Modelando SOA a partir de Modelos Organizacionais

Orlando Silva de Oliveira[1], Carla Silva[2]

[1]Instituto Federal do Sertão Pernambucano, Departamento de Informática
BR 232, Km 508, Zona Rural, Salgueiro-PE, Brasil.
orlando.silva@ifsertao-pe.edu.br
[2]Universidade Federal de Pernambuco, Centro de Informática
Cidade Universitária, 50740-560, Recife-PE.
ctlls@cin.ufpe.br

Abstract. A Arquitetura Orientada a Serviços (SOA) oferece um modelo arquitetônico que visa aprimoramentos para as empresas. Nesse modelo, os serviços permitem que os objetivos estratégicos sejam alcançados. Entretanto, o desenvolvimento de sistemas que utilizam SOA tem exigido novas estratégias da Engenharia de Requisitos (RE). De outro lado, observa-se que as abordagens da Engenharia de Requisitos Orientada a Objetivos (GORE) têm ganhado notoriedade nos últimos anos. De fato, as abordagens orientadas a objetivos apresentam mecanismos não ofertados pela RE tradicional, como por exemplos capturar os objetivos dos *stakeholders* e as características do sistema em um mesmo modelo. Assim, é possível usar esse modelo para analisar e identificar se o sistema proposto atende aos objetivos dos *stakeholders*. Esse é um importante tipo de análise no contexto organizacional. No entanto, a literatura não apresenta uma forma sistemática para identificar serviços em modelos de requisitos orientados a objetivos. Além disso, há uma lacuna a ser preenchida na transição dos requisitos (espaço do problema) para a arquitetura equivalente (espaço da solução), no contexto da SOA. Dessa forma, este trabalho apresenta uma abordagem sistemática para a identificação de serviços em modelos GORE descritos em i* e a posterior obtenção da arquitetura SOA descrita em SoaML. A abordagem foi avaliada com um estudo empírico com usuários reais.

Keywords: Arquitetura Orientada a Serviços (SOA); Engenharia de Requisitos Orientada a Objetivos (GORE); Framework i*; SoaML.

1 Introdução

As soluções convencionais de TI nem sempre conseguem acompanhar o ritmo das constantes mudanças que os negócios exigem num mercado cada vez mais dinâmico. Nesse cenário, é cada vez mais frequente a busca pelos benefícios oferecidos por estratégias emergentes, como por exemplo a *Service-Oriented Architecture* (SOA).

A SOA disponibiliza um modelo arquitetônico que permite aprimorar a eficiência, a agilidade e a produtividade de uma empresa. Nesse modelo, os serviços são os principais meios para se alcançar os objetivos estratégicos da organização [1]. Todavia, a

SOA tem apresentado novos desafios à Engenharia de *Software* (ES), em especial aos tópicos que se referem à disciplina de RE [2].

De outro lado, pesquisas apontam a importância da inclusão de elementos sociais e organizacionais na análise e desenvolvimento do *software*, dando origem à *Goal-Oriented Requirements Engineering* (GORE) [3] que, ao contrário das técnicas tradicionais, permite modelar as motivações (objetivos) que tornam um determinado sistema necessário num contexto organizacional [4]. Das técnicas GORE, destaca-se o *framework* i* [5] que, devido às suas características, permite a modelagem organizacional.

Diante dos benefícios da GORE, [7] propõe a sua utilização num contexto orientado a serviços, através de uma abordagem baseada no i* [5] e por isso chamada neste trabalho de i* Orientado a Serviços. Nessa abordagem, é possível visualizar como os objetivos dos *stakeholders* serão alcançados através de serviços. Contudo, os detalhes do processo de identificação dos serviços e da arquitetura de *software* a nível de componentes são omitidos pela abordagem de [7]. Desse modo, o presente trabalho propõe a descrição da arquitetura de sistemas orientados a serviços usando a notação padrão SoaML [8] a partir de modelos organizacionais descritos com o i* [5].

2 Referencial Teórico

2.1 Engenharia de Requisitos Orientada a Objetivos

Dificuldades na utilização das técnicas tradicionais da RE promoveram maior interesse por abordagens GORE, as quais se baseiam na identificação dos objetivos (*goals*) dos *stakeholders* e na transformação desses objetivos em requisitos de *software*, permitindo abordar questões que expliquem o porquê de um determinado objetivo ser necessário, como ele pode ser alcançado e quem é o responsável por ele [4].

O *framework* i* é destacado como uma das principais abordagens GORE, ele se baseia na identificação de atores e suas dependências para a modelagem organizacional [5]. Uma das vantagens desse *framework* é permitir analisar o contexto organizacional e representar os detalhes do sistema a ser desenvolvido no mesmo modelo.

O i* possui dois modelos básicos [5]: o *Strategic Dependency* (SD), que descreve relacionamentos de dependência entre atores e o *Strategic Rationale* (SR), que explica como os atores atingem seus objetivos.

No SD um ator é uma entidade autônoma e social que tem objetivos estratégicos e intencionalidade. No i*, um ator é representado por um círculo com uma descrição textual ao centro. Uma ligação de dependência relaciona dois atores, indicando qual depende do outro para atingir um objetivo, executar uma tarefa ou receber um recurso.

O outro modelo do i* é o modelo SR, que descreve os elementos internos dos atores, evidenciando o processo pelo qual os objetivos são alcançados, as tarefas realizadas, os recursos fornecidos e os *softgoals* operacionalizados. No SR, há dois outros tipos de relacionamentos [5]: o meio-fim, que representa uma alternativa e a decomposição de tarefas, no qual um conjunto de elementos permite a realização de uma tarefa específica.

