PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

Universidade Federal do Rio de Janeiro

**DENILSON DOS SANTOS GUIMARÃES**

# INCONSISTÊNCIAS EM REGRAS DE NEGÓCIO: um método para identificação automatizada usando alloy

**DISSERTAÇÃO DE MESTRADO**

Instituto de Matemática

Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE MATEMÁTICA
INSTITUTO TÉRCIO PACITTI DE APLICAÇÕES E PESQUISAS COMPUTACIONAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

DENILSON DOS SANTOS GUIMARÃES

# INCONSISTÊNCIAS EM REGRAS DE NEGÓCIO: um método para identificação automatizada usando alloy

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática, Instituto de Matemática e Instituto Tércio Pacciti, Universidade Federal do Rio de Janeiro, como requisito parcial à obtenção do título de Mestre em Informática.

Orientador: Prof. Eber Assis Schmitz, Ph.D

Co-orientador: Profa. Priscila Machado Vieira Lima, Ph.D

Rio de Janeiro
2015

G963i

Guimarães, Denilson dos Santos
Inconsistências em regras de negócio: um método para identificação automatizada usando alloy / Denilson dos Santos Guimarães. -- Rio de Janeiro, 2015.
87 f.

Orientador: Eber Assis Schmitz.
Coorientadora: Priscila Machado Vieira Lima.
Dissertação (mestrado) - Universidade Federal do Rio de Janeiro, Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais, Programa de Pós-Graduação em informática, 2015.

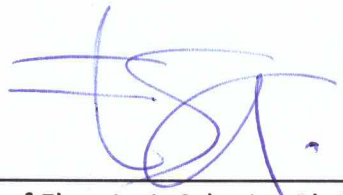
1. Regras de Negócio. 2. Alloy. 3. Inconsistências. I. Schmitz, Eber Assis, orient. II. Lima, Priscila Machado Vieira, coorient. III. Titulo.

DENILSON DOS SANTOS GUIMARÃES

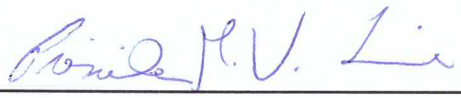
# INCONSISTÊNCIAS EM REGRAS DE NEGÓCIO: UM MÉTODO PARA IDENTIFICAÇÃO AUTOMATIZADA USANDO ALLOY

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática, Instituto de Matemática e Instituto Tércio Pacciti, Universidade Federal do Rio de Janeiro, como requisito parcial à obtenção do título de Mestre em Informática.

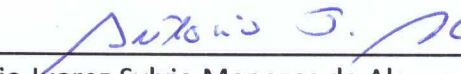
Aprovada em : Rio de Janeiro, _____.de _____ de _____ .



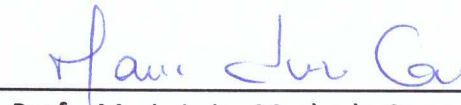
Prof.Eber Assis Schmitz, Ph.D, PPGI/DCC/IM-UFRJ (Orientador)



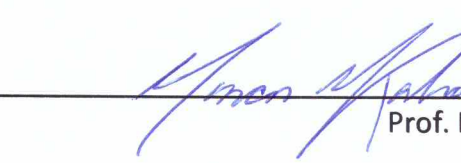
Profa.Priscila Machado Vieira Lima, Ph.D, PPGI/DCC/IM-UFRJ (Co-orientador)



Prof. Antônio Juarez Sylvio Menezes de Alencar, D.Phil., PPGI/DCC/IM-UFRJ



Profa. Maria Luiza Machado Campos, Ph.D, PPGI/DCC/IM-UFRJ



Prof. Marcos Kalinowski, DSc, UFF

Dedicado aos meus queridos pais, que sempre acreditaram na educação como meio de evolução e inclusão social.

# AGRADECIMENTOS

Aos meus pais, Manoel Guimarães (*in memoriam*) e Maria José dos Santos Guimarães, pelo carinho, por me proporcionarem a oportunidade de estudar e por sempre estarem perto, física e mentalmente, nos momentos em que mais precisei de apoio emocional.

A toda minha família, que é o meu grande suporte emocional para enfrentar todos os desafios que a vida me oferece.

Aos meus orientadores, Eber Assis Schmitz e Priscila Machado Vieira Lima, por acreditarem na minha capacidade e compartilharem comigo o grande conhecimento que possuem. Agradeço a possibilidade de fazer parte dessa comunidade acadêmica tão valorosa e reconhecida mundialmente. Agradeço os momentos de orientação, as revisões cuidadosas e o incentivo constante.

Aos professores Antônio Juarez Alencar, Maria Luiza Machado Campos e Marcos Kalinowski, por terem gentilmente aceitado o convite para participar da banca examinadora deste trabalho e pelas contribuições e sugestões dadas.

À minha amiga Rosane Sfair, por ter me aberto as portas do estudo acadêmico, levando-me pelas mãos durante todo o início da caminhada.

Ao amigo Aldo Furriel Gonçalves, pelo incentivo, apoio, preocupação e companheirismo, principalmente nos momentos difíceis.

A todos os amigos que torceram por mim e entenderam meus períodos de ausência e afastamento da convivência social.

À professora Mônica Ferreira da Silva e aos colegas do curso de mestrado, pelas contribuições dadas ao longo das aulas de Metodologia da Pesquisa Científica.

Aos professores que participaram da avaliação deste trabalho nos diversos seminários do programa de mestrado.

À Petrobras, por proporcionar aos seus funcionários a oportunidade de continuar evoluindo em seus estudos e conhecimentos.

À minha ex-gerente na Petrobras, Martha Gomes Souza, por ter me apresentado este tema para estudo e aplicação na empresa, além de ter compartilhado o seu vasto conhecimento sobre o assunto, materializado na sua dissertação de mestrado defendida no PPGI UFRJ em 2002.

Aos meus gerentes na Petrobras, Luís Antônio Araújo, Eugênio Pedrosa e Marta Vidal Modenesi, pelo incentivo e pela liberação das horas de trabalho de que necessitei para a realização desta pesquisa.

Por fim, agradeço a todos aqueles que, de alguma forma, contribuíram na elaboração deste trabalho.

# RESUMO

GUIMARÃES, Denilson dos Santos. **Inconsistências em regras de negócio**: um método para identificação automatizada usando alloy. 2015. 87 f. Dissertação (Mestrado em Informática) – Instituto de Matemática, Instituto Tércio Pacciti, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2015.

Regras de negócio representam uma das formas mais efetivas de expressar conhecimento de negócio e, hoje em dia, empresas ágeis estão procurando identificar, representar e documentar essas regras, de forma que sejam claramente compreendidas e corretamente usadas por toda a empresa. Além disso, as regras de negócio são parte primordial de todos os sistemas de informação (SI) e neles estão sempre presentes. Assim, a identificação de erros em regras de negócio assume importância cada vez maior no desenvolvimento de SI, pois esses erros impactam em seus custos finais. Em particular, quanto mais tarde essa identificação ocorre, mais os custos de correção aumentam, podendo chegar a centenas de milhões de dólares por ano.

Esta dissertação apresenta o método MIIRNA para a verificação da consistência de um conjunto de regras de negócio aplicadas em um domínio específico de negócio, usando a ferramenta *Alloy* e os paradigmas *instance finding* e *small scope hypothesis*, propostos na literatura acadêmica.

A pesquisa é baseada em um exemplo de uso aplicado em dois domínios fictícios de negócio, simulando empresas reais, contendo um conjunto representativo de regras de negócio.

Verificou-se que o método é aplicável e eficiente para conjuntos de regras de negócio de tamanho e complexidade médios. Para conjuntos de porte e complexidade maiores, a *small scope hypothesis* traz grandes possibilidades de escalabilidade do método MIIRNA.

A partir desses resultados, podemos concluir que o método é viável, podendo ser usado tanto por analistas de sistemas quanto por analistas de negócio, de forma desacoplada da implementação dos SI. Também concluímos que o método é aderente ao ciclo proposto pela metodologia ABRD (*Agile Business Rule Development*), o que só aumenta a sua aplicabilidade.

Palavras-chave: Regras de Negócio. Alloy. Inconsistências.

# ABSTRACT

GUIMARÃES, Denilson dos Santos. **Inconsistências em regras de negócio**: um método para identificação automatizada usando alloy. 2015. 87 f. Dissertação (Mestrado em Informática) – Instituto de Matemática, Instituto Tércio Pacciti, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2015.

Business rules represent one of the most effective way of express business knowledge. Agile enterprises are investing in identify, represent and document them, so that they are easily understood and correctly used. In addition, business rules are primordial part of every Information System (IS) and are always present in them. This way, identifying mistakes in business rules is increasingly becoming a question of great concern for IS development, as they may have great impact on final costs. In particular, the later an error identification happens, the higher the correction cost, increasing these figures to hundreds of millions of dollars per year.

This dissertation presents the method MIIRNA for verifying the consistency of a business rules set applied to a particular business domain, using the *Alloy* tool and the *instance finding* and *small scope hypothesis* paradigms, proposed in academic literature.

The research is based on example of use applied to two fictional business domains, simulating actual enterprises, containing a significant business rules set.

It was verified the applicability and efficiency of the method for business rules set with medium size and complexity. For sets with bigger size and complexity, the *small scope hypothesis* provides great possibility of scalability of method MIIRNA.

With these results, it can be concluded that the method is viable and it can be used either by system analysts or by business analysts, dissociated from the IS implementation. It can also be concluded that the method is adherent to the cycle proposed in the ABRD methodology (*Agile Business Rule Development*), what makes it even more applicable.

Keywords: Business Rules. Alloy. Inconsistencies.

# LISTA DE FIGURAS

# Lista de Tabelas

# Lista de Siglas

| | |
|---|---|
| ABRD | *Agile Business Rule Development* |
| API | *Application Program Interface* |
| BDD | *Binary Decision Diagram* |
| BPM | *Business Process Management* |
| BRMS | *Business Rules Management System* |
| CNF | *Conjunctive Normal Form* |
| IDE | *Integrated Development Environment* |
| MDA | *Model Driven Architecture* |
| MIIRNA | Método para Identificação de Inconsistências em Regras de Negócio usando *Alloy* |
| OCL | *Object Constraint Language* |
| OMG | *Object Management Group* |
| PIM | *Platform Independent Model* |
| SBVR | *Semantics of Business Vocabulary and Rules* |
| SMV | *Symbolic Model Verifier* |
| SOA | *Service Oriented Architecture* |
| UML | *Unified Modeling Language* |

# Sumário

# INTRODUÇÃO

## 1.1 MOTIVAÇÃO

Regras de negócio são declarações que definem ou restringem algum aspecto do negócio e têm por fim afirmar a estrutura do negócio ou controlar ou influenciar o comportamento do negócio (BUSINESS RULES GROUP, 2015). De um modo genérico, são restrições que definem condições que devem ser satisfeitas em situações específicas. Não são a descrição de um processo ou de um procedimento, mas, na verdade, definem as condições sob as quais um processo é executado ou as novas condições que existirão após um processo ser completado (MORGAN, 2002).

Um conjunto de expressões de regras de negócio define a lógica desejada do negócio e, assim, representa uma das formas mais efetivas de expressar conhecimento de negócio. Por conta disso, hoje em dia, empresas ágeis e dinâmicas estão procurando identificar, representar e documentar essas regras, de forma que sejam compreendidas e claramente usadas por toda a empresa (SOUZA, 2002). Além disso, as regras de negócio são parte primordial de todos os sistemas de informação e neles estão sempre presentes, o que impacta diretamente nos prazos e custos envolvidos nos projetos de desenvolvimento/manutenção desses sistemas.

Um exemplo simples de regra de negócio que ilustra o exposto acima seria:

> *Um novo empréstimo não pode ser ofertado para um cliente com saldo devedor.*

Segundo DENNE e CLELAND-HUANG (2003), os inúmeros relatos de projetos cancelados ou não terminados indicam claramente que o desenvolvimento de *software* pode ser um empreendimento caro e arriscado. Ainda segundo esses autores, institutos de pesquisa dimensionaram o custo de projetos de *software* fracassados em impressionantes bilhões de dólares por ano, apenas nos Estados Unidos.

A forma mais efetiva de reduzir os custos e os riscos de um projeto de desenvolvimento de *software* se daria antes da própria escrita do *software*, sendo necessário buscar mais do que nunca aproximar as equipes de desenvolvimento (analistas e desenvolvedores) das

pessoas que definem os requisitos e articulam as necessidades para aquele *software*. Lamentavelmente, a falta de envolvimento dos *stakeholders* é uma fraqueza da maioria dos processos de desenvolvimento de *software*, em que o produto criado só se torna visível quando um *release* é disponibilizado (DENNE; CLELAND-HUANG, 2004).

Assim, o crescente aumento da importância dos sistemas de informação estimula o desenvolvimento de tecnologias que buscam tornar mais fácil, mais rápido e menos dispendioso o processo de desenvolvimento de aplicações (CUNHA, 2009). Sendo que todo esse processo se baseia num problema antigo e fundamental que aflige há anos os analistas de negócio e os analistas de sistema, que é a comunicação do conhecimento da área funcional para a área de TI (SOUZA, 2002).

Além disso, outra questão primordial é o processo de retenção do conhecimento empresarial, que está contido dentro das regras de negócio embutidas nos sistemas, e que pode ser afetado com a perda de pessoal próprio das empresas, dificultando principalmente as atividades de manutenção dos sistemas de informação, cujo volume é altíssimo e costuma trazer impactos negativos na satisfação dos usuários finais (SOUZA, 2002). Isso demanda fortemente a construção de mecanismos que elucidem essas regras e aproximem de forma sistemática os mundos empresarial e de TI, através de uma linguagem compreensível por toda a empresa. De acordo com ROSS (2009), essa abordagem tem potencial comprovado para reduzir a lacuna existente na passagem de conhecimento entre as equipes do negócio e as equipes de TI.

As regras de negócio estão espalhadas pelas organizações como um todo, seja embutidas nos processos de negócio, nos manuais e políticas da empresa, nas linhas de codificação dos sistemas de informação ou ainda na cabeça de seus *stakeholders*. As regras "*corretas*" trazem sucesso para a empresa e satisfação para seus clientes. As regras "*erradas*" trazem problemas e incertezas. Nunca se pode prever ao certo quando uma regra "*errada*" pode ocorrer, nem mesmo saber se ela existe, nem qual o impacto dessa falha (VON HALLE; GOLDBERG, 2006).

Dessa forma, a identificação de erros[1] em regras de negócio assume uma importância cada vez maior no desenvolvimento de sistemas de informação, pois esses erros impactam nos custos finais desses sistemas. Em particular, quanto mais tarde essa identificação ocorre, mais os custos de correção aumentam.

BOEHM (1981) estimou o custo percentual relativo para a correção de erros encontrados em cada uma das atividades do desenvolvimento de sistemas de informação, que aumentaria significativamente ao longo do tempo, sendo que um erro detectado na fase inicial teria custos de correção muito pequenos, de acordo com esses estudos, como exibido na Figura 1.


Figura 1: Evolução dos custos para correção de erros (BOEHM, 1981) (Figura adaptada pelo autor)

Podemos estender essa análise de custos para o ciclo de vida de um sistema de informação. Um sistema será concebido, construído, operado e, eventualmente, descontinuado. Apesar dos grandes benefícios alcançados ao se focar nos estágios iniciais desse ciclo, quando o sistema está sendo definido e criado e quando ele adquire suas características fundamentais, a maior parte do custo de propriedade de um sistema ocorre durante a sua vida operacional. De acordo com MORGAN (2002), a maioria das empresas de médio e grande porte investe em torno de 25% do seu orçamento de TI no desenvolvimento

[1] *Neste trabalho, utilizaremos a terminologia "erros", no lugar de "defeitos", adotada na norma* ***IEEE 1044 Standard Classifications for Software Anomalies****, por considerarmos que defeito é um erro descoberto após o software ser entregue, o que não é o caso tratado nesta pesquisa*

de novos sistemas e, consequentemente, 75% na manutenção e operação de sistemas existentes.

Dessa forma, pode-se perceber como é recomendável investir esforços para reduzir os custos de manutenção dos sistemas de informação, o que também pode ser alcançado com a identificação rápida e antecipada de erros nas especificações das regras de negócio, quando esses sistemas passarem por manutenções. Como citam ROSS e LAM (2011), é preciso deixar bem claro que regras de negócio e requisitos de *software* não são a mesma coisa. A principal diferença está no fato de que os requisitos evoluem antes e durante o desenvolvimento de um sistema, enquanto que as regras de negócio evoluem mesmo após a entrega desse sistema, o que faz toda a diferença, em relação principalmente aos custos envolvidos nas manutenções que o sistema irá sofrer. O *release* oficial de um sistema é apenas o começo da sua vida. E as principais mudanças a que ele se submete estão relacionadas com as regras de negócio, tendo em vista que o negócio constantemente necessita ajustar seus parâmetros e decisões. Ou seja, os ciclos de vida dos requisitos e das regras de negócio são diferentes.

A outra vantagem de se tratar as regras de negócio separadamente dos requisitos, através de fluxos específicos, é que o tratamento das regras pode ser atribuído a equipes multidisciplinares especializadas no negócio (analistas de negócio, *stakeholders*, analistas de processos), enquanto que o tratamento dos requisitos (funcionais e não funcionais) pode ser atribuído a equipes especializadas em *system design*. Desta forma, pode haver menor necessidade de entregáveis e provavelmente de codificação, de acordo com a tecnologia que for usada para a implementação (VON HALLE, 2002).

## 1.2 ARCABOUÇO CONCEITUAL

Regras de negócio começaram a receber um foco mais intenso na modelagem de sistemas de informação durante os anos 80, quando passaram a ser objeto de discussões pelos profissionais de gerenciamento de dados (VON HALLE; GOLDBERG, 2006). À medida que a tecnologia avançou e a demanda por maior agilidade na construção de sistemas de informação aumentou, as regras de negócio passaram a assumir um papel mais proeminente na arquitetura e no desenvolvimento desses sistemas.

O princípio básico dessa abordagem é que as regras de negócio devem ser especificadas e implementadas para que sejam rapidamente acessíveis e facilmente alteráveis, quando necessário, da forma mais amigável possível para o negócio (VON HALLE;

GOLDBERG, 2006). O principal objetivo a ser atingido com essa abordagem é melhorar a clareza e a completude das especificações de negócio usadas para o desenvolvimento de aplicações, mas com o mínimo impacto no prazo e no custo envolvidos.

Para isso, as regras de negócio devem exprimir a lógica do negócio da forma o mais independente possível dos procedimentos de implementação. Dependendo da forma como o sistema foi desenhado, as várias decisões de negócio envolvidas precisam ser otimizadas e orquestradas, mas isso é na verdade um problema de *design*, não de análise de negócio.

Um modelo contendo apenas as regras de negócio essenciais, verdadeiras e fundamentais é mais maleável e ajustável a verificações e mudanças, do que um modelo orientado a questões de *design* e implementação. Esse princípio é chamado por ROSS (2009) de *Rule Independence*, em que as regras de negócio devem ser externalizadas dos processos de negócio e das implementações de sistemas, de forma a serem tratadas como um ativo separado.

VON HALLE e GOLDBERG (2006) citam um exemplo real de um experimento, em que foi adotada uma abordagem estritamente hierárquica em uma modelagem de processos de negócio, sendo que foi dada prioridade às questões de *design* e as regras de negócio foram especificadas somente ao final da modelagem. Os modelos produzidos foram considerados extremamente complexos e de difícil navegação e entendimento. Ao se aplicar a abordagem orientada a regras de negócio desacopladas da implementação, de acordo com o princípio *Rule Independence*, os resultados indicaram uma redução superior a 50% dos componentes individuais de modelagem, ao mesmo tempo em que se aumentou a clareza do modelo como um todo, tanto da perspectiva do negócio quanto do desenvolvedor.

Outro experimento citado por ROSS e LAM (2011) descreve o caso de uma companhia financeira de porte médio, especializada em serviços de detecção de fraudes. Anteriormente, essa empresa mantinha suas regras de negócio embutidas nos códigos de implementação, o que acarretava um tempo médio de 30 a 60 dias para a disponibilização de um novo *release* da aplicação, com os critérios de detecção de fraude atualizados. Após o uso da abordagem orientada a regras de negócio desacopladas da implementação, esse tempo diminuiu para em torno de três a seis dias.

O mundo empresarial é altamente regulamentado, sujeito a mudanças rápidas, e com um universo muito grande de informações sendo manipuladas. Por conta disso, as operações

de negócio são de alta complexidade e se tornam cada vez mais complexas. Os analistas de negócio são as pessoas que deverão lidar com essa complexidade (ROSS; LAM, 2011).

O papel do analista de negócio é justamente especificar uma série de declarações claras sobre a lógica do negócio. A ênfase na clareza do entendimento é fundamental, porque as declarações de regras de negócio devem ser escritas de forma a não dar margem a dúvidas ou ambiguidades. Para isso, de um modo geral, recomenda-se que sejam expressas na forma de pequenas unidades gerenciáveis, como proposições lógicas (MORGAN, 2002).

Assumimos, então, que o formato geral de uma regra de negócio seja:

$$P_1, P_2, \ldots, P_n \rightarrow R$$

em que *P*'s são condições, cada vírgula significa um "*and*" lógico, *R* é a conclusão e o símbolo "→" é lido como "*então*".

O que nos leva para outra questão, que é a referente aos níveis de expressão possíveis para essa lógica de negócio. MORGAN (2002) propõe a existência de três níveis de detalhe para a expressão de regras de negócio, de acordo com os níveis de acessibilidade do conhecimento sobre o negócio e de possibilidade de automação dessa lógica:

1. *Informal:* caracterizada por uso de expressões escritas em linguagem natural/ coloquial, com um conjunto limitado de padrões. Por exemplo:

   Um cliente de conta-crédito deve ter no mínimo 18 anos de idade.

