Mantendo a consistência entre diagramas de Casos de Uso e modelos iStar com apoio de suporte computacional

Diego D.B. Thedoldi
Unioeste
Cascavel, PR Brasil
diego.thedoldi@unioeste.br

Victor F.A.Santander
Unioeste
Cascavel, PR Brasil
victor.santander@unioeste.br

Vitor L.C.Gilnek
Unioeste
Cascavel, PR Brasil
vitor.gilnek@unioeste.br

Alexssandro Morgenroth
Unioeste
Cascavel, PR Brasil
alexssandro.morgenroth@unioeste.br

***Abstract*— The computer tools used in the software development process should also support the consistency of the artifacts produced. If you use a particular artifact to create other artifacts, it is important that both remain consistent after changes. In previous work, a tool for deriving use cases in UML from iStar models was presented. However, changes to use cases with this tool are not mapped in iStar. Therefore, use cases and iStar models can become inconsistent during the development of requirements. With this in mind, in this article we present the new functionality that has been implemented to semi-automatically reflect changes made over time to the generated use cases back into the iStar models that serve as the basis. This functionality supports requirements engineers in the difficult task of maintaining consistency between requirements documentation artifacts.**

***Keywords*—software documentation; consistency, Requirements Engineering; Automatization**

***Resumo ou Resumen*— Ferramentas CASE devem também considerar apoiar engenheiros de software a garantir a consistência entre artefatos gerados. Em particular, se um artefato é usado como base para gerar outro é importante que ambos permaneçam consistentes após mudanças realizadas durante a evolução do software. Em trabalhos prévios têm sido proposta uma ferramenta computacional para apoiar a derivação de casos de uso UML a partir de modelos organizacionais iStar. Contudo, mudanças posteriores realizadas nos modelos de casos de uso não eram refletidas nos modelos iStar usados como base, gerando inconsistências na documentação de requisitos. Considerando este problema, regras de garantia de consistência (RGCs) foram propostas e implementadas na ferramenta, permitindo que engenheiros de requisitos possam fazer mudanças em modelos de casos de uso e, de forma semiautomática, atualizar os modelos iStar associados. Esta funcionalidade é apresentada neste artigo.**

***Palavras-chave*—documentação de software; consistência, engenharia de requisitos; automatização.**

I. Introdução

O desenvolvimento de software envolve várias etapas cruciais que devem ser levadas a cabo de forma sistemática e apoiada por ferramentas computacionais [16]. A primeira etapa, denominada de engenharia de requisitos, está preocupada com a elicitação, análise e negociação, documentação, validação e gerenciamento de requisitos funcionais e não funcionais de sistemas computacionais [31]. Neste contexto, uma vertente que impulsiona práticas e técnicas no processo de engenharia de requisitos vem recebendo destaque. Denomina-se engenharia de requisitos orientada a objetivos, do inglês, *goal oriented requirements engineering* (GORE) [6, 32, 34]. Esta vertente propõe que requisitos podem ser desenvolvidos em duas fases com focos diferentes, a *early requirements*, a qual trata mais do entendimento do problema, organização, suas necessidades, processos de negócio, entre outros e a *late requirements*, a qual trata do detalhamento dos requisitos de forma a serem já considerados para as etapas posteriores do desenvolvimento. Uma das técnicas da GORE utilizada para a fase de *early requirements* é a iStar [19, 20]. Esta técnica permite modelar ambientes organizacionais considerando as intencionalidades dos atores envolvidos. As intencionalidades são expressas na forma de objetivos, tarefas, recursos e qualidades.

No entanto, para a etapa de *late requirements*, requisitos são comumente documentados usando técnicas da UML [22], em particular casos de uso [9]. Vislumbrando benefícios da integração dessas duas técnicas no processo de engenharia de requisitos, em trabalhos prévios têm sido propostas diretrizes para essa integração [18] bem como suporte computacional para apoiar esse processo. Mais especificamente, uma ferramenta computacional denominada Jgoose (*Java goal into object oriented standard extension*) [23] tem sido desenvolvida para apoiar o processo de derivação de casos de uso UML a partir de modelos iStar. Contudo, no uso da ferramenta, percebeu-se um problema recorrente na engenharia de software, manter a consistência entre artefatos [8, 9, 21] gerados após a natural evolução desses artefatos.

Quando modelos iStar são modificados, basta usar novamente a funcionalidade da ferramenta (opção "*Generate Use Cases*" no editor de modelos iStar) para gerar um novo modelo de casos de uso, garantindo assim a consistência entre ambos. O problema está em que engenheiros de requisitos podem necessitar modificar o modelo de casos de uso de

forma direta, sem alterar antes o modelo organizacional iStar associado. Isto pode ocorrer por diversos motivos, entre eles, adicionar novos requisitos funcionais, modificar requisitos já elicitados, realizar correções, etc. Entretanto, neste contexto, mudanças realizadas nos casos de uso na Jgoose não eram refletidas de forma automatizada nos modelos iStar. Este processo deveria ser realizado manualmente necessitando de conhecimento e tempo considerável para realizá-lo.

Desta forma, mais recentemente, foram criadas regras de garantia de consistência (RGCs) [10], as quais foram incorporadas à ferramenta Jgoose para permitir o processo de derivação Casos de uso-> iStar de forma semi-automatizada. Assim, usuários da ferramenta podem agora fazer a derivação em ambos sentidos iStar <-> Casos de Uso. Isto permite que, principalmente, profissionais envolvidos na documentação de requisitos possam evoluir modelos de casos de uso e refletir essas alterações em modelos iStar, mantendo assim ambos os modelos consistentes. Neste sentido, considera-se que os potenciais usuários da ferramenta são todos aqueles envolvidos na elaboração da documentação de requisitos de sistemas (ou leitura desse documento), entre os quais destacam-se, engenheiros de requisitos, analistas de negócio, arquitetos, projetistas, programadores, testadores, entre outros, além de usuários e clientes do software, com conceitos básicos sobre os artefatos envolvidos. Assim, neste trabalho apresenta-se a ferramenta Jgoose e, em particular, a funcionalidade implementada visando apoiar a manutenção da consistência entre artefatos. Em [38] também é possível acessar um vídeo sobre a ferramenta. Além desta seção, na seção II apresentam-se os elementos essenciais para o entendimento da funcionalidade implementada; na seção III, esta nova funcionalidade é descrita com exemplos; na seção IV, são apresentadas algumas limitações e trabalhos relacionados; na seção V, são realizadas as considerações finais do trabalho.

