisadores da área de RE. As seguintes informações relativas às abordagens de modelagem de requisitos são sistematicamente identificadas: representações gráficas propostas, linguagens de especificação, notações, ferramentas de modelagem, metodologias, normas internacionais e metamodelos (ou ontologias).

A identificação dos modelos gráficos de requisitos adotou uma metodologia de **mapeamento sistemático da literatura** (*Systematic Literature Mapping* – SLM). Essa metodologia foi adaptada a partir de ideias e princípios propostos por pesquisadores de “Engenharia de *Software* baseada em experimentos” (*Empirical Based Software Engineering* - EBSE) (KITCHENHAM et al., 2004) (PETERSEN et al., 2008) (KITCHENHAM et al., 2011).

O objetivo principal de um SLM é prover uma visão geral da pesquisa em determinada área, de forma a identificar a quantidade e o tipo de resultados da área (PETERSEN et al., 2008). O processo genérico original proposto por Petersen (Figura 17) inspirou o processo adotado por este SLM.

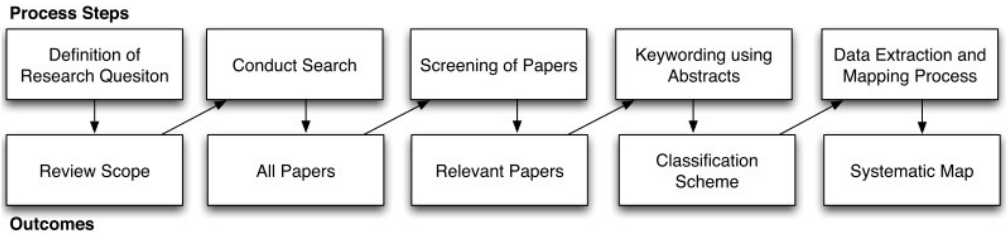


Figura 17 – O processo de mapeamento sistemático
Fonte: (PETERSEN et al., 2008).

Dentre os benefícios deste tipo de estudo pode-se destacar (KITCHENHAM et al., 2011):

a) Fornecer um ponto de partida para pesquisadores que precisem organizar e entender o trabalho de pesquisa em determinado domínio;
b) A longo prazo, permite prover a próxima geração de pesquisadores com um corpo de conhecimento como ponto de partida (*starting point*) para suas pesquisas ao invés de forçar todo pesquisador a iniciar do zero.

As próximas seções apresentam resultados quantitativos e qualitativos visando responder questões de pesquisa (Q1 a Q9) colocadas para o SLM realizado (APÊNDICE A - Metodologia), obtidos a partir da sumarização das informações do APÊNDICE B sobre descrição dos modelos, da seguinte forma:

Seção 2.3.1: resumos quantitativos (tabelas e gráficos) relativos aos dados obtidos neste estudo (questões Q1, Q2);

Seção 2.3.2: linguagens de especificação (questão Q3);

Seção 2.3.3: ferramentas de modelagem (questão Q4);

Seção 2.3.4: metodologias ou abordagens (questão Q5);

Seção 2.3.5: normas internacionais (questão Q6);

Seção 2.3.6: metamodelos ou ontologias (questão Q7).

Seção 2.3.7: evolução temporal das abordagens (questão Q8).

Seção 2.3.8: comparação de características dos modelos (questão Q9).

No APÊNDICE A é apresentada a **metodologia** adotada neste SLM sobre modelos gráficos em RE, realizada em consonância com os princípios supracitados (PETERSEN et al., 2008) (KITCHENHAM et al., 2011).

O APÊNDICE B contém a descrição padronizada dos modelos gráficos em RE. Este apêndice visa responder de qualitativamente as questões Q1 e Q2 propostas na metodologia de pesquisa do SLM (APÊNDICE A).

No APÊNDICE C estão referenciados os **artigos selecionados** no processo de SLM (artigos principais, prefixados com a letra “P”), bem como artigos complementares (prefixados com a letra “C”) utilizados para obter informações adicionais sobre os modelos gráficos encontrados no SLM.

2.3.1 Resumos quantitativos sobre modelos identificados

Nesta seção são apresentados resumos quantitativos e gráficos sobre quantidades de identificações de modelos nos artigos analisados. Na Tabela 15, é apresentada a lista dos modelos identificados, o tipo de modelo, a quantidade de artigos em que o modelo foi identificado e o código de referência associado a cada artigo (disponíveis no APÊNDICE C).

N	Modelo	Tipo de Modelo	Quantidade de Artigos (identific.)	Artigos em que o modelo foi identificado
01	i*	Objetivos	55	P011, P019, P020, P021, P022, P024, P026, P027, P029, P030, P032, P033, P035, P036, P047, P048, P049, P050, P052, P053, P061, P062, P063, P064, P065, P066, P067, P068, P069, P070, P071, P072, P073, P074, P075, P076, P077, P078, P079, P080, P081, P082, P083, P084, P085, P086, P087, P088, P089, P090, P091, P092, P093, P107, P194
02	Tropos	Objetivos	26	P019, P023, P033, P036, P054, P055, P056, P057, P058, P059, P060, P135, P136, P152, P153, P154, P155, P156, P157, P158, P159, P161, P162, P189, P190, P193
03	KAOS	Objetivos	25	P010, P014, P016, P019, P025, P028, P031, P032, P034, P035, P036, P061, P088, P095, P096, P097, P098, P099, P100, P101, P102, P103, P104, P107, P108
04	NFR	Objetivos	25	P017, P018, P019, P022, P025, P035, P036, P037, P038, P039, P051, P053, P107, P109, P110, P111, P112, P113, P114, P115, P116, P117, P118, P119, P160
05	URN	Objetivos	24	P002, P006, P007, P021, P035, P042, P043, P044, P045, P168, P169, P170, P171, P172, P173, P174, P175, P176, P177, P178, P179, P180, P181, P182
06	SysML	Hierárquico	12	P139, P140, P141, P142, P143, P144, P145, P146, P147, P148, P149, P150

N	Modelo	Tipo de Modelo	Quantidade de Artigos (identific.)	Artigos em que o modelo foi identificado
07	UML UC	Cenário	7	P013, P091, P092, P093, P186, P187, P188
08	UCM	Cenário	5	P163, P164, P183, P184, P185
09	PF	Problem Frames	5	P107, P122, P124, P125, P142
10	V-Graph	Objetivos	3	P105, P106, P121, P195
11	RecVisu	Cluster	3	P040, P041, P046
12	Techne	Objetivos	2	P131, P151
13	AMORE	Multimídia	2	P004, P005
14	URML	Grafo	2	P166, P167
15	UML RD	Hierárquico	1	P165
16	VLML	Processo	1	P127
17	OPM	Hierárquico	1	P137
18	RDN	Grafo	1	P001
19	AGORA	Objetivos	1	P003
20	ATHENA	Grafo	1	P008
21	Archimate	Grafo	1	P128
22	IRD	Grafo	1	P094
23	RCAT	Grafo	1	P126
24	RIG	Grafo	1	P129
25	RDG	Grafo	1	P138
26	VRDL	Multimídia	1	P192
27	RON	Grafo	1	P196
		TOTAL	210	

Tabela 15 – Resumo quantitativo (Modelos x Aparecimento em artigos)
Fonte: o autor

Na Tabela 15, o total de identificações de modelos em artigos é de **210**, enquanto a quantidade de trabalhos analisados é de **185 artigos**. Isso ocorre porque há trabalhos em que vários modelos aparecem simultaneamente, tais como: comparação entre vários modelos; integração entre modelos; e transformações de um modelo para outro.

A Figura 18 apresenta o percentual de identificações de cada modelo em relação ao total de aparecimentos nos artigos selecionados neste SLM. Os modelos que apareceram com maior frequência (e.g.: i*, Tropos, KAOS, NFR, URN, SysML) provavelmente são os mais pesquisados na área de RE segundo os dados e período de análise considerados neste estudo (1998 e 2018).

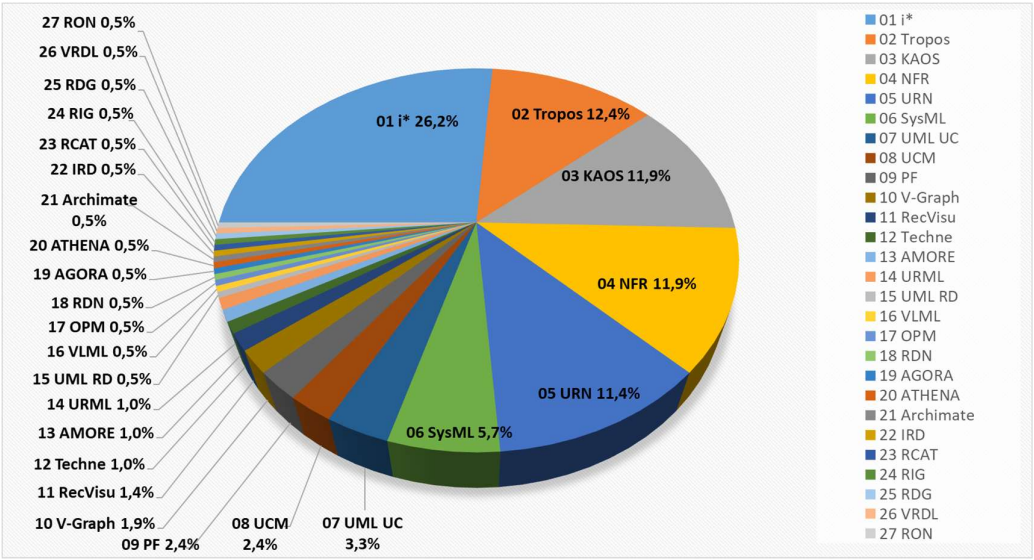


Figura 18 – Quantidade % de identificações de modelos gráficos
Fonte: o autor

Para quantificar a frequência de identificação das abordagens de modelagem foi elaborada a Tabela 16, na qual foi aplicada o bem conhecido princípio de pareto ou regra dos 80/20 (*e.g.* em SWE: (PANDEY et al., 2013)). Nesta tabela percebe-se que 6 modelos (22% do total de 27 modelos) representam em torno 80% das identificações ocorridas. Os 6 “principais” modelos (i*, Tropos, KAOS, NFR, URN, SysML) correspondem aos casos em que houve o mínimo de 10 identificações (ou pelo menos 5% do total).

N	Modelo	Quantidade de Identificações	% de Identificações	% Acumulado de Identificações
01	i*	55	26,19%	26,19%
02	Tropos	26	12,38%	38,57%
03	KAOS	25	11,90%	50,48%
04	NFR	25	11,90%	62,38%
05	URN	24	11,43%	73,81%
06	SysML	12	5,71%	79,52%
07	Outros	43	20,48%	100,00%
	Total	210	100,00%	

Tabela 16 – Modelos com % de identificações superior a 5%
Fonte: o autor

O gráfico da Figura 19 apresenta a quantificação de identificação de artigos em relação ao total, realizada de acordo com o tipo de modelo (Cenário, *Cluster*, Objetivos, Grafo, Hierárquico, Multimídia, *Problem Frames*, Processo).

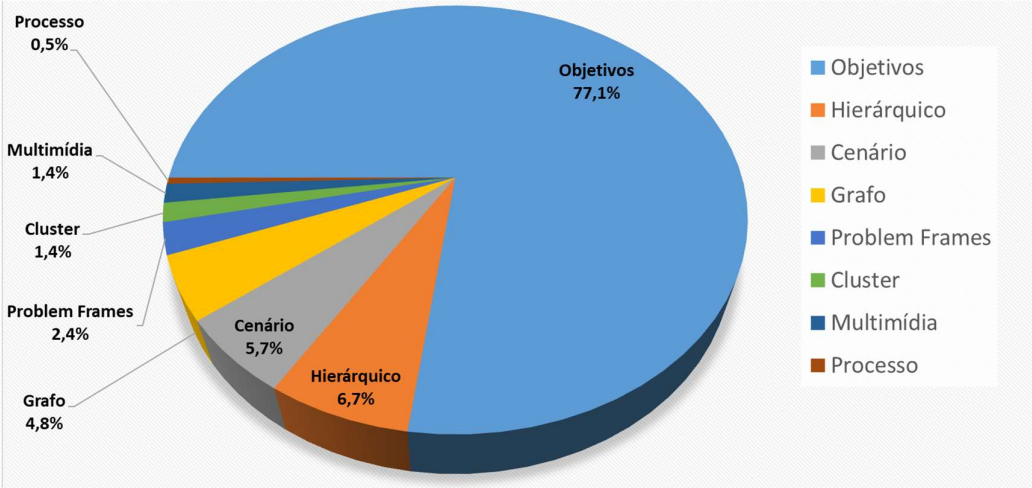


Figura 19 – Quantidade % de identificações de tipos de modelos gráficos
Fonte: o autor

A análise do gráfico da Figura 19 permite constatar que modelagens gráficas orientadas a objetivos (*goal based*) e suas derivações foram as mais pesquisadas (em torno de 77%) segundo os dados deste SLM.

2.3.2 Linguagens de especificação identificadas

A SLM incluída nesta dissertação resultou na identificação das linguagens de especificação apresentadas na Tabela 17. A primeira coluna apresenta os modelos gráficos, enquanto a segunda coluna apresenta as linguagens de especificação correspondentes e suas respectivas referências.

Modelo	Linguagens de especificação
i*	*Structured Modal Action Logic (MAL) (integration)* [P050]
Tropos	*Formal Tropos* [P055] [P193]*, Tropos4AS Language* [P056], *Tropos XML Specification Language (Extension)* [P155], *Tropos integration to OASIS formal Language* [P157]
KAOS	*KAOS Formal Language* [P016] [P096] [P099] [P103] [P157]*, SCR (Software Cost Reduction) Language (integration)* [P099]*, KAOS Language + KBRE (Knowledge Based RE) Language* [P104]*, RELAX Language (integration)* [P108]
NFR	*Σ language (integration)* [P051]
URN	*URN integrated to OCL constraints language* [P006], *URN integrated to ASPIC+ formal argumentation framework* [P045], URN + TUCM (Timed Use Case Maps) formal Semantics [P168], *URN (UCM Use Case Maps) formal semantics with ASM (Abstract State Machines)* [P184]
SysML	*SysML + ATLAS Transformation Language (integration)* [P142]*, SysML + TransML (Model Transformation Language)* [P145]
UML UC	*RPL (Requirements Pattern Language)* [P186]*, GBRAM (Goal Based Requirement Analysis Method) + CREWS L'ecritoire* [P013]
PF	*Causal Logic (integration)* [P122]*, ATLAS Transformation Language (integration)* [P142]
V-Graph	*PL-AOVgraph Language* [P121]
Techne	*Techne Requirements Modeling Language* [P131]
IRD	*MAUDE Formal Language* [P094]
RCAT	*LTL Formula* [P126]
RDG	*SpecTRM-RL* [P138]
Outros	NA

Tabela 17 – Linguagens de especificação identificadas
Fonte: o autor

Pode-se distinguir em relação às linguagens identificadas que há:

- Linguagens formais especificamente desenvolvidas para representar e validar requisitos (e.g.: *KAOS formal language*, *Formal Tropos*);
- Linguagens que formalizam requisitos como norma internacional (e.g.: *SysML*, *UML*, *URN*);

- Linguagens formais que foram integradas a um modelo de requisitos visando realizar validação formal do modelo, porém não são parte da definição do modelo (e.g.: *Structured Modal Action Logic* (MAL)*, Causal Logic, LTL Formula,* Σ *language, MAUDE Formal Language*).

2.3.3 Ferramentas de modelagem identificadas

A SLM incluída nesta dissertação resultou na identificação das ferramentas de modelagem (ferramentas de software) apresentadas na Tabela 18. A primeira coluna apresenta os modelos, enquanto a segunda coluna apresenta as ferramentas correspondentes e suas respectivas referências.

Modelo	Ferramentas de modelagem
i*	*Descartes Architect CASE-Tool* [P080]*, Goal to Architecture tool (GATO)* [P029]*, GOOD* [P086]*, GrowingLeaf* [P067]*, HUCRE* [P068]*, iPref-R framework* [P033], *iStarML Tool* [P072]*, i*ToNuSMV* [P069]*, JGOOSE* [P073] [P079]*, NòmosBPMN* [P089]*,* OME [P088], *OpenOME* [P048] [P081]*, REDEPEND* [P047]*, RE-Tools - StarUML plugin* [P107]*, Sira Framework* [P052]
Tropos	*GR-Tool* [P059]*, RE-context CASE-Tool* [P023], *Secure Tropos UML-Sec* [P060]*, Si* [P135], ST-Tool* [P058] [P153]*, S&D Tropos* [P136]*; TAOM4e* [P054] [P056] [P059], *T-Tool* [P055] [P193]
KAOS	*FAUST Toolset* [P100]*, GRAIL* [P016] [P098], *k-tool* [P101]*, Objectiver* [P031] [P034] [P088], *OMEGA2 profile UML/SysML Profile + Ifx Toolset* [P108], *QARAT* [P096]*, REInDetector tool* [P104]*, RE-Tools - StarUML plugin* [P107]
NFR	LEL [P116], *NFR Assistant* [P017]*, NDR-Tool - StarUML plugin* [P109], *rΣ Tool* [P053]*, RE-Tools - StarUML plugin* [P107]
URN	*AoURN-based SPL framework (ASF)* [P006]*, jUCMNav* [P002] [P168] [P174] [P176] [P177] [P178] [P179] [P180] [P182]*, jUCMNav tool adapted for Aspects* [P007]*, jUCMNav extended to LEGAL-URN* [P044]*, jUCMNav integrated to ASPIC+ framework* [P045]*, JUCMNav integrated to TouchCore Tool* [P171]*, UCMNav* [P170] [P172] [P175]*, UCMNav integrated to LQNGenerator tool* [P172]*, WebGRE (WebURN and WebUCM)* [P169]
SysML	*Papyrus Modeling Environment* [P142]*, Enterprise Architect (EA)* [C060]*, Magic Draw* [P149], *Visual Paradigm software* [P144]
UML UC	*AGG Tool* [P188]*, StarUML* [C061]*, Enterprise Architect (EA)* [C060]

Modelo	Ferramentas de modelagem
UCM	*UCMNav* [P163] [P183]*, jUCMNav* [P164] [P185]
PF	*Papyrus Modeling Environment* [P142]*, RE-Tools - StarUML plugin* [P107]
AMORE	*VRAT Tool* [P004]*, AMORE Modeling Tool* [P005]
UML RD	*Enterprise Architect (EA)* [P165] [C060]
URML	*UNICASE* [C017], *URML Add-In to Enterprise Architect* [P167] [C060]
ATHENA	*Metis Visual Modeling Tool* [P008]
Archimate	*Archimate 2.0 Tool* [P128]
RecVisu	*ReCVisu Tool* [P040] [P041] [P046]
V-Graph	*ReqSys-MDD tool* [P121]
RCAT	*RCAT Tool* [P126]
RIG	*Graphviz Tool (DOT format)* [P129]
OPM	*OPCat* [P137]
Techne	*Analytic Graph* [P151]
Outros	NA

Tabela 18 – Ferramentas de modelagem identificadas
Fonte: o autor

Dentre as ferramentas listadas acima, pode-se destacar a ferramenta *RE-Tools* [P107]. Esta ferramenta (gratuita) é um *plugin* do aplicativo *StarUML* [C061] (gratuito), que permite modelar diferentes notações visuais de RE:

a) NFR para modelagem de requisitos não-funcionais;
b) I* para modelagem orientada a agentes;
c) KAOS para modelagem formal orientada a objetivos;
d) Problem Frames para modelagem de problemas;
e) BPMN para modelagem de processos de negócio;
f) UML para modelagem orientada a objeto;
g) SysML nativamente na própria ferramenta *StarUML*.

2.3.4 Metodologias, processos e abordagens de RE identificados

A SLM incluída nesta dissertação resultou na identificação das ferramentas de metodologias, processos e abordagens mostrados na Tabela 19. A primeira coluna apresenta os modelos, enquanto a segunda coluna apresenta as metodologias, processos e abordagens relacionados.

Modelo	Metodologias, processos e abordagens de RE
i*	*AIRDoc-i* Process* [P075], *Analytic Hierarchy Process (AHP)* [P026], *AWARE (Analysis of Web Applications Requirements Engineering)* [P066], *DHARMA (Discovering Hybrid ARchitectures by Modelling Actors) Method* [P063], *Eri*c methodology* [P071], *GORE (Goal Oriented Requirements Engineering)* [P021] [P022] [P024] [P032] [P035], *HUCRE (Human-Centred Requirements Engineering) Method* [P068], *RESCUE Process* [P047]
Tropos	*GORE (Goal Oriented Requirement Models)* [P021] [P023] [P152], *Tropos Methodology* [P054] [P055] [P154] [P156] [P157] [P158] [P159] [P193], *Tropos4AS (Tropos for Adaptative Systems)* [P056], *Secure Tropos Methodology* [P057] [P058] [P153], *Secure Tropos + UMLSec* [P060], *GAM - Goal Argumentation Model* [P152], *GAIA Methodology* [P159], *MESSAGE Methodology* [P159], *Secure i* (tropos)* [P135] [P136]
KAOS	*KBRE (Knowledge Based Requirements Engineering) based on KAOS* [P104], *KAOS (Knowledge Acquisition in automated specification)* [P016] [P031] [P034] [P095] [P096] [P098] [P100] [P101] [P102], *KAOS + Scenarios* [P014], *GORE (Goal Oriented Requirements Engineering)* [P031] [P032] [P034] [P035] [P095], *KAOS derived to SCR (Software Cost Reduction) language* [P099], *RELAX Method* [P108], *UNC-Method* [P107]
NFR	*GORE (Goal Oriented Requirements Engineering)* [P022] [P035 [P110], *GOals to Statecharts (GO2S)* [P037], *BPMNFR approach* [P111]
URN	*CDD (Concern Driven Development)* [P007] [P178], *GORE (Goal Oriented Requirements Engineering)* [P169 [P170] [P182], *URN extended to LEGAL-GRL* [P044]

Modelo	Metodologias, processos e abordagens de RE
SysML	*MDEReq (Model Driven Engineering for Requirement Management)* [P139], *MBSE (Model Based Systems Engineering)* [P147 [P150], *MBRE (Model Based Requirements Engineering)* [P149], *CORAMOD (checklist-oriented requirements analysis modelling)* [P149], *ACRE (Approach to Context-Based Requirements Engineering)* [P150], *SysML Profile Extension for Safety Requirements* [P146], *SysML Profile Extension for Prioritizing Requirements* [P148]
UML UC	*GBRAM (Goal Based Requirement Analysis Method integrated to UC Models)* [P013]
V-Graph	*Goal Oriented Models integrated to Aspect Oriented Programming (AOP)* [P105] [P106] [P121] [P195]
RecVisu	*Visual Requirements Analytics* [P040] [P041] [P046]
AMORE	*AMORE (advanced multimedia organizer for requirements elicitation)* [P004] [P005]
UML RD	*UML Profile Extension for Requirements with COCA-MDA (Context Oriented Component Based Application for Model Driven Architecture)* [P165]
VLML	*VLRM (Very Lightweight Requirements Modeling)* [P127]
OPM	*OPM applied to requirements modeling* [P137]
AGORA	*AGORA (Attributed Goal-Oriented Requirements Analysis Method)* [P003]
ATHENA	*ATHENA Dynamic Requirements Definition System (DRDS)* [P008]
VRDL	*Structured Visual SRS using VRDL* [P192]
RON	*NOP Based Requirements Modeling* [P196]
Outros	NA

Tabela 19 – Metodologias, Processos e Abordagens de ER Identificadas
Fonte: o autor

2.3.5 Normas internacionais identificadas

A SLM incluída nesta dissertação resultou na identificação das normas internacionais identificadas na Tabela 20. A primeira coluna apresenta os modelos, a segunda coluna apresenta a norma internacional correspondente emitida pela entidade padronizadora relacionada na terceira coluna da tabela.

Modelo	Norma internacional	Entidade padronizadora
URN	Z.151 [C032]	ITU-T
SysML	SysML [C043]	OMG
UML UC	UML [C044]	OMG
URML	UML [C044]	OMG
UML RD	UML [C044]	OMG
OPM	ISO-19450 [C053]	ISO
Outros	NA	NA

Tabela 20 – Normas Internacionais Identificadas
Fonte: o autor

2.3.6 Metamodelos e ontologias identificadas

A SLM incluída nesta dissertação resultou na identificação dos metamodelos e ontologias identificadas na Tabela 21. A primeira coluna apresenta os modelos, enquanto a segunda coluna apresenta os metamodelos e/ou ontologias associadas.

Modelo	Metamodelo ou ontologia
i*	*i* Metamodel* [C037] [P088], *GORO (Gore Oriented Requirements Ontology)* [P036], *User story structuring Metamodel* [P080]
Tropos	*Tropos Metamodel* [P059] [P190], *GORO (Gore Oriented Requirements Ontology)* [P036]
KAOS	*KAOS Metamodel* [P088], *KBRE Metamodel* [P104], *SysML/KAOS Metamodel* [P108], *GORO (Gore Oriented Requirements Ontology)* [P036], *UML profile to support RE with KAOS* [C063]
NFR	*NFR Metamodel with Scenarios* [P160], *GORO (Gore Oriented Requirements Ontology)* [P036], *Quality requirements modeling Metamodel* [P051], *Metamodel for SIG* [P096]
URN	*Activity Theory Metamodel based on OCL (Object Contraints Language)* [P002], *Timed UCM core metamodel* [P168], *UCM availability metamodel* [P176], *GRL Abstract Metamodel* [P182]
SysML	*SysML Metamodel* [P108], *ACRE Ontology* [P150]

Modelo	Metamodelo ou ontologia
UML UC	*UML Metamodel* [C044]
UCM	*Abstract UCM Metamodel* [C032]
Techne	*Analytic Graph Language Management Module Metamodel* [P131] [P151]
URML	*URML Metamodel* [P166]
AGORA	*AGORA Goal Graph Metamodel* [P003]
Outros	NA

Tabela 21 – Metamodelos e Ontologias Identificadas
Fonte: o autor

2.3.7 Evolução temporal das abordagens de modelagem identificadas

Nesta seção é apresentada uma evolução temporal dos modelos de representação gráfica em RE identificados neste estudo, construída com base nos artigos dos autores proponentes de cada abordagem. A **linha do tempo** apresentada permite verificar a evolução dos modelos, ao representar os processos de fusão e de derivação de sub-abordagens (ver Figura 20):

- A abordagem NFR influenciou a elaboração do *framework* i*, que por sua vez derivou as abordagens Tropos e Vgraph;
- As abordagens i* e UCM foram integradas para formar a URN em torno do ano 2000. Em 2008, a URN foi padronizada pela ITU-T como Z.151 (versão atual: 10/12 de 2012);
- A UML passou a ser elaborada a partir de 1990, foi padronizada pela OMG em 1997 e está na versão 2.5 de 2015. A partir da UML foram derivados os modelos UCM, SysML, URML e UML RD;
- A SysML foi derivada a partir da UML em esforço iniciado a partir de 2001 e padronizada pela OMG em 2007 (versão atual: 1.5 de 2017);
- Diversas abordagens com somente uma publicação (pontuais) foram propostas no período considerado, tais como: AGORA, ATHENA, Archimate, RDG, RIG, RCAT, OPM, VRDL, UML RD, VLML e RON. Estes trabalhos propuseram modelos alternativos e ideias originais para modelar requisitos em RE.

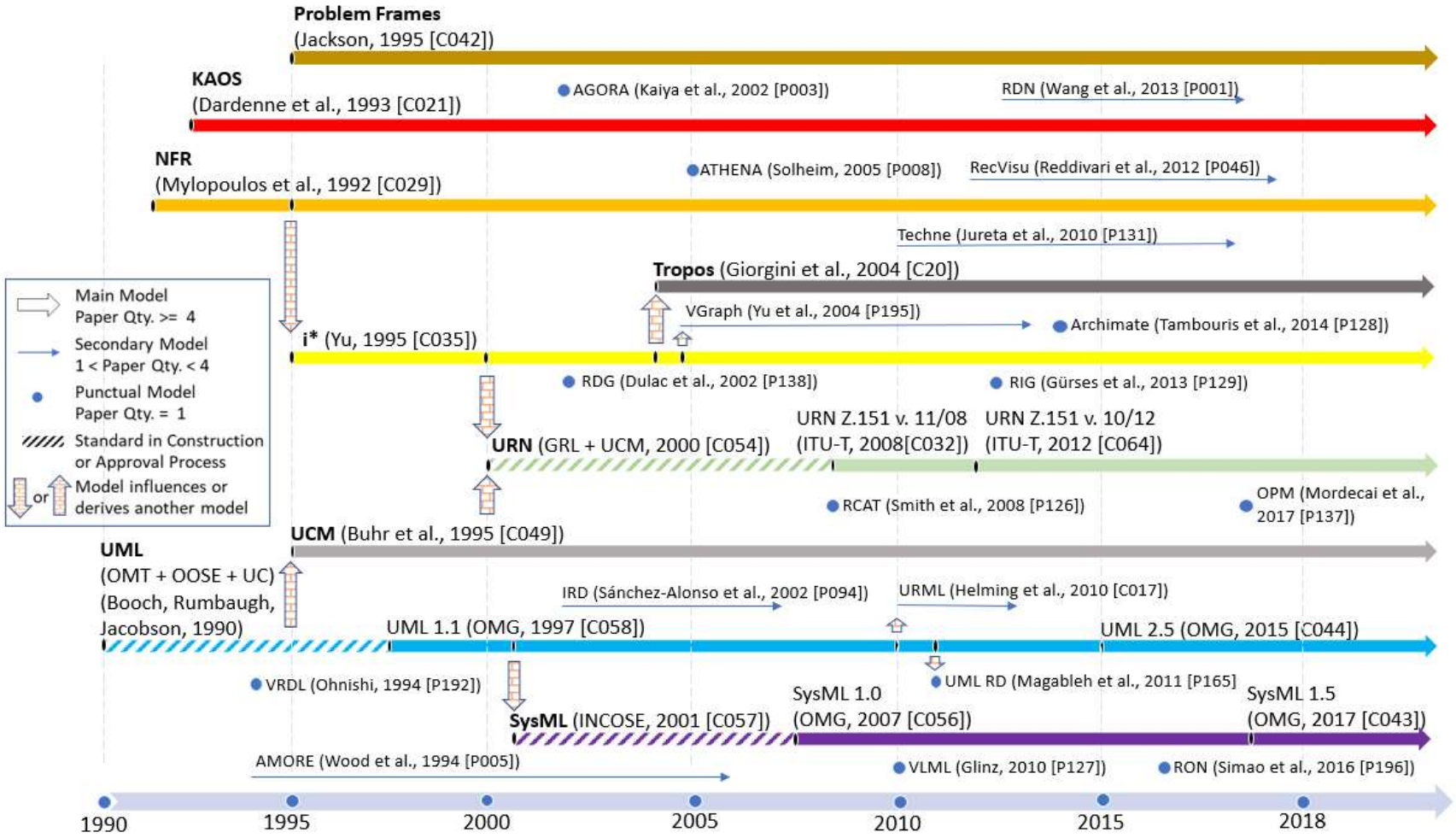


Figura 20 – Evolução Temporal de Modelos Gráficos de Requisitos
Fonte: o autor

2.3.8 Comparação de modelos gráficos em RE em termos de características

A Tabela 22 apresenta uma comparação entre modelos gráficos em RE, elaborada com base em informações obtidas a partir do estudo das abordagens apresentadas nesta seção 2.3. As características de comparação escolhidas são (identificadas de C1 a C18):

- Contexto (LOUCOPOULOS et al., 2013): conforme explicado na seção 2.1.4, os modelos gráficos podem se enquadrar nos contextos de *Early Requirements* (C1), *Late Requirements* (C2) ou ambos;
- Modelagem (intra-diagrama): são capacidades intrínsecas que representam o que a notação permite fazer: modelar problemas (C3), modelar objetivos (C4), modelar cenários (C5), modelar FRs (C6), modelar NFRs (C7), modelar conflitos (C8), modelar refinamentos (C9), modelar restrições de forma geral (C10), modelar restrições como interdependências (C11), possuir mecanismo de escalabilidade (C12), possuir mecanismo de rastreabilidade (C13). Enquanto C3 a C7 se referem às entidades de modelagem comuns em modelos gráficos (seção 2.3), as características C8, C9, C10 e C13 se relacionam a tipos de interdependências (ver seção 2.3.8). A característica C12 está ligada ao princípio de gerenciamento de complexidade de PoN (seção 2.2.3), que indica se a notação fornece um mecanismo para gerenciar a escalabilidade de diagramas. A característica C11 se refere à possibilidade de representar restrições lógico-matemáticas diretamente nas interdependências de um diagrama de requisitos;
- Características Gerais: são propriedades gerais dos modelos gráficos (ver seção 2.3): ter linguagem formal (C14); ter ferramentas (C15); ter norma internacional (C16); ter metamodelo (C17). A característica “seguir princípios PoN” (C18) indica que uma notação visual foi desenvolvida conforme os princípios de PoN (ver seção 2.2.4).

		C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18
		Contexto		Características de modelagem (intra-diagrama)											Características gerais do sistema de modelagem				
N	Nome (sigla)	Modelar *Early Req.*	Modelar *Late Req.*	Modelar problemas	Modelar objetivos	Modelar cenários	Modelar FRs	Modelar NFRs	Modelar conflitos	Modelar refinamentos	Modelar restrições (geral)	Modelar restrições como dependências	Possuir mecanismo de escalabilidade	Possuir mecanismo de rastreabilidade	Ter linguagem Formal	Ter ferramenta	Ter norma (padrão)	Ter metamodelo	Seguir princípios PoN
01	i*	X			X				X	X	X				X	X		X	
02	Tropos	X	X		X				X	X	X			X	X	X		X	
03	KAOS	X	X		X		X	X	X	X	X			X	X	X		X	
04	NFR	X	X		X			X	X	X	X				X	X		X	
05	URN	X	X		X	X			X	X	X		X	X	X	X	X	X	
06	SysML		X				X	X		X	X			X	X	X	X	X	
07	UML UC		X			X				X					X	X	X	X	
08	UCM		X			X			X		X		X			X	X	X	
09	PF	X		X					X	X	X				X	X			
10	V-Graph	X			X				X	X					X	X			
11	RecVisu		X				X									X			
12	Techne		X		X				X		X				X	X		X	
13	AMORE	X				X										X			
14	URML		X		X		X	X	X		X					X		X	X
15	UML RD		X				X	X								X			
16	VLML	X	X		X		X	X											
17	OPM		X				X	X								X			
18	RDN		X				X				X								
19	AGORA	X			X				X	X								X	
20	ATHENA		X				X									X			
21	Archimate	X	X		X		X				X					X			
22	IRD		X				X				X				X				
23	RCAT		X				X		X		X				X	X			
24	RIG		X				X		X	X									
25	RDG		X				X						X			X			
26	VRDL	X				X													
27	RON		X				X	X			X	X							

Tabela 22 – Comparação de modelos gráficos em termos de características
Fonte: o autor

Ao analisar a Tabela 22, é possível concluir que:

a) Há várias abordagens de modelagem de requisitos orientadas a objetivos (*goal-oriented*) tais como (C4): i*, Tropos, KAOS, NFR, URN, V-Graph, Techne, URML, AGORA, Archimate;
b) A modelagem gráfica por meio de cenários é característica de algumas abordagens (C4): URN, UML UC, UCM, AMORE, VRDL;
c) Somente a abordagem de *Problem-Frames* (PF) realiza a modelagem de problemas (C3), segundo os dados desta SLM;
d) A modelagem de restrições é algo possível em várias das abordagens (C10), porém somente a abordagem RON permite a modelagem de restrições como interdependências (C11);
e) Somente três abordagens possuem um mecanismo claro para gerenciar a complexidade visual em diagramas (C3): URN, UCM, RDG;
f) Um mecanismo para rastreabilidade de requisitos é algo que foi implementado em (C13): Tropos, KAOS, URN, SysML;
g) As modelagens que possui norma internacional associada são (C16): URN, SysML, UML UC, UCM;
h) A definição de um metamodelo é algo presente em menos da metade das abordagens identificadas (C17): i*, Tropos, KAOS, NFR, URN, SysML, UML UC, UCM, Techne, URML, AGORA;
i) Somente uma notação visual foi desenvolvida segundo os princípios de física das notações (C18): URML.

2.4 TIPOS DE INTERDEPENDÊNCIAS ENTRE REQUISITOS

Interdependências entre requisitos podem ser entendidas como conexões semânticas (com significado) entre requisitos. As interdependências podem tanto ser parte da semântica de modelos gráficos de requisitos ao representar ligações (caminhos gráficos) entre os elementos do modelo, quanto podem ser parte da semântica associada a métodos de gerenciamento de requisitos textuais (e.g. ligações referenciais textuais em tabelas de requisitos).

As dependências entre requisitos individuais têm uma importante influência nas atividades de SWE. Apesar de dezenas de tipos de dependências entre requisitos terem sido sugeridas na literatura a partir de diferentes pontos

de vista, ainda há falta de estudos de avaliação da aplicabilidade dos tipos de dependências em RE (ZHANG et al., 2014).

Pesquisadores de temas relacionados a gerenciamento e rastreabilidade de requisitos procuraram estudar a semântica das interdependências e os tipos de interdependências entre requisitos (POHL, 1996) (DAHLSTEDT, 2001) (DAHLSTEDT e PERSSON, 2005) (SPIJKERMAN, 2010) (GOKNIL et al., 2011) (LI et al., 2012) (ZHANG et al., 2014). Ao avaliar estes estudos, nota-se que há pouca ou nenhuma conexão entre estes estudos e o desenvolvimento e propostas de modelos gráficos de requisitos, tais como os apresentados e citados na seção 2.3.

Nas próximas seções, serão apresentados estudos que se preocuparam em modelar genericamente os tipos de interdependências entre requisitos.

2.4.1 Modelo de interdependências P-Model

Em 1996 Pohl propôs uma classificação de interdependências de requisitos, posteriormente apelidada de “P-Model”, baseada em 30 estudos (artigos) de RE da época (POHL, 1996).

No primeiro nível da classificação (Figura 21), os tipos de interdependências poderiam ser: restrições (*constraints*), conteúdo (*content)*, documentos (*documents*), evolução (*evolutionary*) e abstração (*abstraction*).

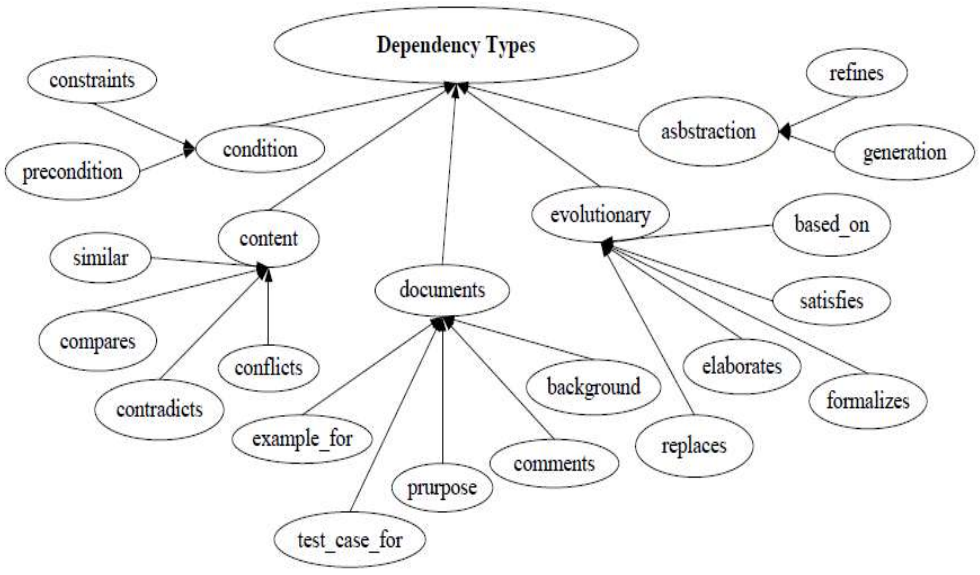


Figura 21 – Modelo de dependências de Pohl (P-Model).
Fonte: (POHL, 1996)

No segundo nível da classificação havia 18 tipos de interdependências (Figura 21), que poderiam representar relacionamentos entre requisitos e outros objetos de informação (não somente requisitos), tais como entidades pertencentes ao projeto de *software* (*e.g.* documentos) (DAHLSTEDT, 2001).

2.4.2 Modelo de interdependências D-Model

Em 2005, Dahlstedt e Persson propuseram um novo modelo genérico de tipos de interdependências de requisitos (D-Model), em que somente relacionamentos entre requisitos foram incluídos (Figura 22) (DAHLSTEDT e PERSSON, 2005). As interrogações "?" mostradas na figura indicam que o modelo necessita de evolução e que outros tipos de interdependências poderiam ser incluídos futuramente.

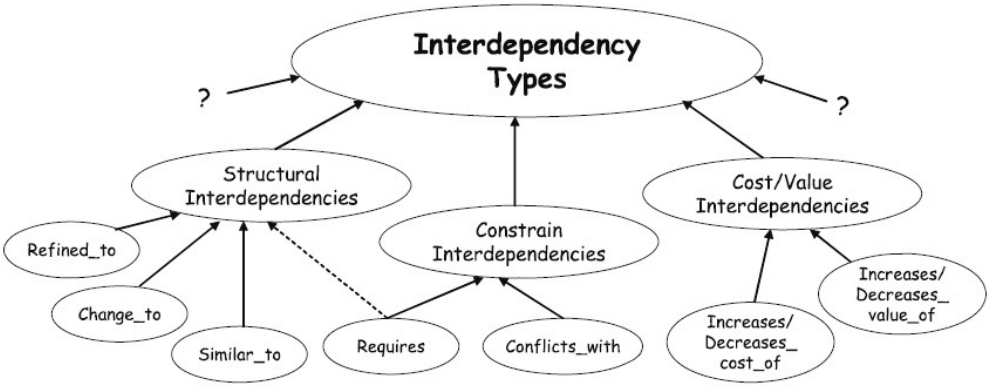


Figura 22 – Classificação dos tipos fundamentais de interdependências (D-Model)
Fonte: (DAHLSTEDT e PERSSON, 2005)

O D-Model é uma síntese de abordagens anteriores sobre rastreabilidade e gerenciamento de requisitos (KARLSSON et al., 1997) (CARLSHAMRE et al., 2001) (RAMESH e JARKE, 2001) (ROBINSON et al., 2003) e classificação de tipos de interdependências entre requisitos (POHL, 1996) (DAHLSTEDT, 2001).

Este modelo contém 7 tipos fundamentais de interdependências entre requisitos, classificados em 3 categorias (DAHLSTEDT e PERSSON, 2005):

a) **Interdependências estruturais** (*structural interdependencies)*: preocupam-se com o fato de que, dado um conjunto específico de requisitos, eles podem ser organizados em uma estrutura na qual os relacionamentos são de natureza hierárquica, bem como de estrutura

cruzada (transversais). Os requisitos de negócios de alto nível são gradualmente decompostos em requisitos de *software* mais detalhados, formando uma hierarquia. Além disso, pode haver relações estruturais (transversais) entre requisitos em diferentes partes da hierarquia geral (DAHLSTEDT e PERRSON, 2005). Nesta categoria, foram incluídos os seguintes tipos de interdependências:

- **Refined_to** (refinado_para): um requisito de alto nível é refinado para um ou mais requisitos específicos. Este tipo de dependência é usado para descrever estruturas hierárquicas, em que requisitos mais detalhados podem estar relacionados a seus requisitos de origem. O requisito de origem (nível superior) pode ser visto como uma abstração do requisito detalhado (nível inferior).
- **Change_to** (modificado_para): um requisito é modificado se uma nova versão do requisito é desenvolvida e substitui o requisito antigo. Este tipo é utilizado para descrever a história de um requisito, ou seja, como o mesmo requisito evolui durante o tempo. Um requisito poderia ser modificado com o tempo por diversas razões, tais como: para se tornar mais compreensível, para modificar detalhes do requisito ou para se tornar mais formal.
- **Similar_to** (similar_a): indica que um requisito é similar a outros requisitos. Este tipo descreve situações em que um mesmo requisito é expresso de formas diferentes, porém contém a mesma ideia básica em termos de conteúdo. Também pode ser entendido como uma forma de identificar requisitos duplicados.

b) **Interdependências restritivas** (*constraining interdependencies)*: estão relacionadas a requisitos que são dependentes ou restringem outros requisitos (DAHLSTEDT e PERSSON, 2005). Dahlstedt e Persson colocam como hipótese a existência de outros tipos de interdependências restritivas não incluídas no modelo por eles proposto. Eles optam por incluir somente tipos genéricos de interdependências:

- **Requires** (requer): a realização de um requisito depende da realização de outro requisito. Este tipo é usado para descrever que, caso um requisito seja incluído na especificação de requisitos, outro requisito também deverá ser incluído (obrigatoriedade). Este tipo de

dependência também pode ser entendido como uma dependência estrutural.

- **Conflicts_with** (conflita_com): um requisito conflita com outro requisito se ambos não podem existir simultaneamente ou se o incremento da satisfação de um requisito provoca o decremento da satisfação do outro.

c) **Interdependências de custo/valor** (*cost/value interdependencies)*: estão relacionadas ao custo envolvido na implementação de um requisito em relação ao valor percebido pelo cliente (DAHLSTEDT e PERSSON, 2005). Podem ser dos seguintes tipos:

- **Increases_cost_of / Decreases_cost_of** (incrementa_custo_de/ decrementa_custo_de): se um requisito é escolhido para implementação, então o custo de implementar outro requisito aumenta ou diminui.
- **Increases_value_of / Decreases_value_of** (incrementa_valor_de / decrementa_valor_de): se um requisito é escolhido para implementação, então o valor de implementar outro requisito aumenta ou diminui.

2.4.3 Modelo de interdependências S-Model

Em 2010, como parte de uma dissertação de mestrado sobre análise de impacto de mudanças em requisitos (*Change Impact Analysis* – CIA), Wietze Spijkerman definiu um metamodelo (Figura 23) associado a um modelo formal de requisitos, inclusive definindo os tipos de relacionamentos entre requisitos. Essa dissertação irá se referir a este modelo como *S-Model*.

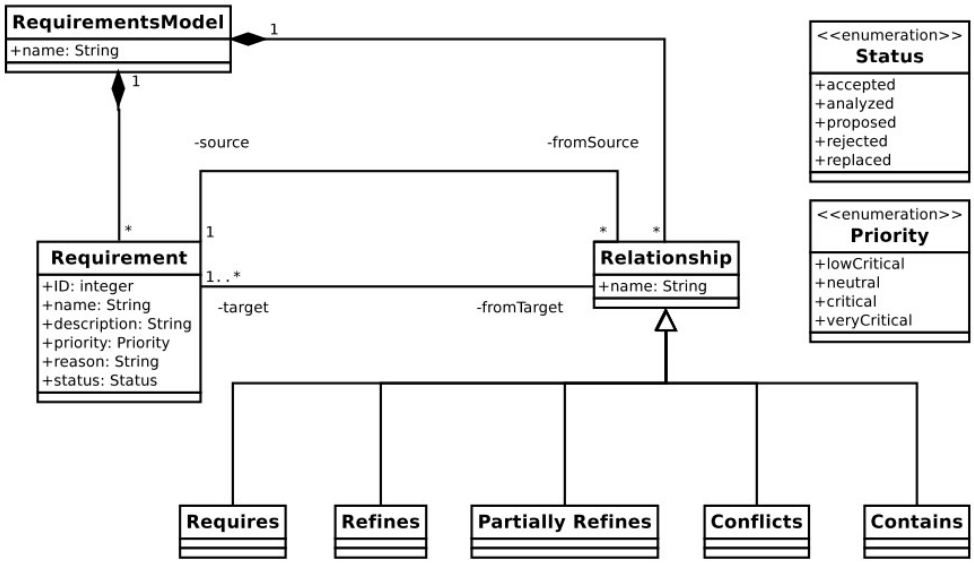


Figura 23 – Metamodelo de requisitos com relações formais (S-Model)
Fonte: (SPIJKERMAN, 2010)

A descrição dos relacionamentos (*relationships*) propostos no modelo/metamodelo de Spijkerman é dada a seguir (SPIJKERMAN, 2010):

a) **Contém** (*contains*): esse relacionamento permite que um requisito complexo seja decomposto em partes.
b) **Refina** (*refines*): um requisito R2 refina um requisito R1 se R2 é derivado de R1 quando novos detalhes são adicionados a R1. O requisito refinado R1 pode ser visto como uma abstração do requisito detalhado R2.
c) **Refina parcialmente** (*refines partially*): um requisito R2 refina parcialmente um requisito R1 se R2 é refinado a partir de R1 a partir da adição de mais detalhes a R1 e exclusão de partes não-refinadas de R1.
d) **Requer** (*requires*): um requisito R2 requer um requisito R1 se R2 é realizado somente quando R1 é realizado. O requisito R1 pode ser visto como uma pré-condição para o requisito R2.
e) **Conflita** (*conflicts*): um requisito R1 conflita com um requisito R2 se a realização de R1 exclui a realização de R2 e vice-versa.

2.4.4 Classificação de dependências Z-Model

Em 2012 (LI et al., 2012) e em 2014 (ZHANG et al., 2014), um grupo de pesquisadores conduziu um estudo empírico tomando como base os modelos P-

Model (POHL, 1996) e D-Model (DAHLSTEDT e PERSSON, 2005). Essa dissertação irá se referir a este modelo como *Z-Model*.

Naquele estudo, 3 participantes identificaram se os tipos de interdependências propostos nos modelos P-Model e D-Model existiam em documentos de especificação de requisitos de projetos reais (ZHANG et al., 2014). A partir dos resultados obtidos, foi proposta uma nova classificação de dependências de requisitos (Figura 24).

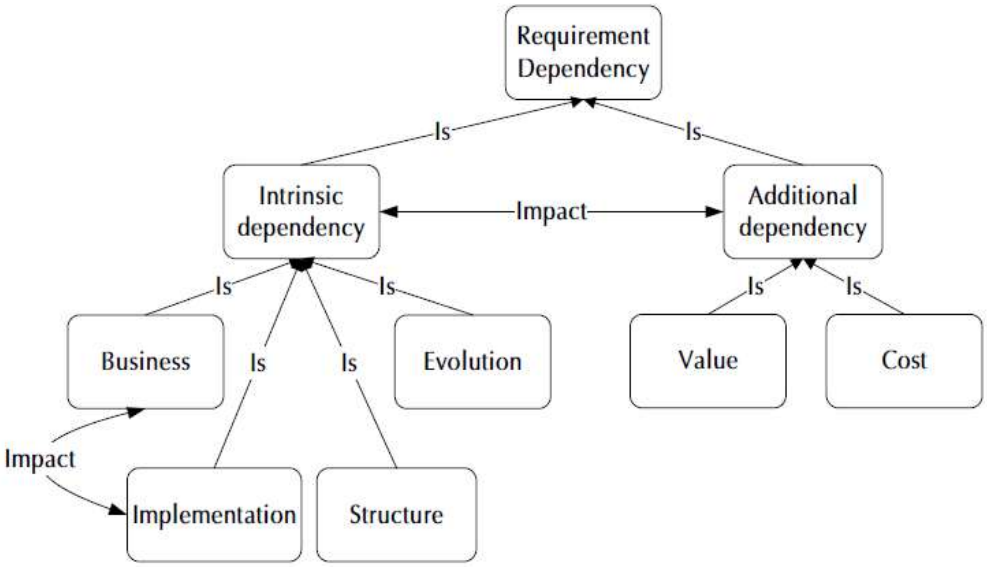


Figura 24 – Classificação de dependências de Requisitos (Z-Model)
Fonte: (ZHANG et al., 2014)

O primeiro nível da classificação proposta (Figura 24) divide os tipos de dependências entre **intrínsecas** (*intrinsic*) e **adicionais** (*additional*). Dependências intrínsecas retratam relacionamentos essenciais e refletem informações semânticas e estruturais entre requisitos. Dependências adicionais refletem finalidades específicas de SWE, tais como dependências de valoração e custo entre requisitos durante a implementação (ZHANG et al., 2014).

O segundo nível da classificação apresenta seis categorias: negócio (*business*), implementação (*implementation*), estrutura (*structure*), evolução (*evolution*), valor (*value*) e custo (*cost)*. Estas categorias se baseiam na finalidade de cada tipo de interdependência (ZHANG et al., 2014).

O terceiro nível da classificação acrescenta os tipos de interdependências em si (Tabela 23) (ZHANG et al., 2014).

Tipo de dependência	Categoria	Definição
restringe (*Constrain*)	Negócio, Implementação	Um requisito é uma restrição de outro requisito. Esse tipo representa relações transversais entre requisitos. (*e.g:* NFRs)
precede (*precede*)	Negócio	Se o requisito A precede o requisito B, então A é uma pré-condição de B.
é_similar_a (*be_similar_to*)	Implementação	Dois requisitos que compartilham o mesmo conteúdo são similares.
refina (*refine*)	Estrutura	Um requisito é refinado por requisitos mais específicos (detalhamento).
é_exceção_de (*be_exception_of*)	Negócio, Implementação	Um requisito descreve um evento excepcional de outro requisito. Ex: mensagens de exceção.
conflita (*conflict*)	Implementação	Implementar um requisito impacta negativamente em outro requisito.
evolui_para (*evolve_into*)	Evolução	Se um requisito B é uma nova versão de um requisito A, então A evolui para B.
incrementa_custo_de decrementa_custo_de (*increase_cost_of / decrease_cost_of)*	Custo	A implementação de um requisito provoca o aumento do custo de implementação de outro requisito.
incrementa_valor_de decrementa_valor_de (*increase_value_of / decrease_value_of)*	Valor	A implementação de um requisito aumenta o valor de implementação de outro requisito. Valor equivale à satisfação percebida pelo cliente.

Tabela 23 – Tipos de Interdependências entre requisitos
Fonte: Adaptado e sintetizado a partir de (ZHANG et al., 2014)

2.5 REQUISITOS ORIENTADOS A NOTIFICAÇÕES – RON

Em 2016, pesquisadores da UTFPR e outros (SIMÃO et al., 2016) propuseram uma abordagem de modelagem gráfica de requisitos que se baseou em primitivas de modelagem originadas do Paradigma Orientado a Notificações (PON), que posteriormente foi denominada RON (*Requirements Oriented to Notifications*) (NOVAES et al., 2018).

Para que se possa compreender a abordagem RON esta seção abordará inicialmente o PON com os seguintes tópicos: breve introdução sobre o PON (subseção 2.5.1); panorama de pesquisas PON relacionadas a SWE, SE e RE (subseção 2.5.2); conceitos básicos do PON (subseção 2.5.3). Na sequência, os resultados dos proponentes originais da abordagem RON (SIMÃO et al., 2016), são apresentados conforme segue: notação visual para RE baseada em conceitos do PON (subseção 2.5.4); técnica de modelagem de requisitos ("*NOP Based Requirements Modeling*") (subseção 2.5.5); e exemplo da modelagem aplicada aos requisitos de um sistema de segurança "*Access Security System"* (INCOSE, 2010) (subseção 2.5.6).

2.5.1 Paradigma Orientado a Notificações - PON

Um **paradigma de programação** pode ser entendido como uma abordagem para programar um computador baseada em uma teoria matemática ou conjunto coerente de princípios (ROY, 2009). Ele também pode ser entendido como um padrão ou modelo computacional que caracteriza uma escola de pensamento de programação de computadores. Cada escola de pensamento desenvolve uma forma diferente de visualizar a programação e a isso se dá o nome de paradigma, como o orientado a objetos, lógico, imperativo declarativo e funcional (ROY e HARIDI, 2004).

De forma geral (simplificada), considera-se que há dois grandes paradigmas de programação: o Paradigma Imperativo (PI) e o Paradigma Declarativo (PD). Estes paradigmas se interseccionam, sendo por vezes o PD considerado uma abstração sobre o PI. O PI pode se dividir em dois outros paradigmas: o Paradigma Procedimental (PP) e o Paradigma Orientado a objetos (POO). O PD, por sua vez, pode ser subdividido em o Paradigma Funcional (PF) e o Paradigma Lógico (PL) (GABBRIELLI e MARTINI, 2010)

(PORDEUS, 2017). Van Roy estabelece uma taxonomia de paradigmas de forma mais precisa (mais ampla), baseada em conceitos oriundos de linguagens de programação (ROY, 2009). Posteriormente, esta taxonomia foi estendida com a inclusão do PON (XAVIER, 2014).

Os conceitos básicos do PON foram propostos inicialmente pelos pesquisadores Jean Marcelo Simão e Paulo Cézar Stadzisz (SIMÃO, 2001) (SIMÃO, 2005), na forma de uma solução de controle discreto para sistemas inteligentes de manufatura. Esta abordagem define uma forma de colaboração entre entidades de manufatura holônica que foi transformada em uma solução de inferência de *software* genérica (SIMÃO e STADZISZ, 2008), com suporte a mecanismos de resolução de conflitos de execução e determinismo (SIMÃO e STADZISZ, 2009) (SIMÃO e STADZISZ, 2010).

Posteriormente, o PON alcançou a forma de um paradigma de desenvolvimento de sistemas computacionais, através da criação de ferramentas e *frameworks* para o suporte ao desenvolvimento de *software* (BANASZEWSKI et al., 2007) (BANASZEWSKI, 2009) (SIMÃO et al., 2012) (RONSZCKA, 2012) (VALENÇA, 2012).

Nos últimos anos, o PON se constituiu em uma alternativa aos paradigmas tradicionais de programação, permitindo o desenvolvimento de aplicações em diferentes plataformas, dentre as quais destaca-se:

a) ***Software***: Linguagem de programação *LingPON* e Compilador (FERREIRA, 2015), Método para a criação de linguagem e compilador para o PON (RONSZCKA et al., 2017), *software* de controle de futebol de robôs (SANTOS, 2017), dentre outros;
b) ***Hardware***: Coprocessador VHDL (PETERS et al., 2012), Arquitetura de *hardware* ARQPON (LINHARES, 2015), simulador de *hardware ARQPON* (PORDEUS, 2017), implementação de *hardware* digital com PON (KERSCHBAUMER, 2018), (KERSCHBAUMER et al., 2018).

Na seção 2.5.3 são apresentados os conceitos básicos do PON, de forma a tornar mais clara a originalidade e diferencial deste paradigma em relação aos demais paradigmas de programação existentes.

2.5.2 Panorama de pesquisas do PON em SWE, SE e RE.

Durante o desenvolvimento das pesquisas relacionadas ao PON, houve uma expansão de seus conceitos para novos domínios de conhecimento (PORDEUS, 2017). Dentre estes domínios, é possível salientar aqueles relacionados ao contexto do presente trabalho (SWE, SE e RE), considerando-se o panorama apresentado na Figura 25, adaptado de (PORDEUS, 2017).

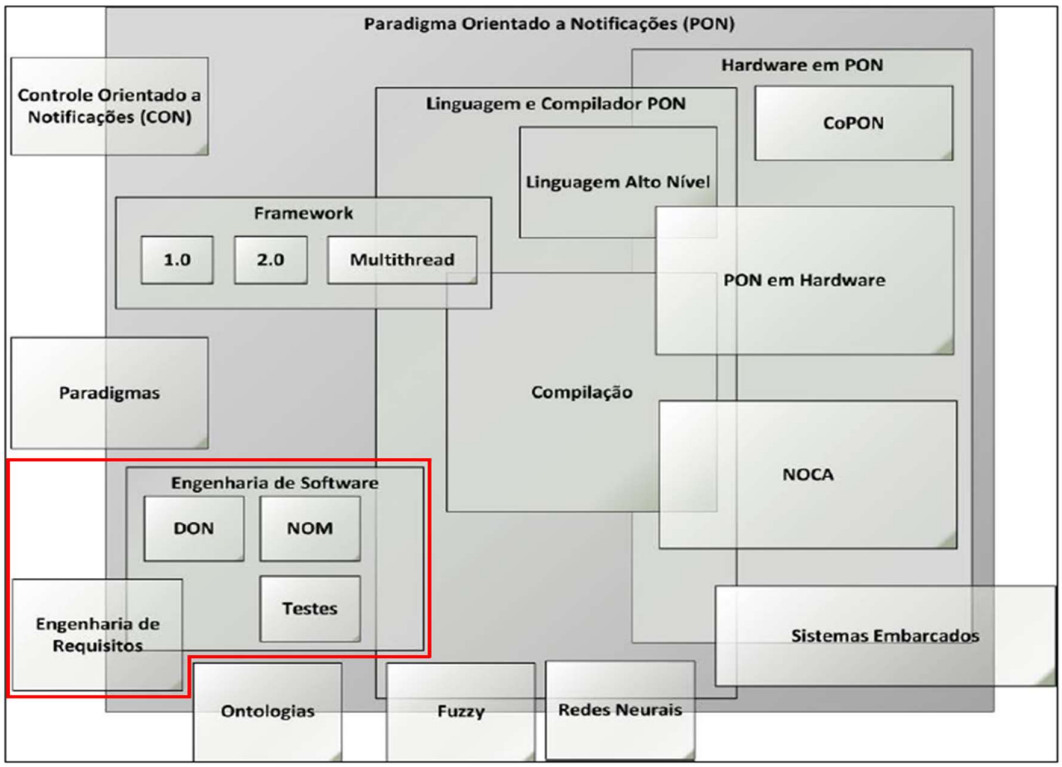


Figura 25 – Panorama das pesquisas relacionadas a SWE, SE e RE no PON
Fonte: Adaptado de (PORDEUS, 2017)

Na Figura 25 aponta-se em vermelho a existência de trabalhos relacionados a RE e SWE tais como DON, NOM e testes de *software*. Uma lista completa destes trabalhos é dada na Tabela 24.

Tema	Referências
DON - Método de Desenvolvimento de *Software* Orientado a Notificações	(WIECHETECK et al., 2011) (WIECHETECK, 2011) (MENDONÇA, 2015)
Aplicação da teoria de projeto axiomático (PA) ao desenvolvimento de *software* PON	(BATISTA, 2013)

Tema	Referências
Método de testes para desenvolvimento de *software* usando PON	(KOSSOSKI et al., 2014) (KOSSOSKI, 2015)
Abordagem orientada a notificações para requisitos de sistemas de engenharia Integração entre DON e RON em um estudo de caso de sistema de segurança	(SIMAO et al., 2016) (NOVAES et al., 2018)
NOM – Metodologia de Projeto de *Software* Orientado a Notificações	(MENDONÇA, 2018)

Tabela 24 – Listas das pesquisas relacionadas a SWE, SE e RE no PON
Fonte: o autor

Dentre os trabalhos listados na Tabela 24, os trabalhos de (SIMAO et al., 2016) e (NOVAES et al., 2018), cujo foco está em RE e SE, fazem parte da base conceitual e contextual da presente dissertação.

2.5.3 Conceitos Básicos do PON

O PON se propõe a resolver certos problemas existentes nos paradigmas tradicionais de programação PD e PI, normalmente relacionados a redundâncias estruturais e temporais (*i.e.*, redundâncias de códigos lógico-causais) e o acoplamento forte de entidades computacionais (FERREIRA, 2015).

Uma das inovações do PON consiste na introdução de um mecanismo de inferência baseada em notificações (SIMÃO e STADZISZ, 2008) associado a um mecanismo de resolução de conflitos para ambientes mono e multiprocessados (SIMÃO e STADZISZ, 2009). O PON apresenta uma nova maneira de estruturar *software* em entidades computacionais de pequeno porte (reativas ou ativas) e desacopladas que colaboram por meio de notificações pontuais e precisas, criadas a partir do conhecimento de regras lógico-causais (BANASZEWSKI, 2009) (SIMÃO et al., 2012). Uma das vantagens da programação neste paradigma é uma maior flexibilidade e facilidade na concepção de sistemas que apresentem paralelismo ou distribuição, por evitar implicitamente o acoplamento excessivo entre entidades computacionais (PORDEUS, 2017) (SIMÃO et al., 2012).

Na Figura 26, é possível observar o modelo de entidades computacionais do PON.

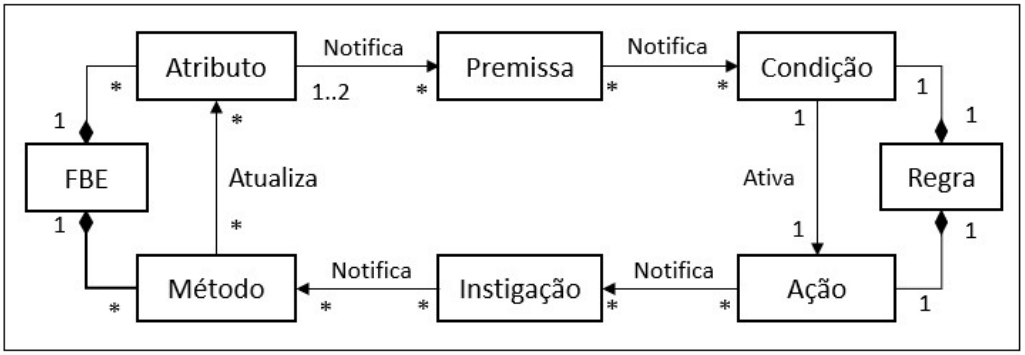


Figura 26 – Entidades do PON
Fonte: Adaptado e traduzido de (NOVAES et al., 2018), baseado em (Simão et al., 2016)

A explicação de cada entidade do modelo da Figura 26 é dada conforme segue (PORDEUS, 2017):

a) **FBEs**: são utilizadas para representar objetos do mundo (entidades físicas ou abstratas) em um sistema computacional. Essa representação ocorre por meio de conjuntos exclusivos de estados tratados por entidades chamadas de **atributos** *(attributes)* e de serviços tratados por entidades chamadas de **métodos** (*methods)*.

b) **Regras** (*rules*): definem, em conjunto com suas entidades, como o cálculo lógico-causal deve ser efetuado sobre os estados das FBEs (atributos) ao controlar a execução de seus serviços (métodos). Uma regra compõe-se da entidade **condição** (*condition*) que está associada a **premissas** (*premisses*) e de uma entidade **ação** (*action*) que está associada a **instigações** (*instigations*). Um exemplo de regra é apresentado na Figura 27.

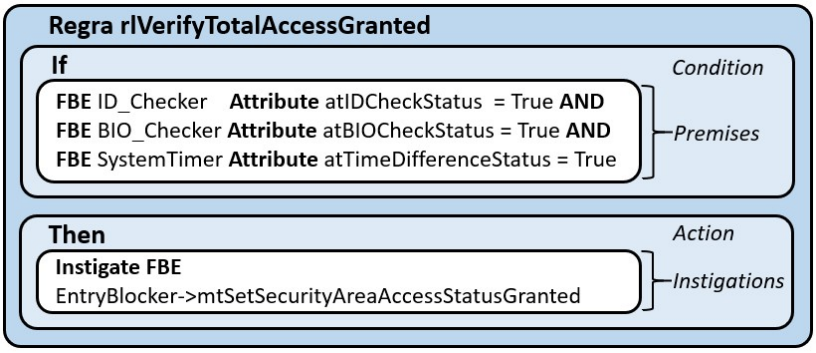


Figura 27 – Exemplo de composição de uma regra do PON
Fonte: (NOVAES et al., 2018)

O exemplo da Figura 27 trata de um sistema de controle de acesso de segurança a uma porta com verificação de ID (crachá) e BIO (Biometria). A **regra** *rlVerifyTotalAccessGranted* é composta de (NOVAES et al., 2018):

a) Uma **condição** composta de três FBEs: *ID_Checker, BIO_Checker e SystemTimer*. Cada um destes FBEs possui uma premissa associada através de um atributo correspondente e um valor de condição;
b) Uma **ação** composta de uma instigação associada ao FBE *EntryBlocker*, que poderia ativar o método *mtSetSecurityAreaAccessStatusGranted.*

Neste exemplo, uma modificação na **condição** nos estados dos atributos (**premissas**) para (*atIDCheckStatus* = True AND *atBIOCheckStatus* = *True* AND *atTimeDifferenceStatus* = *True*), ativa a **ação** que notifica a **instigação** associada ao método *mtSetSecurityAreaAccessStatusGranted*. Este método, por sua vez, modifica os atributos necessários para que ocorra a abertura da porta de segurança. A Figura 28 ilustra a descrição acima, ao mostrar o esquema de colaboração entre entidades do metamodelo do PON.

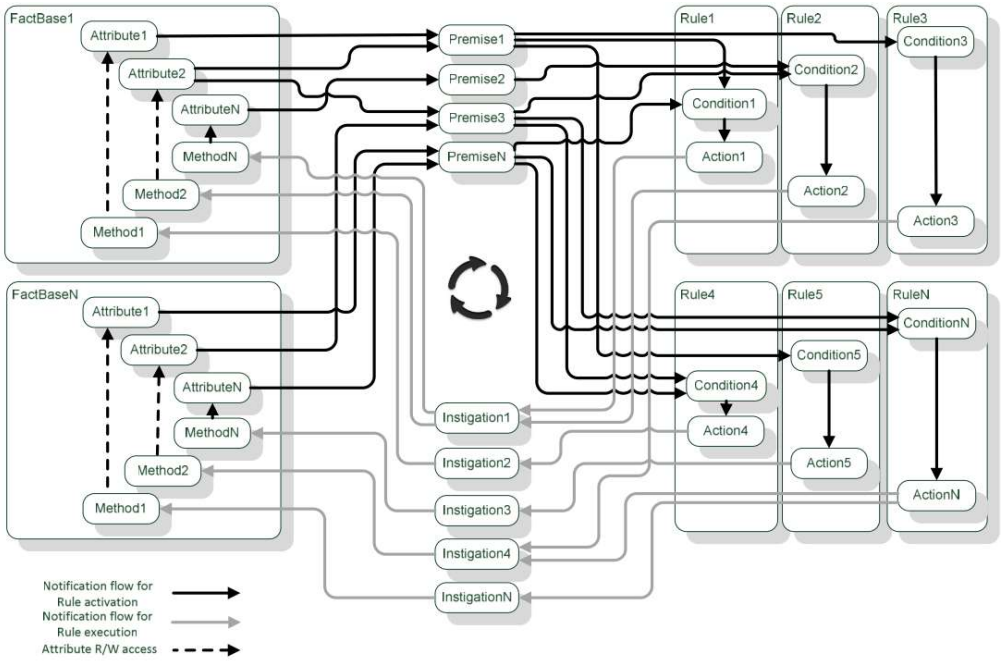


Figura 28 – Esquema de colaboração entre as entidades do metamodelo PON
Fonte: (LINHARES, 2015)

Em resumo, o processamento exemplificado acima se constitui de notificações diretas e precisas entre as entidades que compõem o modelo, de forma reativa e em decorrência de mudanças de estado destas entidades. (PORDEUS, 2017). O processo detalhado de inferência entre as entidades

computacionais do PON pode ser encontrado em literatura específica (SIMÃO e STADZISZ, 2008) (SIMÃO e STADZISZ, 2009) e (SIMÃO e STADZISZ, 2012).

2.5.4 Notação de modelagem RON

A proposta de uma nova abordagem de modelagem de requisitos para engenharia de sistemas se baseou nas seguintes primitivas de modelagem do PON (SIMÃO et al., 2016):

a) **Regra** (*rule*): em PON, uma regra é definida como uma unidade lógica que comanda um conjunto de ações quando suas condições são satisfeitas. Na abordagem RON, a regra representa uma parte ou o todo de um **requisito do sistema**, incluindo restrições tais como pré-condições (*rule preconditions*) e pós-condições (*rule postconditions)*;

b) **Elemento da base de fatos** (*Fact Base Element - FBE*): em PON, um FBE é um elemento de *software* que pode conter atributos e executar funções em métodos. Na abordagem RON, os FBEs são utilizados para representar **elementos do sistema** identificados nas sentenças dos requisitos. Desta forma, FBEs estão limitados aos elementos conhecidos na fase de especificação.

c) **Notificações** (*notifications*): em PON, uma notificação corresponde a uma mensagem entre elementos de *software* PON que indica a ocorrência de mudanças de estado (*i.e.*: mudanças em atributos). Na abordagem RON, as notificações representam ligações (*links*) entre regras e FBEs. Estes *links* podem ter dois significados, de acordo com a direção:

- FBE → Regra: essa ligação indica uma pré-condição. Uma expressão lógica deve ser atribuída na ligação para descrever a condição lógica;
- Regra → FBE: essa ligação indica uma pós-condição. Uma ação deve ser escrita sobre a ligação correspondente.

A notação original proposta (SIMÃO et al, 2016) é mostrada na Figura 29.

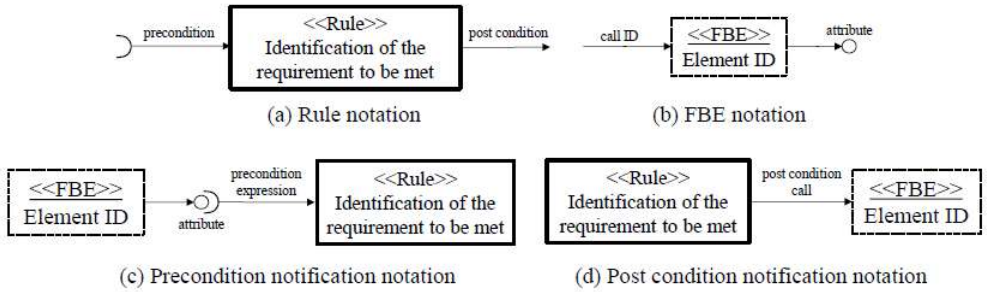


Figura 29 – Notação RON - utilizada para desenhar requisitos de sistemas
Fonte: (SIMÃO et al., 2016)

2.5.5 Técnica de modelagem – NOP Based Requirements Modeling

A partir das primitivas de modelagem RON citadas anteriormente e conforme a notação estabelecida na Figura 29, foi proposta a construção de um processo de requisitos de sistema. A entrada para o processo de modelagem são as sentenças de requisitos do sistema. Uma visão geral da técnica proposta é dada conforme segue (SIMÃO et al., 2016):

"Para cada requisito da especificação de requisitos do sistema, deve-se:

1. Analisar a sentença do requisito focando em:
   i. Identificar requisitos funcionais e não funcionais da sentença do requisito;
   ii. Identificar condições para o requisito funcional ou não funcional;
   iii. Identificar atributos envolvidos nas condições;
   iv. Identificar entidades relacionadas aos atributos para o requisito;
   v. Identificar entidades relacionadas às ações indicadas no requisito;
2. Criar uma regra para cada requisito identificado na etapa 1.
3. Criar uma FBE para cada entidade identificada na etapa 1.
4. Criar ligações (*links),* ou seja, notificações entre regras e FBEs de acordo com as condições e ações relacionadas às regras identificadas na etapa 1.
5. Combinar (unir) FBEs e regras com outras similares que tenham sido criadas previamente."

A etapa 1 envolve analisar cada sentença de requisito de forma a identificar a exigência existente em cada requisito. Essa necessidade pode expressar uma ação que se espera que o sistema realize (requisito funcional) ou pode especificar restrições, propriedades e condições (requisito não-funcional).

Nesta etapa, as condições, atributos, entidades envolvidas nas condições também devem ser identificados (SIMÃO et al., 2016).

As etapas 2 a 4 envolvem a construção do modelo de requisitos em si, a partir dos elementos identificados na etapa 1. A etapa 5 envolve a integração dos modelos gráficos construídos para cada requisito do sistema, através da interconexão e união (*merging*) de regras e FBEs (SIMÃO et al., 2016).

2.5.6 Estudo de Caso: *Access Security System* da INCOSE

Esta seção apresenta um estudo de caso de requisitos extraídos do INCOSE *Systems Engineering Handbook v. 3.2* (INCOSE, 2010). Seis requisitos para um sistema de controle de acesso a uma área de segurança foram estabelecidos (SIMÃO et al., 2016). A tradução dos requisitos originais é apresentada abaixo:

- SS11-a: As áreas seguras devem ser protegidas por verificação de segurança com base no ID do funcionário;
- SS11-b: As áreas seguras devem ser protegidas por uma segunda verificação de segurança independente com base em dados biométricos;
- SS11-c: O tempo entre as duas verificações de segurança independentes não deve exceder um período configurável;
- SS11-d: O usuário deve ter três tentativas de identificação biométrica;
- SS11-e: O usuário terá três tentativas de identificação do cartão;
- SS11-f: Qualquer tentativa de acesso negado deve ser enviada ao administrador.

A partir dos requisitos originais, a técnica de modelagem e a notação foi aplicada, de forma a produzir o modelo final de requisitos do sistema mostrado na Figura 40 (SIMÃO et al., 2016).

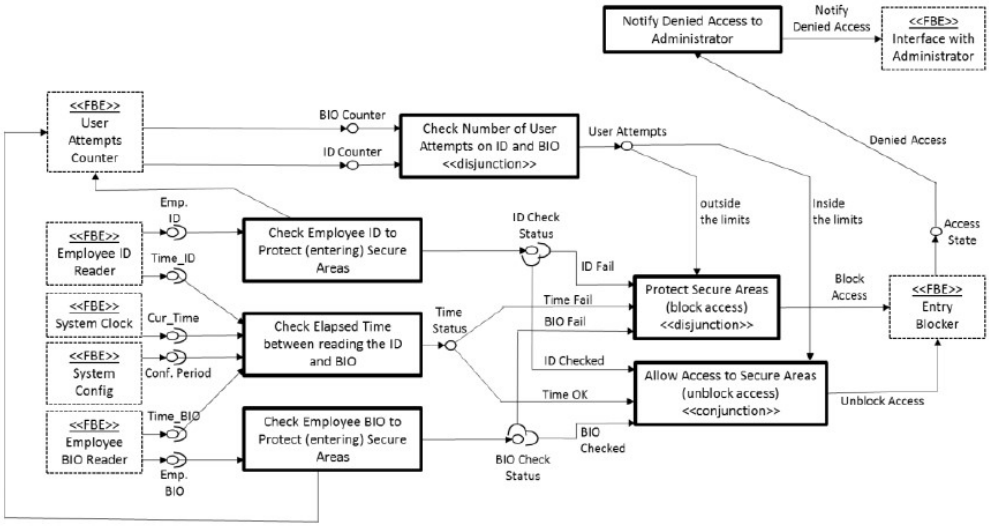


Figura 30 – Modelo final de requisitos do sistema de segurança
Fonte: (SIMÃO et al., 2016)

2.6 FUNDAMENTOS PARA DESENVOLVIMENTO DE MODELOS GRÁFICOS EM RE

O objetivo desta seção é estabelecer fundamentos que possam servir como base para o desenvolvimento de modelos gráficos em RE. Apesar da norma ISO 29148 apresentar recomendações (diretrizes) para a **sintaxe** (forma) e **semântica** (conteúdo) de **requisitos textuais** (ISO/IEC/IEEE 29148, 2011), não há até o momento norma equivalente que estabeleça o mesmo para modelos gráficos em RE**.** Para modelos gráficos, a teoria de física das notações (seção 2.2.3) sugere um conjunto de princípios e recomendações cujo escopo se foca somente **na sintaxe visual** das notações gráficas (Tabela 2). Em relação à **semântica**, não há atualmente normas ou princípios estabelecidos para a criação de um novo modelo gráfico, cabendo aos postulantes de cada sistema de modelagem determinar quais são as diretrizes necessárias.

A estrutura desta seção é a seguinte: a subseção 2.6.1 contém um resumo sobre recomendações sintáticas e semânticas para elaboração de requisitos textuais (ISO/IEC/IEEE 29148, 2011) e a subseção 2.6.2 apresenta objetivos e diretrizes propostas na literatura para os principais modelos gráficos em RE.

2.6.1 Recomendações para a elaboração de requisitos textuais

Esta subseção apresenta recomendações sintáticas e semânticas sugeridas na norma "*Systems and software engineering - Life cycle processes - Requirements engineering*" (ISO/IEC/IEEE 29148, 2011). Em termos de **sintaxe**, a norma estabelece que **requisitos textuais** devem seguir recomendações para a escrita, tais como por exemplo:

a) Utilizar termos na escrita do requisito que o identifiquem como uma obrigação**:** "deve" (*shall*);
b) Utilizar frases positivas (evitar frases negativas);
c) Utilizar voz ativa (evitar frases em voz passiva).

A norma sugere uma sintaxe para a escrita dos requisitos (Tabela 25).

[Condição] [Sujeito] [Ação] [Objeto] [Restrição]
EXEMPLO: Quando o sinal x é recebido **[Condição]**, o sistema **[Sujeito]** deve definir **[Ação]** o sinal x bit recebido **[Objeto]** dentro de 2 segundos **[Restrição].**
OU
[Condição] [Ação ou restrição] [Valor]
EXEMPLO: No estado marítimo 1 **[Condição]**, o Sistema de Radar deve detectar alvos em faixas de **[Ação ou Restrição]** 100 milhas náuticas **[Valor].**
OU
[Sujeito] [Ação] [Valor]
EXEMPLO: O Sistema de Faturamento **[Sujeito]** exibirá faturas de clientes pendentes **[Ação]** em ordem crescente **[Valor]** conforme as faturas devem ser pagas.

Tabela 25 – Exemplos de sintaxe para requisitos
Fonte: traduzido de (ISO/IEC/IEEE 29148, 2011)

Em termos de **semântica**, a norma sugere que **requisitos individuais textuais** devem possuir certas características, para que se capture com precisão as intenções dos *stakeholders* (ISO/IEC/IEEE 29148, 2011):

a) **Necessário**: deve definir uma capacidade, característica, restrição ou fator de qualidade que seja essencial (indispensável);

b) **Independente de implementação**: deve estabelecer o que é necessário, e não como deve ser implementado;
c) **Não-ambíguo**: deve ser escrito de forma que possa ser interpretado somente de uma forma, sem ambiguidades;
d) **Consistente**: deve estar livre de conflitos em relação a outros requisitos;
e) **Completo**: deve conter todo o conhecimento necessário para descrever a necessidade do *stakeholder* sem a necessidade de deduções adicionais, além de ser mensurável e quantificável;
f) **Singular**: deve corresponder somente a uma necessidade do *stakeholder* e evitar o uso de conjunções;
g) **Factível**: deve ser tecnicamente alcançável, não requerer avanços extremos em termos de tecnologia e estar dentro das restrições do sistema com um risco aceitável (*e.g.*: custo, cronograma, técnica, legal, regulatório);
h) **Rastreável**: deve ser rastreável em relação às necessidades dos *stakeholders* (para cima) e em relação a seus sub-requisitos (para baixo). Isso significa que devem ser rastreáveis desde sua origem até a implementação;
i) **Verificável**: deve permitir que haja meios de verificar se o sistema satisfaz o que for estabelecido no requisito.

2.6.2 Objetivos de modelos gráficos em RE

Segundo a SLM apresentada anteriormente (seção 2.3.1), determinados modelos gráficos possuem uma maior incidência nas publicações e pesquisas da área de RE (Tabela 16), incluindo **i*, Tropos, KAOS, NFR, URN e SysML**. Esta subseção visa apresentar os objetivos destas notações mais comuns da área de RE. A ideia é embasar a determinação de características desejáveis para uma nova abordagem de modelagem gráfica de requisitos.

Nesta seção, foram incluídos exemplos de modelagem com o objetivo de permitir ao leitor visualizar as características de cada abordagem, sendo que a descrição destes modelos é dada de forma mais completa no “APÊNDICE B”.

i* é uma abordagem orientada a agentes (e a objetivos) que pode ser usada em RE, reengenharia de processos de negócio, análise de impacto

organizacional e modelagem de processo de *software* (YU, 1995). Dentre suas premissas encontram-se a autonomia de agentes, a intencionalidade, a sociabilidade, a racionalidade, limites e identidade contingentes de agentes e a reflexividade estratégica (YU, 2009).

