Modelando Serviços a partir de Processos de Negócio: Uma Abordagem Dirigida a Modelos

Alternative Title: Service Modeling from Business Process: A Model-Driven Approach

João Pedro Bittencourt
Departamento de Ciência da Computação
Universidade Federal da Bahia
jpqueiroz@dcc.ufba.br

Edlane Proencia
Departamento de Ciência da Computação
Universidade Federal da Bahia
edlaneproencia@dcc.ufba.br

Rita Suzana Pitangueira Maciel
Departamento de Ciência da Computação
Universidade Federal da Bahia
ritasuzana@dcc.ufba.br

RESUMO

A modelagem de processos de negócio ajuda a entender e identificar atividades realizadas por uma organização, as quais podem ser usadas como fonte de requisitos para a modelagem de sistemas. Ela também pode ser usada para derivar serviços candidatos para apoiarem esses requisitos. Entretanto, gerenciar informações entre modelos com diferentes níveis de abstração não é uma tarefa simples. Vários trabalhos foram propostos com o objetivo de resolver este problema através do uso de técnicas de derivação de serviço através de processos de negócios. Este artigo apresenta um método para auxiliar a derivação de serviços dos processos de negócio e utiliza uma abordagem Dirigida a Modelos e heurísticas de forma a obter um modelo de serviços capaz de gerar código. A abordagem é dividida em duas fases. Na primeira fase, um processo de negócio, modelado em BPMN, é transformado em um diagrama de atividades UML. Na segunda fase, o modelo UML é transformado em um modelo SoaML.

Palavras-Chave

BPMN; SOA; modelagem de processos; modelagem de serviços; diagrama de atividades.

ABSTRACT

Business processes modeling helps to understand and identify activities performed in an organization, which serve as a source of requirements for system modeling. It also can be used to discover candidate services to support these requirements. However, managing information between models of different levels of abstraction is not a trivial task. Several works have been proposed to solve this problem by deriving services from business models. This paper presents a method to assist services derivation from business process using the Model Driven Development approach and heuristics and in order to achieve services models that can be used to generate code. The method is divided into two phases. In the first phase, a business process, modeled in BPMN, is transformed into a UML activity diagram. In the second phase, UML models are transformed into a SoaML model.

SBSI 2016, May 17–20, 2016, Florianópolis, Santa Catarina, Brazil.

Categories and Subject Descriptors

K.6.0 [**Management of Computing and Information System**]: General. [**Operating Systems**]: Process Management.

General Terms

Management.

Keywords

BPMN; SOA; Business modeling; Service modeling; Diagram activity.

1. INTRODUÇÃO

Os modelos de processos de negócio de uma organização podem servir como fonte de requisitos para o desenvolvimento de serviços que apoiem a construção de aplicações baseadas em uma arquitetura orientada a serviço (*SOA-Service Oriented Architecture*) [6]. Identificação de serviços é uma tarefa essencial para obter as vantagens reais de SOA, porém, o gerenciamento de informações em diferentes níveis de abstração não é uma tarefa trivial [9].

Na modelagem de sistemas, a linguagem mais utilizada atualmente nos contextos empresariais é a UML (*Unified Modeling Language*) [15]. A UML possui diferentes diagramas para modelar aspectos relacionados com a estrutura e o comportamento de um sistema em diferentes níveis de abstração. Dentre esses diagramas, pode-se destacar o DA (Diagrama de Atividades), que representa os fluxos conduzidos por processamentos, mostrando o fluxo de controle de uma atividade para outra.

Várias propostas têm sido feitas no sentido de guiar os desenvolvedores na derivação de serviços a partir de modelos de processos de negócio [1, 2]. A maior parte delas está focada na derivação diretamente dos modelos de processos de negócio. Sendo assim, oferecem, comumente, modelos incompletos ou com

granularidade inadequada, devido ao diferente nível de abstração existente entre a camada de negócio e a camada serviços [11]. Estas abordagens passam da abstração no nível do problema para uma solução de sistema já no nível arquitetural (SOA). A fase de identificação de uma solução computacional mais adequada, independente de estilo arquitetural ou plataforma estaria, a princípio, não sendo contemplada por estas abordagens.

Este texto apresenta uma estratégia de derivação de serviços que, a partir de modelos BPMN (Business Process Model and Notation) [13], tendo o Diagrama de Atividade como modelo intermediário, usando heurísticas e estratégias de desenvolvimento dirigido a modelos gera serviços em SoaML (*SOA Modeling Language*). Na nossa abordagem, Diagramas de Atividades são utilizados como uma camada intermediária entre a modelagem de negócios e a modelagem arquitetural de um sistema de informação, sendo possível então enriquecer os modelos em SoaML, melhorando a completude dos serviços gerados. O Diagrama de Atividades permite modelar o processo de negócio do ponto de vista do sistema de informação. Assim, pode-se enriquecer o modelo com informações ao nível do sistema, que não podiam ser adicionadas ao modelo BPMN sem fugir do escopo da modelagem do negócio, como por exemplo, detalhar ou adicionar tarefas de sistema que realizam um determinado processo de negócio. Esta flexibilidade fornecida pelo Diagrama de Atividades o torna um excelente candidato para preencher a lacuna entre os diferentes níveis de abstração existentes entre os modelos BPMN e os modelos SoaML. Adicionalmente, transformações de modelos, baseadas em heurísticas para descoberta de serviços, são utilizadas para gerar uma arquitetura orientada a serviços expressos na linguagem SoaML [14] para minimizar o trabalho laboral de e suscetível a erros da modelagem manual.

