Using of i*(iStar) 2.0 for Improving the use Cases Derivation

B. L. Casarotto, G. C. L. Geraldino, V. F. A. Santander, I. F. Silva and M. A. T. Cespedes

***Abstract*— The process of deriving use cases from organizational models built using the i * framework has been presented in previous works. However, the guidelines used to derive use cases are based on the elements of the original version of i *. More recently, the new version i * 2.0 has been proposed incorporating important changes which must be evaluated in relation to their impact on the use-case derivation process. In this work, this evaluation is carried out, the guidelines modified and applied to an example showing the improvements obtained.**

***Index Terms* — Organizational Modeling, Requirements Engineering, Use Cases.**

I. Introdução

A Engenharia de Requisitos é uma importante subárea da Engenharia de Software [4] a qual abrange atividades da etapa inicial do desenvolvimento de sistemas. Compreende as atividades como a análise de domínio do problema, a elicitação, bem como a análise, a especificação, a negociação e a evolução dos requisitos de um software [1], [3].

Esta subárea, em uma perspectiva proposta pela engenharia de requisitos orientada a metas [2], [7], pode ser observada como um processo composto por duas fases: a fase de requisitos iniciais (*early requirements phase*) e a fase de requisitos detalhados (*late requirements phase*). A fase de requisitos iniciais analisa e modela o ambiente a ser transformado em software, o contexto organizacional, os *stakeholders*, seus objetivos e seus relacionamentos. A fase de requisitos detalhados concentra-se na modelagem do sistema em conjunto com o ambiente, determinando e ajustando as fronteiras do sistema, sendo possível identificar requisitos do software [5]. Os requisitos de software correspondem a descrições do que o sistema deve fazer, serviços que deve oferecer e restrições a seu funcionamento [6]. Requisitos de software podem ser especificados de diversas maneiras, como, por exemplo, por meio da abordagem orientada a objetivos GORE (*Goal Oriented Requirements Engineering*) [2], [21]. A GORE evidencia a motivação para o desenvolvimento do sistema, focando na expectativa do usuário em relação ao que o sistema deve fazer ou como ele deve se comportar [2], [7]. Uma das técnicas que representam essa abordagem, a mais conhecida e foco desse trabalho, é a i* (lê-se istar). A i* tem como objetivo representar os membros da organização, suas intencionalidades, suas motivações e seus relacionamentos, possibilitando melhor compreensão das relações organizacionais. Tipicamente, esta técnica é utilizada na fase de requisitos iniciais (*early requirements phase*). Contudo, esta técnica também tem sido utilizada para apoiar o processo de obtenção de requisitos de software. Em [8] a proposta tem seu foco na derivação de requisitos funcionais de sistemas computacionais na forma de casos de uso UML [9] por meio de modelos i*. São propostas diretrizes com o intuito de auxiliar engenheiros de requisitos no processo de derivação de casos de uso considerando as características presentes nos modelos i*, categorizados em modelos de *Strategic Dependence (SD)* e *Strategic Rationale (SR)*.

Contudo, o que se apresenta em [8] considera a primeira versão de i* descrita em [10]. Essa versão do *framework* i* vem recebendo várias propostas de mudanças pela comunidade científica e industrial, a partir do uso do mesmo em diversos domínios de aplicação. Algumas destas propostas são descritas em [22], [23], as quais apontam novos construtores ou mudanças naqueles já existentes, bem como detalham melhor alguns aspectos semânticos da linguagem. Algumas destas mudanças estão incorporadas na versão i* 2.0 [11]. Também, mais recentemente, em [24], [25], [26], são apresentados estudos de extensões sobre i* e uma proposta de um processo para dar suporte à proposição dessas extensões.

Neste contexto, é importante estudar as mudanças na versão i* 2.0 e como as mesmas podem beneficiar a derivação de casos de uso conforme proposto em [8]. Isto implica modificar as diretrizes, as quais possibilitam a derivação dos casos de uso por meio de modelos i*, permitindo tanto melhorar a descrição diagramática quanto textual dos casos de uso gerados. A completude da descrição de requisitos funcionais na forma de casos de uso é um aspecto fundamental na produção de softwares de qualidade [27]. Este é um forte motivador para a realização deste trabalho.

Este processo de derivação de diagramas de casos de uso por meio de modelos i* pode ser desenvolvido com auxílio de ferramenta computacional, a *Java Goal Object into Oriented Standard Extension* – JGOOSE [12], [13]. Alicerçada nas diretrizes propostas por [8], a JGOOSE desenvolve seus processos com base nos modelos organizacionais i* para gerar diagrama de casos de uso UML [9] bem como as descrições

B. L. Casarotto, Universidade Estadual do Oeste do Paraná (UNIOESTE), Cascavel, PR, Brasil (brunocasarotto@hotmail.com).

G. C. L. Geraldino, Universidade Estadual do Oeste do Paraná (UNIOESTE), Cascavel, PR, Brasil (gustavo.geraldino@unioeste.br).

V. F. A. Santander, Universidade Estadual do Oeste do Paraná (UNIOESTE), Cascavel, PR, Brasil, (victor.santander@unioeste.br).

I. F. Silva, Universidade Estadual do Oeste do Paraná (UNIOESTE), Cascavel, PR, Brasil, (ivonei.silva@unioeste.br).

M. A. T. Cespedes, Universidad Catolica del Maule (UCM), Maule, Chile, (mtoranzo@ucm.cl).

textuais desses casos de uso usando o *template* proposto em [14].

Este artigo está estruturado, além da introdução, em: Seção II descreve alguns trabalhos relacionados, enfatizando as diferenças em relação à proposta deste artigo; Seção III apresenta, resumidamente i*, as características da técnica e as mudanças da versão 2.0; Seção IV apresenta o processo e diretrizes de derivação dos casos de uso por meio desta técnica; Seção V expõe impactos desta versão i* (2.0) em relação ao processo de derivar casos de uso com base nas diretrizes; Seção VI as diretrizes modificadas são aplicadas a um exemplo visando demonstrar sua viabilidade; Seção VII há a apresentação das considerações finais e a explanação de trabalhos futuros.

II. Trabalhos relacionados

Primeiramente é importante destacar que este trabalho é uma evolução do trabalho apresentado inicialmente em [8] e, posteriormente, em [21]. Mais especificamente, as diretrizes apresentadas nesses trabalhos são modificadas para considerar a versão i* 2.0, conforme pode ser observado na seção V. A principal contribuição e originalidade do presente trabalho, está na inclusão de novas descrições textuais que relacionam requisitos não funcionais (qualidades) e recursos necessários, a passos específicos em um caso de uso. Essas informações adicionais melhoram, mesmo que pontualmente, o entendimento e utilização de casos de uso nas demais etapas do processo de software. Cabe também destacar, que estas novas descrições textuais, não são consideradas na ferramenta JGOOSE [12], [13], [15], [17], [18], [19], a qual suporta apenas as diretrizes originais de derivação de casos de uso a partir de i*.