II. Referencial Teórico

A. *iStar e Casos de Uso*

A técnica iStar [4, 20], no processo de descrição do ambiente organizacional, apresenta dois modelos: *Strategic Dependence* (SD) e *Strategic Rationale* (SR). O primeiro, SD, tem seu foco na descrição de relacionamentos de dependências dos diversos atores. O SD tem sua definição por meio de ligações e de nós. No SD, os atores em seu ambiente são representados pelos nós, já as dependências entre eles são apresentadas pelas ligações. A entidade ator, com a finalidade de alcançar seus objetivos, desenvolve ações, tarefas e obtém recursos neste contexto organizacional. Na relação de dependência presente no SD tem-se o *Depender*, caracterizado pelo ator que possui dependência em outro, já o *Dependee* é responsável nesta relação de dependência, o qual deve cumpri-la. Ainda nesta relação há o foco de dependência, o *Dependum*, caracterizado pelo elemento ou objeto de dependência entre atores. Logo, ocorrerá a situação *Depender→Dependum→Dependee*. No caso do SR, o mesmo possibilita a expressão de meios e formas alternativas do *Dependee* necessários para satisfazer suas dependências. A Fig. 1 apresenta um exemplo desta modelagem em um cenário genérico para compra e venda em uma organização. Os atores possuem duas categorias, o de *Agent* (veja Jaqueline na Fig. 1), que possui expressões de um humano, de um departamento ou de uma organização - concretas - ou ainda o de *Role* (veja Gerente na Fig. 1). Há ainda circunstâncias - estágio de modelagem ou cenário em questão - em que a noção de *actor* (ator genérico) pode ser utilizada (Cliente na Fig. 1). Há dois tipos de relacionamentos entre atores, os quais são:

a. *is-a*: representa o conceito de generalização/especialização. Somente os atores do tipo role e genérico (*actor*) podem ser especializados (Gerente e Funcionário na Fig. 1).
b. *participates-in*: representa algum tipo de associação, além da generalização/especialização, entre dois atores. A relação ocorre entre os atores Jaqueline (*Agent*) e Gerente (*Role*) expressando que o agente Jaqueline desempenha o papel de Gerente na Fig. 1.

As intencionalidades de atores usadas tanto no modelo SD (como *dependum*) quanto no SR (razões internas) podem ser dos tipos a seguir; a) *Goal* (objetivo): um estado do sistema que o ator deseja alcançar e que possui requisitos claros para completá-lo; b) *Quality* (qualidade): atributo em que a entidade ator almeja uma realização, em algum nível. c) *Task* (tarefa): tem relação com o ator, mais especificamente com anseio de desenvolver suas ações que, comumente objetivam alcançar um *Goal*; d) *Resource* (recurso): trata-se de uma exigência de um ator para o desenvolvimento de uma tarefa ligada a ele, o qual pode ser considerado como uma entidade (informativa ou física).

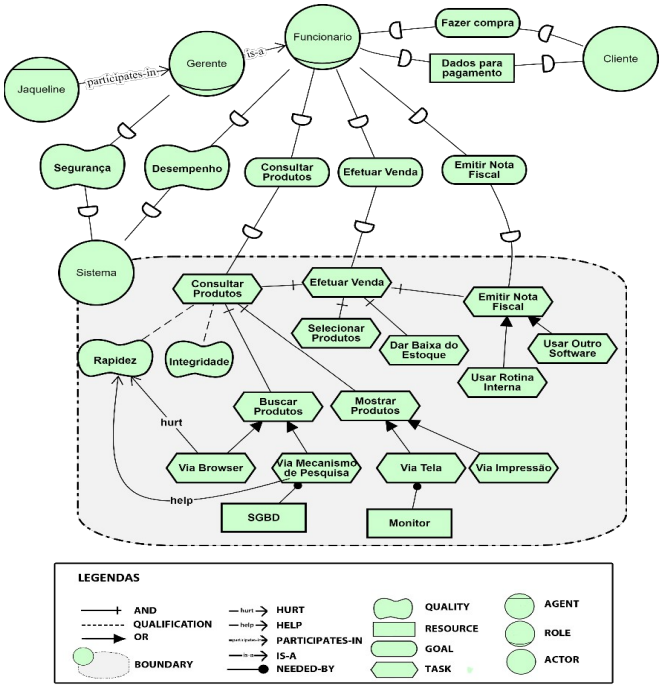


Fig. 1. Exemplo de modelo iStar.

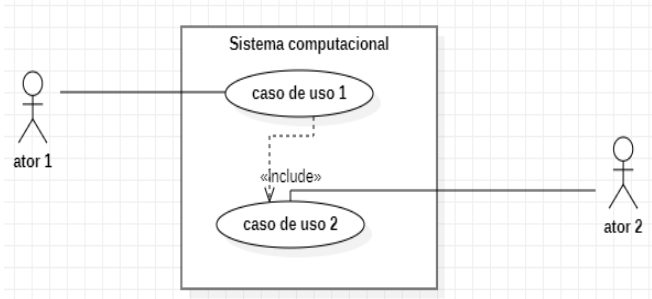
Em relação às ligações entre intencionalidades, os tipos são: *refinement* (refinamento), *needed-by* (necessário em), *contribution* (contribuição) e *qualification* (qualificação). O *refinement* é uma relação n-ária que relaciona um dos pais a um ou demais filhos. Existem dois tipos de refinamento: a) *AND* - os pais são satisfeitos quando o cumprimento dos n filhos (todos) ocorre; decomposição da tarefa Efetuar Venda em Consultar Produtos, Selecionar Produtos, Dar Baixa no Estoque e Emitir Nota Fiscal na Fig. 1; b) *OR* Inclusivo - os pais são cumpridos quando o cumprimento dos n filhos (ao menos um) ocorre; decomposição da tarefa Buscar Produtos em Via Browser *OR* Via Mecanismo de Pesquisa na Fig. 1. Já a ligação *needed-by* é usada para expressar que a tarefa Via Mecanismo de Pesquisa necessita do recurso SGBD; A ligação *contribution* indica que a tarefa Via Mecanismo de Pesquisa ajuda (*help*) na satisfação da qualidade Rapidez. Por fim, a relação de qualificação indica, por exemplo, que a tarefa Consultar Produtos deve ser realizada considerando a qualidade Rapidez.