Os dois diagramas principais da modelagem i* são o *Strategic Dependency Model* (SD), cujo exemplo é apresentado na Figura 31, e o *Strategic Rationale Model* (SR), que pode ser visualizado na Figura 32

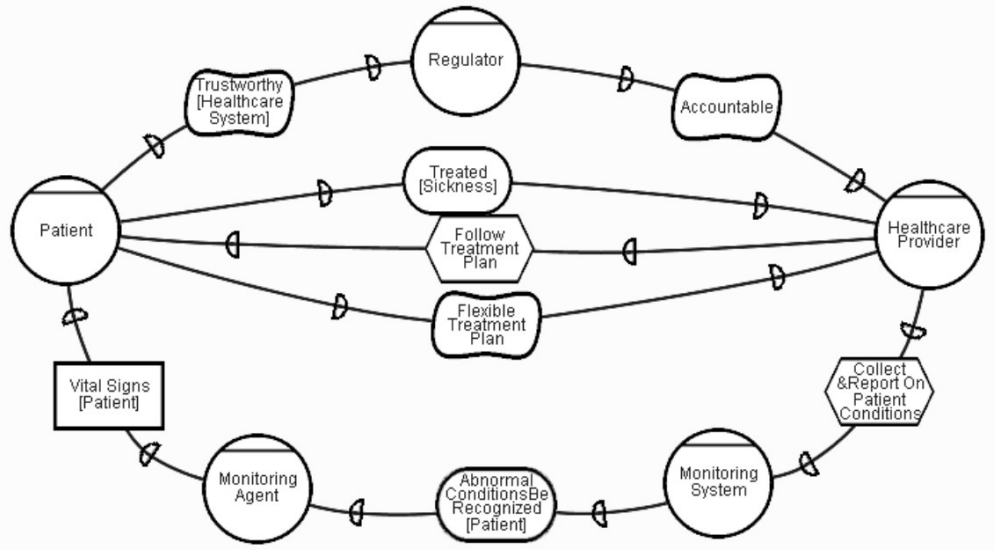


Figura 31 – i* – Exemplo de modelagem SD de um sistema de *HealthCare*
Fonte: (YU, 2002)

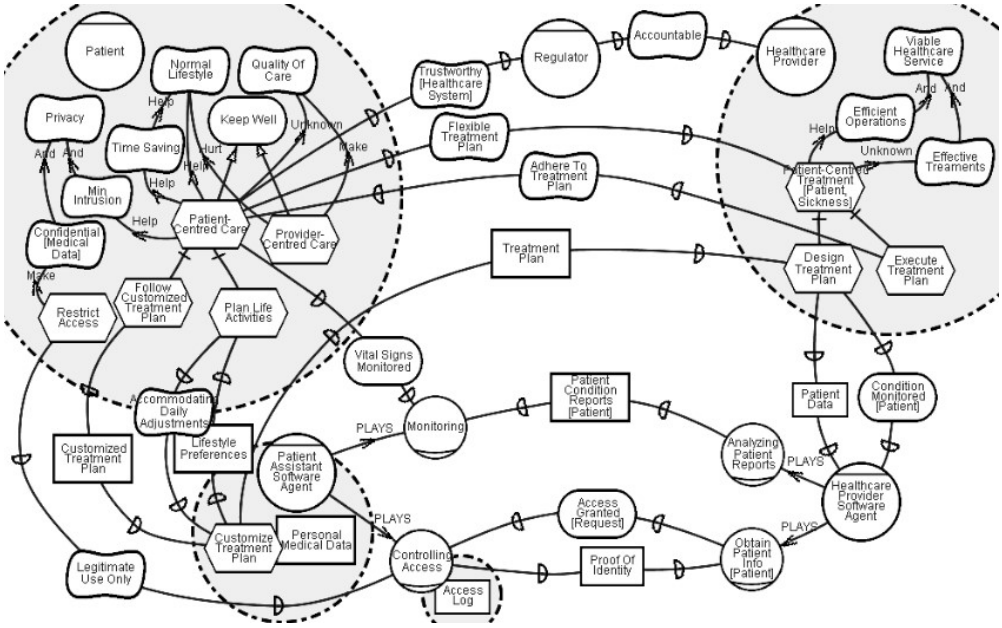


Figura 32 – i* – Exemplo de modelagem SR de um sistema de *HealthCare*
Fonte: (YU, 2002)

Dentre os objetivos da abordagem **i***, é possível destacar (YU, 1997):

a) Suportar a modelagem do contexto de ambientes organizacionais e seus sistemas (dependências estratégicas);
b) Suportar a modelagem da intencionalidade de agentes e *stakeholders* (racionalidade estratégica);
c) Suportar a análise de requisitos em suas fases iniciais (*early-phase RE);*
d) Suportar a representação do conhecimento em RE e sua racionalização;
e) Permitir um grau mínimo de formalização durante as fases iniciais de análise de requisitos (*early-phase RE);*
f) Incorporar a intencionalidade (racionalidade e objetivos) dos agentes aos modelos;
g) Suportar relacionamentos intencionais dos agentes de forma distribuída e de forma multilateral;
h) Suportar o raciocínio de meio-termo (impreciso), que permita captar intenções de agentes ainda não claramente definidas.

TROPOS é uma metodologia que tem o objetivo de modelar sistemas de *software* orientados a agentes (GIORGINI et al., 2004), incluindo etapas tais como *Early Requirements*, *Late Requirements*, projeto de arquitetura, projeto detalhado e implementação. Tropos possui elementos de notação e conceitos derivados do modelo i*. (BRESCIANI et al., 2004).

Os objetivos da abordagem TROPOS são (GIORGINI, 2010):

a) Definir uma metodologia de desenvolvimento de *software* para agentes orientados a sistemas de *software*;
b) Definir uma linguagem tipo UML e uma metodologia para *software* orientado a agentes, que cubra um espectro mais amplo do processo de desenvolvimento de *software*;
c) Cobrir os requisitos iniciais (*early RE*), requisitos posteriores (*late* RE), projeto arquitetônico e projeto detalhado, de forma a interagir com plataformas de programação de agentes.

Um exemplo de diagrama de objetivos da abordagem TROPOS é apresentado na Figura 33.

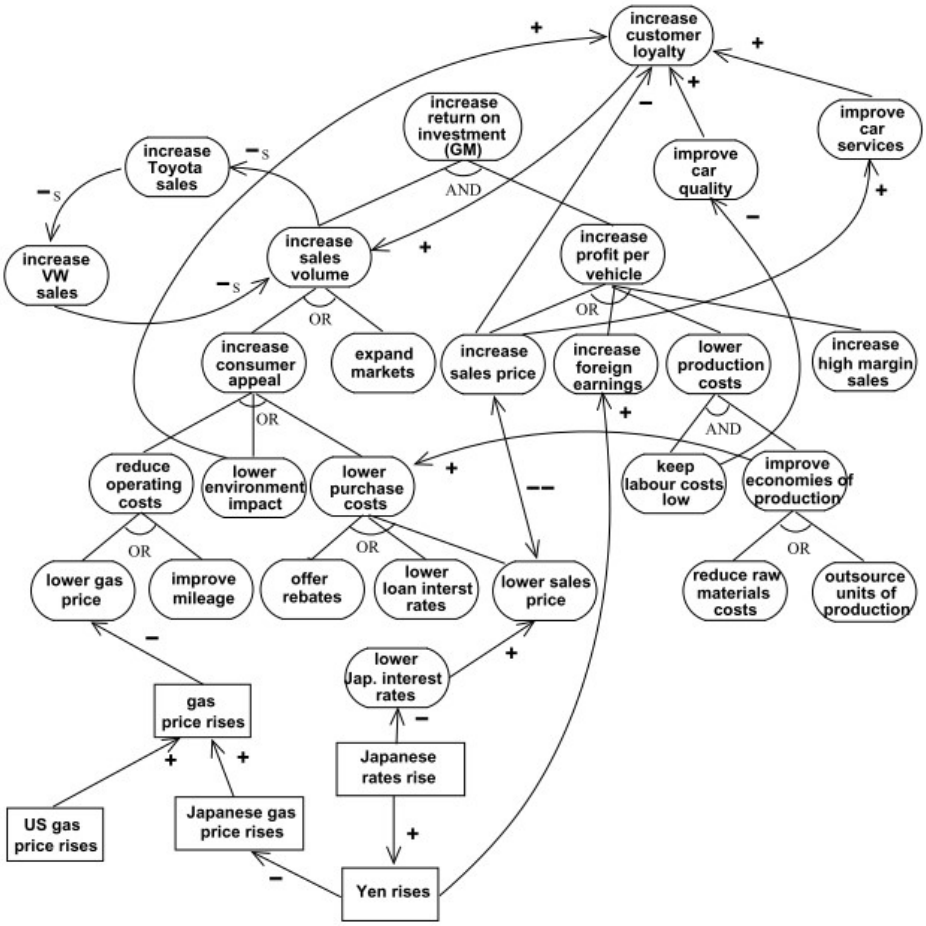


Figura 33 – TROPOS – Exemplo de diagrama de objetivos para a empresa GM
Fonte: (GIORGINI et al., 2004)

KAOS é uma metodologia de aquisição e modelagem de requisitos orientada a objetivos (DARDENNE et al., 1993), descrita como um *framework* multiparadigma (MYLOPOULOS e CHUNG, 1999) com diferentes níveis de expressão: semi-formal para modelagem e estruturação de objetivos; qualitativo para seleção entre alternativas; formal, se necessário, um raciocínio mais preciso (LAPOUCHNIAN, 2005).

Os objetivos da abordagem KAOS são (AL-SUBAIE e MAIBAUM, 2006):

a) Descrever problemas utilizando conceitos predefinidos por meio de modelagem de objetivos;
b) Analisar o problema por meio de uma técnica sistemática para elicitar, descobrir e estruturar objetivos;
c) Identificar papéis e responsabilidades das partes interessadas;

d) Fornecer uma definição formal dos requisitos relativos às partes mais críticas do sistema;
e) Estabelecer uma comunicação eficiente das partes interessadas.

Um exemplo de diagrama de objetivos segundo a abordagem KAOS pode ser visualizado na Figura 34.

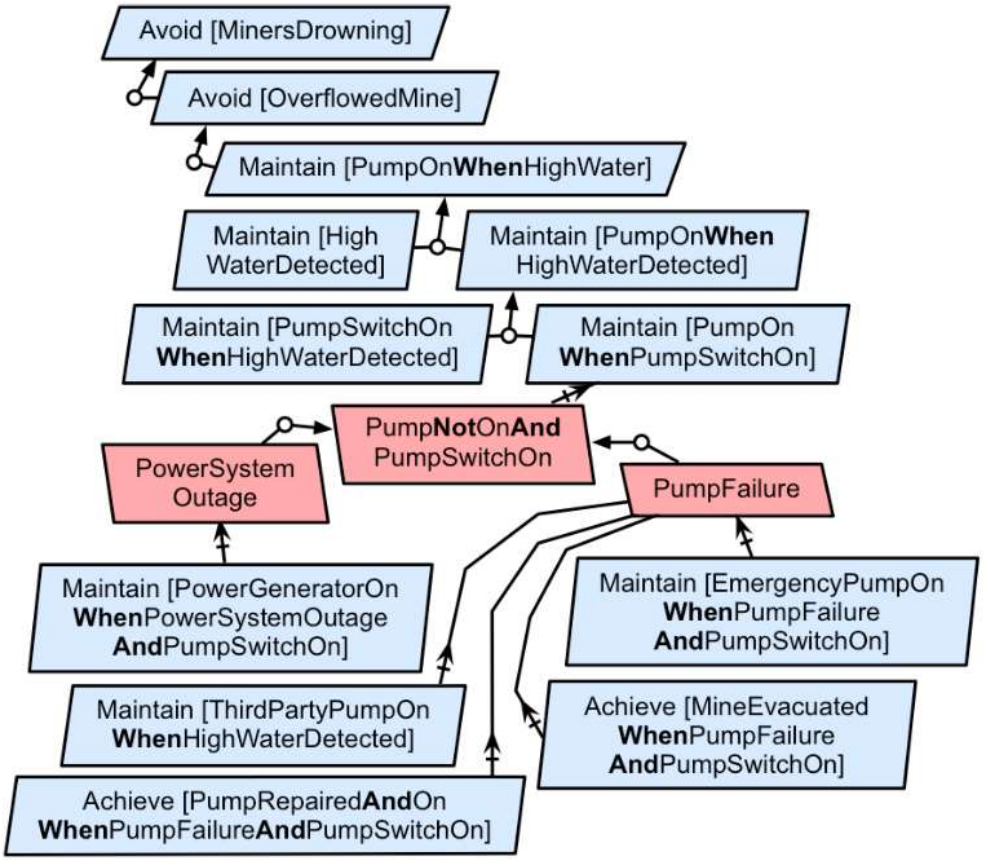


Figura 34 – KAOS – Exemplo de diagrama para um sistema de mineração
Fonte: (CAILLIAU e LAMSWEERDE, 2014)

NFR é um *framework* abrangente usado para representar requisitos não funcionais durante o processo de desenvolvimento (MYLOPOULOS et al., 1992), na forma de uma abordagem orientada a processos, na qual os requisitos não-funcionais são explicitamente representados como metas a serem atingidas (CHUNG et al., 1995). Um exemplo de diagrama da abordagem NFR é apresentado na Figura 35.

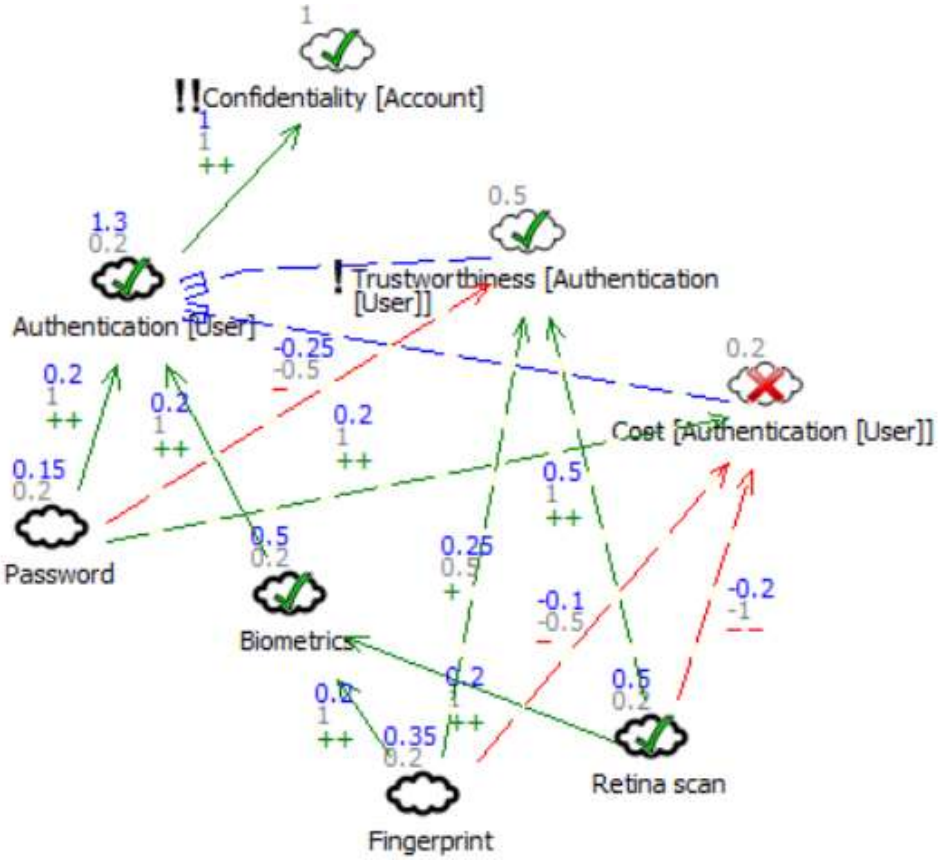


Figura 35 – NFR – Exemplo de diagrama para um sistema de acesso biométrico
Fonte: (SUPAKKUL e CHUNG, 2012)

Suas características e objetivos incluem (CHUNG et al., 1995) (MYLOPOULOS et al., 1992):

a) Modelar requisitos não-funcionais como objetivos a serem atingidos de forma a integrá-los ao processo de desenvolvimento do sistema;
b) Facilitar a identificação de conflitos entre requisitos não-funcionais;
c) Suportar a evolução dos requisitos não-funcionais do sistema por meio da documentação das modificações nestes requisitos;

URN (*User Requirements Notation*) é uma notação gráfica padronizada pela ITU-T como norma internacional (ITU-T Z.151, 2012), usada para modelar e analisar requisitos utilizando objetivos (*goals*) e cenários. Os diagramas propostos pela URN são o diagrama de objetivos GRL (ver exemplo na Figura 36) e o diagrama de mapa de casos de uso UCM (ver exemplo na Figura 37).

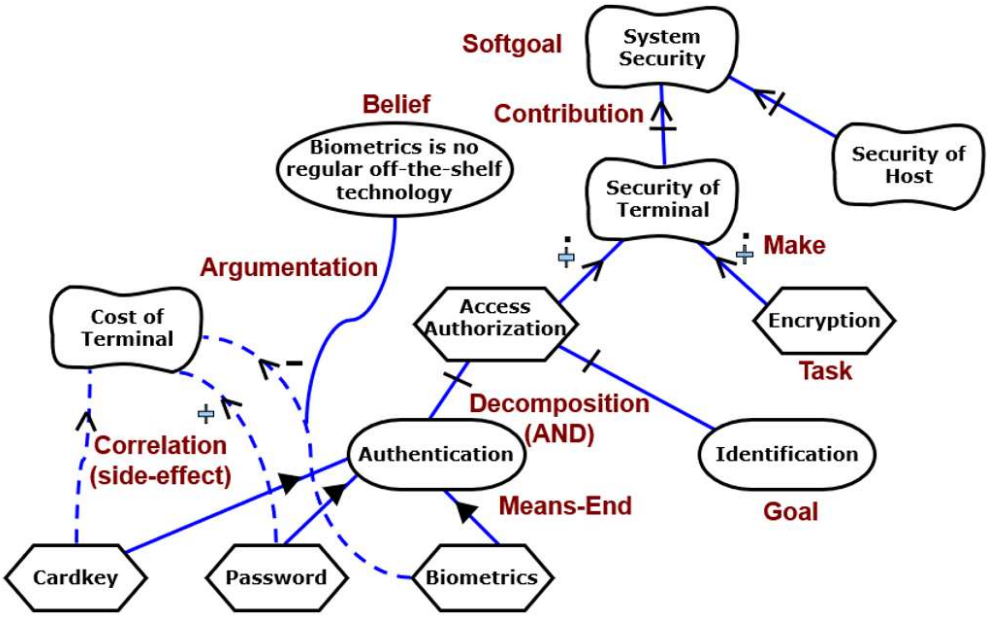


Figura 36 – URN – Exemplo de modelo GRL para sistema de acesso biométrico
Fonte: (AMYOT, 2002)

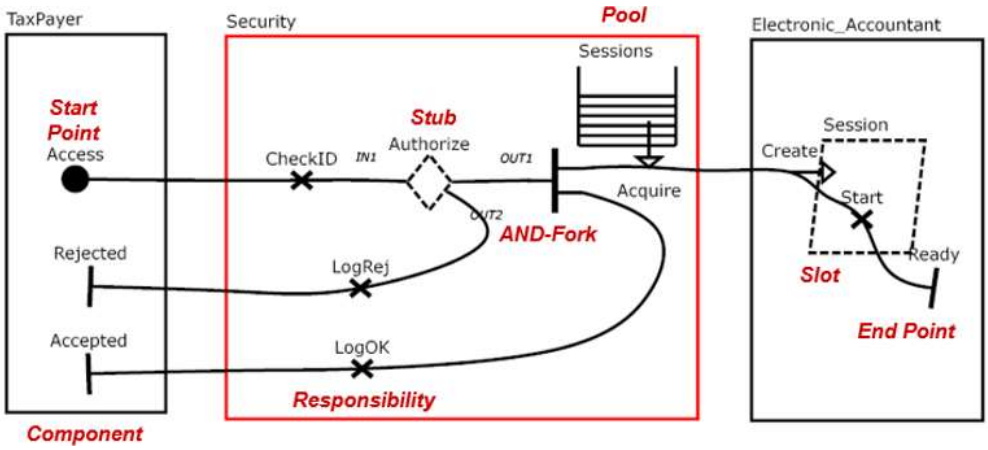


Figura 37 – URN – Exemplo de modelo UCM para sistema de acesso biométrico
Fonte: (AMYOT, 2002)

Os objetivos iniciais propostos para a URN são (AMYOT, 2002):

a) Concentrar-se nos estágios iniciais do *design* utilizando cenários;
b) Capture os requisitos do usuário quando houver poucos detalhes de *design* disponíveis;
c) Nenhuma mensagem, componente ou estado de componente é necessário para a modelagem;
d) Reutilizar cenários e alocação de componentes;
e) Possuir recursos de refinamento dinâmico;

f) Capacidade de modelar sistemas de agentes, analisar desempenho inicial e detectar precocemente interações indesejáveis;
g) Expressar, analisar e lidar com requisitos não funcionais (NFRs);
h) Expressar a relação entre objetivos de negócio e requisitos de sistema;
i) Capturar análise reutilizável (argumentação) e conhecimento de *design* (padrões) para abordar requisitos não funcionais;
j) Conectar-se a outras linguagens da ITU-T (e à UML).

SysML é uma linguagem de modelagem de sistemas (*hardware* + *software*), criada na forma de uma profile da UML padronizada pela OMG em 2007 (OMG, SysML v.1.0, 2007) em colaboração com o INCOSE. A última versão (1.5) está disponível deste maio de 2017 (OMG, SysML v.1.5, 2017). A representação de requisitos na SysML pode ser realizada por meio de um diagrama RD (*requirements diagram*), tal como o exemplo da Figura 38.

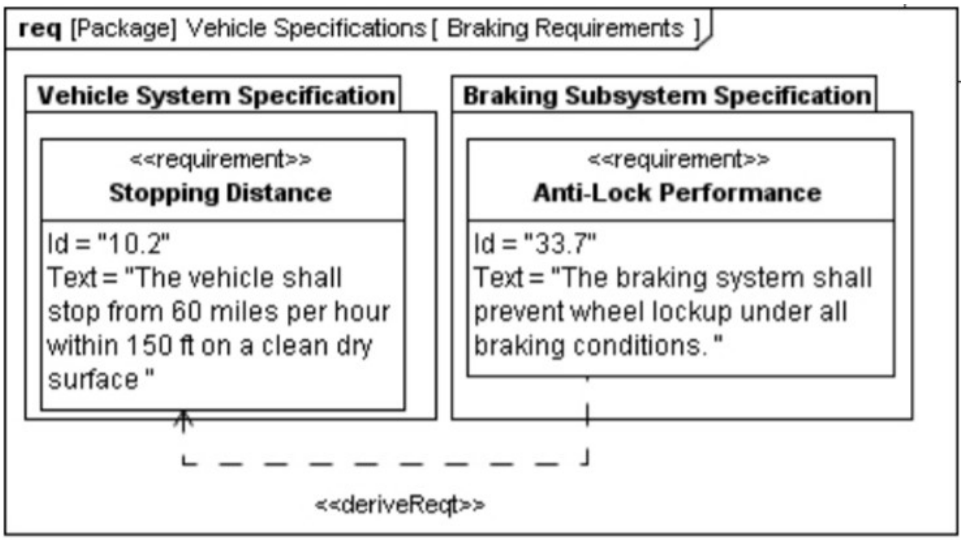


Figura 38 – SysML – Exemplo de diagrama parcial de requisitos de um sistema
Fonte: (FRIEDENTHAL et al., 2008)

As principais características da SysML incluem (FRIEDENTHAL et al., 2008) (FRIEDENTHAL et al., 2014):

a) Suportar a representação de requisitos textuais por meio de um diagrama de requisitos associado a uma representação textual-tabular;
b) Suportar a representação de requisitos funcionais e não-funcionais por meio de um diagrama de requisitos;

c) Suportar relacionamentos entre requisitos e com outros elementos do modelo por meio de interdependências (*e.g.:* satisfy, verify, refine, trace, copy), que facilitem a rastreabilidade entre requisitos e elementos de outros diagramas de modelagem SysML relacionados a projeto, análise, implementação e casos de teste.

2.7 CONCLUSÃO DO CAPÍTULO

O objetivo deste capítulo foi o de fornecer um referencial teórico para fundamentar esta dissertação. Pode-se destacar as seguintes contribuições:

a) A **contextualização** (seção 2.1) apresentou o inter-relacionamento entre as disciplinas de SE, SWE e RE e mostrou como a GRM se insere em processos de RE. Com base nestas informações, posteriormente, a seção 6.4 deste trabalho contextualiza a modelagem RIMON, ao inseri-la no contexto dos processos de RE apresentados na seção 2.1.5;
b) A teoria de **física das notações** (seção 2.2) forneceu o embasamento teórico para o desenvolvimento da linguagem de modelagem gráfica RIMON (capítulo 3), especialmente no que se refere aos aspectos visuais da linguagem (notação visual). Além disso, a subseção 2.2.5 apresenta mecanismos para que seja possível verificar a notação visual, o que é posteriormente realizado na seção 6.1;
c) A **SLM sobre modelos gráficos em RE** (seção 2.3) apresentou os resultados de uma SLM sobre modelos gráficos em RE. Essa SLM fundamentou as diretrizes da linguagem gráfica RIMON propostas na seção 2.3, e forneceu a base de informações sobre os modelos gráficos comparados com a RIMON na seção 6.4;
d) A revisão sobre **tipos de interdependências de requisitos** (seção 2.3.8) forneceu informações para fundamentar o estudo dos tipos de interdependências suportados pela linguagem RIMON (seção 6.2);
e) A revisão sobre **RON** (seção 2.5) forneceu princípios para fundamentar o desenvolvimento da linguagem gráfica RIMON (capítulo 3).
f) Os **fundamentos para desenvolvimento de modelos gráficos em RE**: apresentou uma breve revisão dos fundamentos para o desenvolvimento de modelos gráficos em RE, incluindo objetivos destes modelos que

pudessem fundamentar o desenvolvimento da linguagem RIMON. Estas informações serão utilizadas na definição das diretrizes da linguagem RIMON na seção 3.2.

Por fim, com base no comparativo entre modelos gráficos em RE apresentado e conclusões da subseção 2.3.8, destaca-se o fato de que há características de modelagem pouco exploradas pelas abordagens existentes. Dentre estas características, é possível apontar que a grande maioria das abordagens de modelagem foi desenvolvida sem a preocupação em adotar princípios visuais de PoN, além do fato de que muitos dos modelos identificados não possui um metamodelo ou mesmo a preocupação em definir de forma precisa as interdependências entre requisitos.

A linguagem de modelagem gráfica RIMON proposta neste trabalho tem por objetivo incorporar em seu escopo as características supracitadas, de forma a realizar uma evolução da abordagem RON. Nesse sentido, busca-se neste trabalho a elaboração de uma notação visual embasada em princípios de PoN, a definição de uma linguagem de modelagem gráfica de forma precisa (capítulo 3), e a criação de um método de modelagem (capítulo 4) que sistematicamente permita modelar requisitos e suas interdependências de forma precisa e expressiva.

3 LINGUAGEM DE MODELAGEM GRÁFICA RIMON

Este capítulo tem por objetivo apresentar o desenvolvimento e a definição de uma linguagem de modelagem gráfica para especificação de requisitos por meio de uma notação visual, que se fundamenta nos princípios de física de notações (PoN). Essa linguagem, chamada de **RIMON** (*Requirements and Interdependencies Modeling Notation*), corresponde a uma evolução da abordagem original RON apresentada na seção 2.5 (SIMÃO et al., 2016). A estrutura do capítulo é apresentada em formato BPMN na Figura 39.

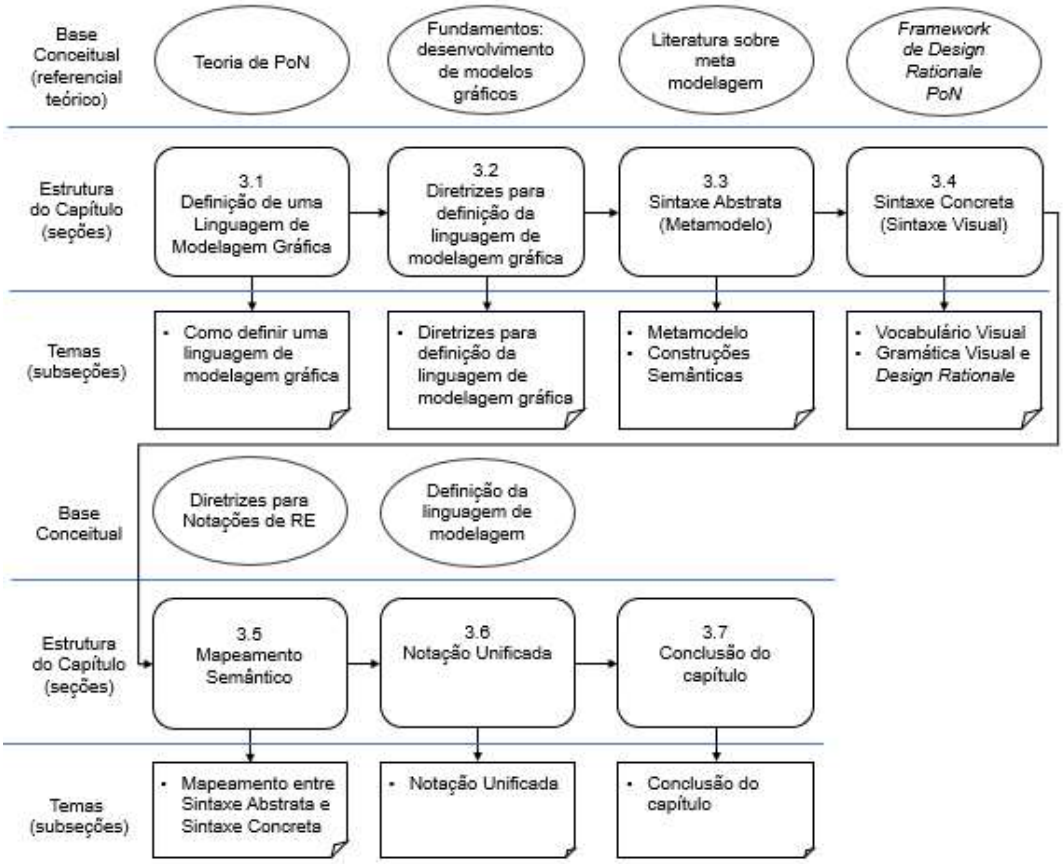


Figura 39 – Mapa conceitual/estrutural: Capítulo 3
Fonte: o autor

3.1 MÉTODO PARA DESENVOLVIMENTO DE LINGUAGENS DE MODELAGEM GRÁFICA BASEADAS EM PoN

Esta seção apresenta um novo método (genérico) para o desenvolvimento de linguagens de modelagem gráfica baseadas em PoN. A necessidade de criar este método surgiu durante o desenvolvimento desta

dissertação, em decorrência da inexistência de métodos específicos para isso na literatura pesquisada pelo autor. A teoria de PoN (MOODY, 2010) é conhecida como o trabalho mais difundido sobre o tema de projeto de aspectos visuais de linguagem de modelagem (TEIXEIRA et al., 2016), ao definir princípios para a criação de notações visuais (ver seção 2.2.3). Esta teoria, no entanto, não prescreve nenhum método para aplicar seus princípios de forma sistemática (STÖRRLE e FISH, 2013) (TEIXEIRA et al., 2016) e não tem por escopo abordar os aspectos semânticos da linguagem (MOODY et al., 2010).

Nesse sentido, o método apresentado que será apresentado nesta seção tem no mínimo com as seguintes características:

a) Sistematiza as fases e etapas necessárias para definição de uma linguagem de modelagem gráfica baseada na teoria de PoN;
b) Inclui etapas para a aplicação e verificação sistemática dos princípios da teoria de PoN;
c) Inclui etapas para a definição semântica da linguagem (e.*g*. metamodelo – vide apêndice D – Terminologia, item 24).

Para definir uma linguagem de modelagem gráfica, usualmente, deve-se definir a **sintaxe abstrata** e a **sintaxe concreta** da mesma (OMG, 2005), bem como um **mapeamento semântico** entre elas (HE et al., 2007).

A **sintaxe abstrata** corresponde a um **metamodelo**, que define os conceitos, relações e restrições dos elementos construtores do modelo (HINKELMANN, 2015). Essa sintaxe é, geralmente, definida por meta-classes, suas relações estruturais e algumas restrições (HE et al., 2007). A criação da sintaxe abstrata também envolve uma definição da **semântica** (*i.e*. significado) dos elementos do metamodelo (HINKELMANN, 2015).

Para linguagens gráficas de modelagem, a **sintaxe concreta** é a notação visual (HE et al., 2007). A sintaxe concreta define a notação e a aparência dos elementos de modelagem (HINKELMANN, 2015).

A Figura 40 apresenta uma representação conceitual de notações visuais segundo a teoria de física das notações. A parte superior apresenta o “nível de tipo”, equivalente ao metamodelo da notação, enquanto a parte inferior

apresenta o nível de instância, correspondente a instanciação do metamodelo em modelos gráficos e seus componentes.

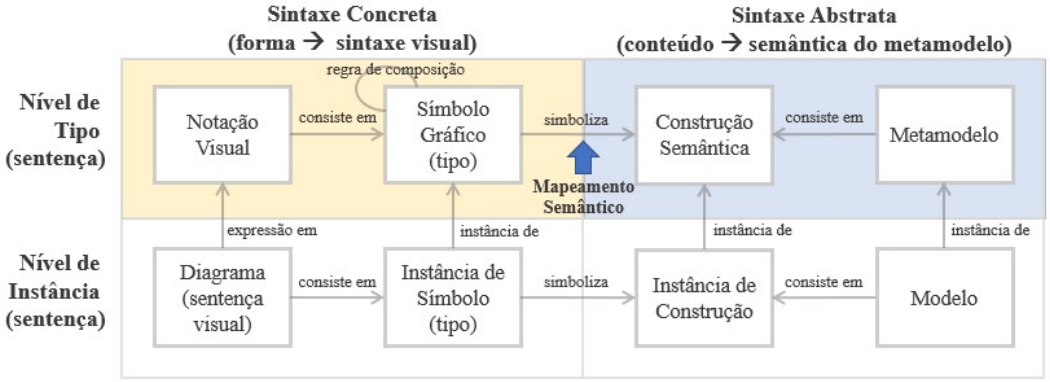


Figura 40 – Representação conceitual de notações visuais segundo PoN
Fonte: Adaptado e traduzido de (MOODY et al., 2010)

Segundo a teoria de PoN (seção 2.2.1), a representação conceitual de notações visuais é dada por (MOODY et al. 2010):

a) **Sintaxe Abstrata** (Figura 40 - parte superior direita), que consiste em um **metamodelo** e suas respectivas **construções semânticas**;
b) **Sintaxe Concreta** ou **Sintaxe Visual** (Figura 40 - parte superior esquerda), que pode ser chamada também de **notação visual,** e é composta por:
   - **Símbolos gráficos** ou **vocabulário visual** – definição visual dos símbolos;
   - **Regras de composição** ou **Gramática visual** - definição de regras para relacionamentos visuais entre símbolos**;**
c) **Semântica visual** ou **Mapeamento Semântico** (Figura 40 - parte central): define o significado de cada símbolo gráfico ao realizar a correlação entre a sintaxe concreta e a sintaxe abstrata.

Com base nos conceitos do referencial teórico (PoN e *Design Rationale*) e na exposição acima (metamodelagem e PoN), esta dissertação propõe um método para a definição e o desenvolvimento de notações visuais, ilustrado na Figura 41 em formato BPMN.

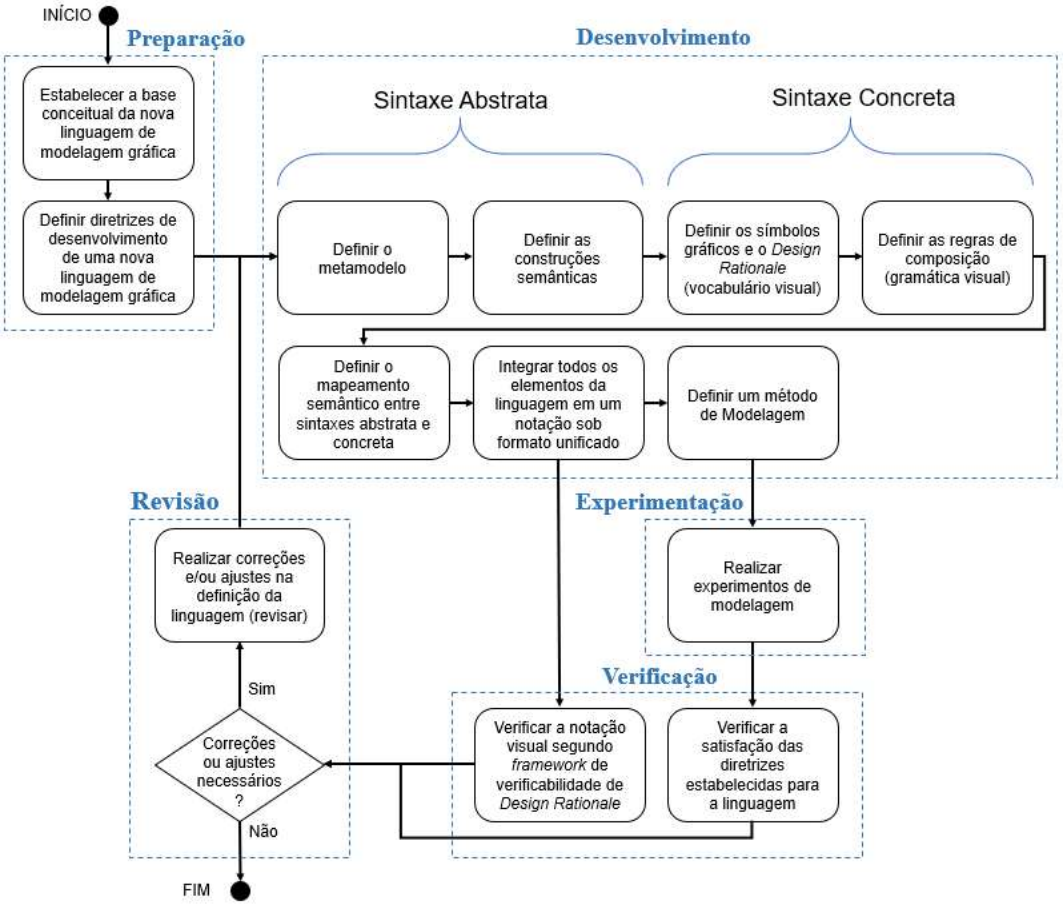


Figura 41 – Método para desenvolvimento de linguagens de modelagem gráfica
Fonte: o autor

Esse método se subdivide nas seguintes etapas:

a) **Preparação**: consiste em estabelecer a base conceitual da nova linguagem de modelagem gráfica (realizado nas seções 2.5 e 2.6) bem como definir as diretrizes para a definição da linguagem (seção 3.2);
b) **Desenvolvimento**: contempla as atividades de definição da sintaxe abstrata (seções 3.3 e 3.4), da sintaxe concreta (seção 3.4) e do mapeamento semântico entre elas (seção 3.5). Também inclui a integração de todos os elementos gráficos em uma notação unificada (seção 3.6) e a definição do método de modelagem (capítulo 4);
c) **Experimentação**: consiste na realização de experimentos práticos de modelagem (capítulos 4.4 e 5.2), visando aplicar na prática o método e a linguagem gráfica proposta;
d) **Verificação**: consiste em verificar se a linguagem gráfica foi desenvolvida como uma notação visual condizente com os princípios de

PoN e verificar se a finalidade (objetivo) esperada para a linguagem foi atingido, por meio de:

- Verificação da notação visual sob a perspectiva do *framework* de verificabilidade de *Design Rationale*;
- Verificação da satisfação das diretrizes estabelecidas para a linguagem gráfica.

e) **Revisão**: consiste em revisar a definição da linguagem de modelagem gráfica, caso ajustes ou correções sejam identificados no processo de verificação. Estes ajustes devem ser realizados em um novo ciclo do processo de desenvolvimento da linguagem.

O método proposto acima foi elaborado a partir da união de diversos conceitos e referências, e o **porquê de sua atual forma** pode ser explicado conforme segue:

a) A fase de **preparação**, que estabelece fundamentos e diretrizes para definir a linguagem de modelagem gráfica, foi inspirada em ideia proposta na linguagem URML (BERENBACH et al., 2012), que propõe **requisitos** para definir uma nova notação visual específica para uso industrial **antes** de realizar a criação da linguagem;

b) A fase de **desenvolvimento** está baseada na teoria **PoN** que foi apresentada na seção 2.2, especialmente no que se refere a parte visual da notação (sintaxe concreta). Como a teoria de PoN não estabelece como definir a semântica da linguagem (Sintaxe abstrata), foi necessário elaborar esta fase do método segundo conceitos gerais de metamodelagem (OMG, 2005) (HINKELMANN, 2015) e também com base em artigo específico (HE et al., 2007), em que um metamodelo para a notação de linguagens gráficas de modelagem é apresentado;

c) A fase de **experimentação** foi incluída para a linguagem de modelagem gráfica fosse utilizada (em experimentos), de forma a permitir sua posterior verificação;

d) A fase de **verificação** foi criada de forma a aplicar os conceitos de verificação de notações baseadas em PoN, conhecido como *framework* de verificabilidade de *Design Rationale* (LINDEN et al., 2017). Este *framework* foi apresentado na seção 2.2.5 desta dissertação. Nesta fase, também foi criada uma etapa para verificar se a linguagem de

modelagem gráfica atende as diretrizes estabelecidas na fase de preparação;

e) Por fim, a fase de **revisão** tem por objetivo garantir que eventuais falhas encontradas durante as fases de experimentação ou verificação sejam sanadas, por meio de um laço que permite retornar à fase de desenvolvimento.

3.2 DIRETRIZES PARA DEFINIÇÃO DA LINGUAGEM RIMON

Com base nas recomendações, objetivos e diretrizes potenciais das subseções anteriores, foi definido um conjunto de diretrizes para a definição de uma nova notação visual, que será materializada posteriormente neste trabalho por meio de uma nova linguagem de modelagem gráfica.

Os **fundamentos** para estas diretrizes estão baseados em:

i) **Objetivo Geral** (seção 1.2.1): as diretrizes devem estar alinhadas (ou derivarem) com o objetivo geral desta dissertação: “Desenvolver uma linguagem e um método de modelagem gráfica de requisitos de *software* e sistemas, que permita representar requisitos e suas interdependências de forma sistemática, precisa e expressiva”;

ii) **Teoria de PoN** (seção 1.1)**:** a notação visual deve se fundamentar nos princípios da teoria de PoN, desde que aplicáveis: Clareza Semiótica, Discriminabilidade Perceptiva, Transparência Semântica, Gerenciamento de Complexidade, Integração Cognitiva, Expressividade Visual, Codificação Dual, Economia Gráfica, Ajuste Cognitivo;

iii) **Modelos gráficos em RE** (seção 2.6.2): serão escolhidas diretrizes derivadas dos objetivos propostos em outros modelos gráficos em RE que estejam alinhados com o objetivo principal da presente dissertação.

A Tabela 26 apresenta o conjunto de diretrizes selecionadas para a nova linguagem de modelagem gráfica a ser desenvolvida neste trabalho. Para simplificar a escrita dos requisitos, foi utilizado o termo **abordagem de modelagem**, para se referir ao conjunto dos termos citados no objetivo geral: linguagem de modelagem gráfica e método de modelagem.

No	Origem	Diretriz (descrição)
D.1	Objetivo Geral	A abordagem de modelagem deve permitir modelar requisitos tanto de *software* quanto de sistemas.
D.1.1	Objetivo Geral	A abordagem de modelagem deve permitir modelar requisitos de ***software.***
D.1.2	Objetivo Geral	A abordagem de modelagem deve permitir modelar requisitos de **sistemas.**
D.2	Objetivo Geral	A abordagem de modelagem deve permitir modelar requisitos e suas interdependências de forma sistemática, precisa e expressiva.
D.2.1	Objetivo Geral	A abordagem de modelagem deve permitir modelar requisitos e suas interdependências de forma **sistemática.**
D.2.2	Objetivo Geral	A abordagem de modelagem deve permitir modelar requisitos e suas interdependências de forma **precisa.**
D.2.3	Objetivo Geral	A abordagem de modelagem deve permitir modelar requisitos e suas interdependências de forma **expressiva**.
D.3	Teoria PoN	A abordagem de modelagem deve satisfazer os seguintes princípios de notações visuais de PoN: clareza semiótica, discriminabilidade perceptiva, transparência semântica, expressividade visual, codificação dual, economia gráfica, ajuste cognitivo.
D.3.1	Teoria PoN	A abordagem de modelagem deve satisfazer ao princípio PoN de **clareza semiótica**, ou seja, deve haver uma correspondência semântica de um para um entre construções semânticas e símbolos gráficos.
D.3.2	Teoria PoN	A abordagem de modelagem deve satisfazer ao princípio PoN de **discriminabilidade perceptiva**, ou seja, símbolos devem ser claramente distinguíveis uns dos outros.
D.3.3	Teoria PoN	A abordagem de modelagem deve satisfazer ao princípio PoN de **transparência semântica**, ou seja, usar símbolos cuja aparência sugira seu significado.
D.3.4	Teoria PoN	A abordagem de modelagem deve satisfazer ao princípio PoN de **expressividade visual**, ou seja, usar toda a gama e capacidades de variáveis visuais na representação de símbolos.

No	Origem	Diretriz (descrição)
D.3.5	Teoria PoN	A abordagem de modelagem deve satisfazer ao princípio PoN de **codificação dual**, ou seja, usar texto para complementar gráficos.
D.3.6	Teoria PoN	A abordagem de modelagem deve satisfazer ao princípio PoN de **economia gráfica**, ou seja, manter gerenciável cognitivamente o número de diferentes símbolos gráficos.
D.3.7	Teoria PoN	A abordagem de modelagem deve satisfazer ao princípio PoN de **ajuste cognitivo**, usar diferentes dialetos visuais para diferentes tarefas e/ou públicos (audiências).
D.4	Modelos gráficos (SysML)	A abordagem de modelagem deve permitir representar requisitos funcionais e não-funcionais por meio de um diagrama de requisitos.
D.5	Modelos gráficos (NFR)	A abordagem de modelagem deve possibilitar a identificação visual de conflitos em requisitos funcionais e não-funcionais.

Tabela 26 – Diretrizes para uma nova notação de modelagem de requisitos
Fonte: o autor

Em relação às diretrizes definidas acima é possível notar que:

a) As diretrizes 1 e 2 e derivadas foram embasadas no objetivo geral;
b) A diretriz 3 e derivadas foram especificadas com base na teoria de PoN. Dois princípios **não** foram aplicados:
   - Gerenciamento de Complexidade: este princípio se refere à inclusão de mecanismos explícitos para lidar com a complexidade, ou seja, capacidade de suportar a modelagem de sistemas com grandes quantidades de requisitos. O motivo da não aplicação deste princípio neste trabalho é que isto demandaria aumentar de forma significativa o esforço e tempo necessários para o desenvolvimento da notação, em função da complexidade envolvida na aplicação deste princípio. Sugere-se que a inclusão deste princípio seja realizada em trabalhos futuros;
   - Integração Cognitiva: este princípio se refere à inclusão de mecanismos explícitos para integração de informações de diagramas diferentes. Este princípio não é aplicável a este trabalho, que possui somente um tipo de diagrama.

c) A diretriz 4 foi definida a partir de características da linguagem **SysML**: “Suportar a representação de requisitos funcionais e não-funcionais por meio de um diagrama de requisitos”.
d) A diretriz 5 foi definida a partir de características da modelagem **NFR**, que é focada na modelagem de requisitos não-funcionais por meio de objetivos. Desta abordagem, foi extraída a ideia de “facilitar a identificação de conflitos entre requisitos não-funcionais”. Além disso, a diretriz foi estendida para a identificação de requisitos funcionais;
e) Nenhuma diretriz foi escolhida a partir das abordagens **i*, Tropos e KAOS e URN**, pois são modelos orientados a agentes e objetivos cujos conceitos diferem muito da abordagem proposta nesta dissertação.

Para facilitar a futura difusão desta pesquisa em periódicos e conferências internacionais, optou-se neste trabalho por realizar a definição da notação nas próximas seções (diagramas e figuras) na língua inglesa, com tradução para português onde se julgou necessário.

3.3 SINTAXE ABSTRATA – METAMODELO

Esta seção apresentará o metamodelo e a semântica dos elementos do metamodelo (construções semânticas).

3.3.1 Metamodelo

Para definir uma linguagem de modelagem gráfica, é necessário definir a notação gráfica em um processo que compreende a metamodelagem (HE et al., 2017). Metamodelagem é o processo de gerar metamodelos (modelos de modelos), que podem ser entendidos como sendo construtores a serem utilizados para expressar modelos (HINKELMANN, 2015). Um metamodelo corresponde a um modelo de informações lógicas que especifica os elementos de modelagem usados em outra (ou na mesma) notação de modelagem (ISO/IEC/IEEE 24765, 2010).

A partir da análise dos conceitos propostos pela abordagem RON (seção 2.5), dos requisitos para notações visuais (seção 2.6) e com base na experiência obtida pelo autor durante as aplicações da modelagem gráfica proposta (capítulo 5), foi possível elaborar o metamodelo apresentado na Figura 42.

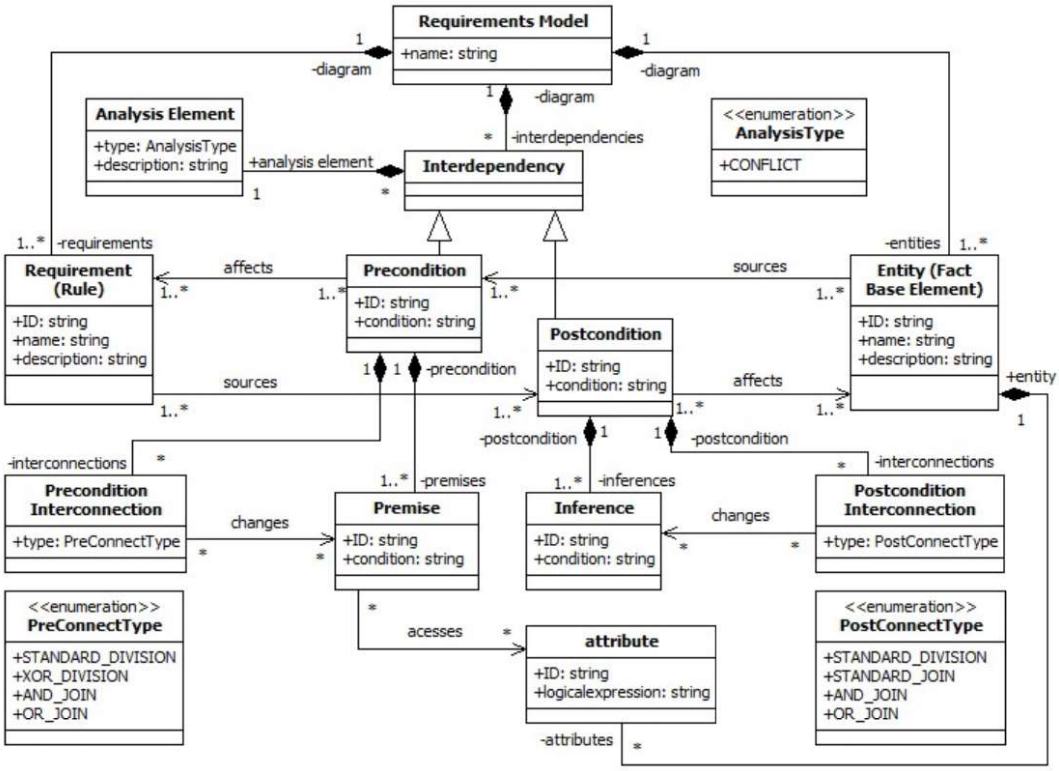


Figura 42 – Metamodelo proposto para a notação visual RIMON
Fonte: o autor

O metamodelo acima foi elaborado com base na notação de meta-classes da UML (OMG, 2005), que usualmente é utilizada para realizar este tipo de representação (HE et al., 2007) (HINKELMANN, 2015). A próxima subseção descreve as construções semânticas (classes) do metamodelo proposto.

3.3.2 Construções Semânticas

A semântica dos elementos do metamodelo (Figura 42) é apresentada a seguir, acompanhada de definições sobre os conceitos empregados:

a) ***Requirements Model*** (modelo de requisitos)**:** é uma abstração semântica fechada sobre os requisitos ou uma descrição completa de um sistema sob a perspectiva de engenharia de requisitos. Este conceito está lastreado na definição de modelo (ISO/IEC/IEEE 24765, 2010), aplicada a requisitos.

A instância do ***Requirements Model*** neste metamodelo corresponde a um diagrama (*diagram*), composto por um ou mais requisitos (*requirement)*, uma ou mais entidades *(entity or FBEs*) e por zero a n interdependências (*Interdependency*).

b) ***Interdependency*** (interdependência): “interdependência entre requisitos é um relacionamento entre requisitos” (ZHANG et al., 2014), em que o conceito de “relacionamento é definido como uma “conexão semântica entre elementos de modelagem” (ISO/IEC/IEEE 24765, 2010).

   No metamodelo proposto, ***Interdependency*** corresponde a uma abstração de dois possíveis tipos de construções semânticas: a *Precondition* (pré-condição) ou a *Postcondition* (pós-condição). Estas construções semânticas serão definidas logo na sequência nesta subseção;

c) ***Requirement / Rule*** (requisito / regra): “requisito é uma declaração que traduz ou expressa uma ‘necessidade’ (*i.e.:* uma demanda ou exigência), bem como suas restrições e condições associadas” (ISO/IEC/IEEE 29148, 2011).

   Nesta definição semântica, o ***Requirement*** tem correspondência com o conceito de *Rule* (Regra) do paradigma PON. No PON, a *Rule* é definida como uma unidade lógica-causal, que comanda um conjunto de ações quando suas condições são satisfeitas (SIMÃO et al., 2016). Como construção semântica do metamodelo, um *Requirement* representa parte ou o todo de um requisito especificado para o sistema, sendo afetada por restrições com relação às exigências (*i.e.*: pré-condições do requisito) e geradora de efeitos potenciais a serem executados (*i.d.*: pós-condições da regra) para este requisito.

d) ***Entity / Fact Base Element*** (entidade / elemento da base de fatos): “uma entidade é um objeto (*e.g.* coisa, evento ou conceito) que ocorre em um modelo”. Também pode ser entendida como “uma coisa de fundamental relevância para o usuário, sobre a qual uma informação é mantida” (ISO/IEC/IEEE 24765, 2010).

   A construção semântica ***Entity*** tem correspondência com o conceito de *Fact Base Element* (FBE ou elemento da base de fatos) do paradigma PON. No PON, um FBE é uma entidade ou elemento de *software* que armazena informações em atributos, contém métodos para a execução de ações e pode interfacear com elementos externos tais como sensores, dispositivos e interfaces de usuário (SIMÃO et. al, 2016). A

construção semântica *Entity* no presente metamodelo representa elementos do sistema (físicos e abstratos), identificados nas sentenças dos requisitos durante o processo de especificação dos requisitos do sistema. A *Entity* também pode agregar *Attributes* (atributos), que serão explicados posteriormente nesta seção.

e) ***Precondition*** (pré-condição): "é uma condição que se requer que seja verdadeira antes de se fazer a solicitação ou operação" (ISO/IEC/IEEE 24765, 2010), sendo que condição neste contexto é "um atributo qualitativo ou quantitativo mensurável que é estipulado para um requisito" (ISO/IEC/IEEE 29148, 2011).

   A construção semântica da ***Precondition*** é uma especialização de uma *Interdependency* (interdependência), sendo composta de *Premises* (premissas) e *Precondition Interconnections* (interconexões de pré-condição). Além disso, uma ou mais *Preconditions* podem se originar (*sources)* a partir de uma *Entity* (entidade) e afetar (*affects*) um *requirement* (requisito).

f) ***Premise*** (premissa): "é uma afirmação ou proposição anterior da qual outra pode ser inferida ou seguir como uma conclusão" (OXFORD DICTIONARY, 2018).

   No presente metamodelo as ***Premises*** compõem uma pré-condição e podem combinar-se entre si em cláusulas lógicas por meio de *Precondition Interconnections* (interconexões de pré-condição). Uma *Premise* pode acessar *Attributes* (atributos) de *Entities* (entidades). Os atributos podem estar associados a expressões e condições lógico-matemáticas.

g) ***Attribute*** (atributo): "é uma propriedade associada a um conjunto de coisas reais ou abstratas, e que é uma característica de interesse" (ISO / IEC / IEEE 24765, 2010).

   A construção semântica de um ***Attribute*** está associada a uma *Entity* (entidade) ou FBE. Um *Attribute* pode ser acessado por *Premises* (premissas), além de poder estar associado a propriedades paramétricas ou expressões lógico-matemáticas.

h) ***Precondition Interconnection*** (interconexão de pré-condição): construção semântica que permite modificar as *Premises* de uma pré-

condição em termos de cardinalidade (divisão gráfica da linha da premissa, sem modificar o conteúdo) ou em termos de conteúdo (operações de XOR, AND e JOIN), por meio de expressões lógicas de *join* (união) ou *division* (divisão):

- *STANDARD_DIVISION*: operação de divisão (*division*) "gráfica" de *Premises* em que o conteúdo das *Premises* resultantes é igual a *Premises* de entrada. Assim, essa interconexão ocorre em um diagrama por meio da divisão gráfica da linha da premissa em 2 ou mais linhas equivalentes;
- *XOR_DIVISION:* operação de união (*join*) de *Premises* sob o operador lógico "ou exclusivo" (*xor*), em que as *Premises* resultantes são mutualmente exclusivas umas em relação as outras;
- *AND_JOIN*: operação de união (*join*) de *Premises* sob o operador lógico "e" (*and*), em que a premissa resultante corresponde a uma conjunção das *Premises* de entrada;
- *OR_JOIN:* operação de união (*join*) de *Premises* sob o operador lógico "ou" (*or*), em que a *Premise* resultante corresponde a uma disjunção das *Premises* de entrada.

i) ***Postcondition*** (pós-condição): "é uma condição que se garante que seja verdadeira após uma solicitação ou operação bem-sucedida" (ISO/IEC/IEEE 24765, 2010), em que "condição é um atributo qualitativo ou quantitativo mensurável que é estipulado para um requisito" (ISO/IEC/IEEE 29148, 2011).

Na presente definição semântica, uma ***Postcondition*** é uma especialização de uma *Interdependency* (interdependência), sendo composta de *Inferences* (inferências) e *Postcondition Interconnections* (interconexões de pós-condição). Além disso, uma *Postcondition* pode se originar (*sources)* a partir de um *Requirement* (*requisito*) e afetar (*affects*) uma *Entity* (entidade).

No PON, o elemento semântico correspondente à *Postcondition* é a **ação**, que normalmente se subdivide em **instigações**. Uma ação do PON corresponde a um elemento dinâmico de *software*, ou seja, que realiza notificações em outras entidades de *software* e ativa a execução de métodos em suas instigações (SIMÃO et al., 2012).

j) ***Inference*** (inferência): “é o ato de passar de uma proposição, afirmação ou julgamento considerado verdadeiro para outro cuja verdade se acredita que decorra do primeiro” (MERRIAN-WEBSTER DICTIONARY, 2018).

   A construção semântica da ***Inference*** compõe uma *Postcondition* (pós-condição). *Inferences* podem se combinar em cláusulas lógicas por meio de *Postcondition Interconnections* (interconexões de pós-condição). Uma *Inference* pode conter ações potenciais (efeitos), descritas como *Postconditions* (pós-condições), que podem afetam *Entities* (entidades). No PON, o elemento semântico correspondente à inferência é a **instigação**. A instigação do PON é uma subdivisão da ação e corresponde a um elemento dinâmico de *software*, ou seja, que ativa a execução de métodos (SIMÃO et al., 2012).

k) ***Postcondition Interconnection*** (interconexão de pós-condição): trata-se de uma construção semântica que permite modificar as *Inferences* (inferências) de uma *Postcondition* (pós-condição) em termos de cardinalidade (divisão ou união padrão sem modificar o conteúdo) ou em termos de conteúdo (operadores AND e OR), por meio de expressões lógicas de união (*join*) ou divisão (*division*):

   - *STANDARD_DIVISION*: operação de divisão (*division*) de *Inferences* em que o conteúdo das *Inferences* resultantes é igual a *Inferences* de entrada. Assim, essa interconexão ocorre em um diagrama por meio da divisão gráfica da linha da inferência em 2 ou mais linhas equivalentes;
   - *STANDARD_JOIN*: operação de união (*join*) de *Inferences* em que o conteúdo da *Inference* resultante é igual as *Inferences* de entrada. Assim, essa interconexão ocorre em um diagrama por meio da união gráfica de 2 ou mais linhas em uma única inferência;
   - *AND_JOIN*: operação de união (*join*) das *Inferences* sob o operador lógico “e” (*and*), em que a *Inference* resultante corresponde a uma conjunção das *Inferences* de entrada;
   - *OR_JOIN*: operação de união (*join*) das *Inferences* sob o operador lógico “ou” (*or*), em que a *Inference* de saída corresponde a uma disjunção das *Inferences* de entrada.

l) ***Analysis Element*** (elemento de análise): permite realizar a análise de interdependências entre requisitos. Neste metamodelo, o elemento de análise poderá ser utilizado opcionalmente para identificar **conflitos** em interdependências de requisitos:

- *CONFLICT*: sinaliza a existência de um conflito em determinada interdependência de um requisito. Por definição, “um requisito conflita com outro requisito se ambos não podem existir simultaneamente ou se o incremento da satisfação de um requisito provoca o decremento da satisfação do outro” (DAHLSTEDT e PERSSON, 2005). No presente metamodelo, um conflito pode ocorrer em uma pré-condição de um requisito (quando as premissas de entrada apresentam condições conflitantes) ou relacionadas a pós-condições (quando as inferências de saída apresentam condições conflitantes).

3.4 SINTAXE CONCRETA – SINTAXE VISUAL

Nesta seção, a sintaxe concreta da modelagem será apresentada na forma de dois **dialetos visuais**. Dialetos visuais são variações na sintaxe concreta da notação visual (não-semânticas) de forma a permitir sua utilização por diferentes públicos e para finalidades distintas. A criação de dialetos visuais é o mecanismo sugerido para atender ao princípio de PoN de **ajuste cognitivo** (VESSEY e GALLETTA, 1992) (MOODY et al., 2010), de forma a permitir sua utilização ajustada a diferentes audiências (LINDEN et al., 2017). Os dialetos visuais propostos são os seguintes:

a) **Notação Técnica (dialeto visual 1)**: caracteriza- se por utilizar símbolos mais simples para requisitos (regras) e entidades (FBEs), sem a utilização de variáveis visuais elaboradas (cores ou imagens, por exemplo), porém seguindo alguns princípios da teoria de PoN. Sua finalidade é permitir a modelagem de requisitos de um ponto de vista técnico, para atender primariamente necessidades de pesquisadores de RE (comunidade científica) ou engenheiros de requisitos;

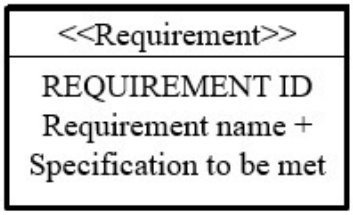
b) **Notação de Negócios (dialeto visual 2)**: se caracteriza por aplicar em maior medida os princípios de PoN, através da utilização de variáveis

visuais tais como cores, ícones e imagens. Sua finalidade é tornar atrativa e intuitiva a notação para fins de modelagem de requisitos em ambientes empresariais, melhorando a comunicação entre as partes envolvidas (*stakeholders*) e engenheiros de requisitos. Uma das maiores preocupações na definição deste dialeto é evitar a saturação visual da notação, que consistiria na utilização excessiva de variáveis visuais, bem como manter a economia gráfica da linguagem.

A definição do vocabulário visual nas próximas subseções será realizada por meio da definição da razão do *design* (*design rationale*) de cada símbolo gráfico, conforme modelo de perfil de captura apresentado na Tabela 3, que é parte do *framework* de verificabilidade de PoN apresentado na seção 2.2.5.

3.4.1 Vocabulário Visual: Notação Técnica - Dialeto Visual 1

a) Requisito / Regra (*Requirement / Rule*)**:**

Título	Requisito / Regra (*Requirement / Rule*)
Princípio	Clareza Semiótica, Codificação Dual, Economia Gráfica, Ajuste Cognitivo.
Símbolo gráfico	
Propriedade visual	Forma, Brilho (espessura da linha)

Título	Requisito / Regra (*Requirement / Rule*)
Escolha de *design*	Representar requisitos com o uso de símbolos **similares** aos de símbolos de classes utilizados nas notações UML (OMG, UML, 2005) e SysML (OMG, SysML, 2017). Foi escolhido utilizar a borda do retângulo externo da figura com uma largura maior (2,25pt) que o usual da UML (1pt), de forma a proporcionar maior destaque do símbolo nos diagramas. O estereótipo UML <<*Requirement*>> foi incluído na parte superior para identificar que é um símbolo de *Requirement* (Requisito), a exemplo da extensão UML realizada na linguagem UML RD (MAGAGLEH e BARRETT, 2011). Sugere-se que visualmente o texto principal do *Requirement* na parte central do símbolo inclua: a) Identificador do requisito: "*REQUIREMENT ID*"; b) Nome do requisito "*Requirement name*"; c) Especificação do requisito: "*Specification to be met*".
Justificativa	Representação técnica, que não visa utilizar variáveis visuais elaboradas e prima pela economia gráfica e codificação dual.
Requisitos / Diretrizes	D.3.1 (clareza semiótica), D.3.5 (codificação dual), D.3.6 (economia gráfica), D.3.7 (ajuste cognitivo), D.4 (diagrama de requisitos funcionais/não funcionais) (seção 3.2).
Evidência	Moderada (avaliação de especialista) baseando-se em observação do autor com base em SLM sobre modelos gráficos em RE (*e.g.:* modelos gráficos SysML, UML e UML RD e RON – ver seção 2.3).

Tabela 27 – Notação Técnica - *Design Rationale* do Símbolo: Requisito
Fonte: o autor

b) Entidade / FBE (*entity* / FBE)**:**

Título	Entidade / FBE (*Entity* / FBE)
Princípio	Clareza Semiótica, Discriminalidade perceptiva, Codificação Dual, Economia Gráfica, Ajuste Cognitivo.
Símbolo gráfico	

Título	Entidade / FBE (*Entity* / FBE)
Propriedade visual	Forma, textura (da linha)
Escolha de *design*	Representar requisitos funcionais e não funcionais com o uso de símbolos **similares** aos de símbolos de classes utilizados nas notações UML (OMG, UML, 2005) e SysML (OMG, SysML, 2017). É importante notar que a linha tracejada do símbolo não existe nos símbolos da UML. Foi escolhido utilizar a borda do retângulo externo da figura com uma largura maior (2,25pt) que o usual da UML (1pt) e com o uso de uma linha tracejada, de forma que houvesse uma diferença clara em relação ao símbolo de *Requirement*. O estereótipo UML <<*Entity*>> foi incluído na parte superior para identificar que é um símbolo de *Entity* (Entidade). Sugere-se que visualmente o texto principal do *Entity* na parte central do símbolo inclua: a) Identificador da entidade: "*ENTITY ID*"; b) Nome da entidade: "*Entity Name*"; c) Descrição da entidade: "*Entity Description*".
Justificativa	Representação técnica, que não visa utilizar variáveis visuais elaboradas. Ainda assim, o símbolo de entidade se diferencia do de requisito pela linha tracejada (segundo o princípio de discriminalidade perceptiva), além de utilizar a codificação dual.
Requisitos / Diretrizes	D.3.1 (clareza semiótica), D.3.2 (discriminalidade perceptiva), D.3.5 (codificação dual), D.3.6 (economia gráfica), D.3.7 (ajuste cognitivo), D.4 (diagrama de requisitos) (seção 3.2)
Evidência	Moderada (avaliação de especialista) baseando-se em observação do autor com base em SLM sobre modelos gráficos em RE (*e.g.:* modelos gráficos SysML, UML e UML RD e RON – ver seção 2.3).

Tabela 28 – Notação Técnica - *Design Rationale* do Símbolo: Entidade
Fonte: o autor

c) Interconexões entre interdependências (Interdependency *interconnections*)**:**

Título	Interconexões entre interdependências (*Interdependency interconnections*)
Princípio	Clareza Semiótica, Codificação Dual, Economia Gráfica, Ajuste Cognitivo.

Título	Interconexões entre interdependências (*Interdependency interconnections*)
Símbolos gráficos	
Propriedade visual	Forma
Escolha de *design*	Representar a interconexão lógica entre interdependências por meio de símbolos padronizados (círculo), com uma codificação em texto que seja clara o suficiente para identificar o significado do símbolo lógico. Os textos a serem escolhidos podem representar visualmente as operações lógicas de conjunção (“AND”), disjunção (“OR”) ou disjunção exclusiva (“XOR”).
Justificativa	Representação técnica, em que as únicas variáveis são a forma (círculo) e o texto que indica claramente a operação lógica realizada em linguagem natural (*AND, OR, XOR*). Evitou-se nessa definição qualquer utilização de símbolos lógico-matemáticos (*e.g.:* ∧ ∨ ⊕) ou outros que requisitassem conhecimentos adicionais além da linguagem natural.
Requisitos / Diretrizes	D.3.1 (clareza semiótica), D.3.5 (codificação dual), D.3.6 (economia gráfica), D.3.7 (ajuste cognitivo), D.4 (diagrama de requisitos) (seção 3.2)
Evidência	Moderada (avaliação de especialista) baseando-se em observação do autor com base em SLM sobre modelos gráficos em RE (*e.g.:* modelos gráficos URN e UML UC – ver seção 2.3).

Tabela 29 – Notação Técnica - *Design Rationale* do Símbolo: Interconexão
Fonte: o autor

3.4.2 Vocabulário Visual: Notação de Negócios - Dialeto Visual 2

a) Requisito / Regra (*Requirement* / *rule*)**:**

Título	Requisito / Regra (*Requirement* / *Rule*)
Princípio	Clareza Semiótica, Discriminabilidade Perceptiva, Transparência Semântica, Expressividade Visual, Codificação Dual, Economia Gráfica, Ajuste Cognitivo

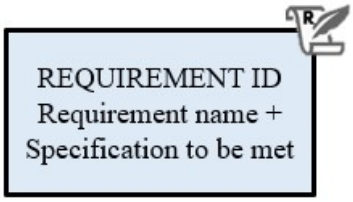
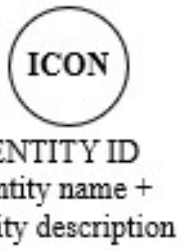
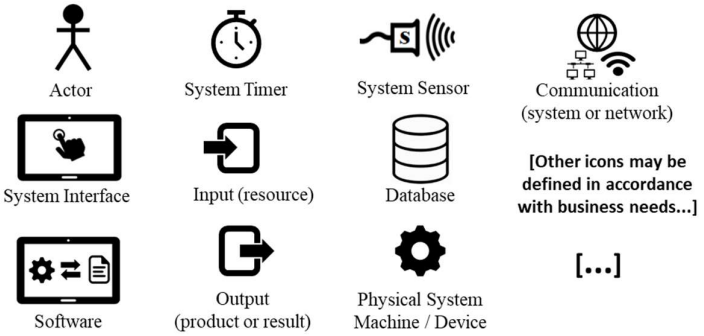
Título	Requisito / Regra (*Requirement* / *Rule*)
Símbolo gráfico	
Propriedade visual	Forma, Cor
Escolha de *design*	Representar requisitos por meio das variáveis visuais forma e cor, para incrementar a expressividade visual e a discriminalidade perceptiva. O ícone padronizado "  ", contendo um pergaminho, uma caneta em formato de "pena", e a letra "**R**", deve ser incluído no canto superior direito do símbolo. Este ícone representa o conceito de **R**equisito (**R**egra) ou ***R****equirement* (***R****ule*), ao simbolizar a escrita destes em um pergaminho. O ícone citado acima identifica que o símbolo se refere a um requisito, tornando desnecessário o uso do estereótipo UML <<*Requirement*>> usado na notação técnica. O texto central do símbolo deve incluir: a) Identificador do requisito: "*REQUIREMENT ID*"; b) Nome do requisito "*Requirement name*"; c) Especificação do requisito: "*Specification to be met*".
Justificativa	Representação de negócios, que prima pela expressividade visual e discriminalidade perceptiva, com o objetivo de deixar claro ao usuário que este elemento se refere a um requisito.
Requisitos / Diretrizes	D.3.1 (clareza semiótica), D.3.2 (discriminalidade perceptiva), D.3.3 (transparência semântica), D.3.4 (expressividade visual), D.3.5 (codificação dual), D.3.6 (economia gráfica), D.3.7 (ajuste cognitivo), D.4 (diagrama de requisitos).
Evidência	Moderada (avaliação de especialista) baseando-se em observação do autor com base em literatura sobre PoN (MOODY et al., 2009) (MOODY et. al., 2010) (GENON et al., 2010) (GENON et al., 2011) (GENON e al., 2012) (CAIRE et al., 2013).

Tabela 30 – Notação de Negócios - *Design Rationale* do Símbolo: Requisito
Fonte: o autor

b) Entidade / FBE (*Entity* / FBE)**:**

Título	Entidade / FBE (*Entity* / FBE)
Princípio	Clareza Semiótica, Discriminabilidade Perceptiva, Transparência Semântica, Expressividade Visual, Codificação Dual, Ajuste Cognitivo.
Símbolo(s) gráfico(s)	 ICON pode ser substituído por um dos seguintes símbolos: 
Propriedade visual	Forma
Escolha de *design*	Representar entidades por meio ícones que incrementem a discriminalidade perceptiva. O símbolo “ICON” pode ser substituído por qualquer um dos **ícones** propostos acima, ou por outros criados especificamente para representar necessidades específicas de usuários de um determinado ambiente de negócios. A notação fornece um conjunto inicial de ícones para algumas entidades comuns, tais como: *Actor, System Timer, System Sensor, Communication, System Interface, Input* (*resource*)*, Database, Software, Output* (*product or result*)*, Physical System* (*Machine or Device*)*.*
Justificativa	Representação de negócios, que prima pela discriminalidade perceptiva (uso de ícones); transparência semântica (obtenção imediata do significado); ajuste cognitivo (é possível criar ícones/símbolos conforme necessidades do negócio) e codificação dupla (texto / imagem).

Título	Entidade / FBE (*Entity* / FBE)
Requisitos / Diretrizes	D.3.1 (clareza semiótica), D.3.2 (discriminalidade perceptiva) D.3.3 (transparência semântica), D.3.4 (expressividade visual), D.3.5 (codificação dual), D.3.6 (economia gráfica), D.3.7 (ajuste cognitivo), D.4 (diagrama de requisitos).
Evidência	Moderada (avaliação de especialista) baseando-se em observação do autor com base em SLM sobre modelos gráficos (ver seção 2.3) e especificamente na notação de RE para aplicações industriais URML (BERENBACH et al., 2012).

Tabela 31 – Notação de Negócios - *Design Rationale* do Símbolo: Entidade
Fonte: o autor

c) Interconexões entre interdependências (Interdependency *interconnections*)**:**

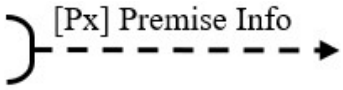
Título	Interconexões entre interdependências (*Interdependency interconnections*)
Princípio	Clareza Semiótica, Codificação Dual, Expressividade Visual, Economia Gráfica, Ajuste Cognitivo.
Símbolos gráficos	
Propriedade visual	Forma, Cor, Brilho (espessura da linha)
Escolha de *design*	Representar a interconexão lógica entre interdependências por meio de símbolos padronizados (círculo) com uma codificação em texto clara o suficiente para qualquer audiência identificar o significado do símbolo lógico. A adição da variável visual cor a estes símbolos gráficos permite distingui-los dos demais elementos da notação, enquanto a variável visual brilho (espessura da linha) permite distingui-los uns dos outros. Os textos a serem escolhidos podem representar visualmente as operações lógicas de conjunção (“AND”), disjunção (“OR”) ou disjunção exclusiva (“XOR”).

Título	Interconexões entre interdependências (*Interdependency interconnections*)
Justificativa	Representação de negócios, em que as variáveis são a forma (círculo), o texto que indica claramente a operação lógica realizada em linguagem natural (*AND, OR, XOR*) e a cor para distinguir estes símbolos dos demais símbolos da notação. Evitou-se nessa definição qualquer utilização de símbolos lógico-matemáticos (*e.g.:* ∧ ∨ ⊕) ou outros que requisitassem conhecimentos adicionais além da linguagem natural.
Requisitos / Diretrizes	D.3.1 (clareza semiótica), D.3.4 (expressividade visual), D.3.5 (codificação dual), D.3.6 (economia gráfica), D.3.7 (ajuste cognitivo), D.4 (diagrama de requisitos) (seção 3.2)
Evidência	Moderada (avaliação de especialista) baseando-se em observação do autor com base em SLM sobre modelos gráficos em RE (*e.g.:* modelos gráficos URN e UML UC – ver seção 2.3) e literatura sobre PoN (MOODY et al., 2010).

Tabela 32 – Notação de Negócios - *Design Rationale* do Símbolo: Interconexão
Fonte: o autor

3.4.3 Vocabulário Visual: Elementos Comuns da Notação

a) Premissa (*Premise*)**:**

Título	Premissa (*Premise*)
Princípio	Clareza Semiótica, Discriminabilidade Perceptiva, Codificação Dual, Economia Gráfica.
Símbolo gráfico	
Propriedade visual	Forma, textura, orientação, posição horizontal (x), posição vertical (y).

Título	Premissa (*Premise*)
Escolha de *design*	Representar premissas por meio de cinco variáveis visuais: • Forma da seta da seta de premissa; • Textura da linha (descontínua) para representação de premissas em oposição a inferências (contínua); • Orientação da seta, indicando onde a premissa se origina e sua direção de influência / propagação; • Posição horizontal (x) e Posição vertical (y) da premissa, que pode ser colocada em qualquer direção, o que não ocorre (*e.g.)* com outros símbolos tais como requisitos e entidades. O símbolo possui um **semicírculo** em sua parte inicial, em que sempre será feita uma conexão a um símbolo de atributo, conforme as regras de composição visual posteriormente definidas. Além disso, o símbolo inclui codificação textual conforme segue: a) Identificador entre colchetes "**[Px]"**, em que x pode assumir números inteiros positivos não nulos (*e.g.* P1, P2, ..., Pn); b) Descrição textual **"*Premise Info*"**, que pode conter uma expressão textual, condição lógica ou matemática representando a semântica da Premissa.
Justificativa	Representação aplicável à notação técnica e de negócios, que prima pela discriminalidade perceptiva (textura da linha diferenciada) e pela codificação dupla (texto / imagem).
Requisitos / Diretrizes	D.3.1 (clareza semiótica), D.3.2 (discriminalidade perceptiva), D.3.5 (codificação dual), D.3.6 (economia gráfica), D.4 (diagrama de requisitos) (seção 3.2)
Evidência	Moderada (avaliação de especialista) baseando-se em observação do autor com base em literatura sobre PoN (MOODY et al., 2009) (MOODY et. al., 2010) (GENON et al., 2010) (GENON et al., 2011) (GENON e al., 2012) (CAIRE et al., 2013).

Tabela 33 – Notação Comum - *Design Rationale* do Símbolo: Premissa
Fonte: o autor

b) Inferência (*Inference*)**:**

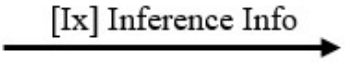
Título	Inferência (*Inference*)
Princípio	Clareza Semiótica, Discriminabilidade Perceptiva, Codificação Dual, Economia Gráfica.
Símbolo gráfico	
Propriedade visual	Forma, textura, orientação, posição horizontal (x), posição vertical (y).
Escolha de *design*	Representar inferências por meio de cinco variáveis visuais: a) Forma da seta de inferência; b) Textura da linha (contínua) para representação de inferências em oposição a premissas (descontínua); c) Orientação da seta, indicando onde a inferência se origina e sua direção de sua influência / propagação; d) Posição horizontal (x) e Posição vertical (y) da inferência, que pode ser colocada em qualquer direção, o que não ocorre (*e.g.)* com outros símbolos tais como requisitos e entidades. Além disso, o símbolo inclui codificação textual conforme segue: a) Identificador entre colchetes "**[Ix]"**, em que x pode assumir números inteiros positivos não nulos (*e.g.* I1, I2, ..., In); b) Descrição textual **"*Inference Info"***, que pode conter uma expressão textual, condição lógica ou matemática representando a semântica da Inferência.
Justificativa	Representação aplicável à notação técnica e de negócios, que prima pela discriminalidade perceptiva (textura da linha diferenciada) e pela codificação dupla (texto / imagem).
Requisitos / Diretrizes	D.3.1 (clareza semiótica), D.3.2 (discriminalidade perceptiva), D.3.5 (codificação dual), D.3.6 (economia gráfica), D.4 (diagrama de requisitos) (seção 3.2)
Evidência	Moderada (avaliação de especialista) baseando-se em observação do autor com base em literatura sobre PoN (MOODY et al., 2009) (MOODY et. al., 2010) (GENON et al., 2010) (GENON et al., 2011) (GENON e al., 2012) (CAIRE et al., 2013).

Tabela 34 – Notação Comum - *Design Rationale* do Símbolo: Inferência
Fonte: o autor

c) Atributo (*Attribute*)**:**

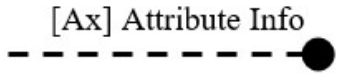
Título	Atributo (*Attribute*)
Princípio	Clareza Semiótica, Discriminabilidade Perceptiva, Codificação Dual, Economia Gráfica.
Símbolo gráfico	
Propriedade visual	Forma, orientação, posição horizontal (x), posição vertical (y).
Escolha de *design*	Representar atributos por meio das variáveis visuais: • Forma do símbolo de atributo; • Orientação do símbolo, indicando onde qual a direção de sua influência / propagação; • Posição horizontal (x) e Posição vertical (y) do atributo, que pode ser colocada em qualquer direção, o que não ocorre com outros símbolos tais como requisitos e entidades, por exemplo. O símbolo de atributo escolhido foi inspirado no símbolo de componente (interface) da notação UML. Este símbolo pode ser usado para que uma interconexão seja feita ao atributo disponibilizado pela entidade, de forma similar, porém não idêntica, a do símbolo equivalente da UML. Além disso, o símbolo inclui codificação textual conforme segue: a) Identificador entre colchetes "**[Ax]"**, em que x pode assumir números inteiros positivos não nulos (*e.g.* I1, I2, ..., In); b) Descrição textual ***"Atribute Info"***, que pode conter uma expressão textual, condição lógica ou matemática representando a semântica do Atributo.
Justificativa	Representação aplicável à notação técnica e de negócios, que prima pela discriminalidade perceptiva (forma e orientação do símbolo) e pela codificação dupla (texto / imagem).
Requisitos / Diretrizes	D.3.1 (clareza semiótica), D.3.2 (discriminalidade perceptiva), D.3.5 (codificação dual), D.3.6 (economia gráfica), D.4 (diagrama de requisitos) (seção 3.2)
Evidência	Moderada (avaliação de especialista) baseando-se em observação do autor com base em literatura de PoN (MOODY et al., 2009) (MOODY et. al., 2010) (GENON et al., 2010) (GENON et al., 2011) (GENON e al., 2012) (CAIRE et al., 2013).

Tabela 35 – Notação Comum - *Design Rationale* do Símbolo: Atributo
Fonte: o autor

d) Elemento de análise (*Analysis element*)**:**

Título	Elemento de Análise (*Analysis element*) Tipo de análise (*Analysis type*): Conflito potencial
Princípio	Clareza Semiótica, Discriminabilidade Perceptiva, Transparência Semântica, Economia Gráfica.
Símbolo gráfico	
Propriedade visual	Forma
Escolha de *design*	Representar conflitos em interdependências de requisitos por meio do símbolo de um raio (representando ruptura, fratura ou divisão brusca) associado a um símbolo de exclamação para representar o fato de ser “potencial” (podendo ser resolvido após um processo de análise).
Justificativa	Representação aplicável à notação técnica e de negócios, que prima pela discriminalidade perceptiva (forma do símbolo) e transparência semântica (sugestão do significado).
Requisitos / Diretrizes	D.3.1 (clareza semiótica), D.3.2 (discriminalidade perceptiva), D.3.6 (economia gráfica), D.4 (diagrama de requisitos), D.5 (identificação visual de conflitos) (seção 3.2)
Evidência	Moderada (avaliação de especialista) baseando-se em observação do autor com base em literatura de PoN (MOODY et al., 2009) (MOODY et. al., 2010) (GENON et al., 2010) (GENON et al., 2011) (GENON e al., 2012) (CAIRE et al., 2013). O símbolo foi inspirado por símbolo similar da notação KAOS (DARDENNE et al., 1993).

Tabela 36 – Notação Comum - *Design Rationale* do Símbolo: Conflito Potencial
Fonte: o autor

3.4.4 Gramática Visual (regras de composição)

As regras de composição determinam como os símbolos gráficos podem se compor para formar expressões visuais válidas (MOODY et al., 2010). Nesse sentido, elas representam de que forma os símbolos podem ser integrados de forma **válida** em um diagrama. Nesta subseção, as possíveis composições entre os símbolos gráficos são apresentadas, sendo que algumas composições não podem subsistir isoladamente em um diagrama (*e.g.* pré-condição, pós-

condição). Estas podem ser entendidas como composições intermediárias, cuja definição é necessária para posteriormente compor regras mais complexas.

Para cada regra, será apresentado o **objetivo**, a **notação** visual (figura), e como realizar a integração dos símbolos (**composição**):

a) **Regra de composição: pré-condição** (*Precondition*)

Objetivo: compor uma **pré-condição**, por meio do agrupamento dos símbolos de **atributo, d**e **premissa** e de **conflito** (este último de uso opcional, caso ocorra o conflito). Esta composição não poderá existir isoladamente, devendo posteriormente ser associada a símbolos gráficos de entidade, requisito, ou interconexões, conforme será mostrado nas regras de composição mais complexas (definidas posteriormente nesta subseção).

Notação: A notação é comum para ambos os dialetos (técnico e de negócios), conforme a Figura 43:

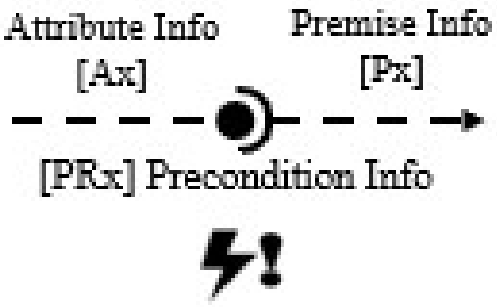


Figura 43 – Regra de composição: pré-condição
Fonte: o autor

Composição:

- Inicialmente, coloca-se o símbolo de a**tributo** (A*ttribute*), que tem um identificador "[Ax]" e uma descrição ou expressão lógico matemática do atributo: "*Attribute Info"*;
- A seguir, conecta-se a **premissa** (*Premise*) ao símbolo do atributo, que tem um identificador "[Px]" e uma descrição da premissa: "*Premise Info"*. Tanto "[Px]" quanto "*Premise Info*" são de uso opcional;
- Um identificador de pré-condição "[PRx]" e a condição "*Precondition Info*" podem ser utilizados opcionalmente;

- Um **conflito potencial** relacionado a uma pré-condição pode **opcionalmente** ser representado por meio do símbolo correspondente, na parte inferior da composição, informando que a interdependência está sob análise e que potencialmente contém algum conflito.

b) Regra de composição: de entidade para requisito

Objetivo: interligar uma **entidade** (*Entity*) a um **requisito** (*Requirement*) por meio de uma composição de **pré-condição** (ver regra anterior). A interpretação desta composição gráfica, segundo o definido no metamodelo (seção 3.3.1), é que que uma *Entity* pode afetar um *Requirement* por meio de uma *Precondition*. A *Precondition* pode conter uma *Premise*, que pode acessar um *Attribute* da Entity.

