Melhorando a Ferramenta JGOOSE

Mauro Brischke, Victor Francisco Araya Santander, Ivonei Freitas da Silva

UNIOESTE - Universidade Estadual do Oeste do Paraná, Cascavel – PR – Brasil
guardianmauro@yahoo.com.br, vfasantander@unioeste.br,
ivonei.silva@unioeste.br

Resumo. Um dos grandes desafios no desenvolvimento de software está em garantir que modelos funcionais tais como casos de uso em UML sejam desenvolvidos considerando intencionalidades expressas em modelos organizacionais. O uso de ferramentas computacionais pode facilitar este processo. Neste sentido, este trabalho apresenta uma nova versão da ferramenta JGOOSE (Java Goal Into Object Oriented Standard Extension), a qual permite gerar de forma automatizada casos de uso UML a partir de modelos organizacionais construídos via framework i*. Esta nova versão da ferramenta adiciona funcionalidades essenciais tais como, permitir alterar e salvar descrições textuais de casos de uso e gerar diagramas de casos de uso no formato XMI.

Palavras chave: JGOOSE, Modelagem Organizacional, Framework i*, Casos de Uso, Engenharia de Software.

1 Introdução

A engenharia de requisitos é uma fase de grande importância na elaboração de um sistema computacional. O entendimento e posterior documentação de requisitos funcionais e não-funcionais passa pela necessidade de entender inicialmente o contexto no qual o mesmo funcionará. A elaboração de modelos organizacionais pode auxiliar fortemente nesta tarefa inicial. Estes modelos devem possibilitar expressar as relações na organização bem como a lógica dos processos de negócio da mesma, de forma que engenheiros de requisitos possam considerar estas informações como base para elaborar documentos de requisitos mais completos, consistentes e não ambíguos. Neste contexto, destaca-se o uso do framework de modelagem organizacional i*. Esse framework propõe uma abordagem orientada a atores com foco nas intencionalidades, relacionamentos e motivações entre os mesmos, o que permite compreender melhor a organização e as relações entre os participantes [1][2][5]. Também, esse framework i* é um padrão internacional de modelagem de software aprovado pela norma ITU-T Recommendation Z.151 em Genebra, na Suíça [1].

É importante salientar que a construção destes modelos organizacionais deve apoiar o desenvolvimento de outros artefatos do processo de engenharia de requisitos e/ou engenharia de software. Entre os artefatos mais utilizados no processo de engenharia de requisitos destaca-se casos de uso em UML [9].

Casos de uso são utilizados normalmente no contexto de processos de desenvolvimento orientado a objetos e são considerados essenciais em diversas metodologias de desenvolvimento de software. Neste sentido, a proposta apresentada em [3] propõe uma abordagem que considera a construção de modelos organizacionais via framework i* e posterior derivação de casos de uso em UML a partir destes modelos. Para este fim, é proposto um processo guiado por diretrizes que auxiliam engenheiros de requisitos na utilização da abordagem.

Entre algumas vantagens deste processo de integração que podemos destacar é a possibilidade de derivar casos de uso com base nas intencionalidades associadas aos atores no ambiente organizacional, bem como compreender o ambiente para elicitar requisitos funcionais e não-funcionais do sistema computacional pretendido.

Para apoiar este processo de integração foi desenvolvida uma ferramenta computacional denominada JGOOSE (Java Goal Into Object Oriented Standard Extension) [4]. A versão atual desta ferramenta implementa as seguintes características: permite derivar atores e casos de uso a partir dos modelos em i*, permite derivar passos do cenário principal para cada caso de uso descoberto e permite gerar o diagrama de casos de uso em formato de imagem.

Contudo, a versão atual da ferramenta apresenta várias fraquezas. Dentre elas a necessidade de integração da ferramenta com outras que usam casos de uso como artefato, a necessidade de especificação de cenários secundários dos casos de uso e a possibilidade de alteração textual dos casos de usos.

Para tornar a ferramenta JGOOSE relevante para os Engenheiros de Requisitos, a nova versão da ferramenta permite alterar e salvar uma descrição textual do caso de uso gerado pela ferramenta, permitir gerar de forma automatizada o cenário secundário para um caso de uso e gerar o diagrama de casos de uso em formato XMI (XML Metadata Interchange) [6] para ser importando por outras ferramentas.

Essas novas funcionalidades permitem ao Engenheiro de Requisitos maior flexibilidade na especificação dos casos de uso, pois exclusivas modificações nos casos de uso são realizadas sem necessidade de alteração no modelo i*. A exportação do diagrama de casos de uso para o formato XMI permite a integração da ferramenta JGOOSE com outras ferramentas que exploram casos de uso como artefato. Além disso, a nova ferramenta torna-se mais completa em relação a anterior, pois permite a especificação o mapeamento de fluxos alternativos ao fluxo principal do caso de uso.

Desta forma, este trabalho apresenta uma nova versão da ferramenta JGOOSE, a qual acrescenta a versão atual as características ainda não implementadas já mencionadas. Estas novas características são essenciais para a utilização da ferramenta tanto em nível profissional quanto em nível acadêmico. Esta nova versão da ferramenta é apresentada utilizando um estudo de caso de um sistema de compra e venda genérico.