Por outro lado, requisitos funcionais podem ser modelados sob a visão do paradigma de desenvolvimento orientado a objetos através da *Unified Modelling Language* (UML) [22]. Dentre os diversos diagramas e descrições definidos pela UML que podem ser utilizados no processo de engenharia de requisitos, destacam-se os Casos de Uso. Os Casos de Uso já são bem conhecidos e especificam o comportamento de um sistema, ou parte dele, sendo uma descrição de um conjunto de sequências de ações, incluindo as possíveis variações nestas ações, que produzem um resultado observável de valor para um ator [7, 22]. Assim, casos de uso são representados tanto na forma diagramática [22] quanto textual [3].

B. RGCs

De que forma os elementos de um modelo de casos de uso são considerados na transformação em modelos iStar é definido pelas Regras de Garantia de Consistência (RGCs). Estas regras não foram construídas apenas considerando o processo inverso, conforme as diretrizes propostas em [14], as quais permitem derivar casos de uso a partir de modelos iStar. O processo foi mais abrangente, partindo pela revisão dessas diretrizes bem como estudo aprofundado de casos de uso UML e técnica iStar. Na modelagem de casos de uso, foram incorporados alguns elementos opcionais como "*system boundary*" (RGC 3) e "*secondary actor*" (RGC 8) bem como análise de situações específicas, antes não identificadas, como na RGC 7, 11 e 12. Isto implicou também a inclusão de novos elementos no editor de casos de uso da Jgoose (ver figura 6). Em relação à técnica iStar, foram avaliadas também as mudanças na versão 2.0, utilizando as novas nomenclaturas já no processo de mapeamento (RGC 5, 6 e 7). Para os novos elementos da iStar 2.0 como "*needed-by*" e "*qualification*", não foram encontrados mapeamentos a partir de modelos de casos de uso. Segue breve descrição das RGCs.

A RGC 1 estabelece que as alterações nos nomes dos elementos no diagrama de caso de uso devem ser refletidas no modelo iStar correspondente para promover uniformidade de nomenclatura. A RGC 2 define que um ator primário inserido no modelo de casos de uso deve ser mapeado como um ator genérico no modelo iStar. A RGC 3 determina que a representação do limite do sistema (retângulo no qual insere-se os casos de uso – "*system boundary*") no modelo de casos de uso deve ser mapeado para um ator genérico com mesmo nome no iStar. A RGC 4 indica que todo caso de uso será registrado como uma dependência no modelo iStar (objetivo, tarefa ou recurso), devendo essa dependência ser satisfeita no ator que representa o sistema computacional, por um elemento do tipo tarefa e esse elemento deve estar dentro da fronteira do ator sistema no modelo iStar. Por padrão, a dependência inicial gerada é do tipo objetivo, mas pode ser alterada para



tarefa ou recurso, de acordo com a análise do engenheiro de requisitos. A RGC 5 indica que o estereótipo «include» no modelo de casos de uso será mapeado para uma relação *AND* no modelo iStar, com um *link* de dependência para o ator que faz a associação com o caso de uso. Veja figuras 2 e 3.

Fig. 2. Exemplo <<include>> - RGC 5.

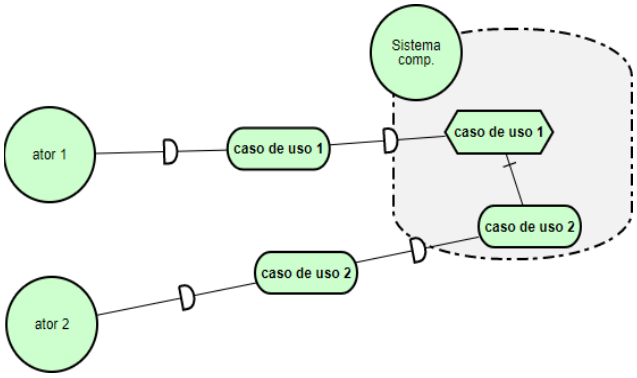


Fig. 3. Exemplo de modelo iStar gerado para a RGC 5.

A RGC 6 indica que o estereótipo «*extend*» no modelo de casos de uso deve ser mapeado para uma relação *OR* no modelo iStar, com um *link* de dependência para o ator que faz a associação com o caso de uso. A RGC 7 define como sub caso de uso todo aquele caso de uso que não possui nenhuma ligação com um ator e é um filho incluído («*include*») ou estendido («*extend*») de um ou mais casos de uso. Esse sub caso de uso será mapeado no modelo iStar como uma tarefa *AND* ou *OR* ligada ao caso de uso pai, dentro do ator sistema. A RGC 8 indica que um ator secundário «*secondary actor*» inserido no modelo de casos de uso, deve ser mapeado como um novo ator genérico no modelo iStar. Este ator secundário deve gerar uma nova relação de dependência com o ator sistema, sendo ele o *dependee* na relação ligada a uma tarefa ou objetivo dentro do sistema. A RGC 9 indica que um *link* de associação entre um ator e um caso de uso gera uma ligação de dependência no modelo iStar. A RGC 10 indica que uma nova relação de generalização entre atores no modelo de casos de uso deve ser mapeada para uma ligação do tipo *IS-A* entre atores no modelo iStar. A RGC 11 trata da exclusão de qualquer elemento no modelo de casos de uso e como isto deve ser refletido no modelo iStar. Estabelece subregras de garantia de consistência (SRGC) para nortear esse mapeamento. A SRGC 11.1 indica que todo elemento do diagrama de caso de uso removido de um caso de uso é necessariamente excluído do modelo iStar; a SRGC 11.2 indica que a exclusão do "limite do sistema" leva à remoção de todos os elementos incluídos nesse limite; a SRGC 11.3 indica que a exclusão de um ator no modelo de caso de uso deve gerar a remoção de todas as suas relações de associação, implicando na remoção de todas as relações de dependência associadas no modelo iStar. Os casos de uso ligados a este ator também serão removidos desde que satisfaçam a SRGC 11.4; a SRGC 11.4 indica que antes de remover um caso de uso, deve-se verificar se existe algum ator ligado a esse caso de uso. Na hipótese de haver um ator ligado a esse caso de uso, deve-se verificar se esse caso de uso possui ligação com algum outro elemento do diagrama. Se não possuir nenhuma ligação, o caso de uso deve ser removido do diagrama. Na hipótese desse caso de uso possuir relacionamento de associação com outro ator, o caso de uso não deve ser removido.