Notação: Na Figura 44, os dois dialetos visuais são mostrados: notação técnica (parte de acima) e notação de negócios (parte de baixo).

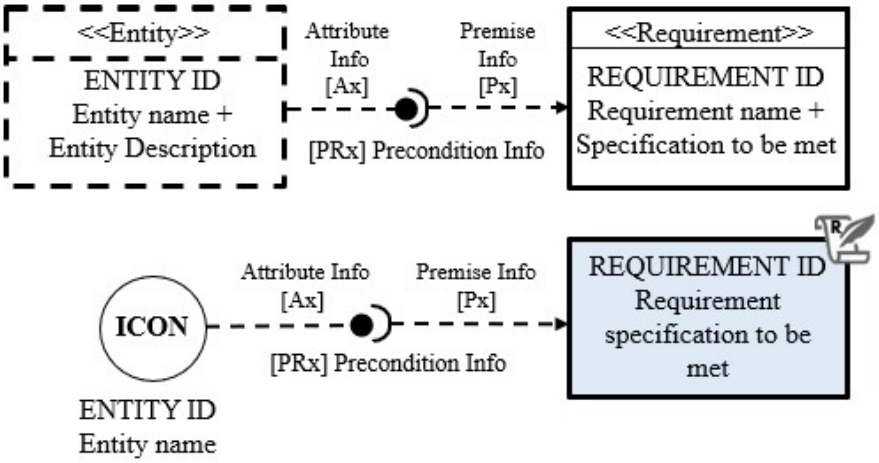


Figura 44 – Regra de composição: de entidade para requisito
Fonte: o autor

Composição:

- Inicialmente, a composição da sentença visual se dá pela colocação do símbolo de **entidade** (*Entity*), ao qual o atributo de pré-condição estará conectado;
- Na sequência, é colocada uma **composição de pré-condição**, contendo o atributo, a premissa e os identificadores correspondentes;
- A seguir, a premissa é conectada ao **requisito** (*Requirement*), simbolizando a influência da pré-condição / premissa sobre o mesmo.

Caso haja interconexões entre premissas (tais como uniões ou divisões, descritas nas próximas subseções), as mesmas deverão ocorrer entre o símbolo de atributo e o símbolo de requisito.

c) **Regra de composição: pós-condição** (*Postcondition*)

Objetivo: compor uma **pós-condição**, por meio da utilização do símbolo de **inferência** (*Inference*). Esta composição não poderá existir isoladamente, devendo estar associada a símbolos gráficos de entidade, requisito, ou interconexões, conforme será mostrado nas regras de composição mais complexas definidas posteriormente nesta subseção.

Notação: A notação é comum para ambos os dialetos (técnico e de negócios), e é ilustrada na Figura 45:

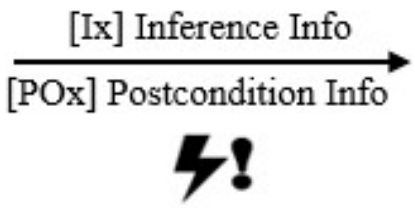


Figura 45 – Regra de composição: pós-condição
Fonte: o autor

Composição:

- Coloca-se o símbolo de **inferência** (*Inference*), que tem um identificador "[Ix]" (uso opcional) e uma descrição da inferência: "*Inference Info"* (uso obrigatório);
- Opcionalmente, pode-se colocar o identificador de pós-condição "[POx]" e uma informação sobre a pós-condição "*Postcondition Info*";
- Um **conflito potencial** pode **opcionalmente** ser representado por meio do símbolo correspondente, na parte inferior da composição.

d) **Regra de composição: de requisito para entidade**

Objetivo: interligar um **requisito** (*Requirement*) a uma **entidade** (*Entity*) por meio de uma composição de **pós-condição**. A interpretação desta composição gráfica, segundo o definido no metamodelo (seção 3.3.1), é que um *Requirement* pode afetar uma *Entity* por meio de uma *Postcondition,* que pode conter uma *Inference*.

Notação: Dois dialetos visuais são mostrados na Figura 46: notação técnica (parte de cima) e notação de negócios (parte de baixo).

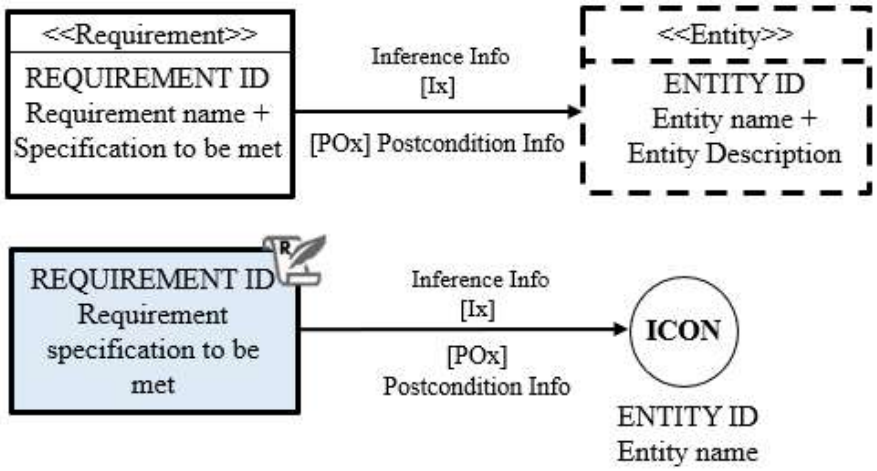


Figura 46 – Regra de composição: de requisito para entidade
Fonte: o autor

Composição:

- Inicialmente, a composição da sentença visual se dá pela colocação do símbolo de **requisito** (*Requirement*), ao qual o símbolo de inferência da pós-condição estará conectado;
- Na sequência, é colocada uma **composição** de **pós-condição**, contendo a inferência e os identificadores correspondentes;
- Por fim, a inferência é conectada à **entidade** (Entity), simbolizando a influência da pós-condição / inferência sobre a mesma. Caso haja interconexões entre inferências (tais como uniões ou divisões, descritas nas próximas subseções), as mesmas deverão ocorrer entre o símbolo de requisito e o símbolo de entidade.

e) Regra de composição: interconexão para pré-condição

Objetivo: permitir interconectar uma ou mais **pré-condições**, que são compostas por premissas (*Premises*), de forma a realizar graficamente operações de divisão (*division*) ou de junção (*join*). As *Premises* podem conter sentenças de texto ou condições lógico-matemáticas, conforme definido na seção 3.3.2. As *Premises* de entrada (*inputs*), segundo o definido no metamodelo (seção 3.3.1), podem ser modificadas por uma *Precondition Interconnection*, que altera o conteúdo da(s) *Premise(s)* transformando-as em *Premises* de saída (*outputs*). Essas interconexões podem envolver operadores lógicos (AND, OR ou XOR) ou simplesmente serem representadas por uma divisão (ou junção) das linhas no diagrama.

Notação: A Figura 47 apresenta a regra de composição relativa a interconexões para pré-condição, para o dialeto visual da notação de **negócios**. Caso o dialeto técnico seja usado, os símbolos de interconexão (AND, OR e XOR) serão modelados em preto e branco (sem a cor verde).

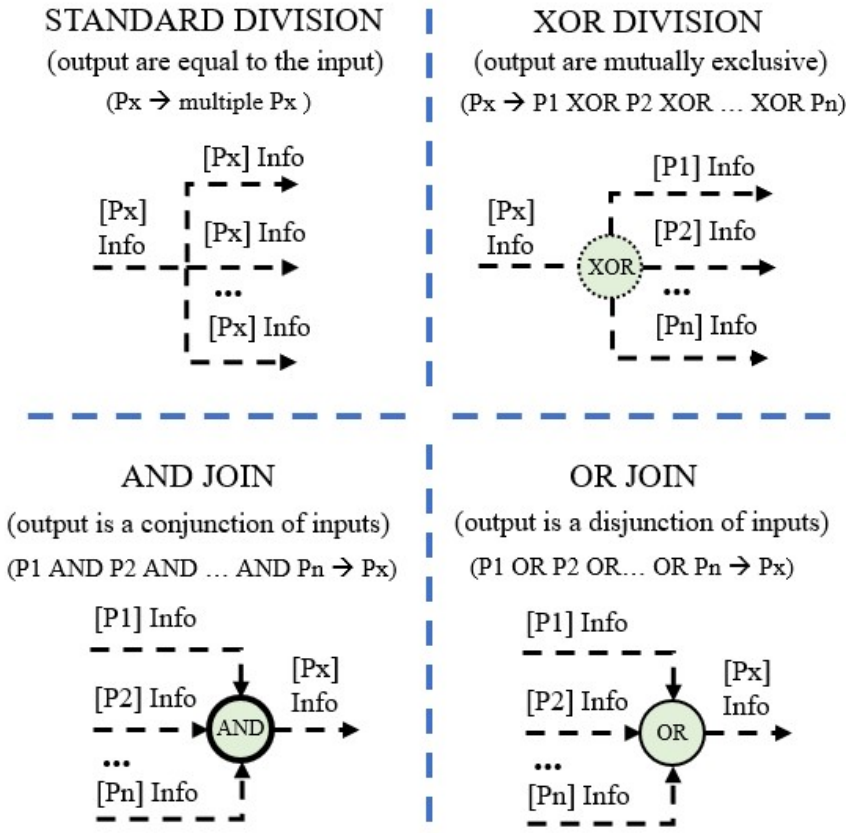


Figura 47 – Regra de composição: interconexões de pré-condição
Fonte: o autor

Composição:

- **STANDARD_DIVISION:** dada uma **premissa** (*Premise*) de entrada (*input*) com um identificador [Px], a mesma pode ser subdividida em 2 a n (em que $n \in \mathbb{N}\ e\ n > 2$) premissas **iguais**, por meio da divisão gráfica da linha da premissa de entrada. Cada premissa de saída (*output*) poderá conter opcionalmente o mesmo identificador [Px] de entrada e uma descrição "*Info*";
- **XOR_DIVISION:** dada uma premissa de entrada (*input*) com um identificador [Px], a mesma pode ser subdividida em 2 a n (em que $n \in \mathbb{N}\ e\ n > 2$) premissas **mutuamente exclusivas**, por meio do uso

de um símbolo de interconexão lógica XOR. Cada **premissa** (*Premise*) de saída (*output*) poderá **opcionalmente** ter um identificador ([P1] a [Pn]), além de **obrigatoriamente** possuir uma descrição "*Info*", de forma a mostrar o conteúdo das premissas de saída;

- **AND_JOIN:** dado um conjunto de **premissas** (*Premisses*) de entrada (inputs) com identificadores entre [P1] e [Pn], estas podem ser unidas por uma operação lógica de **conjunção (AND)** em uma mesma premissa de saída (*output*). A premissa de saída poderá **opcionalmente** ter um identificador [Px], além de **obrigatoriamente** possuir uma descrição "*Info*", de forma a mostrar o conteúdo da premissa de saída;
- **OR_JOIN:** dado um conjunto de **premissas** (*Premisses*) de entrada (inputs) com identificadores entre [P1] e [Pn], estas podem ser unidas por uma operação lógica de **disjunção (OR)** em uma mesma premissa de saída (*output*). A premissa de saída poderá **opcionalmente** ter um identificador [Px], além de **obrigatoriamente** possuir uma descrição "*Info*", de forma a mostrar o conteúdo da premissa de saída.

f) Regra de composição: interconexão para pós-condição

Objetivo: permitir interconectar uma ou mais **pós-condições**, que são compostas por inferências (*Inferences*), de forma a realizar graficamente operações de divisão (*division*) ou de junção (*join*). As *Inferences* podem conter sentenças de texto ou condições lógico-matemáticas, conforme definido na seção 3.3.2. As *Inferences* de entrada (*inputs*), segundo o definido no metamodelo (seção 3.3.1), podem ser modificadas por uma *Postcondition Interconnection*, que altera o conteúdo da(s) *Inferences(s)* transformando-as em *Inferences* de saída (*outputs*). Essas interconexões podem envolver operadores lógicos (AND, OR ou XOR) ou simplesmente serem representadas por uma divisão (ou junção) das linhas no diagrama.

Notação: A Figura 48 apresenta a regra de composição relativa a interconexões para pós-condição, para o dialeto visual da notação de negócios. Caso o dialeto técnico seja usado, os símbolos de

interconexão (AND, OR e XOR) serão modelados em preto e branco (sem a cor verde).

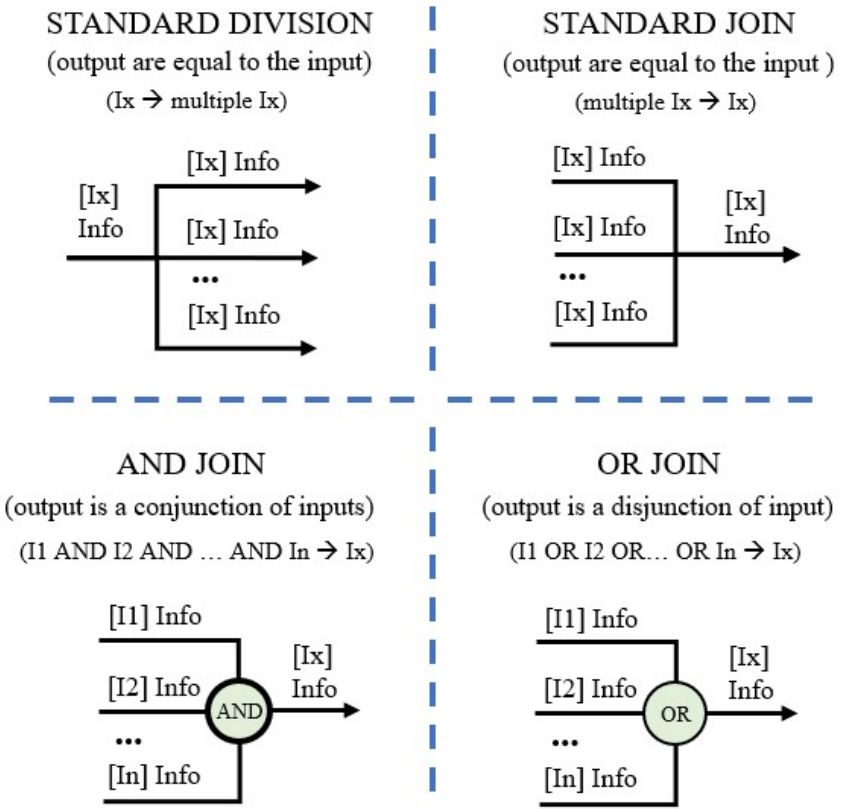


Figura 48 – Regra de composição: interconexões de pós-condição
Fonte: o autor

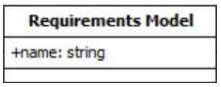
Composição:

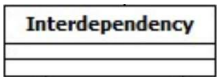
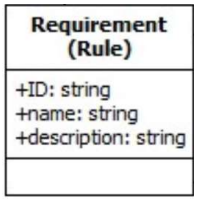
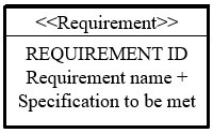
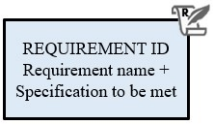
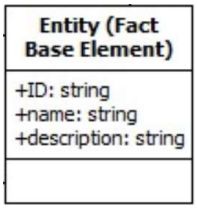
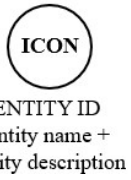
- **STANDARD_DIVISION:** dada uma **inferência** (*Inference*) de entrada (*input*) com um identificador [Ix], a mesma pode ser subdividida em 2 a n (em que $n \in \mathbb{N}\, e\, n > 2$) inferências **iguais**, por meio da divisão gráfica da linha da inferência de entrada. Cada inferência de saída (*output*) poderá conter opcionalmente o mesmo identificador [Ix] de entrada e uma descrição "*Info*";
- **STANDARD_JOIN:** dado um conjunto de **inferências** (*Inferences*) de entrada (inputs) **iguais,** com identificador genérico [Ix], estas podem ser unidas por meio da junção gráfica das linhas das inferências de entrada em uma mesma inferência de saída (*output*). A inferência de saída deverá ter o mesmo identificador [Ix] e a mesma descrição "*Info*";

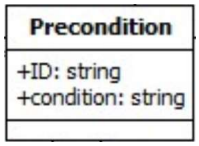
- **AND_JOIN:** dado um conjunto de **inferências** (*Inferences*) de entrada (inputs) com identificadores entre [I1] e [In], estas podem ser unidas por uma operação lógica de **conjunção (AND)** em uma mesma inferência de saída (*output*). A inferência de saída poderá **opcionalmente** ter um identificador [Ix], além de **obrigatoriamente** possuir uma descrição "*Info*", de forma a mostrar o conteúdo da inferência de saída;
- **OR _JOIN:** dado um conjunto de **inferências** (*Inferences*) de entrada (inputs) com identificadores entre [I1] e [In], estas podem ser unidas por uma operação lógica de **disjunção (OR)** em uma mesma inferência de saída (*output*). A inferência de saída poderá **opcionalmente** ter um identificador [Ix], além de **obrigatoriamente** possuir uma descrição "*Info*", de forma a mostrar o conteúdo da inferência de saída;

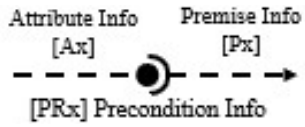
3.5 MAPEAMENTO SEMÂNTICO

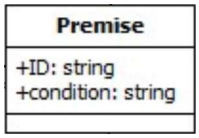
Depois de definir a sintaxe abstrata e a sintaxe concreta separadamente, um mapeamento semântico entre as sintaxes deve ser estabelecido (HE et al., 2007). Esse mapeamento consiste em realizar a correspondência entre as construções semânticas do metamodelo e os símbolos gráficos da sintaxe visual. A Tabela 37 mostra o mapeamento semântico para a linguagem de modelagem gráfica RIMON, para os dois dialetos visuais (técnico e de negócios).

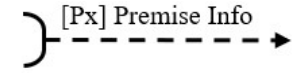
SINTAXE ABSTRATA (construções semânticas)	SINTAXE CONCRETA (símbolos gráficos)	
Elementos do metamodelo	Dialeto Visual 1: Notação Técnica	Dialeto Visual 2: Notação de Negócios
Requirements Model  Modelo de Requisitos É uma composição de entidades, requisitos e interdependências.	Diagrama modelado como um todo, sem símbolo específico. +name → nome do diagrama	

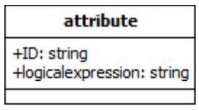
SINTAXE ABSTRATA (construções semânticas)	SINTAXE CONCRETA (símbolos gráficos)	
Elementos do metamodelo	Dialeto Visual 1: Notação Técnica	Dialeto Visual 2: Notação de Negócios
Interdependency  interdependência É uma abstração de pré-condições ou de pós-condições	A interdependência, por se tratar de uma abstração de uma pré-condição ou de uma pós condição, não possui um símbolo gráfico definido. As especializações da interdependência, que são a pré-condição e a pós-condição – têm seus símbolos definidos logo abaixo nesta tabela.	
Requirement / Rule  requisito / regra	 +ID → *REQUIREMENT ID* (identificador) +name → *Requirement name* (nome do requisito - opcional) +description → Specification to be met (especificação a ser satisfeita)	 +ID → *REQUIREMENT ID* (identificador) +name → *Requirement name* (nome do requisito - opcional) +description → Specification to be met (especificação a ser satisfeita)
Entity / Fact Base Element  entidade / elemento da base de fatos	 +ID → ENTITY *ID* +name → *Entity name* (nome da entidade) +description → Entity description (descrição da entidade – opcional)	 +ID → ENTITY *ID* +name → *Entity name* (nome da entidade) +description → Entity description (descrição da entidade – opcional)

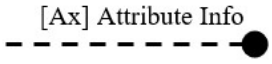
<table>
<tr><th>SINTAXE ABSTRATA
(construções semânticas)</th><th colspan="2">SINTAXE CONCRETA
(símbolos gráficos)</th></tr>
<tr><th>Elementos do metamodelo</th><th>Dialeto Visual 1:
Notação Técnica</th><th>Dialeto Visual 2:
Notação de Negócios</th></tr>
<tr><td>Precondition
 

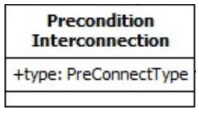
pré-condição
É uma composição de premissas que acessam atributos.</td><td colspan="2"> 

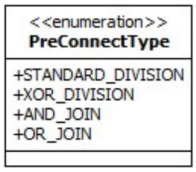
A pré-condição consiste na união gráfica de símbolos de atributo e premissa, conforme definido em regra de composição da subseção 3.4.4.
+ID → [PRx] (identificador da pré-condição - opcional)
+condition → Precondition Info (informação ou descrição da pré-condição - opcional)
[Ax] Attribute info → Attribute Info (informação em texto ou expressão lógica sobre o atributo)
[Px] Premise info → Premise Info (informação ou descrição da premissa - opcional)</td></tr>
<tr><td>Premise
 

premissa</td><td colspan="2"> 

+ID → [Px] (identificador da premissa)
+condition → Premise Info (informação ou descrição da premissa - opcional)</td></tr>
<tr><td>Attribute
 

atributo</td><td colspan="2"> 

+ID → [Ax] (identificador do atributo)
+logicalexpression → Attribute Info (informação em texto ou expressão lógica sobre o atributo)</td></tr>
<tr><td>Precondition interconnection
 

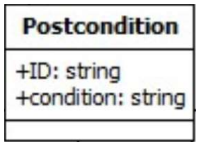
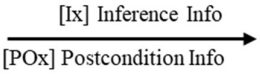
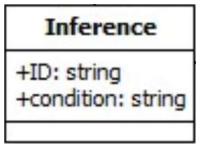
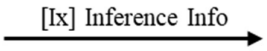
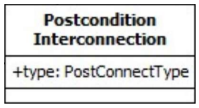
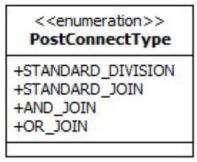
interconexão de pré-condição</td><td>+type: PreConnectType →
STANDARD_DIVISION: Representada pela divisão de linhas de conexão, conforme a subseção 3.4.4
+XOR_DIVISION:
 

+AND_JOIN:
 

+OR_JOIN:
 
</td><td>+type: PreConnectType →
STANDARD_DIVISION: Representada pela divisão de linhas de conexão, conforme a subseção 3.4.4
+XOR_DIVISION:
 

+AND_JOIN:
 

+OR_JOIN:
 
</td></tr>
</table>

SINTAXE ABSTRATA (construções semânticas)	SINTAXE CONCRETA (símbolos gráficos)	
Elementos do metamodelo	Dialeto Visual 1: Notação Técnica	Dialeto Visual 2: Notação de Negócios
Postcondition  A pós-condição é uma composição de inferências	 A pós-condição é composta por inferências, conforme definido em regra de composição da subseção 3.4.4. +ID → [POx] (identificador da pós-condição - opcional) +condition → *Postcondition Info (*informação ou descrição da pós-condição - opcional) [Ix] inference info → (identificador da inferência)	
Inference  inferência	 +ID → [Ix] (identificador da inferência) +condition → *Inference Info (*informação ou descrição da inferência)	
Postcondition interconnection   interconexão de pós-condição	+type: PostConnectType → STANDARD_DIVISION: Representada pela divisão de linhas de conexão, conforme a subseção 3.4.4 STANDARD_JOIN: Representada pela união de linhas de conexão, conforme a subseção 3.4.4 AND_JOIN:  OR_JOIN:  	+type: PostConnectType → STANDARD_DIVISION: Representada pela divisão de linhas de conexão, conforme a subseção 3.4.4 STANDARD_JOIN: Representada pela união de linhas de conexão, conforme a subseção 3.4.4 AND_JOIN:  OR_JOIN: 

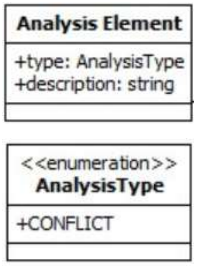
SINTAXE ABSTRATA (construções semânticas)	SINTAXE CONCRETA (símbolos gráficos)	
Elementos do metamodelo	Dialeto Visual 1: Notação Técnica	Dialeto Visual 2: Notação de Negócios
Analysis Element  elemento de Análise	+type: AnalysisType → CONFLICT 	

Tabela 37 – Mapeamento semântico para a linguagem RIMON
Fonte: o autor

3.6 NOTAÇÃO RIMON

A Figura 49 apresenta a notação RIMON de forma unificada, em uma única página, de forma a facilitar sua utilização por engenheiros de requisitos e usuários de forma geral. Ela está estruturada da seguinte forma:

a) Símbolos de Entidades e Requisitos da Notação técnica (*Technical Notation*) são mostrados na parte superior esquerda;
b) Símbolos de Entidades e Requisitos da Notação de Negócios *(Business Notation*) são mostrados na parte superior central;
c) Símbolos comuns de interdependências tais como premissas, inferências, atributos e conflitos potenciais são mostrados na parte superior direita;
d) Símbolos de elementos de interconexão, para ambos os dialetos, são mostrados também na parte superior direita;
e) Regras de composição de interdependências são mostradas na faixa central da figura;
f) Regras de composição para interconexões de interdependências são mostradas na parte inferior da figura.

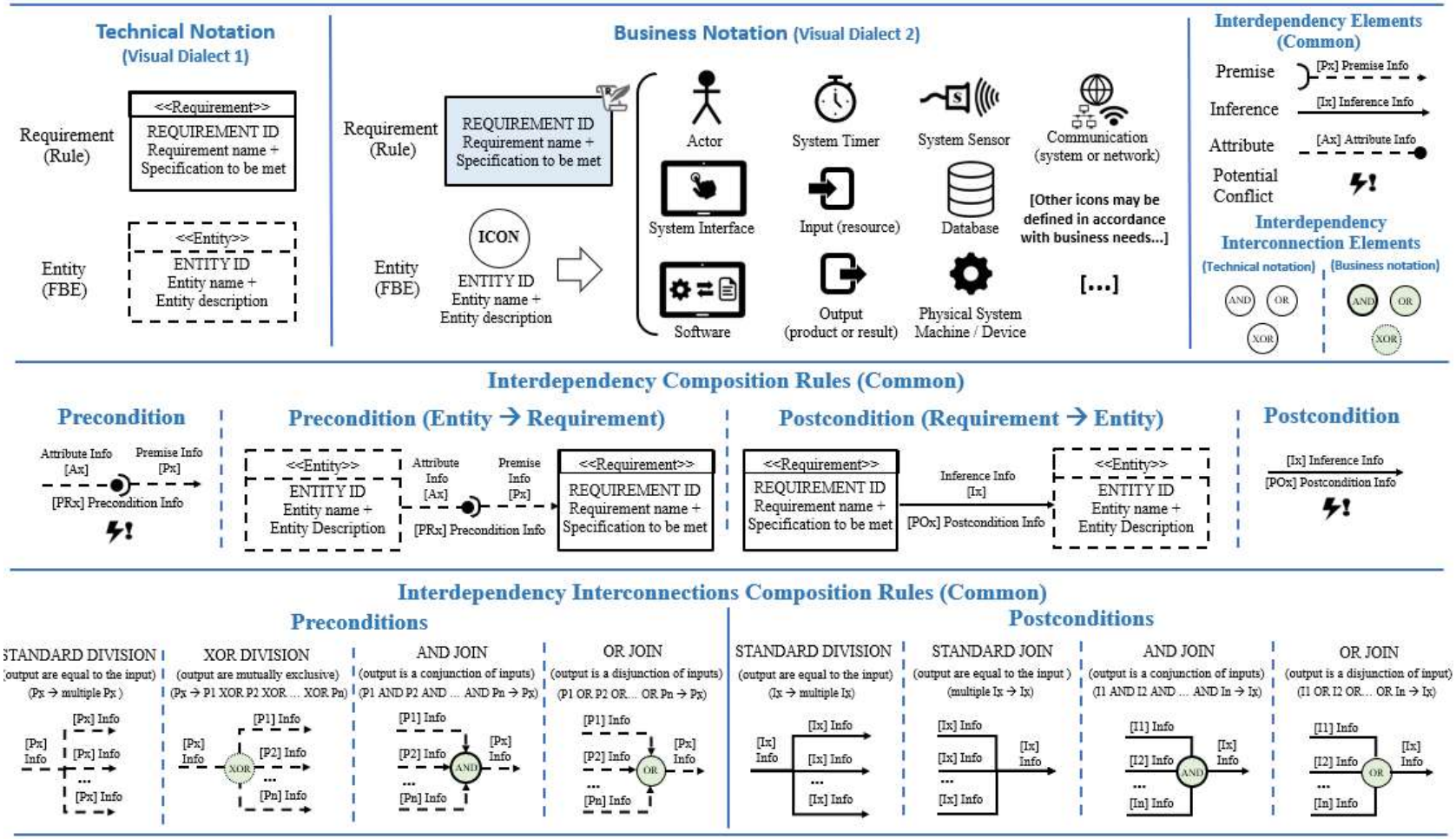


Figura 49 – Notação RIMON
Fonte: o autor

3.7 CONCLUSÃO DO CAPÍTULO

Este capítulo apresentou um método para desenvolvimento de novas linguagens de modelagem gráfica em RE fundamentadas na teoria de física das notações. Este método foi então aplicado para desenvolver a linguagem de modelagem gráfica RIMON, sendo que as seguintes etapas foram realizadas:

a) Elaboração de diretrizes para desenvolver a linguagem RIMON;
b) Definição da sintaxe abstrata: definição do metamodelo e construções semânticas;
c) Definição da sintaxe concreta: definição do vocabulário visual (símbolos) e da gramática visual (regras de composição);
d) Mapeamento semântico: associação entre construções semânticas do metamodelo (sintaxe abstrata) aos símbolos visuais (sintaxe concreta);
e) Notação RIMON: apresentação da notação em formato unificado.

A conclusão é que o método para desenvolvimento de linguagens de modelagem gráfica, proposto na seção 3.1 (Figura 42), permitiu descrever de forma adequada a linguagem de modelagem gráfica, tanto em termos semânticos quanto sintáticos, o que corresponde a maior parte da etapa de **desenvolvimento** da citada figura.

Em continuação à aplicação do método da seção 3.1, o capítulo 4 apresentará o método de modelagem RIMON, o capítulo 5 apresentará as experimentações com a linguagem de modelagem proposta no presente capítulo, enquanto que o capítulo 6 apresentará as verificações e análises realizadas.

4 MÉTODO DE MODELAGEM RIMON

Esta seção apresenta o método de modelagem RIMON por meio de um processo que, associado a uma notação anteriormente apresentada (Figura 49), orienta o especialista na condução da modelagem de requisitos e interdependências para sistemas e *software.* A estrutura deste capítulo é mostrada na Figura 50 em formato BPMN.

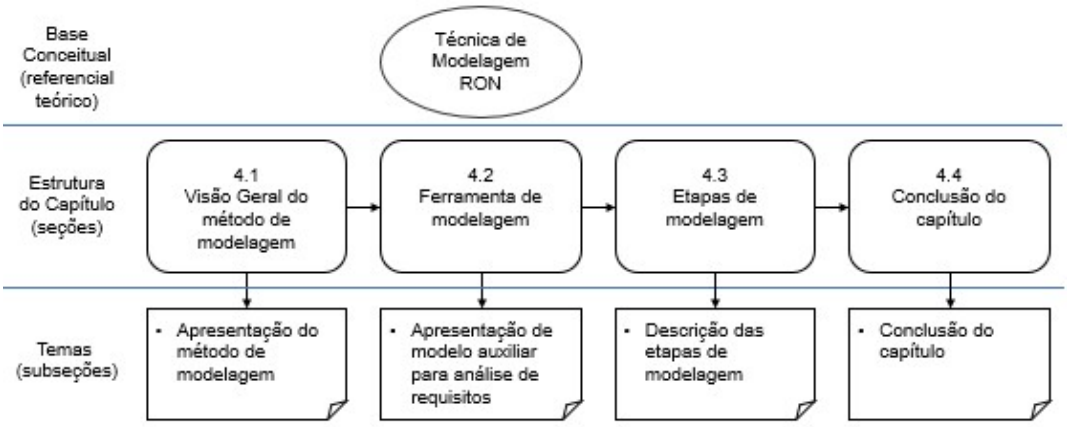


Figura 50 – Mapa conceitual/estrutural: Capítulo 4
Fonte: o autor

4.1 VISÃO GERAL DO MÉTODO DE MODELAGEM RIMON

O método RIMON visa orientar a criação de representações visuais de requisitos de sistemas e de *software*, por meio de diagramas que permitam expressar de forma clara e intuitiva requisitos, seus relacionamentos e as entidades do sistema. O método está estruturado da seguinte forma (Figura 51):

a) As **entradas** potenciais do processo podem incluir:
   - **Modelo de negócios do cliente**: o modelo de negócios do cliente pode oferecer um conjunto de informações que podem ser relevantes para modelar e/ou especificar os requisitos de um futuro sistema ou *software*. A seguintes definições de modelo de negócios são adotadas por este trabalho: “um modelo de negócios descreve o conteúdo, estrutura e governança das transações projetadas de modo a criar valor através da exploração de oportunidades de negócio” (AMIT e ZOTT, 2001) e “o modelo de negócios é um conjunto de expectativas sobre como o negócio será bem-sucedido em seu ambiente“ (DOWNING, 2005);

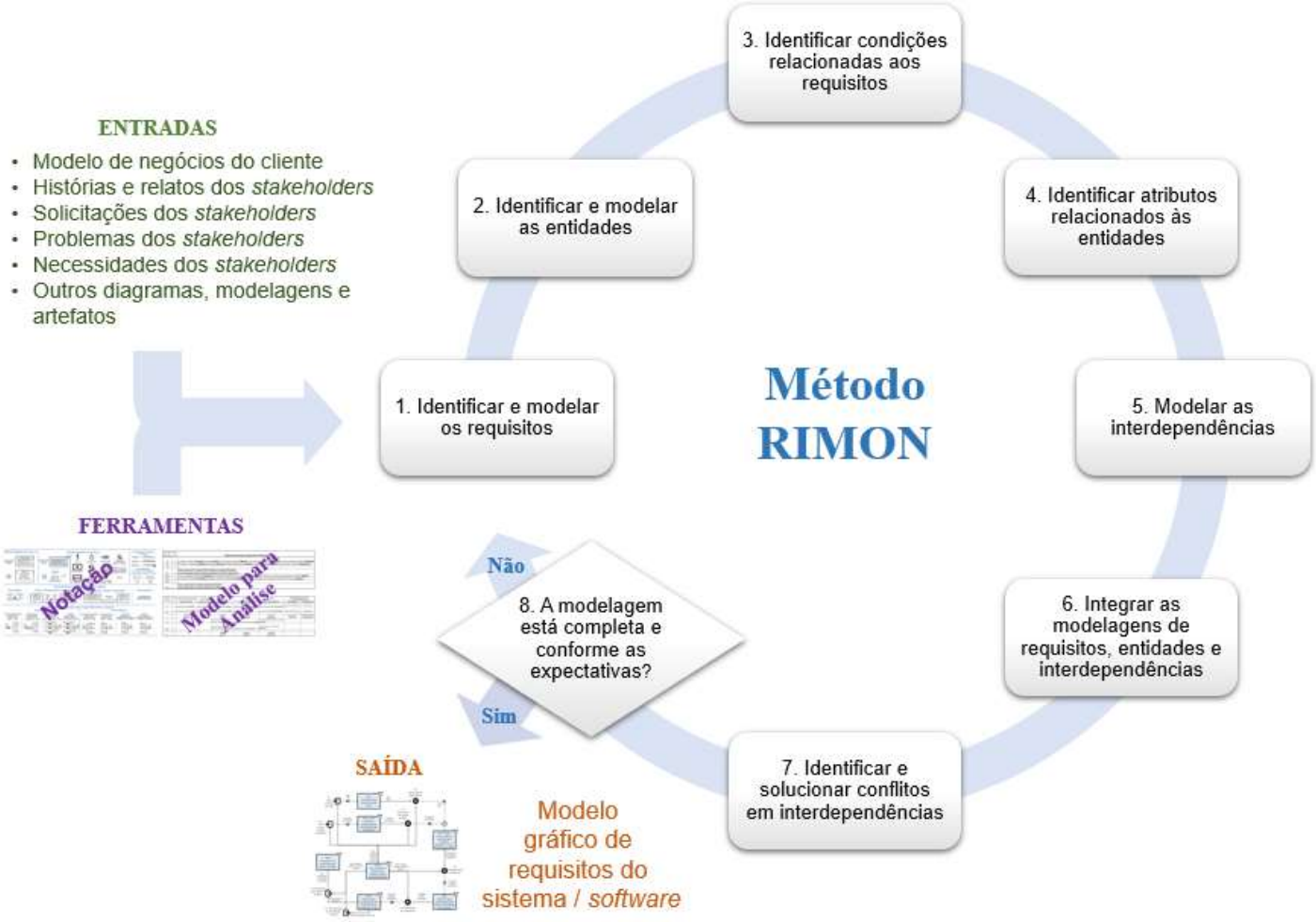


Figura 51 – Método de modelagem RIMON
Fonte: o autor

- **Histórias e relatos de *stakeholders***: uma história de usuário (*user story*) é uma técnica usualmente utilizada em métodos ágeis de SWE e se refere a descrições curtas e de alto nível da funcionalidade requerida (BOURQUE e FAIRLEY, 2014)*;*
- **Solicitações (*requests*) dos *stakeholders***: as *requests* de *stakeholders* representam de forma genérica o que está sendo pedido (solicitado). Segundo o dicionário, uma *request* pode ser definida como "*the act of asking for something*" ou "o ato de pedir por alguma coisa" (MERRIAN-WEBSTER, 2018);
- **Problemas (*problems*) dos *stakeholders***: "um problema pode ser definido como uma questão, que pode causar ou está causando um efeito (negativo) nos negócios da empresa" (SOUZA e STADZISZ, 2016). A percepção dos problemas pode tomar a forma de riscos, perdas de negócios e expectativas de melhorias (SOUZA, 2016). Nesse sentido, entende-se neste trabalho que o conhecimento dos problemas dos *stakeholders* fornece informações (*e.g.* expectativas de melhorias) que podem ser úteis para realizar a modelagem gráfica dos requisitos do sistema ou *software*;
- **Necessidades (*needs*) dos *stakeholders***: as *needs* representam o que o cliente necessita e são a fonte ou justificativa para os requisitos (SOUZA, 2016). Para um sistema, necessidades são capacidades ou coisas que estão faltando e que são procuradas (*wanted*) ou desejadas (*desired)* por uma ou mais partes interessadas (*stakeholders*) (INCOSE, 2015). Também podem ser definidos, segundo outra interpretação, como sendo requisitos de alto nível dos *stakeholders:* "requisitos de alto nível correspondem a requisitos abstratos (não-detalhados) escritos em linguagem natural (sem refinamento) a partir de informações e relatos providos por *stakeholders* (*early requirements*)" (SOMMERVILLE, 2010);
- **Outros diagramas, modelagens e artefatos**: outras entradas podem ser utilizadas, tais como diagramas de vislumbre de *software* (*software glance*), de visão de *software* (*e.g*: subseção 5.3.4,

modelagens gráficas de *early requirements* de outros métodos (*e.g:* seção 2.3). Também podem ser utilizados artefatos que contenham informações relevantes para a modelagem proposta, como por exemplo, atas de reunião com os *stakeholders*.

b) **Ferramentas:**

- **Notação RIMON** (Figura 49): esta ferramenta é **obrigatória** para realizar a modelagem gráfica. Deve ser escolhido um dialeto visual para realizar a modelagem (técnico ou negócios);
- **Planilha para identificação e análise dos elementos de modelagem** (Figura 52)**:** esta ferramenta é de uso **opcional** e sua construção será apresentada na seção 4.2. A planilha poderá ser gerada durante as atividades de identificação do método da Figura 51, tais como: 1 (identificar requisitos), 3 (identificar condições), 4 (identificar entidades) e 6 (identificar atributos).

c) Atividades:

As atividades propostas no método RIMON destinam-se a servir de auxílio ao engenheiro ou analista de requisitos na tarefa de identificar e modelar graficamente os requisitos, entidades e suas interdependências. O objetivo é permitir que seja possível integrar estes elementos em um ou mais diagramas visuais, bem como identificar conflitos em suas interdependências. Essas atividades foram organizadas em um ciclo iterativo em formato de círculo (Figura 51), de forma que possam ser repetidas até que se obtenha uma modelagem completa e de acordo com as expectativas dos *stakeholders.*

d) Saída:

Corresponde a uma especificação na forma de um modelo gráfico de requisitos de sistema ou de *software*, que represente os requisitos e suas interdependências de forma sistemática, precisa e expressiva.

4.2 FERRAMENTA PARA AUXÍLIO À ANÁLISE E MODELAGEM

A ferramenta para auxílio à análise e modelagem gráfica de requisitos proposta neste trabalho é apresentada na Figura 52. O objetivo desta é facilitar as atividades de identificação dos elementos de modelagem (*i.e.* Requisitos, Entidades, Condições, atributos), podendo ser, opcionalmente, utilizada durante as atividades do método RIMON.

Modelo de planilha de solicitações (*requests*) de entrada

Req. ID	Solicitações (sentenças) / *Requests* (*sentences*)
1	O sistema / software **(Sujeito)** deve exibir **(Ação)** a informação requerida **(Objeto)** em uma ordem específica **(Condição)** com uma restrição específica **(Restrição)**.
2	The system / software **(Subject)** shall display **(Action)** the required information **(Object)** in a specific order **(Condition)** with a specific restriction **(Restriction)**.
...	...
n	**Sintaxe (exemplo): (Sujeito) (Ação) (Objeto) (Condição) (Restrição)** **Syntax (example): (Subject) (Action) (Object) (Condition) (Restriction)**
1	Em uma condição específica **(Condição)** uma restrição específica deve se aplicar **(Restrição)** a todo o sistema ou a uma parte particular do sistema **(Object)**.
2	At a specific condition **(Condition)** a specific restriction **(Restriction)** should apply **(Action)** to the entire system or to particular element of the system **(Object)**.
...	...
n	**Sintaxe (exemplo): (Condição) (Restrição) (Ação) (Objeto)** **Syntax (example): (Condition) (Restriction) (Ação) (Object)**

Modelo de planilha para identificação e análise de elementos de modelagem

Req. ID	Entidade Principal (*Main Entity*)	Requisito (*Requirement*)	Entidade (*Entity*)	Condição (*Condition*)	Atributo (*Attribute*)	Padronização de termos (e.g.) Terms Standardization (e.g.)	
	(Sujeito) / (Subject)	(Ação) / (Action)	(Objeto) / (Object)	(Condição) / (Condition)	(Restrição) / (Restriction)	Antigo (original)	Padrão (standard)
1	O sistema / software **(Sujeito)**	deve exibir **(Ação)**	a informação requerida **(Objeto)**	em uma ordem específica **(Condição)**	com uma restrição específica **(Restrição)**	sistema / software (*e.g.*)	sistema (e.g)
2	The system / software **(Subject)**	shall display **(Action)**	the required information **(Object)**	in a specific order **(Condition)**	with a specific restriction **(Restriction)**	system / software (*e.g.*)	system (e.g)
...	...	...	...	...	...	...	...
n	**(Sujeito) (Subject)**	**(Ação) (Action)**	**(Objeto) (Object)**	**(Condição) (Condition)**	**(Restrição) (Restriction)**	**(termo antigo) (old term)**	**(termo padronizado) (standard term)**
1	-	deve se aplicar **(Ação)**	a todo o sistema ou a uma parte particular do sistema **(Object)**	Em uma condição específica **(Condição)**	uma restrição específica **(Restrição)**	-	-
2	-	should apply **(Action)**	to the entire system or to particular element of the system **(Object)**	At a specific condition **(Condition)**	a specific restriction **(Restriction)**	-	-
...	-	-	...	...	...	...	...
n	-	**(Ação) (Action)**	**(Objeto) (Object)**	**(Condição) (Condition)**	**(Restrição) (Restriction)**	-	-

Figura 52 – Modelos para identificação e análise de elementos de modelagem
Fonte: o autor

A Figura 52 apresenta duas planilhas tal como segue:

a) **Modelo de planilha de solicitações (*requests*) de entrada:** esta planilha é um exemplo de uma das possíveis entradas do método de modelagem RIMON (solicitações de *stakeholders*). Ela mostra como as informações poderiam aparecer nas solicitações do cliente, com base em uma sintaxe esperada, porém não obrigatória. Neste modelo, a primeira coluna se refere a um identificador do identificador "ID" das *requests*, enquanto a segunda coluna corresponde às sentenças de *requests* escritas em linguagem natural. Dois exemplos de possíveis **sintaxes** para solicitações são dados como exemplo nesta planilha, segundo o proposto na norma (ISO/IEC/IEEE 29148, 2011) e anteriormente detalhado na seção 2.6.1.

   É possível (pode-se até dizer que esperado) que as *requests* dos *stakeholders* não estejam padronizadas segundo os **formatos sintáticos** citados, uma vez que podem estar em linguagem natural (ou outros formatos). De qualquer forma, os elementos de informação a serem identificados e modelados serão os mesmos: **sujeitos, ações, objetos, condições, restrições**.

b) **Modelo de planilha para identificação e análise de elementos de modelagem:** esta planilha mostra um exemplo de como os elementos de modelagem podem ser identificados a partir das *requests* do cliente e separados nas colunas da planilha em seus elementos sintáticos correspondentes: **sujeito (Entidade Principal), ação (Requisito), objeto (Entidade), condição (Condição), restrição (atributo).**

   A planilha contém uma coluna inicial para identificar o *request* ("ID") do cliente bem como duas colunas ao final que permitem realizar a padronização dos termos empregados pelos *stakeholders* nas sentenças. Recomenda-se que os termos sejam padronizados, pois no uso da linguagem natural é possível (e comum) que um mesmo elemento de informação seja representado por diferentes textos.

   Exemplo: as seguintes sentenças (**S1** e **S2**) possuem alguns elementos de informação com o mesmo significado, porém com escritas diferentes:

   S1: "O sistema deve calcular o consumo de energia da casa do cliente com periodicidade mensal".

S2: “O sistema deve armazenar a informação de consumo energético da residência do consumidor em um banco de dados mensalmente”.

Neste exemplo, há elementos de informação com o mesmo significado:

Entidades:

S1: “consumo de energia” *versus* **S2**: “consumo energético”

S1: “casa do cliente” *versus* **S2**: “residência do consumidor”

Condições

S1: “com periodicidade mensal” *versus* **S2**: “mensalmente”

Neste caso, recomenda-se escolher um dos termos similares e passar a utilizar somente aquele termo em todas as ocorrências da modelagem. É importante que se assegure que os termos realmente se refiram à mesma entidade de informação (verificar com os *stakeholders* se necessário). Tais termos poderiam até mesmo ser adicionados a um “glossário de termos de projeto”, de forma a manter a uniformidade dos termos utilizados (OBS: não é escopo desta dissertação).

4.3 ATIVIDADES DE MODELAGEM

As próximas subseções contêm a descrição das atividades propostas no método RIMON (Figura 51).

As atividades que envolvem a **identificação e determinação** de informações (1 a 4) podem ser realizadas com a utilização **opcional** da planilha para análise e modelagem proposta na Figura 52.

As atividades em que a **modelagem gráfica** é realizada (1 a 6), envolvem a utilização obrigatória da notação RIMON (Figura 49), podendo ser realizadas em ferramentas gerais de desenho (*e.g.: Microsoft Powerpoint* ®) ou em ferramentas específicas desenvolvidas para tal finalidade.

4.3.1 Atividade 1: Identificar e modelar os requisitos

Identificar e modelar os **requisitos** consiste na análise dos potenciais artefatos de entrada do método RIMON (ver Figura 51) de forma a identificar os requisitos, determinar como os mesmos serão modelados e adicioná-los graficamente ao diagrama de requisitos. Segundo a norma ISO 29148, “**requisito** é uma declaração que traduz ou expressa uma necessidade e suas

restrições e condições associadas" (ISO/IEC/IEEE 29148, 2011). Nesse sentido, o analista ou engenheiro de requisitos deve estudar os artefatos de entrada com o objetivo de identificar a "declaração" que traduz ou expressa a necessidade do *stakeholder*.

Para exemplificar, as sentenças **S1** e **S2** a seguir representam solicitações (*requests)* de *stakeholders*, cuja modelagem é dada na Figura 53:

S1: "O sistema **deve calcular** o consumo de energia da casa do cliente com periodicidade mensal".

Nesta sentença, a expressão "**deve calcular**" representa a **ação** que caracteriza o requisito, enquanto que "**o consumo de energia**" representa o **objeto** ao qual a ação se refere. A modelagem é realizada por meio do uso do símbolo gráfico de *Requirement*, ao qual é adicionado o texto do requisito "**Calcular o consumo de energia**" e um identificador **R1**.

S2: "O destilador **deve produzir** água purificada para um sistema local de distribuição de água".

De forma similar ao exemplo anterior, a modelagem é realizada por meio do uso do símbolo gráfico de *Requirement*, ao qual é adicionado o texto do requisito "**Produzir água purificada**" e um identificador **R2**.

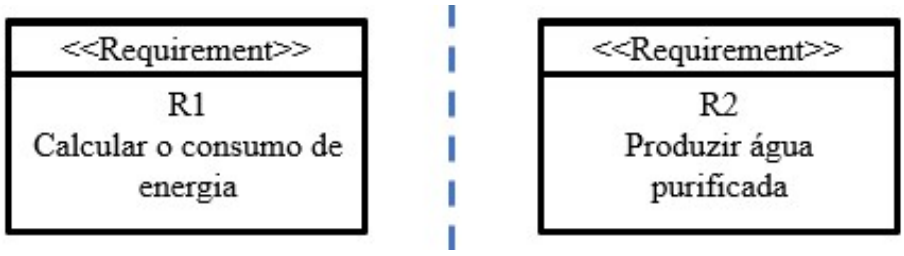


Figura 53 – Exemplo de modelagem de requisitos - método RIMON
Fonte: o autor

A Figura 53 mostra a modelagem do requisito da sentença **S1** por meio do requisito **R1** (lado esquerdo), e do requisito da sentença **S2** por meio do requisito **R2** (lado direito).

Os demais elementos sintáticos da sentença (entidades, condições, atributos etc) serão representados graficamente por outros símbolos, conforme será mostrado nas outras subseções deste capítulo.

4.3.2 Atividade 2: Identificar e modelar as entidades

Identificar e modelar as **entidades** envolve realizar a análise dos potenciais artefatos de entrada do método RIMON (ver Figura 51) de forma a determinar quais elementos de informação representam entidades, e adicioná-las graficamente ao diagrama de requisitos. Segundo a norma ISO 24765, uma "**entidade** é um objeto (ou seja, coisa, evento ou conceito) que ocorre em um modelo" (ISO/IEC/IEEE 24765, 2010) ou "**entidade** é a representação de um conjunto de coisas reais ou abstratas que são reconhecidas como do mesmo tipo porque compartilham as mesmas características e podem participar dos mesmos relacionamentos" (ISO/IEC/IEEE 24765, 2010).

Para exemplificar, as sentenças **S1** e **S2** a seguir representam solicitações (*requests)* de *stakeholders*, cuja modelagem é dada na Figura 54:

> **S1**: "O sistema deve calcular o consumo de energia da **casa do cliente** com periodicidade mensal".

Nesta sentença, o texto "**Casa do cliente**" representa a **entidade**. Este texto é adicionado ao símbolo gráfico de *Entity*, junto com um identificador **E1**.

S2: "O destilador deve produzir água purificada para um **sistema local de distribuição de água**".

De forma similar ao exemplo anterior, a modelagem é realizada por meio do uso do símbolo gráfico de *Entity*, ao qual é adicionado o texto que faz referência à entidade "**Sistema local de distribuição de água**" e um identificador **E2**.

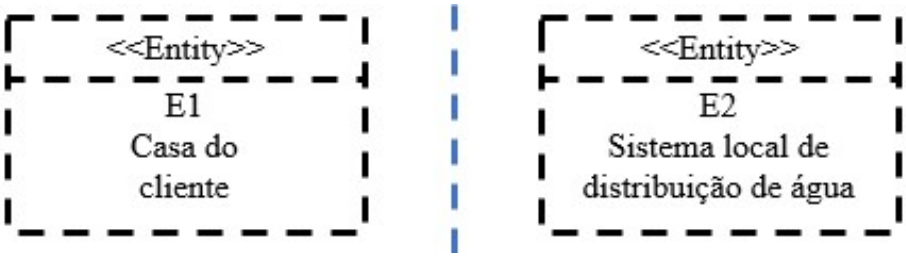


Figura 54 – Exemplo de modelagem de entidades - método RIMON
Fonte: o autor

A Figura 54 mostra a modelagem da entidade da sentença **S1** por meio da entidade **E1** (lado esquerdo), e da entidade da sentença **S2** por meio da entidade **E2** (lado direito).

Os demais elementos sintáticos da sentença (requisitos, condições, atributos etc) serão representados graficamente por outros símbolos, conforme será mostrado nas outras subseções deste capítulo.

4.3.3 Atividade 3: Identificar condições relacionadas aos requisitos

A identificação de **condições** consiste na análise dos requisitos modelados na "Atividade 1" (seção 4.3.1), de forma a determinar pré-condições ou pós-condições relacionadas aos requisitos. Uma definição pode ser dada como segue: "**condição** é algo essencial para o aparecimento ou ocorrência de outra coisa" (MERRIAN-WEBSTER DICTIONARY, 2018).

Para exemplificar, as sentenças **S3** e **S4** a seguir representam solicitações (*requests)* de *stakeholders*, em que se mostra a identificação de **condições**:

> **S3**: "As áreas seguras devem ser protegidas **por verificação de segurança** com base no ID do funcionário".

Nesta sentença, o texto "**por verificação de segurança**" representa a **condição**, para o requisito "As áreas seguras devem ser protegidas...", pois é essencial que a verificação de segurança ocorra para que as áreas seguras possam ser protegidas.

> **S4**: "O tempo entre as duas verificações de segurança independentes **não deve exceder** um período configurável".

Nesta sentença, o texto "**não deve exceder**" representa a **condição**, pois se refere a uma exigência (algo essencial) que deve acontecer para esta solicitação do cliente.

As condições identificadas nesta atividade 3 do método RIMON serão utilizadas para posteriormente modelar graficamente as interdependências na atividade 5 (subseção 4.3.5).

4.3.4 Atividade 4: Identificar atributos relacionados às entidades

A identificação de **atributos** consiste na análise das entidades modeladas na "Atividade 2" (seção 4.3.2), de forma a determinar quais os atributos relacionados a estas entidades. Um "**atributo** é uma propriedade inerente ou característica de uma entidade que pode ser distinguida quantitativa

ou qualitativamente por meios humanos ou automatizados" (ISO/IEC/IEEE 29148, 2011) (ISO/IEC/IEEE 24765, 2010).

As sentenças de exemplos anteriores **S1, S3** e **S4** reapresentadas a seguir contém solicitações (*requests)* de *stakeholders*, em que se mostra a identificação de **atributos**:

> **S1**: "O sistema deve calcular o **consumo de energia** da casa do cliente com periodicidade mensal".

Nesta sentença, o texto "**consumo de energia**" representa um **atributo** relacionado à entidade "casa do cliente", pois é uma propriedade quantitativa/qualitativa relacionada a esta entidade.

> **S3**: "As áreas seguras devem ser protegidas por verificação de segurança com base no **ID do funcionário**".

Nesta sentença, o texto "**ID do funcionário**" representa um **atributo**, relacionado à entidade "funcionário", pois é uma propriedade inerente a esta entidade, definida na solicitação do *stakeholder*.

> **S4**: "O tempo entre as duas verificações de segurança independentes não deve exceder um **período configurável**".

Nesta sentença, o texto "**período configurável**" representa o **atributo**, pois se refere a uma característica quantitativa relacionada a entidade que fará a verificação de segurança.

Os atributos identificados nesta atividade 4 do método RIMON serão utilizadas para posteriormente modelar graficamente as interdependências na atividade 5 (seção 4.3.5).

4.3.5 Atividade 5: Modelar as interdependências

Consiste na modelagem gráfica de **interdependências**, com base nas modelagens realizadas nas atividades anteriores (1. requisitos e 2. entidades) e nas informações identificadas (3. condições e 4. atributos).

Deve-se considerar que há duas possíveis direções para a realização da modelagem de cada interdependência:

a) **Modelagem de pré-condição (Entidade → Requisito)**: entidades devem ser interligadas de forma direcionada aos requisitos, por meio de pré-condições (compostas de **atributos** e **premissas**):
   - O **atributo** será modelado com base em informação de atributo relacionada à **entidade;**
   - A **premissa** será modelada com base em informação de **condição** relacionada ao **requisito**.

   Para exemplificar, a sentença **S3** a seguir representa uma solicitação (*request)* de um *stakeholder*, que foi modelada na Figura 55.

   S3: "As áreas seguras devem ser protegidas por verificação de segurança com base no ID do funcionário".
   Neste exemplo, foram identificados alguns elementos de modelagem:
   Requisito: "Verificar segurança de ID de funcionário para proteger áreas seguras".
   Entidade: "Leitor de Cartão ID" que permite realizar a "verificação de segurança" citada na solicitação.
   Condição: "por verificação de segurança".
   Atributo: "com base no ID do funcionário".

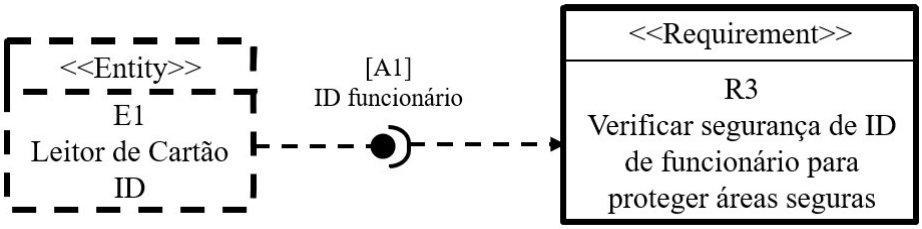


Figura 55 – Exemplo de modelagem de pré-condição - método RIMON
Fonte: o autor

   A Figura 55 mostra a modelagem da pré-condição no sentido de *Entity* para *Requirement*. Este diagrama representa a sentença **S3** e poderá ser posteriormente unido a outros diagramas de forma a compor o modelo de requisitos do sistema, conforme a atividade 6 do método RIMON (seção 4.3.6).

b) **Modelagem de pós-condição (Requisito → Entidade):** requisitos devem ser interligados de forma direcionada às entidades, por meio de

pós-condições (compostas de **inferências**). A **inferência** será modelada com base em informação de **ação** relacionada ao requisito.

Para exemplificar, a sentença **S2** a seguir representa uma solicitação (*request)* de um *stakeholder*, que foi modelada na Figura 56.

S2: "O destilador deve produzir água purificada para um **sistema local de distribuição de água**".

Neste exemplo, foram identificados alguns elementos de modelagem tal como segue

Requisito: "Produzir água purificada para um sistema local de distribuição de água".

Entidade: "Sistema local de distribuição de água.

Condição: "água purificada".

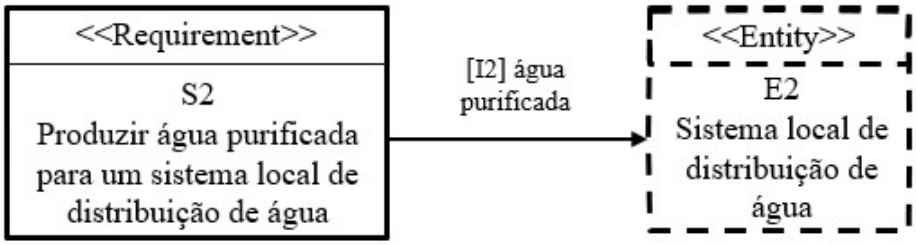


Figura 56 – Exemplo de modelagem de pós-condição - método RIMON
Fonte: o autor

A Figura 56 mostra a modelagem da pós-condição no sentido de *Requirement* para Entity. Este diagrama representa a sentença **S2** e poderá ser posteriormente integrado a outros diagramas de forma a compor o modelo gráfico de requisitos do sistema, conforme a atividade 6 do método RIMON (seção 4.3.6).

4.3.6 Atividade 6: Integrar as modelagens de requisitos, entidades e interdependências

Consiste na progressiva integração dos diagramas de requisitos criados na "Atividade 5" (seção 4.3.5) de forma a produzir um diagrama unificado de requisitos do sistema. A integração destes sub-diagramas poderá envolver a unificação de requisitos e de entidades, bem como a reconexão de interdependências conforme necessário.

A utilização de regras de composição de **interconexões de interdependências** (AND, OR, XOR em modalidades de junção e divisão) provavelmente será mais comum nesta atividade, uma vez que composições de pré-condições e pós-condições mais elaboradas poderão demandar estas representações.

O resultado desta atividade consistirá em um modelo de requisitos do sistema ou *software*, que proporcionará uma representação de requisitos, entidades e interdependências contendo tanto pré-condições com atributos (restrições) e premissas, quanto pós-condições com inferências. Exemplos destes diagramas (experimentos) são apresentados no capítulo 5 .

4.3.7 Atividade 7: Identificar e solucionar conflitos em interdependências

Consiste na identificação manual de conflitos existentes entre interdependências e na sinalização destes conflitos por meio da adição do símbolo de “Conflito Potencial” (*Potential Conflict)*.

O escopo desta dissertação não inclui o desenvolvimento de mecanismos para busca, identificação e resolução de conflitos (*e.g.:* mecanismo automático desenvolvido em uma ferramenta de *software*). Sugere-se que este tipo de desenvolvimento seja realizado em trabalhos futuros em função da complexidade envolvida.

4.3.8 Atividade 8: Verificar se a modelagem está completa e de acordo com as expectativas

A atividade 8 do método RIMON consiste em responder a seguinte pergunta: “A modelagem está completa e conforme as expectativas?”.

Em relação à **pergunta** em si, dois aspectos devem ser considerados:

a) “A modelagem está **completa?**”. Neste trabalho, uma modelagem completa é aquela em que o método RIMON foi aplicado de forma a contemplar **todas** as entradas do processo propostas pelos *stakeholders* (*e.g.:* todas as *requests* do cliente foram analisadas).
b) “A modelagem está **conforme as expectativas**?”. Esta questão visa determinar se a modelagem está **correta**, ou seja, não contém nada além do que os *stakeholders* precisam.

Em relação à **resposta** para a pergunta, há as seguintes possibilidades:

a) Caso a resposta seja "**Sim"**, o modelo gráfico de requisitos do sistema está completo e satisfaz as expectativas dos *stakeholders*. Neste caso, a aplicação do método pode ser encerrada.
b) Caso a resposta seja "**Não"**, o ciclo de modelagem deve ser reiniciado na atividade 1 do método RIMON. Nesta situação, outras entradas (requests) devem ser analisadas ou pode ser necessário rever o diagrama construído até o momento em questão. O ciclo pode se repetir tantas vezes quantas se julgar necessário, de forma a obter um diagrama completo e satisfatório para o cliente.

4.4 DISCUSSÃO SOBRE O MÉTODO

Este capítulo apresentou o método de modelagem gráfica RIMON (Figura 51), que apresenta detalhadamente um conjunto de atividades sugeridas para que o engenheiro ou analista de requisitos possa modelar requisitos, entidades, interdependências e identificação de conflitos de forma iterativa (cíclica) e sistemática, de forma a obter um modelo gráfico de requisitos completo e conforme as expectativas do cliente.

A modelagem pode ser realizada com base em diversos tipos de informações de entrada fornecidas por clientes ou *stakeholders*, tais como: modelo de negócios, histórias e relatos, atas de reunião, solicitações, problemas, necessidades, diagramas de vislumbre de *software* (*software glance*), diagramas de visão de *software* e até mesmo modelagens gráficas de *early requirements* de outros métodos. Adicionalmente, foi apresentada uma ferramenta para auxílio na análise e identificação dos elementos de modelagem, em formato de planilha a ser instanciada em *softwares* como excel ou outros), que pode ser opcionalmente utilizada pelo engenheiro de requisitos na condução das atividades de modelagem.

5 APLICAÇÕES - EXPERIMENTOS

Este capítulo tem por objetivo apresentar aplicações da linguagem de modelagem gráfica proposta neste trabalho. A linguagem RIMON foi utilizada em dois experimentos obtidos a partir da literatura de SE e um experimento originado a partir de um projeto real de s*oftware*, considerado aqui como um experimento de SWE. A estrutura deste capítulo é mostrada na Figura 57 em formato BPMN.

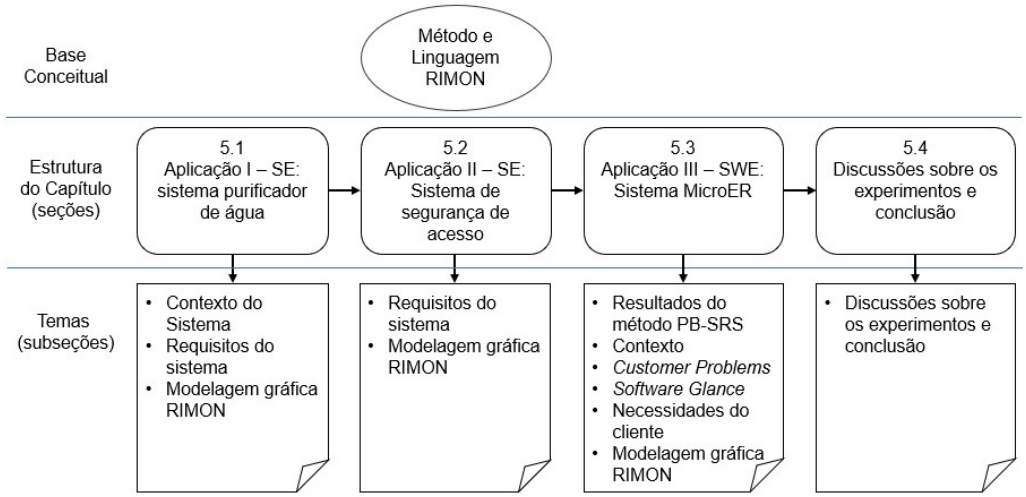


Figura 57 – Mapa conceitual/estrutural: Capítulo 5
Fonte: o autor

5.1 APLICAÇÃO I – SE: SISTEMA PURIFICADOR DE ÁGUA

Esta seção apresenta a aplicação da modelagem RIMON em um conjunto de requisitos de SE para um sistema de purificação de água por destilação (*Distiller System* - DS), extraído de (FRIEDENTHAL et al., 2014).

5.1.1 CONTEXTO DO SISTEMA

O sistema considerado neste primeiro experimento foi proposto para uma situação em que uma organização humanitária está preocupada em fornecer água potável a pessoas com dificuldades em sua obtenção, em regiões geralmente remotas e com poucos recursos financeiros. Como premissa, pode-se se assumir que existe água disponível no ambiente no qual o sistema deve operar, porém a mesma pode estar contaminada por vírus e bactérias, sendo muitas vezes imprópria para consumo (FRIEDENTHAL et al., 2014).

Considerando-se que o custo para estabelecer uma infraestrutura para levar água a lugares remotos é proibitivo, a decisão da organização humanitária foi a de buscar o desenvolvimento de um purificador de água (destilador) que fosse simples. O objetivo seria desenvolver um sistema que pudesse ser essencialmente operado com base em energia barata e disponível (*e.g.* energia solar, combustíveis sólidos ou líquidos) e que atingisse as temperaturas necessárias para fornecer água potável (FRIEDENTHAL et al., 2014).

5.1.2 REQUISITOS DO SISTEMA

A modelagem dos requisitos do sistema *Distiller System* (DS) foi realizada pelos autores originais segundo a abordagem *Model-Based Systems Engineering* (MBSE), utilizando SysML (*requirements diagram*) (FRIEDENTHAL et al., 2014). Foram representados em SysML tanto requisitos de missão (*Mission Requirements*) quanto requisitos derivados *(derived requirements*).

Neste estudo, foram utilizados como base os requisitos identificados pelos autores originais (funcionais e não-funcionais) em formato texto (linguagem natural), por serem de nível mais baixo (Tabela 38).

Req ID	Nome do Requisito	Descrição do Requisito
DS 1	Fonte de calor externa	O destilador deve ser projetado para acomodar uma fonte externa de calor.
DS 2	Entrada de água	O destilador deve acomodar uma fonte externa de água a ser purificada.
DS 3	Saída de água	O destilador deve produzir água purificada para um sistema local de distribuição de água.
DS 4	Complexidade essencial mínima	O destilador deve ser construído a partir de um conjunto mínimo essencial de peças.
DS 5	Purificação	O destilador deve purificar a água para que seja segura para beber.
DS 6	Alimentação por gravidade	O destilador deve permitir a alimentação por gravidade da água a ser purificada.

Req ID	Nome do Requisito	Descrição do Requisito
DS 7	Ebulição	O destilador deve ferver a água para esterilizá-la.
DS 8	Resfriamento	O destilador deve resfriar o que foi esterilizado para que possa ser distribuída com segurança.
DS 9	Remoção de resíduos	O destilador deve incorporar um mecanismo para remover resíduos resultantes do processo de destilação.

Tabela 38 – Requisitos derivados para um sistema de purificação de água
Fonte: adaptado (traduzido) de (FRIEDENTHAL et al., 2014).

Os requisitos apresentados na Tabela 38 seguem uma sintaxe padronizada tal como a sugerida na norma internacional 29148 (ISO/IEC/IEEE 29148, 2011), o que facilita a aplicação do processo de modelagem RIMON. Essa facilidade ocorre dado que os elementos sintáticos padronizados (Sujeito, Verbo, Objeto, Restrição e Condição) possuem correspondência com os principais elementos gráficos da modelagem (FBEs / entidades, regras, condições, atributos), conforme evidenciado na Tabela 39.

Construção Semântica	⇔	Elemento Sintático Padronizado
FBE (entidade)	⇔	(Sujeito) ou (Objeto)
Regra	⇔	(Verbo)+(Objeto)+(Restrição)+(Condição)
Condição	⇔	(Condição)
Atributo	⇔	(Restrição)

Tabela 39 – Relacionamento entre construções semânticas e elementos sintáticos padronizados para requisitos (ISO/IEC/IEEE 29148, 2011)
Fonte: O autor

5.1.3 MODELAGEM GRÁFICA RIMON - SISTEMA PURIFICADOR

Os resultados da modelagem do sistema purificador são ilustrados nos modelos gráficos da Figura 58 (notação técnica) e da Figura 59 (notação de negócios). A Tabela 40 apresenta o uso da planilha modelo auxiliar proposta no método RIMON (seção 4.2), aplicada aos requisitos propostos para sistema purificador.

Req. ID	Requisito (nome)	Entidade Principal	Requisito	Entidade	Condição	Atributo	Padronização de termos	
		(Sujeito)	(Ação)	(Objeto)	(Condição)	(Restrição)	Antigo	Padrão
1	Fonte de calor externa	O destilador	deve acomodar uma fonte externa de calor	mecanismo de aquecimento	fonte externa de calor	calor		
2	Entrada de água	O destilador	deve acomodar uma fonte externa de água (a purificar)	mecanismo de entrada de água	fonte externa de água	água (a purificar)		
3	Saída de água	O destilador	deve produzir água purificada para um sistema local de distribuição de água	sistema local de distribuição de água	água purificada (fria) a ser distribuída		água purificada	água purificada (fria)
4	Complexidade essencial mínima	O destilador	deve ser construído a partir de um conjunto mínimo essencial de peças	mecanismo de aquecimento, mecanismo de entrada de água, mecanismo de purificação, mecanismo de resfriamento	conjunto mínimo essencial de peças			
5	Purificação	O destilador	deve purificar a água (a purificar) para que seja segura para beber	mecanismo de purificação	água purificada (quente)	água purificada (quente)	água	água (a purificar)
6	Alimentação por gravidade	O destilador	deve permitir a alimentação por gravidade da água (a purificar)	mecanismo de entrada de água	alimentação por gravidade	água (a purificar)	água	
7	Ebulição	O destilador	deve ferver a água (a purificar) para esterilizá-la	mecanismo de purificação	água fervida esterilizada		água	água (a purificar)
8	Resfriamento	O destilador	deve resfriar a água purificada (quente) para que possa ser distribuída com segurança	mecanismo de resfriamento	água purificada (fria) a ser distribuída	água purificada (fria)	água	água purificada (quente)
9	Remoção de resíduos	O destilador	deve incorporar um mecanismo para remover resíduos resultantes do processo de destilação	mecanismo de purificação	remoção de resíduos			

Tabela 40 – Modelo aplicado aos requisitos de sistema de purificação de água
Fonte: o autor

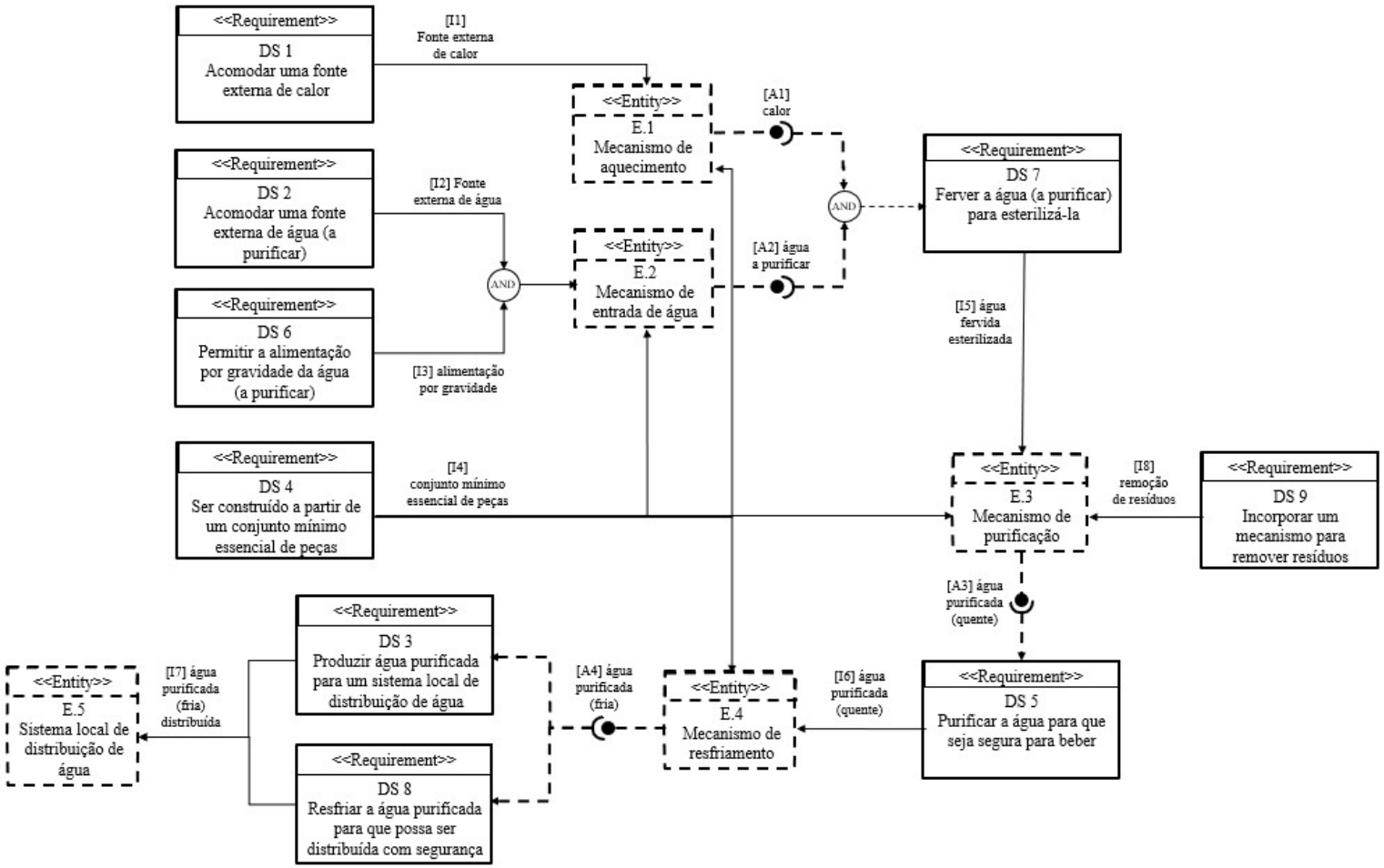


Figura 58 – Modelagem RIMON de sistema purificador (notação técnica)
Fonte: o autor

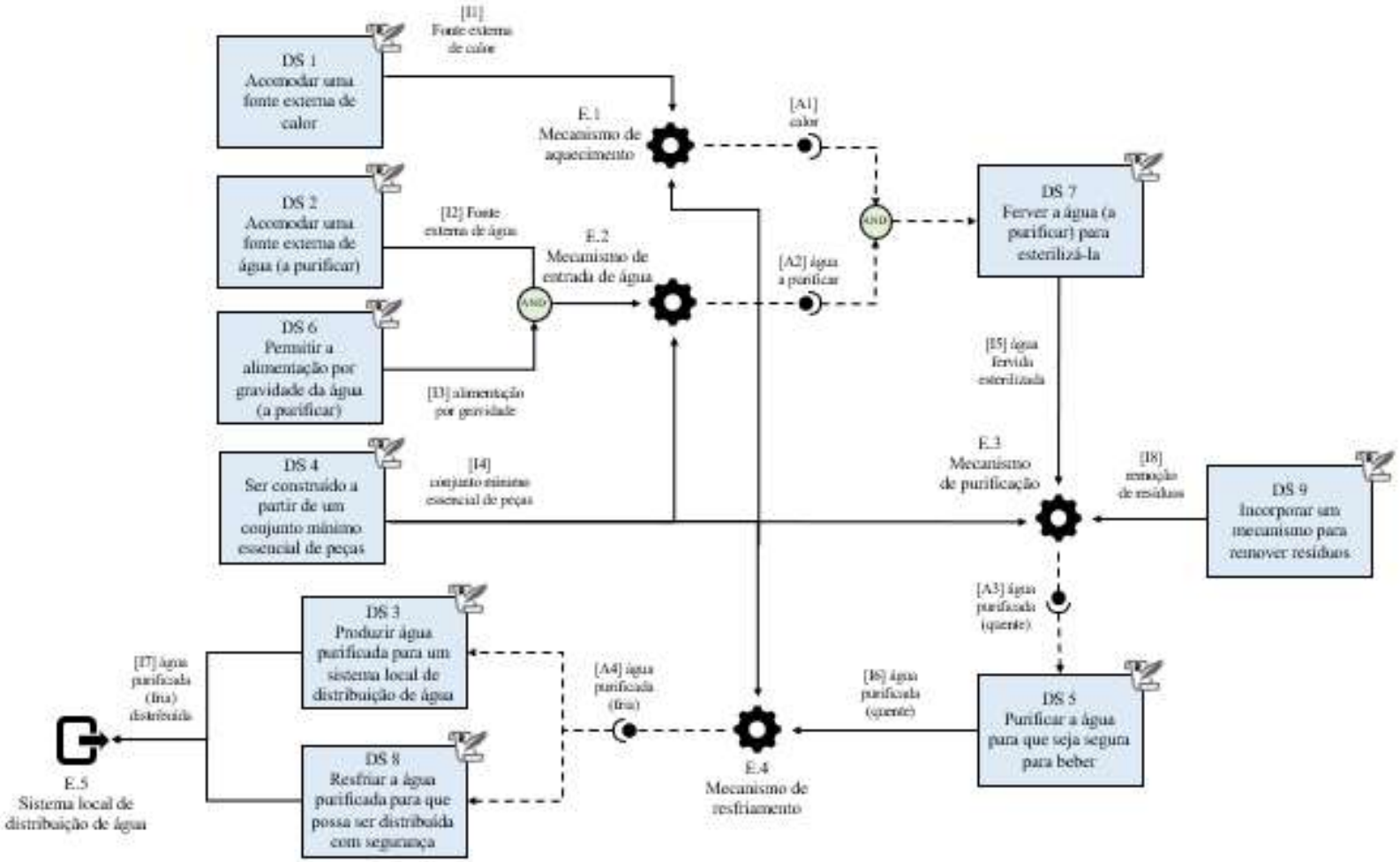


Figura 59 – Modelagem RIMON de sistema purificador (notação de negócios)
Fonte: o autor

5.2 APLICAÇÃO II – SE: SISTEMA DE SEGURANÇA DE ACESSO

Esta seção apresenta a aplicação da modelagem RIMON em um conjunto de requisitos propostos na literatura de SE para um sistema de segurança de acesso (*Access Security System*), extraído a partir do INCOSE *Systems Engineering Handbook v.3.2* (INCOSE, 2010), e apresentados brevemente na seção 2.5.6 (SIMÃO et al., 2016).

5.2.1 REQUISITOS DO SISTEMA

Os requisitos do sistema de segurança de acesso são fornecidos na Tabela 41, traduzidos a partir de (SIMÃO et al., 2016).

Req ID	Requisito
SS11-a	As áreas seguras devem ser protegidas por verificação de segurança com base no ID do funcionário.
SS11-b	As áreas seguras devem ser protegidas por uma segunda verificação de segurança independente com base em dados biométricos.
SS11-c	O tempo entre as duas verificações de segurança independentes não deve exceder um período configurável.
SS11-d	O usuário deve ter três tentativas de identificação biométrica.
SS11-e	O usuário terá três tentativas de identificação do cartão.
SS11-f	Qualquer tentativa de acesso negado deve ser enviada ao administrador.

Tabela 41 – Requisitos para um sistema de segurança de acesso
Fonte: adaptado (traduzido) de (SIMÃO et al., 2016).

5.2.2 MODELAGEM GRÁFICA RIMON - SISTEMA DE SEGURANÇA

Os resultados da modelagem do sistema de segurança são ilustrados nos modelos gráficos da Figura 60 (notação técnica) e da Figura 61 (notação de negócios). A Tabela 42 apresenta o uso da planilha modelo auxiliar proposta no método RIMON (seção 4.2), aplicada aos requisitos do sistema de segurança de acesso.

