Análise em Dependabilidade Orientada a Objetivos para Variabilidade e Automação em BPMN

Yan Victor dos Santos*, Genaína Nunes Rodrigues

Departamento de Ciência da Computação, Universidade de Brasília

Resumo

No que tange a aspectos dinâmicos dos sistemas de software, dentre seus desafios encontra-se a necessidade de (1) verificar se as operações do sistema estão de acordo com seus objetivos, e quando necessário, (2) explorar configurações alternativas que restaurem o sistema ao estado normal de operações na presença de falhas, continuando a satisfazer seus requisitos ainda que em condições adversas. Nesse contexto, a modelagem orientada a objetivos Goal-oriented requirements engineering *tem sido uma abordagem promissora particularmente para modelar aspectos dinâmicos dos sistemas de software. Tal abordagem tem sido usada para customizar um software de acordo com regras organizacionais de implantação e para derivar projetos com alto grau de variabilidade e para maximizar resiliência e adaptabilidade. No entanto, é comum utilizar-se industrialmente a abordagem BPM (*Business Process Management*) para gerenciar processos de negócio, retratando diretamente o ambiente organizacional. Para alavancar o uso de sistemas dinâmicos modelados por meio de objetivos possam ser explorados sob a perspecitva de processos de negócio, este trabalho tem como objetivo automatizar a tradução de um modelo de objetivos para um processo de negócio segundo a abordagem BPM. Para isto, utilizamos o GODA (Goal-Oriented Dependability Analysis) framework e suas ferramentas de suporte que auxiliam especialistas em análise de requisitos de sistemas e que levam em consideração variações de contexto. O GODA compreende a semântica do CRGM (*Contextual and Runtime Goal Model*), o que viabiliza a criação de estratégias alternativas para cenários que incluem mudanças de contexto. Adicionalmente, antecipa a análise de comportamento do sistema, evitando assim comportamentos indesejados durante a execução do sistema ao cumprir um determinado objetivo. Como solução, propõe-se gerar um processo de negócio diretamente de um CRGM sintaticamente bem formado, permitindo uma perspectiva de como os objetivos e tarefas de um modelo de objetivo colaboram-se mutuamente sob uma outra perspectiva dinâmica: a de processo de negócio. Desta maneira, realiza-se o objetivo por meio de um processo consistente, gerado automaticamente e permitindo uma vasta e flexível representação dinâmica do sistema dinâmico de interesse.*

Palavras-chave: *Goal Modeling*, *Dependability*, *Runtime Analysis*, *Business Process Management*

1 Introdução

A abordagem BPM, apresentada em [1], é bastante utilizada em grandes empresas que buscam gerenciar processos de negócio de maneira automatizada, eficiente e produtiva. Porém, ao tratarmos diretamente com ambientes que estão sujeitos a mudanças inesperadas de contexto, as chances de um ou mais objetivos não serem alcançados são aumentadas consideravelmente.

A Engenharia de Requisitos Orientada a Objetivos, ou Goal-Oriented Requirements Engineering (GORE, [2]), oferece meios de especificar precisamente quais são os requisitos do sistema em uma visão hierárquica de como os objetivos são refinados em subobjetivos até que sejam finalmente realizados por meio de tarefas ou funcionalidades concretas do sistema. Tal especificação também permite analisar variabilidade de contextos que poderiam impedir a satisfação de um requisito, por meio de um Goal Model (GM). Ao transformarmos requisitos em metas equivalentes no GM, podemos decompor em diversos subobjetivos ou tarefas, como fundamentado em [3].

No âmbito de GORE, surge o framework GODA (*Goal-Oriented Dependability Analysis*) [3], o qual provê mecanismos para estimar dependabilidade de diferentes estratégias para alcançar os

*Endereço eletrônico: yanvictor_ds@hotmail.com

objetivos, tornando-se daí, adequada à análise de estratégias de adaptação sensíveis a contexto. O processo de verificação do GODA basea-se em verificações de modelo probabilísticas, por meio de cadeias de markov discretas traduzidas diretamente dos modelos de objetivo. Uma vez que a satisfação de um objetivo de um sistema pode ser prejudicada por mudanças inesperadas de contexto, torna-se necessário criar estratégias e estimar suas respectivas confiabilidades para que se possa cumprí-lo da melhor maneira possível. Por meio do GODA, juntamente com sua ferramenta de suporte que extende TAOM4E [3, 4], é possível obter estratégias usando conceitos do Modelo de Objetivos de Contexto, ou Context Goal Model (CGM), e do Modelo de Objetivos de Tempo de Execução, ou Runtime Goal Model (RGM). Tais especificações tornam mais visível a forma de como tarefas e subobjetivos devem ser realizados. Ao juntarmos o CGM com RGM, obtemos o CRGM, um modelo expressivo que engloba em si os conceitos dos dois modelos anteriores.