Segundo [7], o i* apresenta como pontos fracos a ausência de mecanismos que promovam a granularidade e o refinamento dos modelos. Assim, foi proposta uma extensão do i* para sanar tais deficiências [7]. Essa proposta combina elementos do paradigma orientado a serviços e elementos do i*, permitindo a representação de modelos organizacionais modularizados através de serviços. Essa abordagem será chamada neste trabalho de i* Orientado a Serviços.

No i* Orientado a Serviços [7], um ator pode ocupar o papel de consumidor (*depender*) ou provedor de serviços (*dependee*). O serviço, representado por um paralelogramo, é um recurso abstrato que dá suporte para que um consumidor possa alcançar seus objetivos organizacionais através de um processo de negócio. Desse modo, as funcionalidades dos serviços são orquestradas através dos processos de negócio da organização. O i* Orientado a Serviços é composto de três modelos (Fig. 1). O Modelo Global (Fig. 1-a) é uma representação abstrata do contexto organizacional, no qual os serviços são providos e consumidos. O Modelo de Processo (Fig. 1-b) apresenta os processos de negócios relacionados aos serviços. Por último, o Modelo de Protocolos (Fig. 1-c) expressa a coreografia entre os serviços.

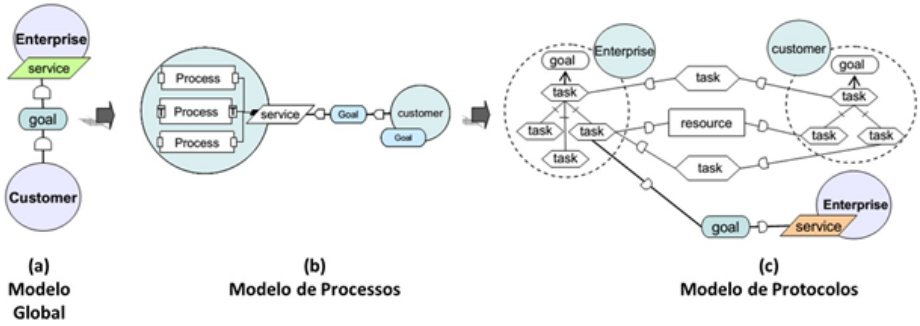

Fig. 1. Modelos do i* Orientado a Serviços.

2.2 Arquitetura Orientada a Serviços e SoaML

A Arquitetura de *Software* (AS) trata da macroestrutura do sistema, incluindo o seu comportamento geral e os elementos computacionais que o compõe [11].

Para criar o projeto arquitetural, o arquiteto de *software* se vale de algum estilo arquitetônico, os quais funcionam como modelos para a construção da arquitetura do *software* [12]. SOA é um estilo arquitetônico, no qual os serviços dão suporte aos objetivos estratégicos da organização [1]. Na SOA, o serviço é o elemento chave, caracterizado como recurso abstrato que representa uma *capability* [13].

Em SOA, *capability* se refere às habilidades disponibilizas pelo negócio ou por sistemas de *software* [14]. Assim, a lógica das soluções é dividida em *capabilities*, sendo que cada *capability* é utilizada para solucionar um tipo específico de preocupação [1].

A UML (*Unified Modeling Language*) [10] tem sido utilizada para a documentação de arquiteturas de sistemas de *software* [15]. Porém, a UML não fornece suporte nativo à modelagem de serviços, mas traz mecanismos, chamados de *profiles*, que possibilitam a sua aplicação em domínios específicos, como a SOA [16]. Um *profile* chamado de

SoaML [8] permite a criação de diagramas UML para a representação SOA. A SoaML permite descrever capacidades, mensagens, interfaces, contratos, participantes e detalhar a arquitetura dos serviços.

3 Abordagem Proposta

3.1 Contextualização

Na SOA, a identificação de serviços é feita através da separação de interesses [1]. Assim, as operações automatizáveis serão encapsuladas em serviços [17]. Para que a identificação de serviços seja possível em modelos i*, faz-se necessário definir quais elementos desse *framework* serão relacionados ao conceito de capacidade da SOA.

Sendo assim, através do modelo SR é possível observar quais elementos intencionais do ator *dependee* satisfazem uma requisição de algum *depender*. Esses elementos intencionais são organizados através de grafos compostos por rotinas[1]. No modelo SR, um ator *dependee* utiliza rotinas para atingir suas próprias metas organizacionais. Mas, os elementos pertencentes a uma rotina também podem ser utilizados para satisfazer as necessidades expostas por um ator *depender* [5,6]. Assim, cada rotina de um ator *depender* apresenta contextos funcionais compostos por elementos intencionais, que representam as operações necessárias para o atendimento dos anseios dos atores envolvidos. Diante disso, contextos funcionais foram usados para realizar a separação de interesses no i*. De modo que, os contextos funcionais das rotinas foram relacionados ao conceito de *capability* da SOA, visto que em SOA uma *capability* representa um conjunto coeso de funções ou recursos que um serviço pode oferecer [8].

3.2 Detalhamento da Abordagem

A abordagem proposta auxilia na obtenção do projeto inicial da arquitetura de *software* para sistemas orientados a serviços. O projeto arquitetural será obtido através de modelos organizacionais descritos através do *framework* i* original [5] e representado através de diagramas da SoaML [8]. A abordagem possui três atividades principais, exibidas na Fig. 2 através de um diagrama BPMN [9] e detalhadas a seguir.

1. **Identificação dos Serviços Candidatos:** recebe um modelo organizacional descrito em i*, identifica rotinas automatizáveis e as encapsula em serviços candidatos, delega os serviços iniciais para um ator que representa o sistema a ser desenvolvido e fornece como saída o Modelo Global [7] do i* Orientado a Serviços.
2. **Análise dos Serviços Candidatos:** analisa a modelagem do i* Orientado a Serviços produzida pela etapa anterior através das operações de conjuntos sugeridas por [18], o que permite melhorar a coesão, granularidade e reuso dos serviços.
3. **Mapeamento entre Modelos:** recebe o modelo de requisitos i* Orientado a Serviços final e produz como saída um modelo arquitetural básico descrito com diagramas

[1] Uma rotina representa um determinado curso de ação, ou seja, uma alternativa, dentre várias alternativas, utilizada para se atingir um objetivo específico [5,6].

da SoaML [8]. Esse procedimento é feito através do mapeamento do Modelo Global do i* Orientado a Serviços para modelos arquiteturais em diagramas SoaML.