A estratégia proposta resultou em um método composto de duas fases: (i) BPMN-DA: transformação de um diagrama BPMN em Diagrama de Atividades e (ii) DA-SOAML: análise e detalhamento do DA gerado em (i) e (iii) transformação deste em um diagrama SoaML. As regras de transformação foram baseadas em dois conjuntos de heurísticas: um conjunto para identificar relações entre os elementos do BPMN com os do DA e outro para identificar padrões de comportamento no DA que pudessem compor uma arquitetura de serviços. Para isso foram adaptadas as heurísticas propostas em [1, 2].

Este artigo está organizado da seguinte forma: a Seção II apresenta o estado da arte e os trabalhos relacionados. Na Seção III, a abordagem desenvolvida no trabalho é detalhada. Na Seção IV, são mostrados exemplos de uso e os resultados obtidos. Finalmente, na Seção V é apresentada a conclusão e os trabalhos futuros.

2. ABORDAGENS PARA DERIVAÇÃO DE SERVIÇOS

SOA representa um modelo arquitetural que visa melhorar a agilidade do desenvolvimento de aplicações [11] e tem como principal objetivo o reuso intenso dos seus serviços para que o desenvolvimento de uma aplicação seja, primordialmente, a composição e a coordenação dos serviços já implementados, aumentando o reuso e diminuindo o dispêndio de recursos [10].

O Desenvolvimento Dirigido a Modelos (DDM) [12] utiliza modelos para especificar e implementar sistemas. Utilizando DDM, aplicações são modeladas em vários níveis de abstração e sob diferentes perspectivas até se obter o código. Diferente das outras abordagens de desenvolvimento de software, os modelos são os artefatos centrais para a construção das aplicações, não o código [10].

Diversas abordagens têm sido propostas com o intuito de derivar serviços tomando como base a modelagem de processos de negócio. Enquanto algumas utilizam uma abordagem manual, outras utilizam DDM para potencializar o reuso dos modelos e auxiliar o trabalho do desenvolvedor durante modelagem de uma arquitetura de serviços.

O framework MINERVA BPSOM [2] guia a derivação de serviços iniciando com a modelagem dos processos de negócio em BPMN. Uma ferramenta de suporte permite que o modelo seja exportado para um formato particular, como XMI (*XML Metadata Interchange*), XSD (*XML Schema Definition*) ou XPDL (*XML Process Definition Language*) e é usada para que desenvolvedores possam marcar o modelo e estendê-lo com informações adicionais para ser utilizado nas transformações. Quando todas as informações são definidas, transformações codificadas em QVT (*Query/View/Transformation*) podem ser executadas para obter os modelos de serviço em SoaML para as tarefas que possuem suporte a serviço. Contudo, as regras de transformação deste framework cobrem apenas os elementos *Pool*, *Lane*, *ActivityMessage* e *PoolMessage* da BPMN, originando apenas os elementos *Participant*, *ServicePoint* e *RequestPoint* em SoaML.

FAZZIKI et. al. [4] propõe um framework que contém técnicas de modelagem de domínio, de modelagem BPMN, de modelagem em SOA e regras de transformação na linguagem ATL. O processo proposto possui 4 passos. O primeiro é o desenvolvimento do *Computacional Independent Model* (CIM), utilizando UML e BPMN, o segundo é a definição do *Platform Independent Model* (PIM), que deve ser representado em diagramas de classe e de sequência para em seguida serem descritos usando conceitos de SOA e de Arquitetura Orientada a Componentes (COA), o terceiro é composto pelas transformações de CIM para PIM SoaML, de PIM SoaML para PIM COA e de PIM COA para o *Platform-Specific Model* (PSM) e o quarto e último passo é a geração de código e desenvolvimento do sistema em um ambiente específico a partir do PSM. Da mesma forma que em MINERVA, suas regras possuem uma baixa cobertura do domínio, abrangendo apenas os elementos *Pool*, *Lane*, *StartMessage* e *EndMessage* da BPMN, gerando apenas os elementos *Participant*, *Service*, *Request* e *ServiceContract*.

No trabalho de SADOVYKH et. al. [7], é proposta uma abordagem que distingue os níveis *Business Architecture Model* (BAM), *System Architecture Model* (SAM) e *Implementation*, em SOA, para sistemas de negócio que correspondem respectivamente aos níveis CIM, PIM e PSM na terminologia MDA. No nível BAM é feita a modelagem do escopo do negócio em BPMN, de onde serão derivadas informações semânticas, no formato de um subconjunto do modelo de classes da UML. Este modelo será utilizado para definir a sequência de dados entre tarefas e participantes, assim como derivar o modelo de persistência. Os esqueletos dos modelos de implementação podem ser gerados a partir dos modelos SAM SoaML. O trabalho foi implementado parcialmente na ferramenta SHAPE, porém, segundo o próprio autor, os passos ainda precisam ser refinados e aperfeiçoados. Além disso, o trabalho foi baseado na BPMN 1.0,

não sendo totalmente compatível com a versão mais nova desta notação, a BPMN 2.0.

Entre os trabalhos que utilizam abordagens manuais destaca-se o trabalho de AZEVEDO et. al. [1], onde é proposto um método baseado em heurísticas para sistematizar a análise dos modelos de processos de negócio, especificados utilizando FAD (*Function Allocation Diagram*) e diagramas EPC (*Event Process Chain*), e identificar serviços candidatos a partir dele. Esse método tem como ponto de partida uma demanda de desenvolvimento de software descrita por um conjunto de requisitos que serão suportados por componentes de software, onde cada componente pode ser implementado como um serviço ou como funcionalidades de uma aplicação. Em seguida, os modelos de processos que estejam relacionados à demanda devem ser analisados a fim de identificar serviços candidatos. Apesar desta abordagem apresentar uma estratégia de ampla cobertura dos elementos do modelo de negócio, a ausência de suporte automatizado a torna laboriosa e suscetível a erros.