No entanto, o modelo de análise produzido pelo GODA está fundamentado em uma abordagem mais formal por meio de cadeias de Markov na linguagem PRISM. Essa linguagem, no entanto, é bastante distante do conhecimento geralmente presente em equipes de engenharia de requisitos ou de desenvolvimento de software tradicionalmente. Sendo mais comum a análise de processos de software por meio da linguagem BPMN. A proposta deste trabalho consiste em usar as regras definidas juntamente com a informações de contexto por meio do modelo CRGM no GODA para automatizar o processo de transformação do CRGM em um modelo na notação BPMN.

O resto do artigo está organizado da seguinte forma. A Seção 2 fornece os conceitos e informações necessárias para a compreensão do trabalho. A Seção 3 apresenta a abordagem utilizada para a transformação do CRGM para BPMN. Um exemplo de transformação será apresentado na Seção 4. A Seção 5 apresenta conclusões e resultados alcançados.

2 Fundamentação Teórica

2.1 Notação de Modelagem de Processos de Negócio

A Notação de Modelagem de Processos de Negócio (BPMN) auxilia grandes empresas no gerenciamento de processos de negócios [1]. Por exemplo, é possível obter maior controle administrativo, custos reduzidos, automação de processos e transparência nas etapas de um processo através de um modelo bem formado. Sua notação é representada por fluxos de atividades que abrangem eventos, tomadas de decisões e diversos outros recursos da notação. Um processo simples descreve a organização em termos de piscinas, e os agentes pertencentes a esta organização em termos raias. Uma raia geralmente contém seqüências de atividades, que por sua vez podem ser executadas por tarefas e até mesmo *Subprocessos*. O processo é iniciado pelo *Evento Inicial*, desenvolvido por eventos intermediários e encerrado pelo *Evento Terminal*.

Dentre os elementos apresentados no trabalho em [1], fazem parte o *Desvio Exclusivo*, onde apenas um caminho criado por ele poderá ser seguido, o *Desvio Paralelo*, onde dois ou mais fluxos podem ser processados paralelamente, o *Desvio Inclusivo*, onde uma ou mais combinações de caminhos podem ser executadas e o *Fluxo de Sequência*, um indicador que revela para onde o fluxo de uma determinada instância do processo deve seguir. *Conditional flow* é usado quando um fluxo pode ou não ser seguido, dependendo de uma condição imposta. Também fazem parte os *Dados de Entrada e Saída*, *Repositório de Dados*, *Conversações* para troca de mensagens logicamente relacionadas, *Links de conversação* para demonstrar relações entre dois objetos, entre outros.

2.2 GODA

2.2.1 Modelo de Objetivos de Contexto e de Tempo de Execução

Como apresentado por Mendonça et al. [3], um CRGM é uma 4-ulpa. Seu primeiro parâmetro é um modelo de objetivos composto por um conjunto N de objetivos e tarefas e suas respectivas relações. O segundo parâmetro é uma *anotação de tempo de execução* associada a um nó de N. O terceiro parâmetro recebe *fórmulas de contexto* associadas a um nó de N. Por fim, o quarto parâmetro é um ID que mapeia todo n pertencente a N.

Os objetivos e as tarefas são organizados e decompostos conforme um modelo estruturado em árvore. Um objetivo pode ser decomposto em subobjetivos, analogamente às tarefas. Onde as tarefas representam a forma concreta de como os objetivos são realizados Uma tarefa que não pode ser decomposta em outras subtarefas é chamada de tarefa folha. As regras do RGM estão descritas na Tabela 1.

Expressão	Significado
AND (n1;n2)	Satisfação em sequência de n1 e n2.
AND (n1#n2)	Satisfação paralela de n1 e n2.
OR (n1;n2)	Satisfação em sequência de n1 ou n2, ou de ambos.
OR (n1#n2)	Satisfação paralela de n1 ou n2, ou de ambos.
n+k	n deve ser satisfeito k vezes, com k>0.
n#k	Satisfação paralela de k instâncias de n, com k>0.
n@k	Máximo k − 1 tentativas de satisfazer n, com k>0.
opt(n)	A satisfação de n é opcional.
try(n)?n1:n2	Se n for satisfeito, n1 deve ser satisfeito; Senão, n2.
n1|n2	Satisfação alternativa de n1 ou n2, nunca de ambos.
skip	Sem ação. Útil para expressões ternárias condicionais.

Tabela 1: Regras RGM presentes no GODA, onde n, n1 e n2 representam objetivos ou tarefas em um modelo de objetivos.