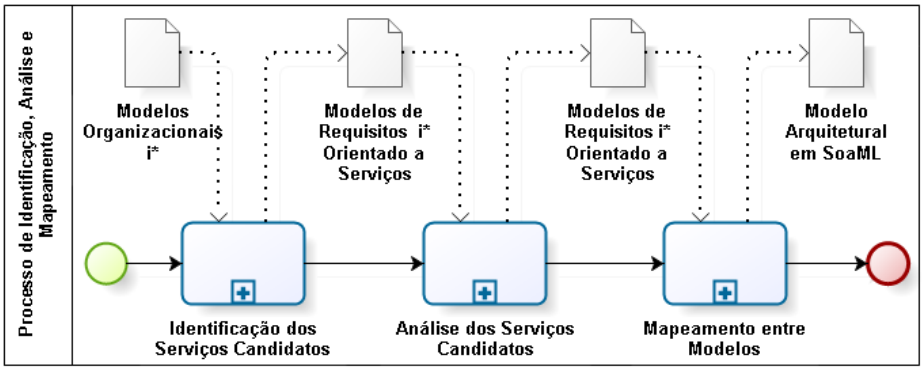

Fig. 2. Processo da abordagem proposta.

Visando melhorar a compreensão da abordagem proposta, foi utilizado um exemplo de aplicação numa modelagem extraída de [5]. Essa modelagem trata dos processos utilizados por três atores num cenário de seguros de saúde, conforme Fig. 3. Nessa figura, percebe-se que o objetivo geral do ator *Paciente* é o de *Estar Bem* e que para alcançar esse objetivo o ator poderá se valer de duas estratégias: *Estar Bem Por Ter Um Seguro Integral* ou *Estar Bem Por Ter Um Seguro Gerenciado*. Em ambas as situações, o *Paciente* terá de *Comprar Seguro* e *Receber Tratamento*. No caso de *Comprar Seguro*, o *Paciente* depende do ator *Cia de Seguros*. Para receber o tratamento desejado o ator *Paciente* depende do ator *Médico*.

Para o ator *Médico*, a meta principal é *Conduzir a Prática Médica*, que tem como objetivo geral o *Paciente Ser Curado*. Para isso, o *Médico* poderá tomar dois caminhos diferentes: *Tratar Paciente de Cobertura Integral* ou *Tratar Paciente de Cobertura Gerenciada*. Para pacientes de cobertura gerenciada o *Médico* terá que *Diagnosticar Doença*, *Tratar Doença* e *Gerar Fatura* para a *Cia de Seguros*. Para que o *Médico* possa tratar uma doença, ele dependerá que o *Paciente* realize a tarefa de *Tomar Medicamentos* e dependerá da *Pré-Aprovação* e do *Reembolso do Tratamento* por parte da *Cia de Seguros*. Do lado do ator *Cia de Seguros* a meta principal é a de *Conduzir a Empresa de Seguros de Saúde*. Para alcançar esta meta a *Cia de Seguros* terá duas alternativas: *Conduzir o Seguro Integral* ou *Conduzir o Seguro Gerenciado*. Nos dois casos a *Cia de Seguros* terá de *Vender Apólice* para o *Paciente* e deverá *Reembolsar o Tratamento* prestado pelo ator *Médico*. Para um seguro do tipo gerenciado a *Cia de Seguros* terá de fornecer também uma *Pré-Aprovação de Tratamento* para que o ator *Médico* possa oferecer o tratamento adequadamente para o *Paciente*. Após a compreensão do domínio do exemplo, segue-se com as atividades que compõem a abordagem.

Atividade I – Identificação dos Serviços Candidatos

Nessa atividade busca-se a identificação de um conjunto de operações automatizáveis (*capability*) que possam ser agregadas através de serviços candidatos.