2. *Técnico*: caracterizada por uso combinado de referência a dados estruturados, operadores e linguagem natural controlada[2]. Por exemplo:

   *CreditAccount*
   *sel.customer.age >= 18*

3. *Formal*: caracterizada por uso de declarações conforme uma sintaxe mais fortemente definida com propriedades matemáticas específicas. Por exemplo:

   *(X,Y,(customerX) (creditAccount Y) (holder X Y) ) ==> (ge (age X) 18)*

Quanto à possibilidade de automação, o nível mais formal de expressão é o mais recomendável. Podem-se levar meses ou semanas para capturar e documentar um conjunto de regras de negócio. Contudo, uma vez completada essa etapa, elas podem ser implementadas em questão de horas (VON HALLE, 2002). Porém, o processo de transformação do nível informal para os níveis mais formais pode vir a acarretar erros de especificação.

O analista de negócio geralmente cria as regras no nível informal, tratando-as como pedaços de texto, que têm a vantagem de serem mais fáceis de serem lidas. Mas, nesse caso, todo o controle a respeito da estrutura, consistência e outras propriedades das regras fica dependendo exclusivamente da disciplina e capacitação do analista. Ou seja, a transformação em estruturas formais, levando a uma ou mais possibilidades de automação da regra, é basicamente uma atividade humana, com consequentes possibilidades de introdução de erros (MORGAN, 2002).

Os erros mais comuns possíveis de ocorrer durante a especificação de regras de negócio podem ser classificados da seguinte maneira, de acordo com MORGAN (2002):

- Regras que estão com erros de formação, não estando conformes a padrões locais ou a modelos para escrita de regras;
- Regras que usam termos inexistentes nos modelos conceituais que suportam o negócio;

---

[2] *CNL - Controlled Natural Language.*

- Regras que estão inconsistentes, levando a resultados ambíguos quando associadas a outras regras.

Os dois primeiros tipos de erros podem ser verificados através de revisões manuais de qualidade, que foquem em garantir que as declarações de regras de negócio formam uma descrição acurada da lógica de negócio que se pretende usar no desenvolvimento de sistemas de informação.

Já o terceiro tipo de erro pode ser mais difícil de se verificar manualmente, porque envolve a possibilidade de erros acarretados pela combinação de várias regras simultaneamente.

De acordo com MORGAN (2002) e ZHANG e NGUYEN (1994), os tipos de inconsistências mais comuns que ocorrem durante a especificação de regras de negócio são: redundância, sobreposição, conflito e incompletude.

- Redundância ocorre quando duas regras têm o mesmo conjunto de condições (cujas condições podem ser organizadas em ordens diferentes) e a mesma conclusão. Então, uma delas é dita redundante (ZHANG; NGUYEN, 1994).
- Sobreposição ocorre quando uma regra está total ou parcialmente contida em outra regra (MORGAN, 2002) ou quando duas regras têm a mesma conclusão e uma delas tem a parte condicional mais restritiva, sendo que a regra mais restritiva pode ser absorvida pela regra menos restritiva, pelo fato de que sempre que a primeira ocorrer, a outra também ocorre (ZHANG; NGUYEN, 1994).
- Conflito ocorre quando duas ou mais regras produzem resultados contraditórios (MORGAN, 2002) ou suas partes condicionais consistem do mesmo conjunto de átomos[3], mas as conclusões são mutuamente exclusivas (ZHANG;NGUYEN, 1994).
- Incompletude ocorre quando nem toda a informação necessária para produzir uma conclusão existe. Isto pode ser causado por lacunas deixadas

[3] *Fórmulas atômicas ou variáveis proposicionais.*

> inadvertidamente, conhecimento incerto ou perda de controle do crescimento da base de conhecimento (ZHANG; NGUYEN, 1994).

Como definir inconsistência? De acordo com HODGES (2001), crenças[4] podem ser expressas através de sentenças declarativas. A consistência, no que diz respeito à lógica, está relacionada com a compatibilidade entre as crenças de um conjunto. Assim, um conjunto de crenças é dito consistente se há alguma situação possível em que todas as sentenças sejam verdadeiras. E esse conjunto é dito inconsistente, no contexto da lógica, se não há nenhuma situação possível em que todas as sentenças sejam verdadeiras.

A representação formal $x \vdash$ significa que $x$ é semanticamente inconsistente, quando $x$ é um conjunto finito de sentenças e não há nenhuma estrutura em que todas as sentenças são definidas e verdadeiras. Dadas duas fórmulas $\psi$ e $\varphi$, elas são ditas logicamente equivalentes quando $\varphi \vdash \psi$ e $\psi \vdash \varphi$, i.e., não há nenhuma estrutura em que todas as sentenças de $\varphi$ sejam verdadeiras e $\psi$ seja falsa e vice-versa. Se duas sentenças podem ser simbolizadas como logicamente equivalentes, usando a mesma interpretação, então as duas sentenças são verdadeiras exatamente nas mesmas situações e se comportam da mesma maneira em todas as maneiras de lógica (HODGES, 2001).

A abordagem tradicional para a verificação de consistência consiste na comprovação de teoremas[5]. Um comprovador automático[6] tenta construir uma prova de que uma afirmação ocorre. Se for bem-sucedido, a afirmação é válida. Se falha, contudo, a afirmação pode ser válida ou inválida (JACKSON, 2006).

Entretanto, Alonzo Church (apud HODGES, 2001) provou em 1936 que é impossível comprovar um teorema através de um método sistemático para testar a consistência de conjuntos de sentenças escritas em lógica de primeira ordem. Entretanto, ainda é possível comprovar a inconsistência desses conjuntos (HODGES, 2001).

De forma similar, Edsger W. Dijkstra (apud JACKSON, 2006) afirmou que o teste de programas pode ser usado para mostrar a presença de erros, mas nunca para mostrar a sua

---

[4] *"Beliefs", no original do autor.*

[5] *"Theorem proving", no original do autor.*

[6] *"Automatic prover", no original do autor.*

ausência. Lamentavelmente, a maioria dos erros nos códigos de programação não é encontrado nos testes.

Assim, de forma a testar a consistência de um conjunto de regras de negócio, fazemos uma simples suposição: todas as inconsistências ocorrerão envolvendo diretamente um simples par de regras. Neste caso, será necessário checar a consistência de pares de sentenças.

Por exemplo, vamos considerar o conjunto, a seguir, composto pelas regras *r1* e *r2*:

*(r1) Gosto de café -> Bebo cafeína*
*(r2) Gosto de chá -> Bebo cafeína*

O número de ocorrências possíveis de *n* sentenças combinadas em pares será dado pela fórmula:

$$C\binom{n}{2} = \frac{n!}{2!\,(n-2)!} = \frac{n.\,(n-1)}{2} = \frac{n^2 - n}{2}$$

A complexidade computacional de tempo deste cálculo é da ordem de grandeza do crescimento quadrático[7], i.e., seus valores crescem proporcionalmente ao argumento da função elevado ao quadrado, representado por $\Theta(n^2)$. Por exemplo, quando *n* representa mil sentenças, o número de operações será na ordem de um milhão (MOORE; MERTENS, 2011). Essa complexidade crescerá conforme a checagem de consistência envolver grupos de três ou mais sentenças, até tornar-se exponencial, quando todos os conjuntos puderem vir a ter tamanho $2^n$.

O número de ocorrências possíveis de *n* sentenças combinadas em grupos de três ou mais sentenças será dado pela fórmula:

$$C\binom{n}{2} + C\binom{n}{3} + C\binom{n}{4} + \cdots + C\binom{n}{n} = 2^n$$

De modo a reduzir esse grande número de operações necessárias, é recomendado usar uma abordagem diferente, como a *instance finding*, proposta por JACKSON (2006). Mais do que tentar construir uma prova de que uma afirmativa acontece, a *instance finding* procura

[7] *"Quadratic growth", no original do autor.*

por uma refutação, checando a afirmativa e procurando por um caso particular em que se descobre que a afirmativa não acontece. Este caso é chamado de “contraexemplo”.

A *instance finding* tem um alcance muito maior do que os testes tradicionais e tende a ser muito mais efetiva em achar erros, porque a maioria dos erros tem pequenos contraexemplos. Em outras palavras, basta um único pequeno contraexemplo para provar que uma afirmativa é inválida. Para tornar a *instance finding* possível, é necessário definir um escopo que limite o tamanho das instâncias consideradas. Mesmo um pequeno escopo, geralmente define um enorme espaço de instâncias. Sistemas que falham em grandes instâncias quase sempre iriam falhar em pequenas instâncias com propriedades similares, mesmo se tais pequenas instâncias não ocorrerem na prática. Assim, checando todas as pequenas instâncias, as grandes também serão efetivamente checadas. Isso é chamado por JACKSON (2006) de *small scope hypothesis*.

## 1.3 PERGUNTA DA PESQUISA

Neste cenário, o problema tratado neste trabalho, e consequentemente a pergunta da nossa pesquisa, é:

> “Como identificar inconsistências em regras de negócio, nos estágios iniciais do ciclo de desenvolvimento dos sistemas de informação, quando ainda é comprovadamente reduzido o custo das correções de erros em *software*?”

## 1.4 QUESTÃO DA PESQUISA

Diante dos argumentos apresentados, a questão de pesquisa deste trabalho é:

> “Usando um método estruturado, baseado na *instance finding* e na *small scope hypothesis*, com o apoio de uma ferramenta de análise de consistência de modelos (no caso, o *Alloy*), é possível identificar previamente inconsistências em um conjunto de regras de negócio.”

## 1.5 OBJETIVO

Esta dissertação apresenta os resultados de um trabalho de pesquisa visando ao desenvolvimento de um método adequado à verificação antecipada da consistência de um conjunto de regras de negócio. Esse método foi denominado de MIIRNA e pode exibir precisamente a inconsistência existente em uma especificação de regras de negócio, sendo que esta versão do método se restringirá às inconsistências envolvendo diretamente um par de regras.

Em continuidade, a inclusão de novas regras ou a atualização das existentes pode ser facilmente absorvida pelo procedimento de verificação. Isso torna este método uma opção robusta e confiável para analistas de regras de negócio, tanto no desenvolvimento quanto na manutenção de sistemas de informação.

A verificação antecipada é possibilitada porque o método MIIRNA é aplicado de forma desassociada do código do *software*, de forma que não demanda a elaboração de casos de teste, que geralmente só podem ser executados em etapas posteriores dos projetos de desenvolvimento de aplicações e não são totalmente confiáveis, como já explanado nesta dissertação.

A relevância deste trabalho se caracteriza por não haver ainda na literatura acadêmica evidências de um método estruturado e automatizado para a identificação de inconsistências em um conjunto de regras de negócio, o que reforça o valor da contribuição do método MIIRNA aqui proposto.

A utilização do método MIIRNA faz acreditar que seja possível alcançar uma redução substancial nos custos de desenvolvimento de sistemas de informação, principalmente no que concerne ao custo de retrabalho.

## 1.6 METODOLOGIA DA PESQUISA

O período de realização da pesquisa não foi suficiente para a aplicação do método MIIRNA em um ambiente real ou para obter a percepção da sua qualidade por parte de um grupo de analistas de negócio, porque um trabalho experimental criterioso necessita de mais de uma etapa de execução e mais de uma equipe envolvida, sendo muito demorado e incompatível com o tempo disponível para a elaboração de uma dissertação de mestrado.

Também não foi detectada, dentro do escopo e do foco do mapeamento sistemático de literatura realizado, a existência de nenhum outro método com objetivo semelhante, de forma que os resultados pudessem ser comparados.

Dessa forma, o objetivo da pesquisa foi buscar indícios de que o método MIIRNA é viável através da sua execução. O método de pesquisa escolhido foi exemplificar o uso do método MIIRNA, usando dois domínios de negócio fictícios e comparando os resultados dessas duas execuções. Os domínios escolhidos foram os de uma empresa fictícia de empréstimos, descrita em MORGAN (2002), de complexidade simples, e o da fictícia locadora de veículos *EU-Rent*, descrita em OBJECT MANAGEMENT GROUP (2015), de complexidade maior.

A comparação foi realizada através da avaliação dos resultados de cada execução. Para isto, foram usados indicadores de complexidade estrutural (conceitos, relações, fatos e regras) relativos ao conjunto de regras de negócio avaliado, propostos pelo próprio autor deste método. Estes indicadores permitiram ter uma percepção da efetividade do método MIIRNA, de forma a permitir a identificação de inconsistências em um conjunto de regras de negócio.

## 1.7 ORGANIZAÇÃO

O restante desta dissertação está organizado como segue. A seção 2 apresenta o mapeamento sistemático de literatura efetuado para este trabalho. A seção 3 apresenta o referencial teórico usado para a dissertação. A seção 4 descreve o método MIIRNA para verificar a consistência de um conjunto de regras de negócio usando a ferramenta *Alloy*. A seção 5 ilustra o método MIIRNA através de um exemplo de uso aplicado a um domínio de negócio de complexidade simples. A seção 6 ilustra o método MIIRNA através de um exemplo de uso aplicado a um domínio de negócio de complexidade média. A seção 7 apresenta a análise dos dois exemplos de uso. A seção 8 apresenta os trabalhos relacionados com o tema deste trabalho. Finalmente, a seção 9 apresenta as considerações finais e aponta para tópicos em desenvolvimento ou a serem explorados futuramente.

# 2 MAPEAMENTO SISTEMÁTICO DE LITERATURA

No mapeamento sistemático de literatura, buscou-se identificar as oportunidades, os desafios e as propostas para identificação e tratamento de inconsistências em regras de negócio usadas no desenvolvimento de sistemas de informação. Além disso, buscou-se verificar a existência de abordagens para a identificação automatizada dessas inconsistências. O objetivo principal foi tentar identificar características similares que definam lacunas de pesquisa (ou oportunidades).

## 2.1 PROCEDIMENTOS METODOLÓGICOS

Para a realização deste estudo, foi definido um processo composto por quatro etapas principais, baseado no processo para mapeamento sistemático em engenharia de *software* aplicado por PETERSEN et al. (2008) e descritas a seguir:

### 2.1.1 ETAPA 1 – DEFINIÇÃO DO ESCOPO DO MAPEAMENTO

Compreendeu a construção do protocolo da pesquisa com o planejamento do estudo e a definição do processo, das questões de pesquisa e dos critérios de busca. Foi definida uma única questão de pesquisa: “Como se deu a evolução histórica das pesquisas acadêmicas internacionais em relação a inconsistências em regras de negócio?”.

### 2.1.2 ETAPA 2 – BUSCA E SELEÇÃO DE ESTUDOS PRIMÁRIOS

A busca dos artigos ocorreu de forma automática, através de mecanismos de busca disponibilizados no acervo das bibliotecas digitais. A seleção ocorreu pela análise do título e resumo das publicações. A lista completa de estudos incluídos no mapeamento pode ser encontrada no Apêndice A desta dissertação. As pesquisas selecionadas foram identificadas pelo código [BRIXXX].

Foram selecionadas bibliotecas digitais que possuem mecanismos de busca que permitam o uso de expressões lógicas ou equivalente e que comportem a busca no texto completo ou em campos específicos das publicações.

Assim, foram selecionadas as seguintes bibliotecas digitais:

- IEEE (em modo *Advanced Search*): <*http://ieeexplore.ieee.org*>;
- Scopus (em modo *Advanced Search*): <*http://www.scopus.com*>;
- ACM Digital Library (em modo *Advanced Search*): <*http://dl.acm.org*>;
- DBLP (em modo *Complete Search*): <*http://dblp.org*>.

A pesquisa nas bibliotecas digitais foi feita buscando como palavras-chaves “business rules”, “Business Rules”, “inconsistency” e “inconsistent”. A *string* de busca utilizada foi (“business rules” OR “Business Rules”) AND (“inconsistency” OR “inconsistent”), com exceção da biblioteca DBLP, em que foi utilizada a *string* “Business~Rules~inconsist~”, por limitação do mecanismo de busca daquela biblioteca.

Foi estipulado o período de 2010 a 2014, para que fossem retornados apenas artigos relativamente recentes e, assim, mais relevantes para a nossa pesquisa. Os estudos com data anteriores a esse período e que foram considerados relevantes para a nossa pesquisa foram analisados no capítulo “Trabalhos Relacionados” desta dissertação.

### 2.1.3 ETAPA 3 – EXTRAÇÃO DE DADOS

Nesta etapa, buscou-se extrair informações de cada um dos artigos encontrados, de forma a responder à pergunta de pesquisa.

Os retornos das pesquisas nas bibliotecas digitais estão relacionados na Tabela 1:

Tabela 1: Retorno das bibliotecas digitais

| BIBLIOTECA DIGITAL | TOTAL DE RESULTADOS |
|---|---|
| IEEE | 199 |
| Scopus | 30 |
| ACM Digital Library | 28 |
| DBLP | 3 |
| TOTAL: 260 resultados | |

Para a seleção dos resultados, foram usados os critérios de inclusão e de exclusão, expostos na Tabela 2:

Tabela 2: Critérios de inclusão e de exclusão

| CRITÉRIOS DE INCLUSÃO | CRITÉRIOS DE EXCLUSÃO |
|---|---|
| Textos relacionados com levantamento de regras de negócio para desenvolvimento de sistemas. | Textos escritos em idiomas diferentes de inglês e português. |
| Textos contendo propostas para a identificação de inconsistências em regras de negócio. | Textos relacionados com assuntos não pertinentes a regras de negócio e a inconsistências. |
| Textos relacionados com automatização de decisões de negócio. | - |

Os resultados foram tabelados em uma planilha *MS Excel*, para identificação de duplicatas e análise dos critérios de inclusão e exclusão citados na Tabela 2, usando como argumentos os campos "autores", "título" e "resumo" das publicações.

### 2.1.4 ETAPA 4 – ANÁLISE E SÍNTESE

A partir dos dados extraídos na fase anterior, deu-se o processo de interpretação dos resultados, criação de tabelas e gráficos para exibição dos dados e descrição das evidências identificadas nos estudos primários selecionados.

Após o processo de consolidação dos resultados iniciais das pesquisas, chegou-se a um total de 32 artigos, de acordo com o ano de publicação, conforme exibido na Figura 2:

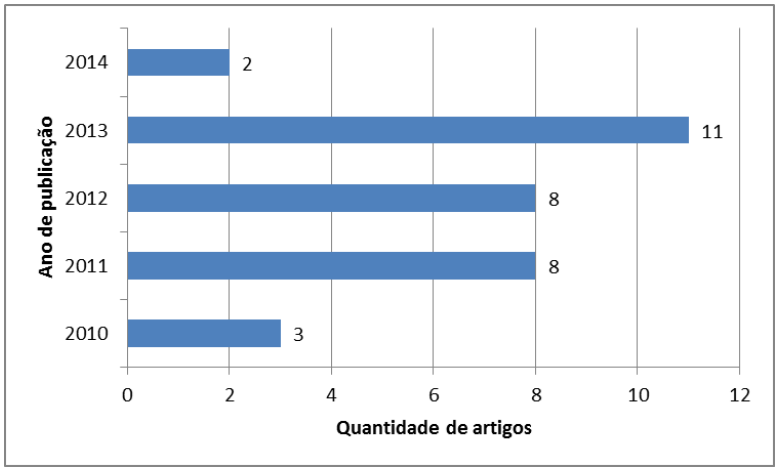



Figura 2: Evolução das pesquisas

Os artigos foram distribuídos nas seguintes categorias, propostas pelo autor desta dissertação, conforme é descrito na Tabela 3 e totalizados na Figura 3:

Tabela 3: Categorias das pesquisas

| CATEGORIAS |
|---|
| Uso de taxonomias para identificar referências cruzadas em especificações de negócio |
| Geração automática de testes baseados em regras de negócio |
| Detecção e extração automáticas de regras de negócio em textos regulatórios |
| Uso de modelos orientados a engenharia reversa para captura de lógica de negócio em sistemas legados |
| Tolerância a determinadas métricas de inconsistência em declaração de regras de negócio |
| Uso de redes *Petri* para modelagem de especificações de negócio |
| Uso ou construção de ontologias para identificação e tratamento de inconsistências em especificações de negócio |
| Uso ou construção de linguagens para especificação de requisitos e/ou declaração de regras de negócio |
| Uso de modelos SBVR[8] da OMG[9] para declarar regras de negócio |
| Uso de linguagens semiformais ou formais para declarar regras de negócio |

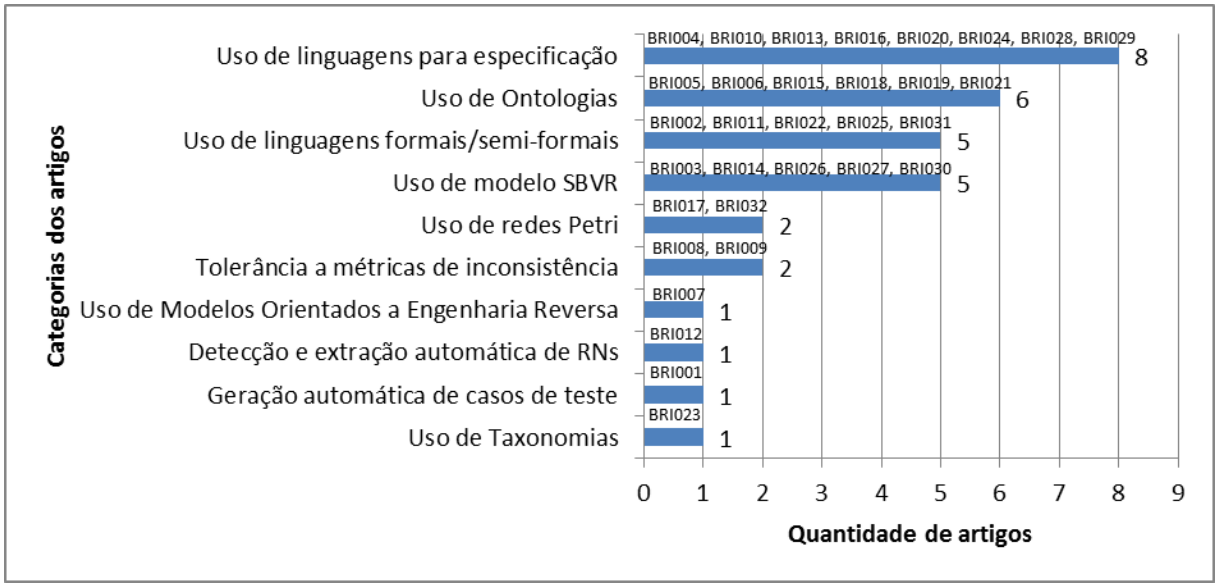



Figura 3: Distribuição das pesquisas por categoria

[8] *Semantic Business Vocabulary and Rules.*

[9] *Object Management Group.*

## 2.2 CONCLUSÕES

As informações extraídas desses artigos permitiram observar a relevância do tema "inconsistência de regras de negócio" para as recentes pesquisas acadêmicas. Porém, há certa variação quanto à abordagem para o seu tratamento.