Trabalhos relacionados têm sido publicados na literatura científica. Em [22] é apresentado algumas ferramentas de apoio às regras de mapeamento entre as técnicas i* e UML, este primeiro autor, desenvolveu uma ferramenta que mapeia i* em classes UML. O trabalho de [21] implementa a modelagem organizacional com modelagem funcional, mapeando objetos i* em casos de uso UML e, como uma evolução deste trabalho, o trabalho de [4], ferramenta JGOOSE, implementa novas características do

mapeamento entre i* e casos de uso UML, como por exemplo, o suporte para abrir arquivos Telos [16] gerados tanto pela ferramenta OME3 [8] quanto OpenOME [8].

Este artigo está estruturado conforme segue. A seção 2 apresenta um breve relato sobre o framework i*, exemplificando cada modelo proposto para uma organização que deseja desenvolver um sistema de compra e venda que apoie duas atividades diárias. A seção 3 apresenta uma descrição sobre casos de uso em UML e a seção 4 descreve a proposta que é a base para a construção da nova versão da ferramenta JGOOSE. A seção 5 apresenta as melhorias da ferramenta e a seção 6 mostra o uso da ferramenta no contexto do estudo de caso apresentado na seção 2. Na seção 7 são apresentadas as considerações finais do trabalho.

2 Framework i*

O framework i* proposto por Eric Yu [5] permite descrever aspectos de intencionalidade e motivações envolvendo atores em um ambiente organizacional. Para descrever estes aspectos são propostos dois modelos: O Modelo de Dependências Estratégicas (**SD**) e o Modelo de Razões Estratégicas (**SR**).

O Modelo SD é composto por **nós** e **ligações**. Os nós representam os **atores** no ambiente e as ligações são as **dependências** entre os atores. Por **ator** entende-se uma entidade que realiza ações para obter objetivos no contexto do ambiente organizacional. Atores dependem uns dos outros para atingir objetivos, realizar tarefas e obter recursos no ambiente organizacional. O ator que depende de alguma forma de outro ator é chamado de ***Depender*** e o ator que atende e satisfaz o *Depender* é denominado de ***Dependee*.** O objeto ou elemento de dependência entre *Depender* e *Dependee* é denominado de ***Dependum***. Portanto, haverá relacionamentos do tipo *Depender -> Dependum -> Dependee* [5][8].

O modelo de SR é complementar ao modelo de SD. O SR permite compreender e modelar de forma mais detalhada as razões associadas com cada ator e suas dependências [8]. Enquanto o modelo de SD provê um nível de abstração, no qual modela-se somente os relacionamentos externos entre atores, o modelo de SR permite uma maior compreensão a respeito das razões estratégicas de atores em relação aos processos da organização e como os mesmos são expressos. O modelo de SR auxilia no processo de Engenharia de Requisitos permitindo que elementos de processos e as razões por detrás dos mesmos sejam expressos.

Na Engenharia de Requisitos o modelo de SR pode ser utilizado para compreender como sistemas estão relacionados/envolvidos em rotinas de atores da organização para gerar alternativas e para modelar e suportar o raciocínio de atores organizacionais a respeito destas alternativas. A figura 1 apresenta um resumo dos elementos utilizados na técnica i*.

Para exemplificar o uso deste framework, nas figuras 2 e 3 são apresentados, respectivamente, o modelo SD e SR para uma organização que deseja obter um sistema computacional que apoie seu processo de compra e venda. Estes modelos serão utilizados posteriormente para mostrar o uso da nova versão da ferramenta JGOOSE. Na figura 2 (modelo SD) podemos observar que o ator “Cliente” depende do ator “Fun-

cionário" para atingir o objetivo de "Fazer Compra" e o "Funcionário" por sua vez depende do ator "Sistema" para atingir o objetivo de "Efetuar Venda". O "Funcionário" depende do ator "Cliente" para obter o recurso "Dados para Pagamento". O ator "Funcionário" também depende do ator "Sistema" para realizar o objetivo de "Consultar Produtos", bem como para realizar a tarefa de "Emitir Nota Fiscal". Ainda, o ator "Funcionário" depende do "Sistema" para atingir o objetivo-soft "Rapidez", o qual claramente representa um requisito não-funcional que deve ser atendido pelo sistema.

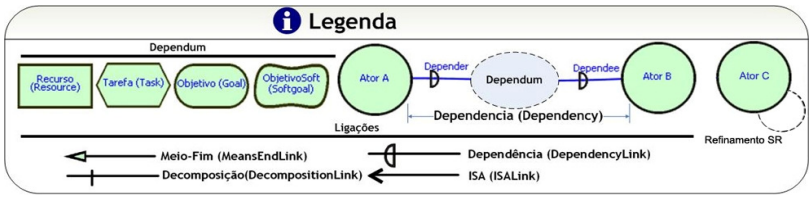

Fig. 1. Elementos da técnica i*.