A proposta apresentada neste trabalho utiliza heurísticas e o DA e baseia-se na abordagem DDM. Com o uso das heurísticas é possível ampliar a cobertura de elementos do modelo de negócios que podem ser usados para identificar serviços, e com o uso de transformações de modelo dar suporte mais automatizado às tarefas do desenvolvedor. Adicionalmente, espera-se melhorar a granularidade e expor as interfaces e operações dos serviços gerados e tornar a análise dos requisitos do sistema mais próxima dos desenvolvedores, a proposta usa Diagramas de Atividades da UML como modelo intermediário entre o modelo de processo de negócio e o modelo de serviços.

3. PROPOSTA DE DERIVAÇÃO DE SERVIÇOS A PARTIR DE MODELOS DE NEGÓCIO

Esta seção descreve as etapas do método proposto, através das fases BPMN2DA e DA2SOAML. O método possui duas fases. Na primeira fase (Fase I: BPMN2DA), processos modelados em BPMN são transformados em DA, e na segunda fase (Fase II: DA2SOAML) o DA pode ser modificado e transformado em um modelo SoaML.

As regras de mapeamento das transformações são baseadas em dois conjuntos de heurísticas: (i) para identificação de elementos do diagrama de atividades a partir de um modelo BPMN e (ii) para identificação de serviços no DA. As regras de mapeamento foram codificadas utilizando a linguagem ATL.

Após a Fase 1 é permitido ao desenvolvedor alterar o DA antes de executar a transformação DA2SOAML. Após as alterações sobre o DA, é executada a transformação deste para SoaML. Neste momento, o desenvolvedor pode completar o modelo obtido com informações pertinentes à camada arquitetural, como, por exemplo, adicionar mais detalhes aos serviços gerados pela transformação ou adicionar operações aos mesmos. A partir do modelo SoaML pode-se, então, gerar código para uma plataforma específica, como Java ou .Net. Para a especificação dos modelos e suas transformações, foram utilizados os metamodelos BPMN 2.0, UML 2.4.4 e SoaML 1.0.1, definidos pelo OMG. Foi utilizada a ferramenta Eclipse para fazer a modelagem do BPMN e realizar as transformações dos modelos. Foram usados, também, os plug-ins BPMN2 Modeler (para modelagem do BPMN) e Papyrus (para modelagem do DA e do SoaML).

3.1 Fase I: BPMN2DA – Modelagem e Transformação do BPMN para Diagrama de Atividades

O método é iniciado com a atividade de modelagem do processo de negócio, em BPMN, que se encontra na camada de negócio. Terminada a modelagem do processo de negócio. Neste modelo é aplicada a transformação BPMN2DA, com o objetivo de explicitar as atividades que compõem o processo de negócio

Tabela 1. Regras de mapeamento BPMN para UML

Regra	Elemento BPMN	Elemento UML	Descrição da Regra
1	*Task* 	*Opaque Action* 	Uma *task* descreve uma atividade que provê uma saída útil que pode ser consumida por participantes de um processo. Pode ser associada a uma *action* da UML.
2	*Inclusive Gateway* 	*Join Node*	Um *inclusive gateway* sincroniza dois ou mais fluxos. A semântica de *join node* permite mapear ele em *inclusive gateway* quando este possui o papel de sincronizar os fluxos do processo.
3	*End Event vazio*	*Flow Final Node*	Um *end event* indica o final do processo. Caso não venha acompanho por nenhum tipo de evento, ele pode ser mapeado para um *flow final node*.
4	*Start Event* 	*Initial Node* 	Um *start event* indica o início do processo. Caso não venha acompanhando por nenhum tipo de evento, ele pode ser mapeado em um *initial node*.
5	*Message* 	*Control Flow* 	O ponto de início de cada mensagem BPMN deve ser a extremidade inicial do canal de dados entre dois participantes. Este é o significado de *control flow* em UML.
6	*Participant* 	*Activity Partition* 	Um *participant* em BPMN representa uma entidade de negócio ou um participante de um processo. Ela também pode ser estruturada de forma a criar uma hierarquia de participantes no processo. Logo pode-se mapear a *pool* para um *activity partition*.
7	*Data Object* 	*Data Store* 	*Data objects* provê as informações sobre o que as atividades precisam para serem realizadas e/ou o que elas produzem. A semântica de *data store* em UML pode ser mapeada para *data object* em BPMN.

através de um DA. O DA pode, então, ser enriquecido, pelo desenvolvedor, com detalhes das tarefas de sistema que não puderam ser modeladas com o BPMN.

As regras de mapeamento utilizadas aqui foram especificadas observando o comportamento dos elementos dos modelos BPMN e UML, seus significados semânticos e as regras de transformação ATL codificadas em [8]. Como estas regras são baseadas na versão BPMN 1.0, elas foram modificadas neste trabalho para se tornarem compatíveis com a especificação mais recente da BPMN 2.0.