Req. ID	Entidade Principal	Requisito	Entidade	Condição	Atributo	Padronização de termos	
	(Sujeito)	(Ação)	(Objeto)	(Condição)	(Restrição)	Antigo (original)	Novo (Padrão)
SS11-a	O sistema	deve proteger	as áreas seguras leitor de cartão ID	por meio de verificação de segurança	status checagem ID ID funcionário	com base no ID do funcionário	ID funcionário
SS11-b	O sistema	deve proteger	as áreas seguras leitor de biometria BIO	por meio de uma segunda verificação de segurança independente	status checagem BIO BIO funcionário	com base em dados biométricos	BIO funcionário
SS11-c	O sistema	deve garantir	que o tempo entre duas verificações de segurança independentes	não exceda um período configurável	status checagem periodo tempo ID tempo BIO tempo atual período configurável		
SS11-d	O sistema	deve permitir	ao usuário	realizar três tentativas	status checagem tentativas contador tentativas BIO	tentativas de identificação biométrica	tentativas BIO
SS11-e	O sistema	deve permitir	ao usuário	realizar três tentativas	status checagem tentativas contador tentativas ID	tentativas de identificação do cartão	tentativas ID
SS11-f	O sistema	deve enviar	ao administrador	informações sobre qualquer tentativa de acesso negado	status acesso		

Tabela 42 – Modelo aplicado aos requisitos de sistema de segurança
Fonte: o autor

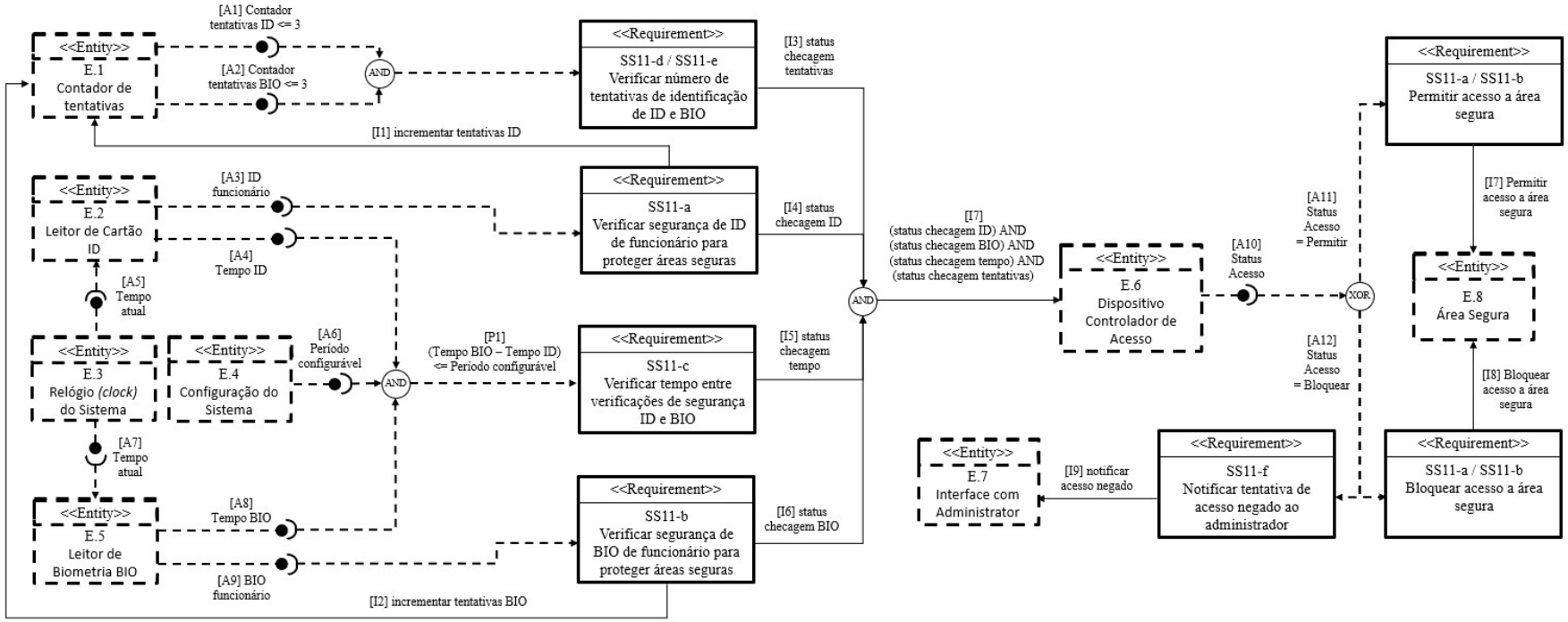


Figura 60 – Modelagem RIMON de sistema de segurança (notação técnica)
Fonte: o autor

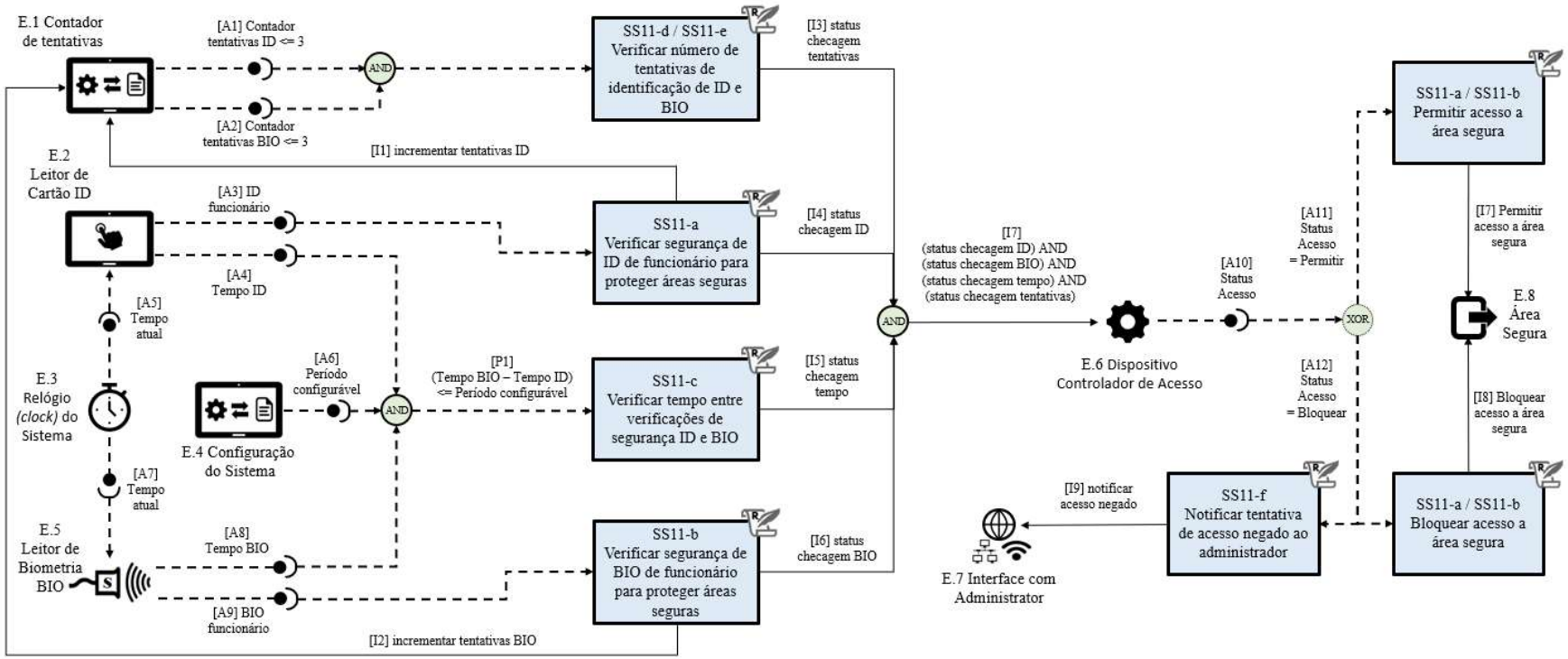


Figura 61 – Modelagem RIMON de sistema de segurança (notação de negócios)
Fonte: o autor

5.3 APLICAÇÃO III – SWE: SISTEMA MICROER

Este capítulo apresenta uma aplicação da modelagem RIMON ao conjunto de requisitos de um projeto real denominado MicroER (STADZISZ, 2016). Este projeto tem como escopo o desenvolvimento de um sistema computacional para gerenciamento residencial da produção e consumo de energia elétrica renovável. Ele está sendo desenvolvido na UTFPR pelo Laboratório de Inovações Tecnológicas (LIT) do Centro de Inovações Tecnológicas (CITEC), com apoio financeiro do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) (SOUZA, 2016).

Este experimento será feito com base nas necessidades do cliente do sistema MicroER, obtidos durante a aplicação do método *Problem-Based SRS* (SOUZA, 2016). Este método não é condição obrigatória para realizar a modelagem, porém torna mais claro o contexto do sistema e facilita a aplicação da técnica, uma vez que resulta em sentenças em linguagem padronizada.

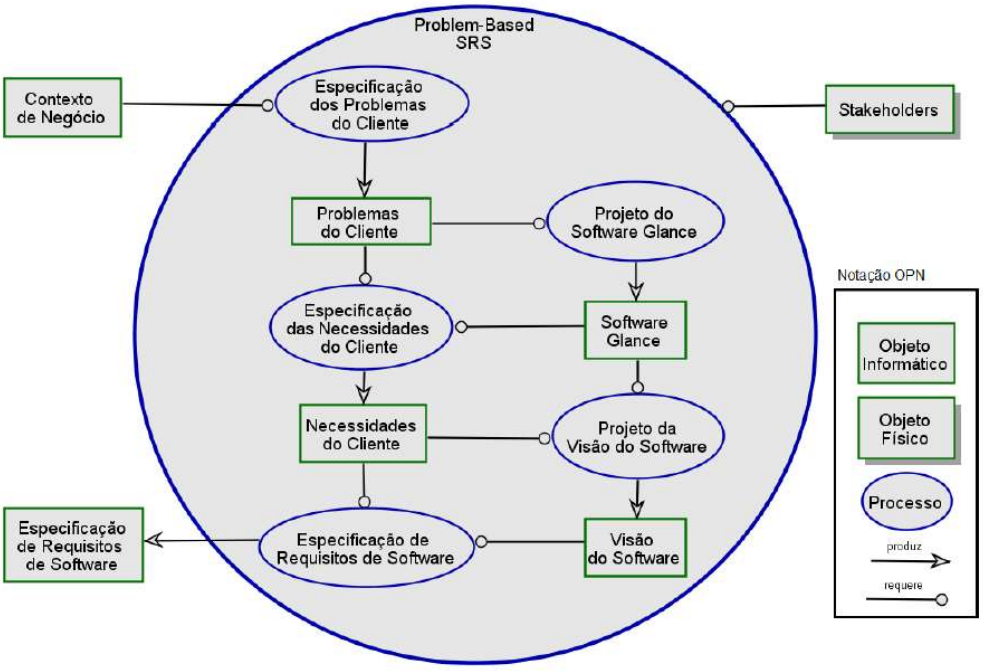


Figura 62 – Diagrama do processo do Problem-Based SRS em OPD
Fonte: (SOUZA, 2016)

A Figura 62 apresenta o processo do método *Problem-Based* SRS modelado em um diagrama OPM chamado de OPD (*Object Process Diagram*), cuja explicação detalhada está disponível no trabalho original do autor (SOUZA,

2016). É possível observar na figura os artefatos gerados a partir do método nas caixas com borda verde: Problemas do Cliente, *Software Glance,* Necessidades do Cliente, Visão do *Software,* Especificação de Requisitos do *Software.*

A seção 5.3.1 apresenta os supracitados artefatos da aplicação do método *Problem-Based* SRS obtidos no trabalho de Souza para o sistema MicroER (SOUZA, 2016), até a atividade de "**necessidades do cliente**". A partir destas, foi realizada a modelagem RIMON e construído o diagrama de requisitos do sistema.

5.3.1 RESULTADOS DO MÉTODO PROBLEM-BASED SRS

Os seguintes resultados do método *Problem-Based* SRS aplicado ao sistema MicroER (SOUZA, 2016) serão apresentados de forma sucinta nas seções 5.3.2 a 5.3.5, conforme segue:

a) **Contexto de negócio**: corresponde ao conjunto de informações que caracteriza uma porção de um domínio de negócios e define o escopo para condução de ações como a análise, modificação e melhoria dos processos de negócio (SOUZA, 2016);
b) **Problemas do cliente** (*Customer Problems - CPs*): corresponde a um conjunto de sentenças escritas sob um formato padrão e técnico, visando representar os problemas percebidos, ou seja, situações que possam causar ou estejam causando um efeito (usualmente negativo) nos negócios do cliente (SOUZA, 2016);
c) ***Software Glance* (SG)*:*** primeira representação da ideia de organização do sistema que está sendo vislumbrada pelo analista após conhecer os problemas do cliente (SOUZA e STADZISZ, 2016);
d) **Necessidades do Cliente** (*Customer Needs - CNs*): corresponde a um conjunto de sentenças escritas sob um formato padrão e técnico, visando representar as necessidades, ou seja, o que é necessário para resolver ou minimizar os problemas percebidos (SOUZA, 2016)

5.3.2 CONTEXTO DE NEGÓCIO

O projeto MicroER da UTFPR visa desenvolver um sistema eletrônico de gerenciamento e otimização relacionados à uma unidade de microgeração

contendo fontes de energia solar e eólica. Ao usar componentes de *hardware* e *software,* o sistema deve auxiliar usuários a ajustar dinamicamente seu consumo de energia segundo suas preferências, de forma a diminuir o consumo e estender a duração de uso de energia gerada por fontes renováveis (STADZISZ, 2016) (SOUZA, 2016). A Figura 63 ilustra uma visão arquitetural dos módulos eletrônicos de gerenciamento que compõem o sistema MicroER e dos módulos que compõem a unidade de micro geração (SOUZA, 2016).

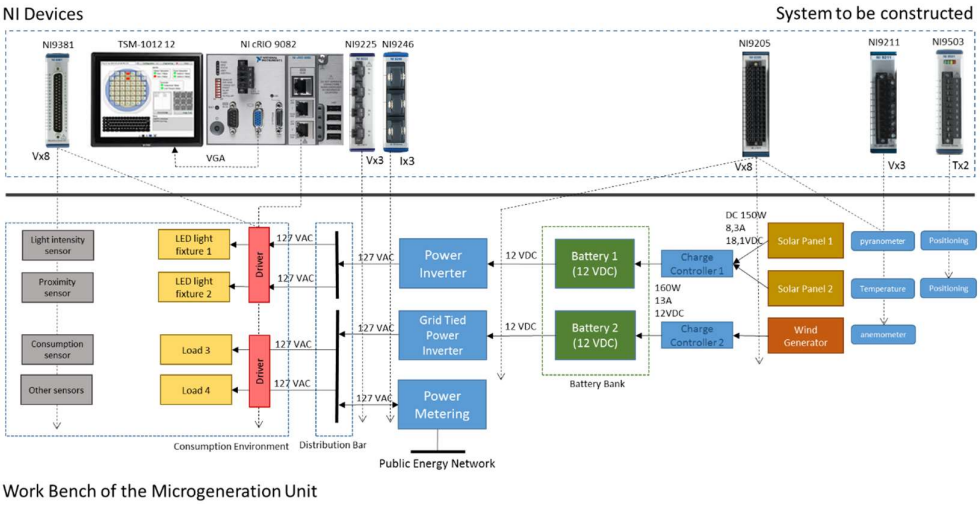


Figura 63 – Esquema do Projeto MicroER
Fonte: (STADZISZ, 2016)

A Tabela 43 apresenta o contexto de negócio definido para aplicação do método *Problem-Based SRS* neste experimento.

Contexto de negócio
O usuário final residencial preocupa-se com o uso racional das fontes de energia e escolheu investir na produção de energia renovável por meio de uma unidade micro geradora de energia com fontes solar e eólica. O usuário também tem interesse em reduzir o seu consumo, como meio para diminuir impacto na matriz energética regional e nacional. O usuário deverá ser capaz de interagir com os mecanismos de controle de forma a poder adequá-los a seu perfil de consumo e de evoluir este perfil de acordo com sua experiência no uso de energias renováveis. O usuário poderá ter conhecimento da eficiência do sistema, de sua qualidade de operação e de como seu perfil de consumo se adequa às condições energéticas providas pelo sistema de microgeração.

Tabela 43 – Contexto de negócio para o sistema MicroER
Fonte: (SOUZA, 2016), adaptado de (STADZISZ, 2016).

5.3.3 PROBLEMAS DO CLIENTE - CPs

Os problemas do cliente documentados para o sistema MicroER são mostrados na Tabela 44.

Problemas do Cliente (CPs)
CP.1 - O cliente intenciona reduzir seu consumo visando diminuir o custo com energia elétrica fazendo uso de um sistema de microgeração de energia renovável baseado em fonte solar e eólica.
CP.1.1 - O cliente deve reduzir o consumo desnecessário de energia para que possa reduzir o custo com energia.
CP.1.2 - O cliente deve mudar seu padrão de consumo, sob pena de não conseguir reduzir seu consumo.
CP.1.3 - O cliente deve acompanhar seu padrão de consumo, sob pena de não conseguir reduzir seu consumo.
CP.1.4 - O cliente deve garantir a eficiência da unidade geradora, sob pena de não obter a energia necessária que a unidade pode gerar e ter que consumir da rede pública.
CP.1.5 - O cliente deve garantir o estado de manutenção da unidade geradora, sob pena de falha de funcionamento e ter que consumir energia da rede pública.
CP.2 - O cliente pretende contribuir para minimizar os impactos ambientais das fontes públicas de energia.
CP.2.1 - O cliente deve ter ciência do impacto ambiental de seu consumo.
CP.2.2 - O cliente intenciona racionalizar o uso da energia.
CP.3 - O cliente intenciona contribuir para a redução da pressão de consumo sobre a matriz energética regional e nacional.
CP.4 - O cliente intenciona influenciar outros indivíduos para a causa energética e ambiental.

Tabela 44 – Problemas do cliente para o sistema MicroER
Fonte: (SOUZA, 2016)

5.3.4 SOFTWARE GLANCE - SG

A Tabela 45 apresenta o *software glance* obtido como resultado para o sistema MicroER (SOUZA, 2016).

Software Glance – sistema MicroER
O *software* do sistema MicroER tem como principal objetivo gerenciar equipamentos e sensores integrados à uma unidade de microgeração de energia na residência do usuário final. Este *software* de gerenciamento deve realizar a aquisição de sinais de diferentes equipamentos e sensores e armazená-los em uma base de dados. Através de uma interface humano-computador, o usuário pode acompanhar seu padrão de consumo, qualidade de operação do sistema e registrar perfis de consumo. Adicionalmente, ele pode ativar ou desativar equipamentos através de componentes de comando.

Tabela 45 – *Software Glance* para o sistema MicroER
Fonte: (SOUZA, 2016)

A Figura 64 ilustra graficamente o *software glance* descrito na Tabela 45 utilizando um diagrama de blocos.

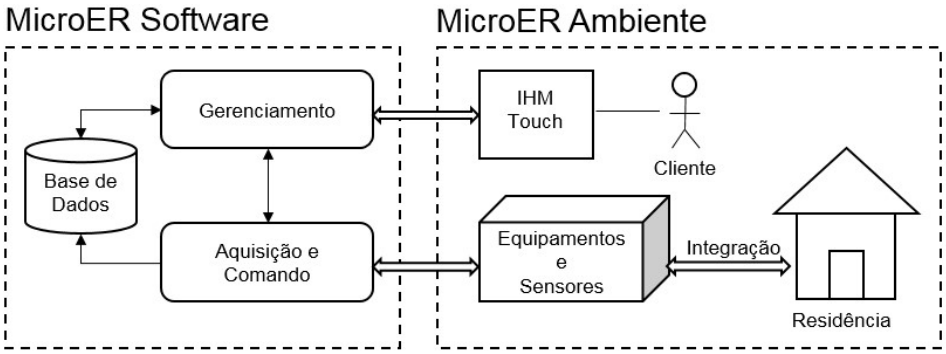


Figura 64 – Diagrama de *Software Glance* para o sistema MicroER
Fonte: (SOUZA, 2016)

Neste diagrama, o sistema MicroER é dividido em dois subsistemas: o MicroER *software* e o MicroER ambiente. O subsistema de *software* apresenta os componentes de controle, aquisição e comando que persistem e consultam informações em uma base de dados. O subsistema de Ambiente contém os equipamentos, sensores e interfaces presentes na residência do usuário final (SOUZA, 2016) .

5.3.5 NECESSIDADES DO CLIENTE - CNs

As necessidades do cliente documentadas para o sistema MicroER são mostrados na Tabela 46.

Necessidades do Cliente (CNs)
CN.1 - O cliente precisa de um *software* para monitorar e controlar a produção e o consumo energético.
CN.1.1 - O cliente precisa de um *software* para ter conhecimento de seu padrão de consumo (quanto e com o que consume energia elétrica).
CN.1.2 - O cliente necessita de um *software* para ajustar o consumo em função da capacidade energética renovável e do perfil de consumo.
CN.1.3 - O cliente deseja um *software* para ter conhecimento da produção energética da unidade geradora.
CN.1.4 - O cliente precisa de um *software* para ser alertado sobre a queda de eficiência da geração de energia.
CN.1.5 - O cliente precisa de um *software* para ser alertado sobre as exigências de manutenção da unidade geradora.
CN.2 - O cliente precisa de um *software* para ter conhecimento da racionalização de seu consumo.
CN.2.1 - O cliente quer um *software* para saber o quanto seu consumo está racionalizado e colaborando com os fatores ambientais.
CN.2.2 - O cliente deseja um *software* para acompanhar a evolução da racionalização de seu padrão de consumo.
CN.3 - O cliente quer saber a relação de independência de seu consumo da rede elétrica pública.
CN.4 - O cliente quer um *software* para saber os benefícios gerados com o melhor aproveitamento da fonte energética para divulgar a outras pessoas.

Tabela 46 – Necessidades do cliente para o sistema MicroER
Fonte: (SOUZA, 2016)

5.3.6 MODELAGEM GRÁFICA RIMON - SISTEMA MICROER

Os resultados da modelagem gráfica RIMON para o sistema microER são mostrados na Figura 65 (notação técnica) e Figura 66 (notação de negócios). A Tabela 47 apresenta o uso da planilha modelo auxiliar proposta no método RIMON (seção 4.2), aplicada às necessidades do cliente do sistema MicroER.

Req. ID	Entidade Principal	Requisito	Entidades	Condições	Atributos	Padronização de termos	
	(Sujeito)	(Ação)	(Objeto)	(Condição)	(Restrição)	Antigo (original)	Novo (Padrão)
CN 1	software	monitorar e controlar a produção e o consumo energético.	cliente base de dados		produção energética consumo energético	produção e consumo energético	produção energética consumo energético
CN 1.1	software	ter conhecimento de seu padrão de consumo (quanto e com o que consume energia elétrica)	cliente equipamentos base de dados		consumo energético	(... com o que consome energia elétrica) padrão de consumo	equipamentos consumo energético
CN 1.2	software	ajustar o consumo em função da capacidade energética renovável e do perfil de consumo	cliente base de dados	em função da capacidade energética renovável e do perfil de consumo	produção energética consumo energético perfil de consumo	capacidade energética renovável consumo	produção energética consumo energético
CN 1.3	software	ter conhecimento da produção energética da unidade geradora.	cliente unidades geradoras base de dados		produção energética		
CN 1.4	software	ser alertado sobre a queda de eficiência da geração de energia	cliente base de dados	queda de eficiência da produção energética	eficiência da produção energética	geração de energia	produção energética
CN 1.5.1	software	registrar exigências de manutenção da unidade geradora	cliente unidades geradoras base de dados		exigências de manutenção		
CN 1.5.2	software	visualizar as exigências de manutenção da unidade geradora	cliente unidades geradoras base de dados		exigências de manutenção		
CN 2	software	ter conhecimento da racionalização de seu consumo	cliente base de dados		consumo racionalizado	racionalização de consumo	consumo racionalizado
CN 2.1	software	saber o quanto seu consumo está racionalizado e colaborando com os fatores ambientais	cliente ambiente base de dados		consumo racionalizado fatores ambientais	quanto de consumo está racionalizado	consumo racionalizado
CN 2.2	software	acompanhar a evolução da racionalização de seu padrão de consumo	cliente base de dados		evolução do consumo racionalizado	racionalização do padrão de consumo	consumo racionalizado
CN 3	software	saber a relação de independência de seu consumo da rede elétrica pública.	cliente rede eletrica pública base de dados		independência energética consumo energético	relação e independência de consumo em relação a rede elétrica pública consumo	independência energética consumo energético
CN 4	software	saber os benefícios gerados com o melhor aproveitamento da fonte energética para divulgar a outras pessoas.	cliente unidades geradoras base de dados		benefícios de economia gerados	fonte energética benefícios gerados com melhor aproveitamento	unidades geradoras benefícios de economia gerados

Tabela 47 – Modelo aplicado às necessidades do cliente do sistema MicroER
Fonte: o autor

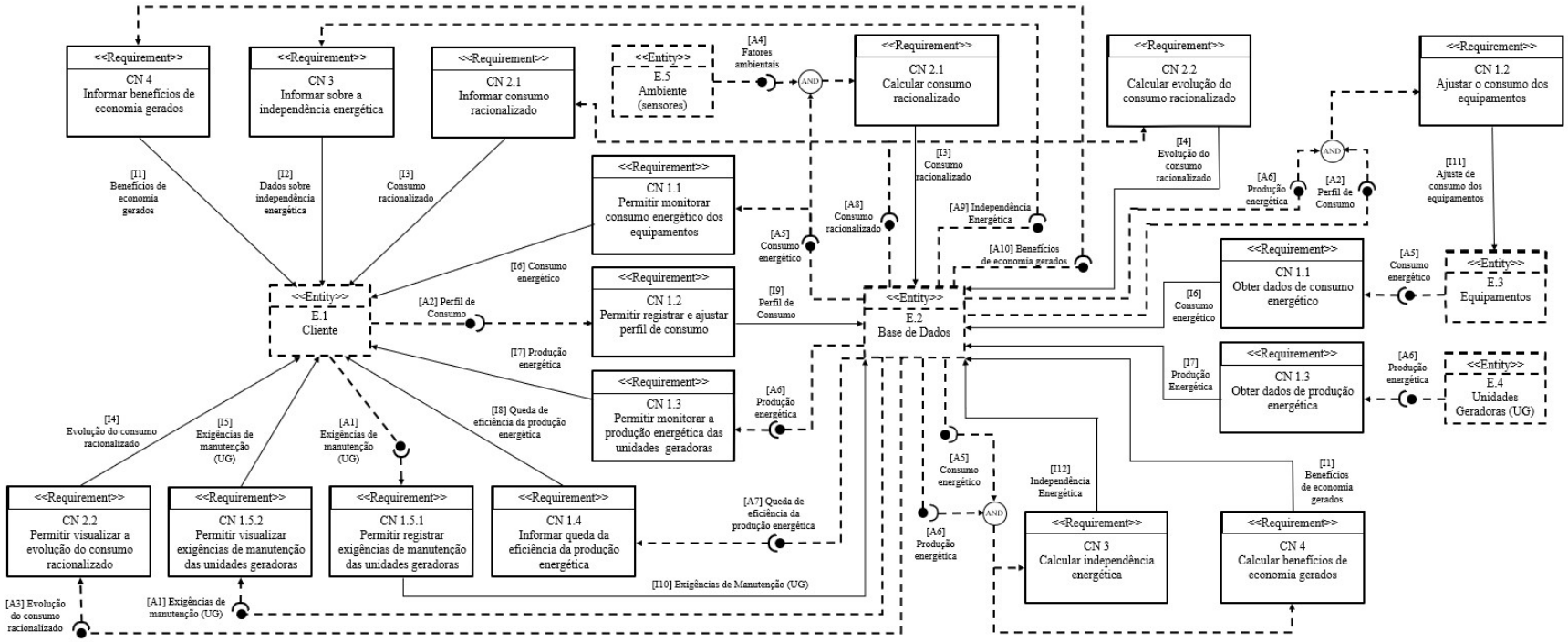


Figura 65 – Modelagem RIMON para o sistema MicroER (notação técnica)
Fonte: o autor

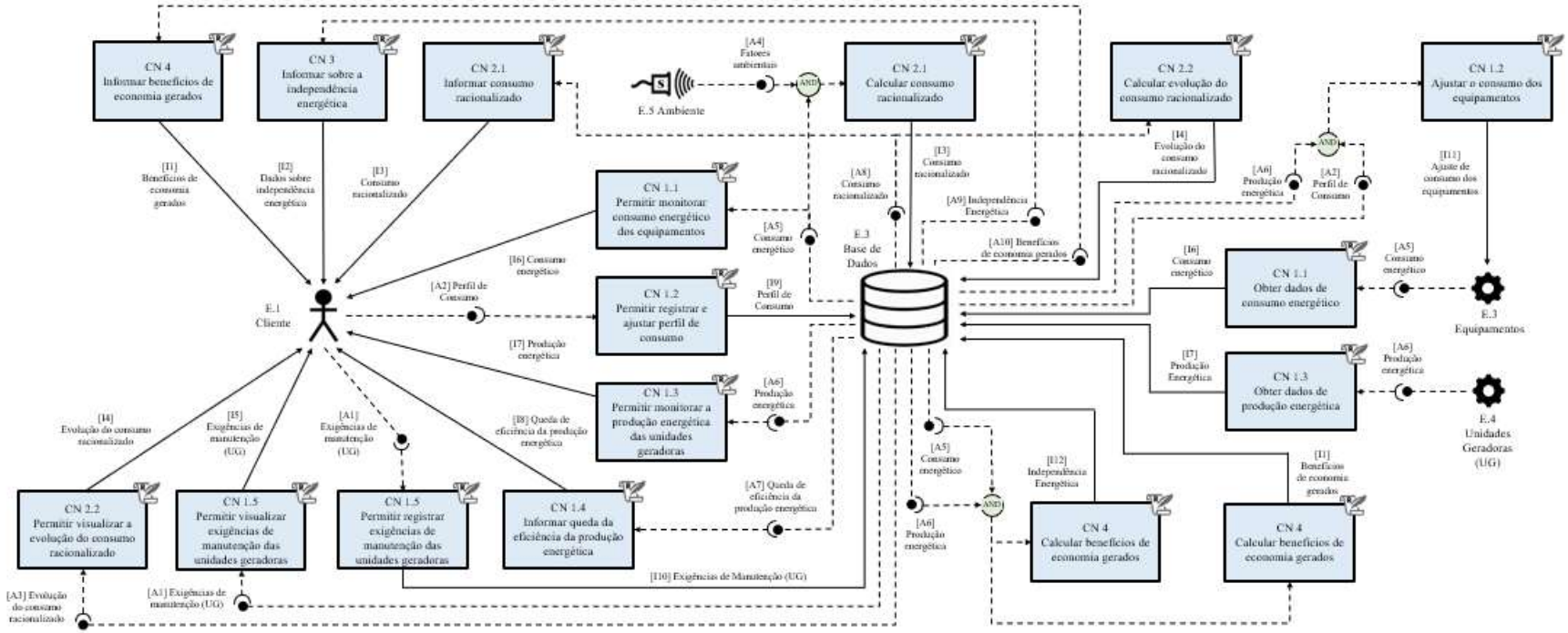


Figura 66 – Modelagem RIMON para o sistema MicroER (notação de negócios)
Fonte: o autor

5.4 DISCUSSÕES SOBRE OS EXPERIMENTOS E CONCLUSÃO

Este capítulo apresentou três experimentos de modelagem de requisitos utilizando a linguagem RIMON, como forma de demonstrar sua aplicabilidade. As modelagens foram conduzidas com sucesso, sendo possível apontar que ambos os dialetos visuais (notação técnica e notação de negócios) foram utilizados em cada um dos experimentos. A Tabela 48 apresenta um breve resumo sobre alguns dados dos experimentos.

Experimento	Tipo de Sistema	Quantidade de r*equests*	Tempo de modelagem (horas)
Sistema purificador de água (*Distiller System)*	Físico	*9*	8
Sistema de segurança de acessos (*INCOSE Access Security System*)	Físico + *software*	6	10
Sistema de microgeração de energia (microER)	Físico + *software*	11	12

Tabela 48 – Verificação PoN: Clareza Semiótica – Dialeto Técnico
Fonte: o autor com base em (LINDEN et al., 2017)

O primeiro experimento realizado, extraído da literatura de SE (FRIEDENTHAL et al., 2014), refere-se a um **sistema purificador de água** para uso em regiões remotas. É interessante notar que se trata de um sistema físico que não envolve a utilização de *software*. A modelagem partiu de 9 *requests* pré-existentes e levou em torno de 8 horas para ser construída pelo autor para os dois dialetos visuais (ver Tabela 48).

O segundo experimento realizado, também extraído da literatura de SE (INCOSE, 2010), refere-se a um **sistema de acesso de segurança**. Este sistema foi modelado originalmente com a linguagem **RON** (SIMÃO et al., 2016), sendo que este trabalho refez a modelagem utilizando a linguagem **RIMON**, que corresponde a uma evolução da RON. Esse sistema envolve a modelagem de *hardware* e *software* e partiu de 6 *requests* pré-existentes*,* tendo consumido em torno de 10 horas para ser construída pelo autor para os dois dialetos visuais (ver Tabela 48). Nesse experimento em particular, destaca-se a modelagem de atributos representando restrições, como por exemplo o “contador de tentativas

de acesso”. Essa restrição representa um dos diferenciais da linguagem RIMON, pois trata-se da modelagem de um requisito não-funcional por meio de informações visuais apresentadas diretamente nas interdependências do modelo (pré-condição com atributo).

O terceiro experimento se originou a partir das necessidades do cliente de um projeto real de um ***software* para automação residencial**, chamado de **MicroER**. Este sistema foi objeto da dissertação de mestrado de Souza (SOUZA, 2016), que apresentou um método de especificação de requisitos de *software* chamado de *Problem-Based SRS.* Neste experimento, as 11 **necessidades** do cliente foram modeladas em 12 horas pelo autor para os dois dialetos visuais (ver Tabela 48).

Os tempos apresentados na Tabela 48, pelo que foi apurado pelo autor, indicam que 40% cento do período foi gasto na construção do desenho em si, enquanto que 60% corresponde ao tempo de “pensar” em como realizar a modelagem, o que leva a crer que há espaço para otimização. Em tempo, a curva de aprendizagem de futuros utilizadores do método de modelagem deverá ser considerada para a utilização do método.

Dada a pequena quantidade de requisitos modelados e o tempo dispendido, é importante notar que a quantidade de *requests* modeladas foi pequena em comparação a projetos de sistemas ou de *software* de caráter comercial. Isso sinaliza a necessidade de uma experimentação que envolva dados de entrada em grande quantidade de forma a garantir a aplicabilidade do método e linguagem em projetos maiores. Para que o método de modelagem seja escalável para grandes quantidades de requisitos, no entanto, se faz necessária a criação de um mecanismo de notação que permita realizar o gerenciamento da complexidade visual crescente. O recurso notacional típico sugerido para gerenciar a escalabilidade do tamanho dos diagramas é o encapsulamento (*packages*). Além disso, a modelagem manual de requisitos em grandes quantidades pode ser inviável, caso não seja criada uma ferramenta de *software* específica e adequada para tal finalidade.

Por fim, na opinião do autor, a modelagem foi um sucesso do ponto de vista visual, pois a possibilidade de visualizar de forma clara a topologia dos inter-relacionamentos entre requisitos foi alcançada durante os experimentos.

6 VERIFICAÇÕES E ANÁLISES

Este capítulo tem por objetivo realizar verificações e análises com base nos desenvolvimentos e resultados produzidos anteriormente neste trabalho. A estrutura do capítulo é apresentada na Figura 67.

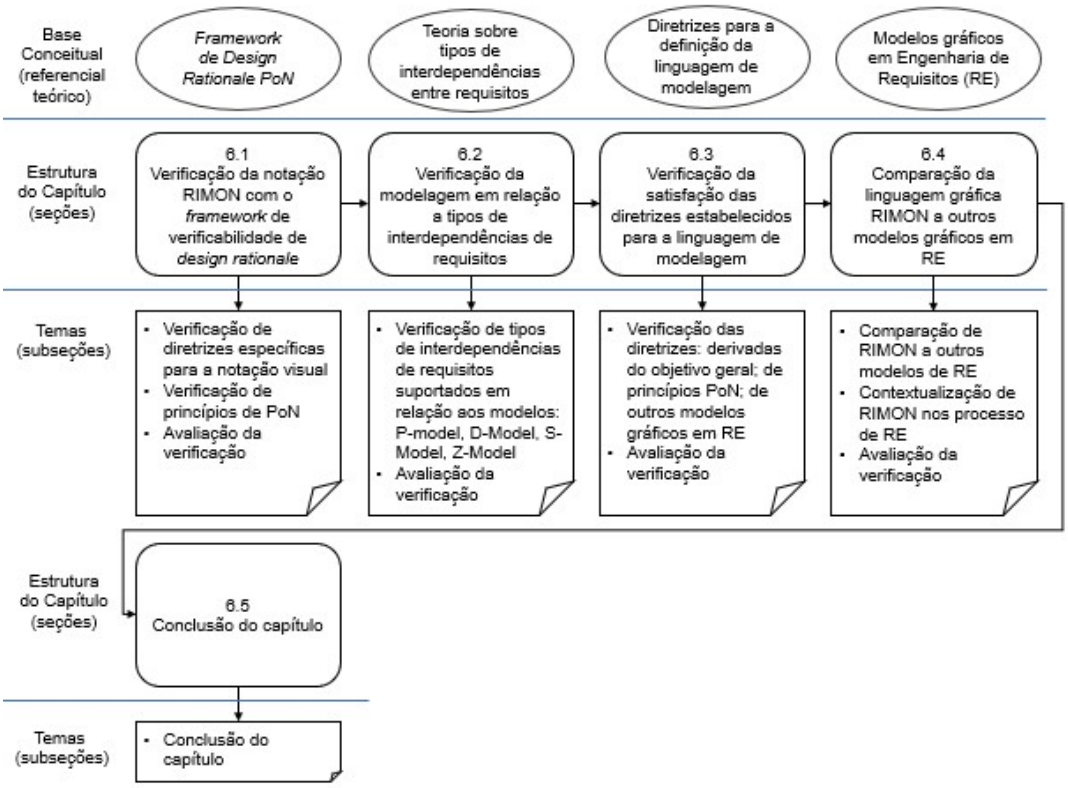


Figura 67 – Mapa conceitual/estrutural: Capítulo 6
Fonte: o autor

6.1 VERIFICAÇÃO DA NOTAÇÃO RIMON COM O *FRAMEWORK* DE VERIFICABILIDADE BASEADO EM *DESIGN RATIONALE*

Esta seção apresenta a verificação da notação visual RIMON por meio da utilização do *framework* de verificabilidade baseado em *design rationale* (LINDEN et. al, 2017). Esta verificação consiste no preenchimento de um conjunto de tabelas de verificação (conforme seção 2.2.5) conforme informações de *design rationale* registradas durante o desenvolvimento da notação visual (seções 3.4.1, 3.4.2 e 3.4.3). A verificação é realizada para cada um dos dialetos visuais definidos: técnico e de negócios. Além de uma verificação inicial de diretrizes específicas, cada princípio PoN aplicável é individualmente verificado.

6.1.1 Verificação de diretrizes específicas

A verificação de diretrizes especificas consiste em checar se há diretrizes ou requisitos específicos para a notação, se as construções semânticas, símbolos gráficos, regras de composição e mapeamento semântico foram realizados, e quais as variáveis visuais foram utilizadas pela notação gráfica. A verificação é apresentada na Tabela 49 para o dialeto técnico e na Tabela 50 para o dialeto de negócios

Notação – dialeto técnico
Assegurar-se de que as seguintes informações sejam relatadas.
Diretrizes (requisitos): **sim (seção 3.2)**
S: **sim, construções semânticas definidas na seção 3.3.2.**
V: **sim, símbolos gráficos definidos nas seções 3.4.1 e 3.4.3.**
M: S→V: **sim, mapeamento semântico definido na seção 3.5.**
VisVar: **sim, variáveis visuais utilizadas {Forma, textura, orientação, posição horizontal (x), posição vertical (y)}, conforme seções 3.4.1 e 3.4.3.**
R: **sim, regras de composição definidas na seção 3.4.4.**

Tabela 49 – Verificação de requisitos da notação RIMON – dialeto técnico
Fonte: o autor com base em (LINDEN et al., 2017)

Notação – dialeto de negócios
Assegurar-se de que as seguintes informações sejam relatadas.
Requisitos: **sim (diretrizes na seção 3.2)**
S: **sim, construções semânticas definidas na seção 3.3.2.**
V: **sim, símbolos gráficos definidos nas seções 3.4.2 e 3.4.3.**
M: S→V: **sim, mapeamento semântico definido na seção 3.5.**
VisVar: **sim, variáveis visuais utilizadas {Forma, Cor, textura, orientação, posição horizontal (x), posição vertical (y)}, conforme seções 3.4.2 e 3.4.3.**
R: **sim, regras de composição definidas na seção 3.4.4.**

Tabela 50 – Verificação de requisitos da notação RIMON – dialeto de negócios
Fonte: o autor com base em (LINDEN et al., 2017)

As tabelas apresentadas acima permitem verificar que as diretrizes específicas da linguagem foram definidas de forma satisfatória, bem como construções semânticas, símbolos gráficos, regras de composição, mapeamento semântico. Além disso, foram identificadas as variáveis visuais utilizadas.

6.1.2 Verificação D.3.1: Clareza semiótica

A verificação do princípio de PoN de clareza semiótica visa determinar se há redundância, sobrecarga, excesso ou déficit de símbolos. A verificação é apresentada na Tabela 51 (dialeto técnico) e na Tabela 52 (dialeto de negócios).

Clareza semiótica – dialeto técnico		
Calcule o valor do símbolo…		
… redundância =	$\lvert\{v \in V \mid \lvert\{s \in S: M(s) = v\}\rvert > 1\}\rvert$	**Valor Calculado = 0**
... sobrecarga =	$\lvert\{s \in S \mid \lvert\{v \in V: M(s) = v\}\rvert > 1\}\rvert$	**Valor Calculado = 0**
... excesso =	$\lvert\{v \in V \mid \neg\exists s \in S: M(s) = v\}\rvert$	**Valor Calculado = 0**
... déficit =	$\lvert\{s \in S \mid \neg\exists v \in V: M(s) = v\}\rvert$	**Valor Calculado = 0**
Para qualquer valor diferente de zero: Fornecer um *design rationale* estruturado (segundo Tabela 3). **Não Aplicável.**		

Tabela 51 – Verificação PoN: Clareza semiótica – dialeto técnico
Fonte: o autor com base em (LINDEN et al., 2017)

A Tabela 51 permite constatar que não há redundância, sobrecarga, excesso ou déficit de símbolos, pois os valores calculados são iguais a zero. Esse é um resultado satisfatório e esperado para a linguagem que foi definida.

Clareza Semiótica – dialeto de negócios		
Calcule o valor do símbolo…		
… redundância =	$\lvert\{v \in V \mid \lvert\{s \in S: M(s) = v\}\rvert > 1\}\rvert$	**Valor Calculado = 0***
... sobrecarga =	$\lvert\{s \in S \mid \lvert\{v \in V: M(s) = v\}\rvert > 1\}\rvert$	**Valor Calculado = 0**
... excesso =	$\lvert\{v \in V \mid \neg\exists s \in S: M(s) = v\}\rvert$	**Valor Calculado = 0**
... déficit =	$\lvert\{s \in S \mid \neg\exists v \in V: M(s) = v\}\rvert$	**Valor Calculado = 0**
Para qualquer valor diferente de zero: Fornecer um *design rationale* estruturado (segundo Tabela 3).		
OBS (0*): O construtor semântico de **entidade** é conceitualmente representado por um símbolo visual equivalente a um ícone, que pode assumir qualquer representação conforme as necessidades de negócio. Essa definição é intencional e está definida no *design rationale* da Tabela 31.		

Tabela 52 – Verificação PoN: Clareza semiótica – dialeto de negócios
Fonte: o autor com base em (LINDEN et al., 2017)

A Tabela 52 permite constatar que não há sobrecarga, excesso ou déficit de símbolos, pois os valores calculados são iguais a zero. Em relação a redundância, a linguagem RIMON intencionalmente permite que diferentes símbolos sejam utilizados para representar entidades (objetos do mundo real). Dessa forma, foi definido um *design* rationale específico que permite a utilização

de múltiplos símbolos para o construtor semântico de **entidade**. A existência deste *design rationale* demonstra que a definição da linguagem RIMON prevê esta situação, tornando o atendimento ao princípio de clareza semiótica satisfatório.

6.1.3 Verificação D.3.2: Discriminalidade perceptiva

O princípio PoN de discriminalidade perceptiva estabelece que "símbolos devem ser claramente distinguíveis" um dos outros (MOODY et al., 2010). Ao utilizar o *framework* de verificabilidade (Tabela 54 e Tabela 56) para este princípio, as seguintes questões técnicas relativas a área de processamento de imagens foram colocadas:

a) Qual é a métrica de similaridade que deve ser adotada para julgar a similaridade entre os símbolos gráficos?

   R: A métrica escolhida é o Índice de Similaridade Estrutural (*Structural Similarity Index* - SSIM), que é parte de um trabalho bem conhecido da área de processamento de imagens com mais de 10.000 citações na IEEE (WANG et al., 2004). O SSIM se baseia na computação multiplicativa de três termos: luminância, contraste e estrutura (WANG et al., 2004).

b) Como operacionalizar tecnicamente o cálculo da métrica?

   R: A ferramenta utilizada foi o MATLAB® da empresa *Mathworks ®,* que implementa o cálculo do SSIM em seu módulo de processamento de imagens (MATHWORKS, 2018).

c) Escolhida uma métrica, qual é o critério a ser adotado que determina que um símbolo é "claramente distinguível" de outro?

   R: Esta é uma das questões cuja resposta é da mais complexas e estudadas na área de processamento de imagens, pois não há uma única forma estabelecida e recomendada para se definir um valor limite (*threshold*) a partir do qual um símbolo é "claramente distinguível" de outro, como sugere o princípio de PoN. A definição do *threshold* depende da aplicação e das condições das imagens, sendo que há várias técnicas de diferentes níveis de complexidade passíveis de serem aplicadas para

seu cálculo e determinação (OTSU, 1979) (KAPUR et al., 1985) (TSAI, 1985) (ABUTALEB, 1989) (LU et al., 2011).

Neste trabalho, as imagens incluem símbolos gráficos sintéticos, pictóricos consistindo de formas geométricas regulares e de baixa complexidade gráfica, em comparação a fotografias e imagens do mundo real. Nesse sentido, a escolha e aplicação de métodos complexos para determinação de um limite de *threshold* está além dos limites do presente trabalho. Optou-se, portanto, por arbitrar um limite de dissimilaridade, o que é comum em trabalhos de processamento de imagens para aplicações de baixa complexidade.

A Tabela 53 apresenta os cálculos de SSIM para o **dialeto técnico** realizados para três diferentes classes de imagens (símbolos, interdependências e interconexões), originados a partir da notação unificada proposta neste trabalho (seção 3.6). Os cálculos foram realizados no módulo de processamento de imagens da ferramenta MATLAB® da empresa *Mathworks*®.

SSIM (S,S') Símbolos	Entity	
Requirement	0,7854	
SSIM (S,S') Interdependências	Premise	Inference
Premise		
Inference	0,6545	
Attribute	0,7793	0,5604
SSIM (S,S') Interconexões	AND	OR
AND		
OR	0,5824	
XOR	0,8038	0,5682

Tabela 53 – Cálculos de SSIM(S,S') para os símbolos do dialeto técnico
Fonte: o autor

A avaliação dos resultados da Tabela 53 e consiste em verificar se todos os valores calculados da função *Sim*(*s, s'*) são menores que o índice de dissimilaridade D=0,9 definido para comparação. É possível constatar que os símbolos definidos para o dialeto técnico são dissimilares segundo os cálculos realizados e critérios adotados, o que está dentro do esperado para a linguagem definida (Tabela 54).

Discriminalidade perceptiva – dialeto técnico
Escolher uma métrica *Sim* que para cada duas formas *s* e *s'* de construtores visuais *v e v'* retorna o seu score de similaridade: **Métrica: Structural Similarity Index (SSIM) – MATLAB Mathworks (MATHWORKS, 2018) (WANG et al., 2004)** Escolher um limite de dissimilaridade D. Se *Sim*(*s, s'*) < D → então *s* e *s'* não são semelhantes. **D = 0,9** Fornecer detalhes para formas *s* e s' de diferentes construções visuais *v* e *v'*: Semelhança de forma: Fornecer *Sim*(*s, s'*) = **Tabela 53.** Se *Sim*(*s, s'*) >= D: Fornecer um *design rationale* estruturado: **Não aplicável.** Inconsistência de forma: Se *s* é um subtipo de *s'* e *Sim*(*s,s'*) <D: Fornecer um *design rationale* estruturado: **Não aplicável.** Discriminabilidade dos relacionamentos: Se *v* e *v'* são relacionamentos e *Sim*(*s,s'*) >= D: Fornecer um *design rationale* estruturado: **Não aplicável.**

Tabela 54 – Verificação PoN: Discriminalidade perceptiva – dialeto técnico
Fonte: o autor com base em (LINDEN et al., 2017)

A Tabela 55 apresenta os cálculos de SSIM para o dialeto de negócios realizados para três diferentes classes de imagens (símbolos, interdependências e interconexões), também realizados na ferramenta MATLAB®.

SSIM (S,S') Símbolos	Requirement	Entity Actor	Entity Timer	Entity Sensor	Entity Comunication	Entity Interface	Entity Input	Entity Database	Entity Software	Entity Output
Requirement										
Entity Actor	0,3421									
Entity Timer	0,3326	0,8905								
Entity Sensor	0,3245	0,8484	0,8434							
Entity Comunnication	0,3216	0,8631	0,8586	0,8215						
Entity Interface	0,2906	0,7638	0,7378	0,7194	0,7355					
Entity Input	0,3306	0,8762	0,8888	0,8362	0,8512	0,7232				
Entity Database	0,3145	0,8423	0,8399	0,8015	0,8252	0,7081	0,8405			
Entity Software	0,2771	0,7214	0,7160	0,7131	0,7084	0,8261	0,7069	0,6969		
Entity Output	0,3306	0,8787	0,9080	0,8429	0,8585	0,7261	0,8848	0,8341	0,7153	
Entity Device	0,3355	0,8899	0,8987	0,8639	0,8637	0,7374	0,8897	0,8371	0,7082	0,8902

SSIM (S,S') Interdependências	Premise	Inference
Premise		
Inference	0,6545	
Attribute	0,7793	0,5604

SSIM (S,S') Interconexões	AND	OR
AND		
OR	0,4949	
XOR	0,3679	0,4869

Tabela 55 – Cálculos de SSIM(S,S') para os símbolos do dialeto de negócios
Fonte: o autor

A avaliação dos resultados da Tabela 55 consiste em verificar se todos os valores calculados da função *Sim*(*s, s'*) são menores que o índice de dissimilaridade D=0,9 definido para comparação. É possível constatar que os símbolos definidos para o dialeto técnico são dissimilares segundo os cálculos realizados e critérios adotados, o que está dentro do esperado para a linguagem definida (Tabela 56).

Discriminalidade perceptiva – dialeto de negócios
Escolher uma métrica *Sim* que para cada duas formas *s* e *s'* de construtores visuais *v e v'* retorna o seu score de similaridade:
Métrica: Structural Similarity Index (SSIM) – MATLAB Mathworks (MATHWORKS, 2018) (WANG et al., 2004)
Escolher um limite de dissimilaridade D. Se *Sim*(*s, s'*) < D → então *s* e *s'* não são semelhantes. **D = 0,9**
Fornecer detalhes para formas *s* e s' de diferentes construções visuais *v* e *v'*:
<u>Semelhança de forma</u>:
Fornecer *Sim*(*s, s'*) = **Tabela 55**
Se *Sim*(*s, s'*) >= D:
Fornecer um *design rationale* estruturado: **Não aplicável.**
<u>Inconsistência de forma</u>:
Se *s* é um subtipo de *s'* e *Sim*(*s,s'*) <D:
Fornecer um *design rationale* estruturado: **Não aplicável.**
<u>Discriminabilidade dos relacionamentos:</u>
Se *v* e *v'* são relacionamentos e *Sim*(*s,s'*) >= D:
Fornecer um *design rationale* estruturado: **Não aplicável.**

Tabela 56 – Verificação PoN: Discriminalidade perceptiva – dialeto de negócios
Fonte: o autor com base em (LINDEN et al., 2017)

Ao observar os índices de similaridade SSIM calculados na Tabela 53 e na Tabela 55, é possível constatar a inexistência de símbolos indistinguíveis que prejudicam o usuário durante o uso da notação, pois todos os valores calculados para a função *Sim*(*s,s'*) são inferiores à métrica D = 0,9. Desta forma, contata-se que a verificação realizada na Tabela 54 (dialeto técnico) e na Tabela 55 (dialeto de negócios) indica que o princípio PoN de discriminalidade perceptiva foi atendido.

6.1.4 Verificação D.3.3: Transparência semântica

A verificação do princípio de PoN de transparência semântica é apresentado na Tabela 57 (dialeto técnico) e na Tabela 58 (dialeto de negócios).

Transparência semântica – dialeto técnico
Para cada símbolo visual *v* ϵ V… **<u>Símbolos específicos do dialeto</u>:** **Requisito (*requirement)* - Tabela 27: Semanticamente Opaco.** **Entidade (*entity*) - Tabela 28: Semanticamente Opaco.** **Interconexões (*interconnections*) - Tabela 29: Semanticamente Opaco.** **<u>Símbolos comuns</u>:** **Premissa (*premise*) – Tabela 33: Semanticamente Opaco.** **Inferência (*inference*) – Tabela 34: Semanticamente Opaco.** **Atributo (*attribute*) – Tabela 35: Semanticamente Opaco.** Fornecer a localização de *v* na escala de transparência semântica da Figura 15 e fornecer um *design rationale* estruturado: ***Design rationale* fornecido nas seções 3.4.1 e 3.4.3.**

Tabela 57 – Verificação PoN: Transparência semântica – dialeto técnico
Fonte: o autor com base em (LINDEN et al., 2017)

A Tabela 57 (dialeto técnico) permite constatar que todos os símbolos são semanticamente opacos, o que significa que a relação entre forma e conteúdo dos símbolos é neutra (MOODY, 2009). Como o *design rationale* foi fornecido e foi evitado o problema de perversidade semântica (escala Figura 15), o princípio PoN de transparência semântica foi atendido para o dialeto técnico.

A Tabela 58 (dialeto de negócios) apresenta uma verificação em que é possível constatar que parte dos símbolos *(requirement, entity, interconnections*) foi definida com um grau maior de proximidade semântica (símbolos semanticamente translúcidos ou imediatos). Isso significa que a relação entre forma e conteúdo dos símbolos é próxima, com uma associação fortemente positiva (MOODY, 2009). Desta forma, no dialeto de negócios há símbolos com maior transparência semântica em comparação ao dialeto técnico. Como o *design rationale* foi fornecido e foi evitado o problema de perversidade semântica (escala Figura 15), o princípio PoN de transparência semântica também foi atendido para o dialeto de negócios.

Transparência semântica – dialeto de negócios
Para cada símbolo visual *v* ϵ V…
<u>Símbolos específicos do dialeto</u>:
Requisito (*requirement)* - Tabela 30: Semanticamente translúcido.
Entidade (*entity*) - Tabela 31: Semanticamente imediato.
Interconexões (*interconnections*) - Tabela 32: Semanticamente translúcido.
<u>Símbolos comuns</u>:
Premissa (*premise*) – Tabela 33: Semanticamente opaco.
Inferência (*inference*) – Tabela 34: Semanticamente opaco.
Atributo (*attribute*) – Tabela 35: Semanticamente opaco.
OBS: As definições acima (semanticamente opaco, translúcido, imediato) se basearam no referencial teórico estabelecido na seção 2.2.3.
Fornecer a localização de *v* na escala de transparência semântica da Figura 15 e fornecer um *design rationale* estruturado: ***Design rationale* fornecido nas seções 3.4.1 e 3.4.2.**

Tabela 58 – Verificação PoN: Transparência semântica – dialeto de negócios
Fonte: o autor com base em (LINDEN et al., 2017)

6.1.5 Verificação D.3.4: Expressividade visual

A verificação do princípio de expressividade visual é apresentado na Tabela 59 (dialeto técnico) e na Tabela 60 (dialeto de negócios). Segundo este princípio, quanto maior a quantidade de variáveis visuais utilizadas, maior é a expressividade visual, desde que não haja poluição visual (MOODY, 2010).

Expressividade visual – dialeto técnico
Fornecer | Variáveis Visuais | = 6 **variáveis {Forma, textura, orientação, brilho, posição horizontal (x), posição vertical (y)}.**
A quantidade de | Variáveis Visuais | = 6? **Não**
Para as variáveis visuais selecionadas *x* ϵ | Variáveis Visuais |:
Fornecer um *design rationale* estruturado, abordando até que ponto os as variáveis visuais x selecionadas contribuem para:
(i) Discriminalidade Perceptiva (ii) Legibilidade, e (iii) Estética: ***Design Rationale* fornecido nas seções 3.4.1 e 3.4.3.**

Tabela 59 – Verificação PoN: Expressividade visual – dialeto técnico
Fonte: o autor com base em (LINDEN et al., 2017)

A Tabela 59 (dialeto técnico) permite identificar as variáveis visuais utilizadas e faz referência ao *design rationale* que justifica esta definição, demonstrando o atendimento ao princípio de expressividade visual.

Expressividade visual – dialeto de negócios
Fornecer | Variáveis Visuais | = **7 variáveis {Forma, cor, brilho, textura, orientação, posição horizontal (x), posição vertical (y)}**
A quantidade de | Variáveis Visuais | = 8? **Não**
Para as variáveis visuais selecionadas *x* ϵ | Variáveis Visuais |:
Fornecer um *design rationale* estruturado, abordando até que ponto os as variáveis visuais x selecionadas contribuem para:
(i) Discriminalidade Perceptiva (ii) Legibilidade, e (iii) Estética: ***Design Rationale* fornecido nas seções 3.4.2. e 3.4.3.**

Tabela 60 – Verificação PoN: Expressividade visual – dialeto de negócios
Fonte: o autor com base em (LINDEN et al., 2017)

A Tabela 60 (dialeto de negócios) permite identificar as variáveis visuais utilizadas e faz referência ao *design rationale* que justifica esta definição, demonstrando o atendimento ao princípio de expressividade visual.

6.1.6 Verificação D.3.5: Codificação dual

A verificação do princípio de PoN de codificação dual é apresentado na Tabela 61 (dialeto técnico) e na Tabela 62 (dialeto de negócios).

Codificação dual – dialeto técnico
Aplicar a teoria de codificação dual de acordo com os requisitos da notação.
Fornecer um *design rationale* estruturado para a implementação e como os requisitos são endereçados: ***Design Rationale* fornecido nas seções 3.4.1 e 3.4.3.**

Tabela 61 – Verificação PoN: Codificação dual – dialeto técnico
Fonte: o autor com base em (LINDEN et al., 2017)

O atendimento ao princípio de codificação dual se verifica pela existência de um *design rationale* estruturado, que indique que a notação utiliza simultaneamente de texto e gráficos em seus símbolos. Tanto a Tabela 61 (dialeto técnico) quanto a Tabela 62 (dialeto de negócios) fazem esta referência, caracterizando, portanto, o atendimento ao princípio de codificação dual.

Codificação dual – dialeto de negócios
Aplicar a teoria de codificação dual de acordo com os requisitos da notação. Fornecer um *design rationale* estruturado para a implementação e como os requisitos são endereçados: **Design Rationale fornecido nas seções 3.4.2. e 3.4.3.**

Tabela 62 – Verificação PoN: Codificação dual – dialeto de negócios
Fonte: o autor com base em (LINDEN et al., 2017)

6.1.7 Verificação D.3.6: Economia gráfica

A verificação do princípio de PoN de economia gráfica é apresentado na Tabela 63 (dialeto técnico) e na Tabela 64 (dialeto de negócios).

Economia gráfica – dialeto técnico
Fornecer | V | = **6 símbolos {requisito, entidade, interconexão, premissa, inferência, atributo}** Se |*Tipos* | = 1 e | V | não é 7 ± 2: **Tipos de Diagramas = 1** Fornecer um *design rationale* estruturado: **Não aplicável** Se |*Tipos* | > 1, para cada tipo *t*: **Não aplicável.** Vt = [ ] (em que Vt é o conjunto V usado em diagramas do tipo t) Fornecer | Vt | para cada t ϵ Tipos: [ ] Se |Vt| não é 7 ± 2: Fornecer um *design rationale* estruturado: **Não aplicável.**

Tabela 63 – Verificação PoN: Economia gráfica – dialeto técnico
Fonte: o autor com base em (LINDEN et al., 2017)

A verificação do princípio de economia gráfica consiste em verificar se a quantidade de símbolos distintos definidos para a notação encontra-se dentro da faixa recomendada de 5 a 9 símbolos (7 ± 2). Tanto a Tabela 63 (dialeto técnico) quanto a Tabela 64 (dialeto de negócios) registram o uso de 6 símbolos (ou tipos de símbolos) distintos, sendo que neste caso é possível verificar o atendimento ao princípio de economia gráfica.

Economia gráfica – dialeto de negócios
Fornecer | V | = **6 símbolos {requisito, entidade, interconexão, premissa, inferência, atributo}**
Se |*Tipos* | = 1 e | V | não é 7 ± 2: **Tipos de Diagramas = 1**
Fornecer um *design rationale* estruturado: **Não aplicável.**
Se |*Tipos* | > 1, para cada tipo *t*: **Não aplicável.**
Vt = [ ] (em que Vt é o conjunto V usado em diagramas do tipo t)
Fornecer | Vt | para cada t ϵ Tipos: [ ]
Se |Vt| não é 7 ± 2:
Fornecer um *design rationale* estruturado: **Não aplicável.**

Tabela 64 – Verificação PoN: Economia gráfica – dialeto de negócios
Fonte: o autor com base em (LINDEN et al., 2017)

6.1.8 Verificação D.3.7: Ajuste Cognitivo

O princípio de PoN de ajuste cognitivo é verificado por meio da Tabela 65, para ambos os dialetos visuais da notação RIMON.

Ajuste cognitivo – dialetos técnico e de negócios
Fornecer dialetos visuais de acordo com os requisitos da notação.
Se relevante para os requisitos da notação, fornecer um dialeto para pelo menos diferentes níveis de especialização e diferentes meios de representação.
A notação conta com dois dialetos (seção 3.4):
Dialeto Técnico: visa permitir modelagem de requisitos de forma técnica, para atender a necessidades internas dos pesquisadores que trabalham com os fundamentos da notação visual.
Dialeto de Negócios: visa tornar atrativa e intuitiva a notação para fins de modelagem de negócios. Procura evitar a saturação visual e manter a economia gráfica.
Se nenhum dialeto for fornecido e seus requisitos indicarem usuários com diferentes níveis de especialização, ou se for possível usar em diferentes mídias de representação:
Fornecer um *design rationale* estruturado: **Não aplicável.**

Tabela 65 – Verificação PoN: Ajuste cognitivo – dialetos técnico e de negócios
Fonte: o autor com base em (LINDEN et al., 2017)

A existência de dois dialetos visuais, relatada na Tabela 65, sinaliza que a linguagem RIMON foi concebida de forma a atender diferentes audiências conforme preconizado pelo princípio de ajuste cognitivo. Desta forma, caracteriza-se o atendimento a este princípio de PoN.

6.1.9 Avaliação da verificação

A linguagem gráfica RIMON atende aos princípios PoN, segundo os critérios propostos no *framework* de verificabilidade baseado em *design rationale*. Inicialmente, foi verificado que as construções semânticas, símbolos gráficos, mapeamento semântico e regras de composição foram definidos adequadamente. Além disso, foram identificadas as variáveis visuais utilizadas e diretrizes específicas da notação (seção 6.1.1). Nas demais seções (6.1.2 a 6.1.8), cada princípio PoN foi verificado individualmente: clareza semiótica, discriminalidade perceptiva, transparência semântica, expressividade visual, codificação dual, economia gráfica e ajuste cognitivo.

A maioria dos princípios foi verificada por mera observação, exceto pelo de discriminalidade perceptiva. Este princípio demandou o uso de uma ferramenta de processamento de imagens (*Matlab*®), além da definição de um índice (SSIM) e uma métrica específica para realização da verificação.

6.2 VERIFICAÇÃO DA MODELAGEM RIMON EM RELAÇÃO A TIPOS DE INTERDEPENDÊNCIAS DE REQUISITOS

O objetivo desta seção é verificar quais tipos de interdependências de requisitos podem ser representados com a utilização da linguagem gráfica RIMON proposta neste trabalho. A verificação se baseará nos modelos de classificação de interdependências identificados na seção 2.3.8: P-Model, D-Model, S-Model, Z-Model.

6.2.1 P-Model

A Tabela 66 mostra a representação tabular de tipos de interdependências em dois níveis segundo o modelo **P-Model** (POHL, 1996).

P-Model (POHL, 1996)		RIMON suporta este tipo de interdependência?
Nível 1	Nível 2	
Condição		
|==>	Restrição	Sim
|==>	Pré-condição	Sim
Conteúdo		
|==>	Similar	Não
|==>	Compara-se	Não
|==>	Contradiz	Não
|==>	Conflita	Sim
Documentos		
|==>	exemplo_para	Não
|==>	caso_de_teste_para	Não
|==>	propósito	Não
|==>	comentário	Não
|==>	embasamento	Não
Evolução		
|==>	baseado_em	Não
|==>	satisfaz	Não
|==>	formaliza	Não
|==>	elabora	Não
|==>	substitui	Não
Abstração		
|==>	refina	Não
|==>	gera	Não

Tabela 66 – Interdependências da modelagem RIMON classificadas em P-Model
Fonte: o autor

Nesta tabela, foi assinalado que a linguagem RIMON suporta os tipos de interdependências de condição “Restrição” e “Pré-condição” e de conteúdo “Conflita” do P-Model. A definição da linguagem RIMON, segundo a classificação P-Model, não suporta os tipos de interdependências relacionados a “Documentos”, “Evolução” e “Abstração”. Isto está de acordo com os objetivos de modelagem definidos para a RIMON até o presente momento.

6.2.2 D-Model

A Tabela 67 apresenta a representação tabular de tipos de interdependências em dois níveis proposta no modelo **D-Model** (DAHLSTEDT e PERSSON, 2005).

D-Model (DAHLSTEDT e PERSSON, 2005)		RIMON suporta este tipo de interdependência?
Nível 1	Nível 2	
Interdependências Estruturais		
|==>	Refinado_para	Não
|==>	Modificado_para	Não
|==>	Similar_a	Não
Interdependências Restritivas		
|==>	Requer	Sim
|==>	Conflita_com	Sim
Interdependências de custo/valor		
|==>	incrementa_custo_de/ decrementa_custo_de	Não
|==>	incrementa_valor_de/ decrementa_valor_de	Não

Tabela 67 – Interdependências da modelagem RIMON classificadas em D-Model
Fonte: o autor

No D-Model, a linguagem RIMON suporta os tipos de interdependências restritivas “Requer” e “Conflita_com”. Os tipos de Interdependências “Estruturais” e de “Custo/Valor” do D-Model não são suportados.

6.2.3 S-Model

A Tabela 68 mostra a representação tabular de um nível do modelo **S-Model** (SPIJKERMAN, 2010).

S-Model (SPIJKERMAN, 2010)	RIMON suporta este tipo de interdependência?
Nível 1	
Relacionamentos	
Contém	Não
Refina	Não
Refina parcialmente	Não
Requer	Sim
Conflita	Sim

Tabela 68 – Interdependências da modelagem RIMON classificadas em S-Model
Fonte: o autor

No S-Model, RIMON suporta os tipos de relacionamentos (*i.e.:* interdependências) “Requer” e “Conflita”. Os demais tipos de interdependências do modelo S-Model (contém, refina e refina parcialmente) não são suportados.

6.2.4 Z-Model

A Tabela 69 apresenta a representação de tipos de interdependências em três níveis proposta no modelo **Z-Model** (ZHANG et al., 2014).

Z-Model (ZHANG et al., 2014)			RIMON suporta este tipo de interdependência?
Nível 1	Nível 2	Nível 3	
Dependência Intrínseca			
|==>	Negócio		Não
	|==>	restringe	Sim
	|==>	precede	Sim
|==>	Implementação		Não
	|==>	é_similar_a	Não
	|==>	é_exceção_de	Não
	|==>	conflita	Sim
|==>	Structure		Não
	|==>	refina	Não
|==>	Evolution		Não
	|==>	evolui_para	Não
Dependência Adicional			Não
|==>	Valor		Não
	|==>	incrementa_custo_de	Não
	|==>	decrementa_custo_de	Não
|==>	Custo		Não
	|==>	incrementa_valor_de	Não
	|==>	decrementa_valor_de	Não

Tabela 69 – Interdependências da modelagem RIMON classificadas em Z-Model
Fonte: o autor

No Z-Model, RIMON suporta tanto os tipos de dependências intrínsecas de negócio “restringe” e “precede”, quanto o tipo de dependência intrínseca de implementação “conflita”. Os demais tipos propostos no Z-model não são suportados.

6.2.5 Avaliação da verificação

Esta seção apresentou uma verificação de quais tipos de interdependências de requisitos podem ser representados com a utilização da linguagem gráfica RIMON. Esta verificação se baseou nos seguintes modelos de classificação de tipos de interdependências de requisitos propostos na literatura: P-Model, D-Model, S-Model, Z-Model.

A linguagem RIMON proposta neste trabalho permite a representação dos seguintes tipos de interdependências, agrupados por similaridade semântica:

a) “Pré-condição” (P-Model), “Requer” (D-Model), “Requer” (S-Model), “Precede” (Z-Model);
b) “Restrição” (P-Model), “Restringe” (Z-Model);
c) “Conflita” (P-Model), “Conflita_com” (D-Model), “Conflita” (S-Model), “Conflita” (Z-Model).

Como limitação, nota-se que a linguagem RIMON proposta neste trabalho **não** suporta tipos de interdependências de categorias como “Documentação”, “Evolução”, “Abstração”, “Refinamento” e similares. Estes tipos (classes) de interdependências geralmente são associados à rastreabilidade, análise e refinamento de requisitos em projetos de *software* e sistemas.

A linguagem RIMON proposta na presente dissertação tem uma definição focada em aspectos **estruturais** relacionados a representação de requisitos e suas interdependências. Os tipos de interdependências que não foram incluídos na presente proposta poderão fazer parte de uma futura evolução da notação, sendo que para tanto é necessário realizar estudos específicos relacionados a integração desta notação às outras etapas de projetos de *software* e sistemas, o que no momento está fora do escopo do presente trabalho.

6.3 VERIFICAÇÃO DA SATISFAÇÃO DAS DIRETRIZES PROPOSTAS PARA A LINGUAGEM RIMON

Esta seção tem por objetivo realizar a verificação das diretrizes estabelecidas para o desenvolvimento da linguagem RIMON, propostas na

seção 3.2 desta dissertação (Tabela 26). As próximas subseções se referem a estas diretrizes.

6.3.1 D.1.1: modelar requisitos de *software*

Para verificar a possibilidade de modelar requisitos de ***software***, foi realizado neste trabalho uma experimentação de modelagem com base nos requisitos de um projeto real chamado MICROER (seção 5.3). Essa modelagem foi realizada a partir das necessidades do cliente (*customer needs*) nos dois dialetos visuais disponibilizados pela notação RIMON: dialeto técnico e dialeto de negócios.

6.3.2 D.1.2: modelar requisitos de sistemas

Para verificar a possibilidade de modelar requisitos de **sistemas**, foram realizados neste trabalho duas experimentações de modelagem: um sistema purificador de água (seção 5.1), no qual *software* não é necessário (sistema físico somente), e um sistema de segurança de acesso (seção 5.2). Ambas as modelagens foram realizadas nos dois dialetos visuais disponibilizados pela notação RIMON: notação técnica e notação de negócios.

6.3.3 D.2.1: modelar requisitos e interdependências de forma sistemática

Segundo o dicionário Michaelis, o termo **sistemático** pode ser entendido como algo que “se processa metodicamente” (MICHAELIS DICIONÁRIO, 2018) e, portanto, que segue um **método**. Um método ou metodologia consiste em uma “especificação de processo a ser seguido juntamente com os produtos de trabalho a serem utilizados e gerados, além da consideração das pessoas e ferramentas envolvidas, durante um esforço de desenvolvimento de um domínio baseado em informações” (ISO/IEC 24744, 2007).

Nesse sentido, para permitir que os requisitos possam ser modelados de forma sistemática, o presente trabalho apresentou um **método** para a aplicação da linguagem gráfica RIMON (Capítulo 3.7). O método especificado consiste em um processo (fluxo) composto de uma série de atividades a serem executadas, inclusive com atividades específicas de modelagem de requisitos e de suas interdependências.

6.3.4 D.2.2: modelar requisitos e interdependências de forma precisa

A existência de um método para a aplicação da linguagem gráfica RIMON (Capítulo 3) é uma condição necessária, porém insuficiente, para garantir que os requisitos e suas interdependências sejam modelados de forma precisa. Segundo o dicionário Michaelis, o termo **preciso** pode ser definido como sendo algo “realizado com exatidão ou rigor” e que “não ofereça dúvida, seja claro e evidente” (MICHAELIS DICIONÁRIO, 2018). Neste trabalho, é possível apontar indícios de que a linguagem gráfica RIMON, pela forma como foi definida, provê elementos e mecanismos que permitem uma maior precisão na modelagem gráfica (visual) dos requisitos e interdependências.

De um ponto de vista **semântico**, pode-se apontar que a existência de uma definição clara de um metamodelo e dos elementos semânticos da linguagem (seções 3.3.1 e 3.3.2), permite prover uma maior exatidão (rigor) semântica aos usuários durante a escolha dos símbolos a serem utilizados na modelagem. Adicionalmente, a existência de regras de composição claras para o estabelecimento de diagramas contribui para a precisão na representação das interdependências (seção 3.4.4).

De um ponto de vista **visual**, é possível apontar que a linguagem RIMON, ao atender aos princípios PoN conforme verificado na seção 6.1, apresenta características visuais atreladas a tais princípios que contribuem para uma maior **precisão** na modelagem, tais como:

a) **Clareza semiótica**: o fato de haver uma correspondência precisa de 1:1 entre construções semânticas e símbolos gráficos contribui para incrementar a precisão visual da linguagem RIMON;
b) **Codificação dual**: o possível uso simultâneo de elementos textuais para complementar os símbolos gráficos possibilita incrementar a precisão da representação;
c) Economia gráfica: a definição parcimoniosa de símbolos gráficos contribui para evitar a saturação visual e uma eventual perda da clareza da representação diagramática da linguagem.

6.3.5 D.2.3: modelar requisitos e interdependências de forma expressiva

Segundo o dicionário Michaelis, o termo **expressivo** pode ser entendido como algo “de boa comunicação; convincente ao transmitir uma ideia” (MICHAELIS DICIONÁRIO, 2018). Uma linguagem gráfica com bom poder de expressividade deve, portanto, oferecer recursos semânticos e visuais que permitam exercer uma boa comunicação entre usuários da notação (criadores e leitores de diagramas).

De um ponto de vista **semântico**, pode-se apontar que a linguagem RIMON permite a representação de tipos de interdependências relacionados a pré-condições, restrições e conflitos, conforme verificado na seção 6.2. Entende-se que boa parte da expressividade da linguagem gráfica RIMON reside nestas representações. Particularmente, destaca-se o fato de que a representação de restrições (condições lógico-matemáticas) nas interdependências dos diagramas é uma característica única que traz grande expressividade (poder de comunicação) para a linguagem em relação a outros modelos gráficos em RE (ver seção 6.4).

De um ponto de vista **visual**, é possível apontar que a linguagem RIMON, ao obedecer aos princípios PoN conforme verificado na seção 6.1, apresenta características visuais atreladas a tais princípios que contribuem para uma maior **expressividade** na modelagem, tais como:

a) **Transparência semântica**: o uso de símbolos cuja aparência sugere seu significado (*e.g.*: entidades como ícones no dialeto de negócios da linguagem) incrementa a expressividade visual da linguagem RIMON;
b) **Expressividade visual**: a utilização de múltiplas variáveis visuais contribuem para a expressividade da linguagem RIMON.

6.3.6 D.3: satisfazer princípios de PoN para notações visuais

A diretriz D.3 estabelece a necessidade de satisfação de alguns dos princípios de PoN para o desenvolvimento de notações visuais. Estes princípios foram individualmente verificados nas subseções da seção 6.1 desta dissertação, da seguinte forma:

a) D.3.1 Clareza semiótica: seção 6.1.2;
b) D.3.2 Discriminabilidade perceptiva: seção 0;

c) D.3.3 Transparência semântica: seção 6.1.4;
d) D.3.4 Expressividade visual: seção 6.1.5;
e) D.3.5 Codificação dual: seção 6.1.6;
f) D.3.6 Economia gráfica: seção 6.1.7;
g) D.3.7 Ajuste cognitivo: seção 6.1.8.

6.3.7 D.4: representar FRs e NFRs por meio de um diagrama de requisitos

A representação de requisitos funcionais (*Functional Requirements* - FR) e não funcionais (*Non-Functional Requirements* - NFR) foi realizada por meio de experimentos apresentados no capítulo 5. Por definição, requisito funcional é "um requisito que especifica uma função que um sistema ou componente do sistema deve ser capaz de executar" (ISO / IEC / IEEE 24765, 2010). Requisitos não-funcionais são "restrições nos serviços ou funções oferecidas pelo sistema. Elas incluem restrições temporais, restrições no processo de desenvolvimento e restrições impostas por normas (padrões)" (SOMMERVILLE, 2010).

a) **Requisitos Funcionais**: todos os três experimentos apresentados contêm requisitos funcionais em seus diagramas de requisitos (seções 5.1.3, 5.2.2 e 5.3.6);
b) **Requisitos Não-Funcionais**: neste trabalho, os requisitos não funcionais SS11-c, SS11-d e SS11-e correspondem a restrições do sistema de acesso de segurança e foram modelados nos diagramas da Figura 60 e da Figura 61 da seção 5.2.2. Os NFRs são modelados como atributos nas pré-condições (interdependências) do diagrama de requisitos.

6.3.8 D.5: identificação visual de conflitos em FRs e NFRs.

Para possibilitar a identificação visual de conflitos em FRs e NFRs, a linguagem gráfica RIMON inclui um símbolo específico para representar um "conflito potencial" (*potential conflict*) apresentado na Figura 68 sob a letra "a".

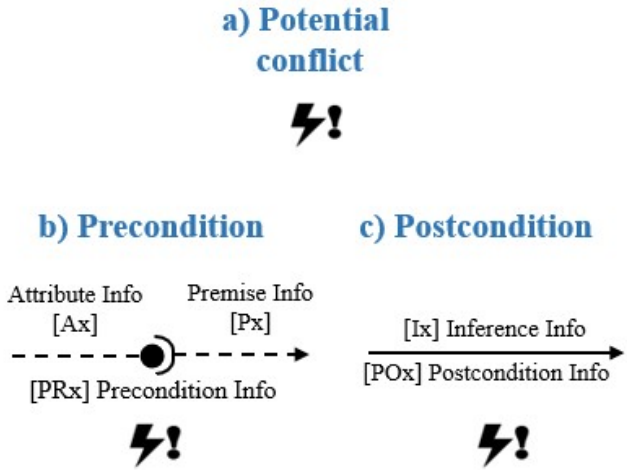


Figura 68 – Símbolo de conflito potencial para uso em interdependências
Fonte: o autor

Na mesma figura, é apresentada a sintaxe visual do símbolo de conflito potencial em termos de sua utilização para representar um conflito potencial em uma pré-condição (letra “b”) e em uma pós-condição (letra “c”). Conforme verificação realizada na subseção 6.3.7, uma pré-condição com um atributo pode ser utilizada para representar um NFR, tornando possível então que conflitos em NFRs possam ser identificados com a adição do símbolo de conflito potencial. Da mesma forma e pela mesma verificação da subseção 6.3.7, pré-condições e pós-condições podem ser associadas a FRs e, portanto, ser passíveis de colocação do mesmo símbolo para identificação de conflitos.

6.3.9 Avaliação e sumarização da verificação

Nesta seção foi realizada a verificação de todas as diretrizes propostas para o desenvolvimento da linguagem gráfica RIMON na seção 3.2. Foi verificado que houve satisfação das diretrizes derivadas do objetivo geral (D.1, D.2 e derivadas), diretrizes relacionadas a PoN (D.3 e derivadas) e diretrizes específicas escolhidas a partir de outros modelos gráficos em RE (D4 e D5).

A Tabela 70 apresenta uma sumarização desta verificação realizada nas seções anteriores. Nesta tabela, é feita a correlação das diretrizes propostas para RIMON com as seções e capítulos em que é feita a verificação de cada diretriz. Também foi incluída na tabela uma breve descrição de como o resultado de cada verificação foi alcançado.

Diretriz		Verificação	Resultado
D.1.1	Modelar requisitos de *software*	Seção 5.3	Verificado por meio de experimento de modelagem de requisitos de *software* microER.
D.1.2	Modelar requisitos de sistemas	Seção 5.1 Seção 5.2	Verificado por meio de experimentos de modelagem de requisitos de sistema purificador de água e de sistema de acesso de segurança.
D.2.1	Modelagem sistemática	Capítulo 4 Capítulo 5	Verificado por meio da definição de um método de modelagem e pela sua aplicação nos experimentos desta dissertação.
D.2.2	Modelagem precisa	Capítulo 3 Seção 6.1	Verificado por meio de uma definição da linguagem de modelagem gráfica RIMON e de sua aplicação do *framework* de verificabilidade baseado em *Design Rationale.*
D.2.3	Modelagem expressiva	Seção 6.1 Seção 6.2	Verificado por meio da aplicação do *framework* de verificabilidade baseado em *Design Rationale*, além da verificação de tipos de interdependências suportados pela linguagem.
D.3	Satisfazer princípios de PoN	Seção 6.1	Verificado por meio da aplicação do *framework* de verificabilidade baseado em *Design Rationale*.
D.4	Representar FRs e NFRs	Capítulo 5	Verificado por meio da aplicação da linguagem RIMON nos experimentos desta dissertação.
D.5	Permitir identificar conflitos	Capítulo 3 Capítulo 5	Verificado por meio de uma definição da linguagem de modelagem gráfica RIMON e por meio de sua aplicação nos experimentos desta dissertação.

Tabela 70 – Sumário da verificação das diretrizes de RIMON
Fonte: o autor

6.4 COMPARAÇÃO DA LINGUAGEM GRÁFICA RIMON COM OUTROS MODELOS GRÁFICOS EM RE

Esta seção apresenta uma comparação da linguagem gráfica RIMON com outros modelos gráficos, de forma a elucidar qual a contribuição oferecida por esta linguagem gráfica para a área de RE.

6.4.1 RIMON comparada a outros modelos em termos de características

A Tabela 71 apresenta uma a comparação de RIMON com outros modelos gráficos em RE (seção 2.3). Esta tabela corresponde a uma extensão da Tabela 22, apresentada e descrita detalhadamente na seção 2.3.8 do referencial teórico. Por esse motivo, esta tabela não será descrita novamente nesta subseção, cujo foco será o de apresentar as diferenças resultantes da introdução da linguagem gráfica RIMON na supracitada tabela.

Ao avaliar a Tabela 71, é possível notar que a característica C11 é exclusiva das notações RON e RIMON e se constitui no principal diferencial destas notações. A capacidade de representar condições lógico-matemáticas diretamente nas interdependências de requisitos em um modelo gráfico de requisitos é exemplificada na Figura 69, que foi extraída do diagrama de modelagem do “sistema de segurança” da seção 5.2.2 (Figura 60).

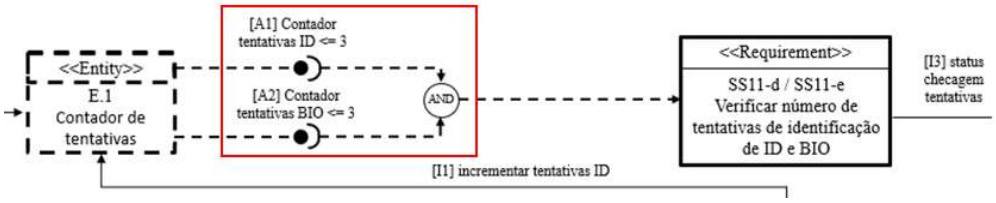


Figura 69 – Exemplo: representação RIMON de condição lógico-matemática
Fonte: o autor

A parte destacada em vermelho na Figura 69 apresenta uma condição lógico-matemática relacionada à pré-condição do requisito “Verificar número de tentativas de identificação ID (cartão de acesso) e BIO (biometria)”. Neste caso, o número de tentativas de identificação deve ser inferior a 3 para ambos os tipos de acesso, o que é representado por uma conjunção (“AND”) na interdependência do diagrama.

		C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18
		Contexto		Características de modelagem (intra-diagrama)											Características gerais do sistema de modelagem				
N	Nome (sigla)	Modelar *Early Req.*	Modelar *Late Req.*	Modelar problemas	Modelar objetivos	Modelar cenários	Modelar FRs	Modelar NFRs	Modelar conflitos	Modelar refinamentos	Modelar restrições (geral)	Modelar restrições como dependências	Possuir mecanismo de escalabilidade	Possuir mecanismo de rastreabilidade	Ter linguagem Formal	Ter ferramenta	Ter norma (padrão)	Ter metamodelo	Seguir princípios PoN
01	i*	X			X				X	X	X				X	X		X	
02	Tropos	X	X		X				X	X	X			X	X	X		X	
03	KAOS	X	X		X		X	X	X	X	X			X	X	X		X	
04	NFR	X	X		X			X	X	X	X				X	X		X	
05	URN	X	X		X	X			X	X	X		X	X	X	X	X	X	
06	SysML		X				X	X		X	X			X	X	X	X	X	
07	UML UC		X			X				X					X	X	X	X	
08	UCM		X			X			X		X		X			X	X	X	
09	PF	X		X					X	X	X				X	X			
10	V-Graph	X			X				X	X					X	X			
11	RecVisu		X				X									X			
12	Techne		X		X				X		X				X	X		X	
13	AMORE	X				X										X			
14	URML		X		X		X	X	X		X					X		X	X
15	UML RD		X				X	X								X			
16	VLML	X	X		X		X	X											
17	OPM		X				X	X								X			
18	RDN		X				X				X								
19	AGORA	X			X				X	X								X	
20	ATHENA		X				X									X			
21	Archimate	X	X		X		X				X					X			
22	IRD		X				X				X				X				
23	RCAT		X				X		X		X				X	X			
24	RIG		X				X		X	X									
25	RDG		X				X						X			X			
26	VRDL	X				X													
27	RON		X				X	X			X	X							
28	**RIMON**		**X**				**X**	**X**	**X**		**X**	**X**						**X**	**X**

Tabela 71 – RIMON comparada a modelos gráficos em termos de características
Fonte: o autor

Outro diferencial é que somente URML e RIMON foram projetadas com base nos princípios de física de notações (PoN). Neste caso, é importante destacar que URML foi definida como sendo uma linguagem de modelagem para uma finalidade específica (aplicação industrial pontual), enquanto RIMON foi desenvolvida para ser uma linguagem de uso geral para modelagem de quaisquer *software* e sistemas.

Por fim, na Tabela 71 é possível comparar a linguagem gráfica RIMON e a abordagem RON que a originou, por meio das seguintes características definidas evolutivamente para a linguagem RIMON:

a) Possui um metamodelo definido (C17);
b) Foi desenvolvida segundo os princípios de PoN (C18);
c) Permite realizar a identificação e modelagem de conflitos (C8).

Assim, a linguagem RIMON corresponde a uma evolução da abordagem RON, por implementar novas características destinadas a modelar os requisitos de forma sistemática, precisa e expressiva. Por outro lado é possível destacar que, assim como a maioria das abordagens caracterizadas na Tabela 71, RIMON ainda **não** dispõe de mecanismos para gerenciar a complexidade de diagramas (C12), mecanismo de rastreabilidade (C13), mecanismo para suportar refinamentos (C9), ou mesmo ferramenta de *software* (C15). Tais características certamente oferecem uma oportunidade de pesquisa em futuros trabalhos de evolução da linguagem.