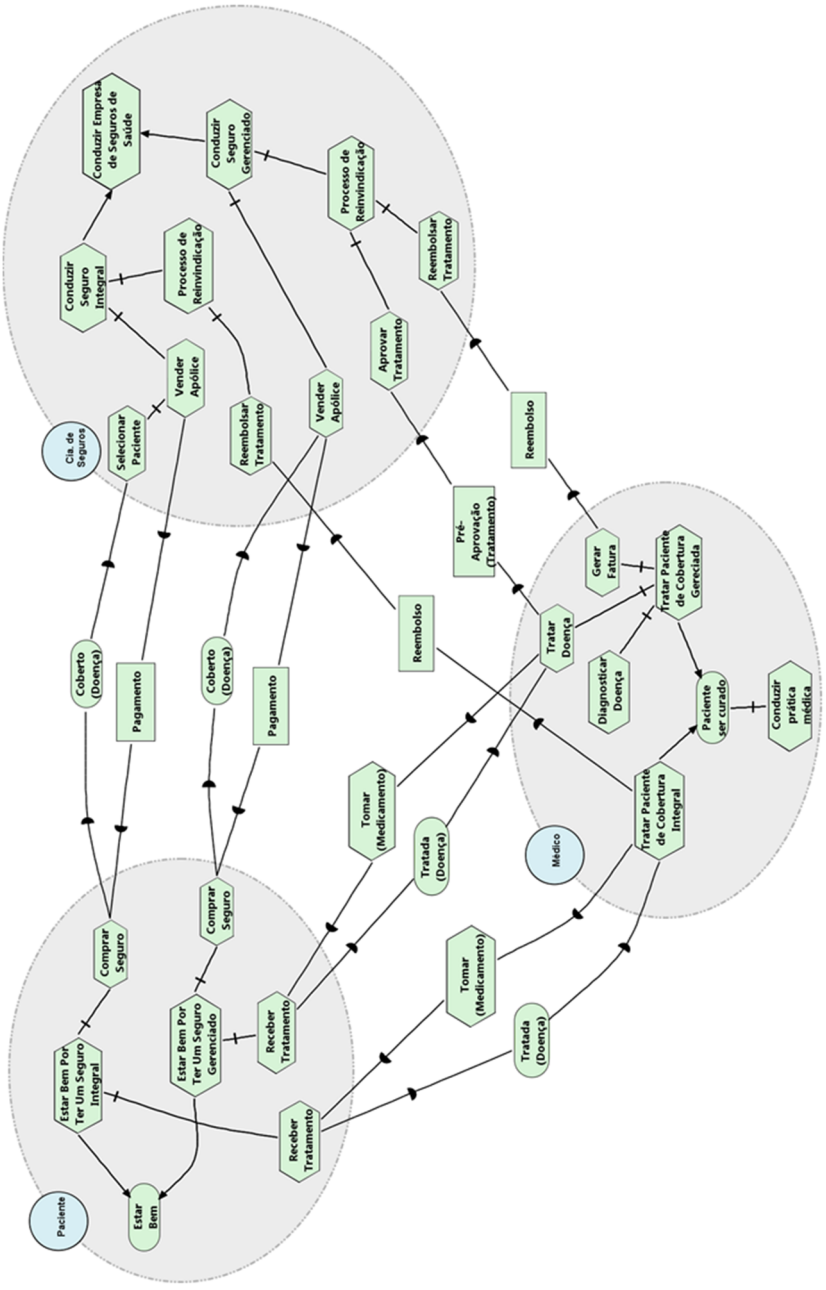

Fig. 3. Modelagem i* de uma empresa de seguros de saúde adaptada de [5].

No i*, as operações serão agrupamentos de elementos intencionais automatizáveis por um serviço. A identificação dos serviços iniciais é dividida nas três etapas a seguir.

Etapa I – Identificação dos atores que oferecem rotinas automatizáveis.

Na modelagem da Fig. 3 identificou-se que o ator *Cia de Seguros* oferece duas rotinas que apresentam tarefas automatizáveis, cujas raízes são *Conduzir Seguro Integral* e *Conduzir Seguro Gerenciado*, respectivamente.

Etapa II – Encapsular as rotinas automatizáveis em serviços candidatos.

Cada uma das rotinas automatizáveis identificadas será encapsulada em um serviço candidato distinto, representado por um paralelogramo. Desse modo, essas rotinas deram origem aos serviços *Gestão de Seguro Integral* e *Gestão de Seguro Gerenciado*.

Etapa III - Delegação dos serviços candidatos para um ator sistémico.

O processo de delegação dos serviços é baseado nas regras de transformação horizontal apresentadas por [19]. Assim, ambas a rotinas dos serviços candidatos foram movidas para o ator *Sistema*, que representa o *software* a ser desenvolvido, conforme Fig. 4.

Os resultados obtidos através dos processos realizados pela atividade de identificação dos serviços podem ser vistos na Fig. 4, na qual são mostrados os serviços *Gestão de Seguro Integral* e *Gestão de Seguro Gerenciado* alocados dentro da fronteira do ator *Sistema*. Na modelagem da Fig. 4 é possível perceber que os atores *Paciente*, *Médico* e *Cia. De Seguros* são consumidores dos serviços ofertados pelo ator *Sistema.*

Atividade II – Análise dos Serviços Candidatos.

O processo de análise dos serviços candidatos é baseado nas operações de conjuntos sugeridas por [18]. Assim, as operações de união, interseção, decomposição, subconjunto e subtração serão utilizadas sobre os contextos funcionais dos serviços.

Através da Fig. 4 é possível perceber que o serviço *Gestão de Seguro Integral* possui três contextos funcionais distintos: *venda de apólices*, *reinvindicações* (reembolso) e a *condução do seguro* em si. Esse serviço será submetido a uma operação de decomposição para desagregar tais contextos, gerando os serviços: *Vendas* e *Reinvindicações*. Os serviços gerados pela decomposição devem manter uma ligação de *features* com o serviço original, como sugerido em [7]. No caso dos serviços *Vendas* e *Reinvindicações*, a ligação utilizada entre os serviços e seu controlador foi a de *features* obrigatória, visto que ambos os serviços gerados são usados pelo serviço controlador *Gestão de Seguro Integral*.

Cenário similar ao do serviço *Gestão de Seguro Integral* é o do serviço *Gestão de Seguro Gerenciado*, com o acréscimo da tarefa *Aprovar Tratamento* ao *Processo de Reinvindicações*. Dessa forma, o serviço candidato *Gestão de Seguro Gerenciado* foi decomposto de modo similar, gerando os mesmos serviços candidatos: *Vendas* e *Reinvindicações*.

Visando melhorar o nível de coesão e reuso das operações, os serviços *Vendas* e *Reinvindicações*, gerados a partir de *Gestão de Seguro Integral*, foram consolidados aos serviços *Vendas* e *Reinvindicações* gerados de *Gestão de Seguro Gerenciado*. Esse procedimento foi realizado através de uma operação de união de conjuntos. Salienta-se

que as ligações originais entre os elementos intencionais devem ser mantidas entre os elementos movidos para os novos serviços, permitindo rastrear a origem dos elementos. Contudo, essas ligações devem receber uma tonalidade de cor mais clara para não interferir na visualização das ligações de *features* utilizadas nas relações entre os serviços.