Chamou a atenção a existência de dois artigos, [BRI008] e [BRI009], escritos pelos mesmos autores, que pesquisaram a tolerância a determinados índices de inconsistência entre as especificações de regras de negócio. Podemos fazer uma correlação desse entendimento com o paradigma *small scope hypothesis*, usado no método MIIRNA proposto neste trabalho.

O artigo [BRI018] foi incluído na categoria “Uso ou construção de ontologias para identificação e tratamento de inconsistências em especificações de negócio”, mas também poderia constar da categoria “Uso de modelos SBVR da OMG para declarar regras de negócio”.

O artigo [BRI003] aborda a tradução de especificações em SBVR para a linguagem Alloy, porém não é a única ferramenta que realiza tais traduções, como é citado no *abstract* do artigo, tendo em vista que a ferramenta NL2AlloyviaOCL (UNIVERSITY OF BIRMINGHAM, 2015) tem funcionamento semelhante.

A maioria dos artigos tende para a pesquisa sobre a adoção de linguagens para especificações de negócio e sobre o uso de ontologias para gerenciar a evolução do conhecimento dentro das empresas, de forma a diminuir ou tratar a ocorrência de inconsistências.

Porém, dentro do escopo e do foco do mapeamento sistemático de literatura realizado, nenhum dos artigos pesquisados abordou a elaboração de um método estruturado e automatizado para a identificação de inconsistências em um conjunto de regras de negócio, reforçando o valor da contribuição do método MIIRNA proposto neste trabalho.

# 3 REFERENCIAL TEÓRICO

O método apresentado nesta dissertação para a verificação da consistência de um conjunto de regras de negócio, de forma a permitir a identificação antecipada de inconsistências nessas regras, foi baseado nos seguintes itens de referencial teórico:

## 3.1 REPRESENTAÇÃO DE REGRAS DE NEGÓCIO

Regras de negócio são declarações que restringem, derivam e dão condições para a existência e a representação do conhecimento sobre o negócio. Regras de negócio não são apenas descrições de um processo ou procedimento. Mais do que isso, também definem as condições sob as quais um processo é executado ou as novas condições que irão existir após um processo ter sido completado (MORGAN, 2002).

Regras de negócio podem ser representadas, por exemplo, usando *Semantics of Business Vocabulary and Rules* (SBVR) (OBJECT MANAGEMENT GROUP, 2015) ou usando linguagem de negócio. A SBVR é apropriada para ser usada por especialistas do negócio, já que permite a representação do vocabulário de negócio, assim como a das regras, usando linguagem natural controlada. Linguagem de negócio é mais intuitiva a representantes do negócio e analistas de negócio e também pode ser usada de forma não estruturada. Ambas as representações geram o vocabulário de negócio, composto por termos e fatos, que serão usados para formular as regras de negócio.

## 3.2 CONSTRUÇÃO DO MODELO *ALLOY*

*Alloy* é uma linguagem de modelagem estrutural baseada em lógica de primeira ordem, usada para expressar restrições e comportamentos de estruturas complexas e para descrever estruturas. Um modelo *Alloy* é uma coleção de restrições que descrevem implicitamente um conjunto de estruturas (JACKSON, 2015).

A linguagem *Alloy* é suportada por uma ferramenta interna embutida chamada *Alloy Analizer*, que é um solucionador o qual, através das restrições de um modelo, encontra estruturas que as satisfaçam. Tecnicamente falando, é um "*model finder*". Dada uma fórmula lógica (escrita em linguagem *Alloy*), tenta achar um modelo, ou seja, uma atribuição de valores a variáveis, que torne a fórmula verdadeira. Pode ser usada tanto para explorar o modelo,

gerando amostras de estruturas (aqui chamadas de exemplos), quanto para avaliar as propriedades do modelo, gerando contraexemplos (JACKSON, 2015).

A estrutura básica de um modelo *Alloy* consiste de (JACKSON, 2006):

- Declarações de assinaturas, identificadas pela palavra reservada *sig*, cada uma representando um conjunto de átomos e podendo também introduzir alguns campos, cada um representando uma relação;
- Parágrafos de restrições, identificados pelas palavras reservadas *fact*, *fun* e *pred*, que registram várias formas de restrições e expressões;
- Parágrafos de assertivas, identificados pela palavra reservada *assert*, que registram propriedades que se espera que ocorram;
- Parágrafos de comandos, identificados pelas palavras reservadas *run* e *check*, com as instruções para o *Alloy Analyzer* executar análises.

As declarações de assinatura configuram uma hierarquia de classificação que pode ser estendida em outros conjuntos disjuntos, usando a palavra reservada *extend*. A marcação de uma declaração de assinatura estendida, usando a palavra reservada *abstract*, indica que todos os elementos do conjunto pertencem a suas extensões (JACKSON, 2006).

Assim, um modelo *Alloy* consiste de um conjunto de restrições lógicas que são declaradas através de parágrafos *sig*. Quando esse modelo é instanciado pelo *Alloy Analizer*, átomos são gerados a partir das assinaturas, respeitando as restrições lógicas do modelo. Em outras palavras, uma assinatura no nível de modelo introduz um conjunto de átomos no nível de instâncias.

A Figura 4 mostra um exemplo de um modelo *Alloy* bem simples, que inclui três assinaturas, “Person”, “Man” e “Woman”, que terão átomos específicos no nível de instâncias gerados pelo *Alloy Analizer*.

```
// Simple example
abstract sig Person {     // Signature
  father: lone Man,       //  A declaration
  mother: lone Woman      //  Another declaration
}
sig Man extends Person {
  wife: lone Woman
}
sig Woman extends Person {
  husband: lone Man
}

pred show {}
run show
```

Figura 4: Exemplo de modelo *Alloy*

As assinaturas podem incluir declarações de campos, que introduzem relações entre as assinaturas. Assim, no exemplo citado, a assinatura “Person” tem dois campos “father” e “mother”, que introduzem as relações “Person -> father” e “Person -> mother”.

Em cada assinatura ou declaração de campo, é possível usar palavras reservadas de multiplicidade para restringir a cardinalidade da relação. Essas palavras reservadas são *one*, *lone*, *some* e *set*, que restringem a cardinalidade para um, um ou zero, um ou mais e zero ou mais elementos, respectivamente. No exemplo citado, a palavra reservada *lone* na declaração de campo “father” da *signature* “Person” indica que para toda instância de “Person” haverá sempre zero ou uma instância de “Man” associada à assinatura através da relação “father”.

Um parágrafo *fact* registra uma restrição que se assume que sempre ocorra. Um parágrafo *assert* introduz uma restrição que se intenciona ser derivada dos fatos do modelo. Um parágrafo *pred* define uma restrição na forma de um predicado lógico. O comando *run* instrui o *Alloy Analizer* para tentar achar uma solução (ou exemplo) para as restrições, geralmente agrupadas em predicados de teste. O comando *check* orienta o *Alloy Analizer* a encontrar um contraexemplo para uma assertiva (JACKSON, 2006).

No exemplo citado, a execução do comando *run* para o parágrafo *pred* “Show” pelo *Alloy Analizer* resultou em vários exemplos de soluções para as restrições lógicas declarados no modelo *Alloy*, como é mostrado na Figura 5.

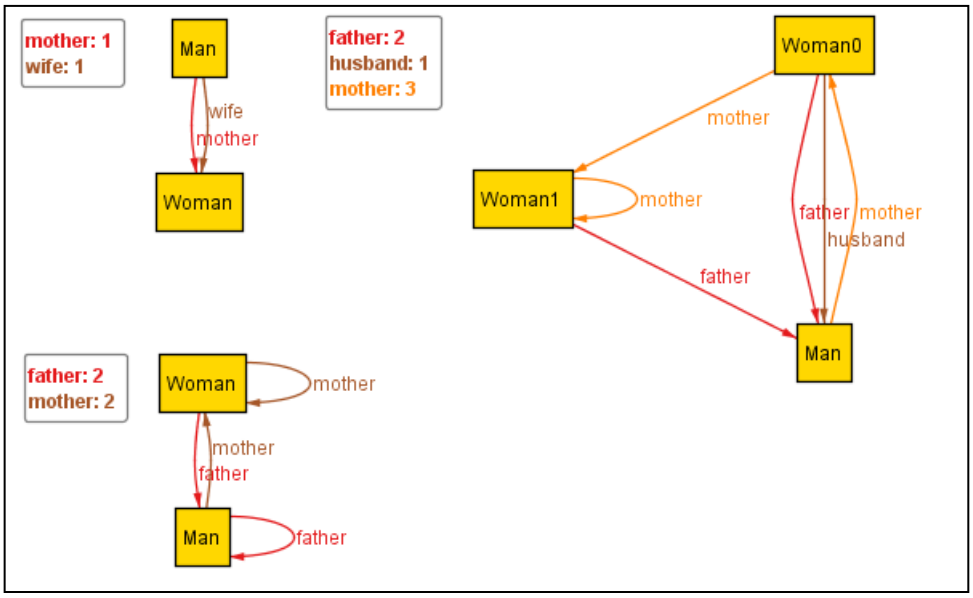



Figura 5: Exemplos de soluções encontradas pelo *Alloy Analizer*

Contudo, os exemplos resultantes indicaram que o modelo *Alloy* estava com poucas restrições lógicas[10], o que possibilitou a existência de instâncias em que as relações “mother” e “wife” poderiam acontecer simultaneamente para um mesmo “Person” do tipo “Man”. Assim, a inclusão de parágrafos *fact,* incluindo essas restrições, fez com que na subsequente execução do comando *run* para o parágrafo *pred* “Show”, o *Alloy Analizer* resultou em exemplos de soluções em que esses erros não ocorrem, como é mostrado na Figura 6.

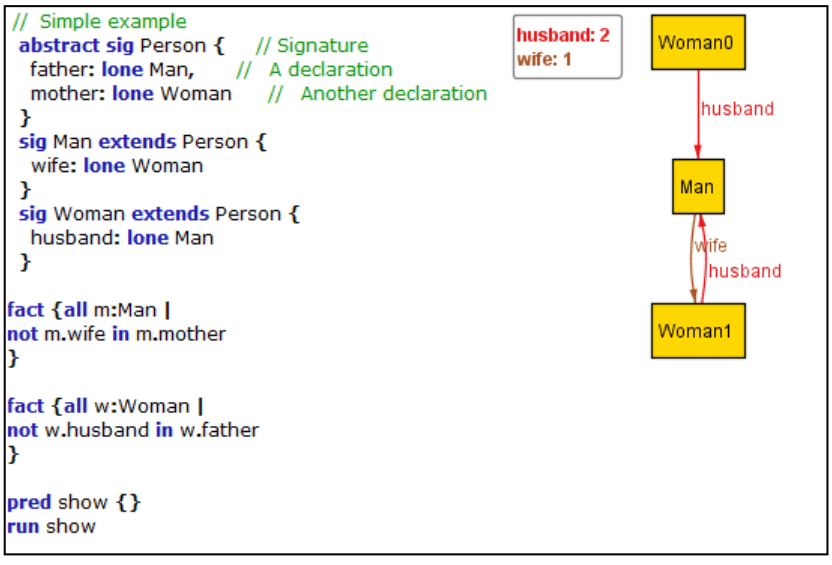



Figura 6: Exemplo de soluções após a inclusão de parágrafos *fact*

[10] *Underconstrained.*

## 3.3 EXECUÇÃO DE CICLO PARA AS REGRAS DE NEGÓCIO

Para o desenvolvimento de aplicações baseadas em regras de negócio, é recomendado adotar uma abordagem bem documentada e estruturada. A metodologia *Agile Business Rules Development* (ABRD) (ECLIPSE FOUNDATION, 2015) é a primeira metodologia gratuita e não associada a fabricante que provê um ciclo de vida completo para as regras de negócio, desde a descoberta até a governança, usando uma abordagem iterativa e ágil[11]. A metodologia ABRD é um processo de desenvolvimento de *software* incremental e iterativo, adaptado para enfrentar o desafio de desenvolver soluções para decisões de negócio. Em contraste com as metodologias tradicionais de desenvolvimento de *software*, a ABRD foca em um forte envolvimento dos definidores do negócio[12] como principais atores do processo de desenvolvimento. Sabendo-se que é impossível definir de uma só vez todas as regras de negócio envolvidas em uma solução de decisões de negócio, a abordagem ágil e iterativa proposta pela ABRD é a mais indicada para se atingir sucesso nos projetos de desenvolvimento dessas soluções (TAYLOR, 2011).

As atividades da ABRD se dividem em seis categorias, cada uma podendo ser executada múltiplas vezes a cada vez que o ciclo é executado (ECLIPSE FOUNDATION, 2015), como mostrado na figura 7.

---

[11] *O termo "ágil" está sendo usado aqui no sentido dos métodos ágeis de gerenciamento de projetos de software.*

[12] S*takeholders, no original em inglês.*

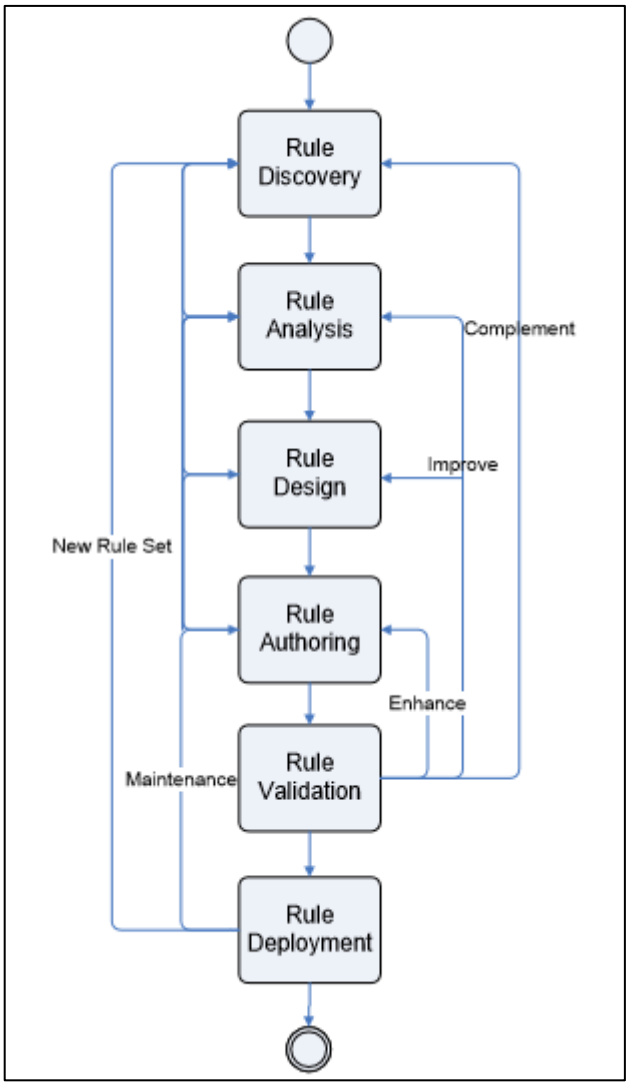



Figura 7: Disciplinas da ABRD (ECLIPSE FOUNDATION, 2015)

As atividades da ABRD são descritas a seguir, conforme BOYER e MILI (2011):

1. *Rule Discovery* constitui um processo iterativo que busca elaborar artefatos simples de modelagem (tais como descrições de regras, diagramas de entidades de negócio e mapas de processos de negócio) e identificar um subconjunto de regras, documentando-as, em vez de dedicar meses identificando todas as regras de uma vez e produzindo um documento massivo;
2. *Rule Analysis* tem como objetivo entender o significado da regra, tal como foi declarada pelo representante do negócio e pelos especialistas de domínio, e remover qualquer ambiguidade e questão semântica, preparando as regras para a implementação futura;

3. *Rule Design* possui o propósito de obter uma primeira passagem completa pelo processo de desenvolvimento, abordando as principais questões de *design*, construindo as bases para futuros refinamentos. Inclui questões arquiteturais e prototipação inicial;
4. *Rule Authoring* procura por questões genéricas sobre a expressão de regras de negócio em um modo independente de tecnologia, particularmente abordando questões relacionadas com o vocabulário de negócio em si e com as estruturas das regras;
5. *Rule Validation* elabora testes unitários para avaliar os resultados das regras e executa os testes, até que todos sejam bem-sucedidos, visando garantir que as regras reflitam precisamente as intenções do negócio;
6. *Rule Deployment* aborda questões comuns a respeito de desenvolvimento, empacotamento e integração das regras, abordando considerações maiores sobre desenvolvimento e integração, tais como suporte a transações, escalabilidade e acesso a dados.

O método proposto neste trabalho é particularmente aplicável à atividade *Rule Analysis* da ABRD, principalmente no que concerne a um conjunto de regras de negócio.

O propósito principal da *Rule Analysis* é refinar esse conjunto de regras, descobertas e não analisadas ainda, em um modelo lógico de regras, caracterizado por ser um conjunto de regras com alta qualidade semântica, ou seja, que possuem integridade em si mesmas e fazem sentido em respeito ao relacionamento com as outras regras do conjunto. A *Rule Analysis* foca no entendimento individual de cada regra e como as regras semanticamente se relacionam umas com as outras, independentemente de como serão implementadas e executadas (VON HALLE, 2002).

## 3.4 VERIFICAÇÃO DAS REGRAS DE NEGÓCIO

A busca pela qualidade das regras de negócio exige duas atividades genéricas e complementares: validação e verificação (ROSS, 2009).

Validação significa garantir a correção das regras de negócio. No que diz respeito aos propósitos do negócio, possui o objetivo de certificar que, quando as regras forem aplicadas, os resultados serão apropriados nas circunstâncias relevantes. Consiste majoritariamente em

uma questão de inspeção humana e pode ser alcançada através do uso de diagramas, cenários de testes e análises individuais de cada regra de negócio. Validação de regras se refere a avaliar **"se foram identificadas as regras de negócio certas"** (ROSS, 2009).

Verificação, por outro lado, significa avaliar a adequação das regras de negócio, no que diz respeito à consistência lógica do conjunto (conforme descrito na seção 1.2 desta dissertação), e envolve descobrir regras de negócio (geralmente duas ou mais em combinação) que exibirão alguma anomalia. Verificação de regras se preocupa com **"se as regras de negócio estão certas"** (ROSS, 2009).

A atividade de verificação apoia a de validação, pois ajuda a garantir que, durante as simulações necessárias para a validação de um modelo, não sejam exibidos comportamentos irreais ou instáveis, assim como permite identificar limitações ou restrições para as simulações.

## 3.5 ESCOPO DO MÉTODO

No método proposto neste trabalho, usaremos a ferramenta *Alloy* para suportar e automatizar a atividade de verificação, descrita acima. Contudo, os exemplos de soluções possíveis exibidos pelo *Alloy* também podem ser usados como recursos de simulação do modelo e auxiliar a atividade de validação.

Dos tipos de inconsistências mais comuns que ocorrem durante a especificação de regras de negócio, descritas na seção 1.2 desta dissertação, consideraremos os problemas das regras redundantes e sobrepostas da mesma maneira, já que são tratadas de forma similar. Também vale ressaltar que as regras incompletas estarão fora do escopo deste trabalho, por estarem relacionadas ao escopo da atividade de validação, de caráter essencialmente humano e de difícil automatização. Além disso, esta versão do método se restringirá às inconsistências envolvendo diretamente um par de regras de negócio, porque a correção dessas inconsistências depende da análise visual dos contraexemplos exibidos pelo *Alloy.*

As situações de redundância/sobreposição poderiam ser aceitas, no contexto da lógica, uma vez que toda solução para a sentença mais específica também seria solução para a mais genérica. No contexto de regras de negócio, entretanto, não é desejável manter a regra mais genérica, pois ela admitiria cenários que não contemplariam a mais específica. Isto poderia levar a mal-entendidos por parte dos analistas de negócio que consultassem tal base de conhecimento e também a perda de tempo por parte dos algoritmos que viessem a investigar os cenários que não contemplassem as regras mais específicas.

Assim, o método tratará basicamente as situações de conflito e de redundância de regras de negócio, conforme descrito na seção 1.2 desta dissertação.

Para solucionar as situações de conflito identificadas pelo método proposto neste trabalho, é necessário buscar o envolvimento dos definidores do negócio, durante a atividade de validação descrita acima. Já para as situações de redundância, o próprio analista de negócio pode solucionar, bastando excluir a redundância indicada entre as regras.