A *decomposição AND* requer a satisfação de todos nós envolvidos. A *decomposição OR* permite que pelo menos um dos nós seja satisfeito. A satisfação em sequência da *decomposição AND* requer uma sequencialidade temporal durante execução das tarefas. Já a satisfação em paralelo da *decomposição AND* permite intercalação (*interleaving*) na execução de processos, desde que os nós envolvidos sejam satisfeitos. A satisfação em sequência da *decomposição OR* propõe que, para mais de um nó a ser satisfeito, os mesmos devem ser cumpridos na ordem descrita pela *anotação RGM*. Já a satisfação em paralelo da *decomposição OR* não impõe ordem de realização dentre os nós, desde que pelo menos um seja realizado.

Para os outros casos, $n+k$ define que n deve ser satisfeito k vezes. A regra $n\#k$ define a satisfação de k instâncias de n. A regra $n@k$ define que n deve ser satisfeito em no máximo k-1 tentativa(s). Casos onde a realização de n é opcional, pode-se representar por *opt(n)*. Já a regra *try(n)?n1:n2* representa a satisfação de n seguida da satisfação de $n1$ (caso n seja satisfeito), ou $n2$ (caso n não seja satisfeito). Por fim, *n1|n2* representa o *ou exclusivo*, onde apenas um dos dois pode ser satisfeito.

2.2.2 Modelagem Orientada a Objetivos e Contexto

A modelagem orientada a objetivos fornece meios para analisar requisitos das diferentes partes interessadas em um sistema de software [5, 6, 7]. Temos que objetivos são realizados por meio de atores, e os atores podem interdepender uns dos outros para alcançar seus objetivos. Os objetivos podem ser refinados em tarefas, que denotam processos a serem executados por um determinado ator.

Os caminhos alternativos na árvore de objetivos podem ser avaliados com respeito a objetivos qualitativos chamados de *SoftGoals* que são objetivos sem um critério concreto para serem satisfeitos. Geralmente expressam requisitos qualitativos e identificam o impacto positivo ou negativo em um objetivo ou tarefa por meio dos *links de contribuição* .

A Figura 1 contém um exemplo de CRGM onde é mostrado o objetivo principal $G0$ como uma propriedade de um ator. $G0$ é refinado em $G1$ e $G2$, subobjetivos que devem ser satisfeitos obrigatoriamente por consequência da definição de *decomposição AND*. Para satisfazer $G1$ é necessário que apenas $G3$ ou $G4$ seja satisfeito, como definido pela regra de *decomposição OR exculsiva* do RGM: $[G3|G4]$. $G3$ e $G4$ são refinados pela tarefa folha $T1$. Do outro lado, $G2$ é decomposto em uma única tarefa $T1$, que por sua vez é refinada em duas subtarefas folhas. $T1$ está restrita a uma das regras do RGM: $[try(T1.1)?skip:T1.2]$. Tal regra define que primeiro tentará executar $T1.1$. Em caso de sucesso, o comando escolhido será *skip*, ignorando completamente $T1.2$. Em caso de falha, a subtarefa que deverá ser executada será $T1.2$. Uma observação importante é que $G2$ será satisfeita apenas se a condição de contexto $C1$ for viável, sendo este um exemplo de aplicação CGM e RGM.

Por fim, dentre os elementos mostrados no exemplo da Figura 1 , o *Context Condition*, por exemplo *C1*, é relevante para reduzir o problema de variação de contexto. Tal elemento é responsável por indicar o contexto necessário para cumprir um determinado objetivo. Sua aplicação em CRGM terá um formato equivalente dentro da notação BPMN, após a conversão de modelo.

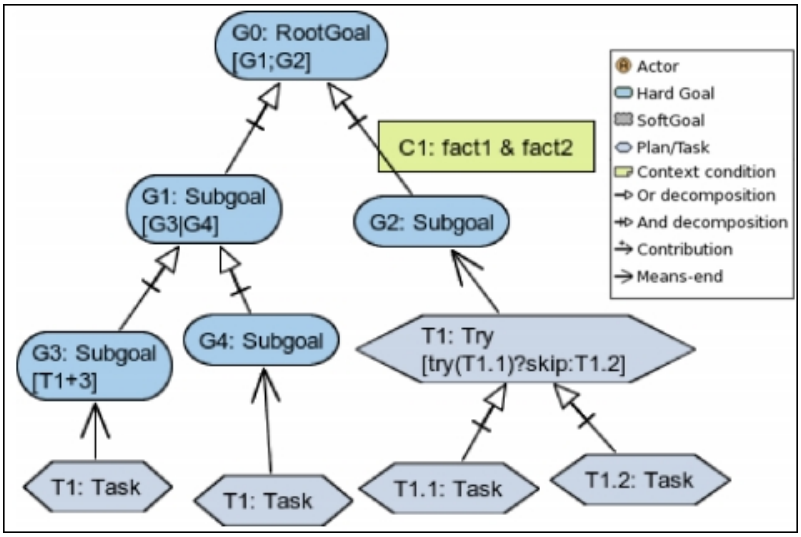

Figura 1: Um exemplo CRGM extraído de [3]

3 Abordagem de Conversão de CRGM para BMPN

O CRGM descreve e define a satisfação de um objetivo principal através de um fluxo gerado por regras. Nossa hipótese, desenvolvida por meio deste trabalho, é que este fluxo pode ser representado em um processo de negócio que utiliza notação BPMN. Dessa forma, para transfomarmos um modelo orientado a objetivos em um processo de negócio, primeiro torna-se necessário definir a equivalência entre os elementos dos dois modelos. A conversão de um CRGM para BPMN será apresentada dentro das próximas duas subseções.