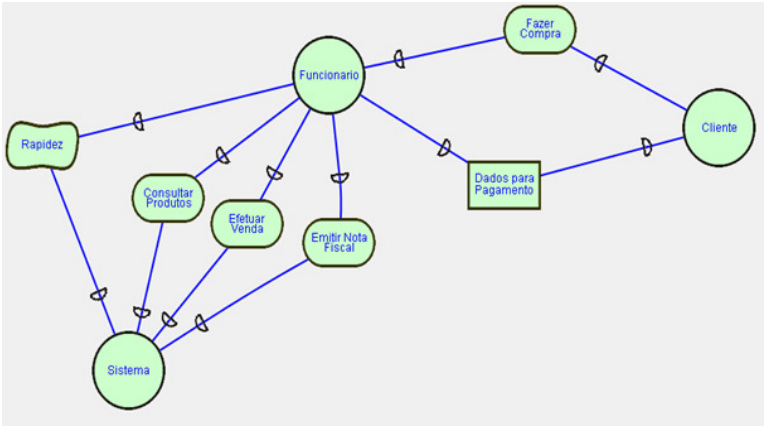

Fig. 2. Exemplo de Modelo de Dependências Estratégicas (SD)

Na figura 3 (modelo SR), podemos observar que a satisfação do objetivo "Consultar Produtos" pelo ator "Sistema" é obtida através dos objetivos "Buscar Produtos" e "Mostrar Produtos", os quais estão representados via mecanismo de decomposição de tarefas. A busca ainda pode ser feita de duas formas, "Via Mecanismo de Pesquisa" e "Via Browser" demonstrados pelas ligações meio-fim, nas quais, "Via Browser", e "Via Mecanismo de Pesquisa" representam meios para obter o fim de "Buscar Produtos". Da mesma forma, o objetivo "Mostrar Produtos" possui dois meios para ser alcançado, os quais são "Via Impressão" e "Via Tela". Já a satisfação do objetivo "Efetuar Venda", representado internamente no ator "Sistema" pela tarefa de mesmo nome, é levada a cabo pela realização das tarefas "Consultar Produtos", "Selecionar Produto", "Dar Baixa do Produto" e "Emitir Nota Fiscal". Cabe destacar que esta última tarefa pode ser realizada de forma independente para satisfazer o objetivo "Emitir Nota Fiscal" para o ator "Funcionário".

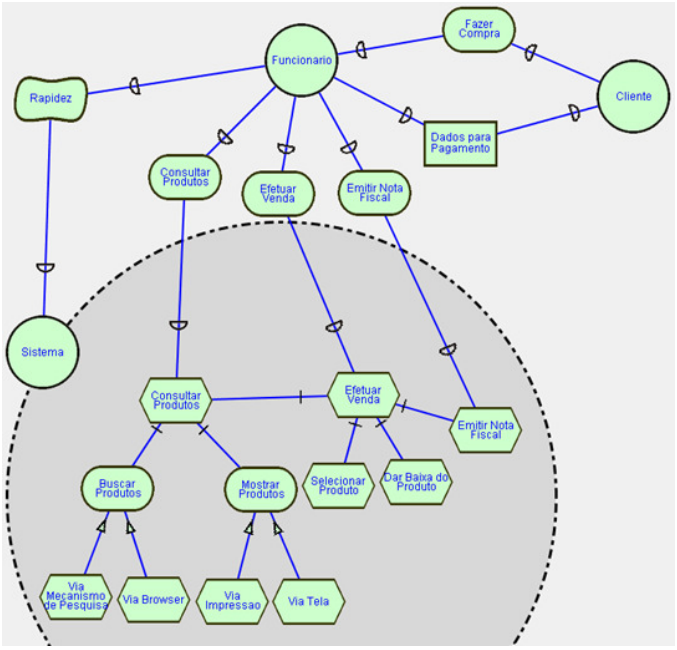

Fig. 3. Exemplo de Modelo de Razões Estratégicas (SR)

3 Casos de Uso em UML

O bom entendimento dos aspectos organizacionais através dos modelos produzidos pelo framework i* pode auxiliar na construção de um software que atenda as reais necessidades para as quais ele foi proposto. No entanto, é importante transformar essa representação de modelos organizacionais em implementação do sistema. Para apoiar essa transformação pode-se adotar a UML (Unified Modeling Language).

A UML pode ser bem descrita como "*uma linguagem-padrão para a elaboração da estrutura de projetos de software. Ela poderá ser empregada para a visualização, a especificação, a construção e a documentação de artefatos que façam uso de sistemas complexos de software*" [9]. A linguagem UML facilita modificações futuras no sistema auxiliando na qualidade do software em longo prazo.

Existem diversos diagramas e descrições em UML que auxiliam no desenvolvimento de um software, entre os quais podemos destacar os casos de uso. Os casos de uso podem ser expressos em forma de texto, permitindo representar as ações que devem ser realizadas entre o usuário e o sistema para levar a cabo o objetivo associado ao caso de uso. Isto permite uma melhor compreensão do caso de uso por todos os envolvidos. Este trabalho adota a descrição textual dos casos de uso via *template* proposto por Cockburn [11]. Juntamente com esta descrição textual, representa-se casos de uso através de diagramas. Uma referencia completa para diagramas de casos de uso pode ser encontrada em [9].

4 Proposta de Derivação de Casos de Uso a partir de i*

Na engenharia de requisitos é cada vez mais comum a ideia de que a fase de especificação de requisitos tenha informações relacionadas à organização, modelos de negócios e outras informações além das especificações do software. Estas informações propiciam um maior entendimento do contexto em que o sistema irá funcionar [12]. Uma dificuldade dos engenheiros de software é encontrar o que realmente é importante para o usuário, considerando os objetivos organizacionais. Para alcançar esse objetivo existem técnicas que auxiliam nesse processo, mas que necessitam de complementos [3]. Abordagens baseadas em cenários têm se destacado pela facilidade de entendimento dos usuários e desenvolvedores do sistema [13]. Essas abordagens têm ajudado a elicitação de requisitos e até mesmo a validação do sistema ao longo de seu desenvolvimento. Contudo, estas abordagens não expressam todos os desejos organizacionais envolvidos na criação do sistema [14][15].

