Uma Arquitetura para Integrar Processos de Negócios e Situações Contextuais por Meio de Regras de Negócio

Luis Augusto Melo Rohten
Universidade Federal do Espírito Santo - UFES
luis.rohten@aluno.ufes.br

José Gonçalves Pereira Filho
Universidade Federal do Espírito Santo - UFES
zegonc@inf.ufes.br

ABSTRACT

In recent years, the concept of situational-awareness has been explored by the information systems areas in order to analyze context-sensitive data. This approach seeks to detect a particular state of interest in reality, that is, a contextual situation and react accordingly, helping to improve adaptability to changing domain conditions. In this paper, we describe a conceptual architecture focused on the representation of *contextual situations* through *business rules* and their integration with *business processes*. As a proof of concept, an application for real-time monitoring of medical devices was implemented. This work has shown that bringing situational awareness to information systems can reduce the complexity of business processes as well as increase the reliability of the domain representation model.

KEYWORDS

Processos de Negócio, Regras de Negócio, Detecção de Situações, Sensibilidade a Situações

1 INTRODUÇÃO

Nas últimas duas décadas, as regras de negócios e a modelagem de processos de negócios (*BP – Business Process*) se tornaram um tópico de pesquisa importante, principalmente nas áreas de Sistemas de Informação e Ciência da Computação [1, 2]. Essa importância se dá, devido a complexidade de separar as regras de negócios, que representam a lógica do negócio, dos processos de negócio e aplicativos operacionais.

Os dois componentes principais nas arquiteturas de sistemas de informações atuais em termos de modelagem de requisitos de negócios são processos de negócios e regras de negócios. Embora a necessidade de ambos serem modelados de forma integrada seja bem estabelecida, o conjunto de conhecimento sobre a modelagem integrada dos dois é limitado [2]. As regras de negócios são declarações usadas para definir ou restringir alguma ação nos processos de negócio. Elas guiam comportamentos, descrevendo como determinadas operações devem ser realizadas e se há algum limite que precisa ser aplicado. Elas permitem a especificação do conhecimento comercial de uma maneira que seja compreensível pelo "negócio", mas também executável pelos mecanismos de regras, contemplando assim, a lacuna entre os negócios e a tecnologia [3, 4]. Devido a sua relevância, foram desenvolvidos *Business Rule Management Systems (BRMS)* com a finalidade de gerenciar as regras em sistemas baseados em processos de negócio [5].

Diferentemente de eventos, que se iniciam e terminam num dado instante e são detectados somente quando ocorrem – e disparam ações em um sistema de gerenciamento de regras tradicional - uma situação pode ser detectada no primeiro instante em que ela ocorre, sem necessariamente saber quando ou se ela terminará. Os eventos não suportam o conceito de "*acontecendo*", que é parte intrínseca do conceito de situação. Quando ações são derivadas do uso de situações, processos e usuários podem tomar decisões baseadas num monitoramento contínuo da realidade observada. Portanto, incorporar uma plataforma de gerenciamento de situações ao ambiente de processos de negócio é acrescentar uma camada de inteligência no topo dos atuais sistemas corporativos baseados em regras.

Com isso, mecanismos de detecção de situações, conhecidos como conceito de *situation-awareness* vêm sendo desenvolvidos para suportar a definição de situações a partir de regras de negócio [6], tendo em vista sua grande importância e aplicabilidade dos inúmeros domínios a serem explorados dentro das organizações. Estes sistemas são uma extensão dos sistemas de regras de negócio tradicionais baseados no paradigma ECA (Evento-Condição-Ação), uma situação é uma abstração de eventos do mundo real derivado de contextos relevantes e os relacionamentos entre eles, e hipóteses sobre como um particular estado da realidade se relaciona com os interesses dos *stakeholders* (partes interessadas) e aplicações.

Esses sistemas são denominados sensíveis a situação (*SA - Situation-Awareness*) porque tentam, quase em tempo real, perceber e compreender uma situação de algum tipo (por exemplo, alterações em condições do domínio) e projetar uma reação à situação detectada [7], permitindo que processos de negócio sejam melhorados continuamente e aderentes a mudanças de contexto, constituindo uma funcionalidade interessante para a necessidade estratégica e uma vantagem competitiva para as organizações. Em outras palavras, uma situação é um estado das coisas da realidade de particular interesse de usuários e aplicações [8, 9].

Este artigo apresenta uma arquitetura conceitual que busca integrar o processamento de situações contextuais ao ambiente de processos de negócios, a qual foca na separação de interesses entre o especialista de processos de negócio – e desenvolvedor de sistemas.

O restante deste artigo está estruturado como segue. A Seção 2 apresenta trabalhos que exploram conceitos tratados neste artigo e que inspiraram algumas das escolhas arquiteturais e tecnológicas aqui adotadas. A Seção 3 introduz a arquitetura proposta, apresentando os seus componentes e discutindo alguns aspectos de implementação. A Seção 4 apresenta um estudo de caso e a Seção 5 conclui o artigo, apontando perspectivas futuras de pesquisas.