Por fim, a RGC 12 fornece diretrizes adicionais para manter a coerência entre os modelos. A SRGC 12.1 estabelece que em um refinamento de uma de tarefa no modelo iStar, um pai só pode ser refinado via *AND* ou *OR*, mas não ambos simultaneamente. Já a SRGC 12.2 estabelece que, em um caso de uso que possua os dois tipos de relacionamentos «*include*» e «*extend*» simultaneamente, ambos os relacionamentos, em um primeiro passo, devem ser mapeados para o modelo iStar usando a relação *AND*. Na sequência, as relações do tipo «*extend*» originais devem ser mapeadas em uma subtarefa da relação *AND* gerada no primeiro passo.

C. *A ferramenta Jgoose*

A ferramenta Jgoose [13, 23] é um software que tem como objetivo suportar a derivação de casos de uso de forma visual e textual a partir de modelos organizacionais elaborados com o *framework* iStar 1.0 [19, 20] ou BPMN [5]. Em particular, a ferramenta automatiza a obtenção de casos de usos via diretrizes já propostas em [14]. A partir das diretrizes, a ferramenta interpreta os modelos organizacionais, gerando os casos de uso UML [7, 22] bem como suas descrições textuais, seguindo o *template* adaptado de [3]. Desta forma, a ferramenta Jgoose gera casos de uso com base nas intencionalidades e dependências dos atores, em um determinado ambiente organizacional. Essencialmente, esta ferramenta é composta por editores específicos para iStar [35], BPMN [5], Casos de Uso [11] bem como das rotinas [13] que permitem mapear casos de uso a partir de modelo iStar e BPMN. Mais recentemente, esta ferramenta foi alvo de modificações para permitir o mapeamento Casos de uso -> iStar, permitindo que engenheiros de requisitos possam fazer alterações nos modelos de casos de uso, as quais podem ser mapeadas para os modelos iStar através da nova funcionalidade implementada. Esta ferramenta é de código aberto e está disponível em [37]. A figura 4 apresenta a tela inicial da ferramenta, a figura 5, a tela do editor de modelos iStar e a figura 6, o editor de casos de uso.

Fig. 4. Tela inicial da Jgoose.

No que tange à arquitetura da Jgoose, a mesma foi construída utilizando o padrão arquitetural MVC (*Model View Controller*) sendo particionada em quatro grandes pacotes, cuja estrutura interna segue o padrão mencionado. O pacote

denominado "iStar" está relacionado à importação de modelos iStar elaborados em outra ferramenta denominada OME [29], oriunda ainda da primeira versão da ferramenta, antes de possuir editor próprio para iStar; o pacote denominado "*log*" foi criado para gravar o *log* do sistema; o pacote denominado "E4J" (*Editors for Jgoose*) concentra todas as classes relacionadas aos editores para iStar e BPMN; e o pacote Jgoose, o qual utiliza os outros pacotes para apoiar a implementação das rotinas de mapeamento/derivação de iStar -> BPMN e IStar <-> Casos de uso. A figura 7 apresenta estes pacotes na plataforma Github. No que tange às tecnologias, a Jgoose foi construída utilizando a plataforma Netbeans, a linguagem de programação Java, o pacote Jgraph para a construção dos elementos gráficos, entre outras.

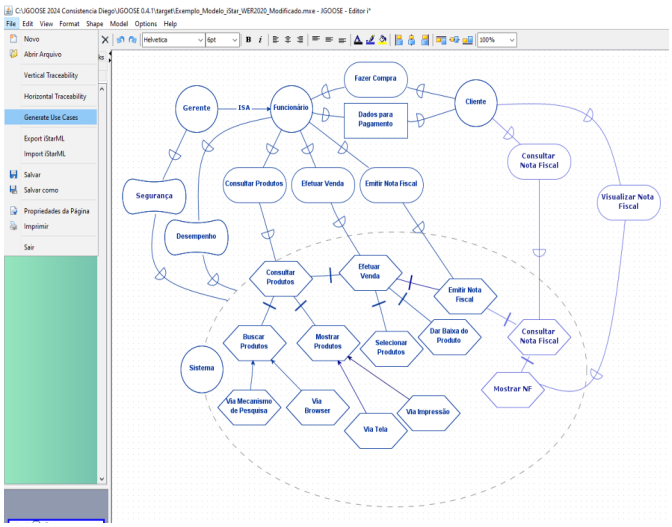


Fig. 5. Tela do editor de modelos iStar com exemplo.