Nesse contexto, uma possível alternativa para aproximar modelos organizacionais de modelos funcionais é proposta em [3]. Usando esta proposta, podemos elicitar e documentar casos de uso em UML a partir de modelos organizacionais construídos utilizando o framework i* [5]. Para este fim, são propostas algumas diretrizes que são brevemente descritas a seguir e aplicadas ao exemplo apresentado nas figuras 2 e 3.

1ºPasso da Proposta: Descoberta de atores

Diretriz 1: todo ator em i* deve ser analisado para um possível mapeamento para ator em caso de uso. Por exemplo, o ator "Funcionário" na Figura 2 é um candidato;

Diretriz 2: inicialmente, deve-se analisar se o ator em i* é externo ao sistema computacional pretendido. Caso o ator seja externo ao sistema, o mesmo é considerado candidato a ator em Casos de Uso. Por exemplo, o ator "Funcionário" é externo ao Sistema;

Diretriz 3: o ator i* candidato deve ter pelo menos uma dependência com o sistema computacional pretendido. Por exemplo, o ator "Funcionário" possui várias ligações de dependência com o sistema;

Diretriz 4: atores em i* relacionados através do mecanismo ISA, ou seja, com heranças de suas atividades nos modelos organizacionais e mapeados individualmente para atores em casos de uso (aplicando diretrizes 1, 2 e 3), serão relacionados no diagrama de casos de uso através do relacionamento do tipo <<generalização>>. Por exemplo, se existisse na Figura 2 um ator "Gerente" que tivesse uma relação ISA com o ator "Funcionário", ambos seriam candidatos a atores em casos de uso e no diagrama seriam relacionados via mecanismo de <<generalização>>.

2º Passo da Proposta: Descoberta de Casos de Uso.

Diretriz 5: para cada ator descoberto para o sistema (1º passo da proposta), devemos observar todas as suas dependências (*dependum*) como *dependee* em relação ao ator que representa o sistema computacional pretendido (*depender*), visando descobrir casos de uso para o ator.

SubDiretriz 5.1: deve-se avaliar as dependências do tipo objetivo associadas com o ator. Objetivos em i* podem ser mapeados para objetivos 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 mapeada 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 objetivo-*soft* associadas com o ator. Tipicamente uma dependência do tipo objetivo-*soft* em i* é um requisito não-funcional associado ao sistema pretendido. Por exemplo, o objetivo-*soft* "Rapidez" representa um requisito não-funcional do sistema como um todo.

Diretriz 6: analisar a situação especial na qual um ator de sistema (descoberto seguindo as diretrizes do passo 1) possui dependências (como *depender*) em relação ao ator em i* que representa o sistema computacional pretendido ou parte dele (ator → *dependum*→ sistema computacional). Por exemplo, o ator "Funcionário" possui as dependências do tipo objetivo "Consultar Produtos", "Efetuar Venda" e "Emitir Nota Fiscal" em relação ao sistema, e estas dependências geram casos de uso do sistema para o ator "Funcionário".

Diretriz 7: classificar cada caso de uso de acordo com seu tipo de objetivo associado (objetivo contextual, objetivo de usuário, objetivo de subfunção). A nova versão da ferramenta permite ao Engenheiro de Requisitos preencher e salvar esta informação.

3º Passo da Proposta: Descoberta e descrição do fluxo principal e alternativo dos Casos de Uso.

Diretriz 8: analisar cada ator e seus relacionamentos no Modelo de SR para extrair informações que possam conduzir à descrição de fluxos principal e alternativos, bem como, pré-condições e pós-condições dos casos de uso descobertos para o ator. Para isso precisamos analisar os subcomponentes em uma ligação de **decomposição de tarefa** mapeando-os para passos na descrição do cenário primário (fluxo principal) de casos de uso. Também devemos analisar ligações do tipo **meio-fim** mapeando os meios para passos alternativos na descrição de casos de uso. Por exemplo, as tarefas "Consultar Produtos", "Selecionar Produto", "Dar Baixa do Produto" e "Emitir Nota Fiscal" na figura 3, que decompõem a tarefa "Efetuar Venda" já mapeada para caso de uso, fazem parte do cenário principal deste caso de uso (ver figura 7). Também podemos observar que a tarefa "Consultar Produtos" é decomposta nos objetivos "Buscar Produtos" e "Mostrar Produtos", os quais geram os passos do cenário principal para este caso de uso. Observando mais atentamente a figura 3, verifica-se que há dois meios "Via Browser" e "Via Mecanismo de Pesquisa" para satisfazer o objetivo "Buscar Produtos". Um destes meios fará parte do cenário principal e o outro será mapeado para um passo no cenário secundário representando uma alternativa a execução do caso de uso. A nova versão da ferramenta JGOOSE automatiza este processo.

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. Por exemplo, observando a figura 3, o engenheiro de requisitos poderia considerar que a tarefa "Dar baixa do Produto", mesmo não sendo decomposta no modelo SR, necessita ser refinada em vários passos. Esta análise levaria a definição

de um novo caso de uso. A nova versão da ferramenta permite realizar esta mudança bem como salvá-la.