6.4.2 RIMON em processos e contexto de RE

Em termos de etapas de processos de RE e contexto (explicados na seção 2.1.4), o modelo gráfico proposto na presente dissertação foi incluído na Figura 70 (seta sobre “MODELO PROPOSTO”), por ser aplicável ao contexto de ***late requirements*** e poder ser utilizado nas etapas de especificação e análise de requisitos. A abordagem do qual este modelo faz parte poderá ser modificada futuramente de forma a contemplar mecanismos de validação e verificação, ampliando o seu escopo de atuação.

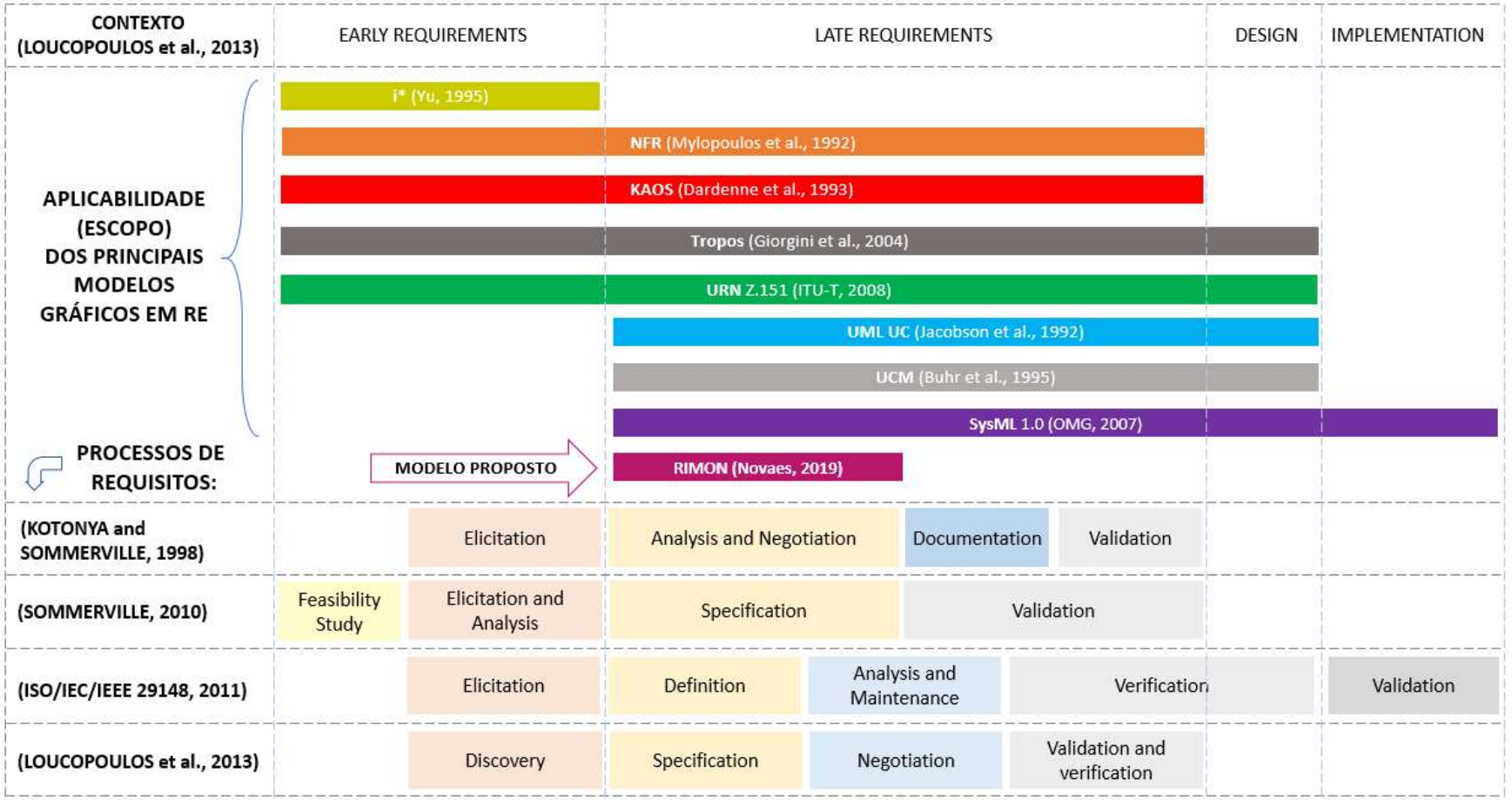


Figura 70 – RIMON comparada com os principais modelos gráficos em termos de processos de RE
Fonte: o autor

6.5 CONCLUSÃO DO CAPÍTULO

Este capítulo apresentou verificações realizadas sob diferentes perspectivas para determinar o alcance das proposições apresentadas nesta dissertação. Todas as verificações estão em consonância com os objetivos específicos propostos para este trabalho.

Na seção 6.1 foi realizada a verificação da notação visual RIMON com o *framework* de verificabilidade baseado em *design rationale*, com objetivo de verificar se a notação foi desenvolvida segundo os princípios de física das notações (PoN). Foi possível constatar que a notação, de um ponto de visual, está em perfeita sintonia com os princípios PoN.

A seção 6.2 contém uma verificação sob a perspectiva de tipos de interdependências entre requisitos, realizada com base nos modelos propostos na literatura: P-Model, D-Model, S-Model e Z-Model. Nesta verificação, foi possível identificar quais tipos de interdependência são suportados pela linguagem RIMON, bem como determinar suas limitações.

A seção 6.3 apresentou uma verificação das diretrizes propostas na seção 3.2 para o desenvolvimento da linguagem RIMON. Estas diretrizes foram fundamentadas no objetivo geral desta dissertação, nos princípios de PoN e em objetivos propostos para alguns dos principais modelos gráficos de RE existentes na literatura. Nesta seção, foi possível constatar a satisfação de todas as diretrizes propostas para a linguagem de modelagem gráfica.

A seção 6.4 apresentou uma comparação em formato tabular da linguagem gráfica RIMON em relação a outros modelos gráficos de RE. Foram escolhidas dezoito características para comparação relacionadas ao contexto de uso (*early/late requirements*), capacidades de modelagem intra-diagrama bem como características gerais relativas a cada modelo gráfico.

7 CONSIDERAÇÕES FINAIS

Este capítulo apresenta as considerações finais deste trabalho de pesquisa, destacando as contribuições, limitações, conclusão e oportunidades de trabalhos futuros.

7.1 CONTRIBUIÇÕES

O presente estudo apresenta as contribuições para a melhoria da qualidade da especificação de requisitos, bem como para fazer progredir o conhecimento em engenharia de requisitos:

a) **Mapeamento sistemático da literatura (SLM) sobre modelos gráficos em RE (seção 2.3 e apêndices A, B e C)**

   Até o presente momento, não foi possível encontrar SLMs que tenham versado especificamente sobre o tema de “modelos gráficos de requisitos”. Os estudos mais próximos deste tema ou se concentram somente em abordagens conhecidas (*mainstream*) tais como as orientadas a objetivos (*Goal-Based*) (HORKOFF et al., 2016) ou não abordam as modelagens diretamente (ABAD et al., 2016).

   O SLM incluído nesta dissertação fornece um embasamento técnico sistemático e abrangente para pesquisas futuras em modelos gráficos em RE;

b) **Método para desenvolvimento de linguagens de modelagem gráfica de requisitos (seção 3.1)**

   Neste trabalho foi proposto e aplicado um método original para desenvolver novas linguagens de modelagem gráfica de requisitos. Este método incluiu a elaboração de todos os elementos formais necessários para definição de uma nova linguagem, tais como sintaxe abstrata (metamodelo e construções semânticas), sintaxe concreta (sintaxe visual) e o mapeamento semântico correspondente. Além disso, o método está em consonância com os princípios de notações visuais de física das notações (PoN), e adota um *framework* específico e apropriado para a verificação de notações visuais.

c) **Linguagem e método de modelagem gráfica de requisitos RIMON (capítulos 3 e 4)**

Este trabalho desenvolveu uma nova linguagem e método para modelagem gráfica de requisitos a RIMON (*Requirements and Interdependencies Modeling Notation*). A RIMON foi definida seguindo princípios de PoN de forma a ser atrativa para a utilização por parte de engenheiros de requisitos e usuários de forma geral.

A linguagem proposta possui uma fundamentação teórica sólida e consistente, contendo uma sintaxe abstrata (metamodelo e construções semânticas) e sintaxe concreta (sintaxe visual) bem definidas. Adicionalmente, a linguagem apresenta um diferencial único em relação aos demais modelos gráficos de requisitos existentes, que pode ser visualizado na comparação apresentada na Tabela 71 (seção 6.4).

Ela permite representar **restrições** como atributos de interdependências em um diagrama de requisitos, por meio de expressões lógico-matemáticas (paramétricas) ou combinações de condições lógicas. Essa característica confere à RIMON a possibilidade de representar requisitos não-funcionais (NFRs) visual e diretamente, algo que a torna única em relação aos demais sistemas de modelagem gráfica de requisitos.

7.2 LIMITAÇÕES

Como limitações e lacunas no desenvolvimento atual da linguagem de modelagem RIMON proposta, é possível apontar:

a) Falta de recurso para gerenciar a escalabilidade de diagramas, de forma a suportar a modelagem de grandes quantidades de requisitos (mecanismo para gerenciar a complexidade visual);
b) Falta de recurso para integração com etapas posteriores de projetos de *software* e sistemas, de forma a permitir a rastreabilidade dos requisitos em relação a artefatos posteriores do projeto;
c) Falta de recurso para gerenciar a evolução e refinamento de requisitos, de forma a facilitar etapas de análise e negociação de RE;
d) Falta de ferramenta (*software*) específica para realização da modelagem gráfica utilizando a notação RIMON.

7.3 CONCLUSÃO

A RIMON é uma linguagem de modelagem gráfica que permite representar requisitos e suas interdependências de forma sistemática, precisa e expressiva. Além de bem representar requisitos funcionais, ela possui uma característica única, que é a de suportar a representação de requisitos não-funcionais por meio da modelagem de condições lógico-matemáticas em atributos de interdependências. Adicionalmente, a linguagem suporta a identificação de conflitos em interdependências de requisitos, e conta com um visual atrativo baseado em princípios de física das notações.

O progresso do conhecimento em RE se evidencia neste trabalho pelas contribuições apresentadas. O mapeamento sistemático da literatura sobre modelos gráficos em RE fornece uma base de conhecimento sólida para futuros pesquisadores interessados em desenvolver modelos gráficos em RE. O método para desenvolvimento de novas linguagens de modelagem gráfica de requisitos oferece uma forma sistemática e precisa para desenvolver notações visuais sólidas e bem fundamentadas. A linguagem e método de modelagem gráfica RIMON, por suas características visuais e semânticas únicas, poderá ser utilizada tanto na indústria por engenheiros de requisitos quanto na academia por pesquisadores da área, contribuindo de forma efetiva para o sucesso de empreendimentos que necessitem de uma linguagem de modelagem sólida e visualmente atrativa.

Assim, o presente trabalho atingiu os objetivos propostos, ao apresentar contribuições para a melhoria da qualidade da especificação de requisitos de *software* e sistemas, e por fazer progredir o conhecimento da área de Engenharia de Requisitos.

7.4 TRABALHOS FUTUROS

Com base nos resultados apresentados, conclusões e limitações desta dissertação, foi elaborado o mapa de pesquisa da Figura 71.

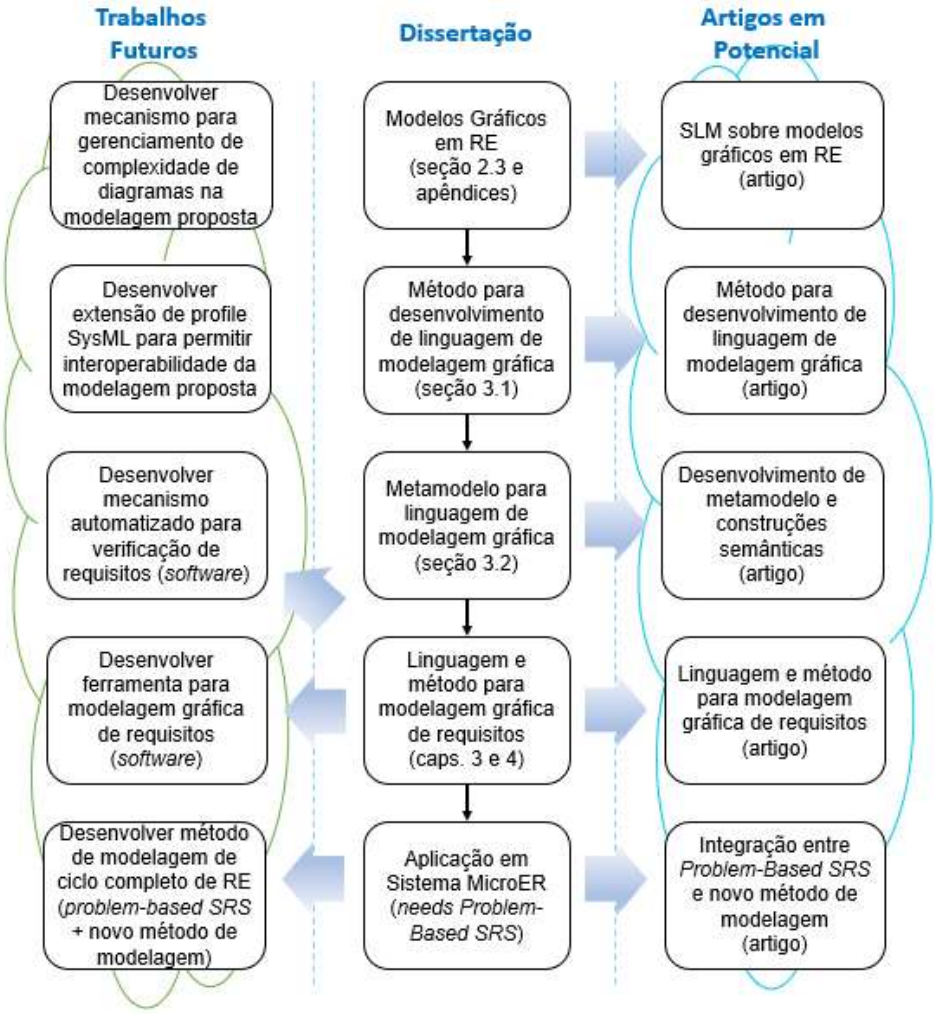


Figura 71 – Mapa de pesquisa evidenciando trabalhos futuros
Fonte: o autor

Nas possibilidades de trabalhos futuros elencados da Figura 71, é possível destacar: desenvolver mecanismo para gerenciamento de complexidade em diagramas da RIMON; desenvolver extensão da linguagem RIMON para integração com SysML; desenvolver mecanismo automatizado para verificação de requisitos; desenvolver ferramenta para modelagem gráfica de requisitos baseada na notação RIMON; desenvolver a integração entre a linguagem de modelagem gráfica RIMON e a abordagem *Problem-Based SRS*. Além disso, há a possibilidade de escrita e publicação de vários artigos científicos sobre cada um destes temas, conforme mostrado na figura.

REFERÊNCIAS

ABAD, Zahra Shakeri Hossein; NOAEEN, Mohammad; RUHE, Guenther. **Requirements Engineering Visualization: A Systematic Literature Review**. IEEE 24th International Requirements Engineering Conference (RE), p. 6–15, 2016.

ABUTALEB, Ahmed S. **Automatic thresholding of gray-level pictures using two-dimensional entropy**. Computer Vision, Graphics and Image Processing, v. 47, n. 1, p. 22–32, 1989.

AL-SUBAIE, Huzam S. F.; MAIBAUM, Tom S. E. **Evaluating the Effectiveness of a Goal-Oriented Requirements Engineering Method**. 4th International Workshop on Comparative Evaluation in Requirements Engineering, CERE 2006, p. 8–19, 2006.

AMIT, Raphael; ZOTT, Christoph. **Value Creation in E-Business**. v. 520, n. February 2000, p. 493–520, 2001.

AMYOT, Daniel. **User Requirements Notation (URN) Overview - ITU-T**. Workshop on Framework and Scope of Formal Languages. 2002. Disponível em: <https://www.itu.int/itudoc/itu-t/com17/urn/urnp1_pp7.ppt>. Acesso: Set/2018.

BANASZEWSKI, Roni F. **Paradigma Orientado a Notificações: Avanços e Comparações.** Dissertação de Mestrado, Universidade Tecnológica Federal do Paraná (UTFPR), Curitiba, PR, Brasil, 2009.

BANASZEWSKI, Roni F.; SIMÃO, Jean M.; TACLA, Cesar A.; STADZISZ, Paulo C. **Notification Oriented Paradigm (NOP): A Software Development Approach based on Artificial Intelligence Concepts**. *In*: 6th Congress of Logic Applied to Technology (LAPTEC 2007). Santos, SP, Brazil, 2007.

BATISTA, Márcio V. **Proposta de um Método de Aplicação da Teoria de Projeto Axiomático ao Desenvolvimento de Software PON-POR.** Dissertação de Mestrado, Universidade Tecnológica Federal do Paraná (UTFPR), Curitiba, PR, Brasil, 2013.

BERENBACH, Brian; SCHNEIDER, Florian; NAUGHTON, Helmut. **The use of a requirements modeling language for industrial applications**. *In*: 20th IEEE International Requirements Engineering Conference (RE), p. 285–290, 2012.

BOEHM, Barry W. **Unifying Software Engineering and Systems Engineering**. Computer, v. 33, n. 3, p. 114–116, 2000.

BOURQUE, P.; FAIRLEY, R. E. (eds.) **Guide to the Software Engineering Body of Knowledge**, Version 3.0. IEEE. Piscataway, NJ, 2014.

BRESCIANI, Paolo; PERINI, Anna.; GIORGINI, Paolo; GIUNCHIGLIA, Fausto; MYLOPOULOS, John; **Tropos: An Agent-Oriented Software Development Methodology. Autonomous Agents and Multi-Agent Systems**, v. 8, p. 203–236, 2004.

BUHR, Ray J. A.; CASSELMAN., Ron S. **Use Case Maps for Object-oriented Systems**. Prentice-Hall, Inc, 1995.

CAILLIAU, A; LAMSWEERDE, Axel Van. **Integrating exception handling in goal models**. *In*: IEEE 22nd International Requirements Engineering Conference (RE), p. 43–52, 2014.

CAIRE, Patrice; GENON, Nicolas; HEYMANS, Patrick; MOODY, Daniel L. **Visual notation design 2.0: Towards user comprehensible requirements engineering notations**. *In*: 21st IEEE International Requirements Engineering Conference (RE), p. 115–124, 2013.

CARLSHAMRE, Pär; SANDAHL, Kristian; LINDVALL, Mikael; *REGNELL, Bjorn; DAG, Johan Natt och*. **An Industrial Survey of Requirements Interdependencies in Software Product Release Planning**. *In*: Proceedings Fifth IEEE International Symposium on Requirements Engineering. p. 84–92, Toronto, Ontario, Canada, 2001.

CHARETTE, Robert N. **Why Software Fails**. IEEE Spectrum, n. 9, p. 42–49, 2005.

CHUNG, Lawrence; NIXON, Brian A.; YU, Eric. **Using Non-Functional Requirements to Systematically Support Change***. In: Proceedings of the Second IEEE International Symposium on Requirements Engineering. York, UK, p. 132–139, 1995.

DAHLSTEDT, Åsa G. **Requirements Interdependencies – a Research Framework**. Department of Computer Science, University of Skövde, Sweden 2001.

DAHLSTEDT, Åsa G.; PERSSON, Anne. **Requirements Interdependencies: State of the Art and Future Challenges (chapter)**. *In*: Engineering and Managing Software Requirements. p. 95–116. Springer-Verlag Berlin Heidelberg, 2005.

DARDENNE, Anne; VAN LAMSWEERDE, A.; FICKAS, Stephen. **Goal-directed requirements acquisition**. Science of Computer Programming, v. 20, p. 3–50, 1993.

DAVIS, A.; DIESTE, O.; HICKEY, A.; JURISTO, N.; MORENO, A. M. **Effectiveness of Requirements Elicitation Techniques: Empirical Results Derived from a Systematic Review**. *In*: 14th IEEE International Requirements Engineering Conference (RE'06), p. 179–188, 2006.

DEMARCO, Tom. **Structured Analysis and System Specification**. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1979.

DOWNING, Steve. **The Social Construction of Entrepreneurship: Narrative and Dramatic Processes in the Coproduction of Organizations and Identities**. Entrepreneurship: Theory and Practice, v. 29, n. 2, p. 185–204, 2005.

FERREIRA, Cleverson A. **Linguagem e compilador para o paradigma orientado a notificações (PON): Avanços e comparações.** Dissertação de Mestrado. Universidade Tecnológica Federal do Paraná (UTFPR), Curitiba, PR, Brasil, 2015.

FRIEDENTHAL, Sanford; MOORE, Alan; STEINER, Rick. **A Practical Guide to SysML: The Systems Modeling Language**. Morgan Kaufmann / OMG Press, 3rd Ed., 2014.

FRIEDENTHAL, Sanford; MOORE, Alan; STEINER, Rick. **OMG Systems Modeling Language (OMG SysML™) Tutorial**. 2008. Disponível em: <http://www.omgsysml.org/INCOSE-OMGSysML-Tutorial-Final-090901.pdf>. Acesso em: Dez- 2018.

GABBRIELLI, Maurizio; MARTINI, Simone. **Programming Languages: Principles and Paradigms**. Undergraduate Topics in Computer Science. 1st. ed. London: Springer London, 2010.

GENON, Nicolas; AMYOT, Daniel; HEYMANS, Patrick. **Analysing the Cognitive Effectiveness of the UCM Visual Notation**. *In*: Proceedings of the 6th international conference on System analysis and modeling: about models (SAM 2010). Oslo, Norway: Springer-Verlag Berlin, p. 221–240, 2010.

GENON, Nicolas; HEYMANS, Patrick; AMYOT, Daniel. **Analysing the Cognitive Effectiveness of the BPMN 2.0 Visual Notation**. *In*: Proceedings of the Third International Conference on Software Language Engineering (SLE'10). Berlin, Heidelberg: Springer-Verlag, p. 377–396, 2011.

GENON, Nicolas; CAIRE, Patrice; TOUSSAINT, Hubert; HEYMANS, Patrick; MOODY, Daniel. **Towards a More Semantically Transparent i* Visual Syntax**. Requirements Engineering: Foundation for Software Quality SE - 12, v. 7195, p. 140–146, 2012.

GILLAIN, J., BURNAY, C., JURETA, I., FAULKNER, S. **AnalyticGraph.com: Toward Next Generation Requirements Modeling and Reasoning Tools**. Proceedings of 24th IEEE International Requirements Engineering Conference, 341-346, IEEE Computer Society, 2016.

GIORGINI, Paolo. **Tropos: Basics (presentation)**. Agent-Oriented Software Engineering course - Trento University (Italy). 2010. Disponível em: <http://www.troposproject.eu/files/8-Tropos-Basics.pdf>. Acesso em Set/2018.

GIORGINI, Paolo; KOLP, Manuel; MYLOPOULOS, John; PISTORE, Marco. **The Tropos Methodology**. *In*: BERGENTI, Federico; GLEIZES, Marie-Pierre; ZAMBONELLI, Franco (Orgs.). **Methodologies and Software Engineering for Agent Systems: The Agent-Oriented Software Engineering Handbook**. Boston, MA: Springer US, p. 89–106, 2004.

GOKNIL, Arda; KURTEV, Ivan; VAN DEN BERG, Klaas; VELDHUIS, Jan-Willem. **Semantics of trace relations in requirements models for consistency checking and inferencing**. Software and Systems Modeling, v. 10, n. 1, p. 31–54, 2011.

GONZALES, Regina. **Developing the requirements discipline: Software vs. systems**. IEEE Software, v. 22, n. 2, p. 59–61, 2005.

GOOLKASIAN, Paula. **Pictures, words, and sounds: From which format are we best able to reason?** Journal of General Psychology, v. 127, n. 4, p. 439–459, 2000.

HE, Xiao; ZHIYI, Ma; SHAO, Weizhong; LI, Ge. **A metamodel for the notation of graphical modeling languages**. *In*: 31st Annual International Computer Software and Applications Conference (COMPSAC 2007). Beijing, China: IEEE, v. 1, p. 219–222, 2007.

HELMING, Jonas; KOEGEL, Maximilian; SCHNEIDER, Florian; HAEGER, Michael; KAMINSKI, Christine; BRUEGGE, Bernd; BERENBACH, Brian. **Towards a Unified Requirements Modeling Language**. In: Fifth International Workshop on Requirements Engineering Visualization (REV). Sydney, Australia: IEEE Computer Society, p. 53–57, 2010.

HINKELMANN, Knut. **Meta-Modeling and Modeling Languages**. Tutorial - Università di Camerino, Itália. 2015. Disponível em: <http://didattica.cs.unicam.it/lib/exe/fetch.php?media=didattica:magistrale:abit:ay_1516:abit_03_metamodelling.pdf>. Acesso em Set/2018.

HOFMANN, Hubert F.; LEHNER, Franz. **Requirements engineering as a success factor in software projects**. IEEE Software, v. 18, n. 4, p. 58–66, 2001.

HORKOFF, J; AYDEMIR, F B; CARDOSO, E; *et al*. **Goal-Oriented Requirements Engineering: A Systematic Literature Map**. *In*: IEEE 24th International Requirements Engineering Conference (RE), p. 106–115, 2016.

INCOSE. **Systems Engineering Handbook. A guide for system life cycle processes and activities**. Prepared by INCOSE. 3th Ed., v. 3.2; Hoboken, NJ.; John Wiley & Sons, 2010.

INCOSE. **Systems Engineering Handbook. A guide for system life cycle processes and activities**. Prepared by INCOSE. 4th Ed.; Hoboken, NJ.; John Wiley & Sons, 2015.

ISO/IEC/IEEE. **Systems and software engineering - Life cycle processes - Requirements engineering - ISO/IEC/IEEE 29148:2011**. The Institute of Electrical and Electronics Engineers. Piscataway, NJ. 2011.

ISO/IEC/IEEE. **Systems and Software Engineering - Vocabulary ISO/IEC/IEEE 24765: 2010**. The Institute of Electrical and Electronics Engineers. Piscataway, NJ. 2010.

ITU-T. **User Requirements Notation (URN) – Language definition. Recomendation Z.151 (10/2008)**. 2008. Disponível em: <http://www.itu.int/rec/T-REC-Z.151/en>. Acesso em Set/2018.

ITU-T. **User Requirements Notation (URN) – Language definition. Recomendation Z.151 (11/2012)**. 2012. Disponível em: <http://www.itu.int/rec/T-REC-Z.151/en>. Acesso em Set/2018.

JACOBSON, Ivar; CHRISTERSON, Magnus; JONSSON, Patrik; OVERGAARD, Gunnar. **Object-Oriented Software Engineering: A Use Case Driven Approach**, HarlowEssex England Addison, 1992.

KAINDL, Hermann; JÄNTTI, Marko; MANNAERT, Herwig; *NAKAMATSU, Kazumi; RIEKE, Roland*. **Requirements Engineering for Software vs. Systems in General**. *In*: The Seventh International Conference on Systems (ICONS 2012), p. 190–192, 2012.

KAPUR, J. N.; SAHOO, P. K.; WONG, A. K. C. **A new method for gray-level picture thresholding using the entropy of the histogram**. Computer Vision Graphics and Image Processing, v. 29, p. 273–285, 1985.

KARLSSON, Joachim; OISSON, Stefan; RYAN, Kevin. **Improved Practical Support for Large-scale Requirements Prioritising**. Requirements Engineering, v. 2, n. 1, p. 51–60, 1997.

KEATING, Charles B.; PADILLA, Jose J.; ADAMS, Kevin. **System of systems engineering requirements: Challenges and guidelines**. Engineering Management Journal (EMJ), v. 20, n. 4, p. 24–31, 2008.

KERSCHBAUMER, Ricardo. **Proposição do Paradigma Orientado a Notificações no Desenvolvimento de Circuitos Lógico-Digitais Reconfiguráveis,** Tese de Doutorado. Universidade Tecnológica Federal do Paraná, Curitiba, 2018.

KERSCHBAUMER, Ricardo; LINHARES, Robson R.; SIMÃO, Jean M.; STADZISZ, Paulo C.; LIMA, Carlos R. E. **Notification-Oriented Paradigm to Implement Digital Hardware**. Journal of Circuits, Systems and Computers, v. 27, n. 8, p. 1–28, 2018.

KIM, Jinwoo; HAHN, Jungpil; HAHN, Hyoungmee. **How Do We Understand a System with (So) Many Diagrams? Cognitive Integration Processes in Diagrammatic Reasoning**. Information Systems Research, v. 11, n. 3, p. 284–303, 2000.

KITCHENHAM, B. A.; DYBÅ, T. e JØRGENSEN, M. 2004. **Evidence-based software engineering**. Proceedings of 27th IEEE International Software Engineering Conference, 273-281, IEEE Computer Society, 2004.

KITCHENHAM, Barbara A.; BUDGEN, David; PEARL BRERETON, O. **Using mapping studies as the basis for further research - A participant-observer case study**. Information and Software Technology, v. 53, n. 6, p. 638–651, 2011.

KOSSLYN, Stephen M.; BERTIN, J.; BERG, W. J.; CHAMBERS, J. M.; CLEVELAND, W. S.; KLEINER, B.; TUKEY, P. A.; FISHER, H. T.; SCHMID, C. F.; TUFTE, E. R. **Graphics and human information processing: A review of five books**. Journal of the American Statistical Association, v. 80, n. 391, p. 499-512, 1985.

KOSSOSKI, Clayton. **Proposta de Um Método de Teste para Processos de Desenvolvimento de Software usando o Paradigma Orientado a Notificações.** Dissertação de Mestrado. Universidade Tecnológica Federal do Paraná (UTFPR), Curitiba, PR, Brasil, 2015.

KOSSOSKI, Clayton; SIMÃO, Jean M.; STADZISZ, Paulo C. **Introdução ao teste funcional de software no Paradigma Orientado a Notificações Paradigma Orientado a Notificações**. *In*: VI Congreso Internacional de Computación y Telecomunicaciones, p. 136–146, 2014.

KOTHARI, C. R. **Research methodology: methods and techniques**. New Delhi, India: New Age International Limited Publishers, 2004.

KOTONYA, Gerald; SOMMERVILLE, Ian. **Requirements Engineering : Processes and Techniques.** Worldwide Series in Computer Science. 1st. ed. Chichester, England: John Wiley & Sons, Inc., 1998.

LAPOUCHNIAN, Alexei. **Goal-Oriented Requirements Engineering: An Overview of the Current Research**, Technical Report, Department of Computer Science, University of Toronto, 2005.

LARKIN, Jill H.; SIMON, Herbert A. **Why a Diagram is (Sometimes) Worth Ten Thousand Words**. Cognitive Science, v. 11, n. 1, p. 65–99, 1987.

LEE, Jintae. **Design rationale systems: Understanding the issues**. IEEE Expert-Intelligent Systems and their Applications, v. 12, n. 3, p. 78–85, 1997.

LEHTINEN, Timo O. A.; MÄNTYLÄ, Mika V.; VANHANEN, Jari; ITKONEN, Juha LASSENIUS, Casper. **Perceived causes of software project failures – An analysis of their relationships**. Information and Software Technology, v. 56, n. 6, p. 623–643, 2014.

LI, Juan; ZHU, Liming; JEFFERY, Ross; LIU, Yan; ZHANG, He; WANG, Qing; LI, Mingshu. **An initial evaluation of requirements dependency types in change propagation analysis**. *In*: 16th International Conference on Evaluation & Assessment in Software Engineering. p. 62–71, Ciudad Real, Spain, 2012.

LINDEN, Dirk Van Der; HADAR, Irit. **A Systematic Literature Review of Applications of the Physics of Notation**. IEEE Transactions on Software Engineering, v. 5589, p. 1–26, 2018.

LINDEN, Dirk Van Der; ZAMANSKY, Anna; HADAR, Irit. **A Framework for Improving the Verifiability of Visual Notation Design Grounded in the Physics of Notations**. *In*: IEEE 25th International Requirements Engineering Conference (RE), p. 41–50, 2017.

LINDEN, Dirk Van Der; ZAMANSKY, Anna; HADAR, Irit. **How cognitively effective is a visual notation? On the inherent difficulty of operationalizing the physics of notations**. Lecture Notes in Business Information Processing, v. 248, p. 448–462, 2016.

LINDWELL, William; HOLDEN, Kritina; BUTLER, Jill. **Universal Principles of Design: A Cross-Disciplinary Reference**, Rockport Publishers, 2003.

LINHARES, Robson R. **Contribuição para o desenvolvimento de uma arquitetura de computação própria ao paradigma orientado a notificações**. Tese de Doutorado. Universidade Tecnológica Federal do Paraná (UTFPR), Curitiba, PR, Brasil, 2015.

LOUCOPOULOS, PERICLES; SUN, Jie; ZHAO, Liping; HEIDARI, Farideh. **A systematic classification and analysis of NFRs**. 19th Americas Conference on Information Systems, AMCIS 2013 - Hyperconnected World: Anything, Anywhere, Anytime, v. 1, p. 208–217, 2013.

LU, Haifeng; ZHANG, Tianxu; YAN, Luxin. **Threshold Selection using Partial Structural Similarity**. International Journal of Digital Content Technology and its Applications, v. 5, n. 7, p. 397–407, 2011.

MAGABLEH, Basel; BARRETT, Stephen. **Productivity evaluation of Self-Adaptive software model driven architecture**. International Journal of Information Technology and Web Engineering, v. 6, n. 4, p. 1–19, 2011.

MARQUES, Rita; COSTA, Gonçalo; SILVA, Miguel; GONÇALVES, Pedro. **A Survey of Failures in the Software Development Process**. *In*: 25th European Conference on Information Systems (ECIS). p. 2445–2459, Guimarães, Portugal, 2017.

MATHWORKS. **SSIM - Structural Similarity Index (MATLAB Tool)**. 2018. Disponível em: <https://www.mathworks.com/help/images/ref/ssim.html>. Acesso em Out-2018.

MENDONÇA, Igor T. M. **Metodologia de Projeto de Software Orientado a Notificações.** Trabalho de qualificação de doutorado. Universidade Tecnológica Federal do Paraná (UTFPR), Curitiba, PR, Brasil, 2018.

MENDONÇA, Igor T. M.; SIMÃO, Jean M.; WIECHETECK, Luciana V. B.; STADZISZ, Paulo C. **Método para Desenvolvimento de Sistemas Orientados a Regras utilizando o Paradigma Orientado a Notificações**. Congresso Brasileiro de Inteligência Computacional, n. 12, p. 1–6, 2015.

MERRIAN-WEBSTER DICTIONARY, Definições de “inference” e “request”. 2018. <https://www.merriam-webster.com/dictionary/inference> Acesso em Set/2018.

MICHAELIS DICIONÁRIO, Definições de “sistemático”, “preciso” e “expressivo”, 2018. < https://michaelis.uol.com.br/moderno-portugues/> Acesso em Out/2018.

MILLER, George A. **The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information**. Psychological Review, v. 101, n. 2, p. 343–352, 1994.

MOODY, Daniel L. **The “Physics” of Notations: Towards a Scientific Basis for Constructing Visual Notations in Software Engineering**. IEEE Transactions on Software Engineering, v. 35, n. 6, p. 756–779, 2009.

MOODY, Daniel L. **The “physics” of notations: A scientific approach to designing visual notations in software engineering**. *In*: ACM/IEEE 32nd International Conference on Software Engineering, v. 2, p. 485–486, 2010.

MOODY, Daniel; HILLEGERSBERG, Jos. **Evaluating the Visual Syntax of UML: An Analysis of the Cognitive Effectiveness of the UML Family of Diagrams.** *In: Software Language Engineering*. GAŠEVIĆ, Dragan; LÄMMEL, Ralf; WYK, Eric (Orgs.). Springer-Verlag Berlin, p. 16–34, 2009.

MOODY, Daniel L.; HEYMANS, Patrick; MATULEVICIUS, Raimundas. **Improving the Effectiveness of Visual Representations in Requirements Engineering: An Evaluation of i* Visual Syntax**. *In*: 17th IEEE International Requirements Engineering Conference, p. 171–180, 2009.

MOODY, Daniel L; HEYMANS, Patrick; MATULEVIČIUS, Raimundas. **Visual syntax does matter: Improving the cognitive effectiveness of the i* visual notation**. Requirements Engineering, v. 15, n. 2, p. 141–175, 2010.

MOORE, James W. **Converging Software and Systems Engineering Standards**. Computer, v. 39, n. 9, p. 106–108, 2006.

MYLOPOULOS, John; CHUNG, Lawrence. **From Object-Oriented to Goal-Oriented Requirements Analysis**. Communications of the ACM, 1999.

MYLOPOULOS, John; CHUNG, Lawrence; NIXON, Brian A. **Representing and Using Non-Functional Requirements: A Process-Oriented Approach**. IEEE Transactions on Software Engineering, v. 18, p. 483--497, 1992.

NOVAES, Paulo J. D.; SIMÃO, Jean M.; STADZISZ, Paulo C. **Integration between Requirements Modeling and Software Development in the**

Notification Oriented Paradigm: A Security System Case Study. *In*: IX Computer on the Beach (COTB 2018). p. 432–441 Florianópolis, Brasil, 2018.

NORDBOTTEN, J. C.; CROSBY, M. E. **The effect of graphic style on data model interpretation**. Information Systems Journal, v. 9, n. 2, p. 139–155, 1999.

NUSEIBEH, Bashar; EASTERBROOK, Steve. **Requirements Engineering: A Roadmap**. *In*: Proceedings of the Conference on The Future of Software Engineering, (ICSE '00). Limerick, Ireland: ACM, p. 35–46, 2000.

OMG. **Unified Modeling Language: Superstructure Version 2.0**. formal/05-07-04. 2005. Disponível em: < https://www.omg.org/cgi-bin/doc?formal/05-07-04>. Acesso em Set-2018.

OMG, Object Management Group. **Systems Modeling Language (SysML), Version 1.0. 07-09-01**. 2007. Disponível em: <https://www.omg.org/spec/SysML/1.0/>. Acesso em: Jun-2018.

OMG, Object Management Group. **Systems Modeling Language (SysML)**, Version 1.5. SysML formal/17-05-01. 2017. Disponível em: <https://www.omg.org/spec/SysML/1.5/>. Acesso em: Jun-2018.

OTSU, Nobuyuki. **A Threshold Selection Method from Gray-Level Histograms**. IEEE Transactions on Systems, Man, and Cybernetics, v. 9, n. 1, p. 62–66, 1979.

OXFORD DICTIONARY, Definition of "premise". 2018. <https://en.oxforddictionaries.com/definition/premise> Acesso em Set/2018.

PAIVIO, Allan. **Mental Representations: A Dual Coding Approach**, Oxford University Press, 1990.

PANDEY, Vishal; BAIRWA, A; BHATTACHARYA, Sweta. **Application of the Pareto Principle in Rapid Application Development Model**. International Journal of Engineering and Technology, v. 5, p. 2649–2654, 2013.

PETERS, Eduardo; JASINSKI, Ricardo P; PEDRONI, Volnei A; SIMÃO, Jean M. **A New Hardware Coprocessor for Accelerating Notification-Oriented Applications**. *In*: 2012 International Conference on Field-Programmable Technology. Seoul, South Korea, p. 257–260, 2012.

PETERSEN, Kai; FELDT, Robert; MUJTABA, Shahid; MATTSSON, Michael. **Systematic Mapping Studies in Software Engineering**. 12th International Conference on Evaluation and Assessment in Software Engineering (EASE), v. 17, p. 10, 2008.

PETRE, Marian. **Why looking isn't always seeing: readership skills and graphical programming**. Communications of ACM, v. 38, n. 6, p. 33–44, 1995.

POHL, Klaus. **Process-centered Requirements Engineering**. Advanced software development series. New York, NY, USA. John Wiley and Sons, 1996.

PORDEUS, Leonardo F. **Simulação de uma arquitetura de computação própria ao paradigma orientado a notificações (Msc. Thesis)**. Universidade Tecnológica Federal do Paraná (UTFPR), Curitiba, PR, Brasil, 2017.

RANDELL, Brian. **The 1968/69 NATO Software Engineering Reports**. Department of Computing Science, University of Newcastle. 2018. Disponível em

<http://homepages.cs.ncl.ac.uk/brian.randell/NATO/NATOReports/index.html>. Acessado em Jul-2018.

RAMESH, Balasubramaniam; JARKE, Matthias. **Toward Reference Models for Requirements Traceability**. IEEE Transactions on Software Engineering, v. 27, n. 1, p. 58–93, 2001.

ROBINSON, William N; PAWLOWSKI, Suzanne D; VOLKOV, Vecheslav. **Requirements Interaction Management**. ACM Computing Surveys, v. 35, n. 2, p. 132–190, 2003.

RONSZCKA, Adriano F. **Contribuição Para a Concepção De Aplicações No Paradigma Orientado a Notificações (PON) Sob O Viés De Padrões.** Dissertação de Mestrado, Universidade Tecnológica Federal do Paraná (UTFPR), Curitiba, PR, Brasil, 2012.

RONSZCKA, Adriano F.; FERREIRA, Cleverson A.; STADZISZ, Paulo C.; FABRO, João A.; SIMÃO, Jean M. **Notification-Oriented Programming Language and Compiler**. *In*: VII Brazilian Symposium on Computing Systems, p. 125–131, 2017.

ROY, Peter Van. **Programming Paradigms for Dummies: What Every Programmer Should Know**. New Computational Paradigms for Computer Music, p. 9–47, 2009.

ROY, Peter Van; HARIDI, Seif. **Concepts, Techniques and Models on Computer Programming**, Massachusetts Institute of Technology, 2004.

SANTOS, Leonardo Araújo. **Linguagem e compilador para o paradigma orientado a notificações: avanços para facilitar a codificação e sua validação em uma aplicação de controle de futebol de robôs.** Dissertação de Mestrado. Universidade Tecnológica Federal do Paraná, Curitiba, Brasil, 2017.

SCHNEIDER, Florian; BRUEGGE, Bernd; BERENBACH, Brian. **A tool implementation of the unified requirements modeling language as enterprise architect add-in**. *In*: 21st IEEE International Requirements Engineering Conference (RE), p. 334–335, 2013.

SERC, Systems Engineering Research Center. **Systems Engineering and Software Engineering (SERC Wiki)**. 2018. Disponível em: <https://www.sebokwiki.org/wiki/Systems_Engineering_and_Software_Engineering>. Acesso em Jun-2018.

SHANNON, Claude E.; WEAVER, Warren. **The Mathematical Theory of Communication**. *1st ed.*, The University of Illinois, 1963.

SHEARD, Sarah A.; LAKE, Jerome G. **Systems Engineering Standards and Models Compared**. *In*: INCOSE International Symposium, v. 8, p. 591–598, 1998.

SHEARD, Sarah. **Needed: Improved Collaboration Between Software and Systems Engineering**. Software Engineering Institute - Carnegie Mellon University. 2014. Disponível em: <http://blog.sei.cmu.edu/post.cfm/importance-software-architecture-big-data-systems-013>. Acesso em Set-2018.

SIMÃO, Jean M. **Proposta de uma Arquitetura de Controle para Sistemas Flexíveis de Manufatura Baseada em Regras e Agentes.** Dissertação de Mestrado. CPGEI/CEFET-PR, Curitiba - PR, Brasil, 2001.

SIMÃO, Jean M. **A Contribution to the Development of a HMS Simulation Tool and Proposition of a Meta-Model for Holonic Control.** PhD Thesis. Universidade Tecnológica Federal do Paraná, 2005.

SIMÃO, Jean M.; PANETTO, Hervé; LIAO, Yongxin; Stadzisz, Paulo C. **A Notification-Oriented Approach for Systems Requirements Engineering**. *In*: 23rd IPSE International Conference on Transdisciplinary Engineering. p. 229-238. Curitiba, Brasil, 2016.

SIMÃO, Jean M.; STADZISZ, Paulo C. **Paradigma Orientado a Notificações (PON) - Uma Técnica de Composição e Execução de Software Orientado a Notificações**. Pedido de Patente: Privilégio de Inovação. Número do registro: PI08055181, data de depósito: 26/11/2008, Instituto Nacional da Propriedade Industrial (INPI). Universidade Tecnológica Federal do Paraná (UTFPR), 2008.

SIMÃO, Jean M.; STADZISZ, Paulo C. **Inference Based on Notifications: A Holonic Metamodel Applied to Control Issues**. IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, v. 39, n. 1, p. 238–250, 2009.

SIMÃO, Jean M.; STADZISZ, Paulo C. **Mecanismo de Resolução de Conflito e Garantia de Determinismo para o Paradigma Orientado a Notificações (PON).** Pedido de Patente: Privilégio de Inovação. Número do registro: PI10002960, data de depósito: 26/02/2010, Instituto Nacional da Propriedade Industrial (INPI). Universidade Tecnológica Federal do Paraná, 2010.

SIMÃO, Jean M.; TACLA, Cesar A.; STADZISZ, Paulo C.; LINHARES, Robson R.; VALENÇA, Glauber Z.; BANASZEWSKI, Roni F.; FABRO, João A.; TACLA, Cesar A. **Notification Oriented Paradigm (NOP) and Imperative Paradigm: A Comparative Study**. Journal of Software Engineering and Applications, v. 05, n. 06, p. 402–416, 2012.

SOMMERVILLE, I. **Software Engineering**. 9. ed. Boston: Pearson, 2010.

SOUZA, Rafael G. M. **Problem-Based SRS: Método para especificação de requisitos de software baseado em Problemas.** Dissertação de Mestrado, Universidade Tecnológica Federal do Paraná (UTFPR), Curitiba, PR, Brasil, 2016.

SOUZA, Rafael G. M.; STADZISZ, Paulo C. **Problem-Based Software Requirements Specification**. Revista Eletrônica de Sistemas de Informação, v. 15, n. 2, p. 1–25, 2016.

SPIJKERMAN, Wietze. **Tool Support for Change Impact Analysis in Requirement Models (Msc. Thesis)**. University of Twente, 2010.

STADZISZ, Paulo C. **Relatório de Pesquisa – Sistema MicroER**. Laboratório de Inovações Tecnológicas (LIT). Universidade Tecnológica Federal do Paraná (UTFPR). Curitiba, PR, Brasil, 2016.

STÖRRLE, Harald; FISH, Andrew. **Towards an Operationalization of the "Physics of Notations" for the Analysis of Visual Languages**. *In*: *Proceedings of* Models, 2013.

TEIXEIRA, Maria G. S.; QUIRINO, Glaice K.; GAILLY, Frederik; FALBO, Ricardo A.; GUIZZARDI, Giancarlo; BARCELLOS, Monalessa P. **PoN-S: A Systematic Approach for Applying the Physics of Notation (PoN)**. *In*: *Proceedings of*

Enterprise, Business-Process and Information Systems Modeling. BPMDS 2016, EMMSAD 2016. Ljubljana, Slovenia. Springer, 2016.

STANDISH GROUP. **The Standish Group CHAOS Report**. The Standish Group. Disponível em: <https://www.projectsmart.co.uk/white-papers/chaos-report.pdf>. 1994. Acessado em: Nov-2018.

SUPAKKUL, S; CHUNG, Lawrence. **The RE-Tools: A multi-notational requirements modeling toolkit**. *In*: 2012 20th IEEE International Requirements Engineering Conference (RE), p. 333–334, 2012.

TSAI, Wen-Hsiang. **Moment-preserving thresholding: A new approach**. Computer Vision, Graphics, and Image Processing, v. 29, n. 3, p. 377–393, 1985.

VALENÇA, Glauber Z. **Contribuição para a Materialização do Paradigma Orientado a Notificações (PON) via Framework e Wizard.** Dissertação de Mestrado. Universidade Tecnológica Federal do Paraná (UTFPR), Curitiba, PR, Brasil, 2012.

VESSEY, Iris; GALLETTA, Dennis. **Cognitive fit: An empirical study of information acquisition**. Information Systems Research, v. 2, n. 1, p. 63–84, 1992.

VERNER, June; SAMPSON, Jennifer; CERPA, Narciso. **What factors lead to software project failure?** *In*: Second International Conference on Research Challenges in Information Science. , p. 71–80, Marrakech, Morocco, 2008

WANG, Zhou; BOVIK, Alan C.; SHEIKH, Hamid R.; *et al*. **Image quality assessment: From error visibility to structural similarity**. IEEE Transactions on Image Processing, v. 13, n. 4, p. 600–612, 2004.

WEBER, Ron A. **Ontological foundations of information systems**, Coopers & Lybrand, 1997.

WHITE, Stephanie M. **Integrating system and software engineering processes using the views and viewpoints of a new discipline**. *In*: IEEE International Systems Conference Proceedings. Ottawa, ON, Canada, p. 525–530, 2014.

WIECHETECK, Luciana V. B. **Método para projeto de software usando o paradigma orientado a notificações - PON.** Dissertação de Mestrado. Universidade Tecnológica Federal do Paraná (UTFPR), Curitiba, PR, Brasil, 2011.

WIECHETECK, Luciana V. B.; STADZISZ, Paulo C.; SIMÃO, Jean M. **Um Perfil UML para o Paradigma Orientado a Notificações (PON)**. III Congreso Internacional de Computación y Telecom – COMTEL, p. 1–16, 2011.

WIERINGA, Roel; MAIDEN, Neil; MEAD, Nancy; Rolland, Colette. **Requirements engineering paper classification and evaluation criteria: A proposal and a discussion**. Requirements Engineering, v. 11, n. 1, p. 102-107, 2006.

WINN, William. **An Account of How Readers Search for Information in Diagrams**. Contemporary Educational Psychology, v. 18, n. 2, p. 162–185, 1993.

WINN, William. **Encoding and Retrieval of Information in Maps and Diagrams**. IEEE Transactions on Professional Communication, v. 33, n. 3, p. 103–107, 1990.

XAVIER, Robson D. **Paradigmas de Desenvolvimento de Software: Comparação entre abordagens orientada a eventos e orientada a notificações.** Dissertação de Mestrado. Universidade Tecnológica Federal do Paraná (UTFPR), Curitiba, PR, Brasil, 2014.

YU, Eric S. K. **Modelling Strategic Relationships for Process Reengineering (PHD Thesis)**. University of Toronto, 1995.

YU, Eric S. K. **Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering**. In: Proceedings of the Third IEEE International Symposium on Requirements Engineering. Annapolis, USA, p. 226–235, 1997.

YU, Eric S. K. **Agent-oriented modelling: Software versus the world**. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), v. 2222, p. 206–225, 2002.

YU, Eric S. K. **Social modeling and i***. *In*: BORGIDA A.T., CHAUDHRI V.K., GIORGINI P., Yu E.S. (Org.). Conceptual Modeling: Foundations and Applications. Lecture Notes in Computer Science, v. 5600, p. 99–12, 2009.

ZAVE, Pamela. **Classification of research efforts in requirements engineering**. ACM Computing Surveys, v. 29, n. 4, p. 315–321, 1997.

ZHANG, He; LI, Juan; ZHU, Liming; JEFFERY, Ross; LIU, Yan; WANG, Qing; LI, Mingshu. **Investigating dependencies in software requirements for change propagation analysis**. Information and Software Technology, v. 56, n. 1, p. 40–53, 2014.

APÊNDICE A – METODOLOGIA: SLM SOBRE MODELOS GRÁFICOS EM RE

Este apêndice contém o detalhamento da metodologia adotada para a elaboração do SLM sobre modelos gráficos em engenharia de requisitos.

I. HIPÓTESE

Atualmente há falta de estruturação do conhecimento relativo a requisitos de *software* e de sistemas, em forma de representações gráficas que modelem de forma sistemática, precisa e expressiva os requisitos e suas interdependências.

II. OBJETIVO

Identificar e descrever as principais abordagens de modelagem gráfica de requisitos existentes na literatura de RE, em termos de modelos visuais (diagramas, elementos gráficos, notações), linguagens de especificação, ferramentas de modelagem, metodologias (ou métodos), normas internacionais (*international standards*) e metamodelos (ou Ontologias).

III. QUESTÕES DE PESQUISA

As questões de pesquisa propostas para este trabalho foram estruturadas conforme os critérios PICOC (*Population, Intervention, Comparison, Outcome, Context*) sugeridos por Kitchenham e Charters (KITCHENHAM e CHARTERS, 2007) e reunidos na Tabela 72.

População	Modelos gráficos de requisitos
Intervenção	Modelagem gráfica ou visual de requisitos de *software*
Comparação	Modelos visuais, linguagens de especificação, notações, ferramentas de modelagem, métodos, padrões ou normas internacionais e metamodelos.
Resultado	Não concentrado em resultados quantitativos
Contexto	Engenharia de Requisitos e Engenharia de *Software*

Tabela 72 – Critérios base (PICOC) escolhidos para as questões de pesquisa.
Fonte: o autor

As **questões de pesquisa** propostas neste trabalho são: (Tabela 73):

Nº	Questão de Pesquisa
Q1	Quais os principais modelos gráficos e técnicas de modelagem gráficas de requisitos propostos na Literatura?
Q2	Quais os modelos visuais (diagramas, elementos gráficos, notações) associados a cada técnica/modelo identificado?
Q3	Quais as linguagens de especificação associadas a cada modelo visual identificado?
Q4	Quais as ferramentas de modelagem propostas para operacionalizar o modelo visual identificado?
Q5	Quais as metodologias ou abordagens de Engenharia de Requisitos propostas para cada modelo visual identificado?
Q6	Quais as normas internacionais associadas a cada modelo identificado?
Q7	Quais os metamodelos ou ontologias propostas para cada técnica/modelo identificado?
Q8	Como ocorreu a evolução temporal dos sistemas de modelagem gráfica de RE identificados?
Q9	De forma comparativa, quais são as características de modelagem associadas a cada abordagem de modelagem gráfica identificada?

Tabela 73 – Questões de pesquisa
Fonte: o autor

IV. MÉTODO DE PESQUISA

O método de pesquisa escolhido para este trabalho foi o de Mapeamento Sistemática da Literatura ou *Systematic Literature Mapping* (SLM) segundo princípios estabelecidos em estudos de Peterson (PETERSEN et al., 2008) e Kitchenham (KITCHENHAM et al., 2011)

Considerados o objetivo e as questões de pesquisa propostas anteriormente, foi elaborado o método de SLM apresentado na Figura 72, adaptado a partir do processo genérico de Petersen (PETERSEN et al., 2008).

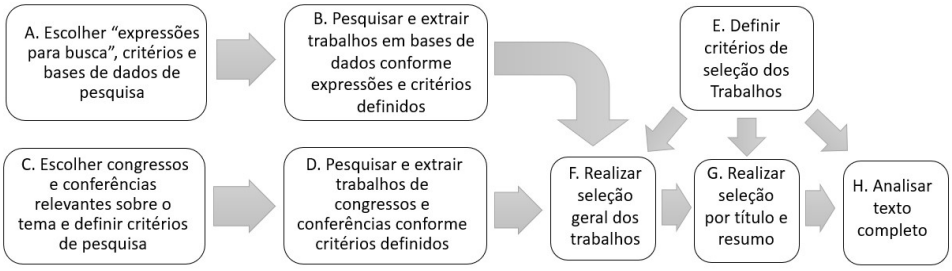


Figura 72 – Método de pesquisa adotado para o SLM
Fonte: o autor

Nas próximas seções (A até H) será apresentada a descrição detalhada de cada etapa do método acima e os resultados intermediários obtidos.

A. Escolher "expressões para busca", critérios e bases de dados de pesquisa

Esta etapa visa determinar informações críticas para a realização da pesquisa nas etapas posteriores de forma a obter o máximo de efetividade possível (Tabela 74), a saber: "expressões para busca" ou "palavras-chave"; "critérios" ou "filtros de busca"; "bases de dados" que serão utilizadas.

Expressões para Busca	• Em Inglês: ○ "*requirements engineering" AND "graphical model"*; ○ "*requirements engineering" AND "visual model"*; ○ "*requirements engineering" AND "graphical modeling"*; ○ "*requirements engineering" AND "visual modeling"*; ○ *"requirements diagram"* ○ *"requirements notation"* ○ "requirements engineering" AND "model-based" • Em Português: ○ "modelo gráfico" *AND* "engenharia de requisitos"; ○ "modelo visual" *AND* "engenharia de requisitos"; ○ "modelagem gráfica" *AND* "engenharia de requisitos"; ○ "modelagem visual" *AND* "engenharia de requisitos"; ○ "diagrama de requisitos" ○ "notação de requisitos ○ "baseada em modelos" AND "engenharia de requisitos"
Critérios (filtros de busca)	• Data de publicação: entre janeiro/1988 e janeiro/2018. • Publicado em inglês ou português. • Publicado em anais de conferências, periódicos, livros, dissertações ou teses. • Não duplicados.

	• Somente textos completos. • Ênfase em: Engenharia de Requisitos, Engenharia de *Software*, Engenharia de Sistemas e áreas afins. • Somente trabalhos revisados por pares.
Bases de Dados	A pesquisa será realizada pelo Portal de periódicos da CAPES, que inclui acesso a mais de 500 bases de pesquisa em nível mundial (em janeiro/2018), entre as quais destacam-se: *ACM Digital Library*, Catálogo de Teses e Dissertações (CAPES), *Computer and Information Systems* (ProQuest), *Computers and Applied Sciences Complete* (EBSCO), *DataSearch* (Elsevier), *Engineering Journals* (ProQuest), *Engineering Research Database* (ProQuest), *Google Scholar*, IEEE *Xplore*, Instituto Brasileiro de Informação em Ciência e Tecnologia (IBICT), Repositórios institucionais de teses e dissertações de universidades brasileiras, SciELO *Citation Index* (*Web of Science*), *ScienceDirect* (Elsevier), SCOPUS (Elsevier).

Tabela 74 – Expressões para busca, critérios e bases de dados
Fonte: o autor

B. Pesquisar e extrair trabalhos em bases de dados

Nesta etapa foram realizadas as pesquisas conforme as expressões para busca, critérios, e bases de dados definidos na etapa “A”. Os trabalhos foram inicialmente extraídos em formato de referências bibtex (.bib), agrupados, convertidos e registrados em uma tabela *Excel* (.xlsx), visando a futura análise e seleção nas etapas “F”, “G” e “H”. Também foram removidos artigos duplicados.

Na Tabela 75 é possível observar um resumo da quantidade de trabalhos (*papers*) obtidos, separados por expressão de busca e idioma.

Expressão para busca	Idioma	Quant. *papers*
“requirements diagram"	Inglês	45
“requirements engineering” AND "visual model”	Inglês	28
“requirements engineering” AND “graphical model”	Inglês	51
“requirements engineering” AND “graphical modeling"	Inglês	61
“requirements engineering” AND “visual modeling"	Inglês	73
“requirements model”	Inglês	16
“requirements notation"	Inglês	133
“requirements engineering” AND “model-based"	Inglês	23
"diagrama de requisitos"	Português	27
"modelagem gráfica" AND "engenharia de requisitos"	Português	1

Expressão para busca	Idioma	Quant. *papers*
"modelagem visual" AND "engenharia de requisitos"	Português	3
"modelo gráfico" AND "engenharia de requisitos"	Português	4
"modelo visual" AND "engenharia de requisitos"	Português	3
"notação de requisitos"	Português	1
“modelo de requisitos”	Português	1
“baseada em modelos” AND “engenharia de requisitos”	Português	0
	Total	470

Tabela 75 – Sumário da quantidade de artigos encontrados no portal da CAPES
Fonte: o autor

C. Escolher conferências relevantes e definir critérios de pesquisa

Nesta etapa, foram identificados congressos ou conferências relevantes que possam conter artigos sobre **engenharia de requisitos**, de forma a aumentar as chances de encontrar artigos específicos sobre os termos de pesquisa desejados (ver Tabela 76).

Congressos Escolhidos	Internacionais: • *International Requirements Engineering Conference* (RE) • *International Conference on Software Engineering* (ICSE) Nacionais / Regionais: • *Workshop* em Engenharia de Requisitos (WER) • Simpósio Brasileiro de Engenharia de *Software* (SBES)
Critérios (filtros de busca)	• Data publicação: entre janeiro/1988 e janeiro/2018. • Publicado em inglês ou português. • Publicado em conferências ou congressos • Não duplicados. • Somente textos completos. • Ênfase em: engenharia de requisitos (RE). • Somente trabalhos revisados por pares.
Bases de Dados	A pesquisa será realizada na web de forma a encontrar os *websites* de cada conferência e extrair os trabalhos a partir dos anais (*Proceedings)* de cada congresso e/ou conferência.

Tabela 76 – Congressos selecionados, critérios e estratégia de busca
Fonte: o autor

A ideia básica para escolha dos congressos (Tabela 76) foi a de selecionar eventos nacionais e internacionais que tenham como tema principal a engenharia de requisitos:

- *International Requirements Engineering Conference* (RE): promovida pela IEEE em parceria com a IREB esta é a maior conferência internacional realizada **especificamente** sobre RE;
- *Workshop* em Engenharia de Requisitos (WER): Este workshop equivale a conferência RE em termos nacionais / regionais. Originalmente o WER era um workshop Brasileiro, porém integrou-se nos últimos anos ao Congresso Ibero-Americano de Engenharia de *Software* (CIBSE).

A segunda decisão foi a de escolher conferências de áreas mais gerais que tenham dentre seus temas o assunto engenharia de requisitos:

- *International Conference on Software Engineering* (ICSE): O evento ICSE é promovido pela IEEE e pela ACM e é considerada a maior conferência mundial de Engenharia de *Software*.
- *Simpósio Brasileiro de Engenharia de Software* (SBES): Nacionalmente, foi escolhido o evento SBES, que é parte do Congresso Brasileiro de *Software* (CBSOFT) promovido pela SBC (Sociedade Brasileira de Computação).

A ideia de mesclar resultados de eventos nacionais e internacionais foi inspirada em abordagem similar adotada no SLM sobre “25 anos de engenharia de requisitos no Brasil” (OLIVEIRA et al., 2013).

Durante as pesquisas, notou-se que praticamente não há trabalhos publicados em congressos de sistemas (SE) que versem especificamente sobre temas ligados a engenharia de requisitos. Por esta razão, artigos de congressos de SE não foram priorizados nesta pesquisa.

D. Pesquisar e extrair trabalhos de conferências conforme critérios definidos

Nesta etapa foi realizada a pesquisa e a extração dos trabalhos conforme as definições da etapa “C”, obtendo-se inicialmente as quantidades **brutas** de trabalhos mostrados na Tabela 77.

Anos	ICSE	SBES	RE	WER	Total
1988	43	15			58
1989	36	27			63
1990	46	18			64
1991	43	17			60
1992	30	22			52
1993	48	23			71
1994	42	30	33		105
1995	32	23			55
1996	61	21	30		112
1997	85	38			123
1998	64	21	32	14	131
1999	103	25		9	137
2000	142	20	33	14	209
2001	133	39		20	192
2002	97	40	52	22	211
2003	106	20	62	22	210
2004	106	19	46	22	193
2005	138	20	71	23	252
2006	172	27	65	19	283
2007	89	24	62	22	197
2008	104	22	58	24	208
2009	69	24	54	16	163
2010	195	22	61	15	293
2011	191	35	50	17	293
2012	232	24	51	17	324
2013	206	17	67	13	303
2014	116	18	64	12	210
2015	270	21	60	15	366
2016	259	16	64	13	352
2017	309	42	81	14	446
Total	3567	730	1096	343	5736

Tabela 77 – Quantidades brutas de trabalhos de conferências / congressos
Fonte: o autor

Os trabalhos foram extraídos a partir dos anais das conferências e congressos selecionados, em formato de referências bibtex (.bib), agrupados, convertidos e registrados em uma tabela *Excel* (.xlsx), visando a futura análise e seleção nas etapas “F”, “G” e “H”. Foram removidos da lista trabalhos duplicados e documentos que não podem ser considerados como artigos completos (*e.g.*: tutoriais, *keynotes*, *panel-sessions*, arquivos de prefácio, índices etc.).

Para os congressos mais gerais ICSE e SBES (Tabela 77) acima, **não** foi aplicado nesta etapa o critério de identificar artigos **com ênfase em**

engenharia de requisitos. Este critério foi aplicado posteriormente na etapa “F. seleção geral dos trabalhos”.

E. Definir critérios de seleção dos trabalhos

Nesta etapa foram definidos os critérios para seleção dos trabalhos a serem selecionados nas etapas “F”, “G” e “H”, conforme resumido na Tabela 78. Além dos critérios aplicados anteriormente, que chamaremos de “Critérios de Seleção Geral”, foram definidos “Critérios específicos” para selecionar os trabalhos durante a fase de leitura de títulos e resumos.

Critérios de Seleção Geral	• Data de publicação: entre janeiro/1988 e janeiro/2018. • Publicado em inglês ou português. • Publicado em conferências ou congressos • Não duplicados. • Somente textos completos. • Ênfase em: engenharia de requisitos • Somente trabalhos revisados por pares.
Critérios Específicos para Seleção por título e resumo	• Mesmos critérios da seleção geral. • Deve se caracterizar por conter tópicos de interesse sobre algum dos seguintes temas: modelagens gráficas ou visuais de requisitos, diagrama de requisitos, notações gráficas ou visuais de requisitos, modelagens de requisitos baseadas em modelos.
Critérios Específicos para Análise do texto completo	• Apresentar uma modelagem gráfica, visual ou notação/diagrama de requisitos, para a aplicação em *software* ou sistemas, incluindo ou não: métodos; linguagens de especificação; ferramentas; padrões ou normas internacionais; metamodelos. • Poderá ser um trabalho de caráter teórico e/ou empírico (prático e/ou experimental).

Tabela 78 – Critérios de Seleção dos Trabalhos
Fonte: o autor

F. Realizar a seleção geral dos trabalhos

A seleção geral dos trabalhos foi realizada conforme critérios definidos na etapa “E”, aplicados aos trabalhos obtidos a partir da busca por expressões de busca (Tabela 75) e congressos e conferências (Tabela 77), de forma a obter somente trabalhos de Engenharia de requisitos, reunidos na Tabela 79.

Anos	ICSE	RE	SBES	WER	Artigos selecionados por expressões de Buscas	Total Geral
1988	4				2	6
1989	3		1		2	6
1990	3					3
1991	1				3	4
1992	1		2		3	6
1993	4				5	9
1994	4	33	4		2	43
1995	5		3		2	10
1996	2	30	3		2	37
1997	8		5		8	21
1998	4	32	1	14	6	57
1999	4		2	9	5	20
2000	5	33	2	14	5	59
2001	8		2	20	9	39
2002	6	52	2	22	4	86
2003	2	62	5	22	10	101
2004	5	46		22	11	84
2005	11	71	3	23	29	137
2006	6	65	2	18	18	109
2007	3	62	1	22	17	105
2008	4	58	4	21	24	111
2009	2	53	1	15	32	103
2010	8	61	1	13	31	114
2011	7	50	3	16	27	103
2012	5	51	2	14	29	101
2013	9	62	2	11	39	123
2014	5	64	1	12	40	122
2015	1	60	2	15	42	120
2016	9	62	1	13	40	125
2017	5	81	4	14	23	127
Total Geral	144	1088	59	330	470	2091

Tabela 79 – Quantidades de trabalhos de RE selecionados (seleção geral).
Fonte: o autor

Todos os trabalhos dos congressos RE e WER foram mantidos para análise posterior por serem de eventos específicos de engenharia de requisitos.

G. Realizar seleção por título e resumo

A seleção dos trabalhos por “título e resumo” foi realizada conforme os critérios definidos na etapa “E”, aplicados aos trabalhos reunidos na Tabela 79.

Para garantir a identificação do máximo possível de modelos gráficos de requisitos, foi realizado o *download* (arquivo .pdf) de todos os trabalhos que apresentassem potencial (mesmo que mínimo) de se enquadrar nos critérios definidos para seleção por título e resumo.

Após obter cada trabalho, leitura do título e do resumo (*abstract*) de todos os trabalhos, bem como uma breve revisão do texto do artigo, suas figuras, tabelas e resultados de forma a verificar se cada artigo deveria ser selecionado.

O sumário das quantidades dos trabalhos selecionados ao final desta etapa é mostrado na Tabela 80.

Anos	ICSE	RE	SBES	WER	Artigos selecionados (expressões de buscas)	Total Geral
1994		2				2
1995	1					1
1996		1				1
1997	2		1		1	4
1998	1					1
1999			1			1
2000		1		2		3
2001	1		1	2	1	5
2002		5		3	1	9
2003		4	2	3	1	10
2004		7		5	2	14
2005	1	5	2	3	6	17
2006		3		1	1	5
2007				3	3	6
2008		2		1	3	6
2009		3		4	5	12
2010		3		2	3	8
2011		1			4	5
2012		5		4	4	13
2013		7		3	7	17
2014		3		3	10	16
2015		4	1	2	5	12
2016		9		1	6	16
2017		3	1	2	6	12
Total Geral	6	68	9	44	69	196

Tabela 80 – Quantidades de trabalhos de RE (seleção título/resumo)
Fonte: o autor

H. Analisar texto completo

Nesta etapa foi realizada a leitura completa dos artigos selecionados na etapa anterior. Foi realizada também uma extração (manual) e síntese das informações relacionadas as questões de pesquisa.

Durante o processo de análise completa, foi realizada nova filtragem segundo os critérios definidos anteriormente, de forma que somente **185** artigos foram selecionados a partir dos **196** originalmente previstos (Tabela 80).

Os 11 (onze) artigos **excluídos** durante o processo de análise completam não apresentavam modelos gráficos de requisitos claramente, seja na forma de figuras representativas do modelo, pela presença de metamodelos ou por meio de um detalhamento claro do modelo.

A Figura 73 apresenta uma visão geral e numérica das quantidades de trabalhos envolvidas na execução do processo de SLM.

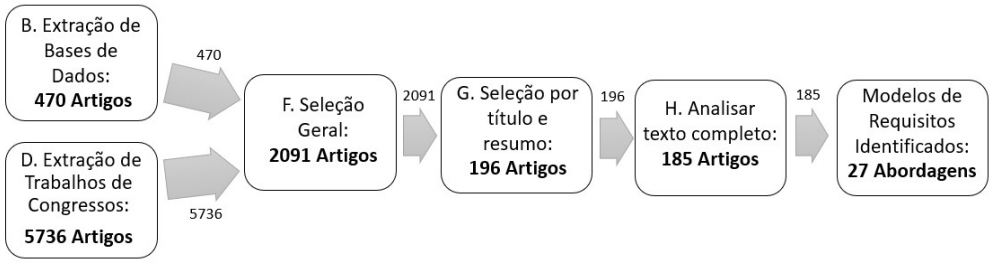


Figura 73 – Visão geral e numérica da execução do processo de *SLM*
Fonte: o autor

V. REFERÊNCIAS DO APÊNDICE A

KITCHENHAM, Barbara A.; BUDGEN, David; PEARL BRERETON, O. **Using mapping studies as the basis for further research - A participant-observer case study**. Information and Software Technology, v. 53, n. 6, p. 638–651, 2011.

KITCHENHAM, B.; CHARTERS, S. **Guidelines for performing systematic literature reviews in software engineering**, Keele University, United Kingdom, Tech. Rep. EBSE-2007-01, 2007.

OLIVEIRA, Karolyne; PIMENTEL, João; SANTOS, Emanuel; DERMEVAL, Diego; GUEDES, Gabriela; SOUZA, Cleice; SOARES, Monique; CASTRO, Jaelson F. B.; ALENCAR, Fernanda M. R.; SILVA, Carla. **25 years of Requirements Engineering in Brazil: a systematic mapping**. *In*: XVI Workshop em Engenharia de Requisitos, p. 118–134, 2013.

PETERSEN, Kai; FELDT, Robert; MUJTABA, Shahid; MATTSSON, Michael. **Systematic Mapping Studies in Software Engineering**. 12th International Conference on Evaluation and Assessment in Software Engineering (EASE), v. 17, p. 10, 2008.

APÊNDICE B – DESCRIÇÃO DE MODELOS GRÁFICOS EM RE

I. Quadro resumo para descrição dos modelos gráficos em RE

Este documento apresenta uma descrição resumida com as principais informações e características dos modelos gráficos em RE identificados, segundo o padrão de **quadro resumo** definido na Tabela 81.

Informação	Descrição da Informação
Acrônimo (sigla)	Acrônimo, sigla ou combinação de letras utilizada para representar cada modelo de requisitos identificado.
Nome (sistema gráfico de requisitos)	Nome do "sistema (modelo) gráfico de requisitos" identificado. Cada sistema poderá conter um ou mais tipos de diagramas para representação de requisitos.
Artigos (artigos relacionados ao modelo)	Lista dos artigos que passaram pelo processo de "Análise Completa" e referenciados no "APÊNDICE C": • Artigos principais (prefixados com "P") • Referências complementares (prefixados com "C")
Tipo (abordagem Gráfica)	Tipo de abordagem gráfica proposta no modelo: • Cenário (*Scenario-Based*); • Cluster (*Cluster-Based*); • Objetivos (*Goal-Based*); • Graph (*Graph-Based*); • Hierarquia (*Hierarchy-Based*); • Multimídia (*Multimedia-Based*); • *Problem Frames* (*Problem Frames-Based*); • Processo (*Process-Based)*; OBS 1: esta divisão proposta pelo autor deste SLM visa classificar e agrupar as abordagens existentes em termos quantitativos, sempre com base em informações fornecidas pelos autores de cada modelo. OBS 2: há modelos que se enquadram em mais de uma das classificações acima. Neste caso se optou por associar cada modelo ao tipo de abordagem preponderante definida pelos autores de cada modelo.

Informação	Descrição da Informação
Descrição (resumo)	Breve resumo visando identificar e caracterizar os aspectos mais relevantes de cada abordagem gráfica. O tamanho máximo para esta descrição é 10 linhas de texto.
Proposto em (Ano)	Ano de proposição da abordagem gráfica.
Proponente	Lista dos proponentes originais da abordagem segundo o artigo origem (autores)
Artigo origem	Título do artigo ou documento de proposta original do modelo identificado e referência (APÊNDICE C).
Linguagem	Linguagem de especificação de requisitos - definição (ISO/IEC/IEEE 24765, 2010) - proposta pelos autores dos modelos ou integradas por terceiros ao modelo gráfico.
Ferramenta	Ferramentas de modelagem – definição (ISO/IEC/IEEE 24765, 2010) – compostas de s*oftwares* completos, *plugins* ou assemelhados, associados a cada modelo gráfico de requisitos. Podem suportar mais de um modelo gráfico.
Metodologia	Metodologias – definição (ISO/IEC 24744, 2007) - propostas de forma associada ou integrada ao modelo gráfico.
Padrão (Norma Internacional)	Padrão ou norma internacional (*standard)* – definição (ISO/IEC Guide 2, 2004) associada ao modelo gráfico.
Diagrama	Nome dos diagramas ou modelos gráficos individuais de representação de informações em RE. Os diagramas podem ser **principais** (elaborados pelos autores originais do modelo) ou **extensões** (elaborados por terceiros. É importante notar que não necessariamente o autor de cada artigo **nomeia** elementos gráficos como sendo diretamente requisitos. É comum a utilização de termos **similares** (e.g.: “objetivos” ao invés de requisitos).
Metamodelo	Metamodelo ou ontologia – definições (ISO/IEC 24744, 2007) (ISO/IEC/IEEE 24765, 2010) – propostas para o sistema gráfico de requisitos.

Tabela 81 – Padronização dos quadros de resumo para os modelos gráficos
Fonte: o autor

Além das informações do quadro resumo, cada modelo apresentará figuras com **diagramas exemplo** (somente diagramas principais – não extensões) e com a **notação** dos elementos gráficos do modelo (se disponível) Além disso, é importante destacar que:

a) Informações indeterminadas ou indisponíveis sobre o modelo estão marcadas com “N” no campo específico do quadro resumo;
b) múltiplas informações podem estar associadas a um mesmo modelo (e.g.: múltiplas ferramentas, metodologias, ontologias etc);
c) Cada um dos modelos identificados foi estudado individualmente para elaboração do quadro resumo correspondente. Nesse processo identificou-se a necessidade de referências mais completas, normalmente elaboradas pelos autores originais de cada abordagem (relacionadas no “APÊNDICE C – referências complementares”);
d) Este trabalho não pretende esgotar todos os detalhes de cada modelo. O objetivo é apresentar aspectos básicos que permitam caracterizar cada abordagem gráfica e mostrar a aplicabilidade de cada representação, bem como responder as questões de pesquisa propostas.