A Tabela 1 mostra as regras de mapeamento entre os elementos da BPMN e do Diagrama de Atividades. São ao todo 24 regras de mapeamento. Porém, por questão de espaço, apenas um subconjunto dessas regras, contendo os elementos do diagrama BPMN mais utilizados e usados neste texto, são apresentadas na Tabela 1. A primeira coluna mostra o elemento BPMN, a segunda o elemento do diagrama de atividades no qual ele foi mapeado e a terceira mostra a descrição do mapeamento. Por exemplo a regra 1, mostra a descrição da transformação de uma *Task*, da BPMN, para uma *OpaqueAction*, da UML.

3.2 Fase II: DA2SOAML - Transformação do Diagrama de Atividades UML para Modelo de Serviços em SoaML

O objetivo desta fase é a identificação de serviços a partir do DA e tem como produto o modelo SoaML. As heurísticas utilizadas nesta seção foram adaptadas das heurísticas encontradas nos trabalhos de [1, 2, 3]. As heurísticas desses trabalhos foram originalmente escritas para identificar serviços diretamente de modelos de processos de negócio em BPMN. Elas foram então adaptadas para identificar serviços a partir de um Diagrama de Atividades. Sendo assim, foi substituído o elemento BPMN da heurística original pelos elementos equivalentes no modelo de atividades. Para realizar as adaptações foram utilizados alguns critérios durante o mapeamento dos elementos de forma a preservar a semântica das heurísticas originais:

- Seja {A, B, ...} um elemento BPMN e C um elemento do Diagrama de Atividades, obtido por transformação a partir de {A, B, ...}, então toda heurística aplicável a {A, B, ...} é também aplicável a C;

Como base nos critérios acima, foram definidas sete novas heurísticas baseadas nas sete heurísticas originais de [1, 2, 3].

Heurística 1: Um serviço candidato deve ser identificado a partir de um conjunto de atividades sequenciais.

Heurística 2: Um serviço candidato deve ser identificado a partir de uma estrutura iniciada em um ponto no *workflow* onde um fluxo de controle simples divide-se em fluxos de controle múltiplos, que podem ser executados em paralelo, e finalizada em um ponto no *workflow* onde os múltiplos fluxos paralelos convergem em um fluxo de controle simples, sincronizando-os, ou onde ramificações terminem em nó final.

Heurística 3: Um serviço candidato deve ser identificado a partir de uma estrutura do *workflow* onde uma ou mais atividades podem ser executadas repetidamente.

Heurística 4: Um serviço candidato deverá ser identificado a partir de uma atividade que recebe uma mensagem enviada por uma atividade de outro *workflow*.

Heurística 5: Um elemento ***Model*** do modelo SoaML deve ser gerado a partir de um elemento ***Package*** do diagrama de atividades da UML.

Heurística 6: Para cada elemento ***ActivityPartition*** do diagrama de atividades deve ser gerado um elemento ***Participant*** do modelo SoaML. Estes ***Participant*** serão utilizados para descrever a arquitetura interna de cada um dos participantes envolvidos no ***ServicesArchitecture***.

Heurística 7: Um elemento ***ServicePoint*** do modelo SoaML deve ser gerado a partir de um elemento ***MessageFlow*** da UML.

Com base nessas heurísticas e através da análise da semântica da descrição dos elementos de ambos os diagramas, UML e SoaML, foram definidas seis regras que mapeiam os elementos do diagrama de atividades em elementos do modelo SoaML.

a) Regra de Mapeamento 1: *Opaque Action* para *UML Action*

   Uma *opaque action* descreve uma ação simples que não pode ser decomposta em outras atividades. Pode ser diretamente associada a uma *action* do modelo SoaML, uma vez que ela provê a interface abstrata para o trabalho finalizado e, ao mesmo tempo, não fornece muitos detalhes do *workflow* da qual faz parte.

b) Regra de Mapeamento 2: *Expansion Region* para *Services Architecture*

   Apesar de agrupar um conjunto de *actions*, uma *expansion region* pode ser vista como uma única atividade, por exemplo quando se deseja ter uma visão geral do fluxo de atividades. Ela pode ser associada a elementos de baixo nível, por exemplo, *services architecture* no nível de participantes, que detalham os papéis e as tarefas de uma região.

c) Regra de Mapeamento 3: *Activity Partition* para *Participant*

   Uma *activity partition* em UML representa uma entidade de negócio em um modelo de negócio. Ela também pode ser estruturada de forma a criar uma hierarquia de participantes no processo. Logo, pode-se mapear a *activity partition* para um *role* em uma *services architecture* ao nível de comunidade onde o tipo de *participant* corresponde a *activity partition*. O *participant* resultante deve aderir à arquitetura de serviços correspondente à partição à qual pertence.

d) *Control Flow "begin"* (início) para *Service*

   O ponto de início de cada controle de fluxo UML tem a seguinte semântica: deve ser a extremidade inicial do canal de dados entre duas atividades. Este é o significado exato de uma service port em SoaML.

e) Regra de Mapeamento 5: *Control Flow* "fim" para *Request*

   O ponto de termino de cada controle de fluxo em UML tem uma semântica muito parecida com a anterior, mas é situada do outro lado do canal de comunicação. A semântica de uma *request port* em SoaML permite construir um mapeamento entre ela e um *control flow end* em UML.

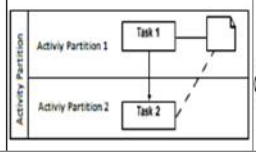
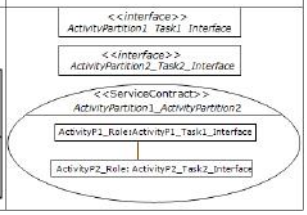
f) Regra de Mapeamento 6: Fragmento do Processo (Padrão) para *Service Contract*

Não existe, no diagrama de atividades, um elemento que apresente comportamento semântico semelhante a um *service contract*. É preciso analisar os diagramas de atividades e identificar os fragmentos que podem ser mapeados para *service contract*.