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. Por exemplo, observando as figuras 2 e 3 e aplicando as diretrizes dos passos 1 e 2, é possível mapear os atores e casos de uso para o sistema em questão (ver figura 8). Também, conforme exemplificado na diretriz 8, é possível perceber que os passos associados a "Consultar Produtos" e "Emitir Nota Fiscal são descritos textualmente (ver figura 7) e representados graficamente (ver figura 8) utilizando o estereotipo <<include>>). O mesmo ocorre no cenário secundário onde um determinado passo estende <<extend>> um caso de uso ou quando atores relacionados via ligação ISA no modelo SD (ver diretriz 4) são ligados no diagrama de casos de uso via estereotipo <<generalization>>.

5 Melhorando a Ferramenta JGOOSE

Apresentaremos uma breve descrição das características mantidas da versão da ferramenta JGOOSE em relação a sua versão inicial. A ferramenta melhorada utiliza os diagramas SD e SR do framework i*, criados no formato Telos (.tel) [16], oferecido pelo ambiente OME3 (Organization Modelling Environment) [8], que também possui integração com outras ferramentas que suportam a engenharia de software. O arquivo no formato Telos é a entrada da ferramenta JGOOSE. Então, efetua-se um prétratamento e em seguida são escolhidas as diretrizes que se deseja utilizar no mapeamento para casos de uso.

A nova versão da ferramenta mantém a sua estrutura inicial. O foco nesta nova versão está na completa implementação das diretrizes 8, 9 e 10 bem como na integração com outras ferramentas da engenharia de software. A estrutura do JGOOSE mostrando os passos tomados pela ferramenta para o seu funcionamento, retirada e modificada de [4], é apresentada na figura 4.

A linguagem utilizada para o desenvolvimento do JGOOSE é a linguagem orientada a objetos Java. O projeto JGOOSE foi desenvolvido na IDE NetBeans, por haver maior conhecimento e familiarização com a ferramenta, bem como por ser uma plataforma *opensource*. O processo de desenvolvimento adotado foi baseado nos princípios de processo iterativo e incremental. Definiram-se iterações mensais, reuniões semanais para acompanhamento, marcos de controle e revisão do projeto, bem como foi gerada uma documentação básica da ferramenta que inclui o diagrama de classes, diagrama de casos de uso e diagramas de seqüência.

Observando a figura 4, destaca-se que o foco desta nova versão está na conclusão dos passos 3 e 4. Assim, mais especificamente, nesta nova versão foram implementadas na sua íntegra as diretrizes 8, 9 e 10.

Em relação a diretriz 8, a mesma já estava implementada parcialmente gerando os passos do cenário principal de um caso de uso através da análise automatizada dos elementos no modelo SR. Contudo, não se permitia qualquer alteração do caso de uso gerado bem como não eram mapeados os relacionamentos do tipo meio-fim, cuja

análise visa encontrar passos alternativos no cenário secundário do caso de uso resultante. Também não se permitia que estas descrições textuais fossem salvas em documentos do tipo texto ou .doc, por exemplo. Essas melhorias permitem que o Engenheiro de Requisitos consiga gerenciar de forma mais adequada um caso de uso, especificando completamente fluxos principais e secundários e gerenciando-os sem necessidade de uso de outra ferramenta. Além disso, as descrições textuais armazenadas em documentos tipo texto ou .doc possibilitam maior integração e aderência por processos de software que usam estes formatos para descrição de casos de uso.

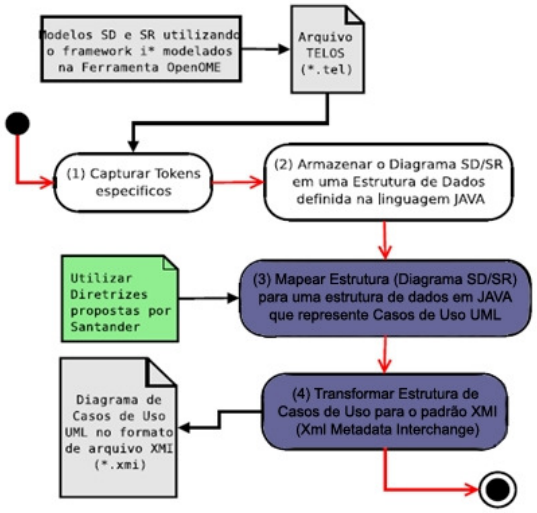

Fig. 4. Estrutura da ferramenta JGOOSE

No que tange a diretriz 9, agora a nova versão permite que o engenheiro de requisitos possa alterar o caso de uso textual gerado na própria ferramenta via *template* proposto por Cockburn [11], podendo agregar ou retirar informações que considere pertinentes. O Engenheiro de Requisitos, por exemplo, pode considerar que alguns passos devem gerar novos casos de uso salvando a descrição textual com esta mudança. Esse procedimento é realizado agregando ao passo analisado uma nova descrição contendo o estereotipo <<include>> seguido do caso de uso a ser incluído. Todos os campos do *template* podem ser alterados. Essa melhoria traz maior liberdade e flexibilidade para o Engenheiro de Requisitos, pois o mesmo pode gerenciar essas descrições e casos de uso, sem necessidade de retornar ao modelo i* para realizar alterações nos casos de uso.