3.1 Equivalência Entre Elementos Estruturais do CRGM e do BPMN

Nessa seção definimos a estrutura básica de equivalência para a representação dos elementos em um CRGM e os elementos correspondentes na representação de um BPMN. Enquanto um modelo CRGM caracteriza-se pela realização de um conjunto de tarefas e objetivos menores para alcançar uma meta principal, considerando as condições de contexto, o BPM é caracterizado por um fluxo de atividades que descreve um processo de negócio, considerando fluxos condicionais de execução.

Elementos do CRGM	**Elementos do BPMN**
Actor	*Pool*
Hard Goal	*Abstract Task*
SoftGoal	*Abstract Task*
Plan/Task	*Abstract Task*
Context Condition	*Exclusive Gateway*
Contribuition	*Conditional flow*; *Data Association*
Means-end	*Sequence Flow*

Tabela 2: Equivalências Entre Elementos Estruturais do CRGM e do BPMN.

Como descrito na Tabela 2, um ator (*actor*) passa a ser retratado como uma piscina (*pool*), pois o mesmo é o principal responsável pelo fluxo de tarefas do modelo de objetivos. Os elementos *Hard Goal* e *Plan/Task* são equivalentes a *Abstract Task*, por serem tratados genericamente como tarefas e objetivos que devem ser realizados. É importante ressaltar que, em nossa equivalência estrutural, o *SoftGoal*, assim como o *Hard Goal*, torna-se uma *Abstract Task*.

Uma *Context Condition* é uma condição de guarda apresentada no CRGM, sendo representada por um *Exclusive Gateway* após a conversão, pois o mesmo é capaz de redirecionar o fluxo das atividades de um processo dependendo do contexto inserido. Tal equivalência contribui para a redução da variabilidade em processos de negócio. *Contribuition* tem como objetivo representar uma informação relacionada a uma outra instância. Tal informação pode ser representada por *Conditional flow* e por uma *Data Association*, informando se a contribuição é positiva ou negativa.

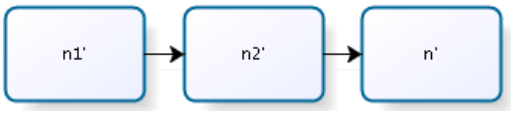

Figura 2: Processo BPMN para a regra **AND(n1;n2)**.

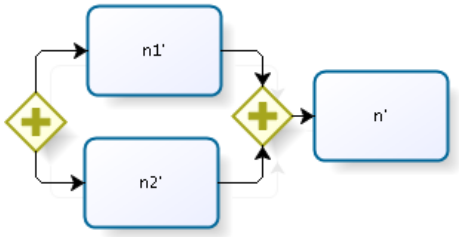

Figura 3: Processo BPMN para a regra **AND(n1#n2)**

Por fim, *Means-end* é equivalente a um *Sequence Flow*, que define a ordem de sequência das atividades. Embora a *decomposição AND* e a *decomposição OR* também sejam representadas por *Fluxos de Sequência*, a forma como cada uma é representada em sua totalidade depende diretamente das regras do CRGM.

3.2 Equivalência Entre as Relações dos Elementos por meio do CRGM

Para cada regra de comportamento do CRGM, conforme apresentado na Tabela 1, apresentamos a seguir como a conversão pode satisfazer a proposta do CRGM dentro de um processo de negócio, preservando a equivalência semântica do comportamento modelado no CRGM e a ser transformado em um comportamento no BPMN.

Para a conversão, serão considerados n, $n1$ e $n2$ como sendo objetivos ou tarefas de um CRGM, representando toda a estrutura de árvore do modelo de objetivos. Da mesma maneira, serão considerados n', $n1$' e $n2$' como sendo as respectivas atividades equivalentes a n, $n1$ e $n2$ em um processo de negócio. A conversão das regras está descrita logo a seguir:

1. n possui a regra AND(n1;n2): *Satisfação em sequência de n1 e n2.*

   Descrição: *Fluxo de Sequência* partindo de $n1$' para $n2$' e *Fluxo de Sequência* partindo de $n2$' para n'. Tal representação define que $n1$ será realizado antes de $n2$, satisfazendo a instância n.