II. Descrição dos modelos gráficos em engenharia de requisitos

1. i*

Acrônimo	i*
Nome	i* Goal Model
Artigos	P011, P019, P020, P021, P022, P024, P026, P027, P029, P030, P032, P033, P035, P036, P047, P048, P049, P050, P052, P053, P061, P062, P063, P064, P065, P066, P067, P068, P069, P070, P071, P072, P073, P074, P075, P076, P077, P078, P079, P080, P081, P082, P083, P084, P085, P086, P087, P088, P089, P090, P091, P092, P093, P107, P194 C022, C035, C036, C037, C038
Tipo	Objetivos (*Goal Based*)
Descrição	i* é uma abordagem orientada a agentes (e a objetivos) que pode ser usada em RE, reengenharia de processos de negócio,

	análise de impacto organizacional e modelagem de processo de *software* [C036]. Na etapa de *early requirements* é usada para modelar agentes, suas intenções, inter-relações e o ambiente organizacional do sistema futuro esperado [C022]. Na etapa de *late requirements* pode ser usada para propor novas configurações do sistema e novos processos, bem como para avaliar se o sistema irá atender aos requisitos funcionais e não funcionais dos *stakeholders* [C022].
Proposto em	1995
Proponente	Eric S. K. Yu
Artigo origem	[C035] *Modelling Strategic Relationships for Process Reengineering (PHD Thesis)*
Linguagem	*Structured Modal Action Logic (MAL) (integration)* [P050]
Ferramenta	*Descartes Architect CASE-Tool* [P080], *Goal to Architecture tool (GATO)* [P029], *GOOD* [P086], *GrowingLeaf* [P067], *HUCRE* [P068], *iPref-R framework* [P033], *iStarML Tool* [P072], *i*ToNuSMV* [P069], *JGOOSE* [P073] [P079], *NòmosBPMN* [P089], OME [P088], *OpenOME* [P048] [P081], *REDEPEND* [P047], *RE-Tools - StarUML plugin* [P107], *Sira Framework* [P052]
Metodologia	*AIRDoc-i* Process* [P075], *Analytic Hierarchy Process (AHP)* [P026], *AWARE (Analysis of Web Applications Requirements Engineering)* [P066], *DHARMA (Discovering Hybrid ARchitectures by Modelling Actors) Method* [P063], *Eri*c methodology* [P071], *GORE (Goal Oriented Requirements Engineering)* [P021] [P022] [P024] [P032] [P035], *HUCRE (Human-Centred Requirements Engineering) Method* [P068], *RESCUE Process* [P047]
Padrão	N
Diagrama	<u>*Principais*</u>*:* *i* Strategic Dependency Model (SD),* *i* Strategic Rationale Model (SR),* <u>*Extensões*</u>*:* *i* Goal Models + e3-value model,* *i* Goal Models + Functional Requirements allocation,* *i* Goal models + temporal constraints,* *i* Goal Models + Aspects,* *i* Goal Models + BPMN* (*Business Process Modeling Notation*)*,* *i* Goal Models + pUML (Precise UML) Use Cases,* *i* Goal Models + UML Use Cases,*

	i goal models mapped to UML Class Diagram,* *i* SD and SR Goal Models tranformed to User Stories* *i* SD and SR Goal Models applied to Web applications,* *i* SR Model + Users histories (Extreme Programming - XP)* *i* Goal Models + (CI-nets) conditional importance networks*
Metamodelo	*i* Metamodel* [C037] [P088]*, GORO (Gore Oriented Requirements Ontology)* [P036]*, User story structuring Metamodel* [P080]

Tabela 82 – Modelo i*
Fonte: o autor

Diagrama(s):

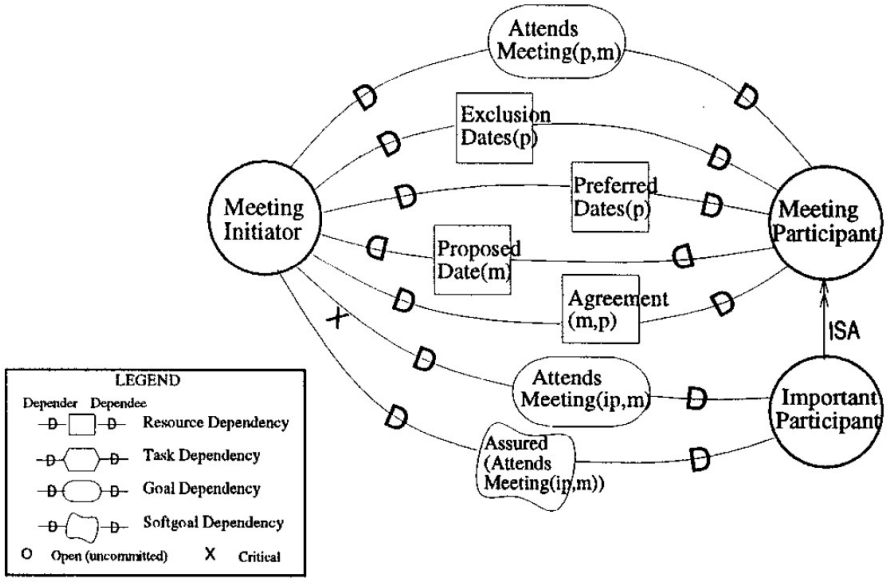


Figura 74 – i* – Modelagem SD de um agendamento de reunião
Fonte: APÊNDICE C [C036]

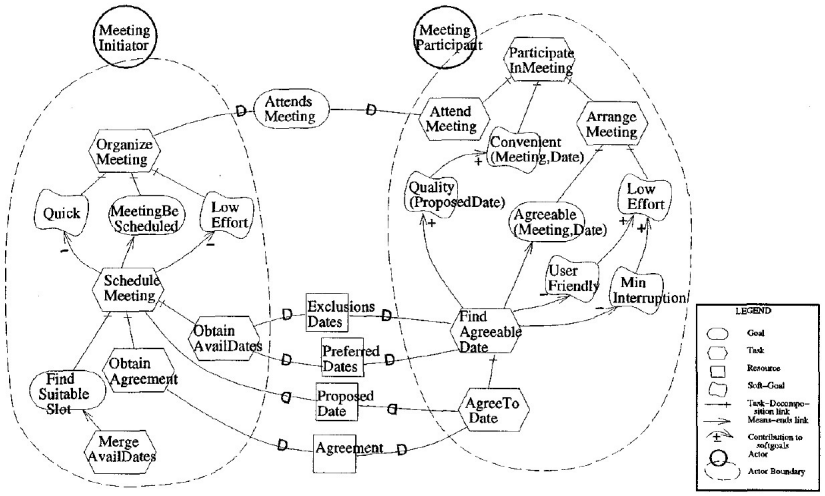


Figura 75 – i* – Modelagem SR de um agendamento de reunião

Fonte: APÊNDICE C [C036]

Figura 76 – i* – Notação (Sumário)
Fonte: APÊNDICE C – Adaptado de [C038]

2. Tropos

Acrônimo	Tropos
Nome	Tropos Goal Model
Artigos	P019, P023, P033, P036, P054, P055, P056, P057, P058, P059, P060, P135, P136, P152, P153, P154, P155, P156, P157, P158, P159, P161, P162, P189, P190, P193 C019, C020
Tipo	Objetivos (*Goal Based*)
Descrição	Tropos é uma metodologia que tem o objetivo de modelar sistemas de *software* orientados a agentes [C020], incluindo etapas tais como *Early Requirements, Late Requirements*, projeto de arquitetura, projeto detalhado e implementação Tropos possui elementos de notação e conceitos derivados do modelo i*. [C019]. As atividades de modelagem da metodologia Tropos consistem em modelagem de agentes modelagem de dependências e modelagem de objetivos [C019].
Proposto em	2004
Proponente	Paolo Giorgini; Manuel Kolp; John Mylopoulos; Marco Pistore
Artigo origem	[C020] *The Tropos Methodology*

Linguagem	*Formal Tropos* [P055] [P193]*, Tropos4AS Language* [P056], *Tropos XML Specification Language (Extension)* [P155], *Tropos integration to OASIS formal Language* [P157]
Ferramenta	*GR-Tool* [P059]*, RE-context CASE-Tool* [P023], *Secure Tropos UML-Sec* [P060]*, Si* [P135], ST-Tool* [P058] [P153]*, S&D Tropos* [P136]*; TAOM4e* [P054] [P056] [P059], *T-Tool* [P055] [P193]
Metodologia	*GORE (Goal Oriented Requirement Models)* [P021] [P023] [P152]*, Tropos Methodology* [P054] [P055] [P154] [P156] [P157] [P158] [P159] [P193]*, Tropos4AS (Tropos for Adaptative Systems)* [P056]*, Secure Tropos Methodology* [P057] [P058] [P153]*, Secure Tropos + UMLSec* [P060]*, GAM (Goal Argumentation Model)* [P152]*, GAIA Methodology* [P159]*, MESSAGE Methodology* [P159]*, Secure i* (tropos)* [P135] [P136]
Padrão	N
Diagrama	*Principais:* *Tropos Goal Model - Actor Diagram,* *Tropos Goal Model - Goal Diagram* *Extensões:* *Tropos Goal Model for Adaptative Systems,* *Tropos Contextual Goal Model,* *Security and Privacy Catalogue (Reference Diagram + Organizational Diagram),* *Execution Dependency Diagram + Permission Delegation Diagram,* *Secure Tropos Goal model,* *Secure Tropos Goal model with security and legal dependency,* *Tropos Goal Refinement Trees,* *Tropos Goal Model + Use Case (UML)* *Tropos Goal Model + (CI-nets) conditional importance networks*
Metamodelo	*Tropos Metamodel* [P059] [P190]*, GORO (Gore Oriented Requirements Ontology)* [P036]

Tabela 83 – Modelo Tropos
Fonte: o autor

Diagrama(s):

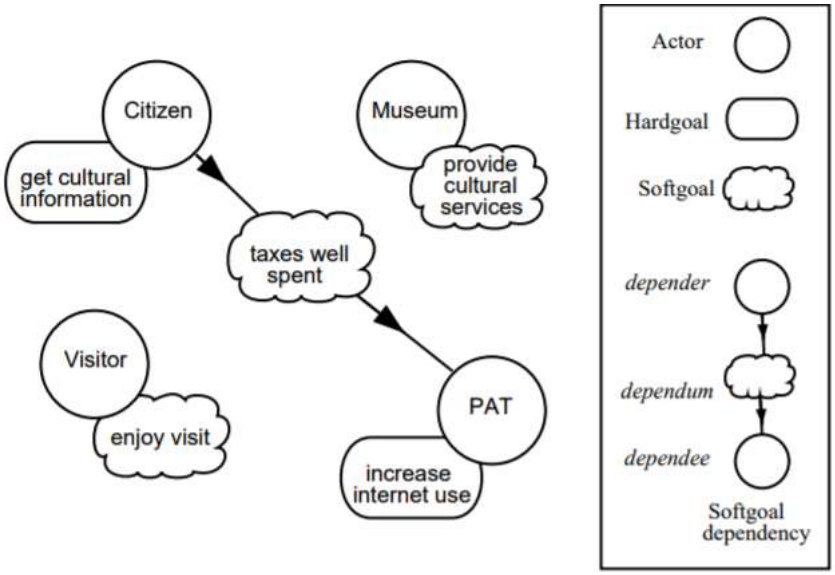


Figura 77 – TROPOS – Exemplo – modelagem de atores (Actor Diagram)
Fonte: APÊNDICE C [C019]

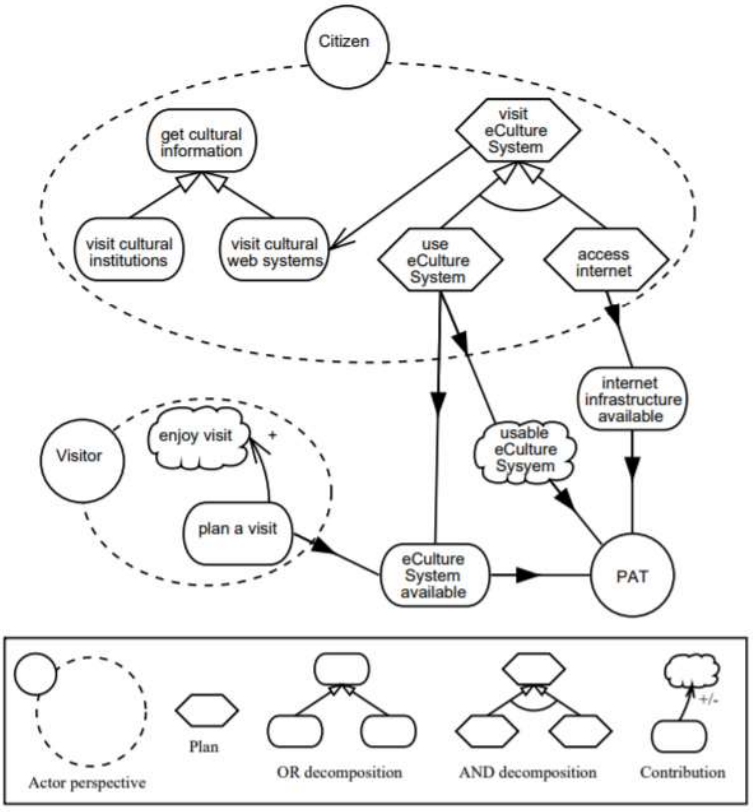


Figura 78 – TROPOS – Modelo de objetivos - cidadãos e turistas
Fonte: APÊNDICE C [C019]

3. Knowledge Acquisition on Automated Specification - KAOS

Acrônimo	KAOS
Nome	*Knowledge Acquisition on autOmated Specification* (KAOS) Goal Model (ou *Keep All Objects Satisfied*)
Artigos	P010, P014, P016, P019, P025, P028, P031, P032, P034, P035, P036, P061, P088, P095, P096, P097, P098, P099, P100, P101, P102, P103, P104, P107, P108 C021, C022, C023, C024, C025, C026, C063
Tipo	Objetivos (*Goal Based*)
Descrição	KAOS é uma metodologia de aquisição e modelagem de requisitos orientada a objetivos [C021], descrita como um *framework* multiparadigma [C023] com diferentes níveis de expressão: semi-formal para modelagem e estruturação de objetivos; qualitativo para seleção entre alternativas; formal, se necessário um raciocínio mais preciso [C022]. A modelagem gráfica e notação mais atualizada de KAOS está materializada na ferramenta *Objectiver* ® da empresa *Respect-IT*®, e é composta de modelos de: objetivos, responsabilidades, operacional e objetos [P088].
Proposto em	1993
Proponente	Anne Dardenne; Axel Van Lamsweerde; Stephen Fickas
Artigo origem	[C021] *Goal-directed requirements acquisition*
Linguagem	*KAOS Formal Language* [P016] [P096] [P099] [P103] [P157], *SCR (Software Cost Reduction) Language (integration)* [P099], *KAOS Language + KBRE (Knowledge Based RE) Language* [P104], *RELAX Language integration* [P108]
Ferramenta	*FAUST Toolset* [P100], *GRAIL* [P016] [P098], *k-tool* [P101], *Objectiver* [P031] [P034] [P088], *OMEGA2 profile UML/SysML Profile + Ifx Toolset* [P108], *QARAT* [P096], *REInDetector tool* [P104], *RE-Tools - StarUML plugin* [P107]
Metodologia	*KBRE (Knowledge Based Requirements Engineering) based on KAOS* [P104], *KAOS (Knowledge Acquisition in automated specification)* [P016] [P031] [P034] [P095] [P096] [P098] [P100] [P101] [P102], *KAOS + Scenarios* [P014], *GORE (Goal Oriented Requirements Engineering)* [P031] [P032] [P034] [P035] [P095], *KAOS derived to SCR (Software Cost Reduction) language* [P099], *RELAX Method* [P108], *UNC-Method* [P107]

Padrão	N
Diagrama	*Principais:* *KAOS (Goal Model + Object Model + Responsibility Model + Operational Model) models,* *Extensões:* *KAOS Models + HTN (Hierarchical Tasks Network) + PDDL 3.0,* *KAOS Models with Scenarios,* *KAOS Models + requirements animated model in FAUST Tool,* *KAOS Models with Alternatives,* *KBRE adapted from KAOS Goal Oriented Models,* *KAOS Models + SysML Textual Requirements*
Metamodelo	*KAOS Metamodel* [P096] [P088], *KBRE Metamodel* [P104], *SysML/KAOS Metamodel* [P108], *GORO (Gore Oriented Requirements Ontology)* [P036], *UML profile to support RE with KAOS [C063]*

Tabela 84 – Modelo KAOS
Fonte: o autor

Diagrama(s):

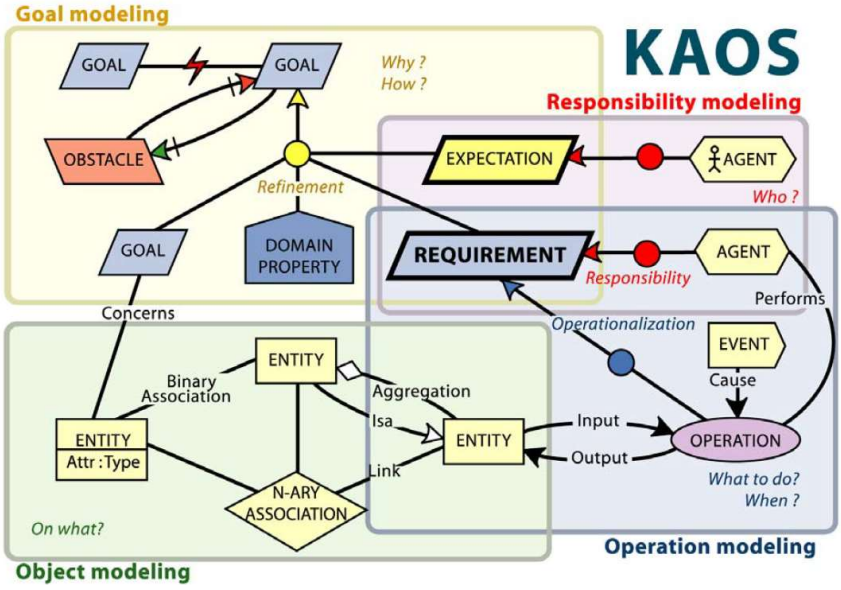


Figura 79 – KAOS – Modelos da Metodologia (*Objectiver Tool*)
Fonte: APÊNDICE C [C025]

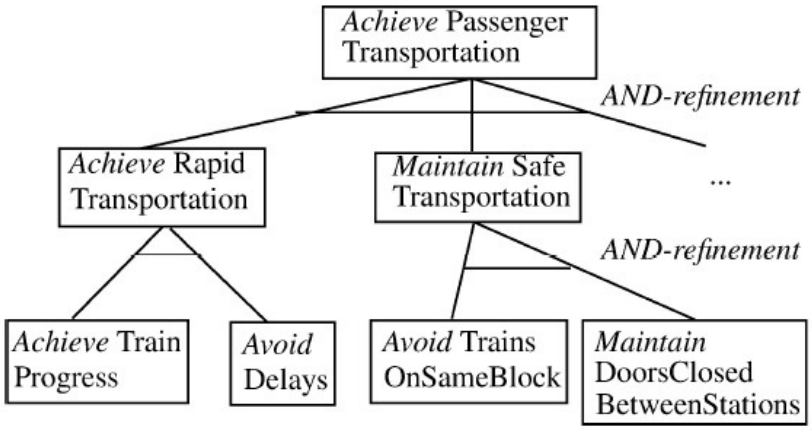


Figura 80 – KAOS – Refinamento de Objetivos para um Sistema de Trens
Fonte: Apêndice B [P016]

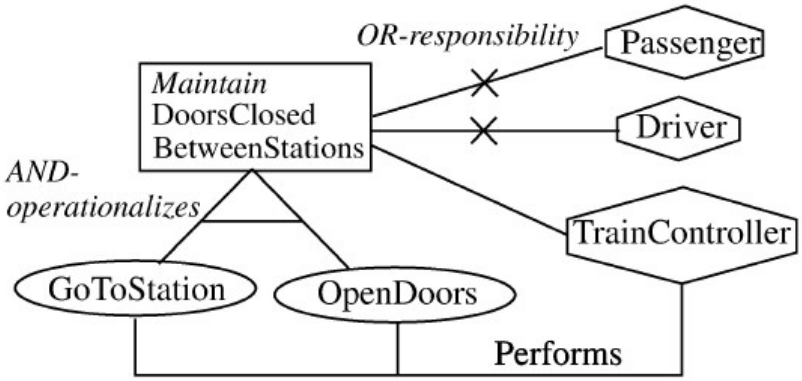


Figura 81 – KAOS – Diagrama de Operacionalização e Responsabilidade
Fonte: APÊNDICE C [P016]

4. Non-Functional Requirements - NFR

Acrônimo	NFR
Nome	NFR *(Non-Functional Requirements) Goal Model*
Artigos	P017, P018, P019, P022, P025, P035, P036, P037, P038, P039, P051, P053, P107, P109, P110, P111, P112, P113, P114, P115, P116, P117, P118, P119, P160 C022, C027, C028, C029, C030, C031
Tipo	Objetivos (*Goal Based*)
Descrição	NFR é um *framework* abrangente para representar requisitos não funcionais durante o processo de desenvolvimento [C029], na forma de uma abordagem orientada a processos, onde os requisitos não-funcionais são explicitamente representados como metas a serem obtidas [C028].

	NFRs podem ser representados por *softgoals*, que são objetivos sem uma definição clara de critério de satisfação, podendo ser subjetivos, relativos e interdependentes [C031].
Proposto em	1992
Proponente	John Mylopoulos; Lawrence Chung; Brian A. Nixon
Artigo origem	[C029] *Representing and Using Non-Functional Requirements: A Process-Oriented Approach*
Linguagem	*Σ language* (integração) [P051]
Ferramenta	LEL [P116], *NFR Assistant* [P017], *NDR-Tool - StarUML plugin* [P109], *rΣ Tool* [P053], *RE-Tools - StarUML plugin* [P107]
Metodologia	*GORE (Goal Oriented Requirements Engineering)* [P022] [P035 [P110], *GOals to Statecharts (GO2S)* [P037], *BPMNFR approach* [P111]
Padrão	N
Diagrama	<u>*Principal*</u>*:* *NFR SoftGoal Interdependency Graph (SIG),* <u>*Extensões*</u>*:* *NFR SIG Model tranformed to Statecharts,* *NFR SIG Model operationalized to BPD (from BPMN),* *NFR SIG Model integrated to Use Cases,* *NFR SIG Model integrated to Aspect Oriented Requirements*
Metamodelo	*NFR Metamodel with Scenarios* [P160], *GORO (Gore Oriented Requirements Ontology)* [P036], *Quality requirements modeling Metamodel* [P051], *Metamodel for Softgoal Interdependency Graph [P096]*

Tabela 85 – Modelo NFR
Fonte: o autor

Diagrama(s):

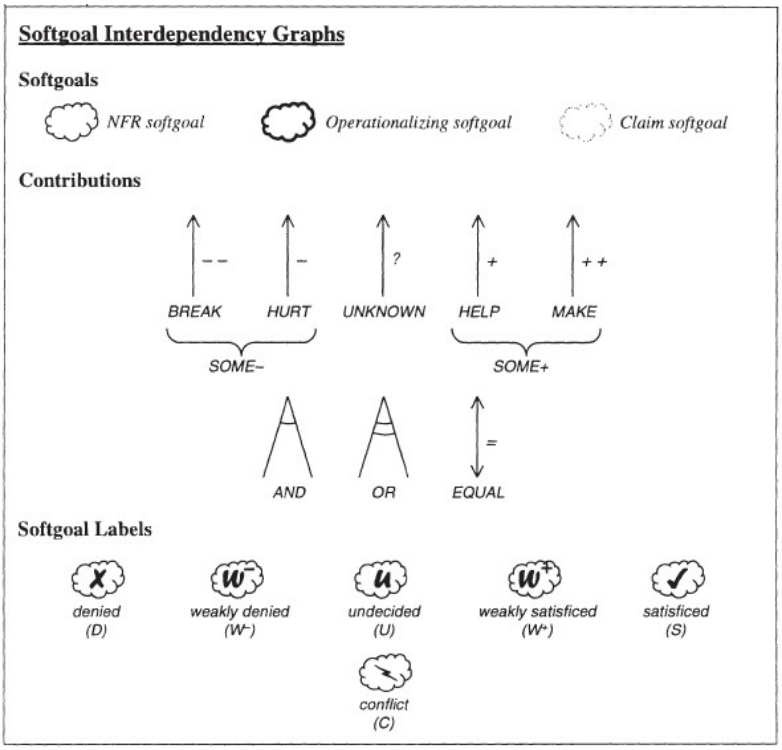


Figura 82 – NFR – *Softgoal Interdependency Graphs* (SIG) – Notação 1 de 2
Fonte: APÊNDICE C [C030]

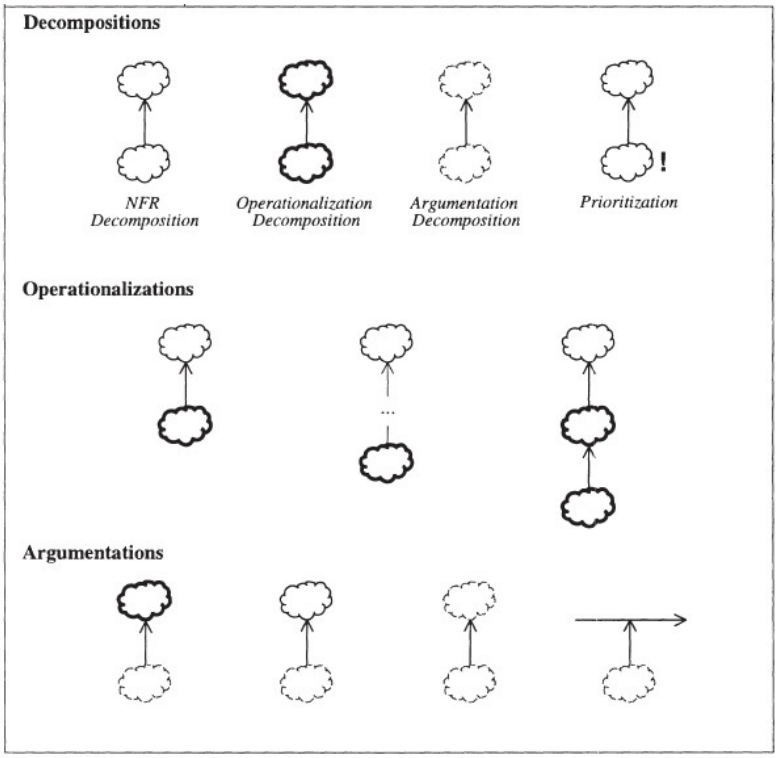


Figura 83 – NFR – *Softgoal Interdependency Graphs* (SIG) – Notação 2 de 2
Fonte: APÊNDICE C [C030]

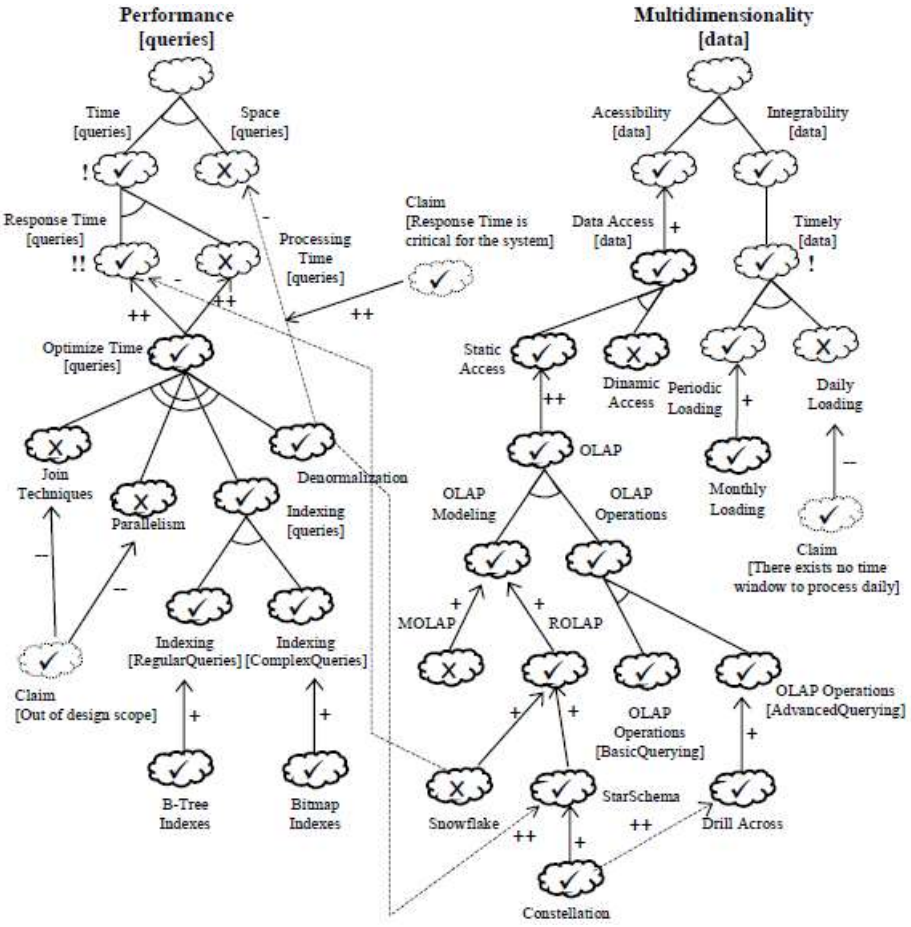


Figura 84 – NFR – Requisitos NFR para sistema *Data-Warehouse*
Fonte: APÊNDICE C [P051]

5. User Requirements Notation - URN

Acrônimo	URN
Nome	*User Requirements Notation* (URN)
Artigos	P002, P006, P007, P021, P035, P042, P043, P044, P045, P168, P169, P170, P171, P172, P173, P174, P175, P176, P177, P178, P179, P180, P181, P182 C007, C032, C033, C054, C055, C064
Tipo	Objetivos (*Goal Based*)
Descrição	URN é uma notação gráfica padronizada pelo ITU-T como norma internacional Z.151 em 2008 [C032], usada para modelar e analisar requisitos utilizando objetivos (*goals*) e cenários (*scenarios*), cujo objetivo é suportar a elicitação, análise, especificação e validação de requisitos [C007].

	URN combina duas sub-linguagens: **GRL** (*Goal-oriented Requirements Language*) para modelagem de atores e suas intenções e **UCM** (*Use Case Maps*) para descrição de cenários e arquiteturas [C007].
Proposto em	Inicialmente proposto ao ITU-T em 2000 [C054], foi padronizado em pelo ITU-T em 2008 [C032], com última versão de 2012 [C064];
Proponente	Diversos proponentes participaram do grupo focal na ITU-T [C054] para padronizar a URN, sendo possível destacar os seguintes participantes [C007]: J. Visser e J. Hodges (*Nortel Networks*); T. Gray (*Mitel Networks*); J. Hodges (2000 ITU-T *rapporteur*); D. Cameron (2001 ITU-T *rapporteur*) Daniel Amyot (2002 ITU-T *rapporteur*).
Artigo origem	[C054] URN *Focus Group in ITU-T*, [C055] GRL - *Goal-oriented Requirements Language,* [C049] UCM - *Use Case Maps for Object-oriented Systems*
Linguagem	*URN integrated to OCL constraints language* [P006], *URN integrated to ASPIC+ formal argumentation framework* [P045], URN + TUCM (Timed Use Case Maps) formal Semantics [P168], *URN (UCM Use Case Maps) formal semantics with ASM (Abstract State Machines)* [P184]
Ferramenta	*AoURN-based SPL framework (ASF)* [P006]*, jUCMNav* [P002] [P168] [P174] [P176] [P177] [P178] [P179] [P180] [P182]*, jUCMNav tool adapted for Aspects* [P007]*, jUCMNav extended to LEGAL-URN* [P044]*, jUCMNav integrated to ASPIC+ framework* [P045]*, JUCMNav integrated to TouchCore Tool* [P171]*, UCMNav* [P170] [P172] [P175]*, UCMNav integrated to LQNGenerator tool* [P172]*, WebGRE (WebURN and WebUCM)* [P169]
Metodologia	*CDD (Concern Driven Development)* [P007] [P178]*, GORE (Goal Oriented Requirements Engineering)* [P169 [P170] [P182], *URN extended to LEGAL-GRL* [P044]
Padrão	ITU-T Z.151 (URN) *Standard* [C032]
Diagrama	<u>*Principal*</u>*:* *URN (GRL Goal Model + UCM Use Case Maps),* <u>*Extensões*</u>*:* *URN + BPM Model,* *URN + CDD (Concern Driven Development) Support,* *URN + LQN (Layered Queueing Networks),* *URN (UCM Use Case Maps) + ASM (Abstract State Machines),* *URN Goal Model + Activity Theory,* *URN Goal Model + View Diagrams,*

	URN Goal Model + TUCM (Timed Use Case Maps), *URN Goal Model + WebUCM (Web Use Case Maps),* *URN Goal Model extended for Legal Requirements,* *URN Goal Model + AoUCM (Aspect Use Case Maps) + CDD (Concern Driven Development),* *URN Goal Model + AoUCM (Aspect Use Case Maps)*
Metamodelo	*Activity Theory Metamodel based on OCL (Object Contraints Language)* [P002], *Timed UCM core metamodel* [P168], *UCM availability metamodel* [P176], *GRL Abstract Metamodel* [P182]

Tabela 86 – Modelo URN
Fonte: o autor

Diagrama(s):

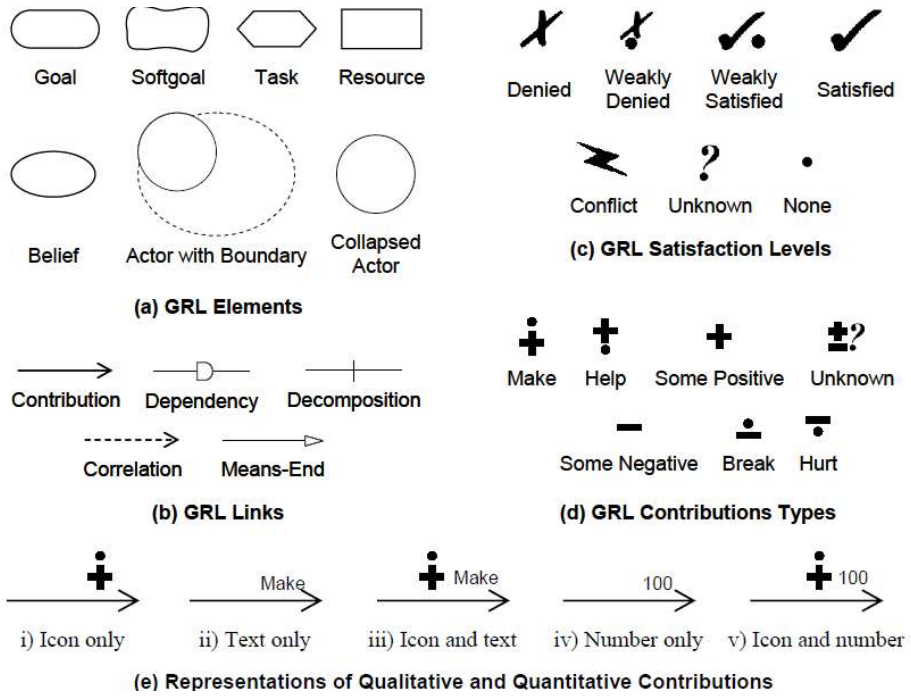


Figura 85 – URN – Elementos básicos da notação GRL
Fonte: APÊNDICE C [C033]

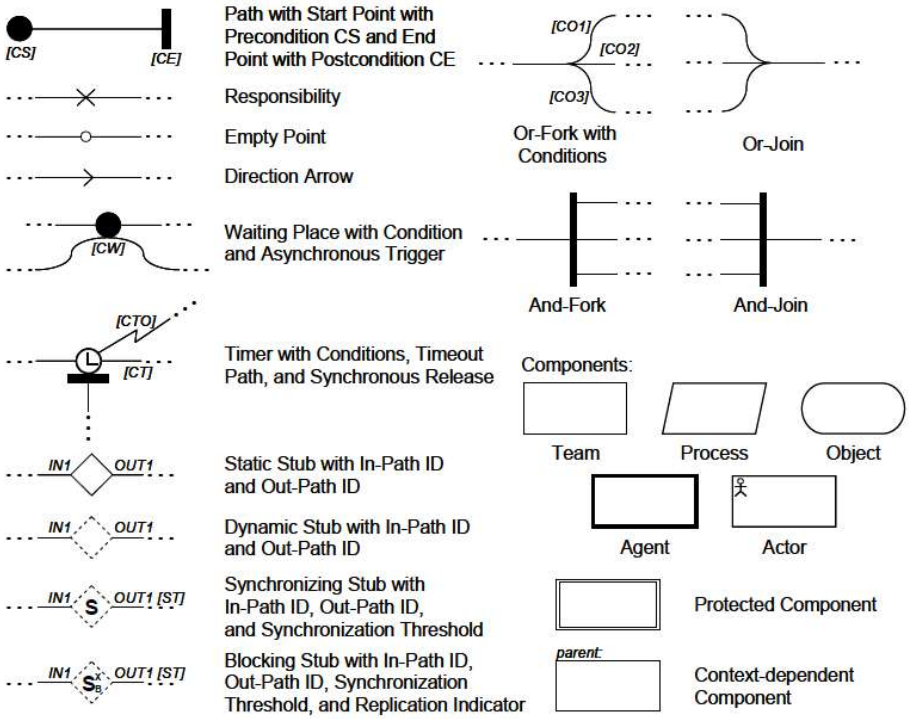


Figura 86 – URN – Elementos básicos da notação UCM
Fonte: APÊNDICE C [C033]

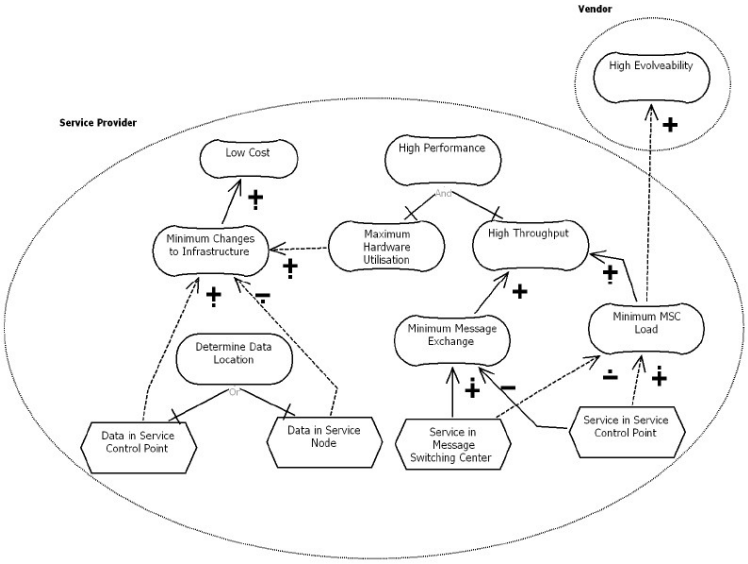


Figura 87 – URN – Exemplo – modelagem GRL
Fonte: APÊNDICE C [C033]

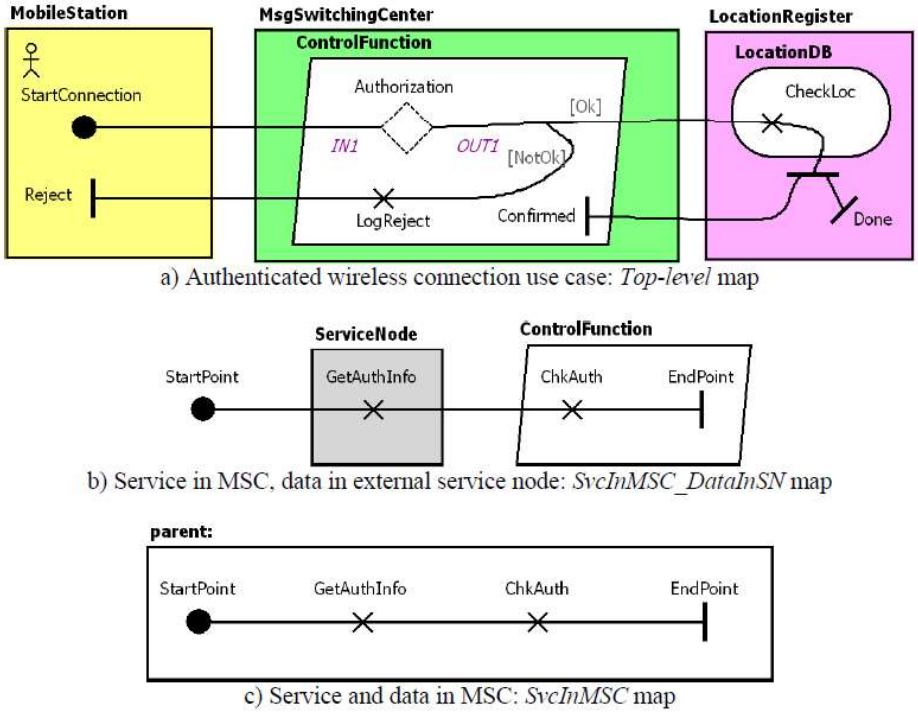


Figura 88 – URN – Exemplo – modelagem UCM
Fonte: APÊNDICE C [C033]

6. Systems Modeling Language - SysML

Acrônimo	SysML
Nome	*Systems Modeling Language* (*SysML*)
Artigos	P139, P140, P141, P142, P143, P144, P145, P146, P147, P148, P149, P150 C043, C044, C045, C046, C056
Tipo	Hierárquico
Descrição	SysML é uma linguagem de modelagem de sistemas (*hardware + software)*, criada na forma de uma profile da UML padronizada pela OMG em 2007 [C056] em colaboração com o *International Council on Systems Engineering* (INCOSE). A última versão (1.5) está disponível deste maio de 2017 [C043]. O *requirements diagram* (RD) da SysML contém requisitos e seus relacionamentos, além de potenciais conexões de rastreabilidade com outros diagramas e elementos de modelagem SysML.
Proposto em	2001
Proponente	INCOSE

Artigo origem	[C057] *UML For Systems Engineering RFP*
Linguagem	*SysML + ATLAS Transformation Language (integration)* [P142]*, SysML + TransML (Model Transformation Language)* [P145]
Ferramenta	*Papyrus Modeling Environment* [P142]*, Enterprise Architect (EA)* [C060]*, Magic Draw* [P149], *Visual Paradigm software* [P144]
Metodologia	*MDEReq (Model Driven Engineering for Requirement Management)* [P139]*, MBSE (Model Based Systems Engineering)* [P147 [P150]*, MBRE (Model Based Requirements Engineering)* [P149]*, CORAMOD (checklist-oriented requirements analysis modelling)* [P149]*, ACRE (Approach to Context-Based Requirements Engineering)* [P150]*, SysML Profile Extension for Safety Requirements* [P146]*, SysML Profile Extension for Prioritizing Requirements* [P148]
Padrão	SysML [C043]
Diagrama	*Principal:* *SysML Requirements Diagram (RD)* *Extensões:* *SysML RD + MARTE Profile Extension,* *SysML RD + URN (User Requirements Notation),* *SysML RD refined with Use Cases,* *SysML RD + Problem Domain Diagram,* *SysML RD extended for Safety Requirements,* *SysML RD + Requirements Definition View + Requirements Context View diagrams*
Metamodelo	*SysML Metamodel* [P108]*, ACRE Ontology* [P150]

Tabela 87 – Modelo SysML
Fonte: o autor

Diagrama(s):

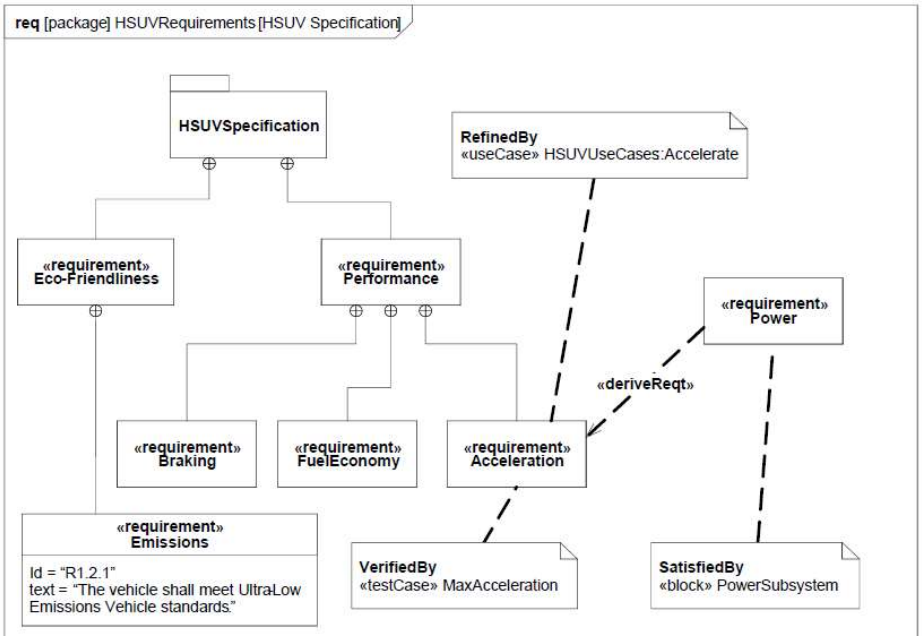


Figura 89 – SysML – Exemplo – Diagrama de Requisitos (RD)
Fonte: APÊNDICE C [C045]

Diagram Element	Notation
Containment Path	⊕ ⊕
Derivation Path	«deriveReqt» ---------->
Satisfaction Path	«satisfy» ---------->
Verification Path	«verify» ---------->
Refinement Path	«refine» ---------->
Trace Path	«trace» ---------->
Copy Path	«copy» ---------->

Figura 90 – SysML – Relacionamentos entre requisitos (*diagram paths)*
Fonte: APÊNDICE C – Adaptado de [C046]

7. UML Use Cases - UML UC

Acrônimo	UML UC
Nome	*UML Use Cases*
Artigos	P013, P090. P091, P092, P093, P186, P187, P188 C044, C047, C048, C058
Tipo	Cenário
Descrição	UML *Use Cases* (UC) é uma abordagem de modelagem orientada a cenários proposta por Jacobson e outros por volta de 1992 [C047] e é parte da UML 1.1 padronizada pela OMG em 1997 [C058]. A UML está na versão atual 2.5 [C044]. "Um UC é uma unidade coerente de funcionalidade externa visível provida por um classificador (chamado de sujeito) e expressa por sequências de mensagens trocadas entre o sujeito e um ou mais atores da unidade do sistema" [C048]. UCs não modela requisitos diretamente, porém há extensões de modelagem que propuseram isso de forma integrada.
Proposto em	1992
Proponente	Ivar Jacobson et al. (Use cases proposition only)
Artigo origem	[C047] *Object-Oriented Software Engineering: A Use Case Driven Approach*
Linguagem	*RPL (Requirements Pattern Language)* [P186]*, GBRAM (Goal Based Requirement Analysis Method)* + *CREWS* [P013]
Ferramenta	*AGG Tool* [P188]*, StarUML* [C061]*, Enterprise Architect* [C060]
Metodologia	*GBRAM (Goal Based Requirement Analysis Method integrated to UC Models)* [P013]
Padrão	Unified Modeling Language (UML) [C044]
Diagrama	*Principal:* *UML Use Case Model* *Extensões:* *Use Case models (Activity Diagrams) with Aspects and a Graph Transformation Tool*
Metamodelo	*UML Metamodel* [C044]

Tabela 88 – Modelo UML Use Cases
Fonte: o autor

Diagrama(s):

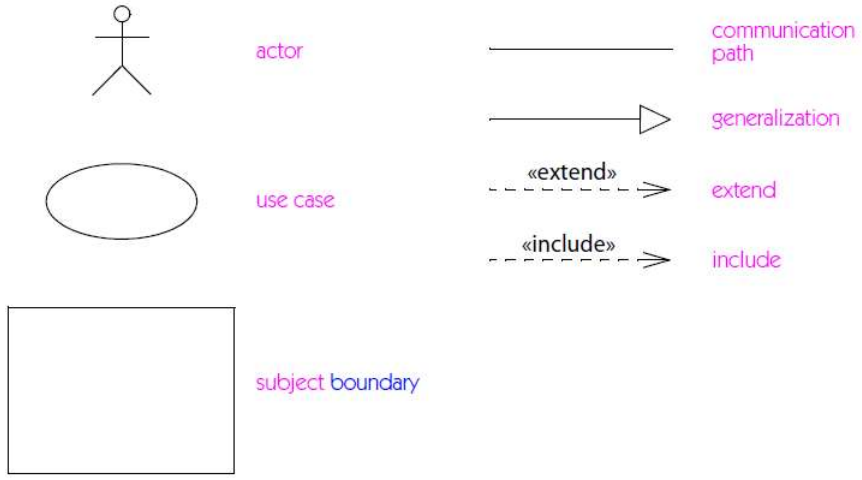


Figura 91 – UC – Notação – ícones em diagramas UC
Fonte: APÊNDICE C [C048]

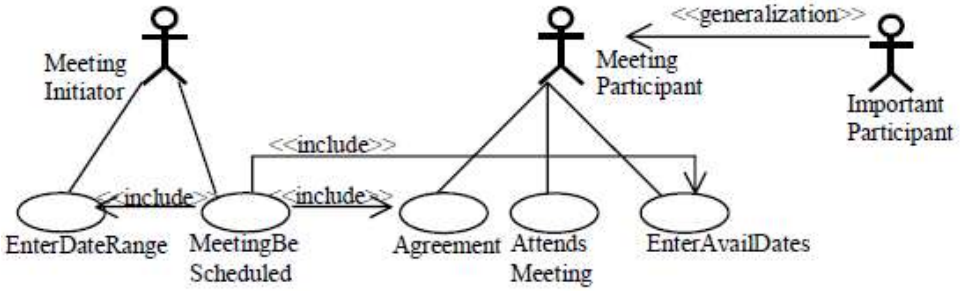


Figura 92 – UC – Exemplo – sistema de agendamento de reuniões
Fonte: APÊNDICE C [P093]

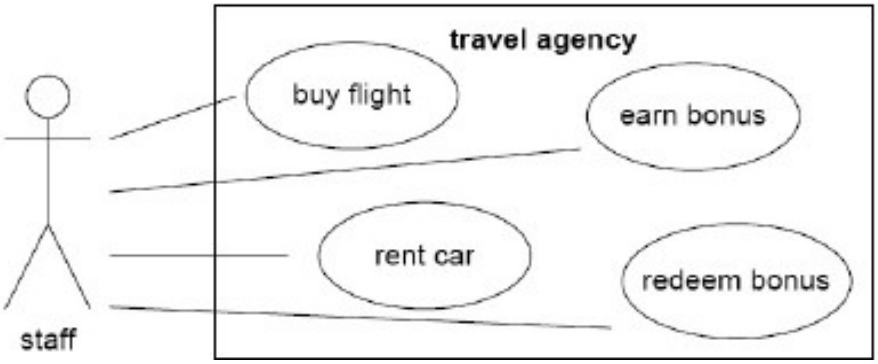


Figura 93 – UC – Exemplo – sistema de agência de viagens
Fonte: APÊNDICE C [P188]

8. Use Case Maps - UCM

Acrônimo	UCM
Nome	*Use Case Maps* (UCM)
Artigos	P163, P164, P183, P184, P185 C008, C009, C010, C011, C012, C013, C014, C015, C032, C049
Tipo	Cenário
Descrição	UCM é uma notação visual que modela cenários focados em representar o fluxo causal de uma estrutura de componentes do sistema [C007]. Um modelo UCM representa cenários compostos de responsabilidades que podem ser atribuídas a arquitetura do sistema [P163]. UCM é uma abordagem proposta em 1995 [C049] que evoluiu para fazer parte da notação URN, que é o *standard Z.151* da ITU-T oficializado em 2008 [C007].
Proposto em	1995
Proponente	Ray J. A, Buhr; Ron S. Casselman
Artigo origem	[C049] *Use Case Maps for Object-oriented Systems*
Linguagem	N
Ferramenta	*UCMNav* [P163] [P183], *jUCMNav* [P164] [P185]
Metodologia	N
Padrão	N
Diagrama	*Principal:* *Use Case Map Diagram* (*UCM*)
Metamodelo	*Abstract UCM Metamodel* [C032]

Tabela 89 – Modelo UCM
Fonte: o autor

Diagrama(s):

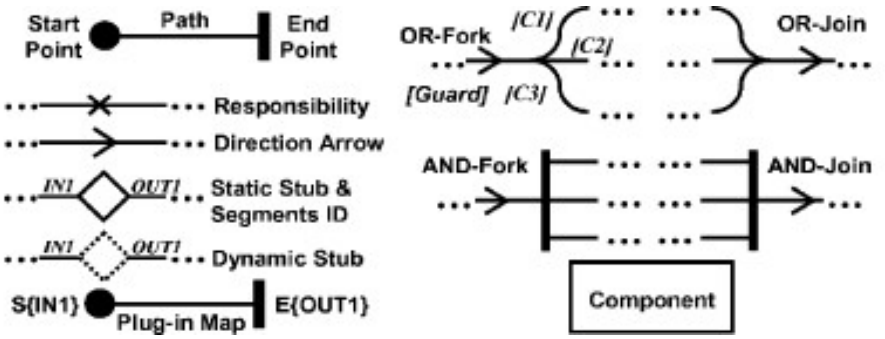


Figura 94 – UCM – Construtores Principais da Notação
Fonte: APÊNDICE C [P163]

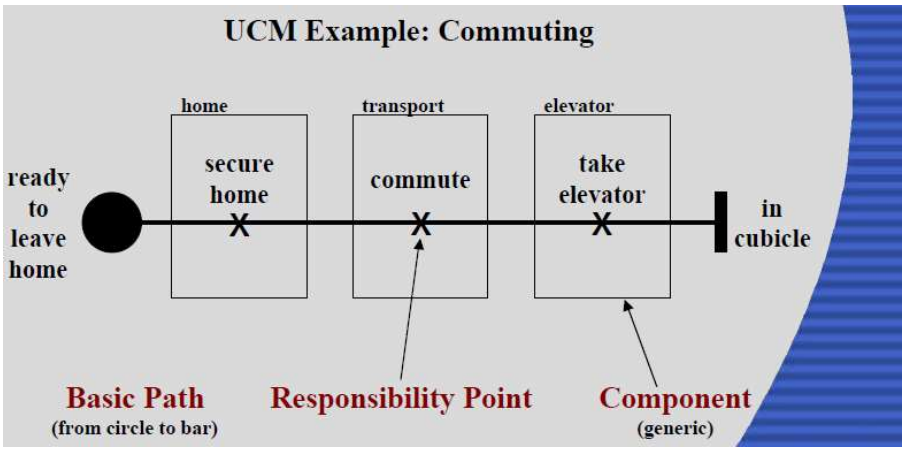


Figura 95 – UCM – Exemplo – processo de comutação
Fonte: APÊNDICE C [C015]

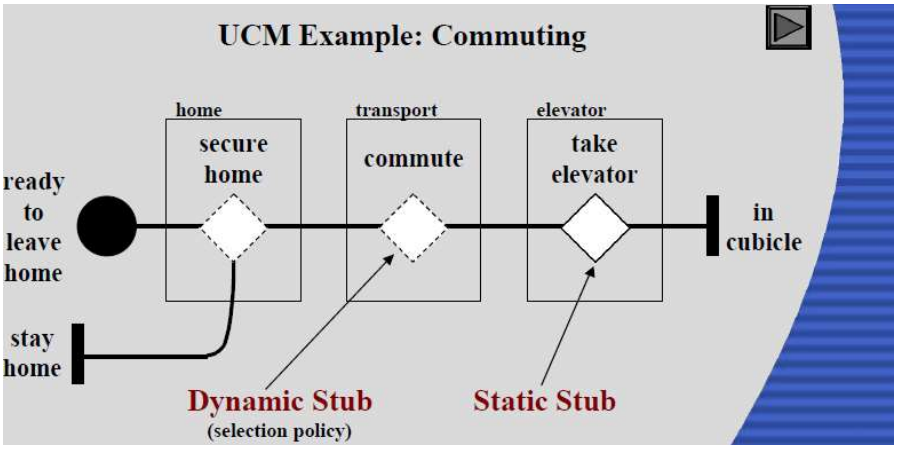


Figura 96 – UCM – Exemplo – comutação com *Dynamic Stub*
Fonte: APÊNDICE C [C015]

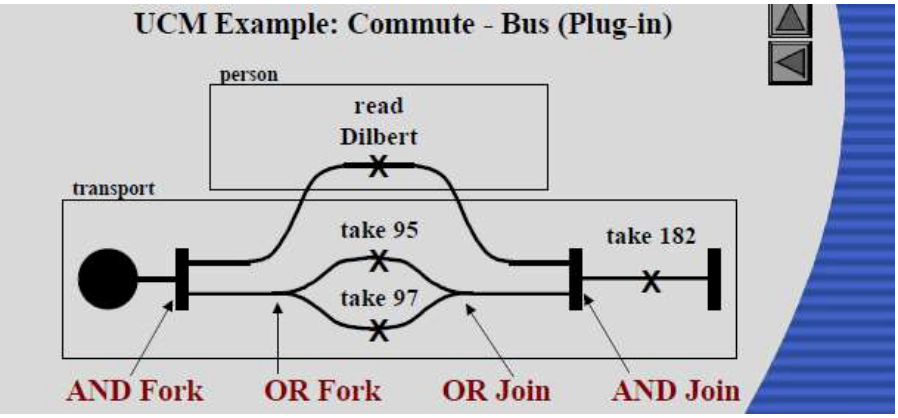


Figura 97 – UCM – Exemplo – *plugin bus* para o *stub* de *Commute*
Fonte: APÊNDICE C [C015]

9. Problem Frames - PF

Acrônimo	PF
Nome	*Problem Frames* (PF)
Artigos	P107, P122, P124, P125, P142 C041, C042
Tipo	Problem Frames
Descrição	*Problem Frames* (PF) é uma abordagem para análise e decomposição de problemas [C041] proposta em 1995 por Michael Jackson e outros autores [C042]. PF permite estruturar a análise do mundo em que o problema está localizado – o domínio do problema – e descrever o que há neste domínio e quais os efeitos espera-se que o sistema atinja [P122]. PF não tem por objetivo modelar graficamente requisitos de forma direta. Apesar disso, há diversos trabalhos que modelam requisitos ao estender os conceitos de PF para outros domínios.
Proposto em	1995
Proponente	Michael Jackson
Artigo origem	[C042] *Software Requirements' Specifications: A Lexicon of Practice, Principles and Prejudices*
Linguagem	*Causal Logic (integration)* [P122], *ATLAS Transformation Language (integration)* [P142]
Ferramenta	*Papyrus Modeling Environment* [P142], *RE-Tools - StarUML plugin* [P107]

Metodologia	N
Padrão	N
Diagrama	*Extensões:* *PF* descarregado para *Annotated Frames,* *Composition Frames* derivados de *PF,* *PF Behaviour Diagram* derivado para *i* task model Diagrams,* *PF Domain Diagram + SysML Requirements Diagram*
Metamodelo	N

Tabela 90 – Modelo *Problem Frames*
Fonte: o autor

Diagrama(s):

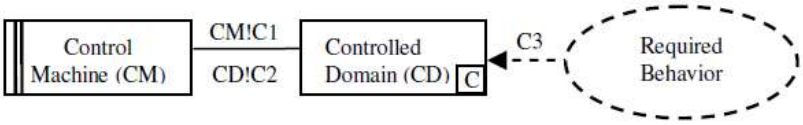


Figura 98 – PF – *Required Behavior Problem Frame*
Fonte: APÊNDICE C [P125]

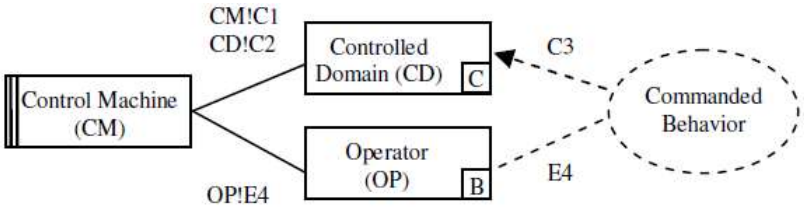


Figura 99 – PF – *Commanded Behavior Problem Frame*
Fonte: APÊNDICE C [P125]

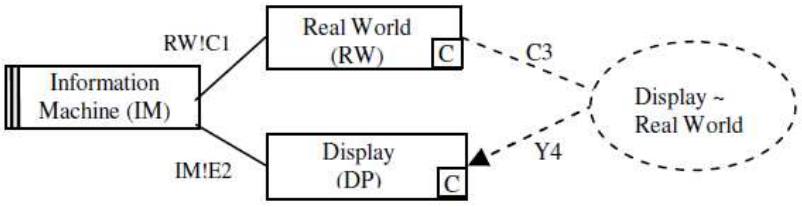


Figura 100 – PF – *Information Display Problem Frame*
Fonte: APÊNDICE C [P125]

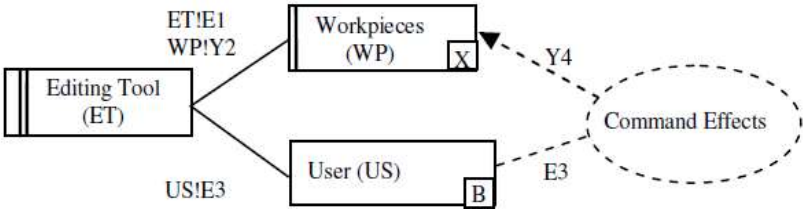


Figura 101 – PF – *Simple Workpieces Problem Frame*
Fonte: APÊNDICE C [P125]

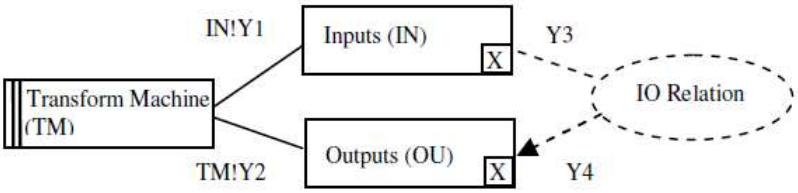
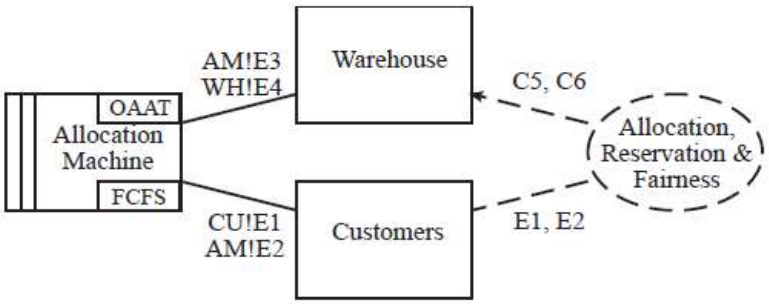


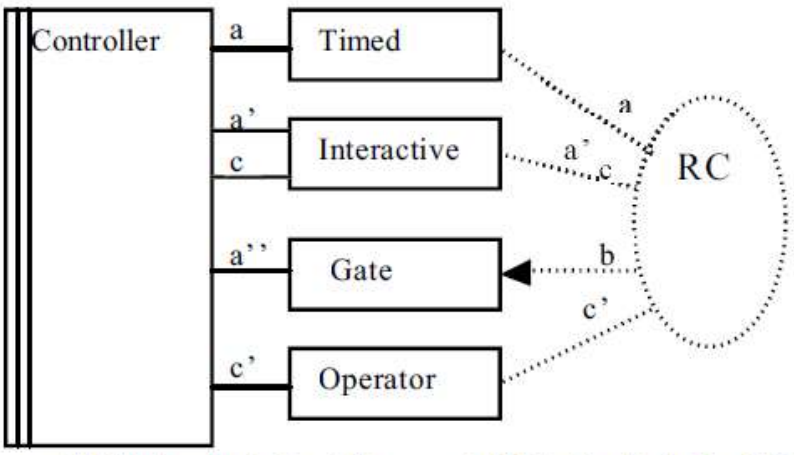
Figura 102 – PF – *Transformation Problem Frame*
Fonte: APÊNDICE C [P125]



CU!E1: Order(O#, Prod, Qty)
AM!E2:Reply(O#, Prod, Success)
AM!E3:Rqst(O#, Prod, Qty)
WH!E4: Alloc(O#, Prod, Qty, Success)
WH!C5: InitQ(Prod, Qty)
WH!C6: Resvd(Prod, Qty)

Services:
OAAT(E3,E4)
FCFS(E1,E3)

Figura 103 – PF – *Annotated Problem Frame*
Fonte: APÊNDICE C [P122]



a: T!{Clockw,Anti, On,Off}
C!{Top,Bottom},
a'': C!{Clockw,Anti, On,Off}
G!{Top,Bottom},
b:G{Open,Shut}

a': I!{Clockw,Anti, On,Off}
C!{Top,Bottom},
c:C!{Raise,Lower,Stop}
c':O!{Raise,Lower,Stop}

Figura 104 – PF – *Composition Problem Frame*
Fonte: APÊNDICE C [P124]

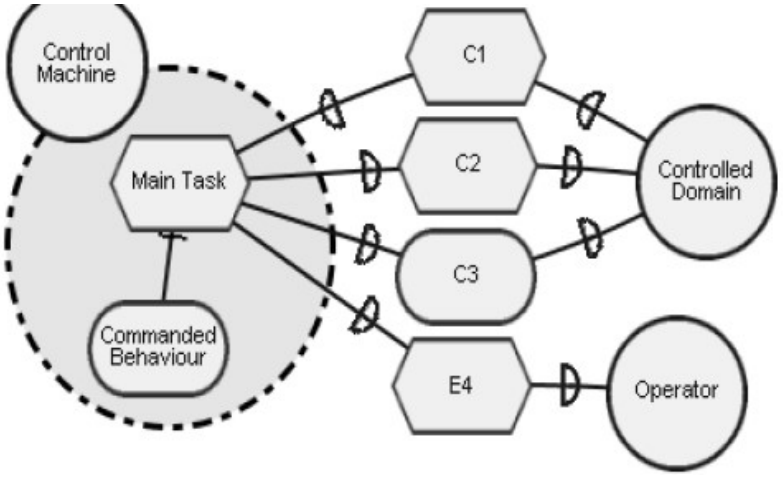


Figura 105 – PF – Commanded behavior PF modelado em i*
Fonte: APÊNDICE C [P125]

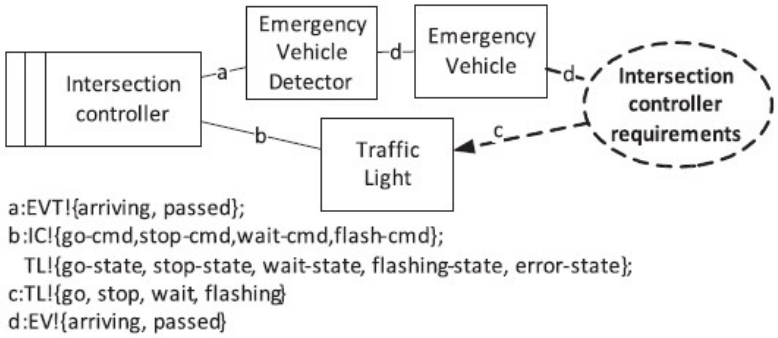


Figura 106 – PF – Exemplo – aplicação para controle de semáforo
Fonte: APÊNDICE C [P142]

10. Visual Analytics Cluster Model - ReCVisu

Acrônimo	RecVisu
Nome	*Visual Analytics Cluster Model*
Artigos	P040, P041, P046 C003, C051
Tipo	Cluster
Descrição	Visual Analytics (VA) é a ciência do raciocínio analítico suportado por interfaces visuais interativas [C003]. Dentre diversas técnicas VA conhecidas, a técnica de representação em **Cluster** foi proposta na ferramenta *ReCVisu* em 2012 para representar requisitos [P046], com o objetivo de facilitar o processo de análise de requisitos [P040].

	Na ferramenta ReCVisu, um grafo é construído tendo requisitos como nodos e suas interrelações como ligações. A partir deste grafo são construídos modelos em cluster com uma separação entre subgrafos coesivos e distâncias interpretativas [P046].
Proposto em	2012
Proponente	Sandeep Reddivari; Zhangji Chen; Nam Niu
Artigo origem	[P046] *ReCVisu: A tool for clustering-based visual exploration of requirements*
Linguagem	N
Ferramenta	*ReCVisu Tool* [P040] [P041] [P046]
Metodologia	*Visual Requirements Analytics* [P040] [P041] [P046]
Padrão	N
Diagrama	*Principal:* *Graph Based Models viewed with the Use of Clustering techniques*
Metamodelo	N

Tabela 91 – Modelo ReCVisu (Cluster)
Fonte: o autor

Diagrama(s):

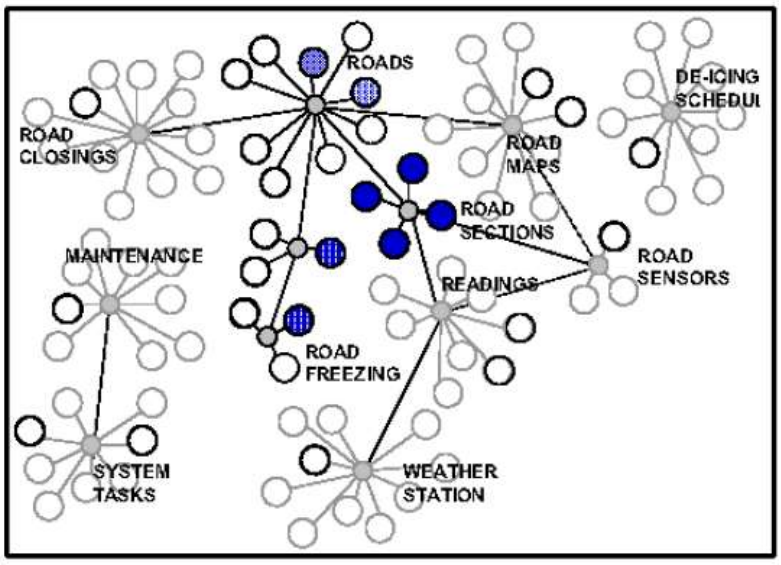


Figura 107 – ReCVisu – Suporte visual baseado em cluster para requisitos
Fonte: APÊNDICE C [P046]

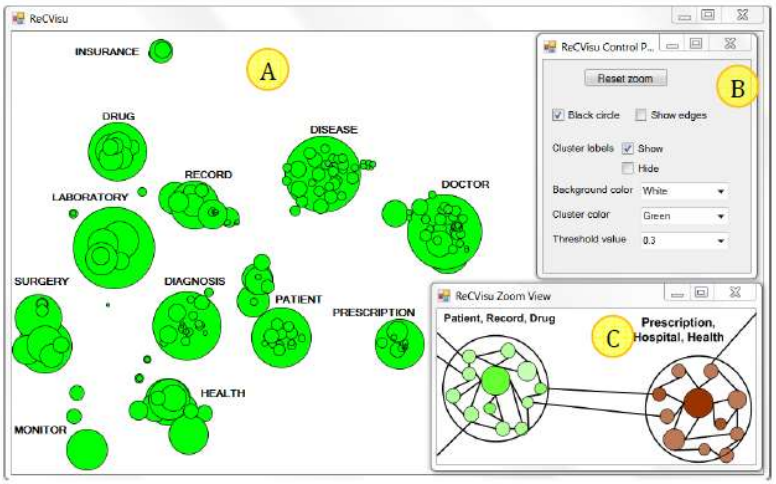


Figura 108 – ReCVisu – Visualização de Requisitos em Cluster
Fonte: APÊNDICE C [P046]

11. V-Graph

Acrônimo	V-Graph
Nome	*V-Graph Goal Model*
Artigos	P105, P106, P121, P195 C040
Tipo	Objetivos (*Goal Based*)
Descrição	V-graph é uma abordagem gráfica proposta em 2004 [P195], derivada de modelos orientados a objetivos e integrada a programação orientada a Aspectos (*Aspects Oriented Programming* - AOP). V-Graph foi desenvolvida em conjunto com um processo para refinamento sistemático de V-Graphs em modelos de objetivos (i*/tropos), de forma a satisfazer os objetivos do modelo, resolver os conflitos e identificar os aspectos ao final do processo [P195].
Proposto em	2004
Proponente	Yijun Yu; Júlio C. S. P. Leite; John Mylopoulos
Artigo origem	[P195] *From goals to aspects: discovering aspects from requirements goal models*
Linguagem	*PL-AOVgraph Language* [P121]
Ferramenta	*ReqSys-MDD tool* [P121]

Metodologia	*Goal Oriented Models integrated with Aspects Programming (AOP)* [P105] [P106] [P121] [P195]
Padrão	N
Diagrama	*Principal:* *V-Graph Goal Model* *Extensões:* *V-Graph Goal Model adapted to Mindmap format* *PL-AOVgraph Goal Model*
Metamodelo	N

Tabela 92 – Modelo V-Graph
Fonte: o autor

Diagrama(s):

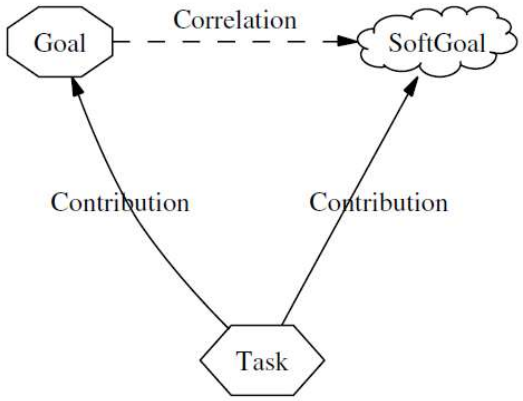


Figura 109 – V-Graph – Um grafo V de objetivos ligando tarefa a *softgoal*
Fonte: APÊNDICE C [P195]

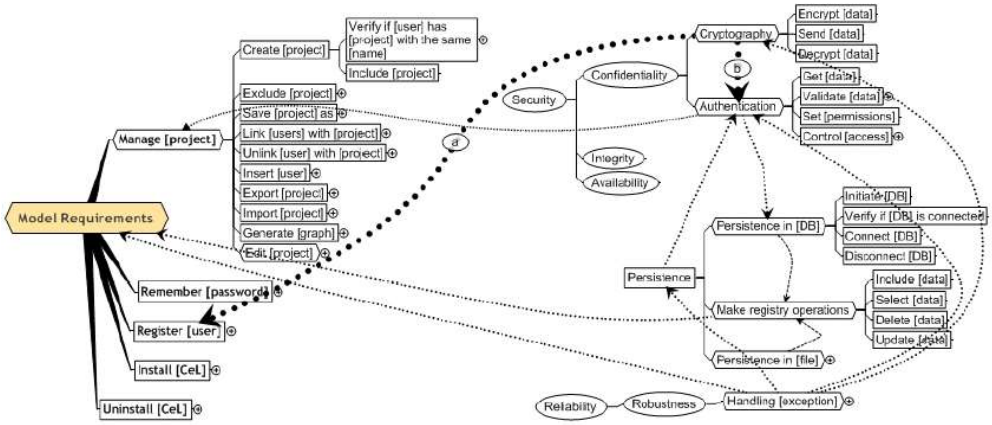


Figura 110 – V-Graph – Relações transversais entre os modelos de metas
Fonte: APÊNDICE C [P106]

12. Techne

Acrônimo	Techne
Nome	*Techne Goal Graph*
Artigos	P131, P151
Tipo	Objetivos (*Goal Based*)
Descrição	Techne é uma abordagem baseada em objetivos (*Goal* based) inicialmente proposta em 2010 [P131], em forma de uma linguagem de modelagem abstrata (Requirements Modeling Language – RML). Posteriormente foi materializada (em 2016) no ambiente gráfico conhecido como AnalyticGraph [P151].
Proposto em	2010
Proponente	Ivan J. Jureta; Alex Borgida; Neil A. Ernst; John Mylopoulos.
Artigo origem	[P131] *Techne: Towards a New Generation of Requirements Modeling Languages with Goals, Preferences, and Inconsistency Handling*
Linguagem	*Techne Requirements Modeling Language* [P131]
Ferramenta	*AnalyticGraph* [P151]
Metodologia	*Techne Graph Graphical Requirements Modeling (GRM)*
Padrão	N
Diagrama	*Principais:* *R-Nets,* *Techne Goal Graph*
Metamodelo	*Analytic Graph Language Management Module Metamodel* [P131] [P151]

Tabela 93 – Modelo Techne
Fonte: o autor

Diagrama(s):

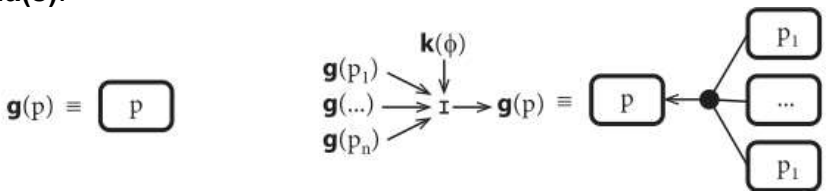


Figura 111 – Techne – Exemplo – sintaxe de objetivos e relação de inferência
Fonte: APÊNDICE C [P131]

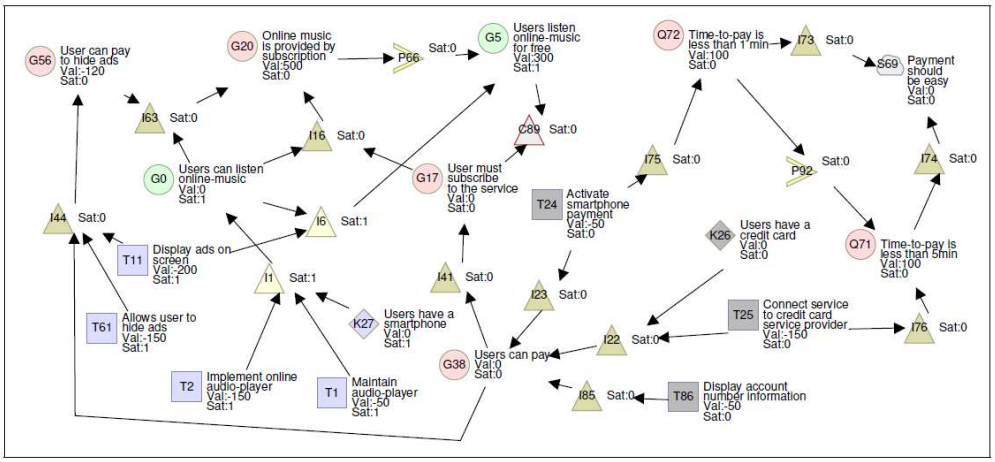


Figura 112 – Techne – Exemplo – grafo de objetivos em *Analytic Graph*
Fonte: APÊNDICE C [P151]

13. Advanced Multimedia Organizer for Requirements Elicitation - AMORE

Acrônimo	AMORE
Nome	AMORE *Visual requirements representation model*
Artigos	P004, P005 C050
Tipo	Multimídia
Descrição	AMORE é uma abordagem “multimídia” originalmente proposta em 1994 [P005] para realizar a elicitação de requisitos, processos e modelos, através de uma ferramenta protótipo. A ferramenta VRAT (de 2002), permite a modelagem inteiramente visual dos requisitos através de componentes MRC (Multimedia Reusable Components) [P004]. VRAT permite criar uma *apresentação* ou filme, que representa os requisitos em uma abordagem baseada em cenários. Cada “cenário” multimídia é composto por “cenas”, representando parte do processo que contém os requisitos [P004].

Proposto em	1994
Proponente	David P. Wood; Michael G. Christel; Scott M. Stevens
Artigo origem	[P005] *A multimedia approach to requirements capture and modeling*
Linguagem	N
Ferramenta	*VRAT Tool* [P004], *AMORE Modeling Tool* [P005]
Metodologia	*Advanced Multimedia Organizer for Requirements Elicitation (AMORE)* [P004] [P005]
Padrão	N
Diagrama	*Principal*: Visual requirements representation model with AMORE multimedia approach
Metamodelo	N

Tabela 94 – Modelo AMORE
Fonte: o autor

Diagrama(s):

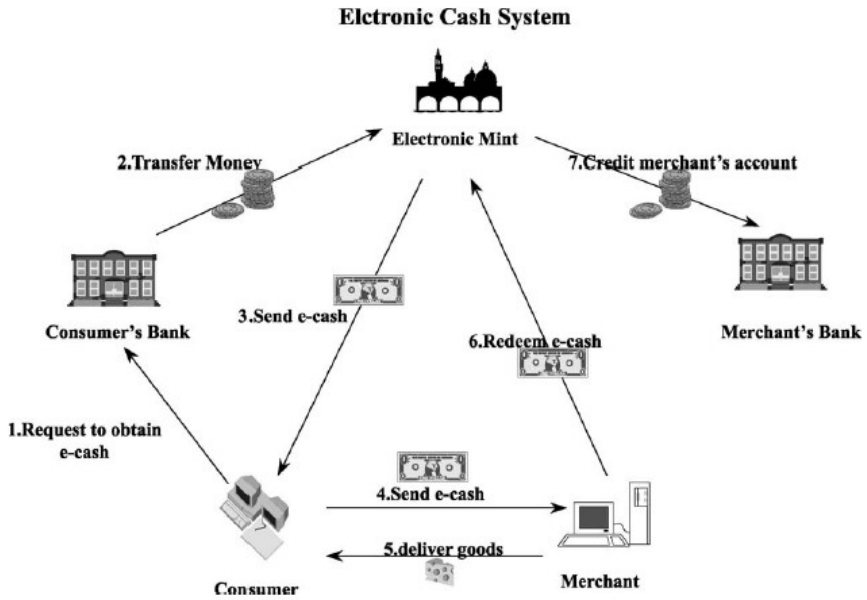


Figura 113 – AMORE – Exemplo – representação de sistema eletrônico bancário
Fonte: APÊNDICE C [P004]

14. Unified Requirements Modeling Language - URML

Acrônimo	URML
Nome	*Unified Requirements Modeling Language (URML)*
Artigos	P166, P167 C017, C052
Tipo	Grafo
Descrição	URML é uma linguagem visual proposta inicialmente em 2010 [C017] e refinada em 2012 como uma extensão de profile UML para realizar a modelagem de requisitos [P166]. Em 2013 foi materializada como um Add-In da ferramenta Enterprise Architect (EA) [P167].
Proposto em	2010
Proponente	Florian Schneider; Bernd Bruegge; Brian Berenbach
Artigo origem	[C017] *A tool implementation of the unified requirements modeling language as enterprise architect add-in*
Linguagem	N
Ferramenta	UNICASE [C017], *URML Add-In to Enterprise Architect* [P167] [C060],
Metodologia	N
Padrão	N
Diagrama	*Principal:* *URML Diagram*
Metamodelo	*URML Metamodel* [P166]

Tabela 95 – Modelo URML
Fonte: o autor

Diagrama(s):

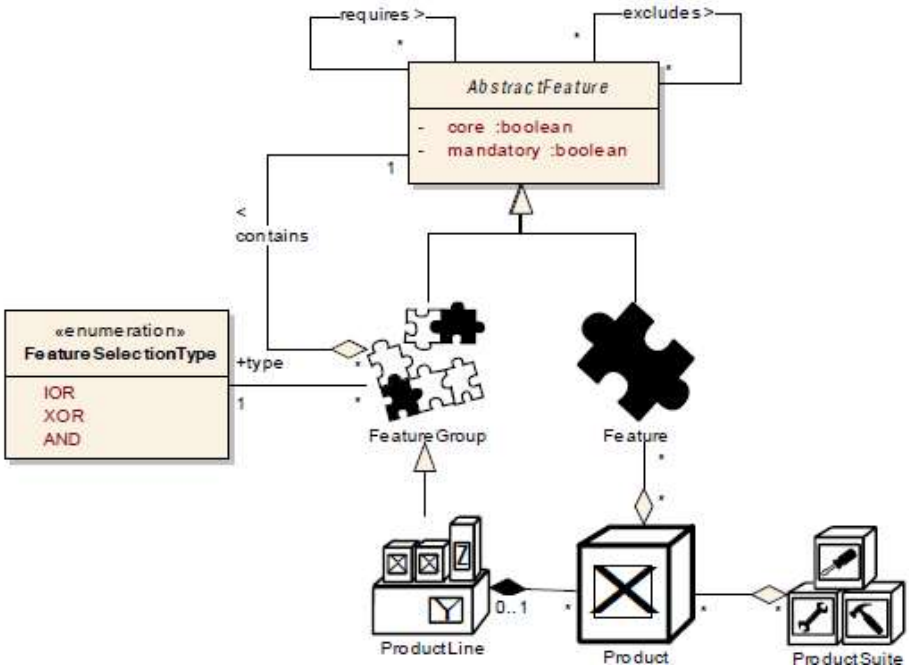


Figura 114 – URML – Modelagem de *Features* no metamodelo
Fonte: APÊNDICE C [P166]

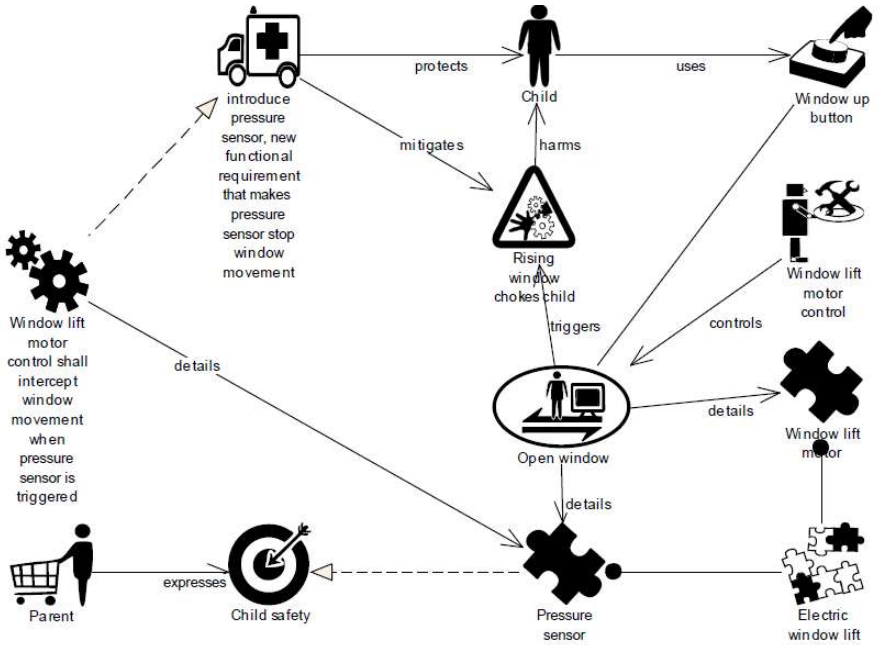


Figura 115 – URML – Aplicação da Modelagem URML
Fonte: APÊNDICE C [P166]

15. UML Requirements Diagram - UML RD

Acrônimo	UML RD
Nome	*UML Requirements Diagram*
Artigos	P165
Tipo	Hierárquico
Descrição	UML *Requirements Diagram* é uma extensão de profile UML proposta em 2011 para suportar a modelagem de requisitos, dentro de um contexto MDA (Model-Driven Architecture) [P165]. A arquitetura COCA-MDA (Context Oriented Component Based Application for Model Driven Architecture) proposta pelos autores visa realizar a modelagem de requisitos integrada a modelagem do comportamento do sistema [P165].
Proposto em	2011
Proponente	Basel Magableh; Stephen Barrett
Artigo origem	[P165] *Productivity evaluation of Self-Adaptive software model driven architecture*
Linguagem	N
Ferramenta	*Enterprise Architect (EA)* [P165] [C060]
Metodologia	*UML Profile Extension for Requirements with COCA-MDA (Context Oriented Component Based Application for Model Driven Architecture)* [P165]
Padrão	UML [C044] (extensão)
Diagrama	*Principal:* *UML Requirements Diagram*
Metamodelo	N

Tabela 96 – Modelo UML Requirements Diagram
Fonte: o autor

Diagrama(s):

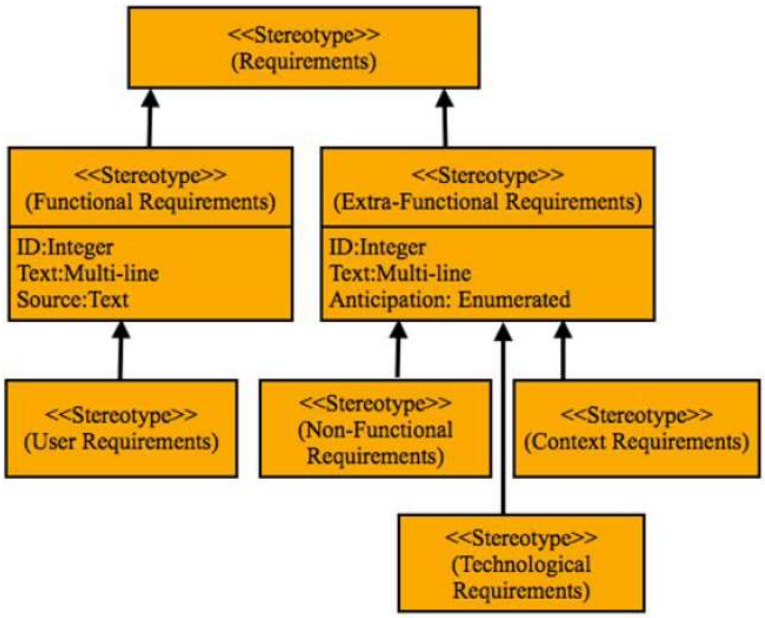


Figura 116 – UML Req. Diagram – *UML Requirements Profile*
Fonte: APÊNDICE C [P165]

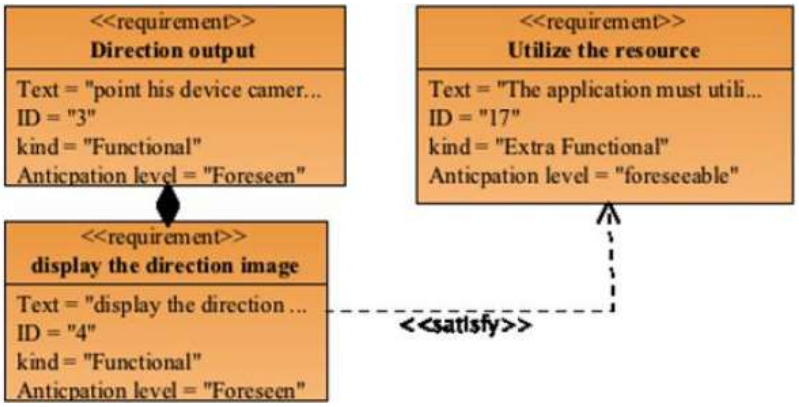


Figura 117 – UML Req. Diagram – Exemplo – diagrama de requisitos (parcial)
Fonte: APÊNDICE C [P165]

16. Very Lightweight Modeling Language - VLML

Acrônimo	VLML
Nome	*VLML (Very Lightweight Modeling Language) Model*
Artigos	P127
Tipo	Processos

Descrição	VLML se aplica aos estágios iniciais de elicitação de requisitos (*early requirements*). Ela evita o uso de linguagem formal, integrando linguagem natural com construtores visuais simples para representar conceitos que as pessoas normalmente não gostam de expressar textualmente: estrutura, relações, influência e fluxo [P127].
Proposto em	2010
Proponente	Martin Glinz
Artigo origem	[P127] *Very Lightweight Requirements Modeling*
Linguagem	N
Ferramenta	N
Metodologia	*VLRM (Very Lightweight Requirements Modeling)* [P127]
Padrão	N
Diagrama	*Principal*: *VLML Diagram* (Requisitos ligados a um *Business Processes Diagrams (BPD)*)
Metamodelo	N

Tabela 97 – Modelo VLML
Fonte: o autor

Diagrama(s):

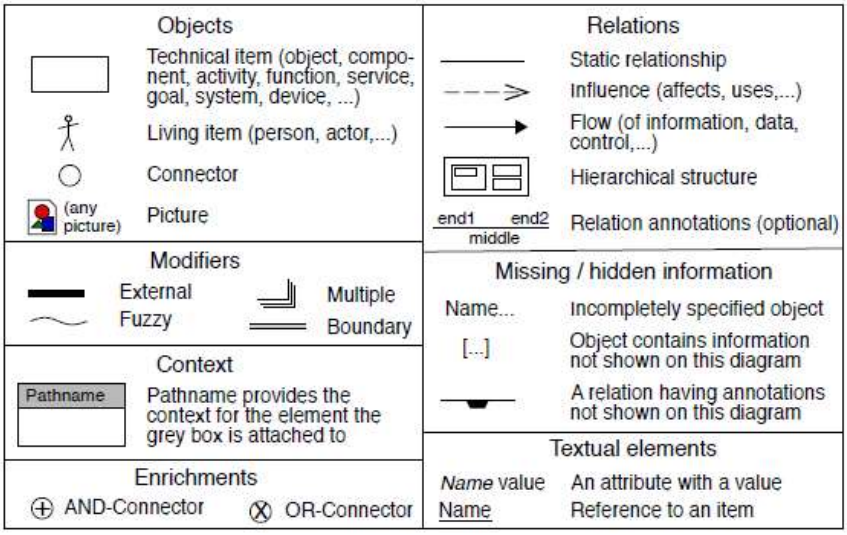


Figura 118 – VLML – Sintaxe Visual
Fonte: APÊNDICE C [P127]