Finalmente, a diretriz 10 é implementada gerando na íntegra o diagrama de casos de uso resultante do mapeamento, inclusive com os estereótipos do tipo <<include>>, <<extend>> e <<generalization>> usados para estruturar as especificações dos casos de uso. Este diagrama, na versão antiga, gerava o diagrama em um formato de imagem. Agora, na nova versão, o diagrama pode ser exportado para o formato XMI (Xml Metadata Interchange) [6], o que permite que o mesmo seja utilizado e modifi-

cado em outras ferramentas de modelagem orientada a objetos. Este é um aspecto importante que a versão anterior da ferramenta não suportava. É fato que engenheiros de requisitos precisam utilizar outras ferramentas para gerar outros artefatos UML. Permitir que os mesmos importem os diagramas gerados na ferramenta JGOOSE, diminui na prática o *gap* existente entre a modelagem organizacional e a modelagem funcional de sistemas computacionais.

Melhorias no que tange a aspectos de usabilidade também foram consideradas na nova versão da JGOOSE. Para verificar aspectos de uma boa interface, a nova versão da ferramenta foi implementada segundo os princípios de boa usabilidade propostos em [17]. Especificamente, verificou-se que nem todas as funcionalidades possuíam mensagens de sucesso e/ou erro implementadas, e que o sistema de ajuda precisava ser mais consistente. Também na fase de visualização dos casos de uso não existia a opção de voltar para a seleção das diretrizes, entre outros aspectos. Essas mudanças foram realizadas.

6 Estudo de Caso

Visando mostrar o funcionamento da nova versão da ferramenta, consideraremos o exemplo apresentado na seção 2 (figuras 2 e 3) e diretrizes apresentadas na seção 4. Inicialmente, foram criados os diagramas SD e SR das figuras 2 e 3 na ferramenta OME3 e no formato Telos (.tel). Utilizando a ferramenta JGOOSE e todos os seus mapeamentos, geramos os casos de uso.

A seguir mostraremos as telas da ferramenta desenvolvida seguindo o seu fluxo de execução. Cabe ressaltar que neste estudo de caso foi utilizado o arquivo Telos gerado para a figura 3. Realizamos o mapeamento completo incluindo a descrição dos cenários primários e secundários a partir do modelo SR.

A figura 5 mostra como é feita a seleção do ator referente ao sistema computacional pretendido. Essa seleção é realizada logo após a importação do arquivo .tel. No caso do exemplo da figura 3, entre os atores "Cliente", "Funcionário" e "Sistema" mapeados, seleciona-se este último como o sistema computacional a ser desenvolvido. Também é possível visualizar informações sobre o arquivo aberto e sobre os atores existentes, elementos mapeados do tipo objetivo, recurso, objetivo-soft e tarefa, como também, sobre as ligações (*links*) do tipo dependência, meio-fim, decomposição de tarefa e relações ISA mapeadas.

Na figura 6 é apresentada a tela de seleção das diretrizes que serão utilizadas no mapeamento dos casos de uso. Esta tela é mostrada logo após selecionar o ator sistema e acionar o botão "Selecionar Diretrizes" na figura 5. É possível naquela tela, selecionar uma ou mais diretrizes conforme desejado pelo Engenheiro de Requisitos. Também é possível acessar um breve tutorial sobre as diretrizes antes de solicitar o mapeamento. No caso do exemplo da figura 3, selecionamos todas as diretrizes para realizar um mapeamento completo.

Observando a figura 7 podemos verificar que através do botão "Diagrama", representado pela simbologia característica de diagramas de casos de uso, é possível visualizar o diagrama gerado. Por exemplo, selecionado o caso de uso "Efetuar Venda" e

acionado o botão “Diagrama” obtém-se o diagrama gerado na figura 8. É possível visualizar nesta figura que o único ator mapeado para o diagrama é “Funcionário” e que os casos de uso “Consultar Produtos”, “Efetuar Venda” e “Emitir Nota Fiscal”, associados ao mesmo, correspondem às dependências entre este ator e o sistema computacional pretendido (ver diretriz 6). Também é importante ressaltar o mapeamento automatizado do estereótipo <<include>> entre o caso de uso “Efetuar Venda” e os casos de uso “Consultar Produtos” e “Emitir Nota Fiscal”. Considerando que os mesmos já haviam sido mapeados para casos de uso no 2º passo da proposta (ver seção 4), eles então são “incluídos” no caso de uso “Efetuar Venda”.