# 4 MÉTODO MIIRNA

Nesta seção, vamos descrever o método MIIRNA (**M**étodo para **I**dentificação de **I**nconsistências em **R**egras de **N**egócio usando o **A**lloy).

## 4.1 PASSOS DO MÉTODO

O método MIIRNA, proposto para a verificação da consistência de um conjunto de regras de negócio (RN), consiste nos seguintes passos, conforme exibido na Figura 8:

1. Obter e especificar o vocabulário de negócio;
2. Identificar as RNs e especificá-las de forma estruturada, usando linguagem de negócio para representação.
3. Construir o modelo conceitual no *Alloy*, descrevendo os conceitos e fatos de negócio e as relações entre eles;
4. Introduzir cada nova RN no modelo *Alloy*, verificando e solucionando possíveis situações de inconsistência geradas;
5. Repetir o passo 4, até que todas as regras de negócio tenham sido introduzidas e nenhuma inconsistência no modelo *Alloy* seja encontrada, de forma que se tenha construído o conjunto consistente de regras de negócio.

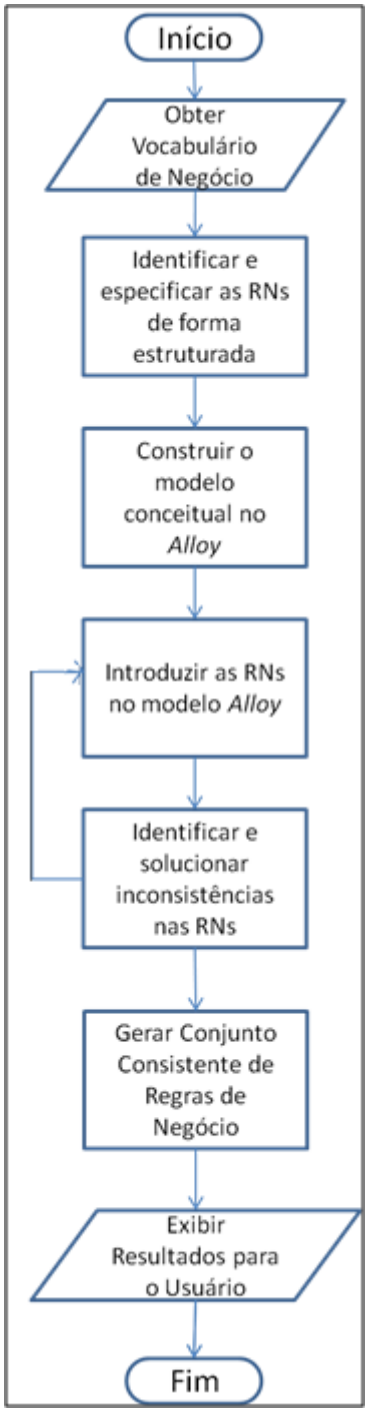



Figura 8: Método MIIRNA

### 4.1.1 ESPECIFICAÇÃO DO VOCABULÁRIO DE NEGÓCIO

Nessa etapa do método MIIRNA, conforme exibido na Figura 9, o vocabulário de negócio deve ser especificado na forma de termos e fatos de negócio e as relações entre eles. Para tal, é recomendável usar um modelo conceitual que represente visualmente o entendimento (criando um novo, caso não exista).

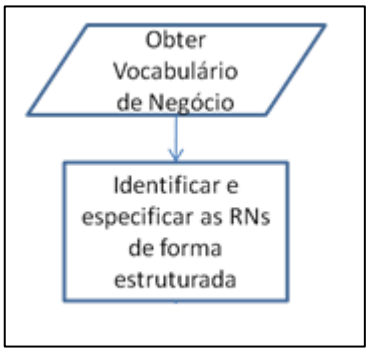



Figura 9: Especificação do vocabulário de negócio no método MIIRNA

Em seguida, as regras de negócio devem ser identificadas e especificadas de forma estruturada, usando linguagem de negócio para a sua representação. MORGAN (2002) propôs algumas categorias de problemas potenciais que podem ocorrer nesse momento e que deveriam sempre ser levadas em consideração, na forma de boas práticas, para que se obtenha a maior qualidade possível na especificação de regras de negócio:

- *Relacionamento entre regras e o modelo de negócio*: buscar garantir que todas as regras estejam relacionadas com o modelo conceitual. A definição de uma regra pode acarretar modificações na modelagem de negócio, incluindo alterações em fatos de negócio, aprofundamento em definições de termos de negócio, explicitar relações e atributos, etc., de forma que o modelo inteiro seja evolutivo;
- *Simplicidade das expressões*: evitar o uso de expressões vagas como "*pode*" ou "*é possível*" e a especificação de regras de negócio muito complexas ou longas. Também se deve ter cuidado com o uso de expressões como "*ou*" e "*e*", de forma a facilitar a manutenção das regras, especialmente se as cláusulas que as formam puderem mudar de forma independente no futuro;

- *Quantificações e qualificações*: verificar a necessidade de qualificações e de termos plurais, buscando manter as regras na sua forma mais básica. Também é recomendável usar expressões como "*cada*" e "*toda*", se isto aumentar a clareza da regra;
- *Estados e eventos*: evitar o uso de eventos de negócio (tais como ações de negócio, horas ou disparadores externos) como sujeitos das regras, assim como estados de negócio e temporais ambíguos, pois podem comprometer a clareza da regra;
- *Atores:* se um ator aparece em uma regra, verificar por que esse ator em particular deve ser especificado, o que pode revelar a necessidade de uma melhor classificação dos atores envolvidos no negócio. Também evitar o uso de atores como sujeitos das regras, para que a lógica de negócio seja expressa de forma independente dos atores;
- *Verbos perigosos*: evitar o uso de comandos verbais, pois podem indicar atividades relacionadas que podem ser mais bem descritas através de processos de negócio, no lugar de regras. Também se deve ter cuidado com regras que usam expressões *CRUD*[13], pois se referem mais a ações do que a restrições e podem representar mais um procedimento do que uma regra propriamente dita;
- *Computação*: evitar o uso de regras de computação ambíguas. Recomenda-se neste caso tornar o resultado da computação o sujeito da regra. Também se devem evitar regras com cálculos de computação embutidos, pois fórmulas e algoritmos devem poder ser alterados de forma independente das restrições especificadas em uma regra;
- *Estrutura e consistência:* todas as referências entre as expressões de regras de negócio e as demais partes do modelo de negócio devem ser completas e consistentes, mesmo que isto seja difícil de ter certeza. O uso de padronização

[13] *Acrônimo para as quatro funções básicas de persistência de dados utilizadas em bancos de dados relacionais:* ***c****reate,* ***r****ead,* ***u****pdate e* ***d****elete.*

de nomenclaturas e de ferramentas para suportar essa atividade é recomendável.

### 4.1.2 CONSTRUÇÃO DO MODELO CONCEITUAL NO *ALLOY*

Nessa etapa do método MIIRNA, conforme exibido na Figura 10, todos os conceitos e as relações do modelo conceitual de negócio devem ser transcritos para o *Alloy* na forma de conjuntos de estruturas, usando os parágrafos *sig*, *extend* e *abstract* e as declarações de campos do *Alloy*, conforme descrito na seção 3.2 desta dissertação.


Figura 10: Construção do modelo conceitual no método MIIRNA

### 4.1.3 INTRODUÇÃO DAS REGRAS DE NEGÓCIO NO *ALLOY*

Nessa etapa do método MIIRNA, conforme exibido na Figura 11, todas as regras de negócio especificadas em linguagem natural devem ser transcritas, uma de cada vez, para o *Alloy*.

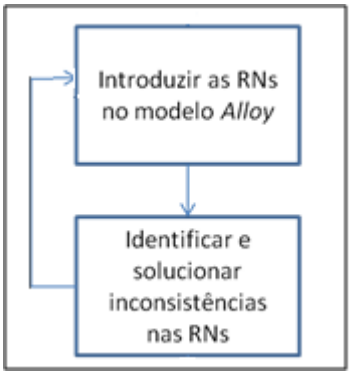



Figura 11: Introdução das regras de negócio no método MIIRNA

Após a introdução de cada regra no modelo gerado no *Alloy*, o método MIIRNA permite a detecção dos seguintes cenários de inconsistência: regras conflitantes e regras redundantes.

A inconsistência pode ocorrer envolvendo mais de duas regras de negócio, pois o *Alloy Analizer* analisa todas as restrições do modelo simultaneamente. Porém, a verificação do método MIIRNA e a subsequente solução da inconsistência ocorrerão usando um par de regras a cada vez, contendo a nova regra introduzida e uma regra já existente no modelo *Alloy*.

Para tal, devem ser seguidos os seguintes passos:

1. Converter cada regra a ser introduzida no modelo em um parágrafo *pred* do *Alloy*;
2. Definir um parágrafo *assert* do *Alloy* para a restrição *pred* usada no passo 1 e executar o comando *check* do *Alloy*, procurando por exemplos inválidos para o conjunto de RN modelado. A quantidade de instâncias a serem verificadas pelo comando *check* do *Alloy* deve seguir os princípios da *small scope hypothesis*, conforme descrito na seção 1.2 desta dissertação.
    a. Se um exemplo inválido for encontrado, então a nova regra possivelmente criou um conflito. A possível regra conflitante deve ser encontrada e o modelo deve ser ajustado, como descrito abaixo:
        i. **Regras conflitantes**: quando o *Alloy Analyzer* indica que o modelo está inconsistente. Neste caso, a nova regra deve ser verificada com toda outra regra possivelmente conflitante no modelo. Isto pode ser feito criando um parágrafo *pred* de teste, contendo a disjunção entre a nova regra e a possível regra conflitante. Um parágrafo *assert* para esse teste deve ser criado e o comando *check* deve ser executado de novo. Se nenhum contraexemplo for encontrado, então o conflito entre as regras está provado e deve ser solucionado;
    b. Se nenhum exemplo inválido for encontrado, então há possibilidade de a nova regra ter criado uma redundância. A possível regra redundante deve ser encontrada e o modelo deve ser ajustado, como descrito abaixo:
        i. **Regras redundantes**: quando o *Alloy Analyzer* indica que o modelo está consistente. Neste caso, a nova regra deve ser verificada com toda outra regra possivelmente redundante no modelo. Isto pode

ser feito criando um parágrafo *pred* de teste, contendo a nova regra e a negação da possível regra redundante. Um parágrafo *assert* para esse teste deve ser criado e o comando *check* deve ser executado de novo. Se um contraexemplo for encontrado, a redundância entre as regras está provada e deve ser solucionada;

3. Após a nova regra ter se tornado consistente e não redundante com o modelo de RN, deve ser convertida em um parágrafo *fact* do *Alloy*;
4. Repetir todo o procedimento desde o passo 1, até que todas as regras tenham sido introduzidas e verificadas no modelo.

### 4.1.4 GERAÇÃO DO CONJUNTO CONSISTENTE DE REGRAS DE NEGÓCIO

Nesta etapa do método MIIRNA, conforme exibido na Figura 12, todas as regras de negócio foram transcritas para o *Alloy* e verificadas quanto à sua consistência, de acordo com a seção 4.1.3 desta dissertação. Nesse momento, todas as regras de negócio introduzidas no modelo estão escritas na forma de parágrafos *fact* do *Alloy.*

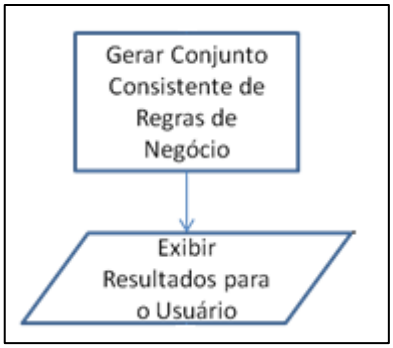



Figura 12: Geração do conjunto consistente de RNs no método MIIRNA

Dessa forma, foi gerado o conjunto consistente de regras de negócio que poderão ser exibidas para os definidores do negócio, para dar suporte às atividades de validação.

A cada atualização das especificações do negócio, o método MIIRNA deve ser executado de novo, de forma que o conjunto consistente de regras de negócio seja sempre evolutivo.

# 5 PRIMEIRO EXEMPLO DE USO

Para fins de ilustração da aplicação do método MIIRNA, primeiramente realizamos um exemplo de uso, baseado no conjunto de regras de negócio de um processo fictício de empréstimo, similar ao apresentado em MORGAN (2002), de forma didática e simples.

A verificação foi executada seguindo o proposto na seção 4.1 desta dissertação, de forma a achar inconsistências no conjunto de regras de negócio.

## 5.1 ESPECIFICAÇÃO DO VOCABULÁRIO DE NEGÓCIO

Neste passo, foram especificados todos os termos, fatos e regras do negócio, conforme exibido nas Tabelas 4, 5 e 6, usando linguagem de negócio para representar o vocabulário do negócio de forma estruturada.

Tabela 4: Termos de negócio do processo fictício

| TERMOS |
|---|
| *Loan* |
| *Manager* |
| *Customer* |
| *Overdrawn Customer* |
| *Underdrawn Customer* |
| *Branch Manager* |
| *Regular Manager* |
| *Approved Loan* |
| *Not Approved Loan* |
| V1000 |
| V1500 |

Tabela 5: Fatos de negócio do processo fictício

| FATOS |
|---|
| *A Customer can be an Underdrawn or an OverDrawn Customer* |
| *A Manager can be a Branch or a Regular Manager* |
| *A Loan can be Approved or Not Approved Loan* |
| *A Value can be V1000 or V1500* |

Tabela 6: Regras de negócio do processo fictício

| REGRAS DE NEGÓCIO |
|---|
| R1: *A loan exceeding $1.000 must be approved by a branch manager or above.* |
| R2: *A loan exceeding $1.000 must be approved by a regular manager.* |
| R3: *A new loan may not be offered to an overdrawn customer.* |
| R4: *An overdrawn customer may not be offered a new loan.* |

## 5.2 CONSTRUÇÃO DO MODELO CONCEITUAL DO DOMÍNIO

O diagrama de classes representa os conceitos de negócio e suas relações. Para este exemplo, o diagrama foi gerado com base no vocabulário de negócio descrito na seção 5.1, usando a ferramenta USE (GOGOLLA; BüTTNER; RICHTERS, 2007).

A Figura 13 mostra o diagrama de classes representando o modelo conceitual do domínio do processo de empréstimo, proposto em MORGAN (2002).

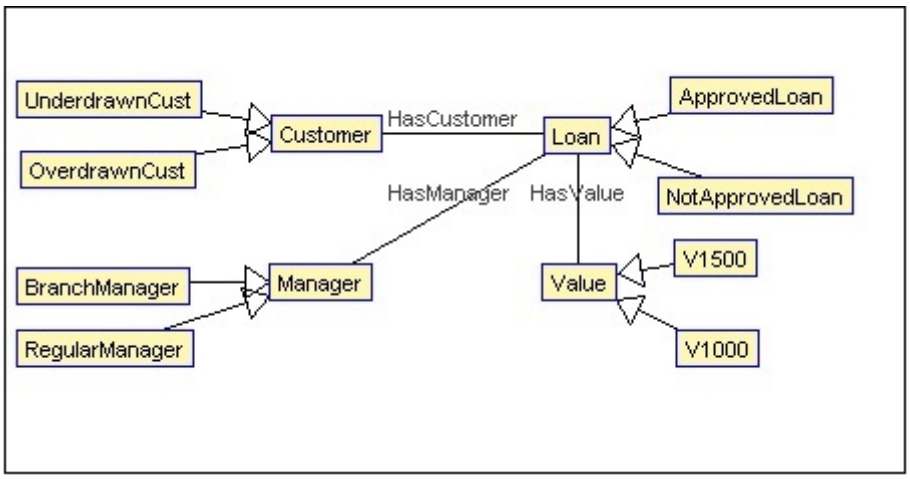



Figura 13: Modelo conceitual do processo fictício

## 5.3 CONSTRUÇÃO DO MODELO *ALLOY*

Neste passo, o modelo *Alloy* foi construído, de forma aderente ao modelo conceitual descrito na Figura 13, representando cada classe e cada relação na forma de parágrafos *sig*, *extend* e *abstract* e de declarações de campos do *Alloy*, conforme descrito na seção 3.2 desta dissertação.

O resultado da construção do modelo *Alloy* completo é mostrado na Figura 14.

```
module loan

abstract sig Customer {}
sig OverdrawnCust extends Customer {}
sig UnderdrawnCust extends Customer {}

abstract sig Loan {v: Value, c:Customer, mg:Manager}
sig ApprovedLoan extends Loan {}
sig NotApprovedLoan  extends Loan {}

abstract sig Value {}
sig V1000 extends Value {}
sig V1500 extends Value {}

abstract sig Manager {}
sig BranchManager extends Manager {}
sig RegularManager extends Manager {}
```

Figura 14: Modelo *Alloy* representando o modelo conceitual do primeiro exemplo de uso

## *5.4* INTRODUÇÃO DAS REGRAS DE NEGÓCIO NO *ALLOY*

Em sequência, as regras descritas na Tabela 6 foram escritas na linguagem *Alloy*. As regras R1 e R3 foram escolhidas para representar o conjunto já consolidado de regras.

Dessa forma, as regras "*A loan exceeding $1.000 must be approved by a branch manager or above*" e "*A new loan may not be offered to an overdrawn customer*" foram escritas no modelo *Alloy*, na forma de parágrafos *fact*, como segue:

*fact* RuleLoan10Approver

{*all* x:Loan | x.v = V1000 *implies* x.mg = BranchManager}

*fact* NoLoanOverdrawn

{*no* z:Loan | z.c = OverdrawnCust}

Em sequência, foi definida uma restrição, apenas para uma simples avaliação estrutural do modelo. Neste exemplo, foi estabelecido que sempre haja um conjunto não vazio de *Loans* no nosso modelo. Essa restrição foi escrita no modelo *Alloy*, como segue:

*pred* SomeLoans

{*some* s:Loan | s *in* ApprovedLoan

*one* s:Loan | s *in* NotApprovedLoan}

Para avaliar o modelo, o comando *run* foi executado. De acordo com a "*small scope hypothesis*" (conforme descrito na seção 1.2 desta dissertação), o escopo da execução limitou-se a três instâncias de todos os objetos do modelo e a duas instâncias do objeto *Value*.

O *Alloy Analyzer* informou que o modelo estava válido, exibindo instâncias que asseguram isto, como mostrado na Figura 15.


Figura 15: O modelo *Alloy* válido e o *snapshot* da execução do *Alloy Analyzer*

## 5.5 GERAÇÃO DOS CENÁRIOS DE INCONSISTÊNCIA

Agora, de acordo com o passo 4 do método, novas regras devem ser introduzidas, representando os cenários de inconsistência que serão avaliados (conflito e redundância). Foram escolhidas as regras R2 e R4 da Tabela 6 para serem introduzidas no conjunto consolidado de regras gerado na seção 5.4 desta dissertação.

### 5.5.1 PRIMEIRO CENÁRIO DE INCONSISTÊNCIA: CONFLITO

O primeiro cenário de inconsistência foi gerado após a inclusão da regra "*A loan exceeding $1.000 must be approved by a regular manager*", descrita na Tabela 6 e escrita no modelo *Alloy* na forma de um parágrafo *pred*, como segue:

```
pred RuleLoanRegularApprover
    { all x:Loan | x.v = V1000 implies x.mg=RegularManager }
```

Essa regra conflita com *"A loan exceeding $1.000 must be approved by a branch manager or above"*, regra já existente no modelo.

De acordo com o passo 4 do método, os comandos *assert* e *check* foram executados. O *Alloy Analyzer* informou que um contraexemplo foi encontrado, mostrando que a assertiva era inválida, como exibido na Figura 16. Logo, a inclusão da nova regra tornou o modelo inconsistente.

Atualizando o modelo, fazendo a disjunção das regras conflitantes (usando o conector *or*) e executando os comandos *assert* e *check* de novo, o *Alloy Analyzer* agora informou nenhum contraexemplo foi encontrado, mostrando que a assertiva era válida e, por conseguinte, a disjunção das regras tornou o modelo consistente, provando o conflito entre as regras, como exibido na Figura 17.

Contudo, essa inconsistência deve ser solucionada com o envolvimento dos definidores do negócio, na atividade "validação de regras" (ROSS, 2009) e na fase *Rule Validation* do ciclo de regras de negócio da ABRD (ECLIPSE FOUNDATION, 2015), ambas executadas essencialmente de forma manual e humana, conforme descrito nas seções 3.3 e 3.4 desta dissertação.