Um *service contract* define a especificação do serviço que define os papéis que cada participante desempenha no serviço, e as interfaces eles implementam para desempenhar esse papel no serviço. Podemos, no entanto, definir um padrão no diagrama de atividades que possa ser mapeado para um contrato de serviço.

O padrão (*activity partition → activity partition*) descreve uma sequência de tarefas ligadas por um controle de fluxo, mas os participantes são representados através de subpartições diferentes na mesma *activity partition*. As duas tarefas que pertencem ao mesmo contrato de serviço também compartilham um *data store*. A Tabela 2 ilustra o mapeamento da notação.

Tabela 2. Mapeamento de Fragmento do processo (Padrão) para service contract

	UML	SoaML
Elemento	*Activity partition → activity partition*	*Service contract*
Notação		

4. AVALIAÇÃO DA ABORDAGEM

Com intuito de avaliar a proposta, esta foi aplicada a exemplos encontrados na literatura [2]. Para facilitar o entendimento, este texto utiliza apenas uma versão simplificada do exemplo que trata de um processo seguido pela secretária de um Hospital Local para acessar o histórico médico do paciente, no Registro Central de Saúde, antes da cirurgia [Figura 1].

Neste cenário, procurou-se verificar se: (i) as heurísticas, as regras de mapeamento e se as transformações implementadas geravam os elementos esperados em cada modelo de saída, (ii) as alterações no diagrama de atividades geram elementos adicionais no SoaML e (iii) esses elementos têm potencial para gerar serviços mais completos, ou seja, com as portas de acesso expostas e tipos de dados de entrada e saída.

As seções a seguir detalham as etapas do processo descrito anteriormente e o exemplifica.

4.1 Etapa Transformação BPMN2DA – BPMN para Diagrama de Atividades

Esta etapa consiste em executar a transformação de BPMN para DA, usando a ferramenta Eclipse. Esta etapa recebeu o modelo BPMN, um conjunto de regras em ATL, bem como os metamodelos BPMN 2.0 e UML 2.4.4.

As piscinas **Hospital Público e Registro Central de Saúde** foram mapeadas para partições de mesmo nome, utilizando a Regra 6 da Tabela 1. A implementação desta regra (*PoolBPMN2PartitionUML*) pode ser vista na Figura 2.

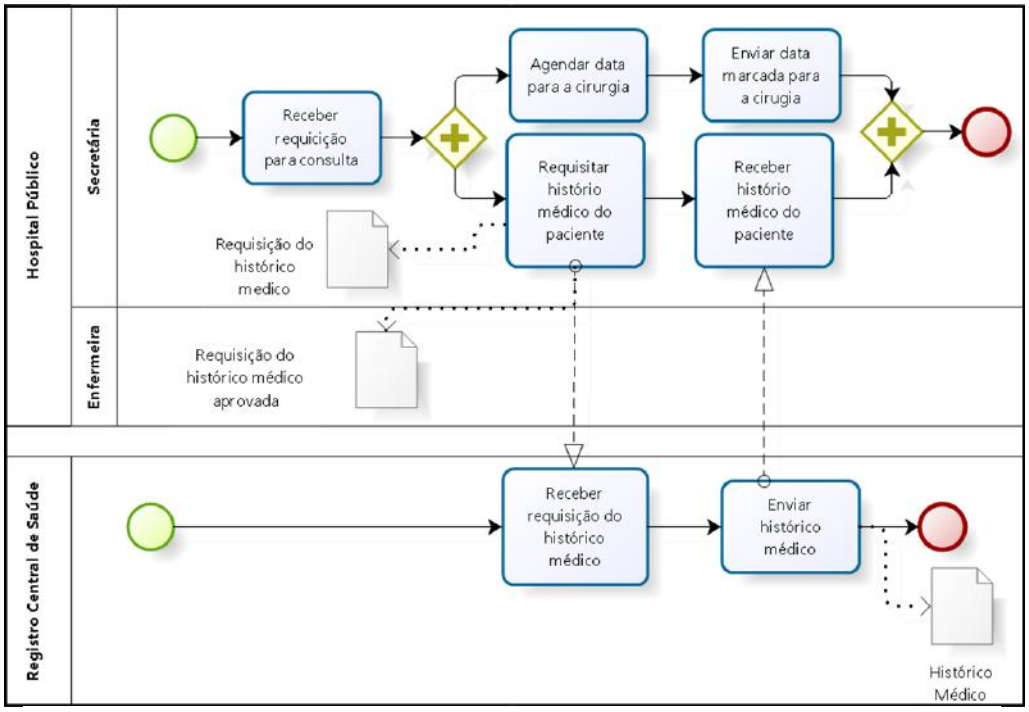


Figura 1. Processo Hospital Local

```
rule PoolBPMN2PartitionUML {
    from
        PoolBPMN: MetaModeloBPMN!Pool
    to
        PartitionUML: MetaModeloUML!ActivityPartition (
            name <- PoolBPMN.name,
            subpartition <- PoolBPMN.lanes
        )
}
```

Figura 2. Implementação da Pool2Partition em ATL

As raias, **Enfermeira e Secretária** foram mapeadas para subpartições de mesmo nome dentro da partição que foi originada pela piscina, na qual as raias fazem parte, utilizando a regra LaneBPMN2PartitionUML (Figura 3).

```
rule LaneBPM2PartitionUML {
    from
        LaneBPMN: MetaModeloBPMN!Lane
    to
        PartitionUML: MetaModeloUML!ActivityPartition (
            name <- LaneBPMN.name,
            superPartition <- LaneBPMN.pool
        )
}
```

Figura 3. Implementação da regra Lane2Partition em ATL

As tarefas de Secretária, Enfermeira no diagrama BPMN foram mapeadas para *actions*, no diagrama de atividades, utilizando a regra de mapeamento 1 da Tabela 1, dentro da partição que foi originada a partir das raias Secretária e Enfermeira, respectivamente. Desta forma é possível notar que a transformação BPMN para UML preserva as relações existentes entre os elementos em ambos os modelos. A Figura 4 mostra o diagrama de atividades completo resultante da primeira transformação.