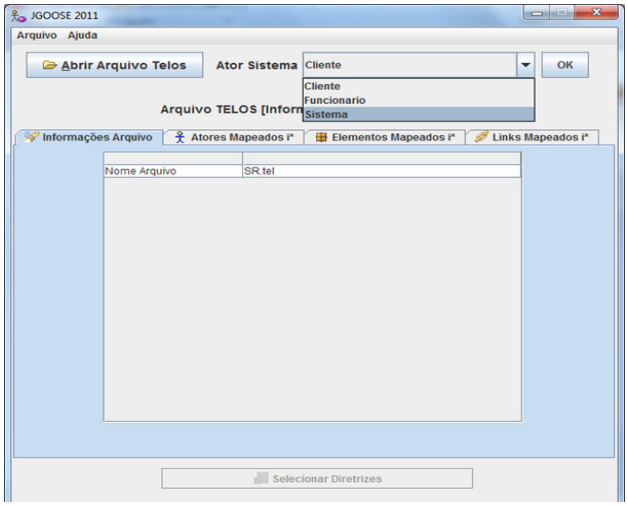

Fig. 5. Selecionar o Ator Referente ao Sistema

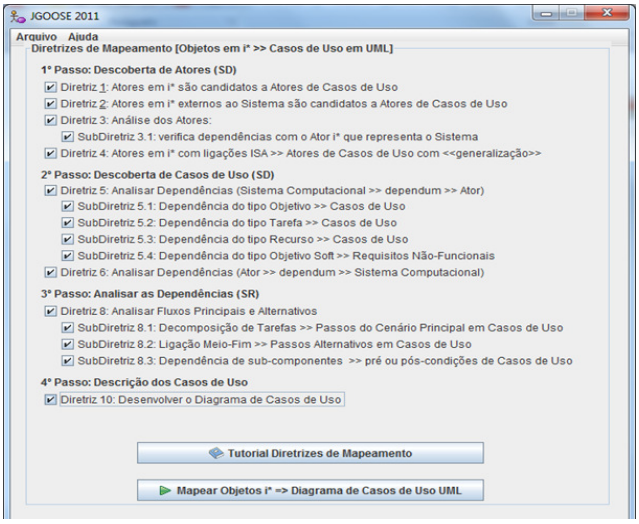

Fig. 6. Seleção das Diretrizes de Mapeamento

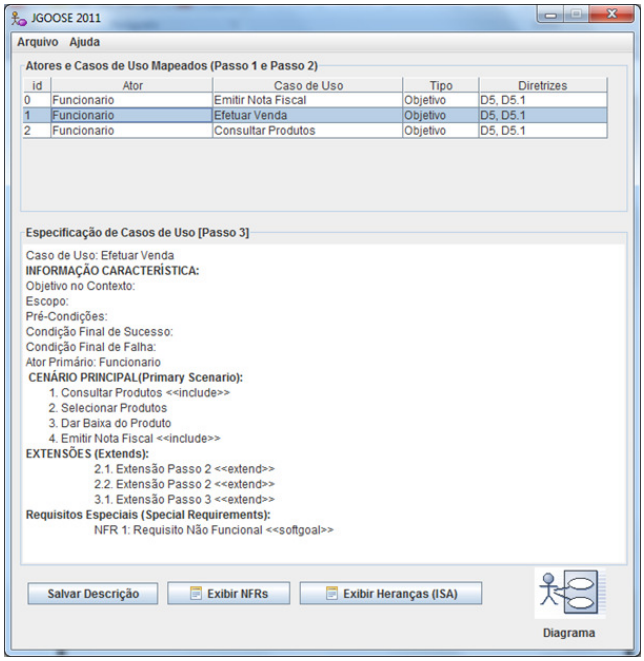

Fig. 7. Casos de Uso Gerados pelo JGOOSE

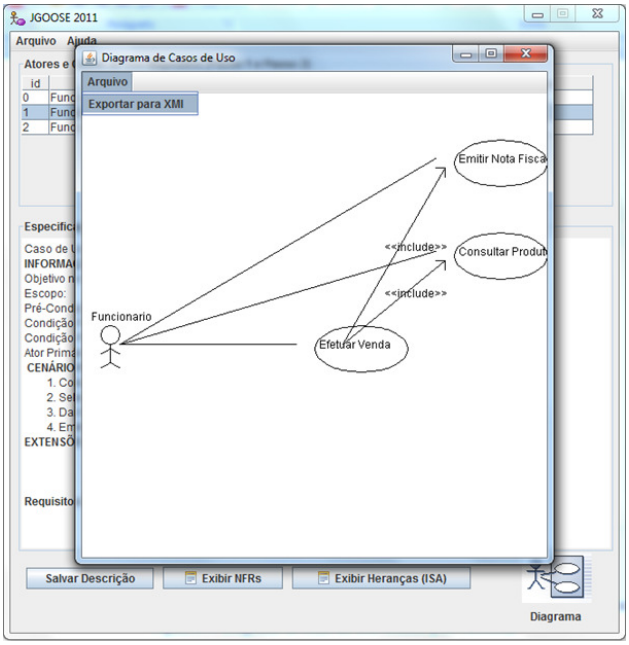

Fig. 8. Diagrama de Caso de Uso Gerado pelo JGOOSE

Adicionalmente, na figura 7, utilizando os botões correspondentes, é possível exibir os requisitos não-funcionais (NFRs) associados ao sistema como também exibir as heranças mapeadas a partir dos relacionamentos ISA existentes nos modelos organizacionais mapeados.

A figura 8 apresenta o diagrama de casos de uso completo para o estudo de caso da organização genérica de compra e venda com base nos modelos SD e SR das figuras 2 e 3. Este diagrama pode ser exportado para o formato XMI possibilitando o uso do diagrama em outras ferramentas que apóiam a engenharia de software. Essa funcionalidade do JGOOSE foi testada utilizando a ferramenta StarUML [18], disponibilizada gratuitamente, e Enterprise Architect [19], ferramenta proprietária.

7 Considerações Finais

Com o intuito de diminuir o *gap* existente entre a modelagem organizacional e modelagem funcional na engenharia de requisitos, neste trabalho apresentou-se a evolução da ferramenta JGOOSE visando torná-la operacional e passível de ser utilizada por engenheiros de requisitos e engenheiros de software. Consideramos que esta ferramenta pode auxiliar estes profissionais na importante fase do processo de desenvolvimento de software denominada de engenharia de requisitos.