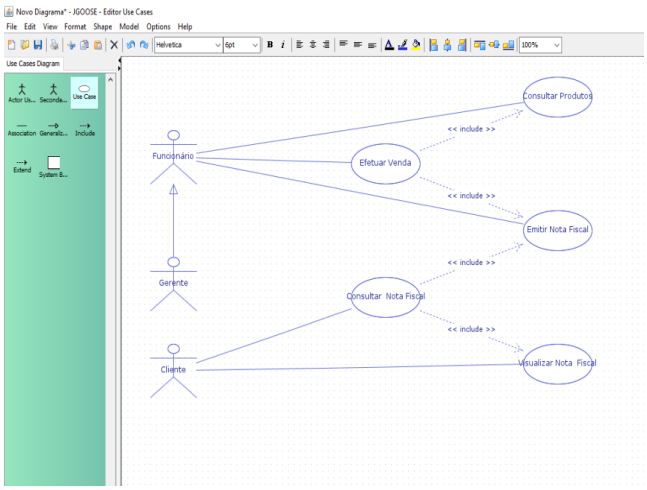


Fig. 6. Tela do editor de casos de uso.

III. A IMPLEMENTAÇÃO DAS RGCS NA FERRAMENTA JGOOSE

Este trabalho apresenta a funcionalidade para apoiar a manutenção da consistência dos modelos de casos de uso e modelos iStar. Por exemplo, ao criar um modelo de casos de uso (ver figura 6) a partir de um modelo iStar (ver figura 5), a ferramenta aplica as diretrizes propostas em [18]. Para isso, basta usar a opção "*Generate Use Cases*" na tela apresentada na figura 5. Esse é um exemplo genérico que considera uma organização que utiliza um sistema de compra e venda para atender clientes. Na engenharia de requisitos, defende-se que modelos organizacionais sejam desenvolvidos inicialmente [1, 33] e, em seguida, os requisitos funcionais e não funcionais [26] podem ser documentados de uma forma mais assertiva. Contudo, na maioria dos casos, inevitavelmente requisitos evoluem. Nesse contexto, engenheiros de requisitos podem desejar modificar o modelo de casos de uso gerado na figura 6. As RGCs apresentadas em [10] orientam esse processo (ver subseção B na seção III). Seguem situações de uso da ferramenta nesse contexto. Contudo, antes de apresentar as mudanças específicas nos modelos de Casos de Uso e sua correspondência nos modelos iStar, cabe ressaltar que após qualquer mudança, o usuário da ferramenta deve utilizar a opção "*Generate iStar*" no editor de Casos de uso conforme apresentado na figura 8.

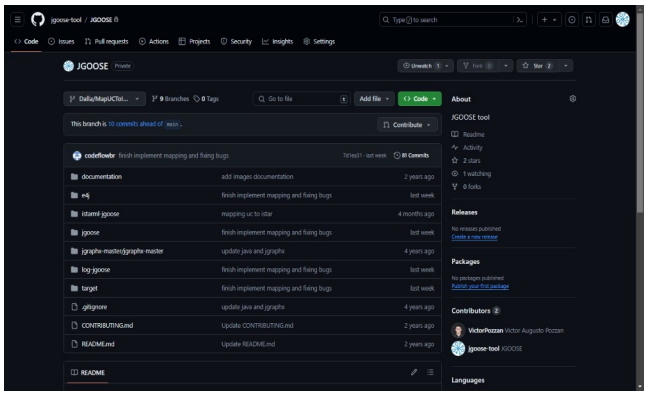


Fig. 7. Estrutura da Jgoose no Github.

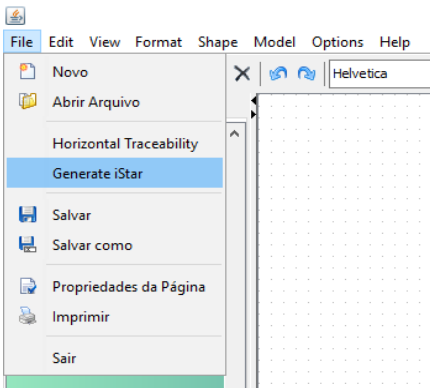


Fig. 8. Opção a ser utilizada para aplicar as RGCs.

Pode-se observar como seria a modificação do nome de um elemento no modelo de casos de uso, por exemplo, modificar o nome do ator primário "Cliente" por "Consumidor" (RGC 1) e adicionar um novo ator primário (RGC 2) denominado "Consumidor Especial" como uma especialização (RGC 10) do ator "Consumidor". Também pode-se associar (RGC 9) a este ator, um novo caso de uso chamado "Solicitar Desconto" (RGC 4) e, finalmente, utilizar

o elemento "*system boundary*" para incluir e agrupar os casos de uso do sistema (RGC 3). A figura 9 apresenta essas alterações realizadas e a figura 10, essas alterações refletidas no modelo iStar.

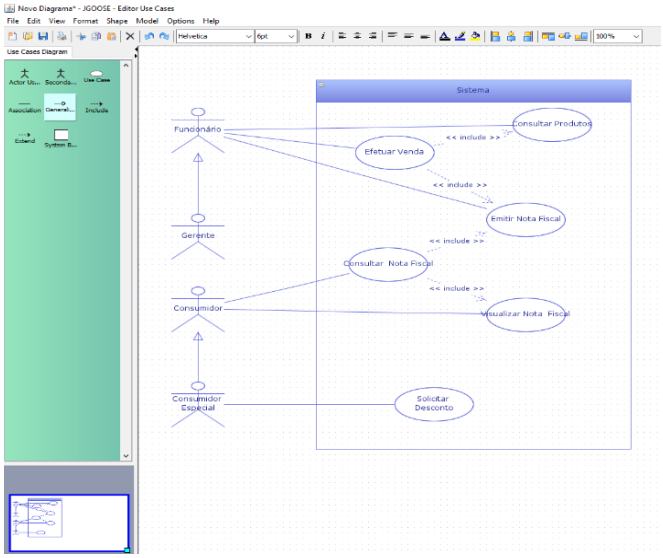


Fig. 9. Tela com alterações de casos de uso.

Outras modificações possíveis incluem a possibilidade de inclusão <<*include*>> de um caso de uso, como por exemplo, uma nova relação desse tipo entre o caso de uso "Solicitar Desconto" e um novo caso de uso denominado "Calcular Desconto" (RGC 5). Ainda, uma nova relação opcional <<*extend*>> pode ser criada, como "Imprimir Nota Fiscal" sendo uma extensão do caso de uso "Visualizar Nota Fiscal" (RGC 6). Veja que estes novos casos de uso também recaem na RGC 7 pois são sub casos de uso. A figura 11 apresenta essas alterações realizadas e a figura 12, essas alterações refletidas no modelo iStar.