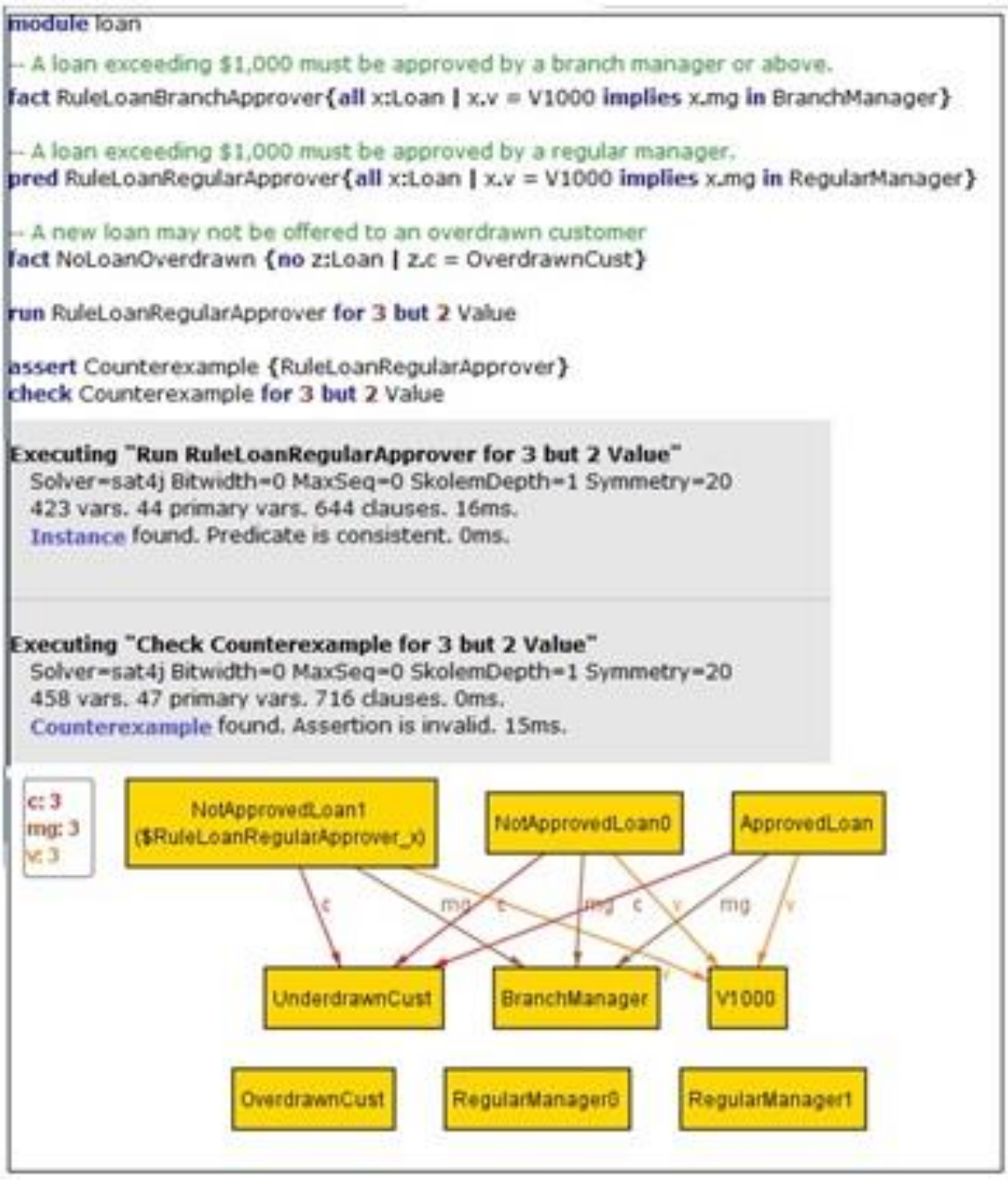



Figura 16: O modelo *Alloy* com o cenário de conflito e o *snapshot* da execução do *Alloy Analyzer*

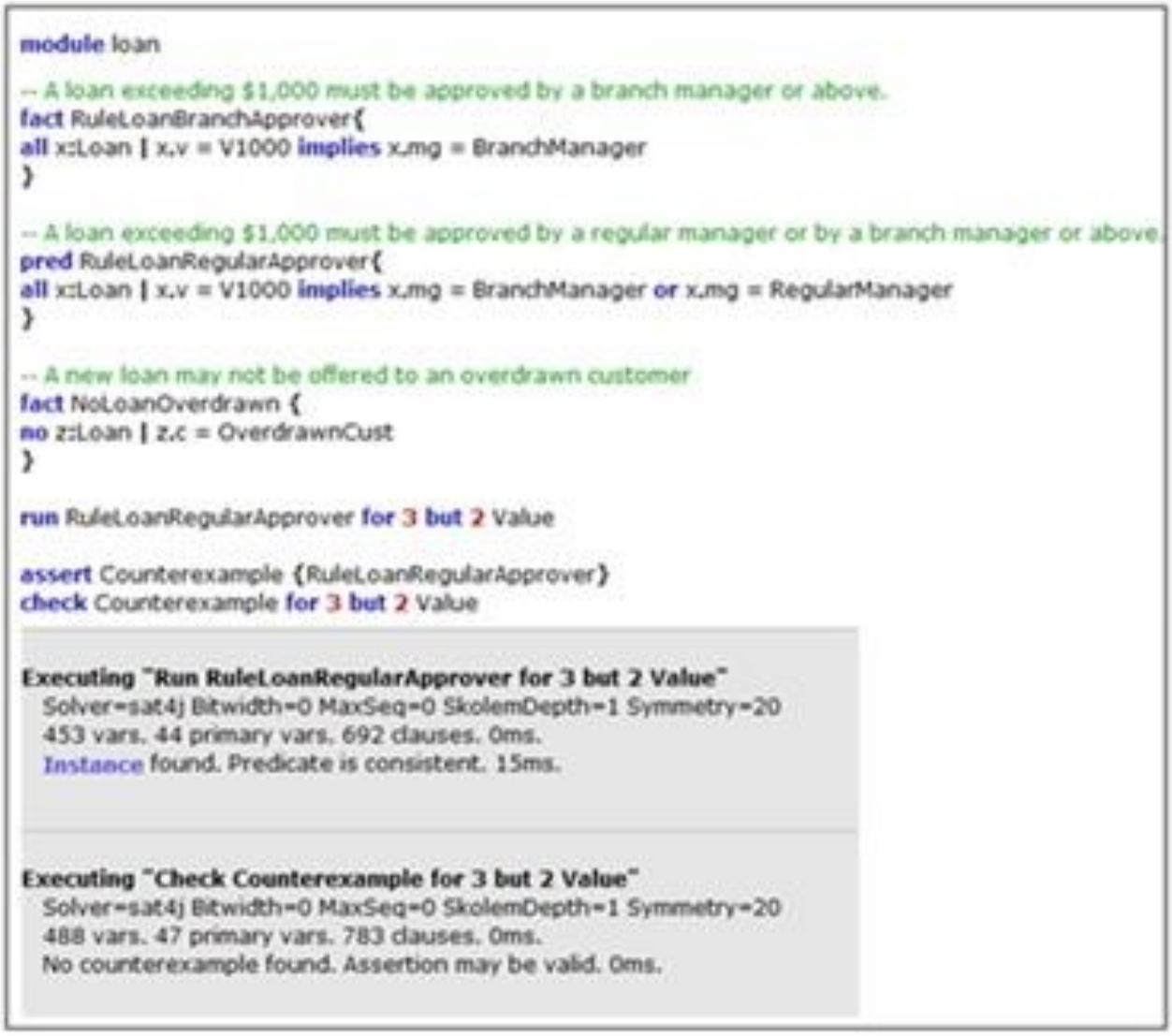



Figura 17: O modelo *Alloy* com a assertiva de prova do cenário de conflito e o *snapshot* da execução do *Alloy Analyzer*.

### 5.5.2 SEGUNDO CENÁRIO DE INCONSISTÊNCIA: REDUNDÂNCIA

O próximo cenário de inconsistência foi gerado após a inclusão da regra "*An overdrawn customer may not be offered a new loan*", descrita na Tabela 6 e escrita no modelo *Alloy* na forma de um parágrafo *pred*, como segue:

*pred* NoLoanOverdrawn02
{ *all* od:OverdrawnCust | #od.l = 0 }

Essa regra é redundante com *"A new loan may not be offered to an overdrawn customer*", regra já existente no modelo.

De acordo com o passo 4 do método, os comandos *assert* e *check* foram executados. O *Alloy Analyzer* informou que nenhum contraexemplo foi encontrado, indicando que a assertiva era válida, como mostrado na Figura 18. Logo, a inclusão da nova regra não afetou os resultados da execução do conjunto original de regras e o modelo ainda estava consistente.

Atualizando o modelo, fazendo a junção da nova regra com a negação da regra original (usando o conector *not*) e executando os comandos *assert* e *check* de novo, o *Alloy Analyzer* informou que um contraexemplo foi encontrado, mostrando que a assertiva era inválida e, por conseguinte, a negação das regras tornou o modelo inconsistente, provando a redundância entre as regras, como exibido na Figura 19.

Esta inconsistência pode ser solucionada pelo próprio analista de negócio, sem necessidade imperiosa de envolvimento dos definidores do negócio, bastando excluir a redundância indicada entre as regras.

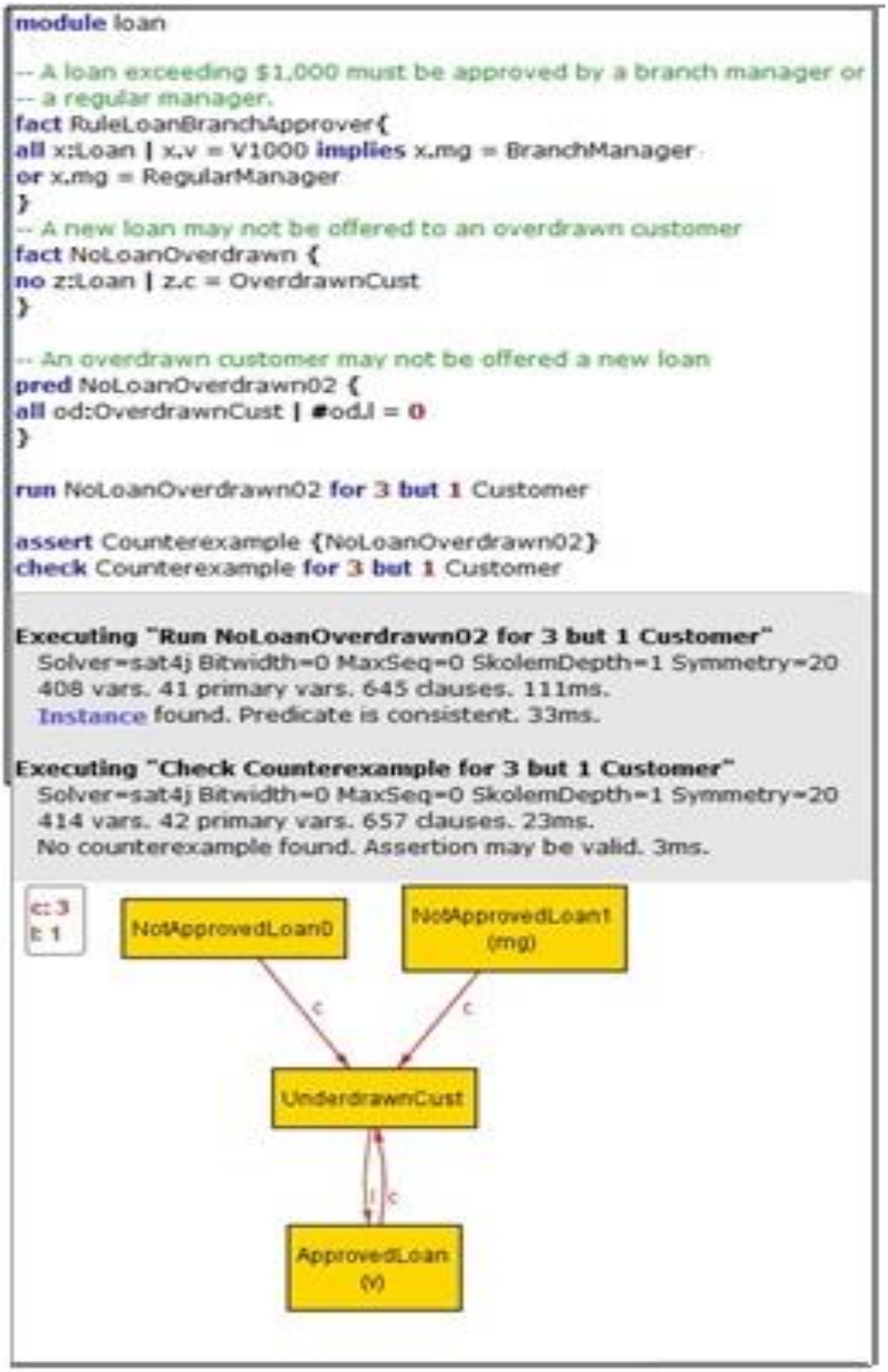



Figura 18: O modelo *Alloy* com o cenário de redundância e o *snapshot* da execução do *Alloy Analyzer*

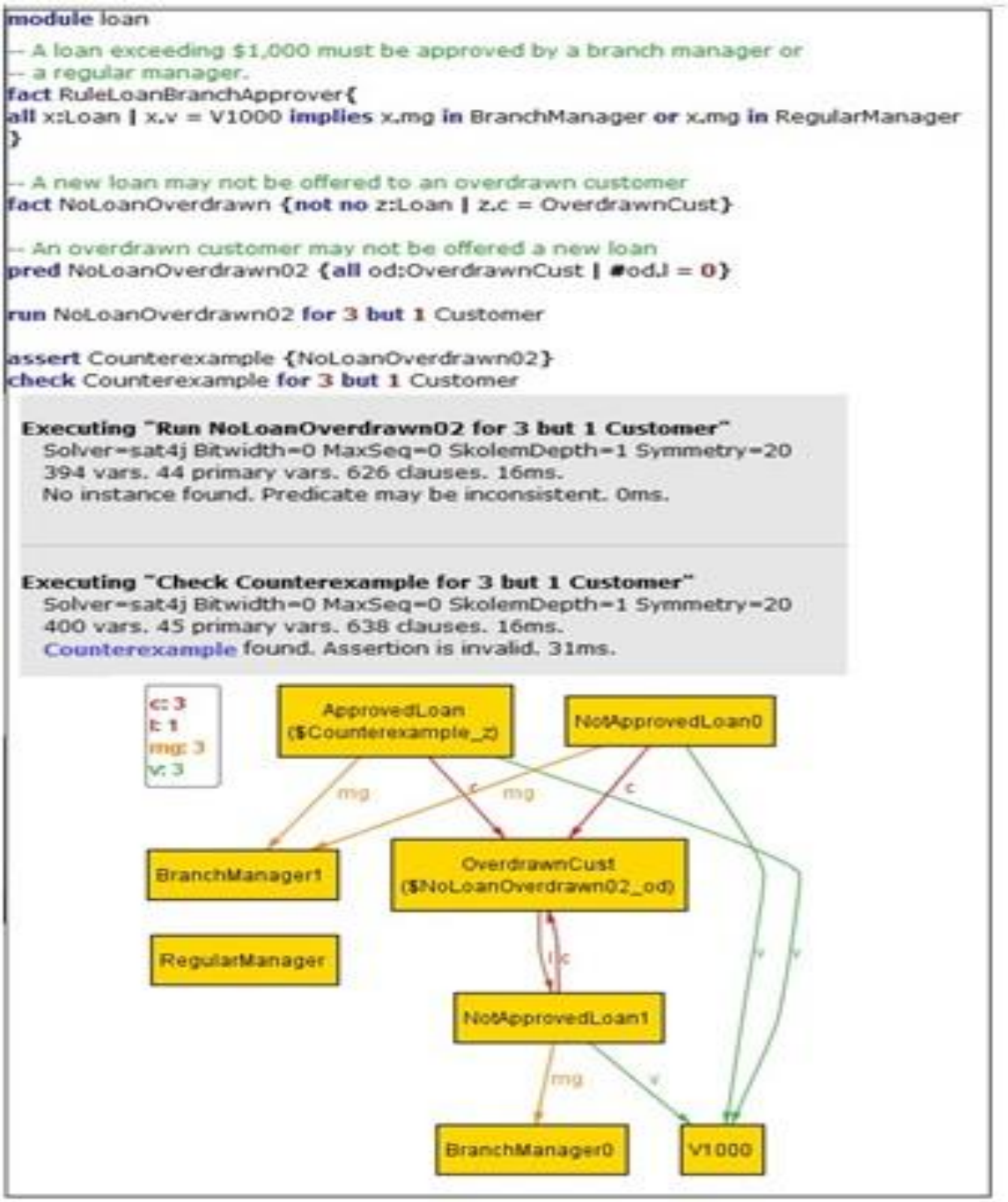



Figura 19: O modelo *Alloy* com a assertiva de prova do cenário de redundância e o snapshot da execução do *Alloy Analyzer*

# 6 SEGUNDO EXEMPLO DE USO

Para demonstrar a aplicabilidade do método MIIRNA em um modelo com nível de complexidade maior, foi realizado um segundo exemplo de uso, aplicado ao conjunto de regras de negócio de uma empresa fictícia de locação de veículos, a *EU-Rent*, descrita em OBJECT MANAGEMENT GROUP (2015).

Nesse caso, a especificação de negócio é bem mais complexa do que a do exemplo descrito na seção 5. A verificação foi executada seguindo o proposto na seção 4.1 desta dissertação, de forma similar à que foi feita no primeiro exemplo de uso.

## 6.1 IDENTIFICAÇÃO DAS REGRAS DE NEGÓCIO

Neste passo, foram especificados todos os termos, fatos e regras de negócio dos domínios de negócio da empresa que se pretende modelar, usando linguagem de negócio para representar o vocabulário do negócio de forma estruturada.

Foram identificados os termos e fatos do vocabulário do negócio, exibidos nas Tabelas 7 e 8:

Tabela 7: Termos de negócio da empresa modelada *EU-Rent*

| TERMOS |
|---|
| Cliente, Motorista |
| Grupo, Modelo, Carro |
| Filial |
| Manutenção |
| Contratempo |
| Reserva |
| Locação |
| Retorno |
| Estação de Serviço |
| Reserva de Crédito |
| Carteira de Habilitação |
| Apólice de Seguro |
| Cartão de Crédito |

Contrato de Locação

Tabela 8: Fatos de negócio da empresa modelada *EU-Rent*

| FATOS |
|---|
| Clientes podem ser pessoas físicas ou jurídicas. |
| Diferentes modelos de carros são oferecidos, organizados em grupos. |
| Todos os carros de um grupo são cobrados pelas mesmas taxas. |
| A empresa tem filiais em vários países. |
| Uma reserva deve especificar o grupo do carro, início/fim da locação e a filial onde o carro será retirado. |
| A EU-Rent registra as más experiências com seus clientes, podendo recusar futuras locações solicitadas pelos mesmos. |

A Tabela 9 mostra as regras de negócio e os respectivos domínios de negócio da empresa fictícia *EU-Rent*:

Tabela 9: Regras de negócio da empresa modelada *EU-Rent*

| REGRAS DE NEGÓCIO | DOMINIO |
|---|---|
| Locação sem reserva depende de haver carros disponíveis. | Locação |
| Carros podem ser devolvidos em filiais diferentes. | Devolução |
| Propriedade será transferida para a filial de retorno. | Devolução |
| Carro pertence a uma filial. | Devolução |
| Apenas uma manutenção agendada por carro por dia. | Manutenção |
| Cliente pode ter somente um carro alugado em um dado tempo. | Clientes |
| Clientes na lista negra não podem fazer reservas. | Clientes |
| Motorista autorizado (MA) deve possuir carteira válida. | Restrições |
| Um carro pode ser alugado através de uma reserva feita com antecedência ou no ato da retirada do carro. | Aceitação |
| Se reserva não especifica modelo, grupo A deve ser escolhido. | Aceitação |
| Reservas só podem ser aceitas até a capacidade da filial no dia. | Aceitação |
| Locações não podem ser feitas para clientes da lista negra. | Aceitação |
| Cliente só pode estar de posse de um carro num dia. | Aceitação |
| Dia da devolução deve ser anterior a qualquer reserva ou manutenção. | Aluguel |

| | |
|---|---|
| Caso exista mais de um carro do mesmo grupo, escolher o de menor km. | Aluguel |
| Motorista autorizado deve ter idade >25 e licença há mais de um ano. | Entrega |
| Cartão de crédito de garantia deve pertencer ao motorista autorizado. | Entrega |
| Antes da liberação do carro, deve ser feita uma reserva de crédito igual ao valor da locação. | Entrega |
| Cliente paga ao retornar o carro. | Retorno |

## 6.2 CONSTRUÇÃO DO MODELO CONCEITUAL DO DOMÍNIO

Com base no vocabulário de negócio descrito na seção 6.1, foi gerado o diagrama de classes, usando mais uma vez a ferramenta USE (GOGOLLA; BüTTNER; RICHTERS, 2007).

A Figura 20 mostra o diagrama de classe representando o modelo conceitual da empresa fictícia EU-Rent, descrita em OBJECT MANAGEMENT GROUP (2015).