A conclusão da implementação das diretrizes de mapeamento visando à integração dos diagramas i* e Casos de Uso UML, gerando de forma automatizada os Casos de Uso a partir dos modelos criados no framework i*, permitiu a conclusão da ferramenta e o seu uso de forma completa. Destaca-se entre as evoluções da ferramenta, a possibilidade de exportar os diagramas de casos de uso para o formato XMI bem como a possibilidade de salvar, editar e alterar as descrições textuais dos casos de uso nos editores de texto mais utilizados atualmente. Como trabalhos futuros, pretende-se utilizar a ferramenta no processo de ensino e aprendizagem da engenharia de requisitos com alunos de cursos de graduação em computação. Também se pretende utilizar a ferramenta em projetos reais junto a empresas desenvolvedoras de software. Estes estudos permitirão avaliar e melhorar a ferramenta com base na opinião de acadêmicos e profissionais da área da computação. Finalmente cabe ressaltar que a ferramenta apresentada é de código aberto e está disponível em [20].

Referências

1. i*: an agent and goal-oriented modeling framework. 2011. Consultado na Internet: http://www.cs.toronto.edu/km/istar/, em Novembro de 2011.
2. YU, E. S. K. Towards modelling and reasoning support for early-phase requirements engineering. In: 3RD IEEE INTERNATIONAL SYMPOSIUM ON REQUIREMENTS ENGINEERING - RE97, 1997. Washington D.C., USA: [s.n.], (1997). p.226–235.
3. SANTANDER, V. F., CASTRO, J. F. B., Deriving Use Cases from Organizational Modeling In: IEEE Joint International Requirements Engineering Conference - RE'02, p. 32-39, Essen, Germany, (2002).

4. VICENTE, A. A., SANTANDER, V. F. A., CASTRO, J. B., FREITAS da Silva, IVONEI., R. M., FRANCISCO G., JGOOSE: A Requirements Enginnering Tool to Integrate i* Organizational Modeling with Use Cases in UML. Ingeniare. Revista chilena de ingeniería. , v.17, p.6 - 20,(2009).
5. YU, E. GIORGINI, P. MAIDEN, N.MYLOPOULOS, J. Social Modeling for Requirements Engineering. 2011. The MIT Press (January, 2011).
6. XMI. XML MetadataInterchange. 2011. Consultado na Internet: http://www.omg.org/spec/XMI/, em Outubro de 2011.
7. SANTOS. B. S. IStar Tool - Uma proposta de ferramenta para modelagem de i*, Dissertação de Mestrado, Centro de Informática, Universidade Federal de Pernambuco, 2008.
8. i* Wiki. 2011. Consultado na Internet: http://istar.rwth-aachen.de/tiki-index.php?page=i*%20Wiki%20Home.
9. BOOCH, G. RUMBAUGH, J. JACOBSON, I. UML: Guia do Usuário, 2º Edição, Editora Campus, 2005.
10. Rational Unified Process. Rational Unified Process, 2002. Consultado na Internet: http://www.wthreex.com/rup/portugues/index.htm
11. COCKBURN, A. Writing Effective Use Cases. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2000.
12. ERIKSSON, H.-E.; PENKER, M. Business Modeling With UML: Business Patterns at Work. New York, NY, USA: John Wiley Sons, Inc., 1998.
13. BREITMAN, K.K., LEITE, J.C.S.P., A Framework for Scenario Evolution, Third International Conference on Requirements Engineering – III, IEEE Computer Society Press. Los Alamitos, CA, U.S.A, (1998), pp 214 – 221.
14. POTTS, C., ScenIC: A Strategy for Inquiry-Driven Requirements Determination, In Proceedings of the Fourth IEEE International Symposium on Requirements Engineering - RE'99, June 7-11, Ireland, (1999).
15. SUTCLIFFE, A., GREGORIADES, A., Validating Functional Requirements with Scenarios, In: IEEE Joint International Requirements Engineering Conference, RE'02, University of Essen, Germany, September, 9-13, (2002), pp. 181-188.
16. MYLOPOULOS, J., BORGIDA, A., JARKE, M., & KOUBARAKIS, M. Telos: Representing knowledge about information systems. (1990). ACM Transaction on Information Systems, 8(4), pp. 483-497.
17. ErgoList, 2011. Consultado na internet: http://www.labiutil.inf.ufsc.br/ergolist/, em Outubro de 2011.
18. StarUML. StarUML - The Open Source UML/MDA Platform. Consultado na Internet: http://staruml.sourceforge.net/en/, em Outubro de 2011.
19. Enterprise Architect. UML tools for software development and modelling – Enterprise Architect UML modeling tool. Consultado na Internet: http://www.sparxsystems.com/, em Outubro de 2011.
20. JGOOSE, 2011. Disponível em http://www.inf.unioeste.br/~victor/JGOOSE.
21. Briscke, M., Santander, V. F. A., Castro, J. F. B., GOOSE: Uma Ferramenta para Integrar Modelagem Organizacional e Modelagem Funcional In: Jornadas Chilenas de Computación - V Workshop Chileno de Ingeniería de Software, Valdivia, Chile. 2005.
22. Pedroza, F.P., Alencar, F.M.R., Castro, J., Silva, F.R.C., and Santander, V.F.A. Ferramentas para Suporte do Mapeamento da Modelagem i* para a UML: eXtended GOOD - XGOOD e GOOSE. In Proceedings of WER. 2004, 164-175.