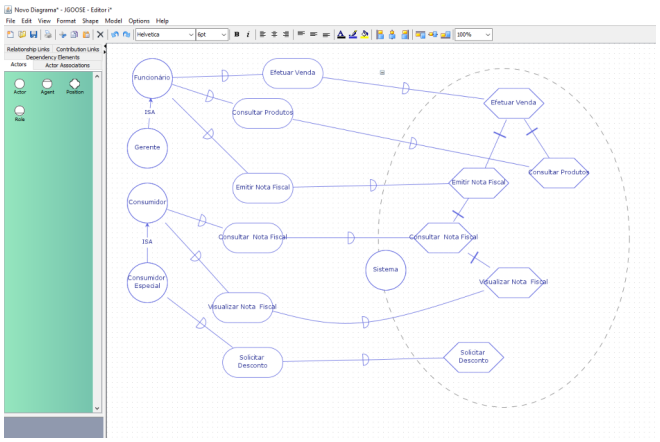


Fig. 10. Tela com o modelo iStar alterado.

Também é possível que o usuário da ferramenta adicione um ator secundário ("*secondary actor*") (RGC 8) ao modelo de casos de uso. Por exemplo, o ator secundário "Sistema Receita Federal" pode ser incluído como um ator de suporte à execução do caso de uso "Emitir Nota Fiscal". Também, o usuário pode excluir qualquer elemento do modelo (RGC 11 e subregras), o que deve resultar em um mapeamento consistente para o modelo de casos de uso. Por exemplo, ao excluir o ator "Gerente" e sua relação de herança com "Funcionário", a relação e ator excluídos devem também ser excluídos do modelo iStar correspondente. Por fim, a RGC 12 é aplicada quando, por exemplo, relações diferentes (<<*include*>> e <<*extend*>>) são ambas associadas a um caso de uso pai. No exemplo, isto ocorre ao acrescentar o caso de uso "Atualizar Cadastro" como uma extensão <<*extend*>> do caso de uso pai "Solicitar Desconto", o qual já possui uma relação de <<*include*>> com o caso de uso "Calcular Desconto". A figura 13 apresenta as alterações realizadas no modelo de casos de uso e a figura 14, essas alterações refletidas no modelo iStar.

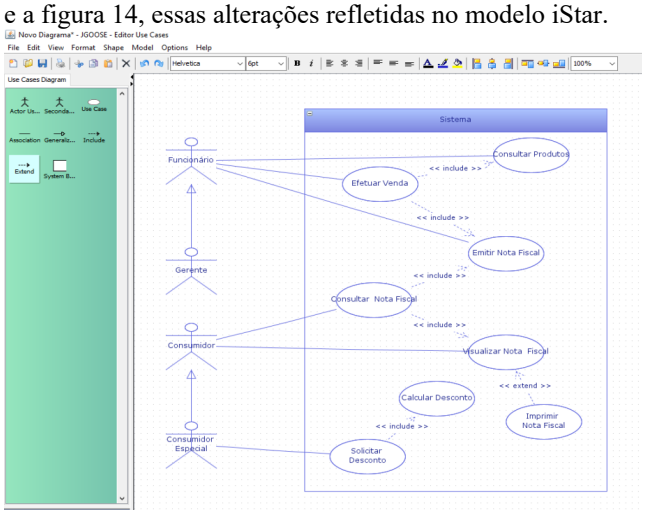


Fig. 11. Tela com modelo de casos de uso alterado.

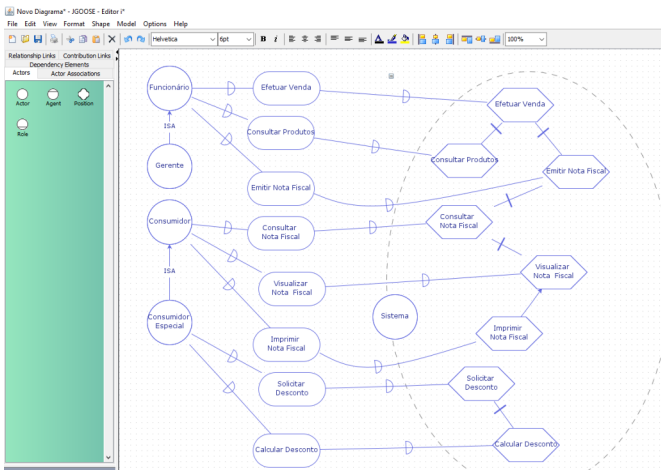


Fig. 12. Tela com modelo iStar alterado.

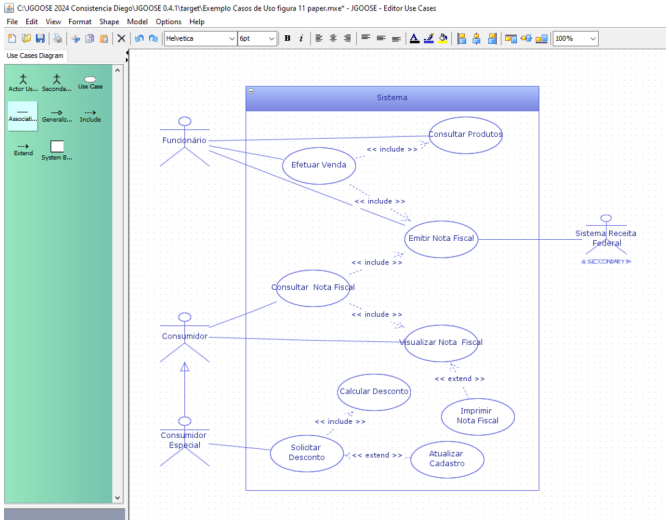


Fig. 13. Tela com modelo de casos de uso alterado.