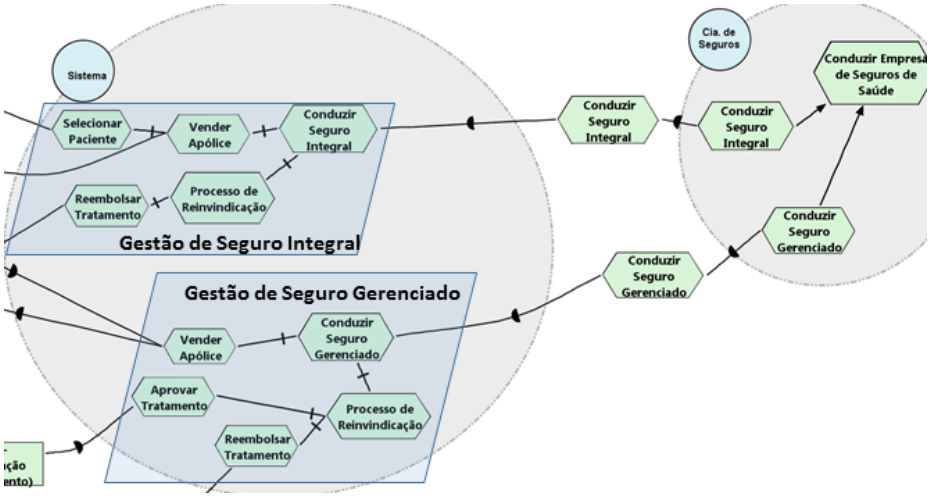

Fig. 4. Serviços candidatos iniciais delegados ao ator Sistema.

Agora o ator *Sistema* possui quatro serviços candidatos: *Gestão de Seguros Integral* e *Gestão de Seguros Gerenciado* que dependem dos serviços *Vendas* e *Reinvindicações*. Contudo, observou-se a existência de similaridade de contexto entre as operações dos serviços candidatos *Gestão de Seguro Integral* e *Gestão de Seguro Gerenciado*, fato que permitiu uma consolidação entre esses serviços através de uma operação de união de conjuntos, gerando um único serviço candidato chamado de *Gestão de Seguros*, que usa *Vendas* e *Reinvindicações*, conforme modelagem final (Fig. 5). Os serviços candidatos podem variar a depender da experiência do analista, características do projeto e da prioridade dos princípios SOA que estão sendo considerados [14].

Atividade III – Mapeamento entre Modelos.

A seguir, é apresentado o processo de mapeamento da modelagem i* Orientada a Serviços para diagramas da SoaML. Todo o mapeamento é aplicado sobre a modelagem final, apresentado na Fig. 5. O mapeamento produzirá os seis principais diagramas da SoaML, permitindo visões complementares da arquitetura de serviços.

I – Diagrama de Capacidades

Os elementos funcionais presentes em cada serviço do modelo apresentado na Fig. 5 foram traduzidos para as operações de uma capacidade em SoaML, conforme Fig. 6(a).

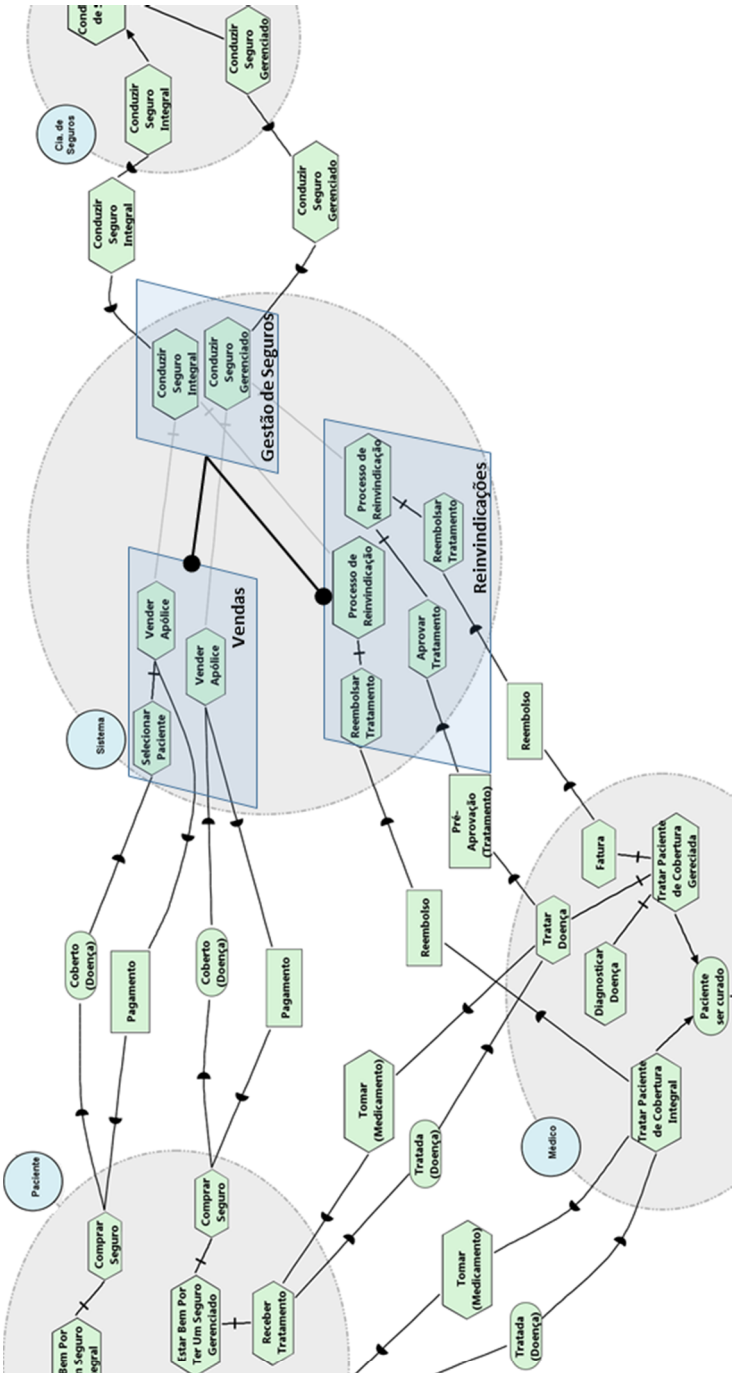

Fig. 5. Modelo Global de [7] com os serviços finais.