Alguns trabalhos relacionados podem ser descritos. Para o contexto de desenvolvimento de linhas de produto de software, o estudo [28] adota GORE. Nele é apresentado um processo para geração e configuração de especificações de cenários de casos de uso para modelos de feature. O estudo também adota diretrizes adaptadas para a geração de cenários de casos de uso e utiliza heurísticas baseadas em i*-c *language*, para gerar modelos SR com cardinalidade e endereçar aspectos de variabilidade nos requisitos. Para o contexto de desenvolvimento ágil de software, os trabalhos [29], [30], [31], exploram o mapeamento entre estórias de usuário e modelos i*. Em [29] (definição de heurísticas) e [30] (ferramenta para automatizar a execução de heurísticas) mapeiam estórias de usuário para modelos i* com o objetivo de mitigar a falta de documentação em desenvolvimento ágil de software. Em [31] transforma-se modelos i* em estórias de usuário com o objetivo de integrar os benefícios de ambos os artefatos. Em [32] propõe-se o uso modelos i* visando enriquecer a elaboração de casos de uso de negócio no âmbito do processo unificado da rational. Embora esses estudos contribuam para integração e rastreamento de elementos dos artefatos i* com outros em contextos específicos, essas heurísticas não abordam a nova versão i* 2.0.

Já em [33], propõe-se uma lista de qualidades associadas à técnica i* 2.0, as quais devem ser avaliadas empiricamente. Destaca qualidades como completude, facilidade de compreensão e aplicabilidade da técnica em âmbito industrial. Em [34], apresenta-se uma revisão sistemática da literatura para investigar estudos relacionados a i* e problemas em aberto no uso da técnica. Ressalta nesse sentido, o uso de i* combinada a outras linguagens ou modelos, destacando a viabilidade dessa combinação. Estes trabalhos fortalecem a importância da integração da i* 2.0 com casos de uso, conforme proposto no presente artigo.

III. Modelagem i*

Nesta seção apresentamos brevemente os conceitos básicos do framework i* e diferenças da versão 2.0 com a versão 1.0. Em seguida são descritas as diretrizes que apoiam o processo de derivação de casos de uso propostas em [8].

A. Conceitos da Modelagem i e Mudanças na Versão 2.0.*

A técnica i*, 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 os mesmos 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. Este exemplo será utilizado para mostrar os novos elementos e mudanças incorporados na versão i* 2.0.

Essa nova versão foi proposta em [11] e possui algumas diferenças em relação a sua versão original apresentada em [10]. Os atores possuem duas categorias na versão i* 2.0, 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). Neste último caso trata-se de expressões de comportamentos de um ator social mais abstratas inserido no domínio ou em um determinado contexto especializado. 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).

Na versão anterior havia o papel de *Position*, o qual foi retirado na versão 2.0. Na nova versão há dois tipos possíveis de relacionamentos entre atores os quais são:

a. *is-a*: já existia na versão original e 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, substituindo assim as nomenclaturas *is-part-of*, *plays*, *occupies*, *covers* que existiam no i* original [10]. Não há restrição quanto ao tipo de atores ligados por este relacionamento. A relação ocorre entre os atores Jaqueline (*Agent*) e Gerente (*Role*) expressando que o agente Jaqueline desempenha o papel de Gerente no ambiente organizacional, conforme apresentado na Fig. 1.

As intencionalidades de atores usadas tanto no modelo SD (como *dependum*) quanto no SR (razões internas) podem ser dos tipos abaixo. A única mudança na versão i* 2.0 foi a alteração da nomenclatura de *Softgoal* para *Quality*.

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. Este elemento é basicamente *Softgoal* da versão original;
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 qualquer, o qual pode ser considerado como uma entidade (informativa ou física).

TABELA I
RESUMO DE LIGAÇÕES ENTRE INTENCIONALIDADES

		Apontado para *Goal*	Apontado para *Quality*	Apontado para *Task*	Apontado para *Resource*
Começa de	*Goal*	Refinamento	Contribuição	Refinamento	-
Começa de	*Quality*	Qualificação	Contribuição	Qualificação	Qualificação
Começa de	*Task*	Refinamento	Contribuição	Refinamento	-
Começa de	*Resource*		Contribuição	Necessário em	-

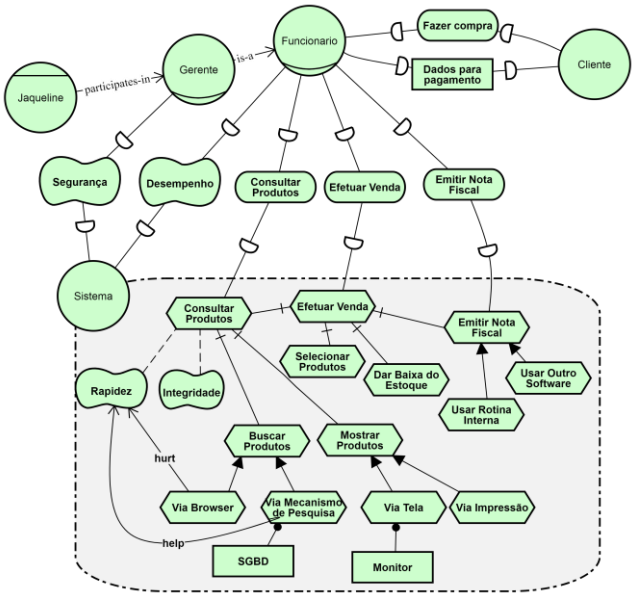


Fig. 1. Exemplo de modelo híbrido no *framework* i*.

Em relação às ligações entre intencionalidades, os tipos são: *refinement* (refinamento), *needed-by* (necessário em), *contribution* (contribuição) *e qualification* (qualificação), descritos na Tabela I. Para promover a facilidade de adoção, o i* 2.0 define algo mais genérico, um relacionamento chamado *refinement* o qual vincula, de forma hierárquica, tarefas e objetivos. O *refinement* é uma relação n-ária que relaciona um dos pais a um ou demais filhos.

Existem dois tipos de refinamento, os quais se aplicam a qualquer tipo de pai podendo ser objetivo ou tarefa, e os mesmos são definidos a partir do operador lógico na relação pai-filhos:

a. *AND*: os pais são satisfeitos quando o cumprimento dos n filhos (todos) ocorre; na versão original este refinamento era denominado *task-decomposition* (decomposição da tarefa Selecionar Produto, Realizar Venda em Consultar Produto, 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; na versão original este refinamento era denominado *means-end* (decomposição da tarefa Buscar Tarefas - tarefas Via Browser ou Via Mecanismo de Pesquisa na Fig. 1).

O relacionamento *needed-by* é um novo elemento proposto no framework i* e faz o vínculo de uma tarefa (qualquer) a um recurso (qualquer) indicando, no caso, que para desenvolver a tarefa o ator necessita do recurso. Este relacionamento é representado graficamente como uma seta com uma ponta de seta circular direcionada para a tarefa. O recurso SGBD é necessário pela tarefa Via Mecanismo de Pesquisa conforme pode ser observado na Fig. 1.