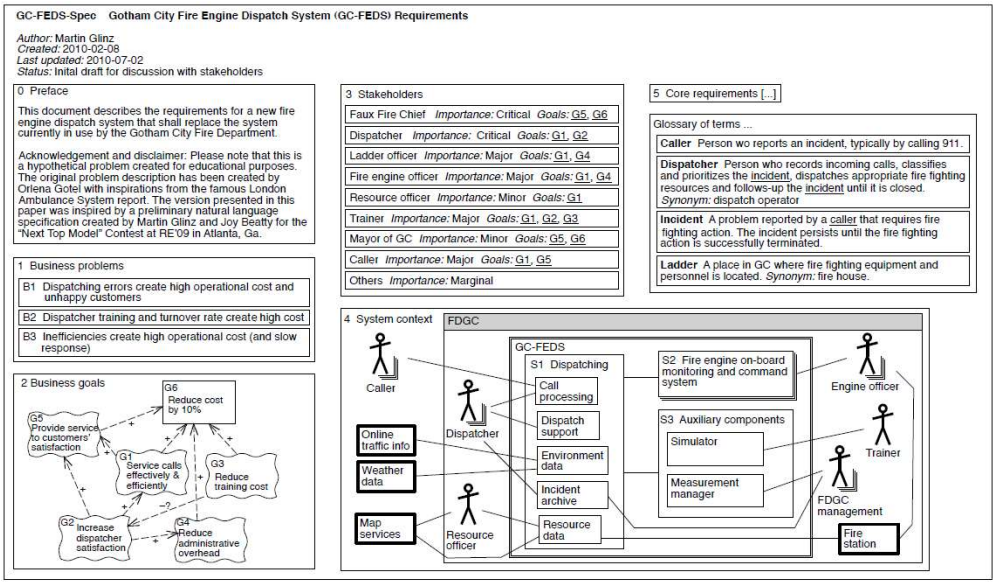


Figura 119 – VLML – Sistema de monitoramento bombeiros *Gotham City* (1de2)
Fonte: APÊNDICE C [P127]

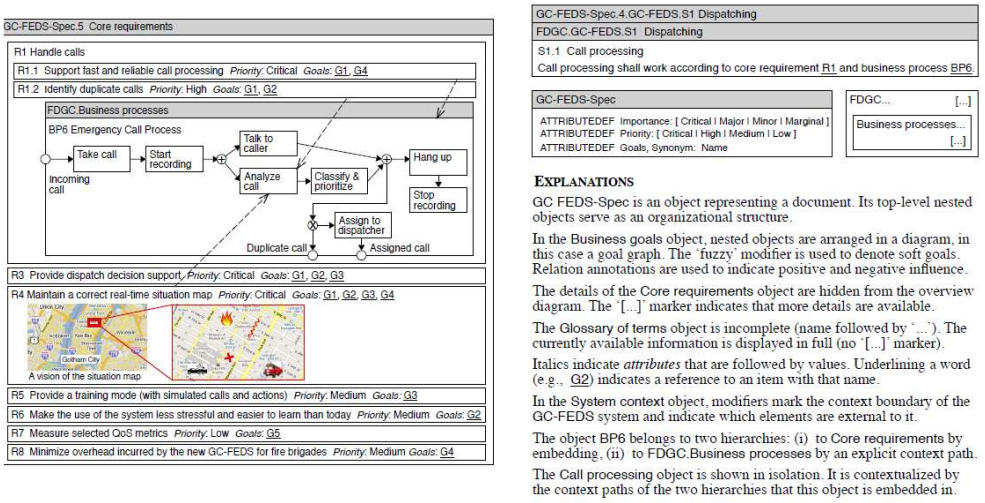


Figura 120 – VLML – Sistema de monitoramento bombeiros *Gotham City* (2de2)
Fonte: APÊNDICE C [P127]

17. OPM Requirements Hierarchical Model

Acrônimo	OPM
Nome	*OPM Requirements Hierarchical Model*
Artigos	P137
Tipo	*OPM (Object Process Methodology)*

Descrição	Em 2017, pesquisadores da *Israel Institute of Technology* apresentaram um estudo em que a notação OPM foi aplicada para modelar requisitos de *stakeholders*. A partir de requisitos textuais de um sistema de carregamento de bagagens aeroportuárias, é aplicado um processo para decomposição dos requisitos e alocação de funcionalidades e sub-funcionalidades em um diagrama com a notação OPM [P137].
Proposto em	2017
Proponente	Yaniv Mordecai; Dov Dori.
Artigo origem	[P137] *Model-based requirements engineering: Architecting for system requirements with stakeholders in mind*
Linguagem	N
Ferramenta	*OPCat Tool* [P137]
Metodologia	*OPM applied to requirements modeling* [P137]
Padrão	N
Diagrama	*Principal*: *Stakeholders requirements hierarchical model using OPM notation*.
Metamodelo	N

Tabela 98 – Modelo OPM Requirements Hierarchical Model
Fonte: o autor

Diagrama(s):

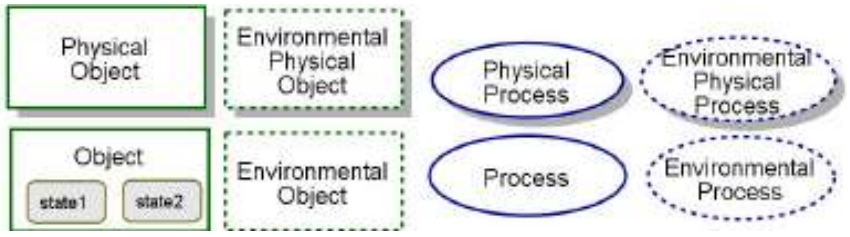


Figura 121 – OPM – Notação: Objeto, Processo, Estado, Essência e Afiliação
Fonte: APÊNDICE C [P137]

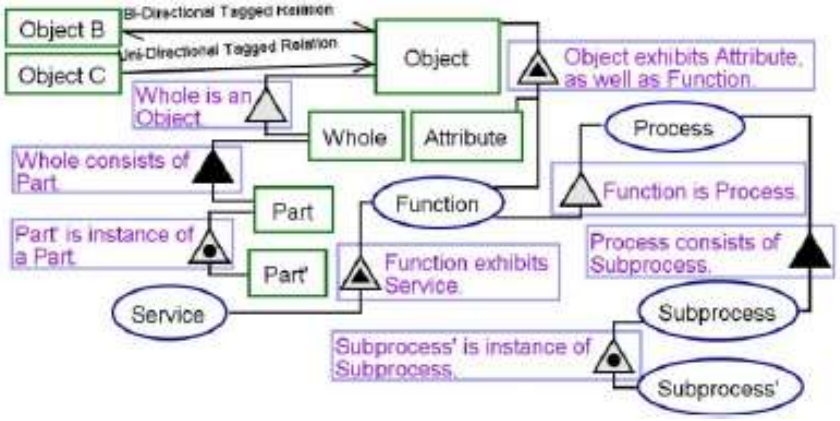


Figura 122 – OPM – Ligações Estruturais
Fonte: APÊNDICE C [P137]

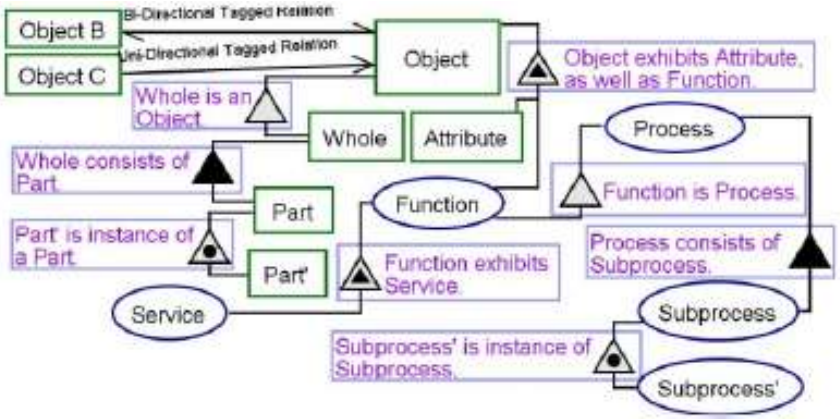


Figura 123 – OPM – Ligações Procedurais
Fonte: APÊNDICE C [P137]

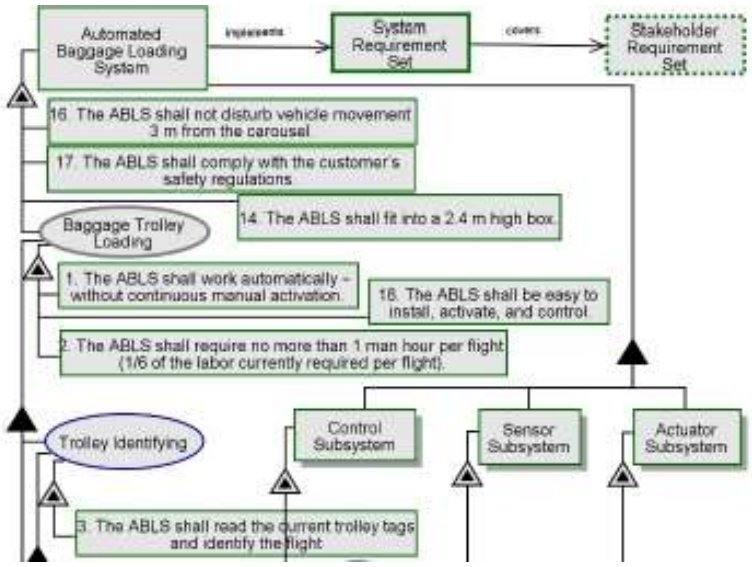


Figura 124 – OPM – Decomposição Funcional-Estrutural e alocação de FRs
Fonte: APÊNDICE C [P137]

18. Requirements Dependency Network - RDN

Acrônimo	RDN
Nome	*Requirements Dependency Network*
Artigos	P001 C001
Tipo	Grafo
Descrição	RDN é uma abordagem proposta em 2013 como forma de identificar de forma antecipada possíveis bugs na integração de softwares [P001]. RDN consiste na construção de diferentes redes de dependências de requisitos, conforme o tipo de interdependência identificada: *PreNetwork (precondition requirements), ConNetwork (constraint requirements), SimNetwork (similar_to requirements) e HybridNetwork* (combinação de todos os tipos).
Proposto em	2013
Proponente	*Junjie Wang; Juan Li; Qing Wang; Da Yang; He Zhang; Mingshu Li*.
Artigo origem	[P001] *Can requirements dependency network be used as early indicator of software integration bugs?*
Linguagem	N
Ferramenta	N
Metodologia	N
Padrão	N
Diagrama	*Principais:* *PreNetwork, ConNetwork, SimNetwork and HybridNetwork.*
Metamodelo	N

Tabela 99 – Modelo RDN
Fonte: o autor

Diagrama(s):

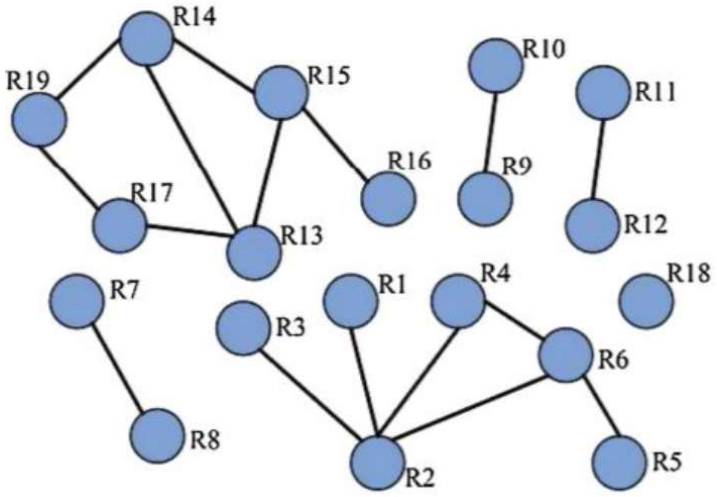


Figura 125 – RDN – Rede de Dependência de Requisitos
Fonte: APÊNDICE C [C001]

19. Attributed Goal-Oriented Requirements Analysis - AGORA

Acrônimo	AGORA
Nome	*AGORA Goal Graphs*
Artigos	P003
Tipo	Objetivos (*Goal Based*)
Descrição	AGORA é uma metodologia com modelagem gráfica orientada a objetivos (*goal based)*, que possui um mecanismo em que atributos de "valores de contribuição" e "matrizes de preferência" são adicionados aos nodos e arestas de grafos de objetivos (*goal-graphs*). O valor de contribuição representa o grau de contribuição do sub-objetivo (*sub-goal*) para o atingimento do objetivo-pai (*parent goa*l), enquanto que a matriz de preferência representa a preferência do *stakeholder* por cada objetivo [P003].
Proposto em	2002
Proponente	Haruhiko Kaiya; Hisayuki Horai; Motoshi Saeki
Artigo origem	[P003] *AGORA: attributed goal-oriented requirements analysis method*
Linguagem	N

Ferramenta	N
Metodologia	AGORA (*Attributed Goal-Oriented Requirements Analysis Method*) [P003]
Padrão	N
Diagrama	*Principal:* *AGORA Goal Graphs (AND-OR Graphs)*
Metamodelo	*AGORA Goal Graph Metamodel* [P003]

Tabela 100 – Modelo AGORA Goal Graphs
Fonte: o autor

Diagrama(s):

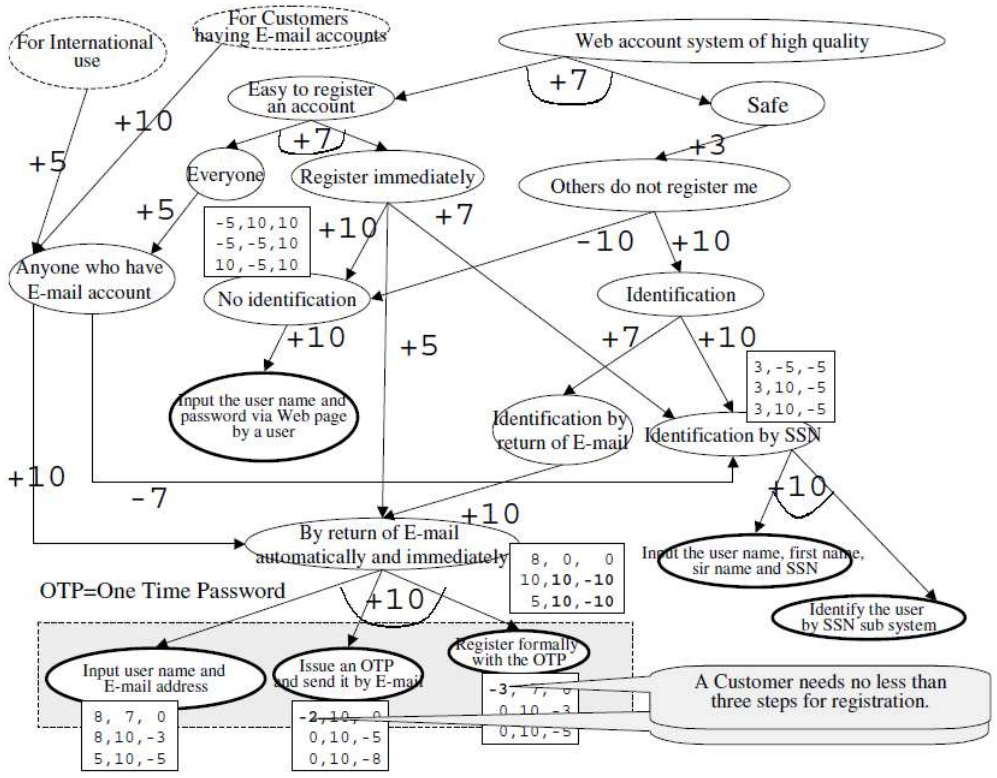


Figura 126 – AGORA Goal Graph – aplicação
Fonte: APÊNDICE C [P003]

20. Advanced Technologies for Interoperability of Heterogeneous Enterprise Networks and their Applications - ATHENA

Acrônimo	ATHENA
Nome	*ATHENA (Advanced Technologies for Interoperability of Heterogeneous Enterprise Networks and their Applications) Requirements Model*
Artigos	P008
Tipo	Grafo
Descrição	ATHENA é uma abordagem proposta como parte do projeto UEML (*Unified Enterprise Modeling Language*), financiado por países da União Européia. A ideia básica era permitir o registro de requisitos em interfaces web por parte de usuários e *stakeholders*, que poderiam ser mantidos e gerenciados em uma ferramenta visual de modelagem visando a elicitação classificação, análise e seleção de requisitos [P008].
Proposto em	2005
Proponente	Helge Solheim; Frank Lillehagen; Sobah A. Petersen; Håvard Jørgensen; Maria Anastasiou
Artigo origem	[P008] *Model-driven visual requirements engineering*
Linguagem	N
Ferramenta	*Metis Visual Modeling Tool* [P008]
Metodologia	*ATHENA Dynamic Requirements Definition System (DRDS)* [P008]
Padrão	N
Diagrama	*Principal:* *ATHENA Requirements Model*
Metamodelo	N

Tabela 101 – Modelo ATHENA
Fonte: o autor

Diagrama(s):

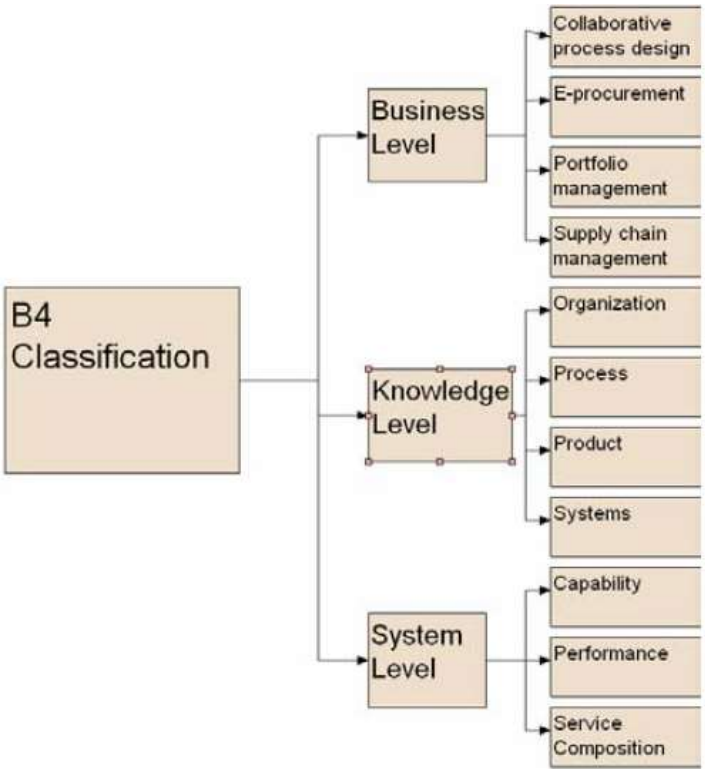


Figura 127 – ATHENA – Exemplo - classificação de requisitos
Fonte: APÊNDICE C [P008]

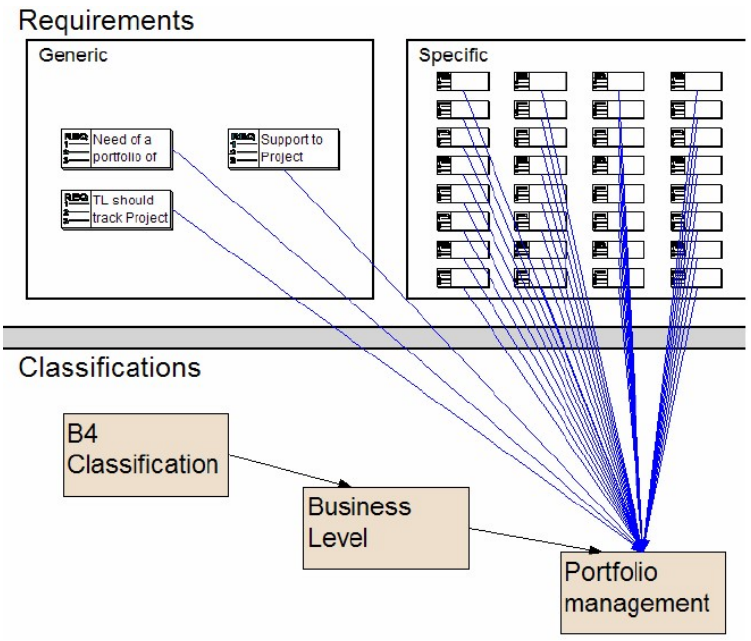


Figura 128 – ATHENA – Associação de requisitos a containers
Fonte: APÊNDICE C [P008]

21. Archimate

Acrônimo	*Archimate*
Nome	*Archimate 2.0 Requirements Model*
Artigos	P128 C002, C062
Tipo	Grafo ou Hierárquico
Descrição	Archimate 2.0 é uma ferramenta disponibilizada pelo “The Open Group” [C002], para modelar ambientes de negócio e arquiteturas organizacionais, incluindo modelagem de requisitos. Essa ferramenta foi usada em 2014 para modelar um conjunto de requisitos de referência para a área de setor público [P128]. Ela define relacionamentos padrão UML de “Associação”, “Agregação” e “Especialização”. Há também o relacionamento de “Influencia”, em que um objeto contribui positiva ou negativamente para a realização de outro objeto, similarmente a conceitos da abordagem i* [P128].
Proposto em	2014
Proponente	Efthimios Tambouris; Eleni Kaliva; Michail Liaros; Konstantinos Tarabanis
Artigo origem	[P128] *A reference requirements set for public service provision enterprise architectures*
Linguagem	N
Ferramenta	*Archimate 2.0 Tool* [P128]
Metodologia	N
Padrão	N
Diagrama	*Principal:* *Archimate 2.0 Requirements Realization Viewpoint Diagram*
Metamodelo	N

Tabela 102 – Modelo Archimate
Fonte: o autor

Diagrama(s):

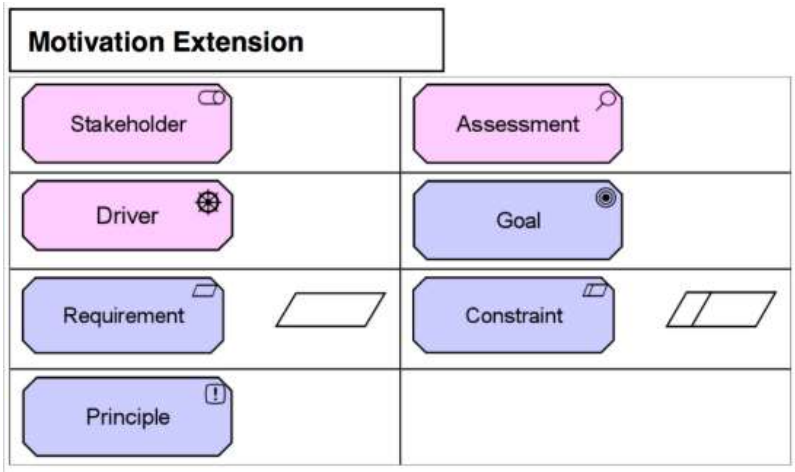


Figura 129 – Archimate – Notação de conceitos da *Motivation Extension*
Fonte: Apêndice B [C062]

Relationship	Description
Association	Indicates that an object is related to another object
Aggregation	Indicates that an object groups a number of other objects
Influence	Indicates the realization of an object that contributes positively or negatively to the realization of another object
Specialization	Indicates that an object is a specializa-tion of another object

Figura 130 – Archimate – Relacionamentos definidos na Motivation Extension
Fonte: Apêndice B [C001]

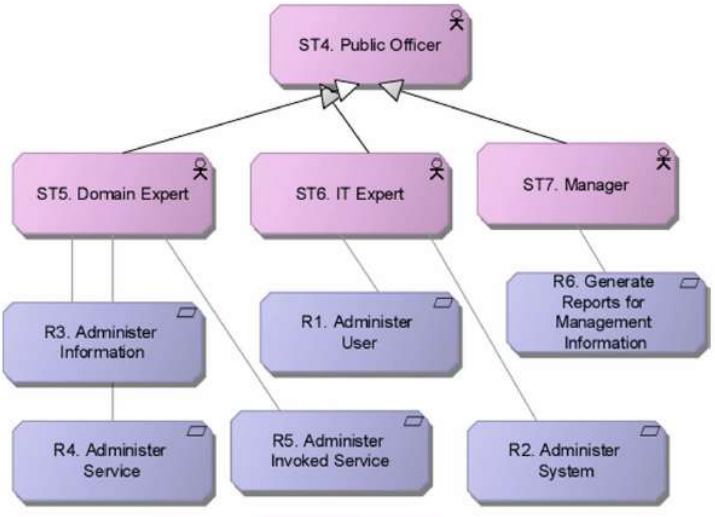


Figura 131 – Archimate – Exemplo – diagrama "*Realization Viewpoint*"
Fonte: APÊNDICE C [C001]

22. Interelement Requirement Diagrams - IRD

Acrônimo	IRD
Nome	*Interelement Requirement Diagrams (IRD)*
Artigos	P094 C004, C005
Tipo	IRD
Descrição	IRD é uma abordagem de modelagem que consiste na junção da linguagem formal MAUDE [C004] com um novo tipo de diagrama conhecido como *Interelement Requirement Diagram* (IRD), de forma a modelar o comportamento esperado do sistema em termos de requisitos [P094]. Nessa abordagem, são definidos os IRDs representando os requisitos do sistema, que posteriormente são convertidos, especificados, simulados e verificados formalmente na linguagem formal MAUDE [P094].
Proposto em	2002
Proponente	Marisol Sánchez-Alonso; Juan M. Murillo
Artigo origem	[P094] *Cooperation Environment Requirements using Formal and Graphical Techniques*
Linguagem	*MAUDE Formal Language* [P094]
Ferramenta	N
Metodologia	N
Padrão	N
Diagrama	*Principal:* *Interelement Requirement Diagram (IRD)*
Metamodelo	N

Tabela 103 – Modelo IRD
Fonte: o autor

Diagrama(s):

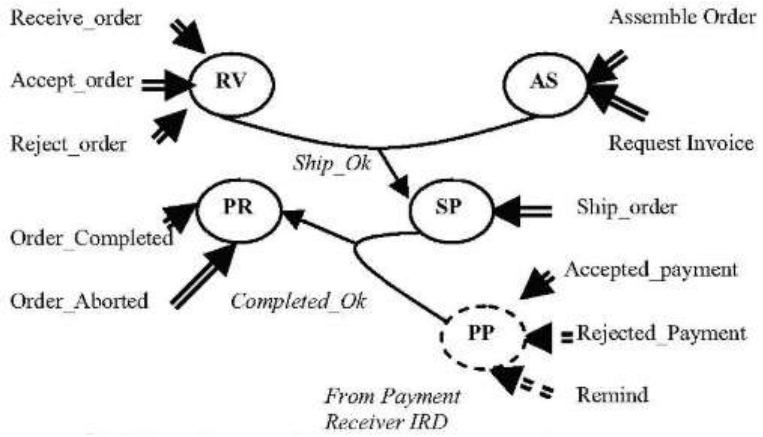
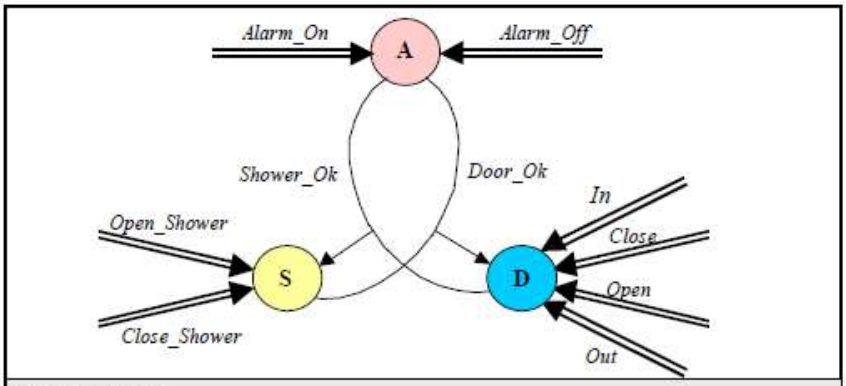


Figura 132 – IRD – Modelo de sistema de processamento de ordens de venda
Fonte: APÊNDICE C [C005]



Element D **is** ::Door.
State closed : Bool
Value inside
Actions In { *inside* ++}
Out { *inside* --}
Close {if *closed* == false and *inside*== 0 then *closed* = true}
Open {if *closed* == true then *closed* = false}

Element A **is** :: Alarm
Actions Alarm_On
Alarm_Off

Element S **is** :: Shower
Actions Open_Shower
Close_Shower

Relation Shower_Ok
Constraint State closed == false **and End** Alarm_On **then** Open_Shower **mode** synchronous

Relation Door_Ok
Constraint End Alarm_Off **and End** Close_Shower **then** Open **mode** synchronous

Figura 133 – IRD – Exemplo – diagrama de um sistema de alarme
Fonte: APÊNDICE C [P094]

23. Requirements Capture Tool - RCAT

Acrônimo	RCAT
Nome	*RCAT (Requirements Capture Tool) Graph Notation*
Artigos	P126
Tipo	Grafo
Descrição	RCAT é uma ferramenta proposta por pesquisadores da NASA Jet Propulsion Lab (JPL), com o objetivo de proporcionar uma notação e ferramenta de suporte gráfico para capturar requisitos formais e convertê-los em autômatos que possam ser usados para verificação de modelos e validação formal. A ferramenta foi projetada para suportar a especificação de requisitos comportamentais de modelos PROMELA [P126].
Proposto em	2008
Proponente	Margaret H. Smith; Klaus Havelund
Artigo origem	[P126] *Requirements Capture with RCAT*
Linguagem	*LTL Formula* [P126]
Ferramenta	*RCAT Tool* [P126]
Metodologia	N
Padrão	N
Diagrama	*Principal:* *RCAT Graph Notation converted to Bücchi Automata Notation*
Metamodelo	N

Tabela 104 – Modelo RCAT
Fonte: o autor

Diagrama(s):

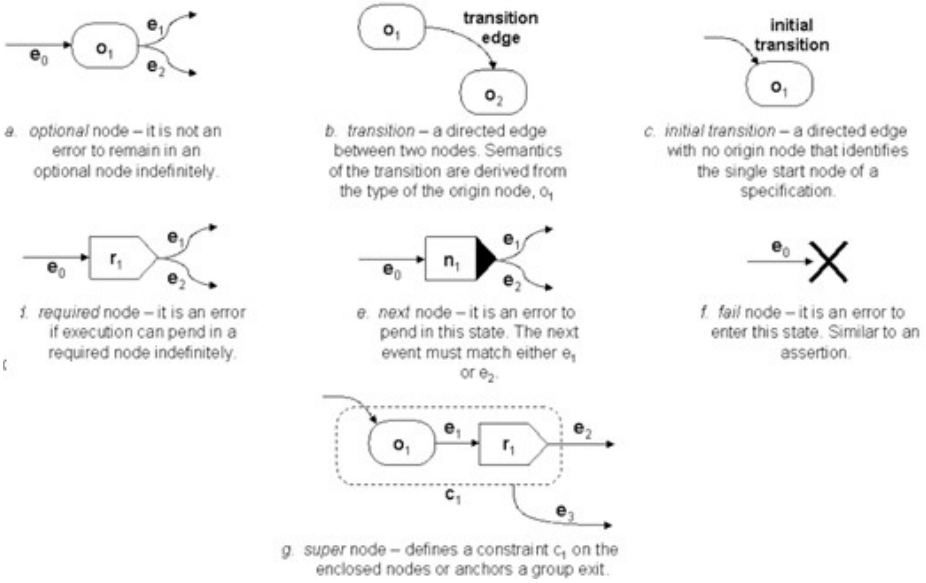


Figura 134 – RCAT – Notação
Fonte: APÊNDICE C [P126]

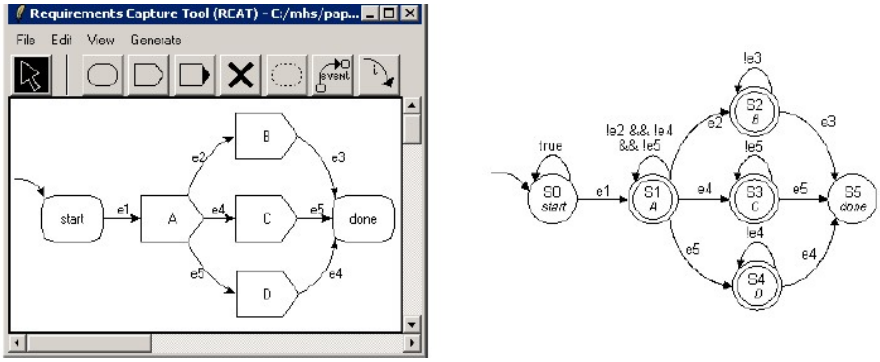


Figura 135 – RCAT – Modelagem na ferramenta RCAT e o Büchi autômata
Fonte: APÊNDICE C [P126]

24. Requirements Interaction Graph - RIG

Acrônimo	RIG
Nome	*Requirements Interaction Graph* (RIG)
Artigos	P129
Tipo	Grafo
Descrição	RIG é uma abordagem que se propõe a construir um “Grafo de Interação de requisitos”, é era parte de um estudo sobre requisitos utilizados em projetos de larga escala, cujos objetivos são [P129]: estabelecer um entendimento comum do que são

	requisitos; definir um processo para elicitar, elaborar e validar requisitos em organizações distribuídas, de larga escala e com numerosos parceiros; identificar inovações necessárias para distinguir requisitos que não podem ser alcançados por soluções existentes [P129].
Proposto em	2003
Proponente	Seda Gürses; Magali Seguran; Nicola Zannone
Artigo origem	[P129] *Requirements engineering within a large-scale security-oriented research project: Lessons learned*
Linguagem	N
Ferramenta	*Graphviz Tool (DOT format)* [P129]
Metodologia	*N*
Padrão	N
Diagrama	*Principal:* *Requirements Interaction Graph*
Metamodelo	N

Tabela 105 – Modelo RIG
Fonte: o autor

Diagrama(s):

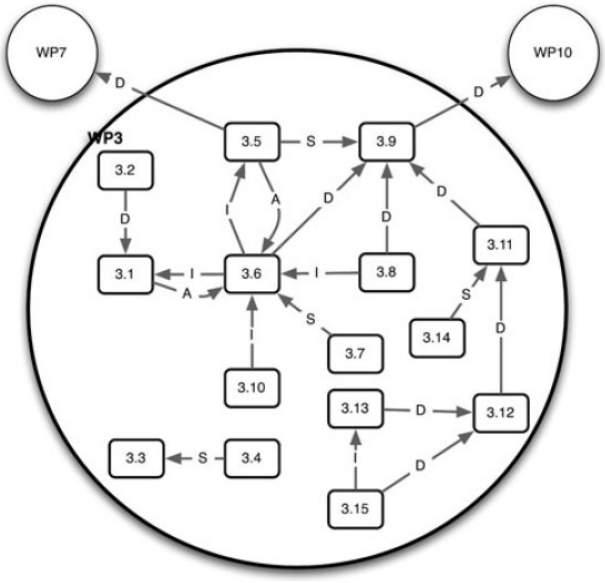


Figura 136 – RIG – Grafo de Interação de Requisitos
Fonte: APÊNDICE C [P129]

25. Requirements Dependency Graph - RDG

Acrônimo	RDG
Nome	*Requirements Dependency Graph (RDG)*
Artigos	P138
Tipo	Grafo
Descrição	RDG se refere a um modelo gráfico de representação desenvolvido como parte da modelagem gráfica de requisitos de um sistema de Gerenciamento de vôo MD-11, que possui uma grande quantidade de requisitos [P138]. A abordagem proposta centra seus esforços no processo de visualização, em que partes do grafo de requisitos são mostrados em *zoom*, de forma a permitir que analista possa lidar de forma mais simples com o volume de informação relacionado às dependências entre os requisitos (incremento da legibilidade) [P138].
Proposto em	2002
Proponente	Nicolas Dulac; Thomas Viguier; Nancy G. Leveson; Margaret-Anne D. Storey
Artigo origem	[P138] *On the use of visualization in formal requirements specification*
Linguagem	*SpecTRM-RL* [P138]
Ferramenta	N
Metodologia	N
Padrão	N
Diagrama	*Principal:* *Requirements Dependency Graph (RDG)*
Metamodelo	N

Tabela 106 – Modelo RDG
Fonte: o autor

Diagrama(s):

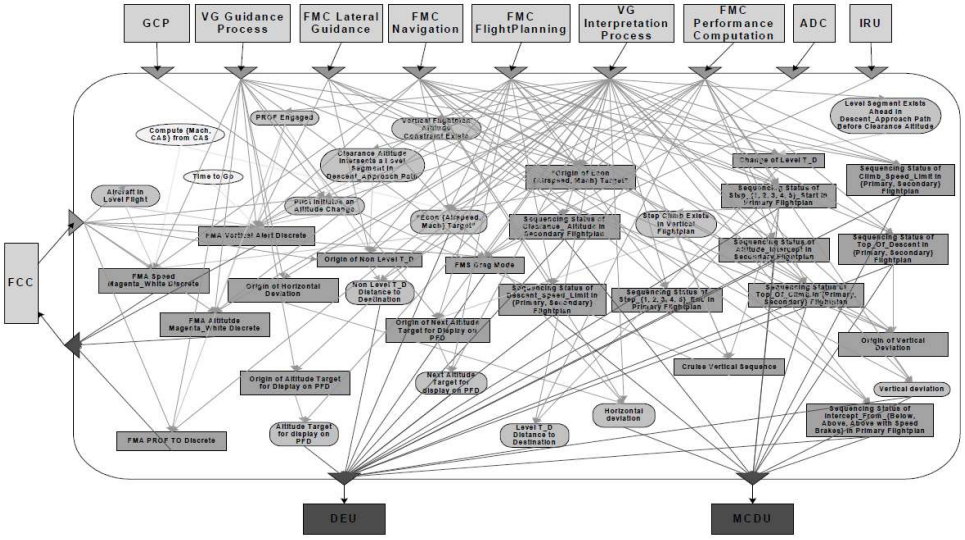


Figura 137 – RDG – Requisitos em sistema de gerenciamento de vôo
Fonte: APÊNDICE C [P138]

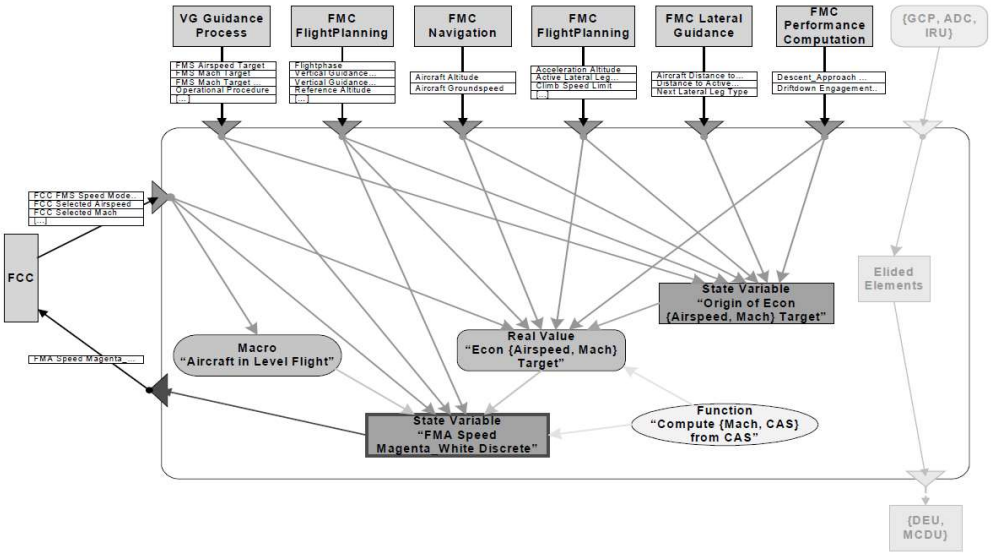


Figura 138 – RDG – Exemplo – *zoom* para o requisito "FMA Speed"
Fonte: APÊNDICE C [P138]

26. Visual Requirements Description Language - VRDL

Acrônimo	VRDL
Nome	*Visual Requirements Description Language* (*VRDL*)
Artigos	P192 C006
Tipo	Multimídia
Descrição	VRDL é uma abordagem cuja proposta visa representar cenários em formato multimídia, com ícones que se movimentam e permitem uma representação dos requisitos baseada em animação gráfica [P192]. Nesta abordagem, o responsável por definir os requisitos identifica os objetos (substantivos), tipos de objetos (atributos), operações entre objetos (verbos) e papéis das operações (casos) e então constrói as sentenças de requisitos. Uma SRS visual pode ser definida em um ambiente gráfico através da adição de objetos (ícones) e ligações. [P192].
Proposto em	1997
Proponente	Atsushi Ohnishi; Norihiro Tokuda
Artigo origem	[C006] *Visual Software Requirements Definition Environment*
Linguagem	N
Ferramenta	N
Metodologia	*Structured Visual SRS using VRDL* [P192]
Padrão	N
Diagrama	*Principal:* *VRDL Visual Model*
Metamodelo	N

Tabela 107 – Modelo VRDL
Fonte: o autor

Diagrama(s):

ICON	LABEL	TYPE
	Customer	human
	Clerk	human
	Receptionist	human
	warehouse	function
	Liquor shop	function
	Order form	data
	liquor	data
	fax	device
	track	device

Figura 139 – VRDL – Definição de ícones
Fonte: APÊNDICE C [P192]

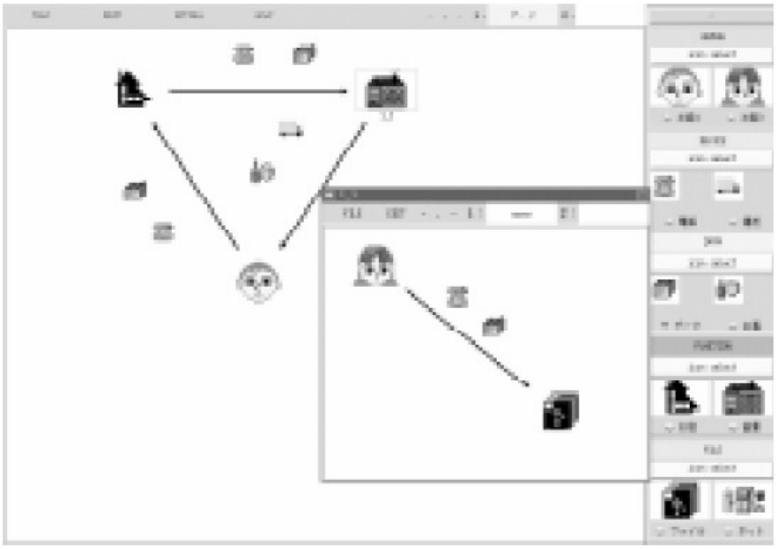


Figura 140 – VRDL – SRS Visual Estruturada
Fonte: APÊNDICE C [P192]

27. Requisitos Orientados a Notificações - RON

Acrônimo	RON
Nome	Requisitos Orientados a Notificacões (RON)
Artigos	P196 C059
Tipo	Grafo
Descrição	RON é uma abordagem de modelagem gráfica de requisitos originada a partir de conceitos do PON (paradigma orientado a notificações) e de MBSE (*Model-Based Systems Engineering)* [P196]. Em RON, as primitivas fundamentais de PON são utilizados para modelar os requisitos, tais como “regras”, elementos da base de fatos (FBE) e notificações (como precondições ou pós-condições) [C059].
Proposto em	2016
Proponente	Jean M. Simão; Hervé Panetto; Yongxin Liao; Paulo C. Stadzisz
Artigo origem	[P196] *A Notification-Oriented Approach for Systems Requirements Engineering*
Linguagem	N
Ferramenta	N
Metodologia	*NOP (Notification Oriented Paradigm) Based Requirements Modeling* [P196]
Padrão	N
Diagrama	*Principal:* *Requirements Oriented to Notifications Diagram*
Metamodelo	N

Tabela 108 – Modelo RON
Fonte: o autor

Diagrama(s):

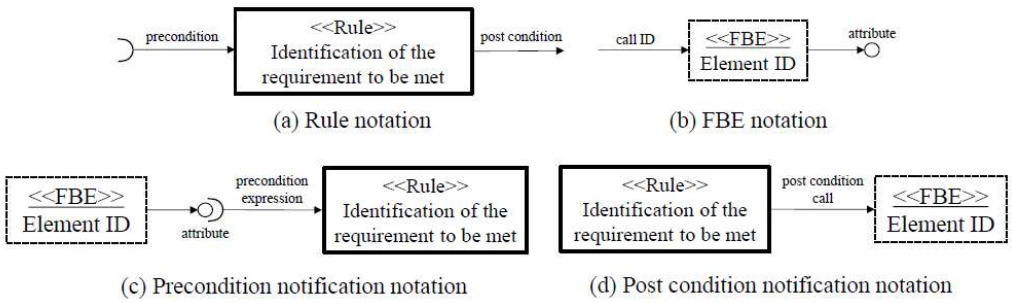


Figura 141 – RON – Notação usada para desenhar modelos de SE
Fonte: APÊNDICE C [P196]

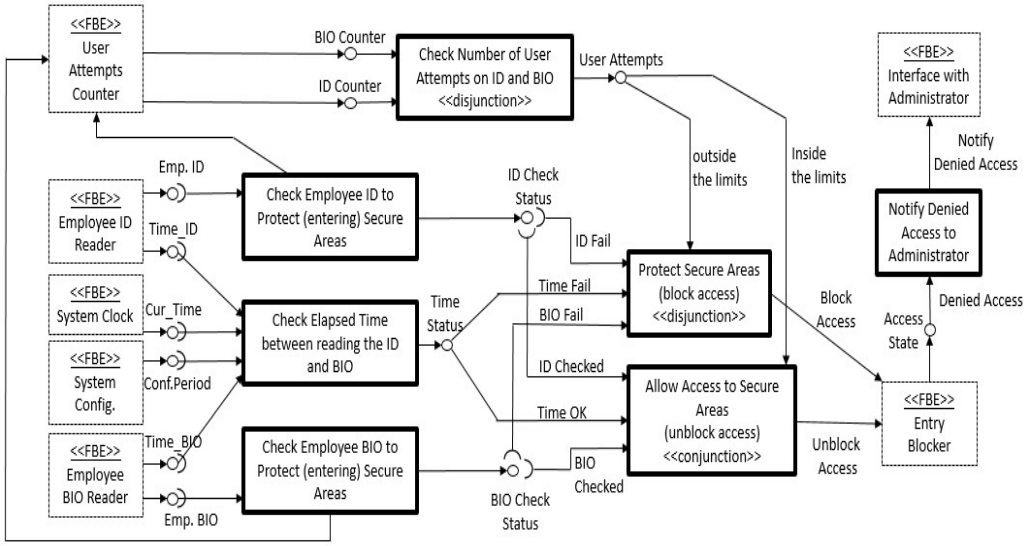


Figura 142 – RON – Modelagem de Requisitos RON
Fonte: APÊNDICE C. Adaptado de [P196]

APÊNDICE C – ARTIGOS SELECIONADOS NA SLM DE MODELOS GRÁFICOS EM RE

I. ARTIGOS PRINCIPAIS

[P001] *WANG, Junjie; LI, Juan; WANG, Qing; YANG, Da; ZHANG, He; LI, Mingshu*. **Can requirements dependency network be used as early indicator of software integration bugs?** *In*: 21st IEEE International Requirements Engineering Conference (RE), p. 185–194, 2013.

[P002] GEORG, Geri; MUSSBACHER, Gunter; AMYOT, Daniel; PETRIU, Dorina; TROUP, Lucy; LOZANO-FUENTES, Saul; FRANCE, Robert. **Synergy between Activity Theory and goal/scenario modeling for requirements elicitation, analysis, and evolution**. Information and Software Technology, v. 59, p. 109–135, 2015.

[P003] KAIYA, Haruhiko; HORAI, Hisayuki; SAEKI, Motoshi. **AGORA: attributed goal-oriented requirements analysis method**. Proceedings IEEE Joint International Conference on Requirements Engineering. 2002.

[P004] CHEN, D.-J.; CHEN, W.-C.; KAVI, K. M. **Visual requirement representation**. Journal of Systems and Software, v. 61, n. 2, p. 129–143, 2002.

[P005] WOOD, David P.; CHRISTEL, Michael G.; STEVENS, Scott M. **A multimedia approach to requirements capture and modeling**. Proceedings of IEEE International Conference on Requirements Engineering. 1994.

[P006] MUSSBACHER, Gunter; ARAÚJO, João; MOREIRA, Ana; AMYOT, Daniel. **AoURN-based modeling and analysis of software product lines**. Software Quality Journal, v. 20, n. 3–4, p. 645–687, 2012.

[P007] AMYOT, Daniel; LEBLANC, Stephane; KEALEY, Jason; KIENZLE, Jorg. **Concern-driven development with jUCMNav**. In: 20th IEEE International Requirements Engineering Conference (RE 2012), p. 319–320, 2012.

[P008] SOLHEIM, Helge; LILLEHAGEN, Frank; PETERSEN, Sobah A.; JØRGENSEN, Håvard; ANASTASIOU, Maria. **Model-driven visual requirements engineering**. *In*: 13th IEEE International Conference on Requirements Engineering (RE'05), p. 421–425, 2005.

[P010] LIASKOS, Sotirios; MCILRAITH, Sheila A.; SOHRABI, Shirin; MYLOPOULOS, John. **Integrating Preferences into Goal Models for Requirements Engineering**. 18th IEEE International Requirements Engineering Conference, pp. 135-144, 2010.

[P011] LIASKOS, Sotirios; LAPOUCHNIAN, Alexei; WANG, Yiqiao; YU, Yijun; EASTERBROOK, Steve. **Configuring common personal software: a requirements-driven approach**. 13th IEEE International Conference on Requirements Engineering (RE'05), pp. 9-18, 2005.

[P013] CARVALHO, Marcia C. F.; ABDELOUAHAB, Zair. **Um Método para Elicitação e Modelagem de Requisitos Baseado em Objetivos**. *In*: IV Workshop em Engenharia de Requisitos, 2001, p. 319–337.

[P014] BOLCHINI, Davide; PAOLINI, Paolo. **Capturing Web Application Requirements through Goal-Oriented Analysis**. *In*: V Workshop em Engenharia de Requisitos, p. 16–28, 2002.

[P016] DARIMONT, R; DELOR, E; MASSONET, P; VAN LAMSWEERDE, A. **GRAIL/KAOS: An Environment for Goal-driven Requirements Engineering**. *In*: Proceedings of the 19th International Conference on Software Engineering (ICSE '97). New York, NY, USA, p. 612–613, 1997.

[P017] CHUNG, Lawrence; NIXON, Brian A. **Dealing with Non-functional Requirements: Three Experimental Studies of a Process-oriented Approach**. *In*: Proceedings of the 17th International Conference on Software Engineering (ICSE '95). New York, NY, USA, p. 25–37, 1995.

[P018] PAIM, Fábio Rilston Silva; CASTRO, Jaelson F. B. **Enhancing Data Warehouse Design with the NFR Framework**. *In*: V Workshop em Engenharia de Requisitos, p. 40–57, 2002.

[P019] JIANG, L.; TOPALOGLOU, T.; BORGIDA, A.; MYLOPOULOS, J. **Incorporating Goal Analysis in Database Design: A Case Study from Biological Data Management**. *In*: 14th IEEE International Requirements Engineering Conference (RE'06), p. 199–207, 2006.

[P020] HASSINE, Jameleddine; AMYOT, Daniel. **A questionnaire-based survey methodology for systematically validating goal-oriented models**. Requirements Engineering, v. 21, n. 2, p. 285–308, 2016.

[P021] HORKOFF, Jennifer; YU, Eric. **Comparison and evaluation of goal-oriented satisfaction analysis techniques**. Requirements Engineering, v. 18, n. 3, p. 199–222, 2013.

[P022] ASADI, Mohsen; GRÖNER, Gerd; MOHABBATI, Bardia; GAŠEVIĆ, Dragan. **Goal-oriented modeling and verification of feature-oriented product lines**. Software and Systems Modeling, v. 15, n. 1, p. 257–279, 2016.

[P023] ALI, Raian; DALPIAZ, Fabiano; GIORGINI, Paolo. **Requirements-driven deployment: Customizing the requirements model for the host environment**. Software and Systems Modeling, v. 13, n. 1, p. 433–456, 2014.

[P024] GORDIJN, J.; PETIT, M.; WIERINGA, R. **Understanding Business Strategies of Networked Value Constellations Using Goal and Value Modeling**. *In*: 14th IEEE International Requirements Engineering Conference (RE'06), p. 129–138, 2006.

[P025] DUBOC, L.; LETIER, E.; ROSENBLUM, D. S.; WICKS, T. **A Case Study in Eliciting Scalability Requirements**. *In*: 16th IEEE International Requirements Engineering Conference, p. 247–252, 2008.

[P026] LIASKOS, S; JALMAN, R; ARANDA, J. **On eliciting contribution measures in goal models**. *In*: 20th IEEE International Requirements Engineering Conference (RE), p. 221–230, 2012.

[P027] REMPEL, P; MÄDER, P; KUSCHKE, T. **An empirical study on project-specific traceability strategies**. *In*: 21st IEEE International Requirements Engineering Conference (RE), p. 195–204, 2013.

[P028] CAILLIAU, A; VAN LAMSWEERDE, A. **Integrating exception handling in goal models**. *In*: IEEE 22nd International Requirements Engineering Conference (RE), p. 43–52, 2014.

[P029] PIMENTEL, J; VILELA, J; CASTRO, J. F. B. **Web tool for Goal modelling and statechart derivation**. *In*: IEEE 23rd International Requirements Engineering Conference (RE), p. 292–293, 2015.

[P030] GRUBB, A M; CHECHIK, M. **Looking into the Crystal Ball: Requirements Evolution over Time**. *In*: IEEE 24th International Requirements Engineering Conference (RE), p. 86–95, 2016.

[P031] DARIMONT, R; PONSARD, C. **Supporting quantitative assessment of requirements in Goal Orientation**. *In*: IEEE 23rd International Requirements Engineering Conference (RE), p. 290–291, 2015.

[P032] HORKOFF, Jennifer; AYDEMIR, Fatma Basak; CARDOSO, Evellin; LI, Tong; MATÉ, Alejandro; PAJA, Elda; SALNITRI, Mattia; MYLOPOULOS, John; GIORGINI, Paolo. **Goal-Oriented Requirements Engineering: A Systematic Literature Map**. *In*: IEEE 24th International Requirements Engineering Conference (RE), p. 106–115, 2016.

[P033] OSTER, Z J; SANTHANAM, G R; BASU, S. **Scalable modeling and analysis of requirements preferences: A qualitative approach using CI-Nets**. *In*: IEEE 23rd International Requirements Engineering Conference (RE), p. 214–219, 2015.

[P034] TOUZANI, M; PONSARD, C. **Towards Modelling and Analysis of Spatial and Temporal Requirements**. *In*: IEEE 24th International Requirements Engineering Conference (RE). p. 389–394, 2016.

[P035] VILELA, Jéssyka; CASTRO, Jaelson F. B.; MARTINS, Luiz Eduardo G; *GORSCHEK, Tony; SILVA, Carla*. **Specifying Safety Requirements with GORE Languages**. *In*: 31st Brazilian Symposium on Software Engineering (SBES'17). New York, NY, USA, p. 154–163, 2017.

[P036] NEGRI, Pedro Pignaton; SOUZA, Vítor E. Silva; LEAL, André Luiz de Castro; *FALBO, Ricardo de Almeida; GUIZZARDI, Giancarlo*. **Towards an Ontology of Goal-Oriented Requirements**. *In*: WER17 XX Workshop em Engenharia de Requisitos. Buenos Aires, Argentina, 2017.

[P037] VILELA, Jéssyka; CASTRO, Jaelson F. B.; PIMENTEL, João; LIMA, Paulo. **On the behaviour of context-sensitive systems**. *In*: XVIII Workshop em Engenharia de Requisitos. Lima, Peru, v. 57, p. 119–131, 2015.

[P038] CYSNEIROS, Luiz Marcio; LEITE, Júlio C. S. P. **Definindo Requisitos Não Funcionais**. *In*: XI Simpósio Brasileiro de Engenharia de Software, p. 49–64, 1997.

[P039] GASTALDO, Denise L.; MIDORIKAWA, Edson T. **Processo de Engenharia de Requisitos Aplicado a Requisitos Não-Funcionais de Desempenho – Um Estudo de Caso**. *In*: VI Workshop em Engenharia de Requisitos., p. 302–316, 2003.

[P040] NIU, N; REDDIVARI, S; CHEN, Z. **Keeping requirements on track via visual analytics**. *In*: 21st IEEE International Requirements Engineering Conference (RE)., p. 205–214, 2013.

[P041] REDDIVARI, S. **Visual analytics for software requirements engineering**. *In*: 21st IEEE International Requirements Engineering Conference (RE), p. 389–392, 2013.

[P042] HASSINE, Jameleddine; AMYOT, Daniel. **An empirical approach toward the resolution of conflicts in goal-oriented models**. Software and Systems Modeling, v. 16, n. 1, p. 279–306, 2017.

[P043] LOSAVIO, Francisca; GUZMÁN, Jean C.; MATTEO, Alfredo. **Semantic Correspondence between GRL and BPMN Languages**. Enl@ce: Revista Venezolana de Información, Tecnología y Conocimiento, v. 8, n. 1, p. 11–29, 2011.

[P044] GHANAVATI, Sepideh; RIFAUT, André; DUBOIS, Eric; AMYOT, Daniel. **Goal-oriented compliance with multiple regulations**. *In*: IEEE 22nd International Requirements Engineering Conference (RE). p. 73–82, 2014.

[P045] VAN ZEE, Marc; BEX, Floris; GHANAVATI, Sepideh. **Rationalization of goal models in GRL using formal argumentation**. *In*: IEEE 23rd International Requirements Engineering Conference, RE 2015, p. 220–225, 2015.

[P046] REDDIVARI, Sandeep; CHEN, Zhangji; NIU, Nam. **ReCVisu: A tool for clustering-based visual exploration of requirements**. *In*: 20th IEEE International Requirements Engineering Conference (RE), p. 327–328, 2012.

[P047] MAIDEN, N. A. M.; MANNING, S.; JONES, S.; GREENWOOD, J. **Generating requirements from systems models using patterns: a case study**. Requirements Engineering, v. 10, n. 4, p. 276–288, 2005.

[P048] HORKOFF, Jennifer; YU, Eric. **Interactive goal model analysis for early requirements engineering**. Requirements Engineering, v. 21, n. 1, p. 29–61, 2016.

[P049] MOODY, Daniel L.; HEYMANS, Patrick; MATULEVIČIUS, Raimundas. **Visual syntax does matter: Improving the cognitive effectiveness of the i* visual notation**. Requirements Engineering, v. 15, n. 2, p. 141–175, 2010.

[P050] ALENCAR, Fernanda M. R.; CASTRO, Jaelson F. B. **Integrating Early and Late-Phase Requirements: A Factory Case Study**. *In*: XIII Simpósio Brasileiro de Engenharia de Software, 1999.

[P051] WEI, Bo; JIN, Zhi; ZOWGHI, Didar; YIN, Bin. **Implementation decision making for internetware driven by quality requirements**. Science China Information Sciences, v. 57, n. 7, p. 1–19, 2014.

[P052] BASTOS, Lúcia R. D.; CASTRO, Jaelson F. B. **Enhancing Requirements to derive Multi-Agent Architectures**. *In*: WER04 VII Workshop em Engenharia de Requisitos, p. 127–139, 2004.

[P053] CLELAND-HUANG, Jane; MARRERO, Will; BERENBACH, Brian. **Goal-centric traceability: Using virtual plumblines to maintain critical systemic qualities**. IEEE Transactions on Software Engineering, v. 34, n. 5, p. 685–699, 2008.

[P054] PENSERINI, Loris; PERINI, Anna; SUSI, Angelo; MYLOPOULOS, John. **High variability design for software agent: Extending Tropos**. ACM Transactions on Autonomous and Adaptive Systems, v. 2, n. 4, p. 16–es, 2007.

[P055] FUXMAN, Ariel; LIU, Lin; PISTORE, Marco; ROVERI, Marco; MYLOPOULOS, John. **Specifying and analyzing early requirements: some experimental results**. *In*: Proceedings. 11th IEEE International Requirements Engineering Conference, p. 105–114, 2003.

[P056] MORANDINI, Mirko; PENSERINI, Loris; PERINI, Anna; MARCHETTO, Alessandro. **Engineering requirements for adaptive systems**. Requirements Engineering, v. 22, n. 1, p. 77–103, 2017.

[P057] MOURATIDIS, Haralambos; ISLAM, Shareeful; KALLONIATIS, Christos; GRITZALIS, Stefanos. **A framework to support selection of cloud providers based on security and privacy requirements**. The Journal of Systems & Software, v. 86, n. 9, p. 2276--2293, 2013.

[P058] MASSACCI, Fabio; MYLOPOULOS, John; ZANNONE, Nicola. **Computer-aided support for secure tropos**. Automated Software Engineering, v. 14, n. 3, p. 341–364, 2007.

[P059] SUSI, Angelo; PERINI, Anna; MYLOPOULOS, John; GIORGINI, Paolo. **The Tropos metamodel and its use**. Informatica, v. 29, n. 4, p. 401–408, 2005.

[P060] ISLAM, Shareeful; MOURATIDIS, Haralambos; JÜRJENS, Jan. **A framework to support alignment of secure software engineering with legal regulations**. Software and Systems Modeling, v. 10, n. 3, p. 369–394, 2011.

[P061] MORALES, José Miguel; NAVARRO, Elena; SÁNCHEZ, Pedro; ALONSO, Diego. **A controlled experiment to evaluate the understandability of KAOS and i∗ for modeling Teleo-Reactive systems**. Journal of Systems and Software, v. 100, p. 1–14, 2015.

[P062] MOODY, D L; HEYMANS, P; MATULEVICIUS, R. **Improving the Effectiveness of Visual Representations in Requirements Engineering: An Evaluation of i* Visual Syntax**. *In*: 17th IEEE International Requirements Engineering Conference, p. 171–180, 2009.

[P063] CARVALLO, Juan Pablo; FRANCH, Xavier. **Descubriendo la Arquitectura de Sistemas de Software Híbridos: Un Enfoque Basado en Modelos i***. Workshop on Requirements Engineering, v. 12th, p. 45–56, 2009.

[P064] LUCENA, Márcia; SILVA, Carla T. L. L.; SANTOS, Emanuel; ALENCAR, Fernanda M. R.; CASTRO, Jaelson F. B. **Modularizando Modelos i*: uma Abordagem baseada em Transformação de Modelos**. *In*: XII Workshop em Engenharia de Requisitos, 2009.

[P065] MATHEW, George; MENZIES, Tim; ERNST, Neil A.; KLEIN, John. **"SHORT"er Reasoning About Larger Requirements Models**. *In*: IEEE 25th International Requirements Engineering Conference (RE), p. 154–163, 2017.

[P066] BOLCHINI, D; PAOLINI, P; RANDAZZO, G. **Adding hypermedia requirements to goal-driven analysis**. *In*: 11th IEEE International Requirements Engineering Conference, p. 127–137, 2003.

[P067] GRUBB, A M; CHECHIK, M. **Modeling and Reasoning with Changing Intentions: An Experiment**. *In*: IEEE 25th International Requirements Engineering Conference (RE), p. 164–173, 2017.

[P068] GREGORIADES, A; SHIH, Jae-Eun; SUTCLIFFE, A. **Human-centred requirements engineering**. *In*: Proceedings. 12th IEEE International Requirements Engineering Conference, p. 154–163, 2004.

[P069] DEB, N; CHAKI, N; GHOSE, A. **i*ToNuSMV: A Prototype for Enabling Model Checking of i* Models**. *In*: 2016 IEEE 24th International Requirements Engineering Conference (RE), p. 397–398, 2016.

[P070] WEBSTER, Ilca; AMARAL, Juliana; CYSNEIROS FILHO, Luiz Márcio. **A Survey of Good Practices and Misuses for Modelling with i* Framework**. *In*: WER05 VIII Workshop em Engenharia de Requisitos, v. 5, p. 148–160, 2005.

[P071] OLIVEIRA, Antônio de P. A.; LEITE, Julio C. S. P.; CYSNEIROS, Luiz Marcio. **Método ERi*c - Engenharia de Requisitos Intencional**. *In*: XI Workshop em Engenharia de Requisitos, p. 155–166, 2008.

[P072] BAÍA, Joás W.; BRAGA, José L.; CARVALHO, Leonardo F. De. **Verificação de Requisitos de Transparência em Modelos iStar**. *In*: XV Workshop em Engenharia de Requisitos. Buenos Aires, Argentina, 2012.

[P073] BRISCHKE, Mauro; SANTANDER, Victor F. A.; SILVA, Ivonei Freitas da. **Melhorando a Ferramenta JGOOSE**. *In*: XV Workshop em Engenharia de Requisitos. Buenos Aires, Argentina, 2012.

[P074] RODRIGUES, Jacqueline A. do N. T.; SALEM, Nathálea; WERNECK, Vera Maria B. **Modelo Intencional Genérico de Sistemas Biométricos**. *In*: XV Workshop em Engenharia de Requisitos. Buenos Aires, Argentina, 2012.

[P075] SOUZA, Cleice; ALENCAR, Fernanda M. R.; GUEDES, Gabriela; SOARES, Monique; SOUZA, Cláudia; RAMOS, Ricardo; CASTRO, Jaelson F. B. **AIRDoc-i*: um processo para avaliação de modelos i***. *In*: XV Workshop em Engenharia de Requisitos. Buenos Aires, Argentina, p. 5–8, 2012.

[P076] MELO, Josenildo; SOUSA, Aêda; AGRA, Celso; JÚNIOR, José; CASTRO, Jaelson F. B.; ALENCAR, Fernanda M. R. **Formalization of Mapping Rules from iStar to Class Diagram in UML**. *In*: 29th Brazilian Symposium on Software Engineering, p. 71–79, 2015.

[P077] CAIRE, Patrice; GENON, Nicolas; HEYMANS, Patrick; MOODY, Daniel L. **Visual notation design 2.0: Towards user comprehensible requirements engineering notations**. *In*: 21st IEEE International Requirements Engineering Conference (RE), p. 115–124, 2013.

[P078] RAMOS, Ricardo Argenton; ALENCAR, Fernanda M. R.; ARAÚJO, João; MOREIRA, Ana; CASTRO, Jaelson F. B.; PENTEADO, Rosângela A. D. **i* with Aspects: Evaluating Understandability**. *In*: X Workshop em Engenharia de Requisitos, p. 171–178, 2007.

[P079] MERLIN, Leonardo P.; SILVA, Alexandre L. B.; SANTANDER, Victor F. A.; SILVA, Ivonei F.; CASTRO, Jaelson F. B. **Integrating the E4J editor to the JGOOSE tool. XVIII Workshop em Engenharia de Requisitos**. Lima, Peru, 2015.

[P080] WAUTELET, Yves; HENG, Samedi; KIV, Soreangsey; KOLP, Manuel. **User-story driven development of multi-agent systems: A process fragment for agile methods**. Computer Languages, Systems and Structures, v. 50, p. 159–176, 2017.

[P081] NIU, Nam; KOSHOFFER, Amy; NEWMAN, Linda; KHATWANI, Charu; SAMARASINGHE, Chatura; SAVOLAINEN, Juha. **Advancing Repeated Research in Requirements Engineering: A Theoretical Replication of Viewpoint Merging**. *In*: IEEE 24th International Requirements Engineering Conference (RE), p. 186–195, 2016.

[P082] SANTOS, Mafalda; GRALHA, Catarina; GOULÃO, Miguel; ARAÚJO, João; MOREIRA, Ana; CAMBEIRO, João. **What is the Impact of Bad Layout in the Understandability of Social Goal Models?** *In*: IEEE 24th International Requirements Engineering Conference (RE), p. 206–215, 2016.

[P083] MOURA, Ana Maria M. **Awareness Driven Software Reengineering**. *In*: IEEE 25th International Requirements Engineering, p. 550–555, 2017.

[P084] PENHA, Fabio; LUCENA, Marcia; LUCENA, Leonardo; *ALENCAR, Fernanda M. R.; AGRA, Celso*. **Uma Notação Textual Modular e Escalável para Modelos de Requisitos iStar**. *In*: XX Workshop em Engenharia de Requisitos. Buenos Aires, Argentina, 2017.

[P085] Filho, Celso Sá; ASSIS, Denise; SOUSA, Aêda; JAQUEIRA, Aline; LUCENA, Márcia; MACIEL, Teresa; ALENCAR, Fernanda M. R. **Transformação do modelo i* em user stories: uma abordagem para documentação ágil baseada nas razões, intenções e NFR**. *In*: XIX Workshop em Engenharia de Requisitos. Quito, Equador, p. 99, 2016.

[P086] ALENCAR, Fernanda M. R.; PEDROZA, Flávio; CASTRO, Jaelson F. B.; AMORIM, Ricardo C. O. **New Mechanism for the Integration of Organizational Requirements and Object Oriented Modeling**. *In*: VI Workshop em Engenharia de Requisitos, p. 109–123, 2003.

[P087] FRANCH, X.; GRAU, G.; QUER, C. **A framework for the definition of metrics for actor-dependency models**. *In*: Proceedings. 12th IEEE International Requirements Engineering Conference, p. 348–349, 2004.

[P088] WERNECK, Vera M. B.; OLIVEIRA, Antônio de P. A.; LEITE, Julio C. S. P. **Comparing GORE Frameworks: i-star and KAOS**. *In*: XII Workshop em Engenharia de Requisitos, p. 15–26, 2009.

[P089] ALBUQUERQUE, Hidelberg; SILVA, Carla; ROUSY, Danielle. **NòmosBPMN: Adaptando o Nòmos para a Modelagem de Processos de Negócio**. *In*: XVII Workshop em Engenharia de Requisitos. Pucón, Chile, p. 23–25, 2014.

[P090] ALENCAR, Fernanda M. R.; CASTRO, Jaelson F. B.; CYSNEIROS, Gilberto; MYLOPOULOS, John. **From Early Requirements Modeled by the i* Technique to Later Requirements Modeled in Precise UML**. *In*: III Workshop em Engenharia de Requisitos, p. 92–108, 2000.

[P091] SANTANDER, Victor F. A.; CASTRO, Jaelson F. B. **Desenvolvendo Use Cases a partir de Modelagem Organizacional**. III Workshop em Engenharia de Requisitos, p. 158–180, 2000.

[P092] SANTANDER, Victor F. A.; CASTRO, Jaelson F. B. **Developing Use Cases from Organizational Modeling**. *In*: IV Workshop em Engenharia de Requisitos, p. 246–265, 2001.

[P093] SANTANDER, Victor F. A; CASTRO, Jaelson F. B. **Deriving use cases from organizational modeling**. *In*: Proceedings IEEE Joint International Conference on Requirements Engineering, p. 32–39, 2002.

[P094] SÁNCHEZ-ALONSO, Marisol; MURILLO, Juan M. Specifying **Cooperation Environment Requirements using Formal and Graphical Techniques**. *In*: V Workshop em Engenharia de Requisitos, p. 225–239, 2002.

[P095] ZAPATA-J, Carlos Mario; VARGAS-AGUDELO, Fabio Alberto. **Specification of problems from the business goals in the context of early software requirements elicitation**. Dyna, v. 81, n. 186, p. 193--199, 2014.

[P096] COOPER, K.; ABRAHAM, S. P.; UNNITHAN, R. S.; CHUNG, Lawrence; COURTNEY, S. **Integrating visual goal models into the Rational Unified Process**. Journal of Visual Languages and Computing, v. 17, n. 6, p. 551–583, 2006.

[P097] ZAPATA, C. M.; ACEVEDO, J. F.; NIÑO, D. M. **Weighted salience allocation to goals in the UNC-Method**. Dyna, v. 80, n. 180, p. 25–32, 2013.

[P098] DARIMONT, Robert; DELOR, Emmanuelle; ROUSSEL, Jean-Luc; RIFAUT, André. **Requirements engineering with GRAIL/KAOS: tell the requirements, all the requirements, and nothing else but the requirements**. *In*: Proceedings IEEE Joint International Conference on Requirements Engineering, p. 299, 2002.

[P099] LANDTSHEER, R. De; LETIER, E.; VAN LAMSWEERDE, A. **Deriving tabular event-based specifications from goal-oriented requirements models**. *In*: Proceedings. 11th IEEE International Requirements Engineering Conference, p. 200–210, 2003.

[P100] RIFAUT, A.; MASSONET, P.; MOLDEREZ, J-F.; PONSARD, C.; STADNIK, P.; VAN LAMSWEERDE, Axel; HUNG, Tran Van. **FAUST: formal analysis using specification tools**. *In*: Proceedings. 11th IEEE International Requirements Engineering Conference, p. 350, 2003.

[P101] NAKAGAWA, H; OHSUGA, A; HONIDEN, S. **A goal model elaboration for localizing changes in software evolution**. *In*: 21st IEEE International Requirements Engineering Conference (RE), p. 155–164, 2013.

[P102] LEONARDI, Maria Carmen; GIANDINI, Roxana. **Una estrategia de integración de Modelos de Objetivos con Análisis Comunicacional**. *In*: XVI Workshop em Engenharia de Requisitos. Montevideo, Uruguay, 2013.

[P103] HEAVEN, W; LETIER, E. **Simulating and optimising design decisions in quantitative goal models**. *In*: IEEE 19th International Requirements Engineering Conference, p. 79–88, 2011.

[P104] NGUYEN, Tuong Huan; VO, Bao Quoc; LUMPE, Markus; GRUNDY, John. **KBRE: A framework for knowledge-based requirements engineering**. Software Quality Journal, v. 22, n. 1, p. 87–119, 2014.

[P105] SILVA, Lyrene F.; LEITE, Júlio C. S. P. **Integração de Características Transversais Durante a Modelagem de Requisitos**. *In*: XIX Simpósio Brasileiro de Engenharia de Software, 2005.

[P106] SILVA, Lyrene F. Silva; LEITE, Julio C. S. P. **Uma Linguagem de Modelagem de Requisitos Orientada a Aspectos**. *In*: VIII Workshop em Engenharia de Requisitos, p. 13–25, 2005.

[P107] SUPAKKUL, S; CHUNG, L. **The RE-Tools: A multi-notational requirements modeling toolkit**. *In*: 20th IEEE International Requirements Engineering Conference (RE), p. 333–334, 2012.

[P108] AHMAD, Manzoor; BELLOIR, Nicolas; BRUEL, Jean Michel. **Modeling and verification of Functional and Non-Functional Requirements of ambient Self-Adaptive Systems**. Journal of Systems and Software, v. 107, p. 50–70, 2015.

[P109] ARAÚJO, Alex Lins De; CYSNEIROS, Luiz Marcio; WERNECK, Vera Maria B. **NDR-Tool: Uma Ferramenta de Apoio ao Reuso de Conhecimento em Requisitos Não Funcionais**. *In*: XVII Workshop em Engenharia de Requisitos. Pucón, Chile, p. 1–15, 2014.

[P110] LEAL, André L. C.; SOUSA, Henrique P.; LEITE, Julio C. S. P. **Modelo orientado à meta para estabelecer relações de contribuição mútua entre Proveniência, Transparência e Confiança**. *In*: XVII Workshop em Engenharia de Requisitos. Pucón, Chile, 2014.

[P111] XAVIER, Laís; ALENCAR, Fernanda; CASTRO, Jaelson F. B.; PIMENTEL, João. **Integração de Requisitos Não-Funcionais a Processos de Negócio: Integrando BPMN e NFR**. *In*: XIII Workshop em Engenharia de Requisitos, p. 107, 2010.

[P112] COUTO, Anselmo de Araújo; MARTINS, Luiz Eduardo Galvão. **Um processo de validação de requisitos NÃO-funcionais baseado no NFR-framework**. *In*: XII Workshop em Engenharia de Requisitos, p. 57–62, 2009.

[P113] HILL, R; WANG, Jun; NAHRSTEDT, K. **Quantifying non-functional requirements: a process oriented approach**. *In*: Proceedings. 12th IEEE International Requirements Engineering Conference, p. 352–353, 2004.

[P114] DANEVA, M.; KASSAB, M.; PONISIO, M. L.; WIERINGA, R. J.; ORMANDJIEVA, O. **Exploiting a Goal-Decomposition Technique to Prioritize Non-functional Requirements**. *In*: X Workshop em Engenharia de Requisitos, p. 190–196, 2007.

[P115] YU, Yijun; WANG, Yiqiao; MYLOPOULOS, J; *et al*. **Reverse engineering goal models from legacy code**. *In*: 13th IEEE International Conference on Requirements Engineering (RE'05), p. 363–372, 2005.

[P116] CYSNEIROS, Luiz Márcio; LEITE, Julio C. S. P. **Driving Non-Functional Requirements to Use Cases and Scenarios**. *In*: XV Simpósio Brasileiro de Engenharia de Software, p. 7–20, 2001.

[P117] SOUZA, Geórgia M. C.; SILVA, Ismênia da; CASTRO, Jaelson F. B. **Adapting the NFR framework to aspect-oriented requirements engineering**. *In*: XVII Simpósio Brasileiro de Engenharia de Software, p. 173–188, 2003.

[P118] GONZALES-BAIXAULI, B; LEITE, Júlio C. S. P.; MYLOPOULOS, J. **Visual variability analysis for goal models**. *In*: Proceedings. 12th IEEE International Requirements Engineering Conference, p. 198–207, 2004.

[P119] CLELAND-HUANG, Jane; SETTIMI, Raffaella; BENKHADRA, Oussama; *BEREZHANSKAYA, Eugenia; CHRISTINA, Selvia*. **Goal-centric Traceability for Managing Non-functional Requirements**. *In*: Proceedings of the 27th International Conference on Software Engineering. (ICSE ’05).New York, NY, USA, p. 362–371, 2005.

[P121] MEDEIROS, Maíra; SILVA, Lyrene; MEDEIROS, Ana Luisa. **A Semi-Automatic Strategy to Identify Crosscutting Concerns in PL-AOVgraph Requirement Models**. *In*: XVI Workshop em Engenharia de Requisitos. Montevideo, Uruguay, p. 46–59, 2013.

[P122] HALL, John G.; JACKSON, Michael; LANEY, Robin C.; NUSEIBEH, Bashar; RAPANOTTI, Lucia. **Relating software requirements and architectures using problem frames**. *In*: Proceedings IEEE Joint International Conference on Requirements Engineering, p. 137–144, 2002.

[P124] LANEY, R.; BARROCA, L.; JACKSON, M.; NUSEIBEH, B. **Composing requirements using problem frames**. *In*: Proceedings. 12th IEEE International Requirements Engineering Conference, p. 122–131, 2004.

[P125] LENCASTRE, Maria; ALVES, Keldjan; MELO, Renata; ALENCAR, Fernanda M. R. **Analyzing Basic Problem Frames in i* Context**. *In*: IX Workshop em Engenharia de Requisitos, p. 33–40, 2006.

[P126] SMITH, Margaret H.; HAVELUND, Klaus. **Requirements Capture with RCAT.** *In*: 16th IEEE International Requirements Engineering Conference, p. 183–192, 2008.

[P127] GLINZ, Martin. **Very Lightweight Requirements Modeling**. *In*: 18th IEEE International Requirements Engineering Conference, p. 385–386, 2010.

[P128] TAMBOURIS, Efthimios; KALIVA, Eleni; LIAROS, Michail; TARABANIS, Konstantinos. **A reference requirements set for public service provision enterprise architectures**. Software and Systems Modeling, v. 13, n. 3, p. 991–1013, 2014.

[P129] GÜRSES, Seda; SEGURAN, Magali; ZANNONE, Nicola. **Requirements engineering within a large-scale security-oriented research project: Lessons learned**. Requirements Engineering, v. 18, n. 1, p. 43–66, 2013.

[P131] JURETA, Ivan J.; BORGIDA, Alex; ERNST, Neil A.; MYLOPOULOS, John. **Techne: Towards a New Generation of Requirements Modeling Languages with Goals, Preferences, and Inconsistency Handling**. *In*: 18th IEEE International Requirements Engineering Conference, p. 115–124, 2010.

[P135] YSKOUT, Koen; SCANDARIATO, Riccardo; JOOSEN, Wouter. **Change patterns: Co-evolving requirements and architecture**. Software and Systems Modeling, v. 13, n. 2, p. 625–648, 2014.

[P136] COMPAGNA, Luca; EL KHOURY, Paul; KRAUSOVÁ, Alžběta; MASSACCI, Fabio; ZANNONE, Nicola. **How to integrate legal requirements into a requirements engineering methodology for the development of security and privacy patterns**. Artificial Intelligence and Law, v. 17, n. 1, p. 1–30, 2009.

[P137] MORDECAI, Yaniv; DORI, Dov. **Model-based requirements engineering: Architecting for system requirements with stakeholders in mind**. IEEE International Symposium on Systems Engineering, 2017.

[P138] DULAC, Nicolas; VIGUIER, Thomas; LEVESON, Nancy G.; STOREY, Margaret-Anne D. **On the use of visualization in formal requirements specification**. *In*: Proceedings IEEE Joint International Conference on Requirements Engineering, p. 71–80, 2002.

[P139] MARQUES, Milena Rota Sena; SIEGERT, Eliane; BRISOLARA, Lisane de. **Uma Abordagem para Engenharia de Requisitos no Domínio de Software Embarcado**. *In*: XVI Workshop em Engenharia de Requisitos. Montevideo, Uruguay, p. 14, 2013.

[P140] AMYOT, Daniel; ANDA, Amal Ahmed; BASLYMAN, Malak; LESSARD, Lysanne; BRUEL, Jean-Michel. **Towards Improved Requirements Engineering with SysML and the User Requirements Notation**. *In*: IEEE 24th International Requirements Engineering Conference (RE), p. 329–334, 2016.

[P141] MHENNI, Faïda; CHOLEY, Jean Yves; PENAS, Olivia; PLATEAUX, Régis; HAMMADI, Moncef. **A SysML-based methodology for mechatronic systems architectural design**. Advanced Engineering Informatics, v. 28, n. 3, p. 218–231, 2014.

[P142] COLOMBO, Pietro; KHENDEK, Ferhat; LAVAZZA, Luigi. **Bridging the gap between requirements and design: An approach based on Problem Frames and SysML**. Journal of Systems and Software, v. 85, n. 3, p. 717–745, 2012.

[P143] PAPATHANASIOU, Jason; KENWARD, Robert. **Design of a data-driven environmental decision support system and testing of stakeholder data-collection**. Environmental Modelling and Software, v. 55, p. 92–106, 2014.

[P144] FERREIRA, T; GORLACH, Ia A; AUTHOR AFFILIATIONS, Nmmuacza. **Development of an Automated Guided Vehicle Controller Using a Model-Based Systems Engineering Approach**. South African Journal of Industrial Engineering August, v. 27, n. 2, p. 206–217, 2016.

[P145] GUERRA, Esther; LARA, Juan de; KOLOVOS, Dimitrios S.; PAIGE, Richard F.; SANTOS, Osmar M. dos; **Engineering model transformations with transML**. Software & Systems Modeling, v. 12, n. 3, p. 555–577, 2013.

[P146] GUILLERM, Romaric; DEMMOU, Hamid; SADOU, Nabil. **Safety evaluation and management of complex systems: A system engineering approach**. Concurrent Engineering, v. 20, n. 2, p. 149–159, 2012.

[P147] WU, Dazhong; ZHANG, Linda L.; JIAO, Roger J.; LU, Roberto F. **SysML-based design chain information modeling for variety management in production reconfiguration**. Journal of Intelligent Manufacturing, v. 24, n. 3, p. 575–596, 2013.

[P148] SOARES, Michel dos S.; VRANCKEN, Jos; VERBRAECK, Alexander. **User requirements modeling and analysis of software-intensive systems**. Journal of Systems and Software, v. 84, n. 2, p. 328–339, 2011.

[P149] BRACE, William; EKMAN, Kalevi. **CORAMOD: A checklist-oriented model-based requirements analysis approach**. Requirements Engineering, v. 19, n. 1, p. 1–26, 2014.

[P150] HOLT, Jon; PERRY, Simon; PAYNE, Richard; BRYANS, Jeremy; HALLERSTEDE, Stefan; HANSEN, Finn Overgaard. **A Model-Based Approach for Requirements Engineering for Systems of Systems**. Systems Journal, IEEE, v. 9, n. 1, p. 252–262, 2015.

[P151] GILLAIN, Joseph; BURNAY, Corentin; JURETA, Ivan; FAULKNER, Stéphane. **AnalyticGraph.com: Toward Next Generation Requirements Modeling and Reasoning Tools**. *In*: IEEE 24th International Requirements Engineering Conference, p. 341–346, 2016.

[P152] JURETA, Ivan J.; FAULKNER, Stéphane; SCHOBBENS, Pierre Yves. **Clear justification of modeling decisions for goal-oriented requirements engineering**. Requirements Engineering, v. 13, n. 2, p. 87–115, 2008.

[P153] GIORGINI, Paolo; MASSACCI, Fabio; MYLOPOULOS, John; ZANNONE, Nicola. **ST-tool: a CASE tool for security requirements engineering**. *In*: 13th IEEE International Conference on Requirements Engineering (RE'05), p. 451–452, 2005.

[P154] PINTO, Rosa Candida; SILVA, Carla; CASTRO, Jaelson F. B. **A process for requirement traceability in agent oriented development**. *In*: VIII Workshop em Engenharia de Requisitos, p. 221–232, 2005.

[P155] MARTÍNEZ, Alicia; VELASCO, Carmen; MORALES, Eliel; ESTRADA, Hugo; GAMA, Luis A. **Lenguaje de Especificación para el Framework Tropos**. *In*: XIII Workshop em Engenharia de Requisitos, p. 121–132, 2010.

[P156] ESTRADA, Hugo; CASTRO, Jaelson F. B.; PASTOR, Oscar; MARTÍNEZ, Alicia. **Goal-based organizational modeling oriented towards late requirements generation using the Tropos Framework**. *In*: XVII Simpósio Brasileiro de Engenharia de Software, p. 142–156, 2003.

[P157] MARTÍNEZ, Alicia; PASTOR, Oscar; ESTRADA, Hugo. **Closing the Gap between Organizational Modeling and Information System Modeling**. *In*: VI Workshop em Engenharia de Requisitos, p. 93–108, 2003.