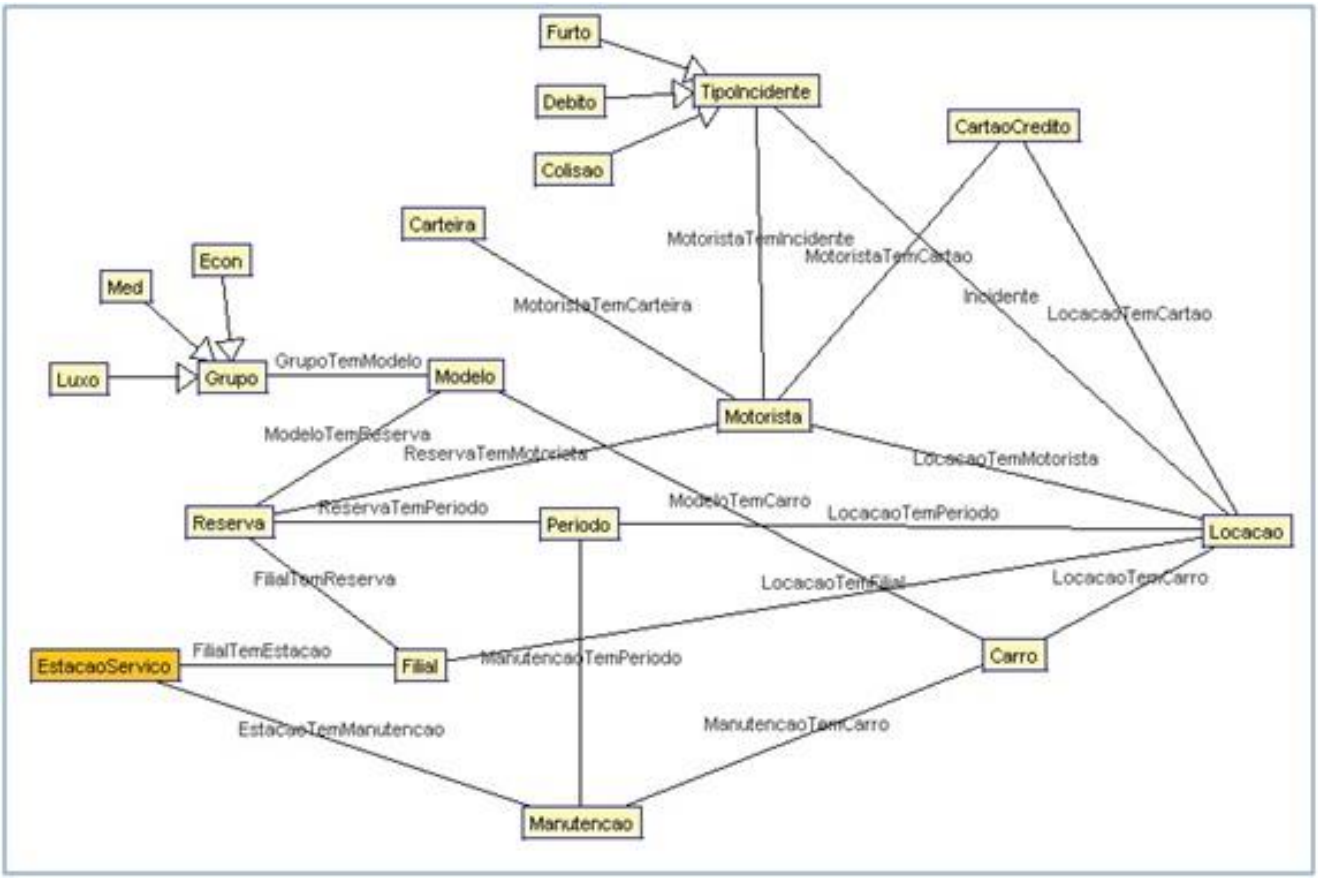



Figura 20: O modelo conceitual da empresa fictícia *EU-Rent*

## 6.3 CONSTRUÇÃO DO MODELO *ALLOY*

Com base no modelo conceitual descrito na Figura 20, o modelo *Alloy* foi construído, representando os conceitos e as relações entre eles (por exemplo, os conceitos Motorista, Reserva, Locação, Carteira e CartaoCredito e as relações ReservaTemMotorista, LocacaoTemMotorista, MotoristaTemCarteira e MotoristaTemCartao) na forma de parágrafos sig, *extend* e *abstract* e de declarações de campo do *Alloy*, conforme descrito na seção 3.2 desta dissertação.

O modelo *Alloy* completo representando o modelo conceitual da empresa EU-Rent é mostrado na Figura 21.

## 6.4 INTRODUÇÃO DAS REGRAS DE NEGÓCIO NO *ALLOY*

Neste passo, as regras descritas na Tabela 9 foram escritas na linguagem *Alloy*, de modo a gerar o conjunto consistente de regras de negócio. Por exemplo, a regra "Clientes na lista negra não podem fazer reservas" foi escrita no modelo *Alloy*, na forma de um parágrafo *fact*, como segue:

```
fact ClienteOKPodeReservar
    {all r:Reserva| not r.cliente in MotoristasComIncidentes}
```

Em seguida, foi definida uma restrição, para uma simples avaliação estrutural do modelo. Neste exemplo, foi estabelecido que para toda locação sempre haveria um cartão de crédito dado como garantia pelo motorista no nosso modelo. Essa restrição foi escrita no modelo *Alloy*, como segue:

```
pred PreCondGarantia
    { all l: Locacao | l.ccgarantia in l.cliente.cc }
```

Para avaliar o modelo, o comando *run* foi executado. De acordo com a *small scope hypothesis* (conforme descrito na seção 1.2 desta dissertação), o escopo da execução se limitou a três instâncias de todos os objetos do modelo e a exatamente uma instância do objeto *Locação*.

Após a execução, o *Alloy Analyzer* exibiu instâncias que exemplificam o modelo, informando que a restrição é válida. Executando os comandos *assert* e *check*, o *Alloy Analyzer*

não encontra nenhum contraexemplo, informando que não há exemplos que tornem a restrição inválida. O modelo, portanto, está consistente, como mostrado na Figura 22.

```
module VcAluga

sig Motorista{
   reserva: set Reserva,
   locacao: set Locacao,
   cm: one Carteira,
   cc:CartaoCredito,
   idade: one Int
}

sig Carteira{
   validade: Int,
   AnosCorridos: Int,
   dono: one Motorista
}

sig CartaoCredito{
   lim:Int,
   dono: one Motorista
}

sig Periodo{
   inicio: one Int,
   fim: one Int
}

{inicio>0
fim>inicio
}

sig Filial {
   estacao: some EstacaoServico,
   reservaRetorno: set Reserva,
   reservaRetirada: set Reserva,
   inventario: set Carro
}

sig EstacaoServico{
   manutencao: set Manutencao,
   serve: Filial
}

sig Manutencao{
   estacao: one EstacaoServico,
   carro: one Carro,
   periodo: one Periodo
}

sig Carro{
   modelo: one Modelo
}

sig Reserva{
   cliente: one Motorista,
   filialRetirada: one Filial,
   filialRetorno: one Filial,
   modelo: one Modelo,
   periodo: one Periodo
}
sig Locacao{
   cliente: one Motorista,
   filialRetirada: one Filial,
   filialRetorno: one Filial,
   carro: one Carro,
   periodo: one Periodo,
   ccgarantia: one CartaoCredito
}
abstract sig TipoIncidente{}
one sig Colisao,Furto,Debito extends TipoIncidente{}

one sig Incidente {
f: Locacao -> some TipoIncidente
}

fun MotoristasComIncidentes[]:Motorista {
((Incidente.f).TipoIncidente).cliente
}

abstract sig Grupo{
   modelo: some Modelo
}
one sig Econ, Med,Luxo extends Grupo{}

sig Modelo{
   grupo: one Grupo,
   carro: some Carro
}
```

Figura 21: Modelo *Alloy* representando o modelo conceitual da empresa fictícia *EU-Rent*

```
}
fact PreLocacaoSReserva {PreLocacaoSReserva[Motorista, Filial, Filial, Carro, Periodo, CartaoCredito]}

--Apenas uma manutenção agendada por carro por dia
pred UnicaMatutencaoDia{
all m1, m2:Manutencao| ( (m1.carro = m2.carro) and (m1 != m2)) implies naoSobreposto[m1.periodo, r
}
fact UnicaMatutencaoDia {UnicaMatutencaoDia}

--Todo cartão de crédito dado como garantia deve pertencer ao locador
pred LocacaoCC {
all lc:Locacao | (lc.ccgarantia).dono = lc.cliente
}
fact LocacaoCC {LocacaoCC}

-- FIM DAS REGRAS JÁ CONSOLIDADAS

---- Regras sendo avaliadas

--Clientes na lista negra não podem fazer reservas
fact ClienteOKPodeReservar{
all r:Reserva| not r.cliente in MotoristasComIncidentes
}


--Toda locação deve ter um cartão de crédito como garantia
pred PreCondGarantia{
   all l: Locacao | l.ccgarantia in l.cliente.cc
}

run PreCondGarantia for 3 but exactly 1 Locacao
assert teste {PreCondGarantia}
check teste for 3 but exactly 1 Locacao
```

```
Executing "Run PreCondGarantia for 3 but exactly 1 Locacao"
   Solver=sat4j Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20
   8167 vars. 552 primary vars. 19225 clauses. 802ms.
   Instance found. Predicate is consistent. 372ms.

Executing "Check teste for 3 but exactly 1 Locacao"
   Solver=sat4j Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20
   8175 vars. 553 primary vars. 19234 clauses. 196ms.
   No counterexample found. Assertion may be valid. 49ms.
```

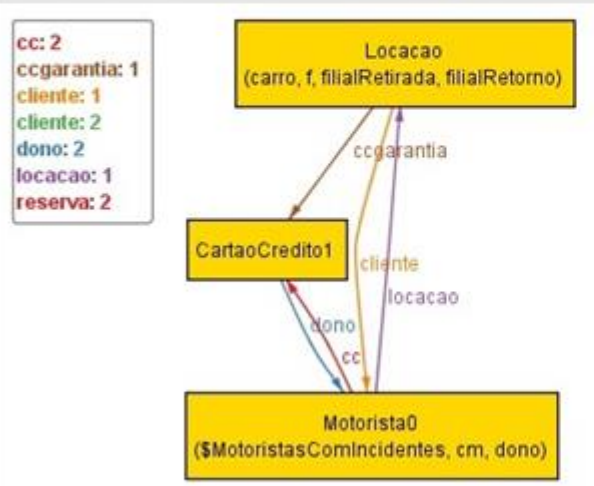



Figura 22: O modelo *Alloy* consistente e o *snapshot* da execução do *Alloy Analyzer*

## 6.5 GERAÇÃO DOS CENÁRIOS DE INCONSISTÊNCIA

Agora, de acordo com o passo 4 do método MIIRNA, novas regras devem ser introduzidas, representando os cenários de inconsistência que foram avaliados (conflito e redundância).

### 6.5.1 PRIMEIRO CENÁRIO DE INCONSISTÊNCIA: CONFLITO

Vamos supor que tenha sido especificada pelo negócio a seguinte regra: “Podem existir reservas feitas por um cliente que esteja na lista negra”. A sua inclusão no modelo vai gerar o primeiro cenário de inconsistência, no caso, conflito com a regra “Clientes na lista negra não podem fazer reservas”, já existente no modelo consistente de regras de negócio.

A regra “Clientes na lista negra não podem fazer reservas”, descrita na seção 6.4, foi reescrita no *Alloy*, para facilitar a aplicação do método, porém continuando na forma de um parágrafo *fact*, conforme mostrado a seguir:

```
pred MotoristaListaNegra [c:Motorista]
    {c in MotoristasComIncidentes}

pred ClienteOKPodeReservar
    {all r:Reserva | not MotoristaListaNegra[r.cliente]}

fact ClienteOKPodeReservar
    {ClienteOKPodeReservar}
```

Em seguida, a nova regra “Podem existir reservas feitas por um cliente que esteja na lista negra.” foi inserida no modelo *Alloy*, sendo escrita na forma de um parágrafo *pred*, como segue:

```
pred QualquerClientePodeReservar
    {some rs:Reserva| MotoristaListaNegra[rs.cliente]}
```

De acordo com o passo 4 do método, os comandos *assert* e *check* foram executados. De acordo com a *small scope hypothesis* (descrita na seção 1.2 desta dissertação), o escopo da execução limitou-se a três instâncias de todos os objetos do modelo e a exatamente uma instância do objeto *Motorista*.

O *Alloy Analyzer* informou que um contraexemplo foi encontrado, mostrando que a assertiva era inválida, como exibido na Figura 23. Logo, a inclusão da nova regra tornou o modelo inconsistente.

```
                and (all l: Locacao | l.periodo = p implies l.carro != c)
}
--Locacoes nao podem ser feitas para clientes da lista negras
pred MotoristaListaNegra[c:Motorista]{
c in MotoristasComIncidentes
}
fact PreLocacaoSReserva {PreLocacaoSReserva[Motorista, Filial, Filial, Carro, Periodo, CartaoCredito]}

--Apenas uma manutenção agendada por carro por dia
pred UnicaMatutencaoDia{
all m1, m2:Manutencao| ( (m1.carro = m2.carro) and (m1 != m2)) implies naoSobreposto[m1.periodo,
}
fact UnicaMatutencaoDia {UnicaMatutencaoDia}

-- FIM DAS REGRAS JÁ CONSOLIDADAS

---- Regras sendo avaliadas

--Clientes na lista negra não podem fazer reservas
pred ClienteOKPodeReservar {
all r:Reserva| not MotoristaListaNegra[r.cliente]
}
fact ClienteOKPodeReservar{ClienteOKPodeReservar}

--Mesmo clientes na lista negra podem fazer reservas
pred QualquerClientePodeReservar{
some rs:Reserva|  MotoristaListaNegra[rs.cliente]
}

run QualquerClientePodeReservar for 3 but exactly 1 Motorista
assert Conflito {QualquerClientePodeReservar}
check Conflito for 3 but exactly 1 Motorista
```

```
Executing "Run QualquerClientePodeReservar for 3 but exactly 1 Motorista"
  Solver=sat4j Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20
  8524 vars. 523 primary vars. 20481 clauses. 207ms.
  No instance found. Predicate may be inconsistent. 25ms.

Executing "Check Conflito for 3 but exactly 1 Motorista"
  Solver=sat4j Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20
  8501 vars. 520 primary vars. 20440 clauses. 59ms.
  Counterexample found. Assertion is invalid. 98ms.
```

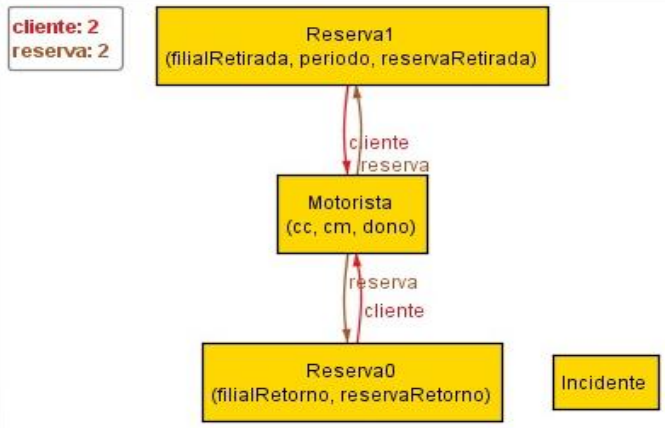



Figura 23: O modelo *Alloy* com o cenário de conflito e o *snapshot* da execução do *Alloy Analyzer*

Atualizando o modelo, fazendo a disjunção das possíveis regras conflitantes (usando o conector *or*) e executando os comandos *assert* e *check* de novo, o *Alloy Analyzer* informou que nenhum contraexemplo foi encontrado, mostrando que a assertiva era válida e, por conseguinte, a disjunção das regras havia tornado o modelo consistente, provando o conflito entre as regras, como exibido na Figura 24.

```
c in MotoristasComIncidentes
}
fact PreLocacaoSReserva {PreLocacaoSReserva[Motorista, Filial, Filial, Carro, Periodo, CartaoCredito]}

--Apenas uma manutenção agendada por carro por dia
pred UnicaMatutencaoDia{
all m1, m2:Manutencao| ( (m1.carro = m2.carro) and (m1 != m2)) implies naoSobreposto[m1.periodo,
}
fact UnicaMatutencaoDia {UnicaMatutencaoDia}

-- FIM DAS REGRAS JÁ CONSOLIDADAS

---- Regras sendo avaliadas

--Clientes na lista negra não podem fazer reservas
pred ClienteOKPodeReservar {
all r:Reserva| not MotoristaListaNegra[r.cliente]
}
fact ClienteOKPodeReservar{ClienteOKPodeReservar}

--Mesmo clientes na lista negra podem fazer reservas
pred QualquerClientePodeReservar{
some rs:Reserva| MotoristaListaNegra[rs.cliente]
}

--Clientes na lista negra podem fazer reservas OU não podem fazer reserva
pred testeconflito {
QualquerClientePodeReservar or ClienteOKPodeReservar
}

run testeconflito for 3 but exactly 1 Motorista
assert Conflito {testeconflito}
check Conflito for 3 but exactly 1 Motorista

Executing "Run testeconflito for 3 but exactly 1 Motorista"
  Solver=sat4j Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20
  8502 vars. 520 primary vars. 20452 clauses. 196ms.
  Instance found. Predicate is consistent. 97ms.

Executing "Check Conflito for 3 but exactly 1 Motorista"
  Solver=sat4j Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20
  8529 vars. 523 primary vars. 20489 clauses. 52ms.
  No counterexample found. Assertion may be valid. 15ms.
```


Figura 24: O modelo *Alloy* com o cenário com a assertiva de prova do conflito e o *snapshot* da execução do *Alloy Analyzer*

Contudo, essa inconsistência deve ser solucionada com o envolvimento dos definidores do negócio, na atividade “validação de regras” (ROSS, 2009) e na fase *Rule Validation* do ciclo de regras de negócio da ABRD (ECLIPSE FOUNDATION, 2015), ambas executadas essencialmente de forma manual e humana, conforme descrito nas seções 3.3 e 3.4 desta dissertação.

### 6.5.2 SEGUNDO CENÁRIO DE INCONSISTÊNCIA: REDUNDÂNCIA

Vamos supor que tenha sido especificada pelo negócio a seguinte regra: “Não pode haver reservas feitas por clientes na lista negra.”. A sua inclusão no modelo vai gerar o segundo cenário de inconsistência, no caso redundância com “Clientes na lista negra não podem fazer reservas”, regra já existente no modelo consistente de regras de negócio.

Dessa forma, a nova regra “Não pode haver reservas feitas por clientes na lista negra” foi inserida no modelo *Alloy*, sendo escrita na forma de um parágrafo *pred*, como segue:

```
pred ReservaSomenteClienteOK
    { all mot: Motorista, rs:Reserva | rs.cliente = mot implies
        (not MotoristaListaNegra[mot]) }
```

De acordo com o passo 4 do método, os comandos *assert* e *check* foram executados. De acordo com a *small scope hypothesis* (descrita na seção 1.2 desta dissertação), o escopo da execução limitou-se a três instâncias de todos os objetos do modelo e a exatamente uma instância do objeto *Motorista*.

O *Alloy Analyzer* informou que nenhum contraexemplo foi encontrado, indicando que a assertiva era válida, como mostrado na Figura 25. Logo, a inclusão da nova regra não afetou os resultados da execução do conjunto original de regras e o modelo ainda estava consistente.

```
                    and (all l: Locacao | l.periodo = p implies l.carro != c)
}
--regra16
--Locacoes nao podem ser feitas para clientes da lista negras
pred MotoristaListaNegra[c:Motorista]{
c in MotoristasComIncidentes
}
fact PreLocacaoSReserva {PreLocacaoSReserva[Motorista, Filial, Filial, Carro, Periodo, CartaoCredito]}

--Apenas uma manutenção agendada por carro por dia
pred UnicaMatutencaoDia{
all m1, m2:Manutencao| ( (m1.carro = m2.carro) and (m1 != m2)) implies naoSobreposto[m1.periodo, r
}
fact UnicaMatutencaoDia {UnicaMatutencaoDia}

-- FIM DAS REGRAS JÁ CONSOLIDADAS

---- Regras sendo avaliadas

--Clientes na lista negra não podem fazer reservas
pred ClienteOKPodeReservar {
all r:Reserva| not MotoristaListaNegra[r.cliente]
}
fact ClienteOKPodeReservar{ClienteOKPodeReservar}

--Não pode haver reservas feitas por clientes na lista negra
pred ReservaSomenteClienteOK {
all mot:Motorista, rs:Reserva | rs.cliente = mot implies (not MotoristaListaNegra[mot])
}

run ReservaSomenteClienteOK  for 3 but exactly 1 Motorista
assert Redundante {ReservaSomenteClienteOK }
check Redundante for 3 but exactly 1 Motorista
```

```
Executing "Run ReservaSomenteClienteOK for 3 but exactly 1 Motorista"
  Solver=sat4j Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20
  8504 vars. 520 primary vars. 20443 clauses. 195ms.
  Instance found. Predicate is consistent. 132ms.

Executing "Check Redundante for 3 but exactly 1 Motorista"
  Solver=sat4j Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20
  8529 vars. 524 primary vars. 20495 clauses. 54ms.
  No counterexample found. Assertion may be valid. 12ms.
```


Figura 25: O modelo *Alloy* com o cenário de redundância e o *snapshot* da execução do *Alloy Analyzer*

Atualizando o modelo, fazendo a junção da nova regra com a negação da regra original (usando o conector *not*) e executando os comandos *assert* e *check* de novo, o *Alloy Analyzer*

informou que um contraexemplo foi encontrado, mostrando que a assertiva era inválida e, por conseguinte, a negação da regra original havia tornado o modelo inconsistente, provando a redundância entre as regras, como exibido na Figura 26.

Essa inconsistência pode ser solucionada pelo próprio analista de negócio, sem necessidade imperiosa de envolvimento dos definidores do negócio, bastando excluir a redundância indicada entre as regras.