Já o relacionamento *contribution* tem sua representação nos efeitos nas *qualities* dos elementos intencionais. Esta ligação pode auxiliar analistas na tomada de decisões no aspecto de optar por tarefas alternativas ou objetivos considerando os efeitos em aspectos de qualidade no ambiente. Este efeito do elemento intencional em relação à *qualities* pode ser:

a. Make: A fonte prove uma positiva contribution e suficiente para que o alvo seja satisfeito;
b. *Help*: A fonte prove um pouco de contribuição positiva para a satisfação do alvo (na Fig. 1 a tarefa Via Mecanismo de Pesquisa ajuda na satisfação da qualidade Rapidez);
c. *Hurt*: A fonte prove uma fraca *contribution* negativa em relação ao alvo (Fig. 1 tarefa Via Browser influencia negativamente na satisfação da qualidade Rapidez);
d. *Break*: A fonte prove *contribution* negativa suficiente em relação ao alvo.

Esses 4 links de "contribuição" já estavam presentes na versão original proposta por [10]. No entanto, a proposta original possuía mais cinco links, sendo eles: *And, Or, Some+, Some-* e *Unknown*, os quais foram retirados na versão 2.0.

Por fim, o link de intenção de "qualificação" relaciona uma "qualidade" ao elemento ligado à mesma: *task*, *goal* ou *resource*. Esta relação exprime uma "qualidade" à qual se deseja acerca do desenvolvimento de uma *task*, de um *goal* realizado ou ainda de um *resource* provido. A relação de "qualificação" é representada graficamente por meio de uma

linha pontilhada conectando a "qualidade" ao elemento sendo qualificado.

Além das visões de SR e SD, é proposta no i* 2.0 também uma visão híbrida. Esta visão é adotada quando apenas alguns dos atores estão abertos (expandidos), concentrando a lógica estratégica de decomposição das razões internas nesses atores. O exemplo apresentado na Fig. 1 é um modelo híbrido já que apenas o ator Sistema está decomposto.

IV. Derivação de Casos de Uso

Em [8] há uma proposta de aproximação entre os modelos de cunho organizacional dos modelos ditos funcionais. Com a utilização de tal preposição há a possibilidade de realizar a elicitação e documentação dos casos de uso em UML [9] com base nos modelos i* (organizacionais) elaborados pelo framework i* [10]. Com esta finalidade, são definidas diretrizes, as quais serão descritas resumidamente nesta seção. A Fig. 2 expõe as etapas, resumidamente, desta proposta.

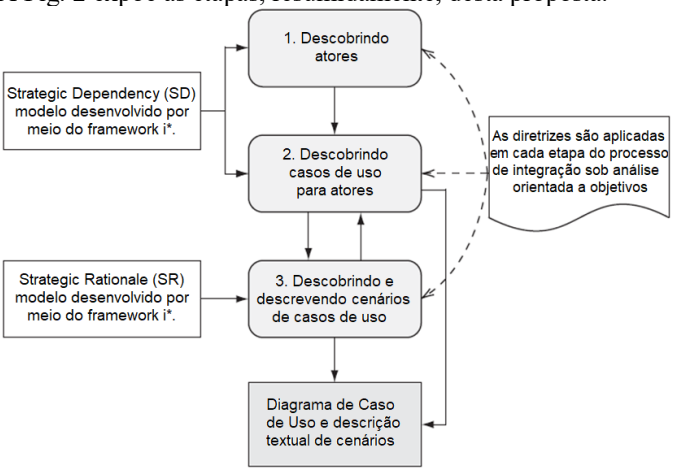


Fig. 2. Processo de derivação de casos de uso com base no framework i*.

Primeira Etapa da Proposta – Descobrindo atores.

Diretriz 1: todo e qualquer ator que conste no modelo i* será mapeado como candidato a ator nos casos de uso.

Diretriz 2: no modelo i*, o candidato deverá ser um ator externo no que se refere ao sistema que se pretende, de outro modo não será um ator mapeado nos casos de uso.

Diretriz 3: no modelo i*, o ator candidato precisa possuir ao menos uma dependência em relação ao que se pretende, o sistema computacional.

Diretriz 4: os atores no modelo i*, os quais foram relacionados por meio de mecanismo IS-A e, que passaram pelo individual mapeamento para atores em casos de uso, serão relacionados com o estereótipo <<generalização>> no diagrama de casos de uso.

Segunda Etapa da Proposta – Descobrindo Casos de Uso para Atores.

Diretriz 5: a cada ator mapeado na etapa 1, observa-se as relações ator_sistema_computacional (*depender*) -> Dependência (*dependum*) -> ator_descoberto (*dependee*) objetivando a descoberta casos de uso para o ator.

SubDiretriz 5.1: Dependências *goal* no modelo i* permitem seu mapeamento para *goal* em Casos de Uso.

SubDiretriz 5.2: deve-se avaliar as dependências do tipo tarefa associadas com o ator. Se um ator depende de outro ator para realizar uma tarefa, deve-se investigar se esta tarefa necessita ser refinada em subtarefas. Este tipo de tarefa pode ser mapeado para caso de uso.

SubDiretriz 5.3: deve-se avaliar as dependências do tipo recurso associadas com o ator. Se um ator depende de outro ator para obter um recurso; por que o mesmo é requerido? Se para esta resposta existe um objetivo, o mesmo será candidato a ser um objetivo de um Caso de Uso para este ator.

SubDiretriz 5.4: deve-se avaliar as dependências do tipo *softgoal* associadas com o ator. Tipicamente uma dependência deste tipo representa um requisito não funcional (RNF) associado ao sistema pretendido.

Diretriz 6: analisar a situação especial na qual um ator (descoberto seguindo as diretrizes da etapa 1) possui dependências como depender em relação ao ator em i* que representa o sistema computacional pretendido ou parte dele (ator_descoberto (*depender*) -> dependência (*dependum*) -> sistema_computacional(*dependee*). Estas dependências podem ser mapeadas para casos de uso. As dependências do tipo *softgoal* são mapeadas para RNFs do sistema;

Diretriz 7: classificar cada caso de uso de acordo com seu tipo de objetivo associado (objetivo contextual, objetivo de usuário ou objetivo de subfunção);

Terceira Etapa da Proposta – Descobrindo e descrevendo cenários de casos de uso.

Diretriz 8: analisar cada ator e seus relacionamentos no Modelo SR para extrair informações de descrição de fluxos principal e alternativos, pré-condições dos casos de uso descobertos para o ator. Para isso precisamos analisar os subcomponentes em uma ligação de decomposição de tarefa (*task-decomposition*) mapeando-os para passos na descrição do cenário primário (fluxo principal) de casos de uso. Também devemos analisar ligações do tipo meio-fim (*means-end*) mapeando os meios para passos alternativos na descrição de casos de uso;

Diretriz 9: investigar a possibilidade de derivar novos objetivos de casos de uso a partir da observação dos passos nos cenários (fluxos de eventos) dos casos de uso descobertos;

***Caso de Uso**: <número> << o nome é um objetivo descrito com uma frase curta contendo um verbo na voz ativa >>*

INFORMAÇÃO CARACTERÍSTICA

***Objetivo no Contexto:** <uma sentença mais longa do objetivo do caso de uso se for necessário>*

***Escopo:** <Qual sistema está sendo considerado (por exemplo, organização ou sistema computacional)>*

***Nível:** <um dos tipos de objetivo: objetivo de usuário, objetivo de contexto ou objetivo de subfunção>*

***Pré-condições:** <o que é necessário que já esteja satisfeito para realizar o caso de uso>*

***Condição Final de Sucesso:** <o que ocorre/muda após a obtenção do objetivo do caso de uso>*

***Condição Final de Falha:** <o que ocorre/muda se o objetivo é abandonado>*

***Ator Primário:** <o nome do papel para o ator primário, ou descrição>*

CENÁRIO PRINCIPAL DE SUCESSO

<coloque aqui os passos do cenário necessários para a obtenção do objetivo >

<passo #> <descrição da ação >

EXTENSÕES

<coloque aqui as extensões, uma por vez, cada uma referenciando o passo associado no cenário principal >

<passo alterado> <condição> : <ação ou sub.caso de uso >

<passo alterado > <condição> : <ação ou sub.caso de uso >

ASPECTOS A CONSIDERAR (OPCIONAL)

<incluir aqui aspectos relacionados ao caso de uso que necessitam resolução>

Fig. 3. *Template* de descrição textual de casos de uso.