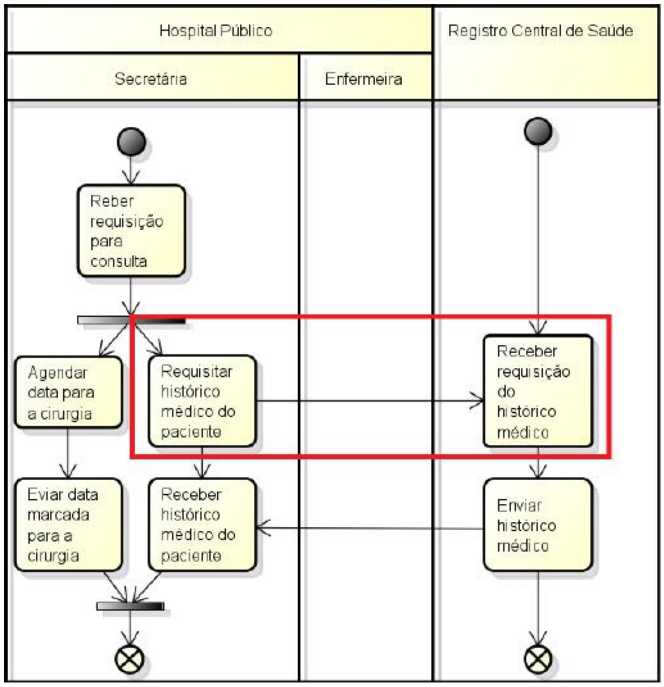


Figura 4. Diagrama de atividades gerado

4.2 Etapa Analisar o Diagrama de Atividades Gerado

Esta etapa tem como objetivo analisar o diagrama, obtido na transformação BPMN2DA, procurar pontos de melhoria ou fazer inclusões no mesmo. O ponto forte do DA é justamente a sua flexibilidade, que permite que seja adicionado informações em um nível de detalhe que antes, com o BPMN, não era possível.

Observe na Figura 4, por exemplo, o par "Requisitar histórico médico do paciente – Receber requisição do histórico médico".