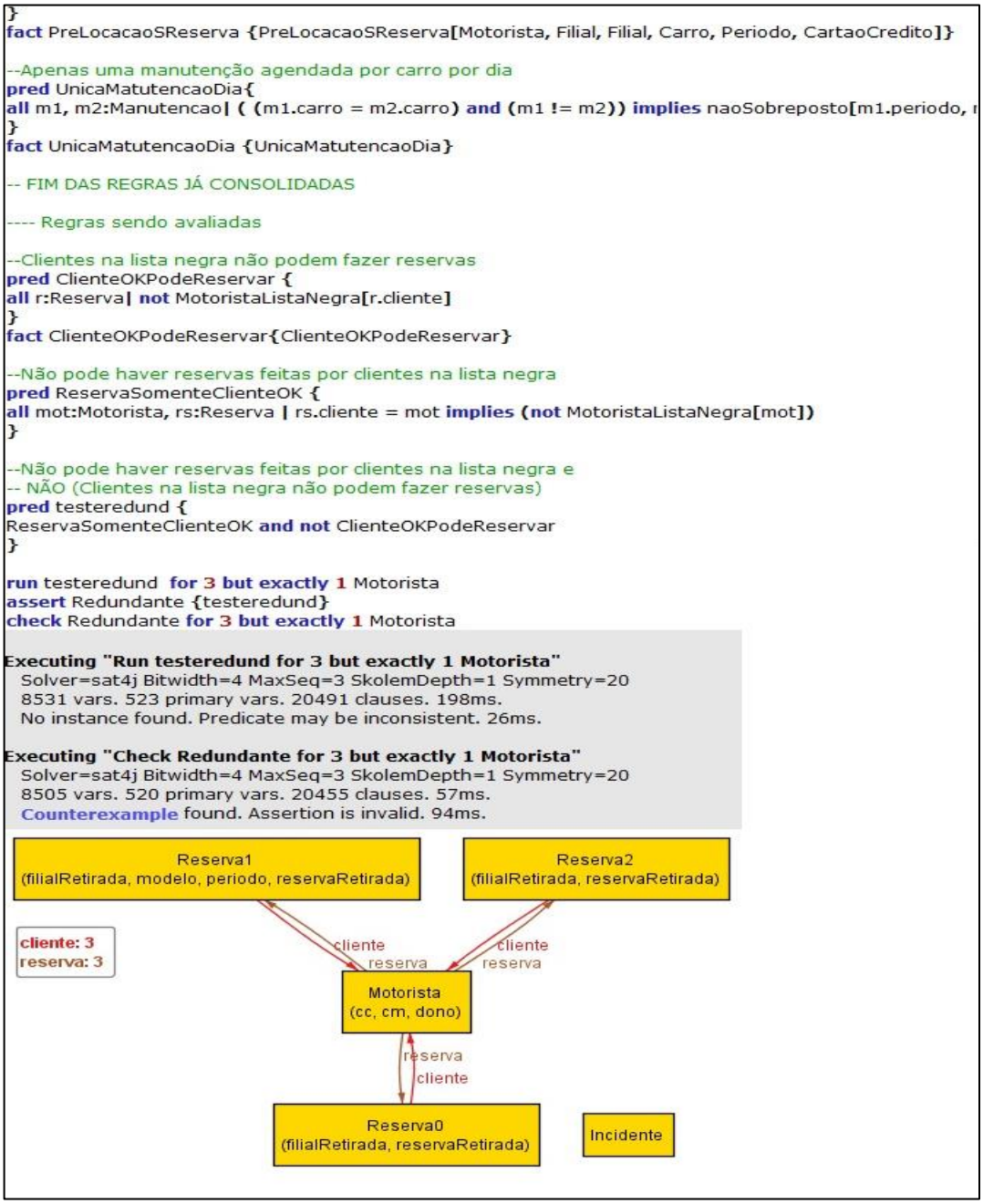



Figura 26: O modelo *Alloy* com a assertiva de prova do cenário de redundância e o *snapshot* da execução do *Alloy Analyzer*

# 7 ANÁLISE DAS EXECUÇÕES

As execuções dos dois exemplos de uso do método MIIRNA foram comparadas, quanto às variáveis de execução do *Alloy Analizer*, possibilitando uma análise da eficiência da aplicação do método, em relação à possível escalabilidade dos modelos de regras de negócio que serão submetidos a ele.

A Tabela 10 lista os resultados das execuções do *Alloy Analizer* para os dois exemplos, com o tempo gasto e a quantidade de variáveis, variáveis primárias e cláusulas checadas para cada execução, de acordo com os cenários testados.

Tabela 10: Resultados da execução do *Alloy Analizer* para os exemplos

| CENÁRIOS | EXEMPLO 1 | | | | EXEMPLO 2 | | | |
|---|---|---|---|---|---|---|---|---|
| | Qtd. de variáveis | Qtd. de variáveis primárias | Qtd. de cláusulas | Tempo gasto pelo SAT Solver | Qtd. de variáveis | Qtd. de variáveis primárias | Qtd. de cláusulas | Tempo gasto pelo SAT Solver |
| Modelo consistente de regras de negócio | 470 | 49 | 704 | 62ms + 16ms | 8175 | 553 | 19234 | 196ms + 49ms |
| Conflito | 458 | 47 | 716 | 0ms + 15ms | 8501 | 520 | 20440 | 59ms + 98ms |
| Conflito Resolvido | 488 | 47 | 783 | 0ms + 0ms | 8529 | 523 | 20489 | 52ms + 15ms |
| Redundância | 414 | 42 | 657 | 23ms + 3ms | 8529 | 524 | 20495 | 54ms + 12ms |
| Redundância Resolvida | 400 | 45 | 638 | 16ms + 31ms | 8505 | 520 | 20455 | 57ms + 94ms |

Ao executar os comandos *run* e *check*, o *Alloy Analizer* procura por uma instância da análise de uma restrição, ou seja, uma atribuição de valores às variáveis dessa restrição para os quais ela se torna verdadeira (JACKSON, 2006), conforme descrito na seção 3.2 desta dissertação.

As variáveis da restrição são atribuídas pelo *Alloy Analizer* de acordo com o tipo de estrutura que foi declarada no modelo *Alloy*, conforme descrito na seção 3.2 desta dissertação:

- Sets associados às declarações de assinaturas;
- Relações associadas às declarações de campos;
- Argumentos dos predicados.

No caso do comando *run*, a análise é feita pela conjunção das variáveis do predicado com os fatos do modelo (declarados explicitamente através de parágrafos *fact* do *Alloy* e implicitamente através das declarações de campos do modelo *Alloy*). Para o comando *check*, a análise é feita pela negação da assertiva conjugada com os fatos do modelo (JACKSON, 2006).

Uma instância de um comando *run* é um exemplo, em que tanto os fatos do modelo, quanto a restrição acontecem. Uma instância de um comando *check* é um contraexemplo, em que os fatos do modelo acontecem, mas a assertiva não (JACKSON, 2006).

O *Alloy Analizer* basicamente é um *constraint solver* para a lógica *Alloy*, que traduz a restrição em uma fórmula booleana e a resolve usando um *SAT Solver*[14] externo. Na execução do *Alloy Analizer*, é indicada a quantidade de cláusulas lógicas embutidas nessa fórmula booleana.

O tempo gasto pelo *SAT Solver* é dividido em duas etapas. A primeira representa o tempo que é necessário para geração do modelo em uma fórmula booleana na forma CNF[15]. A segunda representa o tempo que é necessário para achar a instância que satisfaz a restrição.

Pelos valores indicados na execução do *Alloy Analizer* para os cinco cenários avaliados nos dois exemplos, pode-se verificar que o segundo modelo *Alloy* é bem mais complexo do que o primeiro, dadas as quantidades de variáveis e cláusulas geradas pela ferramenta e o tempo gasto pelo *SAT Solver* para encontrar as instâncias das restrições avaliadas.

Contudo, percebe-se que mesmo com o aumento considerável da quantidade de cláusulas lógicas geradas, o tempo gasto pelo *SAT Solver* no segundo exemplo ainda ficou

---

[14] *Resolvedor de problemas de satisfatibilidade.*

[15] *Conjunctive normal form.*

bastante reduzido, na casa dos milissegundos. Essa otimização da execução sugere a efetividade do uso da *small scope hypothesis* no método MIIRNA.

É interessante ressaltar também que para os dois exemplos o tempo gasto pelo *SAT Solver* para geração do modelo na forma CNF se reduz bastante após o modelo inicial já estar elaborado. Ou seja, o *Alloy Analizer* otimiza a execução do *SAT Solver* a partir da segunda análise do mesmo modelo, o que sugere a evidência da eficiência computacional da utilização do método MIIRNA, mesmo em modelos de complexidade estrutural e tamanho maiores.

Nessa análise das execuções dos dois exemplos de uso do método MIIRNA, não foi medido o esforço humano envolvido (elaboração do vocabulário de negócio, criação do modelo Alloy, escrita das regras de negócio na linguagem Alloy, verificação/correção dos cenários de inconsistência, etc.), tendo em vista que apenas o próprio autor desta dissertação foi objeto da pesquisa, e, assim, a análise não teria valor acadêmico.

Assim, em relação à questão de pesquisa descrita na seção 1.4 desta dissertação, a pesquisa descrita nesta dissertação envolveu apenas a avaliação do esforço computacional do método MIIRNA. A avaliação do esforço humano será parte de um futuro experimento, conforme descrito na seção 9.4 desta dissertação.

# 8 TRABALHOS RELACIONADOS

## 8.1 ARTIGOS E ESTUDOS

EVA Project (CHANG; COMBS; STACHOWITZ, 1990) é uma referência na verificação e validação de regras de negócio. O uso de metaconhecimento (i.e., conhecimento sobre o conhecimento) descreve relações usadas para validar redundância, consistência, completude e correção de uma base de conhecimento. Além de ser muito completo (com 28 critérios diferentes), esse projeto foi estendido para lógica não-monotônica, trazendo-o para mais perto do mundo real. O resultado do processamento é consequentemente complexo. O EVA é de difícil utilização e voltado para usuários com conhecimento avançado em lógica, embora tenha sido estendido para a lógica não-monotônica. O método MIIRNA proposto neste artigo explora a utilização de um *model finder*, cujos resultados são baseados em exemplos e contraexemplos intuitivos.

PREPARE (ZHANG; NGUYEN, 1994) usa redes de predicados/transições para representar conhecimento. Essas redes são classes especiais de redes *Petri* e a verificação é feita através de reconhecimento de padrões sintáticos. Essas redes são uma representação gráfica da lógica de predicados. As redes *Petri* trabalham com um conjunto de posições e transições, que em alguns casos podem ser limitadas a um número máximo. A ferramenta *Alloy* e a *small scope hypothesis,* usadas no método MIIRNA, permitem aumentar ou diminuir o número de instâncias avaliadas, de acordo com o nível de detalhe da avaliação de consistência que se deseja executar.

NuSMV2 (CIMATTI et al., 2002) é uma ferramenta para a checagem de modelos, que estende a linguagem SMV[16] original da Carnegie Mellon University. Com um algoritmo BDD[17] muito eficiente, foi usada com sucesso na verificação de *hardware* e *software*. Infelizmente, sua linguagem de programação não suporta tipos estruturais mais complexos (como relacionais, por exemplo). Isto força o desenvolvedor a codificar relações usando vetores de tipos primitivos. A ferramenta *Alloy* oferece ao desenvolvedor tipos relacionais, aritméticos e lógicos.

---

[16] *Symbolic Model Verifier.*

[17] *Binary Decision Diagram.*

NIET (HICKS, 2007) é um estudo particularmente crítico quanto ao uso de máquinas de inferência. Ele reduz a significância dessas máquinas para abordar principalmente o ordenamento das regras de negócio. Utiliza uma IDE[18] como repositório para a verificação, validação e geração de código em diferentes linguagens de implementação de regras de negócio. Para o NIET, a necessidade de ordenamento é transformada em uma questão primordial, o que inviabiliza o seu uso prático. A estratégia adotada no método MIIRNA não necessita usar nenhuma IDE e questões de ordenamento das regras não têm nenhuma importância no seu resultado.

CPKSA (HUANG; LIN, 2007) é um estudo que advoga o uso de decisão em grupo, com um algoritmo que tem um fator de confiança. Só é apropriado em situações de conflitos em regras de negócio que não podem ser resolvidos através da atividade de "validação de regras". No método MIIRNA, a *small scope hypothesis* permite aumentar ou diminuir o número de instâncias avaliadas, de acordo com o nível de detalhe da avaliação de consistência que se deseja executar.

NL2AlloyviaOCL (UNIVERSITY OF BIRMINGHAM, 2015) é uma ferramenta que gera expressões em linguagem *Alloy* a partir de frases textuais escritas em inglês e de um modelo UML correspondente. Contudo, não proporciona um método para analisar a consistência das regras de negócio geradas através dessa ferramenta.

SILVA e BARRETO (2009) elaboraram uma abordagem para a validação de regras de negócio através da transformação e composição de modelos MDA[19]. O foco desse estudo é a geração automatizada de regras de negócio que podem ser reutilizadas em outros modelos MDA específicos da plataforma Java. Não foram contemplados todos os tipos de regras de negócio, apenas regras de validação de dados e instâncias de objetos. Também não foram abordadas questões como sobreposição de regras de negócio. A estratégia abordada em nosso trabalho avança nesse sentido, além de ser abrangente a qualquer tipo de regra de negócio que possa ser estruturada em linguagem de negócio.

---

[18] *Integrated Development Environment*.

[19] *Model Driven Architecture.*

CUNHA (2009) elaborou uma abordagem para a transformação de regras de negócio escritas na linguagem SBVR[20] para a linguagem OCL[21], de forma a gerar automaticamente o modelo de regras de negócio no formato PIM[22] da MDA. A linguagem OCL é baseada em lógica de predicados de primeira ordem, assim como o *Alloy*, mas usa uma sintaxe similar às linguagens de programação e fortemente relacionada à sintaxe da UML[23]. A linguagem *Alloy* tem uma sintaxe mais convencional e semântica mais simples, além de ser totalmente declarativa, enquanto a OCL mistura elementos declarativos e operacionais, além de ser muito orientada a implementação, não sendo muito adequada para a modelagem em nível conceitual ou de negócio, como o método MIIRNA se propõe.

## 8.2 BRMS

Há várias ferramentas comerciais de gerenciamento e simulação de regras de negócio, conhecidas como BRMS[24]. Essas ferramentas se caracterizam por oferecer um ambiente consolidado para regras de negócio, que facilita as atividades de criação, gerenciamento e até mesmo a execução das regras de negócio, na forma de pequenos componentes executáveis de regras de negócio[25] que podem ser invocados no contexto de *web services* ou de chamadas API[26] por uma aplicação construída dentro da arquitetura SOA[27].

Os exemplos de fabricantes/produtos de BRMS mais reconhecidos no mercado são IBM (ODM - *Operational Decision Management*) e Red Hat (JBoss). Apesar de todos os benefícios que um BRMS pode proporcionar, há questões muito importantes quanto ao custo (geralmente demandam grande investimento financeiro, tanto para aquisição/suporte quanto para capacitação das equipes) e quanto à infraestrutura necessária (geralmente demandam servidores e bancos de dados exclusivos e dedicados e processadores com alta capacidade de

---

[20] *Semantics of Business Vocabulary and Rules.*

[21] *Object Constraint Language.*

[22] *Platform Independent Model.*

[23] *Unified Modeling Language.*

[24] *Business Rules Management System.*

[25] *Business rules applets.*

[26] *Application Program Interface.*

[27] *Service Oriented Architecture.*

execução). Além disso, para ser totalmente efetivo, um BRMS demanda um investimento relevante em arquitetura SOA e em BPM[28]. Assim, como citado por VON HALLE e GOLDBERG (2006), nem todas as regras de negócio demandam uma solução de tão grande porte e tão dispendiosa como um BRMS.

O nosso método demanda pouca capacidade de hardware, tendo em vista que o *Alloy* é uma aplicação Java, que roda até mesmo em computadores desktop e notebooks. A verificação antecipada de consistência de modelos de regras de negócio, abordada no método MIIRNA, independe da escrita do código da aplicação para ser efetuada. E a linguagem *Alloy* demanda apenas o conhecimento básico em lógica de primeira ordem, o que favorece um baixo custo de capacitação necessária para seu uso.

[28] *Business Process Management.*

# 9 CONCLUSÕES

## 9.1 RESUMO DOS RESULTADOS

A identificação antecipada de erros nas especificações de regras de negócio proporciona ganhos significativos para os projetos de desenvolvimento/manutenção de sistemas de informação. A redução de custos desses projetos pode ser alcançada com a diminuição do tempo e do esforço necessários para a entrega dos seus produtos finais. Também se pode aferir na forma de ganhos intangíveis, alcançados com a efetiva satisfação dos clientes e usuários desses sistemas de informação.

O método MIIRNA proposto neste trabalho provê, tanto para analistas de negócio quanto para desenvolvedores, uma ferramenta que permite uma rápida avaliação da consistência de um conjunto de regras de negócio, sob o ponto de vista lógico. Também propõe uma forma incremental do processo de avaliação de um conjunto de regras de negócio, realizada à medida que esse conjunto é atualizado, durante o ciclo de vida da aplicação dessas regras.

Além disso, um conjunto consistente de regras de negócio pode ser gerado com mais facilidade, já que a atividade de verificação de regras de negócio, cuja automatização é proporcionada pelo método MIIRNA, pode viabilizar que a subsequente atividade de validação de regras de negócio, a ser executada de forma manual, venha a ocorrer com mais rapidez e menos probabilidade de ocorrência de erros intrínsecos à semântica ou à lógica das proposições.

O método MIIRNA também se mostrou um complemento bastante recomendável ao uso de casos de testes para se identificar erros nas especificações de negócio. Conforme explanado anteriormente nesta dissertação, os casos de teste dependem da implementação do código para serem efetuados, o que geralmente só ocorre em etapas posteriores dos projetos de desenvolvimento/manutenção de sistemas de informação, quando a correção de possíveis erros é bem mais impactante. Além disso, centenas ou mesmo milhares de casos de teste podem ser necessários para que se chegue a um resultado, que, mesmo assim, pode não ser totalmente confiável.

Este trabalho mostrou nos capítulos 5 e 6 que o método MIIRNA proposto é capaz de identificar as inconsistências mais típicas em regras de negócio (sobreposição, redundância e

conflito). A exibição dos contraexemplos encontrados torna a detecção e a correção desses erros mais fáceis e rápidas. Além disso, demonstrou-se que a escalabilidade do tamanho do conjunto de regras de negócio pode ser facilmente tratada no método MIIRNA, pois a inclusão de novas regras a um modelo já consistente não impactou na performance da sua execução. Isto implica em um modelo de especificação de regras de negócio mais consistente e confiável, cuja verificação se inicia antes da atividade de implementação, podendo ser realizada junto com a atividade de modelagem do sistema de informação.

O método MIIRNA também mostrou ser um complemento prático às funcionalidades do *Alloy*, tendo em vista este ser basicamente um *model finder*, ou seja, tenta achar um exemplo que seja uma solução para um conjunto de restrições lógicas de um modelo escritas na linguagem *Alloy*. Ou, caso não encontre a solução, exibe uma ou mais instâncias que exemplificam a violação dessas restrições lógicas. O método MIIRNA vai além disso e busca identificar precisamente qual a restrição lógica que causou a inconsistência em um modelo de especificação de regras de negócio escritas no *Alloy*.

## 9.2 CONTRIBUIÇÕES

Diferentes abordagens para a verificação automatizada de regras de negócio têm sido apresentadas na literatura. Por exemplo, o uso de metaconhecimento e lógica não-monotônica no EVA Project (CHANG; COMBS; STACHOWITZ, 1990), a técnica PREPARE de representação de conhecimento através de classes especiais de redes *Petri* de predicados/transições (ZHANG; NGUYEN, 1994), a ferramenta NuSMV2 para a checagem de modelos baseada na técnica SMV e em algoritmo BDD (CIMATTI et al., 2002), entre outras.

Entretanto, todas, em um determinado ponto, demandam implementações complexas ou têm limitações lógicas, tornando o seu uso não intuitivo ou limitado a pessoas com bastante experiência em programação e com conhecimento avançado em lógica matemática e métodos de prova.

A principal contribuição deste trabalho é um método formal e estruturado para a verificação automatizada da consistência lógica de um conjunto de regras de negócio, baseado em lógica de primeira ordem, portanto mais simples e amigável que outras técnicas formais existentes. O método pode ser utilizado sem que haja necessidade de um profundo conhecimento de lógica matemática e totalmente dessassociado da implementação do código.

Além disso, a inclusão de novas regras de negócio e a atualização das existentes têm pequeno impacto na verificação de consistência do novo modelo gerado, caracterizando, com isso, que o método é uma opção robusta e confiável, tanto para analistas de negócio quanto para desenvolvedores.

Outro resultado alcançado foi a publicação do artigo "*A Method for Verifying the Consistency of Business Rules Using Alloy*" (GUIMARAES et al., 2014), apresentado no congresso *SEKE 2014 The Twenty-Sixth International Conference on Software Engineering and Knowledge Engineering*, que aconteceu em Vancouver (Canadá), de 1 a 3 de julho de 2014.

## 9.3 LIMITAÇÕES

O método pressupõe o conhecimento prévio pelo usuário sobre lógica de primeira ordem e sobre a linguagem usada pela ferramenta *Alloy*. Mas há bastante material didático disponível no site do *Alloy* (SEATER et al., 2015) ou mesmo no livro publicado pelo criador dessa ferramenta (JACKSON, 2006), além de outras fontes disponíveis na internet sobre esse assunto.

O método não se aplica a erros não possíveis de serem verificados de forma automatizada: erros de formação das regras, uso de termos não existentes no modelo conceitual de negócio e ausência ou incompletude das regras de negócio. Esses tipos de erros devem ser tratados manualmente através de revisões de qualidade ou outras estratégias, conforme abordado no capítulo 1 desta dissertação.

Também não foi possível, conforme abordado no capítulo 1 desta dissertação, realizar um experimento de utilização do método MIIRNA em situações reais, por uma quantidade representativa de analistas e desenvolvedores, obtendo-se dados estatísticos mais precisos, quanto ao esforço humano necessário para a aplicabilidade deste método. Nosso trabalho focou em avaliar a viabilidade do uso do método MIIRNA, através de dois exemplos de uso aplicados a domínios de negócio fictícios, porém similares a domínios reais de negócio.

## 9.4 TRABALHOS FUTUROS

Os próximos desafios estão concentrados em evoluir o método para tornar seu uso ainda mais fácil por pessoas não técnicas. Também serão buscados meios para automatizar a identificação das regras que se tornam inconsistentes com a inclusão de novas regras no já consistente conjunto de regras de negócio.