Diretriz 10: Desenvolver o diagrama de casos de uso utilizando os casos de uso descobertos e os relacionamentos do tipo <<include>>, <<extend>> e <<generalization>> usados para estruturar as especificações dos casos de uso. É importante ressaltar que após a derivação do diagrama e descrições textuais, os casos de uso gerados devem ser revisados pelo engenheiro de requisitos. Casos de uso podem ser excluídos ou incluídos nessa analise.

As diretrizes são suportadas em uma execução semiautomatizada pela ferramenta JGOOSE [15], [16].

Atualmente a ferramenta permite a elaboração de modelos organizacionais do framework i*, através do E4J i* [17], a criação de Diagramas de Casos de Uso [18] e a obtenção automática de Casos de Uso a partir de modelos i* [13], [19] e BPMN (*Business Processo Model and Notation*) [20]. Os Casos de Uso obtidos são apresentados através de Diagrama de Casos de Uso UML [9] e descrições textuais, utilizando uma versão baseada no *template* proposto por [14] (Fig. 3).

V. IMPACTO DA VERSÃO I* 2.0 NA DERIVAÇÃO DE CASOS DE USO

Nesta seção, as mudanças da versão i* 2.0 são analisadas visando averiguar o impacto das mesmas no processo de obtenção de casos de uso utilizando modelos i*. Para facilitar o entendimento desse impacto nas diretrizes apresentadas na seção anterior, as mesmas são apresentadas separadamente conforme segue.

a) *Modificação 1: Na versão original*: Atores possuem, além da representação genérica (*actor*), as representações *Agent*, *Role* e *Position*. *Na versão i* 2.0*: Retirado a noção de *Position*, pois a mesma foi avaliada como uma abstração entre os atores *Agent* e *Role*, o que causava confusão em alguns casos. *Impacto nas diretrizes*: Observando as diretrizes específicas associadas à derivação de atores em casos de uso a partir de modelos i* (primeira etapa) é possível verificar que na proposta inicial somente considera-se o ator genérico *Actor*. Na nova versão do i*, esta categoria de ator não sofreu modificação, não afetando as diretrizes da etapa 1. No entanto, como foi considerado apenas o ator genérico, foi avaliado se a introdução dos conceitos de atores *Role* e *Agent* geram algum impacto em alguma diretriz da primeira etapa. O estudo realizado demonstrou que não há impacto, pois o conceito de ator genérico é apenas uma generalização de ambos os conceitos de atores, portanto, em determinadas situações o uso do *Role* poderá ocorrer, enquanto em outras o uso do *Agent*. Entretanto, as diretrizes da primeira etapa da proposta serão alteradas para deixar explícito que os atores nos modelos i* analisados podem ser do tipo *Agent, Role* e *Actor* (Genérico).

b) *Modificação 2: Na versão original*: Existem algumas relações entre dois atores, são elas: *is-a, is-part-of, plays, occupies, covers*. *Na versão i* 2.0*: Foram retiradas as noções de *is-part-of, plays, occupies, covers* e foram agregadas em *participates-in*, sendo que a mesma representa qualquer tipo de associação, que não seja representável pela relação *is-a*. *Impacto nas diretrizes*: Observando a Diretriz 4 da primeira etapa da proposta, a qual trata da associação *is-a* que representa generalização/especialização, podemos verificar que a mesma não sofrerá alteração. Contudo, esta diretriz não considera as demais associações já mencionadas agregadas na associação *participates-in*. No exemplo da Fig. 1 é possível identificar que o ator Jaqueline do tipo *Agent* está relacionado ao ator do tipo *Role* Gerente através de *participates-in*, o que significa que Jaqueline pode desempenhar o papel de Gerente. Este tipo de associação entre atores não é contemplado em casos de uso conforme expresso em [9]. Um ator em casos de uso pode-se relacionar a outros atores apenas usando a hierarquia de generalização refletida no modelo i* pela relação *is-a*, já considerada.

c) *Modificação 3: Na versão original*: Um dos elementos intencionais que um ator pode possuir é chamado de *softgoal*, o qual representa um objetivo flexível que o ator deseja satisfazer. Este objetivo flexível pode ser considerado um RNF na engenharia de requisitos. *Na versão i* 2.0*: Para retirar a noção de objetivo do *softgoal*, o mesmo foi renomeado para *quality* fazendo com que sua interpretação se transformasse em uma qualidade que deve ser fornecida cuja satisfação é subjetiva. *Impacto nas diretrizes*: A subdiretriz 5.4 da proposta indica que um *softgoal* é mapeado para um RNF do sistema pretendido. Neste sentido, a mudança de nomenclatura desta dependência para *quality* afeta esta diretriz apenas com a mudança do nome da dependência utilizada.

d) *Modificação 4: Na versão original*: existem três tipos de links entre os elementos intencionais objetivo, recurso e *softgoal,* os quais são: *means-end, task-decomposition* e *contribution*. *Na versão i* 2.0*: os links *means-end* e *task-decomposition* foram englobados em um só chamado de *refinement* (refinamento). O link intencional *contribution* (contribuição*)* foi reduzido a quatro categorias sendo *help, break, hurt e make*. Por outro lado, houve a adição de mais dois links intencionais, *qualification* (qualificação) e *neededBy* (necessário em), sendo que o primeiro representa um link entre uma intencionalidade do tipo objetivo ou tarefa e uma qualidade, e o segundo representa um link entre uma intencionalidade do tipo objetivo ou tarefa e um recurso. *Impacto nas diretrizes*: observando as diretrizes específicas e as mudanças ocorridas, é possível identificar um impacto na Diretriz 8, visto que a mesma analisa no modelo SR, ligações do tipo *means-end* para mapear os meios para passos alternativos de casos de uso. Também há uma análise no modelo SR dos subcomponentes de ligações de *task-decomposition* para mapear o cenário primário de casos de uso. Contudo, o impacto é apenas na nomenclatura utilizada na diretriz e não no processo utilizado já que ambas ligações continuam com o mesmo significado sendo agora OR e AND, respectivamente. Por outro lado, é necessário analisar de forma mais específica as ligações do tipo "qualificação", "necessário em" e "contribuição".