2. n possui a regra AND(n1#n2): *Satisfação paralela de n1 e n2.*

   Descrição: a partir de um *Desvio Paralelo*, um *Fluxo de Sequência* segue para $n1$' e outro para $n2$'. O fluxo dos dois caminhos convergem em outro *Desvio Paralelo*. O último *Desvio Paralelo* possui *Fluxo de Sequência* seguindo para n'. Tal representação define que $n1$ será realizado de forma paralela em relação a $n2$, satisfazendo a instância n.

3. n possui a regra OR(n1;n2): *Satisfação em sequência de n1 ou n2, ou de ambos.*

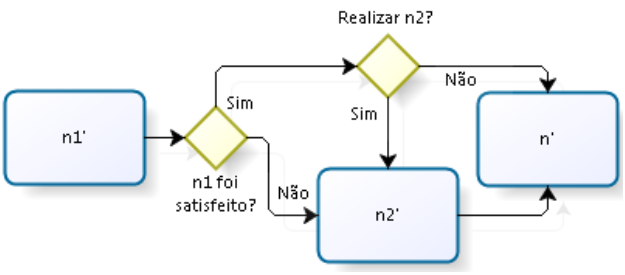

Figura 4: Processo BPMN para a regra **OR(n1;n2)**

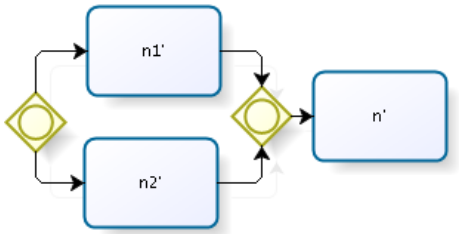

Figura 5: Processo BPMN para a regra **OR(n1#n2)**.

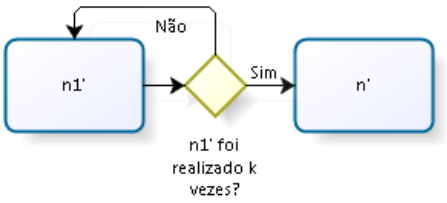

Figura 6: Processo BPMN para a regra **n1+k**.

Descrição: um *Fluxo de Sequência* segue de $n1$' para um *Desvio Exclusivo* com a descrição *"*$n1$' *foi satisfeito?"*. A partir dele, um *Fluxo de Sequência* nomeado de *"Não"* segue para $n2$', e outro nomeado de *"Sim"* segue para um outro *Desvio Exclusivo* com a descrição "Realizar $n2$'?". Este último desvio possui um *Fluxo de Sequência* nomeado de *"Não"* seguindo para n', e outro nomeado de *"Sim"* para $n2$'. A atividade $n2$' possui um *Fluxo de Sequência* que segue para n'. Tal representação abrange as 3 opções: apenas $n1$ será realizado, apenas $n2$ será realizado ou ambos serão realizados, satisfazendo a instância n.

4. n possui a regra OR(n1#n2): *Satisfação paralela de n1 ou n2, ou de ambos.* Tem-se 3 opções:

   - Apenas $n1$ é realizado;
   - Apenas $n2$ é realizado;
   - Ambos são satisfeitos paralelamente.

   Descrição: a partir de um *Desvio Inclusivo*, um *Fluxo de Sequência* segue para $n1$' e outro para $n2$'. O fluxo dos dois caminhos convergem em outro *Desvio Inclusivo*. O último *Desvio Inclusivo* possui *Fluxo de Sequência* seguindo para n'. Tal representação define que pelo menos um dos fluxos paralelos será seguido com sucesso, abrangendo os 3 casos citados e assim garantindo a satisfação de n.

5. n possui a regra n1+k: *n1 deve ser satisfeito k vezes, com k>0.*

   Descrição: a partir de n1', um *Fluxo de Sequência* segue para um *Desvio Exclusivo* com a descrição "$n1$' foi realizado k vezes?". Este desvio possui um *Fluxo de Sequência* nomeado de *"Não"* para $n1$', e um outro nomeado de *"Sim"* para n'. Tal representação define que $n1$ será realizado k vezes, satisfazendo a instância n.

6. n possui a regra n1#k: *Satisfação paralela de k instâncias de n1, com k>0.*

   Descrição: a partir de um *Desvio Paralelo*, k *Fluxos de Sequência* seguem para k instâncias de $n1$'. Todas as k instâncias convergem em outro *Desvio Paralelo*. O último *Desvio Paralelo* possui *Fluxo de Sequência* seguindo para n'. Tal representação define que k instâncias de $n1$ serão realizadas paralelamente, satisfazendo a instância n.