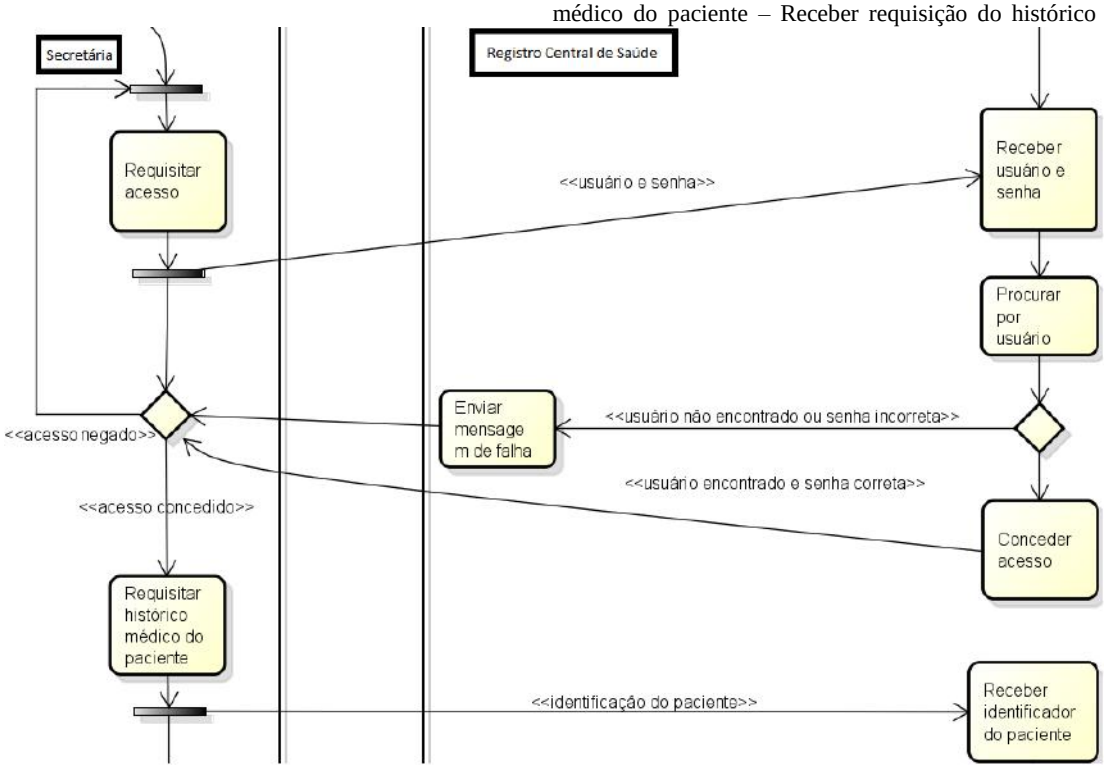


Figura 5. Diagrama de atividades detalhado

Nesse caso não é possível saber quantos serviços são utilizados pelo Hospital Público para que a atividade de negócio "Requisitar histórico médico de paciente" seja concluída. O mesmo se aplica para o Registro Central de Saúde, o qual, poderá precisar trocar mais de uma mensagem com o Hospital Público para concluir a tarefa de negócio "Receber requisição do histórico médico". Desta forma, está modelagem retrata bem a colaboração ao nível de negócio, mas não é suficiente para retratar a troca de mensagens dos serviços ou mesmo a totalidade dos serviços envolvidos no nível do sistema. Para tanto, um conhecimento especializado se faz necessário. Desta forma, é possível expandir este relacionamento de forma a explicitar as trocas de mensagens e serviços acessados ao nível do sistema. Isto pode ser visualizado na Figura 5.

Como é possível observar na Figura 5, quanto mais se detalham as tarefas de negócios ao nível do sistema mais troca de mensagens entre Hospital Público e o Registro Central de Saúde é possível explicitar e consequentemente aumenta-se o número potencial de serviços que podem ser identificados. Também é possível com essa abordagem especificar o tipo de dado que acompanha cada mensagem.

4.3 Transformação DA2SOAML – Diagrama de Atividades para SoaML

Esta etapa consiste em executar a transformação de DA para SoaML. A Figura 6 mostra a arquitetura de serviços, em SoaML, resultante da aplicação das regras de mapeamento entre DA e SoaML descritas anteriormente. As partições e subpartições do modelo UML foram mapeadas para uma arquitetura de serviços e para participantes, respectivamente. Cada participante gerou uma *property* dentro da arquitetura de serviços que foi originada pela partição Solicitação de Material Interno, o que está de acordo com a regra de mapeamento 3, da Sessão III item D.

O fluxo de mensagens entre as partições Secretária e Registro Central de Saúde foram mapeados para *Services* e para *Requests*, isto está de acordo com as regras de mapeamento 4 e 5, respectivamente, da Seção III item D. A relação "Requisitar Acesso → Receber usuário e senha", foi mapeada para um contrato de serviço e encontra-se de acordo com a regra de mapeamento 6, da Seção III item D.

Finalizada a aplicação da abordagem proposta no modelo de negócio para o cenário do hospital é possível afirmar que esta é capaz de gerar um modelo SoaML com elementos relevantes que não poderiam ser gerados usando apenas o diagrama BPMN. É ainda possível observar que as interfaces de serviços RequisitarAcesso e ConcederAcesso não poderiam ter sido geradas usando apenas o diagrama BPMN. Sendo assim, a introdução do DA mostra-se como um candidato para melhorar a completude dos serviços derivados de diagramas BPMN. De fato, com a flexibilidade permitida pelo DA, poderíamos detalha-lo ainda mais com funcionalidades do sistema e com o intuito de expor as interfaces do serviços, parâmetros e tipos de dados.

Sendo assim em relação a (i) fica claro que a transformação gera os elementos mapeados esperados e que mesma preserva a semântica geral do processo. Também é possível observar que em relação a (ii) as modificações realizadas no DA geram mais elementos para o mesmo trecho do BPMN do que é gerado em [2]. Ainda em relação à (ii) é perceptível que as modificações feitas no DA têm potencial para expor não só as portas dos serviços como também os tipos de dados que são esperados em cada porta. E em relação a (iii), de fato, comparado com a abordagem encontrada em [2], para o mesmo par de tarefas do BPMN, o número de colaborações geradas, para esse nível de detalhamento, foi o duas vezes maior.