A ligação do tipo "qualificação" representa uma qualidade desejada na realização de um elemento intencional do tipo objetivo, tarefa ou recurso. Por exemplo, na Fig. 1, a tarefa Consultar Produto possui a qualidade Rapidez associada. Esta relação deve ser incorporada à descrição textual de casos de uso. Para esse fim, foi acrescido ao *template* da descrição textual na Fig. 4, o campo *Open Issues*. Neste caso, a descrição

será realizada da seguinte forma: Qualidade [desejada em] Tarefa ou Objetivo ou Recurso. Para o exemplo da Fig. 1, a descrição seria: Rapidez [desejada em] Consultar Produto. Essa descrição sempre estará associada à descrição textual do caso de uso no qual a tarefa, objetivo ou recurso está envolvido. No nosso exemplo, como a tarefa Consultar Produto é mapeada para caso de uso (Diretriz 6), essa descrição estará contida no mesmo.

Já a ligação "necessário em" exemplificada na Fig. 1, representa que o SGBD é um recurso necessário para executar a tarefa Via Mecanismo de Pesquisa. Esta relação será incorporada na descrição textual do caso de uso cuja tarefa está incluída. Por exemplo, no mapeamento seguindo a Diretriz 8 (ver seção anterior), as tarefas Buscar Produto e Via Mecanismo de Pesquisa correspondem a um passo do cenário principal do caso de uso Consultar Produto (Diretriz 6). Assim a descrição deste passo no cenário principal desse caso de uso fica da seguinte forma: Passo #. Buscar Produto Via Mecanismo de Pesquisa [com o recurso necessário]: SGBD.

Finalmente, em relação à ligação "contribuição", conforme já descrito anteriormente, a versão i* 2.0 considera quatro tipos: *help, break, hurt* e *make*. Neste sentido, o campo *Open Issues* na descrição textual, será usado para descrever relações de "contribuição". Por exemplo, para expressar a contribuição negativa *hurt* entre a tarefa Via Browser e a qualidade Rapidez na Fig. 1, no campo *open issues* será acrescido o seguinte texto: Via Browser [hurt] Rapidez.

É possível perceber que as mudanças descritas afetarão a Diretriz 8, as quais precisarão ser reescritas. Contudo, a introdução de novos elementos na versão i* 2.0 e o mapeamento destas para a descrição textual dos casos de uso gerados, acrescentará informações importantes aos engenheiros de requisitos e demais profissionais envolvidos.

e) *Modificação 5: Na versão original*: As visões suportadas pelo i* original eram apenas as visões SD e SR. *Na versão 2.0*: Há a proposta de uma nova visão, denominada de híbrida, na qual nem todos os atores são expandidos. Isto comumente ocorre por questões de interesse dos modeladores os quais focam em refinar as razões internas de alguns atores mais importantes. *Impacto nas diretrizes*: Cabe ressaltar que a Diretriz 8 utiliza essencialmente o refinamento dos atores para gerar descrições textuais dos casos de uso descobertos. O fato de nem todos os atores serem decompostos em suas razões internas se reflete no processo de mapeamento com menos informações. Dessa forma, a mudança proposta está em deixar mais claro na Diretriz 8 que modelos híbridos podem ser usados no mapeamento, mas isto afeta negativamente o processo de descrição textual dos casos de uso. Com as modificações demonstradas e exemplificadas podemos agora propor as mudanças necessárias para as diretrizes propostas em [8]. Somente serão apresentadas as diretrizes que foram modificadas.

Primeira Etapa da Proposta – Descoberta de Atores

Diretriz 1: todo ator em i* deve ser analisado para um possível mapeamento para ator em caso de uso, podendo ele ser do tipo *Role, Agent ou Actor* (Genérico).

Diretrizes 2, 3 e 4: sem modificações.

Segunda Etapa da Proposta – Descoberta de Casos de Uso

Diretriz 5 e subdiretrizes 5,1, 5,2 e 5,3: sem modificações

SubDiretriz 5.4: avaliar dependências (tipo "qualidade") as quais estão associadas com o ator. Tradicionalmente, em modelos i*, as dependências deste tipo são RNFs associados ao sistema pretendido.

Diretriz 6 e 7: sem modificações.

Terceira Etapa da Proposta – Descoberta e descrição do fluxo principal (cenário principal) e alternativo (extensões) dos Casos de Uso.

Diretriz 8: análise de cada ator que atua como *dependee* na satisfação de alguma dependência mapeada para caso de uso. Analisar suas razões internas e ligações no modelo SR para a extração informações as quais conduzirão a uma descrição dos fluxos (principal e alternativos) e ainda, de pré e pós-condições dos casos de uso descobertos para o ator. Seguir as subdiretrizes abaixo (novas diretrizes propostas considerando a versão i* 2.0).

SubDiretriz 8.1: analisar os subcomponentes do elemento intencional que satisfaz a dependência mapeada para caso de uso. Se esses subcomponentes estiverem relacionados ao elemento intencional via uma ligação de "refinamento" do tipo *AND*, os mesmos são mapeados para passos/eventos na descrição do cenário principal (fluxo principal) do caso de uso mapeado. Ainda, se um desses subcomponentes também estiver satisfazendo uma dependência previamente mapeada para caso de uso, será acrescentado ao passo/evento representado pelo mesmo, a expressão <<include>> significando que aquele passo do cenário principal será tratado por outro caso de uso.

Por outro lado, se um dos subcomponentes possuir ainda ligações de "refinamento" do tipo OR, um dos filhos nessa ligação complementará o passo do cenário principal mapeado e os outros serão mapeados para passos do cenário secundário (extensões) do caso de uso. As extensões indicam possibilidades/alternativas para obtenção daquele passo. Da mesma forma, se um dos passos das extensões for gerado por um elemento intencional que satisfaz uma dependência já mapeada para caso de uso, nesse passo deverá ser acrescido a expressão <<extend>>. Isto significa que esse passo será tratado por outro caso de uso.

SubDiretriz 8.2: Há a situação especial na qual o elemento intencional que satisfaz uma dependência já mapeada em caso de uso é decomposto em um primeiro nível via "refinamento" do tipo OR. Nesse caso, ambos os filhos geram casos de uso indicando caminhos alternativos para a execução do elemento intencional. Outros subcomponentes ligados a um filho via refinamentos AND serão mapeados para passos do cenário principal do caso de uso gerado.

Ainda, se qualquer elemento intencional, o qual satisfaz uma dependência já mapeada em caso de uso, possuir uma dependência (elemento intencional->dependência->ator) em relação a outro ator, essa dependência gera uma pré-condição na descrição textual do caso de uso.

SubDiretriz 8.3: analisar, se existir, "qualidades" ligadas a elementos intencionais do tipo objetivo, tarefa ou recurso através da ligação "qualificação". Essas relações serão

mapeadas para o campo *Open Issues* da descrição textual dos casos de uso envolvendo os elementos intencionais relacionados. O formato de descrição dessa relação deve ser: Qualidade [desejada em] Tarefa, Objetivo ou Recuso.