7. n possui a regra n1@k: *Máximo* $k-1$ *tentativas de satisfazer n1, com k>0.*

   Descrição: a partir de $n1$', um *Fluxo de Sequência* segue para um *Desvio Exclusivo* com a descrição "$n1$' foi satisfeito?". Este desvio possui um *Fluxo de Sequência* nomeado de *"Sim"* para n', e outro nomeado de *"Não"* para um outro *Desvio Exclusivo* com a descrição "Tentativa $< k$?". O último *Desvio Exclusivo* possui um *Fluxo de Sequência* nomeado de *"Sim"* para $n1$', e outro nomeado de *"Não"* para um *Evento Final*. Para que a instância n seja satisfeita, tal representação define que $n1$ deve ser satisfeito em no máximo k-1 tentativa(s).

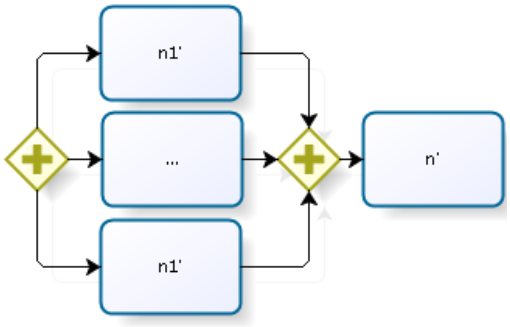

Figura 7: Processo BPMN para a regra **n1#k**.

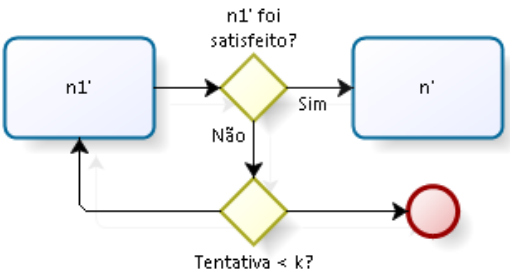

Figura 8: Processo BPMN para a regra **n1@k**.

8. n possui a regra opt(n1): *A satisfação de n1 é opcional.*

   Descrição: a partir de um *Desvio Exclusivo* com a descrição "Realizar $n1$'?", um *Fluxo de Sequência* nomeado de *"Sim"* segue para *n1'*, e outro nomeado de *"Não"* segue para n'. A atividade *n1'* possui *Fluxo de Sequência* seguindo para n'. Tal representação torna opcional a realização de $n1$.

9. n possui a regra try(n0)?n1:n2: *Se n0 for satisfeito, então n1; Senão, n2.* Define-se $n0$ como sendo um objetivo ou tarefa de um CRGM, e $n0$' é a Atividade equivalente a $n0$ em um processo de negócio.

   Descrição: $n0$' possui *Fluxo de Sequência* seguindo para um *Desvio Exclusivo* com a descrição "*n0' foi satisfeito?*". A partir deste desvio, um *Fluxo de Sequência* nomeado de *"Sim"* segue para $n1$', e outro nomeado de *"Não"* segue para $n2$'. As atividades $n1$' e $n2$' possuem *Fluxos de Sequência* que convergem em um outro *Desvio Exclusivo*. O último *Desvio Exclusivo* possui *Fluxo de Sequência* seguindo para n'. Tal representação define que se $n0$ for satisfeito, então $n1$ também será. Senão, $n2$ será satisfeito.

10. n possui a regra n1|n2: *Satisfação alternativa de n1 ou n2, nunca de ambos.*

    Descrição: a partir de um *Desvio Exclusivo*, com X marcado em seu centro (indicando que é um "Ou Exclusivo"), um *Fluxo de Sequência* segue para $n1$' e outro para $n2$'. As atividades $n1$' e $n2$' possuem *Fluxos de Sequência* que convergem em um outro *Desvio Exclusivo* com X marcado no centro. O último *Desvio Exclusivo*, com X marcado em seu centro, possui *Fluxo*

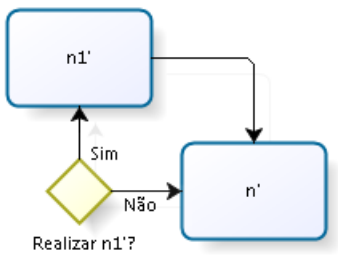

Figura 9: Processo BPMN para a regra **opt(n1)**.

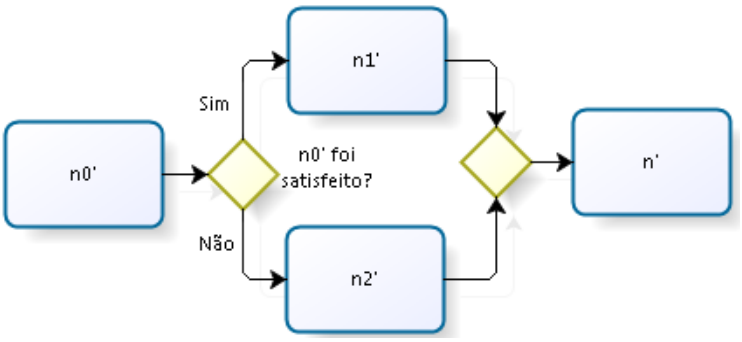

Figura 10: Processo BPMN para a regra **try(n0)?n1:n2**.