[P158] CASTOR, Andréa Pinto; PINTO, Rosa Candida; SILVA, Carla; CASTRO, Jaelson F. B. **Towards Requirement Traceability in TROPOS**. *In*: VII Workshop em Engenharia de Requisitos, p. 189–200, 2004.

[P159] CYSNEIROS, Luiz M.; WERNECK, Vera; YU, Eric. **Evaluating Methodologies: A Requirements Engineering Approach Through the Use of an Exemplar**. *In*: VII Workshop em Engenharia de Requisitos, p. 112–126, 2004.

[P160] GONZÁLEZ-BAIXAULI, Bruno; LAGUNA, Miguel A.; LEITE, Julio C. S. P. **Análisis de Variabilidad con Modelos de Objetivos**. *In*: VII Workshop em Engenharia de Requisitos, p. 77–87, 2004.

[P161] MARTÍNEZ, Alicia; PASTOR, Oscar; ESTRADA, Hugo. **A pattern language to join early and late requirements**. *In*: VII Workshop em Engenharia de Requisitos, p. 51–64, 2004.

[P162] PINTO, Rosa Candida; SILVA, Carla; CASTRO, Jaelson F. B. **Support for requirement traceability: The Tropos case**. *In*: XIX Simpósio Brasileiro de Engenharia de Software, p. 31–46, 2005.

[P163] AMYOT, Daniel; LOGRIPPO, Luigi; WEISS, Michael. **Generation of test purposes from Use Case Maps**. Computer Networks, v. 49, n. 5, p. 643–660, 2005.

[P164] HASSINE, Jameleddine; RILLING, Juergen; DSSOULI, Rachida. **Use Case Maps as a property specification language**. Software and Systems Modeling, v. 8, n. 2, p. 205–220, 2009.

[P165] MAGABLEH, Basel; BARRETT, Stephen. **Productivity evaluation of Self-Adaptive software model driven architecture**. International Journal of Information Technology and Web Engineering, v. 6, n. 4, p. 1–19, 2011.

[P166] BERENBACH, Brian; SCHNEIDER, Florian; NAUGHTON, Helmut. **The use of a requirements modeling language for industrial applications**. *In*: 20th IEEE International Requirements Engineering Conference (RE), p. 285–290, 2012.

[P167] SCHNEIDER, Florian; BRUEGGE, Bernd; BERENBACH, Brian. **A tool implementation of the unified requirements modeling language as enterprise architect add-in**. *In*: 21st IEEE International Requirements Engineering Conference (RE), p. 334–335, 2013.

[P168] HASSINE, Jameleddine. **Early modeling and validation of timed system requirements using Timed Use Case Maps**. Requirements Engineering, v. 20, n. 2, p. 181–211, 2015.

[P169] CHAWLA, Shailey; SRIVASTAVA, Sangeeta; BEDI, Punam. **Improving the quality of web applications with web specific goal driven requirements engineering**. International Journal of System Assurance Engineering and Management, v. 8, n. S1, p. 65–77, 2017.

[P170] AMYOT, Daniel. **Introduction to the user requirements notation: Learning by example**. Computer Networks, v. 42, n. 3, p. 285–301, 2003.

[P171] DURAN, Mustafa Berk; MUSSBACHER, Gunter. **Investigation of feature run-time conflicts on goal model-based reuse**. Information Systems Frontiers, v. 18, n. 5, p. 855–875, 2016.

[P172] PETRIU, Dorin Bogdan; WOODSIDE, Murray. **Software performance models from system scenarios**. Performance Evaluation, v. 61, n. 1, p. 65–89, 2005.

[P173] JASKÓ, Szilárd; DULAI, Tibor; MUHI, Dániel; TARNAY, Katalin. **Test aspect of requirement specification**. Computer Standards and Interfaces, v. 32, n. 1–2, p. 1–9, 2010.

[P174] POURSHAHID, Alireza; AMYOT, Daniel; PEYTON, Liam; *GHANAVATI, Sepideh; CHEN, Pengfei; WEISS, Michael; FORSTER, Alan J.* **Business process management with the user requirements notation**. Electronic Commerce Research, v. 9, n. 4, p. 269--316, 2009.

[P175] WEISS, Michael; AMYOT, Daniel. **Business Process Modeling with URN**. International Journal of E-Business Research, v. 1, n. 3, p. 63–90, 2005.

[P176] HASSINE, Jameleddine. **Describing and assessing availability requirements in the early stages of system development**. Software and Systems Modeling, v. 14, n. 4, p. 1455–1479, 2015.

[P177] MEDVE, Anna. **Advanced steps with standardized languages in the re-engineering process**. Computer Standards and Interfaces, v. 30, n. 5, p. 315–322, 2008.

[P178] LIU, Yanji; SU, Yukun; YIN, Xinshang; MUSSBACHER, Gunter **Combined goal and feature model reasoning with the User Requirements Notation and jUCMNav**. *In*: IEEE 22nd International Requirements Engineering Conference (RE), p. 321–322, 2014.

[P179] MUSSBACHER, Gunter; GHANAVATI, Sepideh; AMYOT, Daniel. **Modeling and Analysis of URN Goals and Scenarios with jUCMNav**. *In*: 17th IEEE International Requirements Engineering Conference, p. 383–384, 2009.

[P180] GHANAVATI, Sepideh; AMYOT, Daniel; PEYTON, Liam. **Compliance Analysis Based on a Goal-oriented Requirement Language Evaluation Methodology**. *In*: 17th IEEE International Requirements Engineering Conference, p. 133–142, 2009.

[P181] LIU, Lin; YU, Eric S. K. **Designing information systems in social context: A goal and scenario modelling approach**. Information Systems, v. 29, n. 2, p. 187–203, 2004.

[P182] AMYOT, Daniel; GHANAVATI, Sepideh; HORKOFF, Jennifer; PEYTON, Liam; YU, Eric S. K. **Evaluating goal models within the goal-oriented requirement language**. International Journal of Intelligent Systems, v. 25, n. 8, p. 841–877, 2010.

[P183] AMYOT, Daniel; MUSSBACHER, Gunter. **Bridging the Requirements/Design Gap in Dynamic Systems with Use Case Maps (UCMs).** *In*: Proceedings of the 23rd International Conference on Software Engineering. Washington, DC, USA: IEEE Computer Society, p. 743–744, 2001.

[P184] HASSINE, Jameleddine; RILLING, Juergen. **An ASM Operational Semantics for Use Case Maps ASM models for Use Case Maps**. *In*: 13th IEEE International Conference on Requirements Engineering, (p. 467–468, 2005.

[P185] MASSBACHER, Gunter. **Evolving use case maps as a scenario and workflow description language**. *In*: X Workshop em Engenharia de Requisitos, p. 56–67, 2007.

[P186] MERRICK, Peter; BARROW, Patrick. **Testing the predictive ability of a requirements pattern language**. Requirements Engineering, v. 10, n. 2, p. 85–94, 2005.

[P187] ALCÁZAR, Enrique G.; MONZÓN, Antonio. **A process framework for requirements analysis and specification**. *In*: Proceedings Fourth International Conference on Requirements Engineering (ICRE 2000), p. 27–35, 2000.

[P188] MEHNER, Katharina; MONGA, Mattia; TAENTZER, Gabriele. **Interaction Analysis in Aspect-Oriented Models**. *In*: 14th IEEE International Requirements Engineering Conference (RE'06), p. 69–78, 2006.

[P189] SIQUEIRA, Fábio Levy. **Comparing the comprehensibility of requirements models: An experiment replication**. Information and Software Technology, v. 96, p. 1–13, 2018.

[P190] HADAR, Irit; REINHARTZ-BERGER, Iris; KUFLIK, Tsvi; PERINI, Anna; RICCA, Filippo; SUSI, Angelo. **Comparing the comprehensibility of requirements models expressed in Use Case and Tropos: Results from a family of experiments**. Information and Software Technology, v. 55, n. 10, p. 1823–1843, 2013.

[P192] OHNISHI, A. **A visual software requirements definition method**. *In*: Proceedings of IEEE International Conference on Requirements Engineering, p. 194–201, 1994.

[P193] PISTORE, Marco; ROVERI, Marco; BUSETTA, Paolo. **Requirements-Driven Verification of Web Services**. Electronic Notes in Theoretical Computer Science, v. 105, p. 1–12, 2004.

[P194] PIMENTEL, João; LUCENA, Márcia; CASTRO, Jaelson; SILVA, Carla; SANTOS, Emanuel; ALENCAR, Fernanda M. R. **Deriving software architectural models from requirements models for adaptive systems: The STREAM-A approach. Requirements Engineering**, v. 17, n. 4, p. 259–281, 2012.

[P195] YU, Yijun; LEITE, Julio C. S. P.; MYLOPOULOS, John. **From goals to aspects: discovering aspects from requirements goal models**. *In*: Proceedings of the 12th IEEE International Requirements Engineering Conference. Kyoto, Japan, p. 38–47., 2004.

[P196] SIMÃO, Jean Marcelo; PANETTO, Hervé; LIAO, Yongxin; Stadzisz, Paulo C. **A Notification-Oriented Approach for Systems Requirements Engineering**. *In*: 23rd IPSE International Conference on Transdisciplinary Engineering. p. 229–238. Curitiba, Brasil, 2016.

II. REFERÊNCIAS COMPLEMENTARES

[C001] WANG, J.; WANG, Q. **Analyzing and predicting software integration bugs using network analysis on requirements dependency network**. Requirements Engineering, v. 21, n. 2, p. 161–184, 2016.

[C002] THE OPEN GROUP®. **Archimate Tool**. Disponível em: <https://www.archimatetool.com/> Acesso em Mai/2018.

[C003] THOMAS, James J.; COOK, Kristin A**. Illuminating the Path: The R&D Agenda for Visual Analytics**. National Visualization and Analytics Center, n. Janeiro, 2005.

[C004] CLAVEL, M.; DURÁN, F.; EKER, S.; LINCOLN, P.; MARTÍ-OLIET, N.; MESEGUER, J.; QUESADA, J. F. **Maude: Specification and programming in rewriting logic**. Theoretical Computer Science, v. 285, n. 2, p. 187–243, 2002.

[C005] KISSOUM, Y; SAHNOUN, Z. **A Formal Approach for Functional and Structural Test Case Generation in Multi-Agent Systems**. IEEE International Conference on Computer Systems and Applications, p. 76–83, 2007.

[C006] OHNISHI, Atsushi; TOKUDA, Norihiro. **Visual Software Requirements Definition Environment**. In: Computer Software and Applications Conference. Washington, DC, USA. p. 624–629, 1997.

[C007] AMYOT, Daniel; MUSSBACHER, Gunter. **User Requirements Notation: The First Ten Years, The Next Ten Years**. Journal of Software, v. 6, n. 5, p. 747–768, 2011.

[C008] HASSINE, Jameleddine; RILLING, Juergen. **An ASM Operational Semantics for Use Case Maps ASM models for Use Case Maps**. In: 13th IEEE International Conference on Requirements Engineering, p. 467–468, 2005.

[C009] BUHR, Ray J. A.; CASSELMAN, R. S.; PEARCE, T. W. **Design Patterns with Use Case Maps**. SCE Journal, 1996.

[C010] BUHR, Ray J. A. **Use Case Maps for Attributing Behaviour to System Architecture**. In: Fourth International Workshop on Parallel and Distributed Real Time Systems (WPDRTS), 1996.

[C011] BUHR, Ray J. A.; HUBBARD, A. **Use Case Maps for Engineering Real Time and Distributed Computer Systems : A Case Study of an ACE-Framework Application**. In: Thirtieth Hawaii International Conference on System Sciences. Wailea, HI, USA: IEEE Computer Society, 1997.

[C012] BORDELEAU, F.; BUHR, R. J. A. **The UCM-ROOM Design Method : from Use Case Maps to Communicating State Machines**. In: International Conference and Workshop on Engineering of Computer-Based Systems. Monterey, CA, USA, p. 1–15, 1997.

[C013] BUHR, Ray J A; ELAMMARI, M; GRAY, T; MANKOVSKI, S.; PINARD, D. **Understanding and Defining the Behaviour of Systems of Agents with Use Case Maps**. In: Practical Applications of Intelligent Agents and MultiAgents Conference., p. 4-15, 1997.

[C014] MIGA, Andrew. **Application of Use Case Maps to System Design with Tool Support (M.Sc. Thesis)**. Carleton University, Ottawa, Canada, 1998.

[C015] AMYOT, Daniel. **Bridging the Gap Between Requirements and Design with Use Case Maps (presentation)**. In: Proceedings of the 23rd International Conference on Software Engineering, 2001.

[C016] MOODY, Daniel L. **The "Physics" of Notations : Toward a Scientific Basis for Constructing Visual Notations in Software Engineering**. IEEE Transactions on Software Engineering, v. 35, n. 6, p. 756–779, 2009.

[C017] HELMING, Jonas; KOEGEL, Maximilian; SCHNEIDER, Florian; HAEGER, Michael; KAMINSKI, Christine; BRUEGGE, Bernd; BERENBACH, Brian. **Towards a Unified Requirements Modeling Language**. In: Fifth International Workshop on Requirements Engineering Visualization (REV). Sydney, Australia: IEEE Computer Society, p. 53–57, 2010.

[C018] GIORGINI, Paolo; MASSACCI, Fabio; MYLOPOULOS, John; ZANNONE, Nicola. **Requirements engineering for trust management : model, methodology, and reasoning**. International Journal of Information Security, v. 5, n. 4, p. 257–274, 2006.

[C019] BRESCIANI, Paolo; PERINI, Anna.; GIORGINI, Paolo; GIUNCHIGLIA, Fausto; MYLOPOULOS, John; **Tropos: An Agent-Oriented Software**

Development Methodology. Autonomous Agents and Multi-Agent Systems, v. 8, p. 203–236, 2004.

[C020] GIORGINI, Paolo; KOLP, Manuel; MYLOPOULOS, John; PISTORE, Marco. **The Tropos Methodology**. *In*: BERGENTI, Federico; GLEIZES, Marie-Pierre; ZAMBONELLI, Franco (Orgs.). **Methodologies and Software Engineering for Agent Systems: The Agent-Oriented Software Engineering Handbook**. Boston, MA: Springer US, p. 89–106, 2004.

[C021] DARDENNE, Anne; VAN LAMSWEERDE, A.; FICKAS, Stephen. **Goal-directed requirements acquisition**. Science of Computer Programming, v. 20, p. 3–50, 1993.

[C022] LAPOUCHNIAN, Alexei. **Goal-Oriented Requirements Engineering: An Overview of the Current Research**, Technical Report, Department of Computer Science, University of Toronto, 2005.

[C023] MYLOPOULOS, John; CHUNG, Lawrence. **From Object-Oriented to Goal-Oriented Requirements Analysis**. Communications of the ACM, 1999.

[C024] LAMSWEERDE, Axel Van. **Elaborating Security Requirements by Construction of Intentional Anti-Models**. In: Proceedings of the 26th International Conference on Software Engineering. Washington, DC, USA: IEEE Computer Society, p. 148–157, 2004.

[C025] RESPECT-IT. **Objectiver General Overview**. Disponível em <http://www.objectiver.com/index.php?id=6>. Acesso em Mar/2018.

[C026] HEAVEN, W.; FINKELSTEIN, A. **UML profile to support requirements engineering with KAOS**. IEE Proceedings - Software, v. 151, n. 1, p. 10–27, 2004.

[C027] CHUNG, Lawrence; NIXON, Brian A.; YU, Eric. **Using Non-Functional Requirements to Systematically Select Among Alternatives in Architectural Design**. In: Proceedings of 1st International Workshop on Architectures for Software Systems, p. 31-43, 1994.

[C028] CHUNG, Lawrence; NIXON, Brian A.; YU, Eric. **Using Non-Functional Requirements to Systematically Support Change***. In: Proceedings of the Second IEEE International Symposium on Requirements Engineering. York, UK, p. 132–139, 1995.

[C029] MYLOPOULOS, John; CHUNG, Lawrence; NIXON, Brian A. **Representing and Using Non-Functional Requirements: A Process-Oriented Approach**. IEEE Transactions on Software Engineering, v. 18, p. 483--497, 1992.

[C030] CHUNG, Lawrence; NIXON, Brian A.; YU, Eric; MYLOPOULOS, John. **Non-Functional Requirements in Software Engineering**. Boston, MA, USA: Springer Boston, copyright by Kluwer Academic Publishers, 2000.

[C031] CHUNG, Lawrence. **Non-Functional Requirements (presentation)**. Disponível em: <http://www.utdallas.edu/~chung/RE/NFR-18.pdf>. Acesso em Jun/2018.

[C032] ITU-T, International Telecommunications Union. **User Requirements Notation (URN) – Language definition. Recomendation Z.151 (11/2008)**.

2008. Disponível em: <http://www.itu.int/rec/T-REC-Z.151/en>. Acesso em Mai/2018.

[C033] AMYOT, Daniel; MUSSBACHER, Gunter. **Development of Telecommunications Standards and Services with the User Requirements Notation**. In: Workshop on ITU System Design Languages, p. 15--16, 2008.

[C034] AMYOT, **Daniel. User Requirements Notation (URN): Application and Research Areas (presentation).** Disponível em: <ftp://www.cs.toronto.edu/pub/eric/O/Amyot07visit-URN/URN-UofT-2007.ppt>. Acesso em Abr/2018.

[C035] YU, Eric S. K. **Modelling Strategic Relationships for Process Reengineering (PHD Thesis)**. University of Toronto, 1995.

[C036] YU, Eric S. K. **Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering**. In: Proceedings of the Third IEEE International Symposium on Requirements Engineering. Annapolis, MD, USA, p. 226–235, 1997.

[C037] CARES, Carlos; FRANCH, Xavier; LÓPEZ, Lidia; MARCO, Jordi. **Definition and uses of the i* Metamodel**. In: CEUR Proceedings of the 4th International i* Workshop. Hammamet, Tunisia, v. 586, p. 20–25, 2010.

[C038] YU, Eric S. K. **Social modeling and i***. In: BORGIDA, A. T.; CHAUDHRI, V. K.; GIORGINI, Paolo; Yu, Eric S. K. (Eds.). **Conceptual Modeling: Foundations and Applications**. Lecture Notes in Computer Science, v. 5600, p. 99–121, 2009.

[C040] SILVA, Lyrene Fernandes da. **Uma Estratégia Orientada a Aspectos para Modelagem de Requisitos (PHD Thesis)**. Pontifícia Universidade Católica do Rio de Janeiro, 2006.

[C041] JACKSON, Michael. **Problem Frames: Analyzing and Structuring Software Development Problems**. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2001.

[C042] JACKSON, Michael. **Software Requirements' Specifications: A Lexicon of Practice, Principles and Prejudices**. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1995.

[C043] OMG, Object Management Group. **Systems Modeling Language (SysML), Version 1.5**. SysML formal/17-05-01. 2017. Disponível em: <https://www.omg.org/spec/SysML/1.5/>. Acesso em: Jun-2018.

[C044] OMG, Object Management Group. **Unified Modeling Language (UML), Version 2.5. UML formal/15-03-01**. Disponível em: <https://www.omg.org/spec/UML/2.5/>. Acesso em: Jun-2018.

[C045] FRIEDENTHAL, Sanford; MOORE, Alan; STEINER, Rick. **OMG Systems Modeling Language (OMG SysMLTM) Tutorial**. 2009. Disponível em: <http://www.omgsysml.org/INCOSE-OMGSysML-Tutorial-Final-090901.pdf>. Acesso em: Jun-2018.

[C046] FRIEDENTHAL, Sanford; MOORE, Alan; STEINER, Rick. **A Practical Guide to SysML**. 3rd. ed., The Morgan Kaufmann / OMG Press, 2014.

[C047] JACOBSON, Ivar; CHRISTERSON, Magnus; JONSSON, Patrik; OVERGAARD, Gunnar. **Object-Oriented Software Engineering: A Use Case Driven Approach**, HarlowEssex England Addison, 1992.

[C048] RUMBAUGH, James; JACOBSON, Ivar; BOOCH, Grady. **Unified Modeling Language Reference Manual**. 2nd. ed. , Addison-Wesley, 2004.

[C049] BUHR, Ray J. A.; CASSELMAN., Ron S. **Use Case Maps for Object-oriented Systems**. Prentice-Hall, Inc, 1995.

[C050] CHEN, Deng-Jyi; TSAI, Ming-Jyh; HUANG, Chung-Yuan. **Visual-based Software Construction Methodology**. In: Proceedings of the 10th WSEAS International Conference on Computers. Wisconsin, USA, p. 841–846, 2006.

[C051] REDDIVARI, Sandeep; NIU, Nan. **SDVisu: A Tool for Clustering-based Visual Exploration of Static Dependencies**. In: Computing Conference. London, UK, p. 1373–1374, 2017.

[C052] SCHNEIDER, Florian; BERENBACH, Brian. **The Unified Requirements Modeling Language: Shifting the Focus to Early Requirements Elicitation**. In: Comparing Requirements Modeling Approaches Workshop. Rio de Janeiro, Brazil: IEEE, 2013, p. 31–36.

[C053] ISO, International Organization for Standardization. **Object Process Methodology (OPM) - ISO/PAS 19450:2015**. Disponível em: <https://www.iso.org/standard/62274.html>. Acesso em Jun-2018.

[C054] ITU-T, International Telecommunications Union. **URN Focus Group in ITU-T**. Disponível em: <https://www.itu.int/ITU-T/studygroups/com17/urn/focusgroup.html>. Acesso em Jun-2018.

[C055] LIU, L.; YU, Eric S. K. **GRL - Goal-oriented Requirements Language**. Disponível em: <http://www.cs.toronto.edu/km/GRL/>. Acesso em Jun-2018.

[C056] OMG, Object Management Group. **Systems Modeling Language (SysML), Version 1.0. 07-09-01**. Disponível em: <https://www.omg.org/spec/SysML/1.0/>. Acesso em: Jun-2018.

[C057] INCOSE, International Council for Systems Engineering. **UML For Systems Engineering RFP**. Disponível em: <https://www.omg.org/syseng/UML_for_SE_RFP.htm>. Acesso em: Jun-2018.

[C058] OMG, Object Management Group. **Unified Modeling Language (UML), Version 1.1**. UML formal/97-12-01. Disponível em: <https://www.omg.org/spec/UML/1.1/>. Acesso em: Jun-2018.

[C059] NOVAES, Paulo J. D.; SIMÃO, Jean M.; STADZISZ, Paulo C. **Integration between Requirements Modeling and Software Development in the Notification Oriented Paradigm: A Security System Case Study**. *In*: IX Computer on the Beach (COTB 2018). p. 432–441 Florianópolis, Brasil, 2018.

[C060] Sparx Systems ®. **Enterprise Architect**. Disponível em: <https://sparxsystems.com/products/ea/>. Acesso em: Jun-2018.

[C061] MKLab Company ®. **StarUML Tool**. Disponível em: <http://staruml.io/>. Acesso em: Jun-2018.

[C062] THE OPEN GROUP ®. **An Introduction to ArchiMate Modeling Language, v2.1**, **2012**. Disponível em: < https://opengroup.org/archimate > Acesso em: Jun-2018.

[C063] HEAVEN, William; FINKELSTEIN, Anthony. **A UML profile to support requirements engineering with KAOS**. IEE SW, v. 151, n. 1, p. 10–27, 2004.

[C064] ITU-T, International Telecommunications Union. **User Requirements Notation (URN) – Language definition. Recomendation Z.151 (10/2012)**. 2012. Disponível em: <http://www.itu.int/rec/T-REC-Z.151/en>. Acesso em Mai/2018.

APÊNDICE D – TERMINOLOGIA

Este apêndice consiste em uma coletânea de termos e definições relevantes no contexto de SWE, SE e RE, que foram amplamente empregados nesta dissertação.

1. SWE: *software*

a) "***Software*** é o todo ou parte dos programas, procedimentos, regras e documentação associada de um sistema de processamento de informações" (ISO/IEC/IEEE 24765, 2010).
b) "***Software*** é definido como programas de computador, procedimentos e, possivelmente, documentação e dados associados relativos à operação de um sistema de computador (ISO/IEC/IEEE 24765, 2010).

2. SWE: Engenharia de *Software*

a) "**Engenharia de *Software*** é a aplicação de uma abordagem sistemática, disciplinada e quantificável ao desenvolvimento, operação e manutenção de *software*; isto é, a aplicação da engenharia ao software" (BOURQUE e FAIRLEY, 2014).
b) "**Engenharia de *software*** (ou *Software Engineering* – *SWE*) pode ser definida como a aplicação sistemática de conhecimento científico e tecnológico, métodos e experiência para o projeto, implementação, teste e documentação de *software"* (ISO/IEC/IEEE 24765, 2010).
c) "**Engenharia de *software*** é uma disciplina de engenharia que se preocupa com todos os aspectos da produção de *software*, desde os estágios iniciais da especificação do sistema até a manutenção do sistema depois que o sistema for utilizado" (SOMMERVILLE, 2010).

A primeira definição acima foi extraída do SWEBOK (*Software Engineering Body of Knowledge*), que é um guia para o corpo de conhecimentos sobre Engenharia de *Software*, que apresenta uma visão consensual dos especialistas da área (IEEE) sobre conhecimentos em Engenharia de *Software*. Inicialmente publicado em 2004, o SWEBOK atualmente está em sua terceira versão (v 3.0 de 2014). Dentre as quinze áreas de conhecimento do SWEBOK, a primeira abordada é a de "Requisitos de *Software*", cujo foco está na elicitação,

análise, especificação e validação de requisitos de *software* bem como com o gerenciamento de requisitos durante todo o ciclo de vida do produto *software.*

A segunda definição acima foi extraída do padrão ISO/IEC/IEEE 24765 apresenta o vocabulário e terminologia propostos pelas organizações ISO, IEC e IEEE para SWE e SE (ISO/IEC/IEEE 24765, 2010).

3. SWE: requisitos de *software*

a) "**Requisitos de *software*** expressam as necessidades e restrições colocadas em um produto de *software* que contribuem para a solução de um problema do mundo real" (BOURQUE e FAIRLEY, 2014).
b) "**Requisito de *software*** é uma capacidade do *software* necessária a um usuário para resolver um problema ou atingir a um objetivo" ou uma "capacidade do *software* que deve ser atingida ou possuída por um sistema ou componente do sistema para satisfazer um contrato, padrão, especificação ou qualquer outro tipo de documento formal imposto" (ISO/IEC/IEEE 24765, 2010).
c) "**Requisitos de *software*** são descrições do que o sistema deve fazer – serviços a serem providos e as restrições de operação. Estes requisitos refletem as necessidades dos clientes para um sistema que serve a um determinado propósito tal como controlar um dispositivo, colocar uma order ou buscar uma informação" (SOMMERVILLE, 2010). Também é possível distinguir que há requisitos abstratos de alto nível conhecidos como requisitos de usuário (*user requirements)* e que há requisitos de sistema (*system requirements)* que fornecem o detalhamento do que o sistema deve fazer (SOMMERVILLE, 2010).

4. SWE: relacionamentos entre requisitos

Tanto o SWEBOK e o padrão ISO/IEC/IEEE 24765 não definem especificamente termos para "relacionamentos", "inter-relacionamentos", "dependências" ou "interdependências" entre requisitos. Ainda assim, é possível extrair explicações sobre "relacionamentos" entre elementos de modelos:

a) No capítulo sobre métodos e modelos para Engenharia de *Software*, o SWEBOK explica o conceito de relacionamento entre elementos de um

modelo (genérico) da seguinte forma: “A expressão primária de um elemento de um modelo é uma entidade. Uma entidade pode representar artefatos concretos (e.g., processadores, sensores, robôs) ou artefatos abstratos (e.g. módulos de software ou protocolos de comunicação). Entidades do modelo são conectadas utilizando **relacionamentos** (em outras palavras), linhas ou operadores textuais entre as entidades alvo) (BOURQUE e FAIRLEY, 2014).

b) O padrão ISO/IEC/IEEE apresenta o conceito genérico de “**relacionamento**” (*relationship*) como “uma conexão semântica entre dois elementos de um modelo” ou “uma associação de interesse entre duas entidades” (ISO/IEC/IEEE 24765, 2010).

Neste trabalho, entende-se que os conceitos de “relacionamento” definidos acimas são plenamente aplicáveis a requisitos, uma vez que requisitos podem ser considerados como elementos ou entidades de um modelo, possuindo conexões semânticas (com significado definido), através da utilização de elementos gráficos de ligação (linhas) associadas a operadores textuais.

5. SWE: modelagem e modelo

a) “A **modelagem** fornece ao engenheiro de software uma abordagem organizada e sistemática para representar aspectos significativos do *software* sob estudo, facilitando a tomada de decisões sobre os *softwares* e seus elementos e a comunicação destas decisões para outros membros da comunidade de *stakeholders”* (BOURQUE e FAIRLEY, 2014).

b) “Um **modelo** é uma abstração ou simplificação de um componente de *software*” (BOURQUE e FAIRLEY, 2014).

c) **“Modelo** é uma coleção relacionada de instâncias de meta-objetos, representando (descrevendo ou prescrevendo) um sistema de informação, ou partes do mesmo, tal como um produto de *software*” ou “**Modelo** é uma abstração semântica fechada sobre um sistema ou uma descrição completa do sistema sob uma perspectiva particular” (ISO/IEC/IEEE 24765, 2010).

Neste trabalho, entende-se que o conceito de “modelo” da ISO/IEC/IEEE 24765 acima é suficiente genérico e se aplica a modelos gráficos de requisitos.

6. SE: sistema

a) “**Sistema** é um conjunto integrado de elementos, subsistemas ou montagens que realizam um objetivo definido. Esses elementos incluem produtos (hardware, software, firmware), processos, pessoas, informações, técnicas, serviços de instalação e outros elementos de suporte.” (INCOSE, 2015).

b) “**Sistema** é uma combinação de elementos interativos organizados para atingir um ou mais objetivos declarados” (ISO/IEC/IEEE 24765, 2010), (ISO/IEC/IEEE 15288, 2008).

7. SE: engenharia de sistemas

a) “**Engenharia de sistemas** é uma abordagem interdisciplinar e visa permitir a realização de sistemas bem-sucedidos” (INCOSE, 2015).

b) “**Engenharia de sistemas** é uma abordagem interdisciplinar que governa todo o esforço técnico e gerencial necessário para transformar um conjunto de necessidades, expectativas e restrições do cliente em uma solução bem como suportar toda a vida dessa solução” (ISO/IEC/IEEE 24765, 2010).

c) “**Engenharia de sistemas** é a aplicação multidisciplinar de princípios analíticos, matemáticos e científicos para formular, selecionar, desenvolver e manter uma solução ótima a partir de um conjunto de opções viáveis que tenham riscos aceitáveis, satisfaçam as necessidades operacionais do usuário e minimizem o desenvolvimento e os custos durante o ciclo de vida do sistema, enquanto equilibra os interesses das partes interessadas” (WASSON, 2005).

A primeira definição apresentada nesta seção foi extraída do SEBoK (*Systems Engineering Body of Knowledge*), que é o documento (ou “manual”) que fornece uma compilação de conhecimentos chave e referências sobre engenharia de sistemas organizados e explicados de forma a auxiliar uma ampla variedade de usuários (INCOSE, 2015). O SEBoK é organizado conjuntamente pela INCOSE (*International Council on Systems Engineering*), pelo SERC (*Systems Engineering Research Center*) e pela *IEEE Computer Society*.

8. SE: requisitos de sistemas e necessidades - *needs*

a) O SEBoK estabelece uma distinção entre “necessidade” (*need*) e “requisito” (*requirement*), na seguinte forma: “Uma **necessidade** é algo desejado ou requerido. Para um sistema, **necessidades** são capacidades ou coisas que estão faltando e que são procuradas (*wanted*) ou desejadas (*desired)* por uma ou mais partes interessadas (*stakeholders*). **Requisitos** são sentenças formais estruturadas que podem ser validadas ou verificadas, sendo que pode haver mais de um requisito (requirement) para cada necessidade (*need*)” (INCOSE, 2015).
b) Uma definição similar à da INCOSE citada acima é dada por Friedenthal em 2014, que adicionalmente distingue entre **necessidades** de usuários (*stakeholders needs)*, **requisitos de sistema** (*system requirements)* e **requisitos de componentes** (*component requirements*) (FRIEDENTHAL et al., 2014).
c) “**Requisito** é qualquer condição, característica ou capacidade que deve ser alcançada e é essencial para a capacidade do item final de executar sua missão no ambiente em que ele deve operar. **Requisitos** devem ser verificáveis” (WASSON, 2005).

9. SE: modelo

a) “Um **modelo** refere-se a uma abstração ou representação de um sistema, entidade, fenômeno ou processo de interesse” (INCOSE, 2015).
b) “Um **modelo** é uma representação de um ou mais conceitos que podem ser realizados no mundo físico” (FRIEDENTHAL et al., 2014).

10. SE: relacionamentos ou caminhos gráficos entre requisitos

O SEBoK não especificamente termos para “relacionamentos”, “inter-relacionamentos”, “dependências” ou “interdependências” entre requisitos no contexto de SE. Uma definição próxima seria a seguinte:

a) A linguagem de modelagem SysML, atual padrão para a área de SE, estabelece um conjunto de relacionamentos entre requisitos nomeados como **caminhos gráficos** (*graphical paths*) (OMG SysML v1.5, 2017).

11. RE: engenharia de requisitos

a) “**Engenharia de requisitos** é a ciência e disciplina preocupada com análise e documentação de requisitos” (ISO/IEC/IEEE 24765, 2010).
b) “**Engenharia de requisitos** é uma função interdisciplinar que realiza a mediação entre os domínios do adquirente e fornecedor para estabelecer e manter os requisitos a serem atendidos pelo sistema, software ou serviço de interesse” (ISO/IEC/IEEE 29148, 2011).
c) “**Engenharia de requisitos** é uma abordagem sistemática e disciplinada para a especificação e gestão ou requisitos com os seguintes objetivos: 1. Conhecer os requisitos relevantes, alcançar um consenso entre as partes interessadas sobre esses requisitos, documentá-los de acordo com padrões determinados e gerenciá-los sistematicamente. 2. Entender e documentar os desejos e necessidades das partes interessadas, especificando e gerenciando os requisitos para minimizar o risco de entregar um sistema que não atenda aos desejos e necessidades das partes interessadas” (POHL e RUPP, 2011).
d) “A **engenharia de requisitos** consiste em atividades relacionadas à descoberta, documentação e manutenção de um conjunto de requisitos para sistemas baseados em computadores, através do uso de técnicas sistemáticas e repetíveis para garantir que os requisitos do sistema sejam completos, consistentes e relevantes” (KOTONYA e SOMMERVILLE, 1998).

Algumas iniciativas de pesquisadores (PENZENSTADLER et al. 2013) propuseram a criação de um REBoK (*Requirements Engineering Body of Knowledge*), a exemplo dos já citados SWEBOK e SEBOK. Até o presente momento, não há um documento que esteja finalizado, padronizado e reconhecido pela maioria da comunidade de RE (e.g.: reconhecido pelo IREB), e que possa fornecer definições adicionais para os termos de RE. Há um documento REBoK em desenvolvimento pelo REQB (*Requirements Engineering Qualification Board)*, que é uma entidade em processo de integração com a IREB (IREB FAQ, 2018).

12. RE: requisito

a) “**Requisito** é uma declaração que traduz ou expressa uma necessidade e suas restrições e condições associadas” (ISO/IEC/IEEE 29148, 2011).
b) “**Requisito** é: 1. condição ou capacidade necessária a um usuário para resolver um problema ou atingir um objetivo. 2. uma condição ou capacidade a ser atingida ou possuída por um sistema, componente de sistema, produto ou serviço para satisfazer um acordo, padrão, especificação ou outro documento formalmente imposto. 3. uma representação documental das condições e capacidades citadas anteriormente (1 e 2). 4. uma condição ou capacidade que deve ser atingida ou possuída por um sistema, produto, serviço, resultado ou componente para satisfazer um contrato, padrão, especificação ou outro documento formalmente imposto” (ISO/IEC/IEEE 24765, 2010). A definição acima também é adotada pelo IREB (POHL e RUPP, 2011) e pelo guia do corpo de conhecimento em análise de negócios BABOK (*Business Analysis Body of Knowledge*) (IIBA, 2015).
c) “**Requisitos** são definidos durante os estágios iniciais do desenvolvimento de um sistema como uma especificação do que deve ser implementado. Eles são descrições de como o sistema deve se comportar, informações do domínio da aplicação, restrições na operação do sistema ou especificação de uma propriedade ou atributo do sistema. Às vezes, eles são restrições no processo de desenvolvimento do sistema” (KOTONYA e SOMMERVILLE, 1998).

13. RE: requisito funcional

a) “**Requisito funcional** é uma declaração que identifica o que um produto ou processo deve realizar para produzir o comportamento e/ou resultados necessários” (ISO / IEC / IEEE 24765, 2010).
b) “**Requisito funcional** é um requisito que especifica uma função que um sistema ou componente do sistema deve ser capaz de executar” (ISO / IEC / IEEE 24765, 2010).
c) “**Requisitos funcionais** são sentenças sobre serviços que o sistema deve fornecer, como o sistema deve reagir a determinadas entradas e

como o sistema deve se comportar em situações particulares. Em alguns casos, os requisitos funcionais também devem determinar explicitamente o que o sistema não deve fazer" (SOMMERVILLE, 2010).

14. RE: requisito não-funcional

a) "**Requisito não-funcional** é um requisito de software que descreve não o que o software fará, mas como o software o fará". Sinônimo: restrição de projeto (ISO / IEC / IEEE 24765, 2010).
b) "**Requisitos não-funcionais** são restrições nos serviços ou funções oferecidas pelo sistema. Elas incluem restrições temporais, restrições no processo de desenvolvimento e restrições impostas por normas (padrões). Requisitos não funcionais tipicamente se aplicam ao sistema como um todo, ao invés de características ou serviços individuais" (SOMMERVILLE, 2010).

15. RE: parte interessada - *stakeholder*

a) "**Parte interessada** (*stakeholder*) é um indivíduo ou organização que tem direito, ação, reivindicação ou interesse em um sistema ou em sua posse de características que atendem às suas necessidades e expectativas" (ISO/IEC/IEEE 24765, 2010) (ISO/IEC/IEEE 15288, 2008) (ISO/IEC/IEEE 29148, 2011).
b) "**Parte interessada** (*stakeholder*) é uma pessoa ou organização (por exemplo, cliente, patrocinador, organização executora ou o público) que esteja ativamente envolvida no projeto ou cujos interesses possam ser positiva ou negativamente afetados pela execução ou conclusão do projeto. Uma parte interessada também pode exercer influência sobre o projeto e suas entregas (ISO/IEC/IEEE 24765, 2010).

16. RE: interdependência, dependências e relacionamentos entre requisitos

Tomando-se como base o conceito de "relacionamento" definido no padrão ISO/IEC/IEEE como sendo "uma conexão semântica entre dois elementos de um modelo" ou "uma associação de interesse entre duas entidades" (ISO/IEC/IEEE 24765, 2010), é possível então conceitualizar "dependência" entre requisitos a partir do conceito de relacionamento.

a) “A **dependência** entre requisitos é um relacionamento entre requisitos...” (ZHANG et al., 2014).
b) “Os relacionamentos podem ser definidos como conexões, associações ou **dependências** entre objetos de informações, por exemplo, é-a (*is-a*), refina (*refines*), baseia-se (*based-on*) e descreve (*describes)*. O termo **dependência** é usado de maneiras bem diferentes por diferentes autores. Isso implica que a **dependência** pode ser vista como outra palavra para relacionamento ou associação, ou como uma conexão mais forte entre dois objetos, onde os objetos afetam um ao outro, por exemplo, em caso de mudanças” (DAHLSTEDT, 2001).

Neste trabalho adota-se a visão, de que uma dependência entre requisitos é um “relacionamento forte” entre os mesmos, na medida em que entidades dependentes afetam uma a outra (DAHLSTEDT, 2001).

O termo escolhido neste trabalho para representar relacionamentos entre requisitos será o de “**interdependência**”. Ainda assim, considerar-se-ão intercambiáveis os seguintes termos relativos a relacionamentos entre requisitos: “relacionamento”, “inter-relacionamento”, “dependência”, “interdependência”, “ligação” e “interligação”.

17. RE: modelagem gráfica de requisitos

a) “A **modelagem gráfica de requisitos** (*graphical requirements modeling* - GRM) se refere a prática bem difundida na área de RE de representar requisitos e seus inter-relacionamentos sob o formato de diagramas visuais e de racionalizar sobre estas representações. Tipicamente, nodos nestes diagramas contém informações sobre requisitos, ambiente ou sistema a construir ou modificar, enquanto as ligações são nomeadas por relacionamentos sobre os nodos conectados.” (GILLAIN et al., 2016).

18. RE: diagrama ou representação diagramática

a) “Uma estrutura de dados na qual a informação é indexada por uma localização bidimensional é o que chamamos de **representação diagramática.”** (LARKIN e SIMON, 1987).

b) “Um **diagrama** é uma representação visual que compartilha as propriedades de texto escrito e imagens representacionais, mas não pode ser reduzida a nenhuma delas” (BLACKWELL, 2001).

19. RE: notação visual, linguagem visual, notação gráfica

a) “Uma **notação visual** (ou **linguagem visual, notação gráfica, notação de diagramação**) consiste em um conjunto de símbolos gráficos (vocabulário visual), um conjunto de regras de composição (gramática visual) e definições sobre o significado de cada símbolo (semântica visual) (MOODY, 2009).

20. RE: linguagem de especificação de requisitos

a) “**Linguagem de especificação de requisitos** é uma linguagem de especificação que possui construções especiais e protocolos de verificação – em alguns casos – utilizados para desenvolver, analisar e documentar requisitos de *software* ou *hardware*” (ISO/IEC/IEEE 24765, 2010).
b) “**Linguagem de especificação** é uma combinação processável por máquina de linguagem natural e formal, usada para expressar **requisitos**, projeto, comportamento ou outras características de sistemas ou componentes” (ISO/IEC/IEEE 24765, 2010).

21. RE: ferramenta de modelagem

a) “**Ferramenta de modelagem** é uma ferramenta que provê suporte para a modelagem (i.e., representação) de um produto de *software* ou sistema de informação” (ISO/IEC/IEEE 24765, 2010).

22. RE: metodologia e método

a) “**Metodologia** é a especificação do processo a seguir juntamente com os produtos de trabalho a serem utilizados e gerados, além da consideração das pessoas e ferramentas envolvidas, durante um esforço de desenvolvimento de um domínio baseado em informações” (ISO/IEC 24744, 2007).
b) “**Método** é um sinônimo para metodologia” (ISO/IEC 24744, 2007).

23. RE: norma internacional - *international standards*

a) “**Norma internacional** é um documento, estabelecido por consenso e aprovado por um órgão reconhecido, que fornece, para uso comum e repetido, regras, diretrizes ou características para atividades ou seus resultados, visando a obtenção do grau ótimo de ordem em um determinado contexto (ISO/IEC Guide 2, 2004).

24. RE: metamodelo e ontologia

a) “**Metamodelo** é um modelo de informações lógicas que especifica os elementos de modelagem usados em outra (ou a mesma) notação de modelagem” (ISO/IEC/IEEE 24765, 2010).
b) “**Metamodelo** é a especificação de conceitos, relacionamentos e regras que são usados para definir uma metodologia” (ISO/IEC 24744, 2007).
c) “**Ontologia** é uma estrutura lógica dos termos usados para descrever um domínio de conhecimento, incluindo as definições dos termos aplicáveis e seus relacionamentos” (ISO/IEC/IEEE 24765, 2010).

REFERÊNCIAS DO APÊNDICE D

BLACKWELL, Alan F. **Thinking with Diagrams**. Artificial Intelligence Review, v. 15, n. 1–2, p. 77–78, 2001.

BOURQUE, P.; FAIRLEY, R. E. (eds.) **Guide to the Software Engineering Body of Knowledge**, Version 3.0. IEEE. Piscataway, NJ, 2014.

DAHLSTEDT, Åsa G. **Requirements Interdependencies – a Research Framework**. Department of Computer Science, University of Skövde, Sweden 2001.

FRIEDENTHAL S., MOORE A., STEINER R. **A Practical Guide to SysML: The Systems Modeling Language**. Morgan Kaufmann / OMG Press, 3rd Ed., 2014.

GILLAIN, J., BURNAY, C., JURETA, I., FAULKNER, S. **AnalyticGraph.com: Toward Next Generation Requirements Modeling and Reasoning Tools**. Proceedings of 24th IEEE International Requirements Engineering Conference, 341-346, IEEE Computer Society, 2016.

IIBA, **A Guide to the business analysis body of knowledge (BABOK Guide)**. International Institute of Business Analysis (IIBA). 3ª Ed., Toronto, Ontario, 2015.

INCOSE. **Systems Engineering Handbook. A guide for system life cycle processes and activities**. Prepared by INCOSE. 4th Ed.; Hoboken, NJ.; John Wiley & Sons, 2015.

IREB FAQ. **“What does it mean: IREB and REQB join forces?”** Disponível em <https://www.ireb.org/en/faqs/tag:reqb>. Acesso em 08/2018.

ISO/IEC. **ISO/IEC Guide 2: 2004 - Standardization and related activities - General vocabulary**. International Organization for Standardization, 2004.

ISO/IEC. **Software Engineering Metamodel for Development Methodologies (ISO/IEC 24744:2007)**. International Organization for Standardization, 2007.

ISO/IEC/IEEE. **Systems and software engineering - System life cycle processes ISO/IEC/IEEE 15288:2008.** The Institute of Electrical and Electronics Engineers. Piscataway, NJ. 2008.

ISO/IEC/IEEE. **Systems and software engineering - Life cycle processes - Requirements engineering - ISO/IEC/IEEE 29148:2011**. The Institute of Electrical and Electronics Engineers. Piscataway, NJ. 2011.

ISO/IEC/IEEE. **Systems and Software Engineering - Vocabulary ISO/IEC/IEEE 24765: 2010**. The Institute of Electrical and Electronics Engineers. Piscataway, NJ. 2010.

KOTONYA, Gerald; SOMMERVILLE, Ian. **Requirements Engineering : Processes and Techniques.** Worldwide Series in Computer Science. 1st. ed. Chichester, England: John Wiley & Sons, Inc., 1998.

LARKIN, Jill H.; SIMON, Herbert A. **Why a Diagram is (Sometimes) Worth Ten Thousand Words**. Cognitive Science, v. 11, n. 1, p. 65–99, 1987.

MOODY, Daniel L. **The "Physics" of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering**. IEEE Transactions on Software Engineering, v. 35, n. 6, p. 756–779, 2009.

OMG, Object Management Group. **Systems Modeling Language (SysML)**, Version 1.5. SysML formal/17-05-01. 2017.

PENZENSTADLER, B.; FERNÁNDEZ, D. M.; RICHARDSON, D.; CALLELE, D.; WNUK, K. **The requirements engineering body of knowledge (REBoK)**. In: 21st IEEE International Requirements Engineering Conference (RE), p. 377–379, 2013.

POHL, Klaus; RUPP, Chris. **Requirements Engineering Fundamentals: A Study Guide for the Certified Professional for Requirements Engineering Exam-Foundation Level**. 1st. ed. Santa Barbara, USA. Rocky Nook Inc., 2011.

SOMMERVILLE, I. **Software Engineering**. 9. ed. Boston: Pearson, 2010.

WASSON, Charles S. **Systems Engineering Analysis, Design and Development**. 2nd. ed. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2005.

ZHANG, He; LI, Juan; ZHU, Liming; JEFFERY, Ross; LIU, Yan; WANG, Qing; LI, Mingshu. **Investigating dependencies in software requirements for change propagation analysis**. Information and Software Technology, v. 56, n. 1, p. 40–53, 2014.

APÊNDICE E – ARTIGO EM CONFERÊNCIA: REQUISITO DE MESTRADO

NOVAES, Paulo J. D.; SIMÃO, Jean M.; STADZISZ, Paulo C. **Integration between Requirements Modeling and Software Development in the Notification Oriented Paradigm: A Security System Case Study**. *In*: IX Computer on the Beach (COTB 2018). p. 432–441 Florianópolis, Brasil, 2018.

Integration between Requirements Modeling and Software Development in the Notification Oriented Paradigm: A Security System Case Study

Paulo J. D. Novaes[1], Jean M. Simão[1], Paulo C. Stadzisz[1]

[1]Graduation Program in Electrical Engineering and Computer Engineering (CPGEI)
Federal University of Technology - Paraná, (UTFPR), Curitiba - PR - Brasil

pnovaes@alunos.utfpr.edu.br, {jeansimao,stadzisz}@utfpr.edu.br

Abstract. *This paper presents the integration between the requirements modeling approach named Notification Oriented Requirements (NOR) and the Software Development method known as Notification Oriented Development (NOD). This integration is demonstrated by means of the case study of a simulated access control security system implemented in the Notification Oriented Paradigm (NOP). Results show that the integration between NOR model and NOD method is possible and facilitates the development of NOP software, since NOR clarifies the necessary elements to perform the software structural modeling (class model) and the behavioral modeling (high level states model and component model).*

1. Introduction

Requirements Engineering (RE) refers to the activity of formulating, documenting, and maintaining systems requirements in order to produce, from users' needs, a set of specification related to what the final system should be [YOUNG, 2004]. A requirement is a statement from the stakeholders' needs to define a product, a system or a process, and must be unambiguous, clear, unique, consistent, stand-alone, and verifiable [INCOSE, 2006]. Graphical approaches to enhance requirements specification (among others system´s characteristics) have gained prevalence, such as the SysML language, which is used in Model-Based Systems Engineering (MBSE) [FRIEDENTHAL et al., 2014].

In this context, Notification Oriented Requirements (NOR) emerges as a requirement modeling approach originated from concepts of the Notification Oriented Paradigm (NOP) and MBSE [SIMÃO et al., 2016]. In brief, NOP is an alternative paradigm using rules and notifications for composing software and hardware systems. Within this paradigm, NOR is a requirements specification approach applicable to both, software and system development processes. The practical integration between NOR and software development processes is an important experimentation for its validation. Currently, the Notification Oriented Development (NOD) method [MENDONÇA et al., 2015] is suited to develop NOP software. Thus, the following questions arise:

- Is it possible to integrate the NOR modeling approach into the NOD method?

- Are there advantages in integrating NOR and NOD into a NOP application project? Which are they?

To answer these questions, this study starts from a previous NOR model [SIMÃO et al., 2016], uses the NOD method to design and implement the corresponding NOP application, and concludes discussing the findings from the case study.

Therefore, the objectives of this study are to integrate NOR modeling approach into the NOD method and to identify the advantages of such integration during NOP application software development.

2. Notification Oriented Paradigm (NOP)

NOP has been improved in recent years by a group of researchers from the Federal University of Technology - Paraná (UTFPR). It is an alternative approach to develop software and hardware systems. NOP has several implementation versions in the form of frameworks and languages [FERREIRA, 2015]. It proposes to solve existing problems in usual programming paradigms [SIMÃO and STADZISZ, 2008] such as the Declarative Paradigm (PD) and the Imperative Paradigm (PI) [GABBRIELLI and MARTINI, 2010]. These problems are related to structural and temporal redundancies, and the strong coupling between computational entities [FERREIRA, 2015].

The fundamental proposal of NOP consists in the introduction of a notification-based inference mechanism, presenting a new way of structuring software in small and decoupled computational entities. These entities include *Fact Base Elements* (*FBE*) and Rules [SIMÃO and STADZISZ, 2008]. An example of a NOP *Rule* is shown in figure 1 (a) and the NOP model is shown in figure 1 (b).

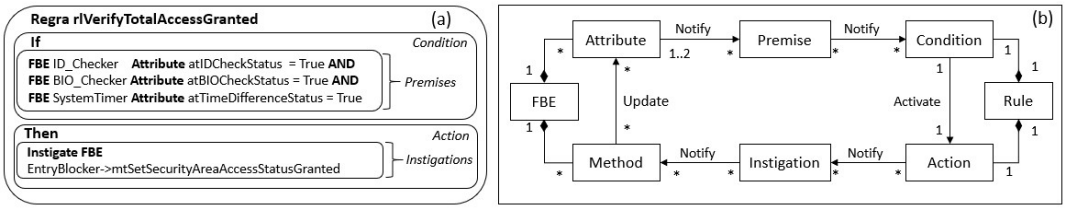


Figure 1. (a) NOP *Rule*. (b) NOP Model. Based on [SIMÃO et al., 2016].

The *Fact Base Element* (*FBE*) stores system facts in *Attributes*. The *FBE* may have *Methods* that modify these *Attributes*. Each *Attribute* when it changes its status notifies only the related *Premises*. Similarly, each *Premise* when it changes its status notifies only the pertinent *Conditions*. Each *Condition* has one or more associated *Premises*, that becoming true, approve a *Rule*. The *Rule* has an *Action*, which notifies one or more *Instigations* to execute *Methods*, which in turn modify other *Attributes* [MENDONÇA et al., 2015]. This sequence characterizes the NOP inference mechanism, based on notifications. This mechanism avoids the need for matching and selection processes in order to execute rules, as usual in Rules Based Systems (RBS).

Many NOP applications have been developed [SANTOS, 2017]. However, NOP application development requires new specific software development methods. For this purpose, techniques such as NOD [WIECHETECK, 2011] and NOR modeling [SIMÃO et al., 2016] have been developed, which will be presented in the next sections.

3. Notification Oriented Requirements (NOR)

In NOR approach, the fundamental primitives of NOP models are used for graphical requirements modeling, such as *Rules*, *FBEs,* and *Notifications* (preconditions or postconditions). These primitives allow to describe the complete set of functional e non-functional software/system requirements [SIMÃO et al., 2016]. The NOR modeling notation is summarized in figure 2.

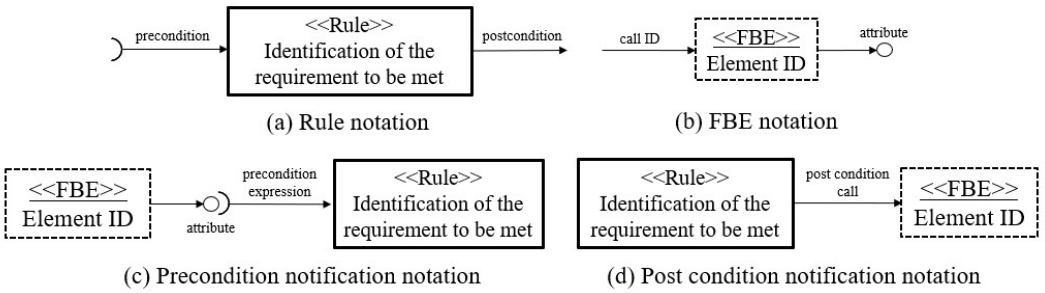


Figure 2. NOR requirements modeling notation [SIMÃO et al., 2016].

The NOR technique to construct the requirements model is [SIMÃO et al., 2016]:

For each requirement in the System Requirements Specification (SRS):

1. *To analyze the requirements sentence aiming at:*
   i) Identifying the functional or non-functional request in the requirement.
   ii) Identifying the Conditions for the functional or non-functional request.
   iii) Identifying the attributes involved in the Conditions.
   iv) Identifying the Actions for the functional or non-functional request.
   v) Identifying the functions related to the Methods instigated in the Actions.
   vi) Identifying the FBEs related to the Attributes for the request.
   vii) Identifying the FBEs related to the Methods indicated by the request.
2. *To create a Rule for every request identified in step 1.*
3. *To create a FBE for every entity identified in step 1.*
4. *To create links (i.e. notifications between Rules and FBEs according to conditions and Actions related to rules) identified in step 1.*
5. *To merge FBEs and Rules with analogous FBEs and Rules previously created.*

The use of NOR modeling was presented in a case study [SIMÃO et al., 2016] whose requirements were extracted from the INCOSE Systems Engineering Handbook [INCOSE, 2006]. Six requirements for the given security area access control system are presented below [SIMÃO et al., 2016]:

- *SS11-a: Secure areas shall be protected by security check based upon employee ID.*
- *SS11-b: Secure areas shall be protected by a second independent security check based upon biometric data.*
- *SS11-c: The time between the two independent security checks shall not exceed a configurable period.*
- *SS11-d: The user shall be allowed three attempts at biometric identification.*
- *SS11-e: The user shall be allowed three attempts at card identification.*
- *SS11-f: Any denied access attempt shall be sent to the administrator.*

Based on the NOR technique above, a model for the given security system was created (figure 3) [SIMÃO et al., 2016] containing 7 requirements in the form of *Rules* and 7 entities in the form of *FBEs*.

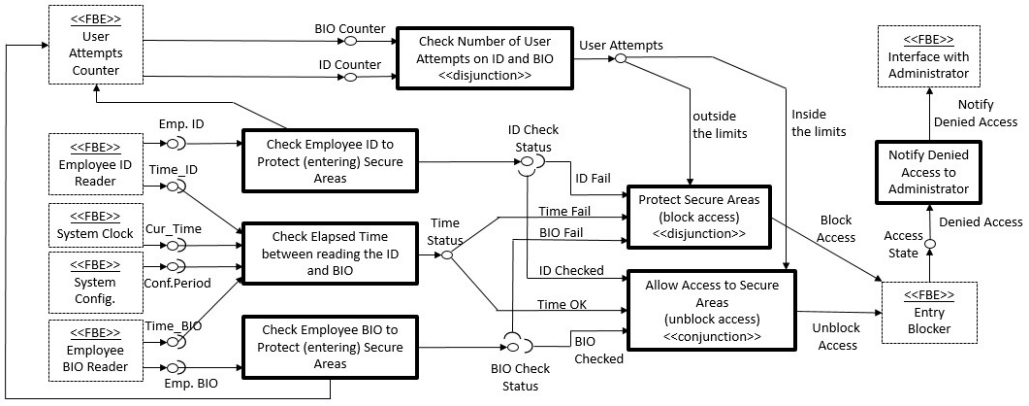


Figure 3. NOR requirements final modeling. Adjusted from [SIMÃO et al., 2016]

The model above facilitates requirements analysis and implicit knowledge identification, in addition to making explicit the dependencies between requirements [SIMÃO et al., 2016]. The next section presents the NOD method propped to conduct the development of NOP software applications.

4. Notification Oriented Development (NOD)

NOD is a software development method developed specifically for NOP applications [WIECHETECK, 2011], which consists of an extension of UML diagrams in the form of a UML profile, that properly represents NOP concepts (*NOP profile*). In addition, NOD establishes a sequence of steps to guide NOP software development.

The *NOP profile* enables to characterize NOP elements more precisely during design phase, allowing to particularize UML for a specific domain of applications. This is done by determining a new syntax and semantics for UML elements using stereotypes, tagged values, and constraints [WIECHETECK et al., 2011].

NOD method contains 8 steps (figure 4). The first two steps are: 1. *Capture Requirements* and 2. *Create Use Case Model*. The next six steps focus on software design through diagrams creation: 3. *Class Model*; 4. *High Level States Model*; 5. *Component Model*; 6. *Sequence Model* (optional, not created in this study[1]); 7. *Communication Model* (optional, not created in this study[1]); 8. *Petri Net Model* [WIECHETECK, 2011].

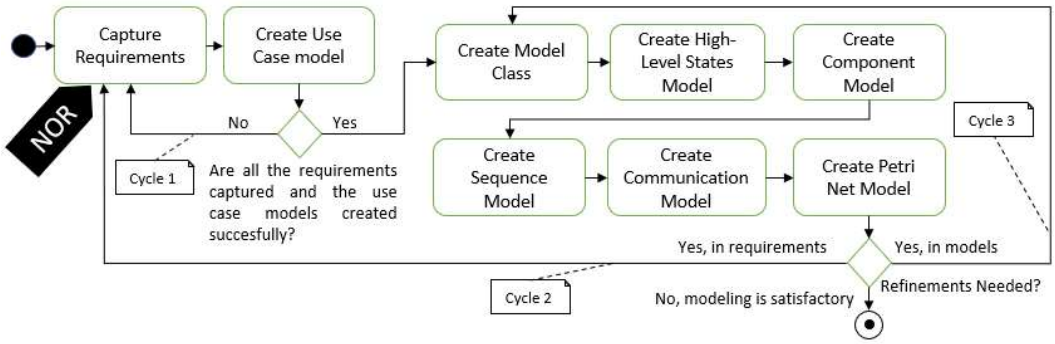


Figure 4. Integrating NOR into NOD. Based on [MENDONÇA et al., 2015].

[1] The creation of the *Sequence Model* and *Communication model* is an optional part of DON method and was not performed in the current study, without compromising the article results.

The method is divided in three cycles (figure 4). In **first cycle**, the requirements are documented, and the *Use Case Model* is created. In **second cycle**, diagrams are created, and requirements are refined as necessary. In **third cycle**, the models are refined to ensure compliance with the requirements [MENDONÇA et al., 2015] [WIECHETECK, 2011].

Software coding can either occur at the end of the method (cascade software process) or during cycles (incremental software process). The incremental software process was used in this case study.

5. Case Study: NOD Modeling of a Simulated Security System

The requirements and NOR modeling described previously were used as the basic scope for this case study. This is the main point of integration between the methods, in which the **NOD 1st Cycle => 1st step** *Capture Requirements* of the NOD method is now performed by the NOR technique, as pointed out in figure 4.

The models for this study were created using tools of the *Enterprise Architect® v.13.5* (*Sparx Systems*) suite by applying the *NOP Profile* [WIECHETECK et al., 2011], except for the Petri Net model, created using *CPN Tools® v.4.0.1* (*Eindhoven University of Technology*).

Given this, based on NOR modeling, the **NOD 1st Cycle => 2nd step** *Create Use Case Model* was performed and resulted in the model shown in figure 5.

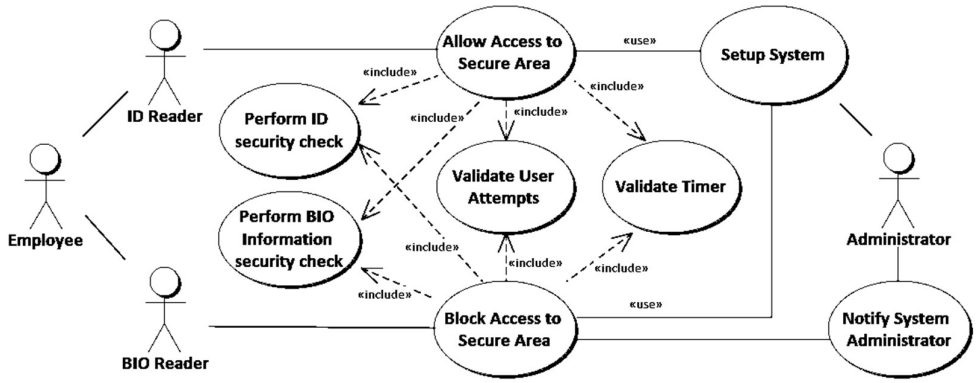


Figure 5. Use Case Model (NOD)

In **NOD 2nd cycle** modeling, the following diagrams were created, later refined in the **NOD 3rd cycle** (figure 4): 1. *Class Model* (*FBEs* definition); 2. *High Level States Model* (*Rules modeling*); 3. *Component Model*; 4. *Petri Net Model*.

The *Class Model* is presented in figure 6. In addition to the stereotyped class *<<NOP_Application>>*, that is a default in NOP applications, stereotyped classes *<<NOP_FBE>>* were created for each *FBE*. In this step, it is possible to notice the easiness achieved by the previous existence of the NOR model, **since it becomes possible to correlate the *FBEs*** (this does not imply necessarily a 1 to 1 relationship) modeled in NOR to those included in the *Class Model*:

- *Employee ID Reader* (NOR) ⇔ *ID_Checker* (NOD)
- *Employee BIO Reader* (NOR) ⇔ *BIO_Checker* (NOD)
- *System Clock* (NOR) ⇔ *SystemTimer* (NOD)
- *User Attempts Counter* (NOR) ⇔ *UserAttemptsCounter* (NOD)

- *Entry Blocker* (NOR) ⇔ *EntryBlocker* (NOD)
- *System Config.* (NOR) ⇔ *SystemConfigurator* and *EmployeeController* (NOD)
- *Interface with Administrator* (NOR) ⇔ *SysAdminNotificationController* (NOD)

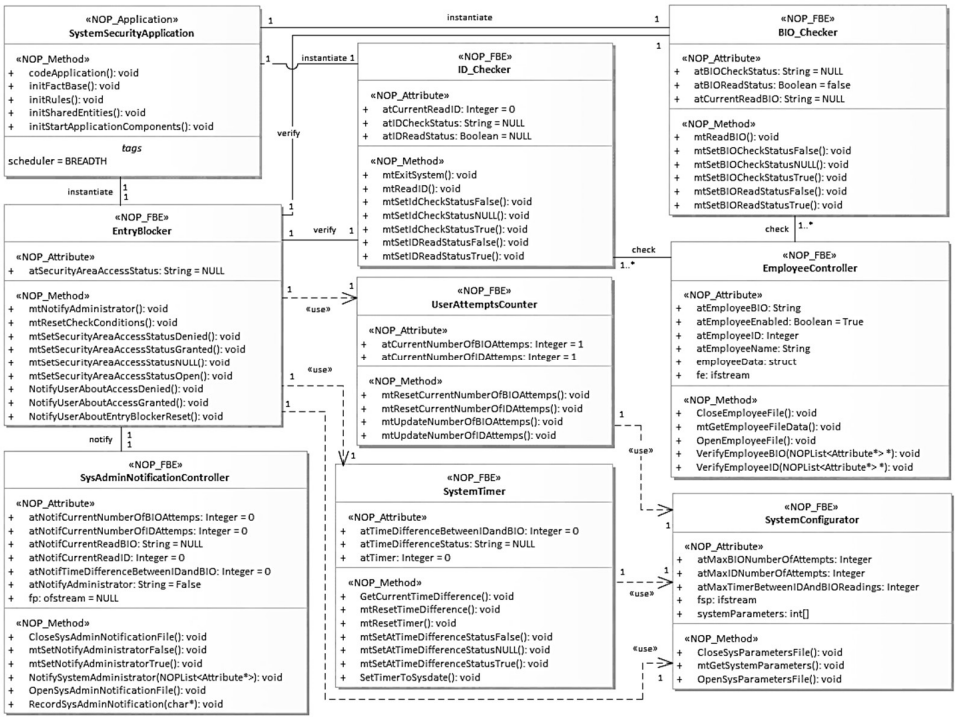


Figure 6. Class Model (NOD)

The *High-Level States Model* establishes the basic logic of system operation and bases the identification of NOP Application *Rules* (figure 7).

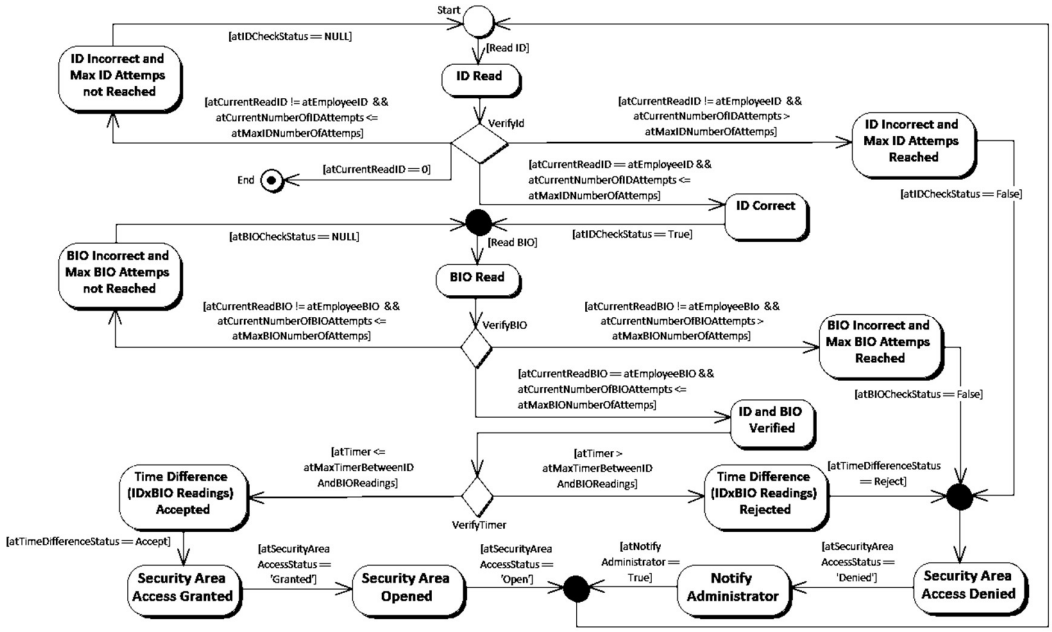


Figure 7. High-Level States Model (NOD)

While creating the *High-Level States Model*, **the NOR model was used as an aid**, **allowing to visually identify states** or **facts** (e.g.: *ID Checked, ID Fail, Time OK, Time Fail, BIO Checked, BIO Fail etc*.) and **activities** (e.g.: *Check employee ID, Check Elapsed Time, Check Employee BIO, Block Access, Unblock Access etc*.).

Based on *Class* and *High-Level States* models, a ***Rules*** **Table** containing the *Rules, Premises, and Instigations* was created (table 1). NOP elements were named using a standard that facilitates identification during both system design and programming. Prefixes were used as follows: *rl* for *Rules*; *pr* for *Premises*; *in* for *Instigations*; *at* for *Attributes* and *mt* for *Methods*, following definitions given by [RONSZCKA et al., 2017].

Table 1. NOP *Rules* identified for the System (NOD)

N	Use Cases	Rules	Premises	Instigations
1	Perform ID Security Check	rlReadID	prAtIDReadStatusFalse	inReadID && inVerifyEmployeeID && inSetAtIDReadStatusTrue
2	Perform ID Security Check	rlVerifyIDIncorrectRetry	prAtIDCheckStatusNULL && prAtIDReadStatusTrue && prIDIncorrect && prAtCurrentNumIDAttemptsSmallerThanMax	inIncrementAtCurrentNumberOfIDAttempts && inSetAtIDCheckStatusNULL && inResetTimer &&
3	Perform ID Security Check	rlVerifyIDIncorrectDenyAcc	prAtIDCheckStatusNULL && prAtIDReadStatusTrue && prIDIncorrect &&	inSetAtIDCheckStatusFalse && inResetTimer
4	Perform ID Security Check	rlVerifyIDCorrectProceed	prAtIDCheckStatusNULL && prAtIDReadStatusTrue && prIDCorrect &&	inSetAtIDCheckStatusTrue && inSetAtTimerToSysdate
5	Perform BIO Security Check	rlReadBIO	prAtIDCheckStatusTrue && prAtBIOReadStatusFalse	inReadBIO && inVerifyEmployeeBIO && inSetAtBIOReadStatusTrue
6	Perform BIO Security Check	rlVerifyBIOIncorrectRetry	prAtBIOCheckStatusNULL && prAtBIOReadStatusTrue && prAtIDCheckStatusTrue && prBIOIncorrect && prAtCurrentNumBIOAttemptsSmallerThanMax	inIncrementAtCurrentNumberOfBIOAttempts && inSetAtBIOCheckStatusNULL &&
7	Perform BIO Security Check	rlVerifyBIOIncorrectDenyAc	prAtBIOCheckStatusNULL && prAtBIOReadStatusTrue && prAtIDCheckStatusTrue && prBIOIncorrect && prAtCurrentNumBIOAttemptsGreaterOrEqualThanMax	inSetAtBIOCheckStatusFalse && inResetTimer
8	Perform BIO Security Check	rlVerifyBIOCorrectProceed	prAtBIOCheckStatusNULL && prAtBIOReadStatusTrue && prAtIDCheckStatusTrue && prBIOCorrect && prAtCurrentNumBIOAttemptsSmallerOrEqualThanMax	inSetAtBIOCheckStatusTrue && inGetTimeDifference
9	Perform Timer Validation	rlVerifyTimerAccept	prAtIDCheckStatusTrue && prAtBIOCheckStatusTrue && prAtTimeDifferenceSmallerOrEqualThanMax	inSetAtTimeDifferenceStatusTrue
10	Perform Timer Validation	rlVerifyTimerReject	prAtIDCheckStatusTrue && prAtBIOCheckStatusTrue && prAtTimeDifferenceGreaterThanMax	inSetAtTimeDifferenceStatusFalse
11	Perform Entry Blocker Security	rlVerifyTotalAccessGranted	prAtIDCheckStatusTrue && prAtBIOCheckStatusTrue && prAtTimeDifferenceStatusTrue	inSetAtSecurityAreaAccessStatusGranted
12	Perform Entry Blocker Security	rlVerifyTotalAccessDenied	prAtIDCheckStatusFalse || prAtBIOCheckStatusFalse || prAtTimeDifferenceStatusFalse	inSetAtSecurityAreaAccessStatusDenied
13	Allow Access to Secure Area	rlOpenSecurityArea	prAtSecurityAreaAccessStatusGranted	inSetAtSecurityAreaAccessStatusOpen &&
14	Allow Access to Secure Area	rlSecurityAreaOpened	prAtSecurityAreaAccessStatusOpen	inNotifyUserAboutEntryBlockerReset && INSTIGATIONS TO RESET ALL
15	Block Access to Secure Area	rlBlockSecurityArea	prAtSecurityAreaAccessStatusDenied	inSetAtNotifyAdministratorTrue && inNotifyUserAboutAccessDenied
16	Notify System Administrator	rlAdministratorNotified	prAtNotifyAdministratorTrue	inSetAtNotifyAdministratorFalse && inNotifySystemAdministrator && inNotifyUserAboutEntryBlockerReset && INSTIGATIONS TO RESET ALL
17	Exit System	rlExitSystem	prAtIDCheckStatusNULL && prAtIDReadStatusTrue && prExitSystem	inExitSystem

In parallel to the elaboration of the ***Rules*** **Table**, the *Component Model* was generated in a creative synthesis activity subdivided in three steps: 1. *Define Rules*; 2. *Define Premises and Instigations*; 3. *Associate Rules to FBEs* [WIECHETECK, 2011].

The existence of NOR modeling facilitated the *Rules* definition and their interdependencies. For example, the requirement modeled in NOR *"Protect Secure Areas"* (figure 3) which is a disjunction (<<*disjunction*>>) between "*ID Fail*", "*Time Fail*", and "*BIO Fail*", was modeled by the NOP *Rule rlVerifyTotalAccessDenied*, as a disjunction between the equivalent corresponding premises *prAtIDCheckStatusFalse*, *prAtBIOCheckStatusFalse,* and *prAtTimeDifferenceStatusFalse* (table 1).

Seventeen (17) *Component Models* were created (one for each *Rule*). The model for the *Rule rlReadID* is illustrated as an example in the figure 8. It is possible to notice the behavior of the *Rule* (*rlReadID*), its *Premises* (*prAtIDReadStatusFalse*), its *Instigations* (*inVerifyEmployeeID*, *inReadID inSetAtIDReadStatusTrue*), and the methods (*mtVerifyEmployeeID*, *mtReadID*, *mtSetIDReadStatusTrue*) that may be triggered in the *FBEs* (*EmployeeController*, *ID_Checker*). Besides that, it is possible to

observe the *Attributes* used by the *Rule* (*atIDReadStatus*, *atCurrentReadID, AtEmployeeID*).

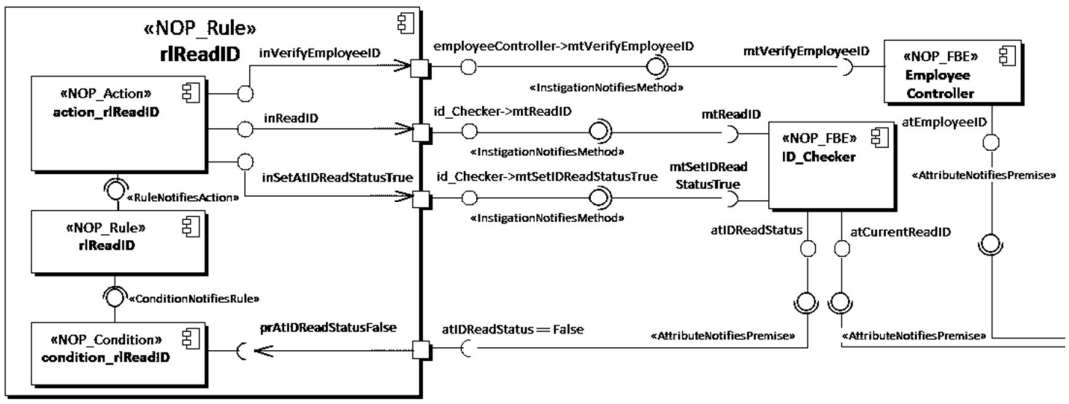


Figure 8. *Rule rlReadID* from Component Model (NOD)

The *Petri Net Model* used in NOD method demonstrates the dynamics between NOP elements. Petri Nets (PN) allows to model, simulate and even verify concurrency and synchronization of resources in systems [CARDOSO and VALETTE, 1997].

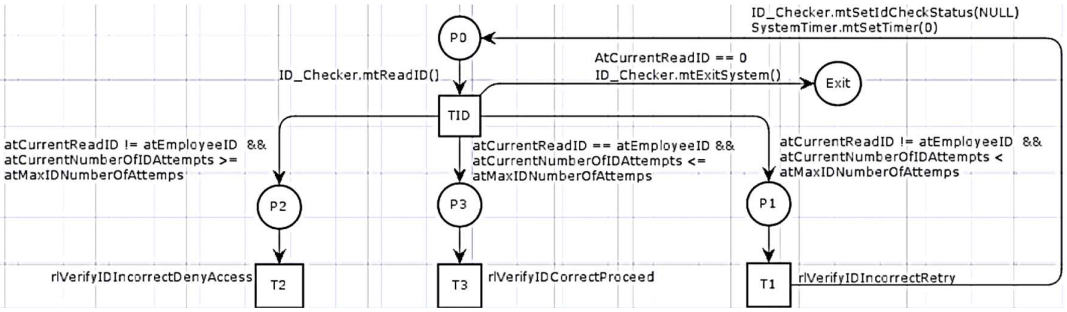


Figure 9. Part of the modeled Petri Net Model (NOD)

In figure 9, part of the **Petri Net** modeled for this study is shown, in which the NOP *Rules* were mapped as PN **transitions:** *rlReadID* (TID), *rlVerifyIDIncorrectDenyAccess* (T2), *rlVerifyIDCorrectProceed* (T3) and *rlVerifyIDIncorrectRetry* (T1). The PN **places** (P1, P2, P3 etc.) are representing the *Premises* of each *Rule*. In this model it is possible to notice the concurrency between the *Rules* (T1, T2, T3), which is a characteristic of several NOP applications. In the current study, the *Petri Net Model* was not executed for validation, which is a possibility for future works.

6. NOP Framework C++ 2.0 Implementation

The system implementation was performed in *Visual Studio 2017®* (*Microsoft*) tool, according to established development standards for *NOP Framework C++ 2.0* [RONSZCKA et al., 2017].

Figure 10 (a) shows the NOP C++ codes for the *Rules rlVerifyTotalAccessGranted* and *rlVerifyTotalAccessDenied*, in which is possible to observe the *Premises* and *Instigations* of each Rule. The *Rule rlVerifyTotalAccessGranted* is the same as previously shown in figure 1 and table 1.

Figure 10 (b) shows an execution *prompt command* regarding the NOP application, in which is possible to notice some approved *Rules* (e.g.: *rlReadID,*

rlVerifyIDCorrectProceed, rlReadBIO) that executed the corresponding *Methods* (e.g.: *mtSetIDReadStatusTrue*, *mtSetIDCheckStatusTrue*).

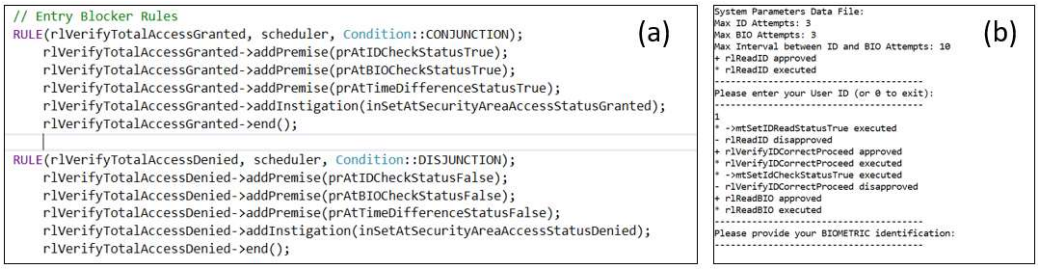


Figure 10. (a) Part of NOP Rules Code (b) NOP Application Execution

Similarly to what was reported in [MENDONÇA et al., 2015] and [WIECHETECK, 2011], it is noted that the creation of a well-structured NOD project allows a near-mechanical implementation in the NOP *C ++ 2.0 Framework*, as may be observed in the corresponding artifacts and codes. Likewise, the NOR project facilitated the project in NOD, due to the visual inputs provided by the requirements model.

7. Discussion and Conclusion

In this case study, the integration between the NOR model into the NOD method was performed during the development of a NOP application. By presenting system requirements graphically, NOR aids to the development of NOP software in three steps:

1. In the **Class Model** creation (system structure).
2. In the **High-Level States Model** creation (system behavior).
3. In the ***Rules*** **Table** and **Component Model** creation (specific tailored behavior of NOP application *Rules*)

Therefore, it can be concluded that the NOR modeling can be harmoniously integrated to the NOD method, facilitating the development of Software Engineering design for NOP applications. This would lead to future works on engineering-oriented requirements models such as SysML. It is also suggested the refinement of NOR modeling techniques through a specific study of interrelationships between requirements.

Acknowledgements

The authors would like to thank CAPES/CNPq from Brazil by the financial support and to UTFPR by all the support and infrastructure provided.

Referências

[CARDOSO and VALETTE, 1997] Janette Cardoso e Robert Valette. *Petri Nets.* Original title: Redes de Petri. Florianópolis, Ed. da UFSC, p. 212, 1997.

[FERREIRA, 2015]. Cleverson A. Ferreira. *Language and Compiler for the Notification Oriented Paradigm*: *Advances and Comparisons*. Original title: Linguagem e Compilador para o Paradigma Orientado a Notificações: Avanços e Comparações. Dissertação de Mestrado. PPGCA/UTFPR, Curitiba, Brasil, 2015.

[FRIEDENTHAL et al., 2014] Sanford Friedenthal, Alan Moore, Rick Steiner. *A Practical Guide to SysML: The Systems Modeling Language.* The Morgan Kaufmann / OMG Press, 3rd Ed., 2014.

[GABBRIELLI and MARTINI, 2010] Maurizio Gabbrielli e Simone Martini. *Programming Languages: Principles and Paradigms. Series: Undergraduate Topics in Computer Science*. 1st Edition, XIX, 440 p., Softcover, 2010.

[INCOSE, 2006] INCOSE *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*. International Council on Systems Engineering, C. Haskins. (Ed.), Version 3, 2006.

[MENDONÇA et al., 2015] Igor T. M. Mendonça, Jean M. Simão, Luciana V. B. Wiecheteck, Paulo C. Stadzisz. *Development Method for Rule-Based Systems using the Notification Oriented Paradigm.* Original title: Método para Desenvolvimento de Sistemas Orientados a Regras utilizando o Paradigma Orientado a Notificações. Cong. Bras. Inteligência Computacional, (12), 1–6., 1 CD–ROM. Curitiba, Brasil, 2015.

[RONSZCKA et al., 2017] Adriano F. Ronszcka, Glauber Z. Valença, Robson R. Linhares, João A. Fabro, Paulo C. Stadzisz, Jean M. Simão *Notification-Oriented Paradigm Framework 2.0: An Implementation Based on Design Patterns. IEEE Latin America Transactions*, Volume 15, Issue 11, pp. 2221-2232, Nov 2017.

[SANTOS, 2017] Leonardo A. Santos. *Language and Compiler for Notification Oriented Paradigm*: *advances in coding and validation for a Robotic Soccer application.* Original title: Linguagem e compilador para o paradigma orientado a notificações: avanços para facilitar a codificação e sua validação em uma aplicação de controle de futebol de robôs. Dissertação de Mestrado, CPGEI/UTFPR. Curitiba, Brasil, 2017.

[SIMÃO and STADZISZ, 2008] Jean M. Simão e Paulo C. Stadzisz. *Notification Oriented Paradigm – A Technique for Notification Oriented Software Composition and Execution.* Original title: Paradigma Orientado a Notificações – Uma Técnica de Composição e Execução de Software Orientado a Notificações. Patente. INPI, 2008.

[SIMÃO et al., 2016] Jean M. Simão, Hervé Panetto, Yongxin Liao, Paulo C. Stadzisz. *A Notification-Oriented Approach for Systems Requirements Engineering*. 23rd IPSE International Conference on Transdisciplinary Engineering, Curitiba, Brazil. IOS Press, 4, pp.229-238, Oct 2016.

[WIECHETECK et al., 2011] Luciana V. B. Wiecheteck, Jean M. Simão, Paulo C. Stadzisz. *A UML Profile for Notification Oriented Paradigm.* Original title: Um Perfil UML para o Paradigma Orientado a Notificações. *Anais do III Congreso Internacional de Computación y Telecomunicaciones*, pp. 1-16., Peru, 2011.

[WIECHETECK, 2011] Luciana V. B. Wiecheteck. *Software Development Method using the Notification Oriented Paradigm.* Original title: Método para Projetos de Software usando o Paradigma Orientado a Notificações. *Master Thesis*, CPGEI/UTFPR. Curitiba, Brasil, 2011.

[YOUNG, 2004] R. R. Young. *The Requirements Engineering Handbook*. 1st Ed., Artech House, Boston, 2004.