SubDiretriz 8.4: analisar, se existir, recurso ligado a alguma tarefa com a ligação de "necessário em" para assim, determinar que tal recurso é obrigatório para o caso de uso em questão. Para refletir essa importante relação, uma descrição textual deve ser acrescida ao passo do cenário principal ou secundário associado à tarefa nessa relação. O formato de descrição dessa relação deve ser: Passo #. Tarefa [com recurso necessário]: Recurso.

SubDiretriz 8.5: analisar, se existir, ligação do tipo "contribuição" para verificar o quanto um elemento intencional influencia uma "qualidade". Se a ligação for do tipo *hurt* ou *break*, sua influência é pouco negativa e muito negativa, respectivamente, enquanto se a ligação for *help* ou *make*, sua influência é pouco positiva ou muito positiva, respectivamente. Para refletir essa influência a relação deve ser descrita no campo *Open Issues* do caso de uso cujo elemento intencional envolvido fizer parte. O formato de descrição dessa relação deve ser: Elemento Intencional [Tipo de Ligação *Contribution*] Qualidade.

Diretriz 9 e 10: sem modificação

VI. VALIDAÇÃO DA PROPOSTA

Para realizar a validação das diretrizes modificadas visando derivar casos de uso por meio de modelos i*, utilizaremos o exemplo da Fig. 1. Seguem as diretrizes aplicadas.

Primeira Etapa da Proposta - Descoberta de Atores

Diretriz 1 (diretriz modificada): Os atores identificados são os seguintes: atores do tipo papel (*role*) Gerente e Funcionário, atores Genéricos (*actor*) Cliente e Sistema e ator do tipo agente (*agente*) Jaqueline.

Diretriz 2 (diretriz modificada): Os atores externos ao sistema computacional são Gerente, Funcionário, Cliente e Jaqueline.

Diretriz 3: Funcionário e Gerente são os únicos atores, em relação ao sistema, que possuem dependências.

Diretriz 4: Há uma relação do tipo *is-a* entre o ator Gerente e Funcionário que origina uma relação entre os mesmos do tipo <<generalização>>. Ver Fig. 5.

Segunda Etapa da Proposta - Descoberta de Casos de Uso

Diretriz 5: Não há dependências do sistema em relação ao ator Funcionário ou Gerente mapeados na Diretriz 3.

SubDiretriz 5.1: Não se aplica; SubDiretriz 5.2: Não se aplica; SubDiretriz 5.3: Não se aplica.

SubDiretriz 5.4 (subdiretriz modificada): Essa diretriz teve apenas uma mudança de nomenclatura de elemento intencional *softgoal* para *quality*. No exemplo, não se aplica.

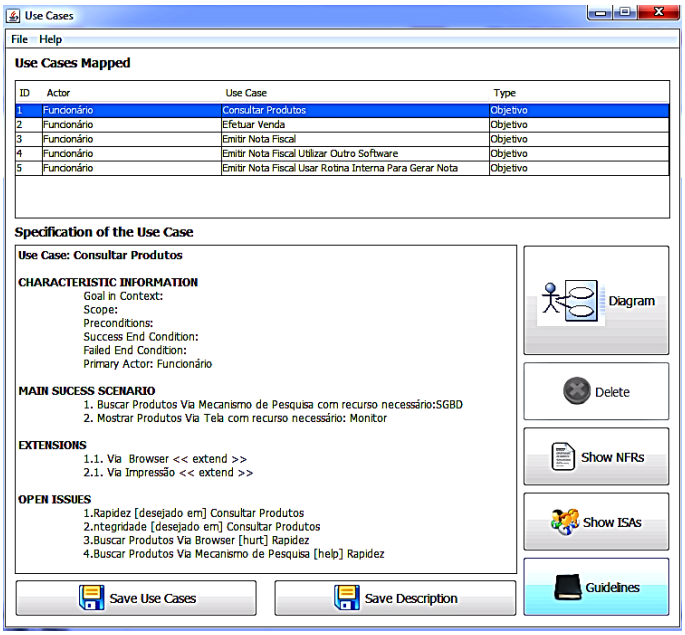


Fig. 4. Descrição Textual do caso de uso Consultar Produtos.

Diretriz 6: De acordo com esta diretriz, as dependências mapeadas para casos de uso são: Consultar Produtos, Efetuar Venda e Emitir Nota Fiscal. Também as dependências do tipo *quality* Segurança e Desempenho são mapeadas para RNFs do sistema.

Diretriz 7: São classificados como objetivos de usuário todos casos de uso oriundos de Emitir Nota Fiscal, Efetuar Venda e Consultar Produtos.

Terceira Etapa da Proposta - Descoberta e descrição do fluxo principal e alternativo dos Casos de Uso

Diretriz 8 (diretriz modificada): Houve uma separação em 4 subdiretrizes.

SubDiretriz 8.1: A tarefa Consultar Produtos possui ligações do tipo AND, antigo *means-end*, com as tarefas Buscar Produtos e Mostrar Produtos, as quais são mapeadas para passos do cenário principal do caso de uso Consultar Produtos. Aplica-se o mesmo procedimento para os casos de uso Efetuar Venda e Emitir Nota Fiscal. Assim, para Efetuar Venda temos os passos/eventos do cenário principal na Fig. 6: Consultar Produtos, Selecionar Produtos, Dar Baixa do Estoque e Emitir Nota Fiscal.

Cabe salientar que no cenário principal do caso de uso Efetuar Venda, os passos que correspondem às tarefas Consultar Produtos e Emitir Nota Fiscal serão acrescidos da expressão <<*include*>> representando que casos de uso com esses nomes serão incluídos.

Em um segundo momento, observa-se que as tarefas Buscar Produtos e Mostrar Produtos são refinadas usando o tipo OR. Assim, Buscar Produtos Via Browser irá compor um passo de cenário principal de Consultar Produtos e Buscar Produtos Via Mecanismo de Pesquisa irá compor um passo alternativo para esse evento no campo Extensões. O mesmo procedimento ocorre para as tarefas via tela e via Impressão do mesmo caso de uso. Isto pode ser observado na Fig. 4.

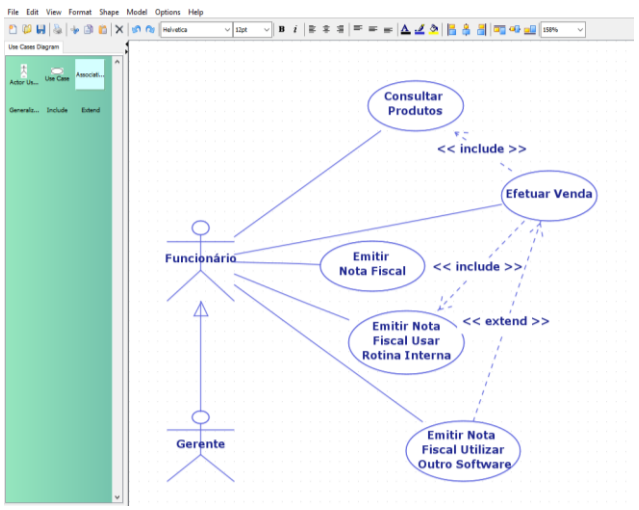


Fig. 5. Diagrama de casos de uso gerado para o modelo i* da Fig. 1.