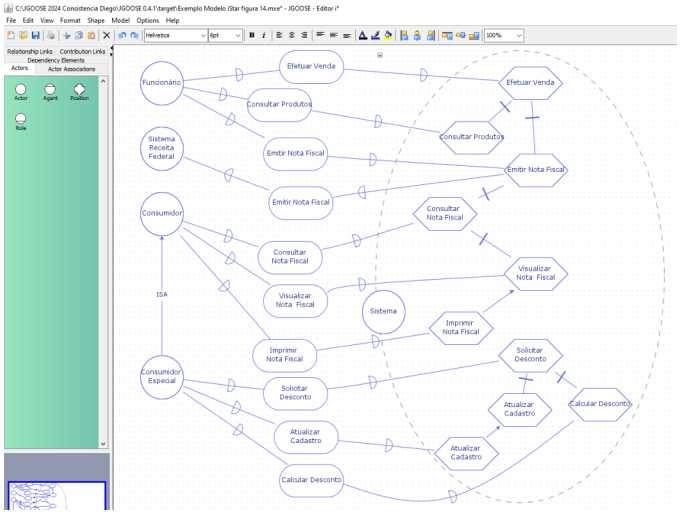


Fig. 14. Tela com modelo iStar correspondente.

IV. Trabalhos Relacionados, Limitações e Trabalhos em Andamento

Além da funcionalidade descrita neste artigo, a ferramenta Jgoose também engloba outras funcionalidades tais como gerar matrizes de rastreabilidade [12], derivar modelos de casos de uso a partir de modelos BPMN [5] e gerar descrições textuais de casos de uso a partir de modelos iStar e BPMN [13, 23].

No que tange às ferramentas relacionadas, pode-se destacar aquelas que permitem construir modelos iStar como as propostas em [28], em particular a ferramenta disponível em [27] e ferramentas que podem ser usadas para construir modelos UML, as quais incluem a elaboração de casos de uso como Astah UML, StarUML, Dia, Umbrella, entre outras. O único trabalho similar à nossa proposta é apresentado em [36]. Este trabalho apresenta uma abordagem para gerar automaticamente casos de uso a partir de modelos iStar 2.0 denominada Goal2UCM, com suporte computacional via ferramenta RM2PT. Apresenta regras de mapeamento e algoritmos para realizar o mapeamento. Entretanto, a ferramenta não possui editor para criar modelos iStar e utiliza arquivos .xmi como entrada para gerar casos de uso e diagramas de sequência. Vale também relacionar o trabalho apresentado em [2], o qual aponta a necessidade de sincronizar e cuidar da consistência entre modelos iStar e Casos de Uso. Os trabalhos relatados foram revisados para dar apoio na construção das RGCs implementadas na Jgoose.

É importante também destacar algumas limitações desta funcionalidade apresentada na Jgoose. O controle das várias versões que podem ser geradas tanto do modelo iStar quanto de Casos de uso é de responsabilidade do usuário. Para isso, a ferramenta permite salvar os modelos tanto no formato nativo .mxe (editável posteriormente) quanto em formatos de imagem (.png, .jpg, .tif, .svg, entre outros). Também é possível perceber nos exemplos apresentados que o modelo iStar original usado para derivar casos de uso, contém elementos que não são mapeados para casos de uso. Isto ocorre porque os mesmos não estão relacionados às funcionalidades do sistema. Estes elementos, no processo inverso de mapeamento de casos de uso -> iStar, não serão apresentados. Assim, é de responsabilidade do usuário adicionar manualmente esses elementos ou deixar explícito no documento de requisitos [25], porque eles não constam nessa versão do artefato.

Alguns trabalhos estão em andamento na Jgoose. O editor de modelos iStar está atualmente sendo modificado para incorporar as mudanças da iStar versão 2.0. Na versão atual, apenas a versão 1.0 é suportada. O processo de derivação de casos de uso a partir de iStar está sendo revisto considerando as RGCs propostas e já implementadas na Jgoose bem como a nova proposta que considera a última versão da iStar [30], a qual essencialmente afeta somente as descrições textuais dos casos de uso gerados. Estes trabalhos estão sendo realizados por acadêmicos de graduação e mestrado.

V. Considerações Finais

Neste trabalho foi apresentada a ferramenta Jgoose e, mais especificamente, uma das funcionalidades da mesma, a qual permite mapear mudanças realizadas em modelos de casos de uso para modelos organizacionais iStar. Isto é realizado de forma semiautomática pela ferramenta, aumentando a produtividade e motivação para manter modelos consistentes. Como trabalhos futuros, se destaca a necessidade de uma validação mais robusta da ferramenta, já que a mesma foi validada apenas em ambiente acadêmico [24]. Para isso, métodos da engenharia de software experimental [15, 17] podem apoiar fortemente esse processo. A ferramenta é de código aberto (*open source*) e está disponível para download em [37]. Contribuições e críticas da comunidade são bem-vindas. Também cabe ressaltar que os envolvidos no desenvolvimento da ferramenta são acadêmicos cujo apoio financeiro se restringe a bolsas governamentais.

Referências

[1] Fernanda M.R.Alencar et al. 2000. From early requirements modeled by the i* technique to later requirements modeled in precise uml. In: WER. [S.l.: s.n.], 2000. p. 92–108.

[2] Moshiur Bhuiyan et al. 2018. Integration of organisational models and uml use case diagrams. Journal of Computers, Academy Publisher, v. 13, n. 1, p. 1–18, 2018.

[3] Alistair Cockburn. 2001. Writing Effective Use Cases. Addison-Wesley, 2001. (Crystal collection for software professionals). ISBN 9780201702255. Disponível em: <https://books.google.com.br/books?id=VKJQAAAAMAAJ>.

[4] Fabiano Dalpiaz et al. 2016. istar 2.0 language guide. CoRR, abs/1605.07767, 2016. Disponível em: <http://arxiv.org/abs/1605.07767>.

[5] Alysson N. Girotto et al.2017. Uma proposta para derivar casos de uso a partir de modelos bpmn com suporte computacional. In: 36th International Conference of The Chilean Computer Science Society (SCCC 2017). [S.l.: s.n.], 2017.

[6] Jennifer Horkoff et al. 2019. Goal-oriented requirements engineering: an extended systematic mapping study. Requirements engineering, Springer, v. 24, p. 133–160, 2019.