Para o diagrama de capacidades, cada serviço deu origem a uma capacidade de mesmo nome. Os elementos intencionais de mesmo nome em um mesmo serviço tiveram seus nomes diferenciados nas capacidades da SoaML através da sua localização de origem. Por exemplo, o elemento *Vender Apólice* que originalmente foi decomposto do serviço *Conduzir Seguro Integral*, recebeu o nome de *venderApoliceSeguroIntegral* na capacidade *Vendas* da SoaML. Esse procedimento foi adotado em todo o processo de mapeamento para a distinção de nomes.

II – Diagrama de Mensagens

No i* uma dependência cumprida implica em duas mensagens: uma mensagem de solicitação do *depender* e uma mensagem de resposta do *dependee* [20], as quais foram representadas através do diagrama de mensagens da SoaML. Assim, foi considerado que uma mensagem de requisição será identificada com base no seu *dependum* e terá no seu identificador o acréscimo da palavra *request*. A mensagem de resposta terá o mesmo nome, porém a palavra *request* será substituída pela palavra *response*, representando a resposta obtida. Na Fig. 6(b) são apresentadas as mensagens *request* e *response* de três dependências: *Cobertura*, *Reembolso* e *Pagamento*.

III – Diagrama de Interfaces.

Uma interface SoaML serve para descrever as operações, parâmetros de entrada e saída e possíveis exceções [21]. Assim, os diagramas de capacidades foram utilizados para a especificação das operações disponibilizadas pelas interfaces dos serviços em SoaML. Os parâmetros de entrada e saída são obtidos através da análise de como as mensagens são trocadas entre consumidores e provedores de serviços. A Fig. 6(c) apresenta a interface do serviço *GestaoDeSeguros* que expõe uma capacidade e realiza uma interface provedora.

IV – Diagrama de Contratos

Em SoaML um contrato especifica quais os provedores e consumidores trabalham juntos para atingir um determinado objetivo. Além disso, pode apresentar a coreografia entre os participantes do serviço [8]. O contrato mostrado na Fig. 6(d) refere-se ao serviço *GestaoDeSeguros* obtido a partir da relação de dependência entre o consumidor (*Cia. de Seguros*) e o respectivo serviço do ator *Sistema*. Esse contrato é feito entre dois papéis: um consumidor anônimo e o provedor, com a interface *GestaoDeSeguros*.

V – Diagrama de Participantes.

Os participantes são entidades que proveem ou utilizam serviços. No i* os participantes são os atores envolvidos na modelagem. Desse modo, os participantes e os respectivos serviços presentes na modelagem da Fig. 5 são: *Paciente* que consome o serviço *Vendas*, *Médico* que consome o serviço *Reinvindicações*, *Cia de Seguros* que consome o serviço *Gestão de Seguros* e o ator *Sistema* que provê os serviços *Vendas*, *Reinvindicações* e *Gestão de Seguros*. Esses atores e serviços foram traduzidos para o diagrama

de participantes disponibilizado pela SoaML. A Fig. 6(e) apresenta cada um dos atores citados, representados através de diagramas de participantes da SoaML. Percebe-se que cada participante possui uma porta com uma interface para o consumo ou a prestação de serviços. No caso de serviços bidirecionais (*Vendas*) há uma interface provedora e uma consumidora na porta do serviço. Assim, *Paciente* solicita o serviço *Vendas* do provedor *Sistema*. *Medico* requer do *Sistema* o serviço *Reinvindicacoes*. *CompanhiaDeSeguros* solicita o serviço *GestaoDeSeguros* do *Sistema*. Já o participante *Sistema* é o provedor de todos os serviços solicitados.

VI – Diagrama de Arquitetura.

O diagrama de arquitetura de serviços da SoaML serve para demonstrar como os participantes e serviços trabalham em conjunto para um propósito [8]. O mesmo comportamento é identificado em modelos i*, nos quais os atores colaboram entre si para um propósito [7]. Dessa forma, cada ator do i* Orientado a Serviços pode ser modelado como um participante de uma arquitetura de serviços e as dependências entre os atores podem ser expressas através dos contratos, como já foi demonstrado anteriormente.

Através da Fig. 6(f) observam-se os participantes *Paciente*, *Medico*, *CompanhiaDeSeguros* e *Sistema* formando uma arquitetura de serviços para que, de forma colaborativa, atinjam um determinado objetivo. Nessa figura é possível perceber o papel que cada um dos participantes ocupa diante dos contratos dos serviços existentes na arquitetura. Desse modo, *Paciente*, *Medico* e *CompanhiaDeSeguros* são consumidores dos serviços *Vendas*, *Reinvindicacoes* e *GestaoDeSeguros*, respectivamente. Já o participante *Sistema* é o provedor dos três serviços mencionados. Esses papéis são identificados através da descrição existente na ligação do participante com o contrato do serviço.

Um outro nível de abstração no qual uma arquitetura de serviços poderia ser apresentada é mostrando como as partes internas de um participante trabalham juntas para atingir um objetivo específico.

Ao término da atividade de mapeamento foi possível obter seis diagramas SoaML, permitindo uma representação da arquitetura de serviços através de múltiplas visões.