SubDiretriz 8.2: A tarefa Emitir Nota Fiscal que satisfaz a dependência de mesmo nome e mapeada para caso de uso (ver Diretriz 6) é refinada via relação OR pelas tarefas Usar Rotina Interna para Gerar Nota e Usar Outro Software. Neste caso, a primeira tarefa gerará o caso de uso Emitir Nota Fiscal Usar Rotina Interna e a outra tarefa irá gerar outro caso de uso com o nome Emitir Nota Fiscal Usar Outro Software. A primeira tarefa muda a descrição do passo do cenário principal do caso de uso Efetuar Venda, e a segunda tarefa gera um caso de uso alternativo a esse passo expresso na seção de Extensões. A expressão <<extend>> Emitir Nota Fiscal Usar Outro Software é acrescida conforme apresentado na Fig. 6.

SubDiretriz 8.3: Há ligações do tipo "qualificação" entre as "qualidades" Rapidez e Integridade com a tarefa Consultar Produtos. Conforme orientado pela diretriz, estas ligações serão acrescidas no campo *Open Issues* da descrição textual do caso de uso Consultar Produtos. São descritas respectivamente: Rapidez [desejada em] Consultar Produtos e Integridade [desejada em] Consultar Produtos. Ver Fig. 4.

SubDiretriz 8.4: A tarefa Via Mecanismo de Pesquisa necessita do recurso SGBD para ser realizada bem como a tarefa Via Tela necessita do recurso Monitor. Esta relação é expressa pela ligação "necessário em" a qual é mapeada para a descrição textual do caso de uso Consultar Produtos, no qual as tarefas estão envolvidas. A descrição textual desta ligação é associada ao passo específico do cenário principal ou de extensões conforme a tarefa já foi mapeada. Neste caso, a seguinte descrição é gerada: Passo #. Buscar Produtos Via Mecanismo de Pesquisa [com recurso necessário]: SGBD, Passo #. Mostrar Produtos Via Tela [com recurso necessário]: Monitor. Ver Fig. 4.

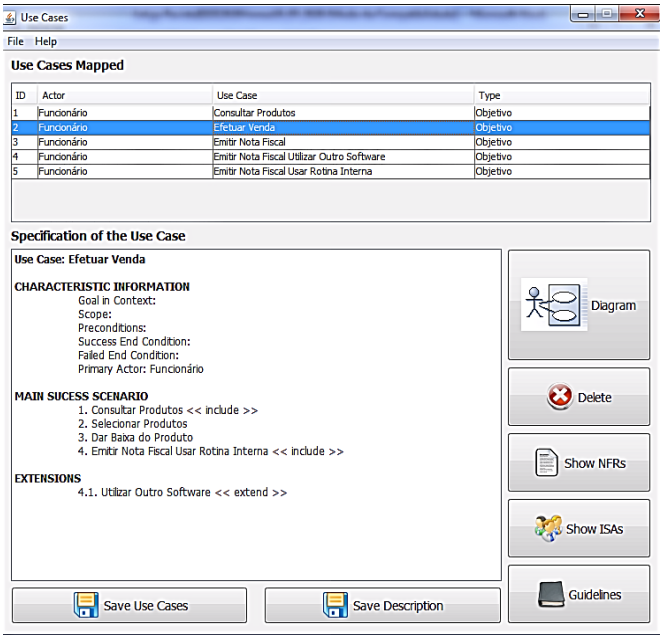


Fig. 6. Descrição Textual do caso de uso Efetuar Venda.

SubDiretriz 8.5: Há uma ligação "contribuição" do tipo *hurt* entre a tarefa Via Browser (a qual é uma possibilidade para Buscar produtos) e a qualidade Rapidez, indicando uma influência negativa da tarefa sobre a qualidade desejada. Do mesmo modo, a qualidade Rapidez possui uma influência positiva (ligação do tipo *help*) da tarefa Via Mecanismo de Pesquisa, indicando uma influência positiva da tarefa sobre a qualidade desejada. Ambas ligações são expressas no campo *Open Issues* do caso de uso Consultar Produtos, ao qual as tarefas em questão, já foram mapeadas. Desta forma, a descrição textual segue como: Via Browser [hurt] Rapidez e Via Mecanismo de Pesquisa [help] Rapidez. Ver Fig. 4.

Diretriz 9: Não se aplica.

Diretriz 10: O diagrama gerado na Fig. 5.

VII. Conclusões

De acordo com as modificações realizadas nas diretrizes que apoiam o processo de derivação de casos de uso a partir de modelos i* bem como os resultados obtidos no exemplo aplicado, é possível tecer algumas considerações. Apesar das mudanças da versão i* 2.0 serem pontuais, as mesmas permitem enriquecer os modelos de casos de uso gerados. Mais especificamente, a descrição textual dos casos de uso pode acrescentar várias informações importantes para o engenheiro de software. Inclusive pode permitir fazer análises de projeto e implementação mais concretas. Por exemplo, ao possuir RNFs descritos no campo *Open Issues* de um caso de uso específico, seja através da relação "contribuição" ou "qualificação", o engenheiro pode focar sua análise e escolher melhores métodos de projeto e implementação de acordo com essas relações. Quanto mais completas forem as descrições textuais bem como os diagramas de casos de uso, maiores são as chances de satisfação dos requisitos do cliente.

Como trabalhos futuros pretende-se implementar as mudanças realizadas nas diretrizes à ferramenta JGOOSE.

Conforme já comentado, esta ferramenta automatiza o processo de derivação de casos de uso e precisa ser atualizada para refletir a versão 2.0. Neste processo, também é importante realizar experimentos em âmbito acadêmico e industrial para averiguar os benefícios das mudanças realizadas. A proposta da versão i* 2.0 é justamente motivar e facilitar o uso da modelagem i* na academia e indústria. Nesse contexto, nós acrescentamos o fato de poder também utilizar esses modelos para gerar casos de uso, amplamente usados no desenvolvimento de software na atualidade.

REFERÊNCIAS

[1] G. Kotonya, I. Sommerville, "Requirements Engineering: Processes and Techniques", USA: Wiley, 1998.

[2] A. Lapouchnian, "Goal-Oriented Requirements Engineering: An Overview of the Current Research", *Department of Computer Science, University of Toronto,* 2005.

[3] A. Lamsweerde, "Requirements Engineering in the Year 00: A Research Perspective" *in 22nd International Conference on Software Engineering* (ICSE 2000), Limerick, Ireland, 2000.

[4] R. Pressmann and B. Mixin, "Software Engineering", 8th ed., USA: McGraw-Hill, 2015.

[5] B. Boehm, *"Software Engineering Economics"*, NJ, USA: Prentice *Hall PTR Upper Saddle River*, 1981.

[6] I. Sommerville, "Software Engineering", 10th ed., USA: Pearson Education, Inc, 2015.

[7] A. Lamsweerde, "Goal-Oriented Requirements Engineering: A Guided Tour", In 5th *IEEE International Symposium on Requirements Engineering*, Toronto, Canada, 2001.

[8] V. F. A. Santander and J. Castro, "Deriving Use Cases from Organizational Modeling" *in IEEE Joint International Requirements Engineering Conference* - RE 02, Essen, Germany, 2002.