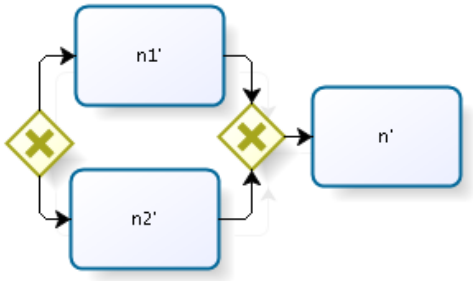

Figura 11: Processo BPMN para a regra **n1|n2**.

de Sequência seguindo para *n*'. Tal representação define que apenas *n*1 ou *n*2 será realizado, satisfazendo a instância *n*.

11. *n* depende de um contexto *C*: é necessário fazer a verificação do contexto antes de começar o processo de satisfação de *n*.

    Descrição: A partir de um *Desvio Exclusivo* com a descrição *"C?"*, um *Fluxo de Sequência* nomeado de *"Sim"* segue para *n*', e outro nomeado de *"Não"* segue para um *Evento Final*. Tal representação define que é necessário um contexto *C* antes de começar a realizar *n*. Caso contrário, tal fluxo é encerrado, evitando desperdício de esforço.

12. *n* possui a regra: *Sem ação*. A regra é representada por *skip*. A representação em BPMN redireciona o fluxo de atividades para *n*' ou para onde a instância do CRGM indicar, assim como no CRGM. Tal representação define que o fluxo será seguido conforme o indicado no CRGM.

13. *n* não possui a regras: se *n* é um nó folha, a atividade *n* irá aparecer por meio de alguma das conversões anteriores. Se *n* não for um nó folha, a atividade equivalente ao nó filho de *n* possui *Fluxo de Sequência* seguindo para n'.

14. *n* é o Objetivo Principal: um *Evento Inicial* possui *Fluxo de Sequência* seguindo para *n*'. Por sua vez, a atividade *n*' possui *Fluxo de Sequência* seguindo para *Evento Terminal*. Tal representação define o incício do processo. Ela também define que o objetivo principal *n* foi satisfeito e o processo é encerrado.

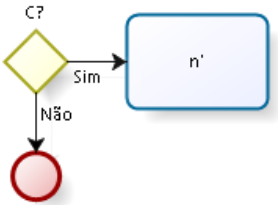

Figura 12: Contexto

3.3 Ferramenta de Conversão e Implementação

Assim como em [3], a ferramenta de conversão utiliza ANTLR (*Another Tool for Language Recognition*), com a mesma gramática definida em seu trabalho. ANTLR emprega como entrada os modelos criados no GODA, com a linguagem TAOME, para construir árvores de expressão por meio da gramática que define as regras do RGM e do CGM. Por este motivo, para gerar um modelo BPM a partir de um CRGM, serve-se da árvore de expressão gerada pelo ANTLR, cuja estrutura define o processo de negócio equivalente.

O formato da saída é textual. Ela descreve os elementos gráficos da notação BPMN 2.0 e como eles se relacionam, permitindo a modelagem gráfica do processo gerado sem quaisquer alterações na estrutura do resultado. O resultado é mostrado dentro da própria ferramenta. Melhores formatos de saída serão apresentados em versões futuras deste trabalho.

4 Exemplo de Conversão

Para uma melhor compreensão da prática do processo de conversão, será descrito abaixo um exemplo de conversão do CRGM da Figura 1 em um modelo em notação BPMN.

Segundo o modelo CRGM, para que se possa alcançar o objetivo deve-se satisfazer primeiro $G1$ para depois satisfazer $G2$. A satisfação de $G1$ requer a execução exclusiva de $G3$ ou $G4$. Após $G1$ ser satisfeito, o processo segue para iniciar o fluxo de realização de $G2$, que depende do contexto $C1$. Pode-se ver na Figura 13 o resultado da transformação.

A transformação permite uma validação da execução das tarefas e satisfação dos objetivos conforme. No modelo de objetivos, a noção de fluxo de execução não está tão explícita, o que permite erros de modelagem sem a devida facilidade de validação visual. No entanto, por meio da estrutura de um processo de negócio como viabilizamos por meio deste trabalho, torna-se mais clara e objetiva como deve ser a execução do modelo de objetivos em um sistema dinâmico. A linguagem utilizada para representar a conversão dentro do GODA tornou, portanto, viável a representação conforme a notação BPMN 2.0. Tal notação padrão é importante, uma vez que evita problemas de compatibilidade de diferentes plataformas. Dessa forma, o resultado gerado por meio da nossa transformação pode revelar a boa formação do CRGM e sua respectiva validação na execução dos objetivos.