4 Avaliação da Abordagem

Para avaliar a abordagem apresentada foi realizado um estudo empírico com 6 participantes reais (4 estudantes e 2 profissionais da área de desenvolvimento de *software* de uma instituição de ensino superior). Nesse estudo, características relacionadas ao processo foram analisadas. A condução do estudo foi feita com o auxílio do método GQM [23], que busca objetivos que possam ser traduzidos em medições quantitativas.

O objetivo principal da avaliação foi analisar o processo da abordagem proposta para mensurar o nível de dificuldade em relação à sua compreensão e uso, do ponto de vista do engenheiro de *software*, no contexto do projeto de *software*. Para a coleta dos dados foi utilizado um questionário objetivo, utilizando a escala Likert [24] para a coleta das opiniões dos participantes. Os tópicos avaliados foram: (i) nível de dificuldade na compreensão da abordagem e o (ii) nível de dificuldade na aplicação da abordagem.

Como resultado, metade dos participantes (50%) consideraram que a abordagem utilizada apresenta um nível razoável de compreensão. Além disso, a maioria dos participantes (67%) consideraram que o processo adotado pela abordagem proposta é de fácil utilização.

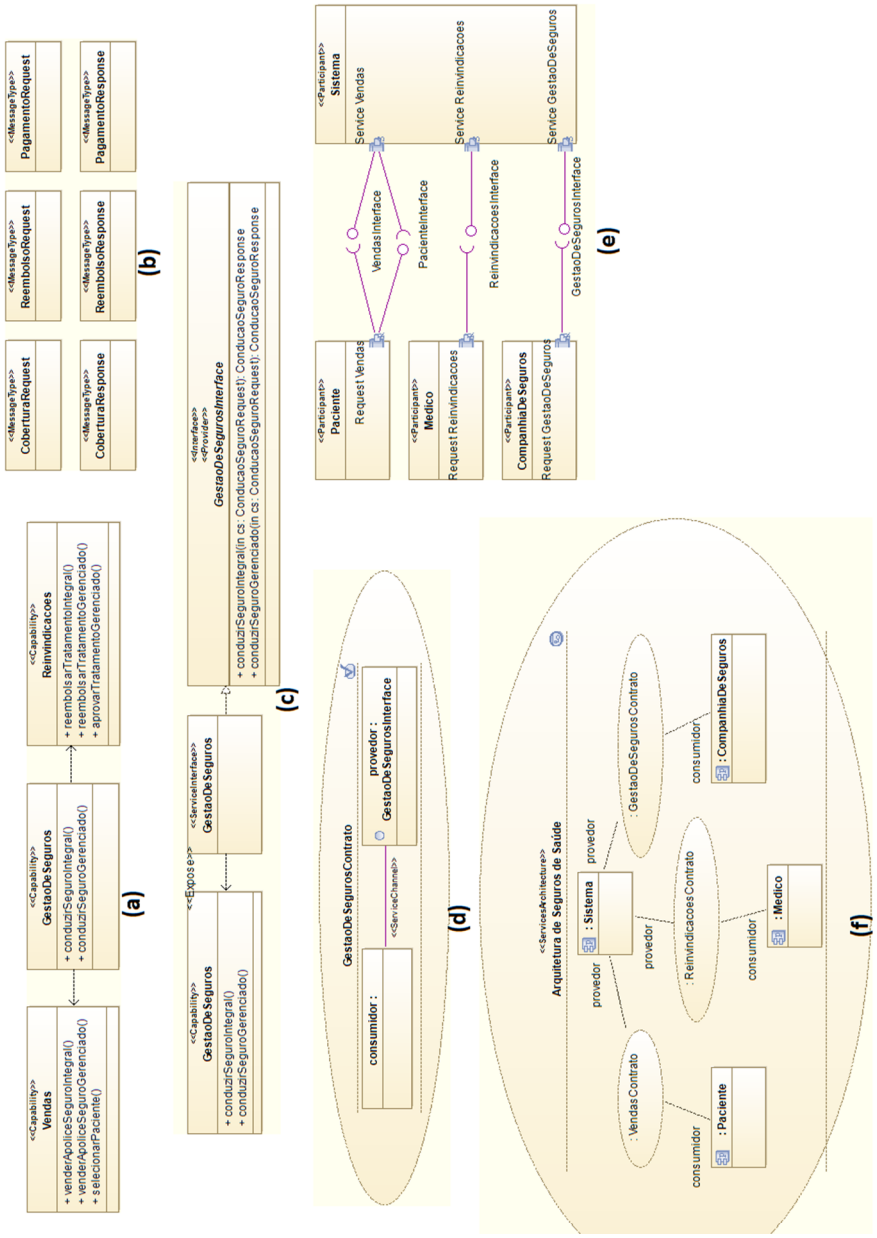

Fig. 6. Artefatos SOA produzidos pela atividade de mapeamento.

5 Trabalhos Relacionados

Durante a pesquisa, foram identificados alguns trabalhos que tratam do relacionamento dos requisitos e SOA. Cada um dos trabalhos apresenta uma abordagem diferente para que a partir de uma modelagem organizacional se obtenha um projeto arquitetural dos serviços identificados para a composição do sistema de *software* a ser desenvolvido.

Os trabalhos analisados foram os de [26], que utiliza BPMN para a representação dos processos de negócio e SoaML para representação arquitetural; [22], que permite identificar serviços através de padrões em diagramas EPC; o de [27], que a partir de casos de uso gera a arquitetura de serviços; e [25], uma abordagem GORE para a representação de serviços e arquitetura através de versões modificadas do i*.

6 Conclusão e Trabalhos Futuros

Este trabalho apresentou uma abordagem GORE que permite a transição de modelos organizacionais descritos em i* [5] para modelos arquitetônicos descritos através da linguagem SoaML [8]. A utilização dessa abordagem possibilita uma obtenção sistemática de um projeto de SOA para sistemas de *software*.