[9] G. Blokdyk, "UML a Complete Guide", USA: Kindle, 2020.

[10] E. YU, "Modelling Strategic Relationships for Process Reengineering", *Ph.D. Thesis. Department of Computer Science*, University of Toronto, 1995.

[11] F. Dalpiaz, X. Franch, and J. Horkoff, "iStar 2.0 Language Guide". 2016. https://sites.google.com/site/istarlanguage/home. Acessado em 17/12/2020.

[12] A. A. Vicente, V. F. A. Santander, J. F. B. Castro, I. F. Silva, F. G. R. Matus, "JGOOSE: A Requirements Engineering Tool to Integrate i* Organizational Modeling with Use Cases in UML", *Ingeniare, Revista Chilena de Ingeniería* (en línea), vol. 17, pp.6-20, 2009.

[13] G. C. L. Geraldino and V. F. A. Santander, "The JGOOSE Tool" in 12th International i* Workshop (iStar 2019) - 38th International Conference on Conceptual Modeling (ER 2019), Salvador, Brasil, 2019.

[14] A. Cockburn, "Writing Effective Use Cases", Boston, MA, USA: Addison Wesley Longman Publishing Co., 2000.

[15] V. F. A. Santander and I. F. Silva, "Avaliando a utilização da Ferramenta JGOOSE no Processo de Ensino e Aprendizagem na Engenharia de Requisitos Um Relato de Experiência", *in XIX Conferência Internacional sobre Informática na Educação* (TISE), Fortaleza, Brasil, 2014.

[16] JGOOSE. http://www.inf.unioeste.br/les/index.php/listadownload. Acessado em 17/12/2020.

[17] L. P. Merlin, A. L. B. Silva, V. F. A. Santander, I. F. Silva and J. Castro, "Integrating the E4J editor to the JGOOSE tool" *in Requirements Engineering Workshop*, Lima, Peru, 2015.

[18] D. Peliser, V. F. A. Santander, I. Freitas, S. C. Andrade and E. Schemberger, "E4J Use Cases: um editor de diagrama de casos de uso integrado à ferramenta JGOOSE", *in: 35th In-ternational Conference of the Chilean Computer Science Society* (SCCC 2016), Valparaí-so, Chile. NY 12571 USA: IEEE Catalog Number CFP16139-ART, 2016

[19] M. Brischke, V. F. A. Santander and I. Silva, "Melhorando a ferramenta JGOOSE", 2012. *In: 15th Workshop on Requirements Engineering*, 2012, Buenos Aires, 24 a 27 de Abril

[20] A. N. Girotto, V. F. A. Santander, I. F. Silva and M. A. Toranzo, "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), Arica, Chile, 2017.

[21] E. Yu, P. Giorgini, N. Maiden and J. Mylopoulos, "Social Modeling for Requirements Engineering". USA: The MIT Press, 2011.

[22] J. Horko, T. Li, F. Li, M. Salnitri, E. Cardoso, P. Giorgini, J. Mylopoulos and J. Pimentel, "Taking goal models downstream: a systematic roadmap". *In International Conference on Research Challenges in Information Science*, pp 1-12. *IEEE*, 2014.

[23] J. Horko and E. Yu, "Comparison and evaluation of goal-oriented satisfaction analysis techniques", *Requirements Engineering*, 18(3):199-222, 2013.

[24] E. Gonçalves, J. Castro, J. Araújo and T. Heineck, "A Systematic Literature Review of iStar extensions", *Journal of Systems and Software*, vol. 137, pp. 1-33, 2018.

[25] E. Gonçalves, M. A. Oliveira, I. Monteiro, J. Castro and J. Araújo, "Understanding what is important in iStar extension proposals: the viewpoint of researchers", *Requirements Engineering*, vol. 24, pp. 55-84, 2019.

[26] E. Gonçalves, J. Araújo and J. Castro, "PRISE: A process to support iStar extensions", Journal of Systems and Software, v. 168, p. 110649, 2020.

[27] S. Tiwari and A. Gupta, "A systematic literature review of use case specifications research", *Information Software Technology*. pp. 128-158, 2015.

[28] G. Guedes, C. Silva and J. Castro, "Goals and Scenarios for Requirements Engineering of Software Product Lines", *Proceedings of the 5th International i* Workshop* (iStar 2011), 2011.

[29] A. Jaqueira, M. Lucena, E. Aranha, F. Alencar and J. Castro, "Using i* Models to Enrich User Stories", *Proceedings of the 6th International i* Workshop* (iStar 2013), 2013.

[30] R. Mesquita, A. Jaqueira, C. Agra, M. Lucena and F. Alencar, "US2StarTool: Generating i* Models from User Stories", *Proceedings of the Eighth International i* Workshop* (iStar 2015), 2015.

[31] C. Agra, A. Souza, J. Melo, M. Lucena and F. Alencar, "Specifying guidelines to transform i* Model into User Stories: an overview", *Proceedings of the Eighth International i* Workshop* (iStar 2015), 2015.

[32] Y. Wautelet and M. Kolp, "On the Integration of i* into RUP", In iStar pp. 61-66, 2013.

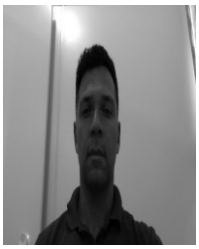
[33] L. López, F. B. Aydemir, F. Dalpiaz and J. Horkoff, "An empirical evaluation roadmap for iStar 2.0", *In Proceedings of the Ninth International* i* Workshop (istar'16) vol. 1674, pp. 55-60, 2016.

[34] A. Yasin and L. Liu, "Recent Studies on i*: A Survey", in iStar pp. 79-84, 2017.

Bruno Luiz Casarotto graduou-se em Ciência da Computação pela UNIOESTE (2019). Realizou sua monografia de conclusão de curso na área de Engenharia de Requisitos.

Gustavo Cesar Lopes Geraldino graduou-se em Análise e Desenvolvimento de Sistemas pela Faculdade Integrado de Campo Mourão (2008), mestrando em Ciência da Computação pela UNIOESTE.

Victor Francisco Araya Santander graduou-se em Ciência da Computação pela UEM (1994), mestre em Ciência da Computação pela USP (1997) e doutorou-se em Ciências da Computação pela UFPE (2002). Atualmente é professor associado da UNIOESTE. Suas pesquisas se concentram na área de engenharia de software e engenharia de requisitos.



Ivonei Freitas da Silva graduou-se em Informática pela UNIOESTE (1998), mestre em Ciências da Computação pela UFSC (2001) e doutorou-se em Ciências da Computação pela UFPE (2013). Atualmente é professor associado da UNIOESTE. Suas pesquisas se concentram na área de engenharia de software.

Marco Antonio Toranzo Cespedes graduou-se em Pedagogia em Matemática pela UNAP – Chile (1987), mestre em Ciências da Computação pela UFPE (1995) e doutorou-se em Ciências da Computação pela UFPE (2002). Atualmente é docente da Universidad Católica del Maule (UCM) - Chile. Suas pesquisas se concentram na Engenharia de Requisitos e Metodologias de Processos de Negócio.