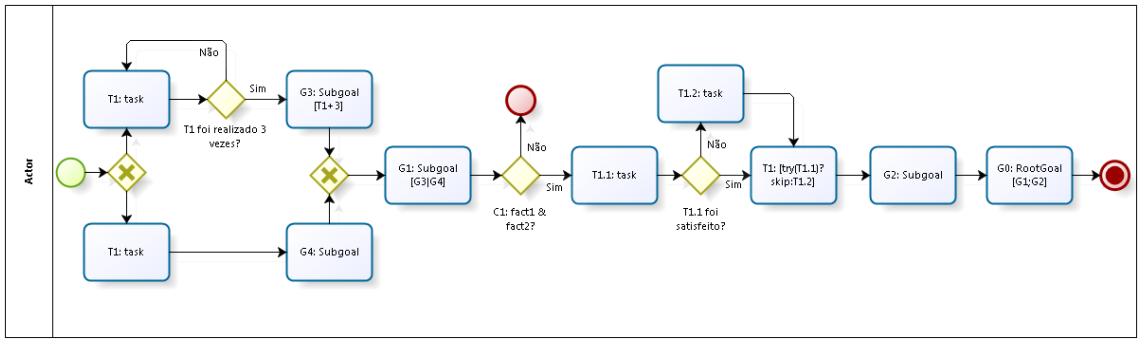

Figura 13: Modelo em BPMN gerado a partir do CRGM da Figura 1

5 Resultados Alcançados e Conclusão

Para a realização do trabalho, primeiro buscou-se fundamentação para reduzir as principais diferenças entre os dois modelos CRGM e BPMN, baseando-se na definições dos elementos apresentados no trabalho em [1], além da semântica e sintaxe do modelo de objetivos. Uma vez definida tal equivalência, testes como o da Figura 14 auxiliram na corretude da equivalência construída após a implementação da ferramenta, ao final da proposta do trabalho.

Por meio dos resultados obtidos, concluiu-se que a estrutura fundamental dos modelos gerados tornou possível a representação de variação de contexto em processos de negócio, permitindo assim uma análise mais confiável antes de uma organização iniciar a implementação de um sistema

representado por meio de objetivos. No entanto, vale observar que futuras versões do atual trabalho podem utilizar modelo multi-objetivos. Dessa forma, será possível maximizar a tradução dos elementos presentes tanto na linguagem TAOME quanto na linguagem BPMN. Pretende-se também melhorar a visualização dos processos gerados.

O frequente uso do BPM torna viável a análise de requisitos para apoiar-se em um processo de negócio que seja válido e confiável. Como discutido ao longo deste trabalho, processos derivados de modelos de objetivos bem estruturados são mais confiáveis e buscam reduzir a violação de requisitos. Deve-se também considerar que analistas podem errar ao tentar realizar a conversão manual proposta por este trabalho. Automatizar o processo de conversão pode não só auxiliar efetivamente facilitando o trabalho de um analista, como pode otimizar seu trabalho dependendo da necessidade do cliente. Portanto, a integração da funcionalidade ao GODA fornece um suporte adequado à criação de processos mais confiáveis em ambientes com variações de contexto.

Referências

[1] O. M. Group, "Business process model and notation (bpmn)," *Documents Associated With Business Process Model And Notation (BPMN) Version 2.0*, 2011.

[2] A. van Lamsweerde, "Goal-oriented requirements engineering: a guided tour," *Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on. pp*, p. 249–262, 2001.

[3] D. F. Mendonça, R. A. Genaína Nunes Rodrigues, and L. B. Vander Alvesa, "Goda: A goal-oriented requirements engineering framework for runtime dependability analysis," *Information and Software Technology*, 2016.

[4] M. Morandini, D. C. Nguyen, A. Perini, A. Siena, and A. Susi, "Tool-supported development with tropos: The conference management system case study," *Proceedings of the 8th International Conference on Agent-oriented Software Engineering VIII. AOSE'07. Springer-Verlag, Berlin, Heidelberg, pp*, p. 182–196, 2008.

[5] A. Dardenne, A. van Lamsweerde, and S. Fickas, "Goaldirected requirements acquisition," *Science of Computer Programming*, vol. 20, no. 1, p. 3–50, 1993.

[6] E. S.-K. Yu, ". modelling strategic relationships for process reengineering," *UMI Order No. GAXNN-02887 (Canadian dissertation)*, 1996.

[7] Bresciani, P. P., G. A., G. P., and J. F., Mylopoulos, "Tropos: An agent-oriented software development methodology," *Autonomous Agents and Multi-Agent Systems*, vol. 8, no. 3, p. 203–236, 2004.