A abordagem apresentada inclui um processo para a identificação, análise e modelagem de serviços candidatos a partir de modelos i*. Além disso, a abordagem não é dependente de nenhuma tecnologia.

As principais contribuições do trabalho são: (i) a modelagem organizacional através do i* no contexto da SOA, pois ao contrário do BPMN o i* representa as motivações dos *stakeholders*; (ii) identificação de serviços em modelos i*; (iii) análise de serviços em modelos organizacionais através das operações de conjuntos de [18]; e (iv) a transição de modelos i* para modelos arquiteturais em SoaML.

Por fim, alguns trabalhos futuros são propostos: (i) analisar a abordagem com outras variantes do *framework* i*; (ii) pesquisar e selecionar um método apropriado de criação de modelos i* com o intuito de facilitar a obtenção dos modelos arquiteturais; (iii) desenvolvimento de uma ferramenta de apoio para abordagem; (iv) estender a abordagem para contemplar outras etapas do desenvolvimento dos serviços; e (v) a realização de novas pesquisas que relacionem SOA e GORE, visando melhorar a abordagem proposta e/ou surgimento de novas abordagens.

Agradecimentos

Esta pesquisa foi parcialmente apoiada pela FACEPE.

Referências

1. Erl, T., (2009) SOA: princípios do design de serviços, Pearson Prentice Hall.
2. Bano, M., Zowghi, D., Ikram, N. and Niazi, M. (2014). What makes service oriented requirements engineering challenging? A qualitative study. IET Software, Vol.8.

3. Castro, J., Alencar, F. (2013). Uso de Modelagem Social na Engenharia de Requisitos. Jornada de Atualização de Informática - JAI 2013. Maceió-AL, Brazil.
4. Aljahdali, S., Bano, J., Hundewale, N., (2011). Goal Oriented Requirements Engineering - A Review. In: 24th International Conference on Computer Applications in Industry and Engineering, Honolulu, Hawaii, USA, CAINE, 2011, p.16-18.
5. Yu, E. (1995). Modelling strategic relationships for process reengineering. Tese de Doutorado, University of Toronto, Canadá.
6. Yu, E. (1997). Towards Modeling and Reasoning Support for Early-Phase Requirements Engineering, pp.226, In: 3rd IEEE Intl. Symp. on Requirements Eng. (RE'97).
7. Estrada, H., (2008). A service oriented approach for the i* framework, Phd. Thesis. Universidad Politecnica de Valencia, Espanha.
8. OMG, (2012). Notation (SoaML). OMG Specification. Object Management Group.
9. OMG, (2011). Notation (BPMN). OMG Specification. Object Management Group.
10. OMG, (2011). Notation (UML). OMG Specification. Object Management Group.
11. Guo, J., (2003), An Approach for Modeling and Designing Software Architecture, in Engineering of Computer-Based Systems - ECBS, IEEE Computer Society, pp. 89-97.
12. Pressman, R. S., (2011). Engenharia de software. 7.ed., AMGH.
13. W3C. (2004). Web Services Glossary. <http://www.w3.org/TR/ws-gloss/>. Acesso em 20 de fevereiro de 2014.
14. Brocke, J. vom., Rosemann, M. (2013). Manual de BPM. Porto Alegre, Bookman.
15. Nakagawa, E. Y. (2006). Uma contribuição ao projeto arquitetural de ambientes de engenharia de software. Tese de Doutorado, Universidade de São Paulo, Brasil.
16. Diirr, T., Azevedo, L. G., Faria, F., Santoro, F., & Baião, F. (2013). Utilizando SoaML para Modelagem e Geração de Código de Serviços em uma Abordagem SOA. Cadernos do IME-Série Informática, 30, 38-49.
17. Fugita, H. S.; Hirama, K. (2012). SOA: modelagem, análise e design. Rio de Janeiro, Elsevier.
18. Marks, E. A.; Bell, M. (2006). Service-Oriented Architecture: a planning and implementation guide for business and techonology, John Willey & Sons Inc.
19. Lucena, M., Castro, J., Silva, C., Alencar, F. and Santos, E., (2011). STREAM: A Strategy for Transition between Requirements Models and Architectural Models. In: 26th Annual ACM Symposium on Applied Computing, pages 699 - 704. ACM Press.
20. Mahfouz, A., Barroca, L., Laney, R., & Nuseibeh, B. (2010). Requirements-driven design of service-oriented interactions. Software, IEEE, 27(6), 25-32.
21. Revoredo, K. C., Azevedo, L. G., Santoro, F., & Baião, F. (2010). Estudos do Profile SoaML. RelaTe-DIA, 4(1).
22. Souza, A., Diirr, T., Azevedo, L. G., Santoro, F. (2011). Identificação e Análise de Serviços a partir de Modelos de Processos de Negócio: Um estudo de caso. RelaTe-DIA, 5(1).
23. Basilli, V. R., Caldiera, G., & Rombach, H. D. (1994). Goal Question Metric Paradigm. Encyclopedia of Software Engineering.
24. Likert, R. (1932). A Technique for the Measurement of Attitudes. Archives of Psychology, 140, 1–55.
25. Becerra, C., Franch, X., and Astudillo, H., (2010). From i* Models to Service Oriented Architecture Models., in Marten van Sinderen & Brahmananda Sapkota, ed., Architectures, Concepts and Technologies for Service Oriented Computing, SciTePress, pp. 52-63.
26. Lemrabet, Y., Touzi, J., Clin, D., Bigand, M., & Bourey, J. P. (2010). Mapping of BPMN models into UML models using SoaML profile. MOSIM'10.
27. Braga, V. (2011). Um Processo para Projeto Arquitetural de Software Dirigido a Modelos e Orientado a Serviços. Dissertação, Universidade Federal de Pernambuco, Brasil.