Assim, pretendemos concentrar esforços para integrar o nosso método à ferramenta NL2AlloyviaOCL (UNIVERSITY OF BIRMINGHAM, 2015), que permite de forma ainda incipiente traduzir sentenças de especificações de negócio escritas em linguagem natural para sentenças escritas em linguagem *Alloy*. Atualmente, a ferramenta NL2AlloyviaOCL não efetua a verificação de consistência dessas sentenças geradas em linguagem *Alloy*.

Iremos estudar a possibilidade de integrar o método MIIRNA à ferramenta OLED (UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO, 2015), que permite criar modelos conceituais, usando um ambiente para desenvolvimento, avaliação e implementação de ontologias de domínio através da linguagem de modelagem OntoUML, baseada na teoria de domínio ontologicamente fundamentada UFO (*Unified Foundational Ontology*) (GUIZZARDI, 2005). O uso do OLED é particularmente útil para a criação de modelos conceituais com qualidade estrutural, pois faz a verificação sintática desses modelos de acordo com os axiomas da UFO, podendo posteriormente transformar esse modelos automaticamente em modelos *Alloy*.

Também pretendemos concentrar esforços para integrar o nosso método aos estudos efetuados por CUNHA (2009) e ARAUJO (2010), de forma que as sentenças de regras de negócio sejam automaticamente convertidas para as linguagens OCL e *Alloy*, de forma que possam ser verificadas automaticamente quanto à sua consistência lógica e quanto à validação da sua aplicação nos processos de negócio, identificando possíveis violações dessas regras. Um facilitador para a integração desses estudos é que usaram o mesmo exemplo da locadora fictícia de veículos, EU-Rent, em seus experimentos.

Uma vez integradas as soluções e os estudos acima citados, pretendemos implantar o método em um ambiente empresarial real, nos cenários de negócio em que a empresa identifica que seus processos de negócio, suas regras de negócio e seus sistemas de informação não estão apresentando os resultados esperados, causando perdas financeiras e insatisfação com os clientes finais de seus produtos. Assim, espera-se que o método indique as inconsistências nas regras de negócio e as possíveis violações causadas pelos processos de negócio, possibilitando a reengenharia dos sistemas de informações e também a melhoria dos processos da empresa como um todo.

Pretendemos fazer um experimento controlado, aplicado a mais de uma equipe de analistas de negócio e desenvolvedores de software, para avaliar o esforço humano necessário para a utilização do método MIIRNA em um ambiente empresarial real, de forma

que possamos ter evidências do esforço necessário e da redução de custos alcançada com a identificação antecipada de inconsistências nas especificações de regras de negócio, bem como do nível de dificuldade da utilização do método MIIRNA.

A escalabilidade computacional do método MIIRNA também fará parte de um experimento futuro. Nesta dissertação, foi verificado que é factível a aplicação do método MIIRNA em exemplos de complexidade e porte pequenos e médios. Levando-se em conta que os *SAT Solvers* vêm se tornando cada vez mais rápidos, também vêm aumentando a velocidade e a escalabilidade do *Alloy Analizer* em resolver problemas maiores (JACKSON, 2006). Graças às recentes evoluções das tecnologias de *SAT Solvers* (Kodkod, MiniSat, zChaff, Berkmin, etc.), hoje em dia o *Alloy Analizer* pode examinar espaços de casos com tamanho na casa de bilhões ou mais e, dessa forma, oferece um grau de cobertura impossível de ser alcançado usando casos de teste. O avanço do hardware também deve ser levado em consideração. Uma análise do *Alloy*, que levava uma hora há dez anos, atualmente leva apenas segundos para ser executada nas máquinas disponíveis hoje em dia (JACKSON, 2006). Além disso, para exemplos maiores e mais complexos, a *small scope hypothesis* também pode garantir a eficiência computacional do método MIIRNA. A quantidade de instâncias definidas para a análise efetuada pelo *Alloy Analizer* através da *small scope hypothesis* pode definir grandes subespaços de análise, sem examiná-los totalmente (JACKSON, 2006).

# REFERÊNCIAS

ARAUJO, B. M. **Um método para validar a conformidade de processos de negócio com regras de negócio**. 2010. Dissertação (Mestrado em Ciência da Computação) - Instituto de Matemática, Núcleo de Computação Eletrônica, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brasil.

BOYER, J.; MILI, H. **Agile business rule development**: process, architecture, and JRules examples. New York: Springer, 2011.

ECLIPSE FOUNDATION. **ABRD 1.2**. Disponível em: <http://epf.eclipse.org/wikis/epfpractices/practice.tech.abrd.base/guidances/practices/abrd_697B2958.html>. Acesso em: 29 jul. 2015.

BUSINESS RULES GROUP. **Defining business rules... What is a business rule?** Disponível em: <http://www.businessrulesgroup.org/defnbrg.shtml>. Acesso em: 29 jul. 2015.

CHANG, C.; COMBS, J.; STACHOWITZ, R. A report on the expert systems validation associate (EVA). **Expert Systems with Applications**, v. 1, n. 3, p. 217-230, 1990. Special Issue: Verification and Validation of Knowledge-Based Systems.

CIMATTI, A. et al. NuSMV 2: an OpenSource tool for symbolic model checking. In: INTERNATIONAL CONFERENCE ON COMPUTER AIDED VERIFICATION (CAV '02), 14., 2002, London. **Proceedings**... London: Springer-Verlag, 2002. p.359-364.

CUNHA, C. A. A. **Uma abordagem para a transformação de regras de negócio na arquitetura dirigida por modelos**. 2009. Dissertação (Mestrado em Ciência da Computação) - Instituto de Matemática, Núcleo de Computação Eletrônica, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brasil.

DENNE M.; CLELAND-HUANG, J. **Software by numbers**: low risk, high return development. Upper Saddle River, N.J.: Prentice Hall, 2004.

ECLIPSE FOUNDATION. **ABRD 1.2**. Disponível em: <http://epf.eclipse.org/wikis/epfpractices/practice.tech.abrd.base/guidances/practices/abrd_697B2958.html>. Acesso em: 29 jul. 2015.

GOGOLLA, M.; BüTTNER, F.; RICHTERS, M. USE: a UML-based specification environment for validating UML and OCL. **Science of Computer Programming**, v. 69, n. 1-3, p. 27-34, 2007.

GUIMARAES, D. et al. A method for verifying the consistency of business rules using alloy. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING AND

KNOWLEDGE ENGINEERING, 26., 2014, Vancouver. **Proceedings**… 2014. p. 381-386.

GUIZZARDI, G. **Ontological Foundations for Structural Conceptual Models**. 2005. PhD Thesis (CUM LAUDE), University of Twente, The Netherlands, Telematica Institut Fundamental Research Series no. 15, ISBN 90-75176-81-3.

HICKS, R. C. The no inference engine theory — performing conflict resolution during development. **Decision Support Systems**, v. 43, n. 2, p.435-444, 2007.

HODGES, W. Logic. New York: Penguin Books, 2001.

HUANG, C.-J.; LIN, Y.-H. A Conflict Treatment Model for Uncertainty Rule-based Knowledge. In: INTERNATIONAL CONFERENCE ON INNOVATIVE COMPUTING, INFORMATION AND CONTROL, 2., 2007, Kumamoto. **Proceedings**... IEEE, 2007. p.152.

JACKSON, D. **Software abstractions**: logic, language and analisys. Cambridge: MIT Press, 2006.

JACKSON, D. **Alloy**: a language and tool for relational models. Disponível em: <http://alloy.mit.edu/alloy/>. Acesso em: 29 jul. 2015.

MOORE, C.; MERTENS, S. **The nature of computation**. Oxford; New York: Oxford University Press, 2011.

MORGAN, T. **Business rules and information systems**: aligning IT with business goals. Boston: Addison-Wesley, 2002.

OBJECT MANAGEMENT GROUP. **Semantics of Business Vocabulary and Business Rules (SBVR)**: version 1.1. Disponível em: <http://www.omg.org/spec/SBVR/1.1/PDF/>. Acesso em: 29 jul. 2015.

PETERSEN, K.; FELDT, R.; MUJTABA, S.; MATTSSON, M. **Systematic Mapping Studies in Software Engineering**. 12th International Conference on Evaluation and Assessment in *Software* Engineering (EASE). University of Bari, Italy, 26 - 27 June 2008.

ROSS, R. G. **Business rule concepts**: getting to the point of knowledge. Houston: Business Rules Solution, 2009.

ROSS R. G.; LAM, G. S. W. **Building business solutions**: business analysis with business rules. Houston: Business Rules Solutions, 2011.

SEATER, R. et al. **Tutorial for alloy analyzer 4.0.** Disponível em: <http://alloy.mit.edu/alloy/tutorials/online/>. Acesso em: 9 out. 2015.

SILVA, J. B.; BARRETO, L. P. Uma nova abordagem para validação de regras de negócio através de transformação e composição de modelos MDA. **iSys - Revista Brasileira de Sistemas de Informação**, Brasília, DF, v. 2, 2009. Disponível em: <http://www.seer.unirio.br/index.php/isys/article/view/303/389>. Acesso em: 29 jul. 2015.

SOUZA, M. G. de. **Uma abordagem de regras de negócio baseada em linguagem natural estruturada**. 2002. Dissertação (Mestrado em Ciência da Computação) - Instituto de Matemática, Núcleo de Computação Eletrônica, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brasil.

TAYLOR, J. **Decision management systems**: a practical guide to using business rules and predictive analytics. Upper Saddle River, NJ: IBM Press, 2012.

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO. **OLED OntoUML Lightweight Editor**. Disponível em: <http://nemo.inf.ufes.br/projects/oled/>. Acesso em: 29 jul. 2015.

UNIVERSITY OF BIRMINGHAM. **NL2OCL Project**. Disponível em: <http://www.cs.bham.ac.uk/~bxb/NL2OCLviaSBVR/NL2Alloy.html>. Acesso em: 29 jul. 2015.

VON HALLE, B. **Business rules applied**: business better systems using the business rules approach. New York: Wiley, 2002.

VON HALLE, B.; GOLDBERG, L. **The business rule revolution**: running business the right way. Cupertino, CA: Happy About, 2006.

ZHANG, D.; NGUYEN, D. PREPARE: a tool for knowledge base verification. **IEEE Transactions on Knowledge and Data Engineering**, v. 6, n. 6, p. 983-989, dez. 1994.

# APÊNDICE

## APÊNDICE A – LISTA DE ARTIGOS INCLUÍDOS NO MAPEAMENTO SISTEMÁTICO

[BRI001] BIZERRA JUNIOR, E.M.; SILVEIRA, D.S.; CRUZ, M.L.P.M.; WANDERLEY, F.J. A. **A Method for Generation of Tests Instances of Models from Business Rules Expressed in OCL**. In: Latin America Transactions, IEEE (Revista IEEE América Latina), vol.10, no.5, pp.2105-2111, Sept. 2012. Disponível em: < http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6362355&isnumber=6362342 >. Acesso em: 29 jul. 2015.

[BRI002] BAJWA, I.S.; BORDBAR, B.; LEE, M.G. **OCL Constraints Generation from Natural Language Specification.** In: Enterprise Distributed Object Computing Conference (EDOC), 2010 14th IEEE International, pp.204-213, 25-29 Oct. 2010. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5630350>. Acesso em: 29 jul. 2015.

[BRI003] BAJWA, I.S.; BORDBAR, B.; ANASTASAKIS, K.; LEE, M. **On a chain of transformations for generating alloy from NL constraints.** In: Digital Information Management (ICDIM), 2012 Seventh International Conference on, pp.93-98, 22-24 Aug. 2012. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6360153>. Acesso em: 29 jul. 2015.

[BRI004] CHAN, L. W.; HEXEL, R.; WEN, L. **Rule-Based Behaviour Engineering: Integrated, Intuitive Formal Rule Modelling.** In: Software Engineering Conference (ASWEC), 2013 22nd Australian, pp.20-29, 4-7 June 2013. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6601289>. Acesso em: 29 jul. 2015.

[BRI005] CHNITI, A., ALBERT, P., CHARLET, J. **A loose coupling approach for combining OWL ontologies and business rules.** In: CEUR Workshop Proceedings, 874, 2012. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-84881148632&partnerID=40&md5=cc1e582d57bde7fbda2930bb21b55c71>. Acesso em: 29 jul. 2015.

[BRI006] CHNITI, A.; ALBERT, P.; CHARLET, J. **MDR ontology: An ontology for managing ontology changes impacts on business rules.** In: CEUR Workshop Proceedings, 890, 2012. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-84892567565&partnerID=40&md5=d46fdee519f87254bed0067b6cb1a485>. Acesso em: 29 jul. 2015.

[BRI007] COSENTINO, V.; CABOT, J.; ALBERT, P.; BAUQUEL, P.; PERRONNET, J. **Extracting business rules from COBOL: A model-based framework.** In: Reverse Engineering (WCRE), 2013 20th Working Conference on, pp.409-416, 14-17 Oct. 2013. Disponível em:

<http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6671316>. Acesso em: 29 jul. 2015.

[BRI008] DECKER, H.; DE JUAN-MARIN, R. **Enabling Business Rules for Concurrent Transactions.** In: P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2011 International Conference on, pp.207-212, Barcelona, 26-28 Oct. 2011. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6103160>. Acesso em: 29 jul. 2015.

[BRI009] DECKER, H.; DE JUAN-MARIN, R. **Inconsistency-Tolerant Belief Revision for Distributed Decision Support**. In: P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2013 Eighth International Conference on, pp.387-393, Compiègne, 28-30 Oct. 2013. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6681259>. Acesso em: 29 jul. 2015.

[BRI010] DRAGICEVIC, S.; CELAR, S. **Method for elicitation, documentation and validation of software user requirements (MEDoV).** In: Computers and Communications (ISCC), 2013 IEEE Symposium on, pp.956-961, 7-10 July 2013. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6755073>. Acesso em: 29 jul. 2015.

[BRI011] DUBAUSKAITE, R.; VASILECAS, O. **The approach of ensuring consistency of UML model based on rules.** In: Proceedings of the 11th International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing on International Conference on Computer Systems and Technologies (CompSysTech '10), ACM, New York, NY, USA, 71-76, 2010. Disponível em: <http://doi.acm.org/10.1145/1839379.1839393>. Acesso em: 29 jul. 2015.

[BRI012] EMANI, C. K. **Automatic detection and semantic formalisation of business rules.** In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), volume 8465 LNCS, pp. 834-844, 2014. Disponível em: < http://www.scopus.com/inward/record.url?eid=2-s2.0-84902590459&partnerID=40&md5=faae40b89a4ba18c8efaf94e6b56e36d >. Acesso em: 29 jul. 2015.

[BRI013] FERREIRA, D. A.; SILVA, A. R. **RSL-IL: An interlingua for formally documenting requirements.** In: Model-Driven Requirements Engineering (MoDRE), 2013 International Workshop on, pp.40-49, 15 July 2013. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6597262>. Acesso em: 29 jul. 2015.

[BRI014] FEUTO, P.B.; CARDEY, S.; GREENFIELD, P.; EL ABED, W. **Domain Specific Language Based on the SBVR Standard for Expressing Business Rules.** In: Enterprise Distributed Object Computing Conference Workshops (EDOCW), 2013 17th IEEE International, pp.31-38, 9-13 Sept. 2013. Disponível em: <

http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6690531>. Acesso em: 29 jul. 2015.

[BRI015] HAKYOUL, C.; SONGCHUN M. **A Methodology for Accurate and Redundancy-Free Business Requirements Description Using Ontology.** In: Information Science and Applications (ICISA), 2011 International Conference on, pp.1-7, 26-29 April 2011. Disponível em: < http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5772352&isnumber=5772305>. Acesso em: 29 jul. 2015.

[BRI016] JANJUA, N.K.; HUSSAIN, F.K. **Rule-Based Business Policies Specification, Reasoning and Integration for Business Process Model Extraction.** In: Broadband and Wireless Computing, Communication and Applications (BWCCA), 2011 International Conference on, pp.51-56, 26-28 Oct. 2011. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6103015>. Acesso em: 29 jul. 2015.

[BRI017] JINJIAO, L.; HAITAO, P.; LIZHEN, C. **OO Petri Net based Business Rule modeling of intelligent taxation system in pervasive environment.** In: Pervasive Computing and Applications (ICPCA), 2010 5th International Conference on, pp.398-401, 1-3 Dec. 2010. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5704135>. Acesso em: 29 jul. 2015.

[BRI018] KARPOVIC, J.; NEMURAITE, L.; STANKEVICIENE, M. **Requirements for Semantic Business Vocabularies and Rules for Transforming Them into Consistent OWL2 Ontologies.** In: Information and Software Technologies, Volume 319 of the series Communications in Computer and Information Science, publisher Springer Berlin Heidelberg, pp 420-435, 2012. Disponível em: <http://rd.springer.com/chapter/10.1007%2F978-3-642-33308-8_35>. Acesso em: 29 jul. 2015.

[BRI019] KUMAR, S. **A Knowledge Acquisition System for a university educational process.** In: Industrial and Information Systems (ICIIS), 2012 7th IEEE International Conference on, pp.1-6, 6-9 Aug. 2012. Disponível em: < http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6304805&isnumber=6304761>. Acesso em: 29 jul. 2015.

[BRI020] LEX, W. **An essential language for declarative business rules.** In: Proceedings of the 3rd Computer Science Education Research Conference on Computer Science Education Research (CSERC '13), Open Universiteit, Heerlen, The Netherlands, Article 5, Pages 59-65, 2013. Disponível em: < http://dl.acm.org/citation.cfm?id=2541917.2541922>. Acesso em: 29 jul. 2015.

[BRI021] LING, L.; HANG, D. **Building a Change Management Model for E-Government Services Evolution**. In: Management of e-Commerce and e-Government (ICMeCG), 2011 Fifth International Conference on, pp.87-92, 5-6 Nov. 2011. Disponível em: <

http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6092637&isnumber=6092607>. Acesso em: 29 jul. 2015.

[BRI022] LUO, Q.; TANG, C.; LI, C.; YU, E. **Detecting self-conflicts for business action rules.** In: Computer Science and Network Technology (ICCSNT), 2011 International Conference on, vol.2, pp.1274-1278, 24-26 Dec. 2011. Disponível em: < http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6182192>. Acesso em: 29 jul. 2015.

[BRI023] MAXWELL, J.C.; ANTON, A.I.; SWIRE, P. **A legal cross-references taxonomy for identifying conflicting software requirements.** In: Requirements Engineering Conference (RE), 2011 19th IEEE International, pp.197-206, Aug. 29 2011-Sept. 2 2011. Disponível em: < http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6051647&isnumber=6051621>. Acesso em: 29 jul. 2015.

[BRI024] NAMBIAR, U.; FARUQUIE, T.A.; PRASAD, K.H.; SUBRAMANIAM, L.V.; MOHANIA, M.K. **Data Augmentation as a Service for Single View Creation.** In: Services Computing (SCC), 2011 IEEE International Conference on, pp.40-47, 4-9 July 2011 Disponível em: < http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6009242&isnumber=6009236>. Acesso em: 29 jul. 2015.

[BRI025] NAZARENKO, A.; LÉVY, F. **Combining acquisition and debugging of business rule models.** In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 8035 LNCS, pp. 234-248, 2013. Disponível em: < http://www.scopus.com/inward/record.url?eid=2-s2.0-84881012668&partnerID=40&md5=9fa08cef3eb691e8fe4b6afb5cc0dadb >. Acesso em: 29 jul. 2015.

[BRI026] NJONKO, P.B.F.; EL ABED, W. **From natural language business requirements to executable models via SBVR.** In: Systems and Informatics (ICSAI), 2012 International Conference on, pp.2453-2457, 19-20 May 2012. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6223550>. Acesso em: 29 jul. 2015.

[BRI027] NJONKO, P.B.F.; CARDEY, S.; GREENFIELD, P.; EL ABED, W. **RuleCNL: A controlled natural language for business rule specifications.** In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 8625 LNAI, pp. 66-77, 2014. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-84905457493&partnerID=40&md5=0f0f60676bf1a629714dbd141004aff9>. Acesso em: 29 jul. 2015.

[BRI028] OLIVEIRA, M.; VIANA, D.; CONTE, T.; VIEIRA, S.; MARCZAK, S. **Evaluating the REMO-EKD technique: A technique for the elicitation of software requirements based on EKD organizational models.** In: Empirical Requirements Engineering (EmpiRE), 2013 IEEE Third International Workshop on, pp.9-16, 15 July 2013. Disponível em: <

http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6615210>. Acesso em: 29 jul. 2015.

[BRI029] PILLAT, R.M.; BASSO, F.P.; OLIVEIRA, T.C.; WERNER, C.M.L. **Ensuring consistency of feature-based decisions with a business rule system.** In: Proceedings of the Seventh International Workshop on Variability Modelling of Software-intensive Systems (VaMoS '13). ACM, New York, NY, USA, Article 15, 2013. Disponível em: < http://doi.acm.org/10.1145/2430502.2430523>. Acesso em: 29 jul. 2015.

[BRI030] SELWAY, M.; GROSSMANN, G.; MAYER, W.; STUMPTNER, M. **Formalising Natural Language Specifications Using a Cognitive Linguistics/Configuration Based Approach.** In: Enterprise Distributed Object Computing Conference (EDOC), 2013 17th IEEE International, pp.59-68, 9-13 Sept. 2013. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6658264>. Acesso em: 29 jul. 2015.

[BRI031] SHARMA, R.; BISWAS, K.K. **Using courteous logic based representations for requirements specification**. In: Managing Requirements Knowledge (MARK), 2011 Fourth International Workshop on, pp.12-16, 30 Aug. 2011. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6046556>. Acesso em: 29 jul. 2015.

[BRI032] ZHOU, G. **Model checking inconsistency recovery costs.** In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 7221 LNCS, pp. 195-200, 2012. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-84865518617&partnerID=40&md5=89b5947bdbf719d37b7d725cfce208bd>. Acesso em: 29 jul. 2015.