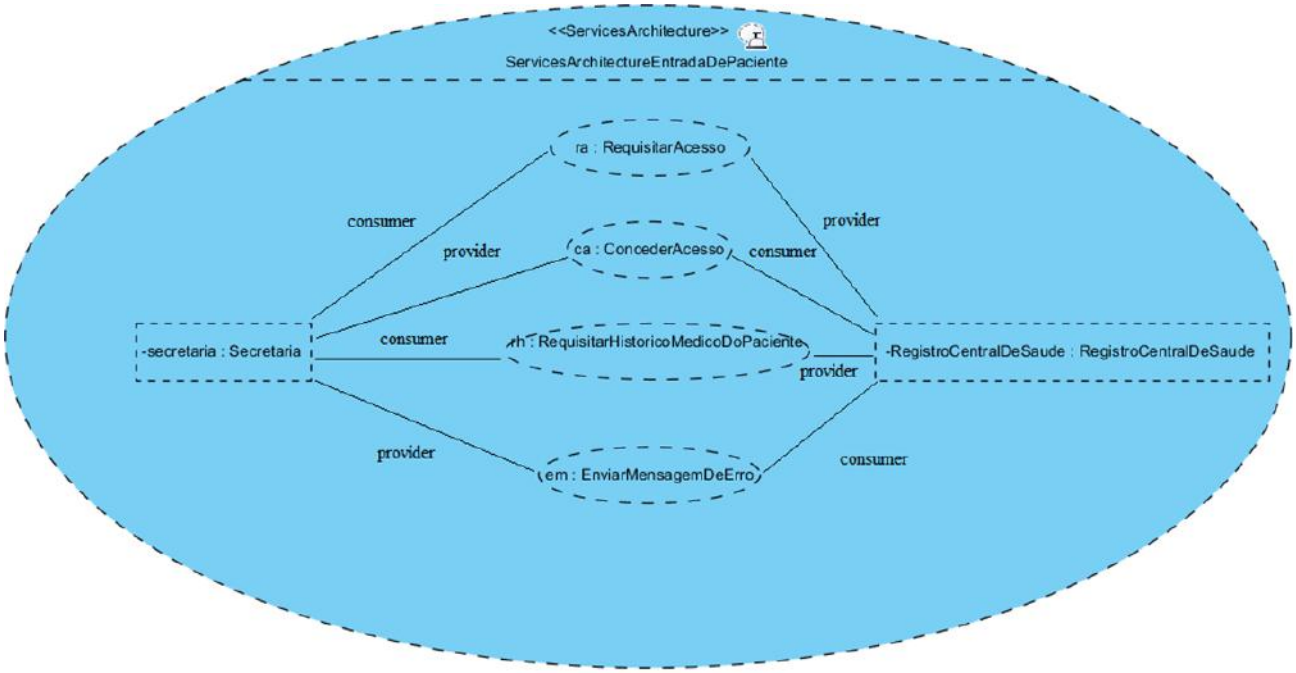


Figura 6. Diagrama SoaML transformado

5. CONCLUSÃO

Este trabalho apresentou uma abordagem para derivação de serviços, expressos em SoaML, a partir de modelos de processos de negócio, especificados em BPMN, utilizando o diagrama de atividades da UML como um diagrama que representa a visão através da ótica de um sistema de informação. Heurísticas são utilizadas para codificar as transformações BPMN2DA e DA2SOAML, em ATL, que apoiam a automação da abordagem proposta.

Esta abordagem é composta de duas etapas e apresenta uma alternativa para melhorar a completude dos serviços especificados introduzindo o DA. A introdução deste oferece uma flexibilidade que não é encontrada em outras propostas, uma vez que ele permite ao especialista detalhar particularidades do sistema que não são possíveis expressar em um diagrama BPMN.

Sendo assim., a derivação proposta por essa abordagem tem potencial para gerar uma arquitetura de serviços detalhada capaz de gerar código. A abordagem foi aplicada em alguns modelos de processos e foi possível perceber que o uso de um conjunto de heurísticas, baseadas no comportamento dos elementos do DA, ao invés de usar heurísticas que foram originalmente desenhadas para BPMN, tem potencial gerar especificações de serviços mais detalhados, com elementos não encontrados em outras propostas.

Embora seja útil para demostrar a viabilidade das propostas, o cenário exemplo não é suficiente para realizar a avaliação completa, sendo necessário, a execução de um estudo mais aprofundado através de experimentos e estudos de caso para uma avaliação mais rigorosa.

Como trabalhos futuros ao estudo desenvolvido espera-se implementar em uma ferramenta que apoie todo o processo de derivação de serviços a partir de requisitos de negócio utilizando-se da abordagem DDM, e adição de estereótipos que ajudem a melhorar o nível de automação regras semiautomáticas.

6. REFERÊNCIAS

[1] AZEVEDO, L. G. et al. Identificação automática de serviços candidatos a partir de modelos de processos de negócio. In: Escola Regional de Sistemas de Informação. 2009.

[2] DELGADO, A. et al. From BPMN business process models to SoaML service models: A transformation-driven approach. In: Software Technology and Engineering (ICSTE), 2010 2nd International Conference on. 2010. p. V1-314-V1-319.

[3] ELVESAETER, B. et al. Aligning business and it models in service-oriented architectures using bpmn and soaml. In: Proceedings of the First International Workshop on Model-Driven Interoperability., 2010. P 61-68

[4] FAZZIKI, A. et al. A service oriented information system: A model driven approach. In: Signal Image Technology and Internet Based Systems (SITIS), 2012 Eighth International Conference on. [S.l.: s.n.], 2012

[5] GOMES, R. et. al. MoDErNE: A model driven process centered software engineering environment. In: Proceedings of CBSoft 2011—II Brazilian Conference on Software: Theory and Practice, Tools Session 2011, São Paulo, Brazil (2011).

[6] HEREDIA, L. R. Transformação de modelos de processos de negócio em BPMN para modelos de sistema utilizando casos de uso da UML. Dissertação (Mestrado) – Pontifícia Universidade Católica do Rio Grande do Sul, 2012.

[7] SADOVYKH, A. et al. Enterprise architecture modeling with soaml using bmm and bpmn - mda approach in practice. In: Software Engineering Conference (CEE-SECR), 2010 6th Central and Eastern European. [S.l.: s.n.], 2010. p. 79–85.

[8] BASTOS, E. C.; FONSECA, V. S.; "Transformação de modelos BPM para Diagramas de Atividades da UML 2.0 usando ATL". Capturado em: http://code.google.com/p/transformacao-bpm, 2013.

[9] ODEH, M.; KAMM, R. "Bridging the gap between business models and system models". Information and Software Technology, vol. 45-15, 2003, pp. 1053-1060.

[10] BRAGA, V. T. Um Processo para Projeto Arquitetural de Software Dirigido a Modelos e Orientado a Serviços. Dissertação (Mestrado) — Universidade Federal de Pernambuco, 2011.

[11] ERL, T. SOA Design Patterns. [S.l.]: Pearson, 2009.

[12] Model Driven Architecture (MDA), OMG, 2003.

[13] Business Process Modeling Notation (BPMN), OMG, 2011.

[14] Soa Modeling Language (SoaML), OMG, 2012.

[15] Unified Modeling Language (UML), OMG, 2008.