[7] Ivar Jacobson. 2004. Object-Oriented Software Engineering: A Use Case Driven Approach. USA: Addison Wesley Longman Publishing Co., Inc., 2004. ISBN 0201403471.

[8] D. Meedeniya et al. 2019. Software artefacts consistency management towards continuous integration: A roadmap. International Journal of Advanced Computer Science and Applications, v. 10, p. 100–110, 04 2019.

[9] D. Monscheinet et al. 2021. Enabling consistency between software artefacts for software adaption and evolution. In: 2021 IEEE 18th International Conference on Software Architecture (ICSA). [S.l.: s.n.], 2021. p. 1–12.

[10] Diogo N. Paza. 2023. Mantendo a consistência na coevolução de modelos iStar e Casos de Uso na Engenharia de Requisitos. Dissertação (Mestrado), 2023. Centro de Ciências Exatas e Tecnológicas. Disponível em: <https://tede.unioeste.br/handle/tede/6747>.

[11] Diego Peliser et al. 2016. Integrating the E4J use cases to the JGOOSE tool. 35th International Conference of the Chilean Computer Science Society (SCCC), Valparaiso, Chile, 2016, pp. 1-11, doi: 10.1109/SCCC.2016.7836003.

[12] Victor A. Pozzan et al. 2020. Suporte ao rastreamento de requisitos na ferramenta JGOOSE, São Jose dos Campos: Workshop on Requirements Engineering - WER, 2020.

[13] Victor A. Pozzan and V.F.A. Santander. 2021. Evolução de uma ferramenta computacional que suporta o processo de derivação de casos de uso a partir de modelos bpmn e i*. In: 7º EAICTI - 7º Encontro Anual de Iniciação Científica e Inovação da Unioeste. Cascavel, Paraná. ISSN 2448-0681.

[14] Victor. F. A. Santander and Jaelson F.B Castro. 2002. Deriving use cases from organizational modeling. Proceedings IEEE Joint International Conference on Requirements Engineering, p. 32–39, 2002. Disponível em: <https://api.semanticscholar.org/CorpusID:18286317>.

[15] F. Shull et al. 2001. An empirical methodology for introducing software processes. ACM SIGSOFT Software Engineering Notes, ACM New York, NY, USA, v. 26, n. 5, p. 288–296, 2001.

[16] Ian Sommerville. 2011. I. Engenharia de software. Pearson Prentice Hall, 2011. ISBN 9788579361081.

[17] C. Wohlin et al. 2012. Experimentation in software engineering. [S.l.]: Springer Science & Business Media, 2012.

[18] R. Young. 2004. The Requirements Engineering Handbook. Artech House, 2004. (Artech House professional development and technology management library). ISBN 9781580536189.

[19] Eric Yu. 1997. Towards modelling and reasoning support for early-phase requirements engineering. In: Proceedings of ISRE '97: 3rd IEEE International Symposium on Requirements Engineering. [S.l.: s.n.], 1997. p. 226–235.

[20] Eric Yu. 1995. Modeling strategic relationships for process reengineering. In: Social Modeling for Requirements Engineering. [s.n.], 1995. Disponível em: <https://api.semanticscholar.org/CorpusID:62091749>.

[21] M. Zekkaoui and A. Fennan. 2013. Consistency management of heterogeneous software artifacts. International Journal of Computer Applications, v. 78, p. 35–41, 2013.

[22] G. Booch et al. 2017. UML: Guia do usuário, São Paulo: GENLTC, 2017.

[23] Gustavo C.L. Geraldino and Victor F. A. Santander. 2019. The JGOOSE Tool, Salvador: 12th International i* Workshop - iStar, 2019.

[24] Gustavo C. L. Geraldino et al. 2020. Avaliando o processo de derivação de casos de uso a partir de i* e BPMN e suporte computacional. In: 23rd Iberoamerican Conference on Software Engineering (CIbSE 2020) - RET Track, 2020, Curitiba.

[25] IEEE recommended practice for software requirements specifications. IEEE Std 830-1998, p. 1–40, 1998.

[26] Lawrence Chung et al. 2012. Non-functional

requirements in software engineering. [S.l.]: Springer Science & Business Media, 2012. v. 5.

[27] UFPE, C. DSM3-goals. Pacote de Ferramentas. [S.l.]: Centro de Informática da UFPE, 2020. <https://www.cin.ufpe.br/~jhcp/dsm3goals/index.html>. Accessed: 2024-05-28.

[28] iStar Wiki.org. http://istarwiki.org/tiki-index.php?page=i%2A+Tools. Accessed: 2024-05-28.

[29] OME Tool. https://www.cs.toronto.edu/km/ome/. Accessed: 2024-05-28.

[30] Bruno L. Casarotto et al. 2022. Using of i*(istar) 2.0 for improving the use cases derivation. IEEE Latin America Transactions, v. 20, n. 2, p. 198–207, 2022.

[31] G. Kotonya and I. Sommerville, (1998). Requirements Engineering: Processes and Techniques, USA: Wiley.

[32] E. Kavakli. 2002. Goal-oriented requirements engineering: A unifying framework. Requirements Engineering, Springer, v. 6, n. 4, p. 237–251, 2002.

[33] A.V. Lamsweerde. 2009. Requirements Engineering: From System Goals to UML Models to Software Specifications. 1st. ed. [S.l.]: Wiley Publishing, 2009. ISBN 0470012706.

[34] A. Lapouchnian. 2005. Goal-oriented requirements engineering: An overview of the current research. 01 2005.

[35] L.P. Merlim e al. 2015. Integrating the E4J editor to the JGOOSE tool. In: XVIII Workshop de Engenharia de Requisitos (WER), Universidad Ricardo Palma, Lima.

[36] Yilong Yang et al. 2021. Goal2UCM: Automatic Generation of Use Case Model from iStar Model. In: iStar. 2021. p. 21-27.

[37] Jgoose. Github. https://github.com/diegodallabt/jgoose.

[38] Jgoose. Video. https://osf.io/qcbxw.