2 TRABALHOS RELACIONADOS

Um mapeamento sistemático foi conduzido com o objetivo de investigar trabalhos que propõem a integração entre sensibilidade ao contexto e processos de negócio. Este mapeamento será descrito de forma resumida e demonstrará que, embora pesquisas estejam sendo conduzidas em direção à integração de processos e regras de negócio, como pode ser visto em [4, 7, 9–13], poucas contam com

uma infraestrutura para a definição de situações contextuais em *design-time* e seu monitoramento em *runtime*.

Um mapeamento através das principais engines de busca foi realizado e trouxe uma quantidade significante de de trabalhos relacionados a temas que envolviam situações contextuais e processos de negócio. A Tab. 1 mostra os resultados no mapeamento e as engines utilizadas.

Tabela 1: Etapa 1 do mapeamento sistemático

Engines	Trabalhos
IEEE	32
Scopus	42
ACM	3
Compendex	10
Web of Science	10
Science Direct	43
Total:	140

Baseando-se nos estudos realizados por [14], 4 etapas foram realizadas com o objetivo de refinamento, a etapa 1 - consiste na etapa de pesquisa; etapa 2 - leitura dos abstracts para encontrar trabalhos relacionados com os temas de interesse; etapa 3 - leitura superficial nos artigos resultantes da etapa anterior; 4 - leitura completa dos trabalhos restantes.

A etapa 1 obteve 140 trabalhos relacionados aos temas de interesse; após a leitura dos abtracts na etapa 2, apenas 44 trabalhos se mostraram relevantes e foram encaminhados para a etapa seguinte; a etapa 3 resultou em 14 artigos relacionados ao tema de interesse; por fim, na etapa 4 foram elencados requisitos relevantes pertinentes a arquiteturas conceituais que abrangem conceitos de situações contextuais e processos de negócio. Com isso, uma discussão dos principais trabalhos analisados é realizado a seguir.

Em (ABBASI;SHAIKH;20019) [15] é descrito uma tecnologia de *workflow* como um meio eficaz de modelagem e gerenciamento de processos de negócios complexos que foi empregada em outros diversos domínios. Embora a tecnologia de fluxo de trabalho seja bastante madura e os produtos comerciais de gerenciamento de *workflow* estejam prontamente disponíveis, observou-se que a tecnologia não possui certos componentes e conceitos relacionados ao contexto quando aplicada a domínios como comércio eletrônico, computação móvel e invasora. Os *workflow*s inteligentes ou sensíveis ao contexto são aqueles que operam de acordo com as informações de contexto reunidas a partir do conhecimento do ambiente ou do domínio. Para tornar o gerenciamento do fluxo de trabalho sensível ao contexto, extensões e modificações são necessárias em arquiteturas e padrões existentes de sistemas de gerenciamento de *workflow*.

Os autores (BUCCHIARONE; MEZZINA; PISTORE; 2013) [16] apresentaram arquitetura de adaptabilidade em tempo de execução como uma característica chave de ambientes de negócios dinâmicos, em que os processos precisam ser constantemente aprimorados e reestruturados para lidar com as mudanças de contexto. Os autores apresentam a *CAptLang*, uma linguagem para modelar processos de negócios sensíveis ao contexto e adaptáveis onde a principal característica é a possibilidade de deixar o manuseio de situações extraordinárias ou improváveis para o tempo de execução. É apresentada a sintaxe e semântica formal da linguagem, mostrando como sua semântica foi usada para guiar a implementação de um mecanismo de execução de processos de negócios baseado em Java, componente de adaptação *ASTRO-CAptEvo framework*.

No trabalho [4] de (VASILECAS; KALIBATIENE; LAVBIC; 2016) os autores descrevem que as abordagens tradicionais usadas para implementar processos de negócios nos sistemas de informação atuais não abrangem as necessidades reais dos negócios que mudam dinamicamente. Surgindo a necessidade de uma nova abordagem de modelagem e simulação de processos de negócios dinâmicos (Business Process Diagram - BPD). Segundo os autores até o momento as abordagens existentes para modelagem e simulação de BPD foram incompletas, ou seja, não possuem teoria ou estudo de caso ou ambos. Além disso, não existe uma definição comumente aceita de BPD. As atuais ferramentas de modelagem de BP são adequadas quase exclusivamente para a modelagem e simulação de uma BP estática que prescreve estritamente quais atividades e em qual sequência executar. Geralmente, um BPD não é definido estritamente no início de sua execução e é alterado sob novas condições no tempo de execução. O trabalho propôs seis requisitos do BPD e uma abordagem para modelagem e simulação de BPD baseada em regras e contexto. A abordagem é baseada na mudança de regras BP, ações BP e suas sequências no tempo de execução da instância de processo, de acordo com o novo contexto do sistema de negócios. Com base na abordagem proposta, foi desenvolvida uma arquitetura de referência, e um protótipo de uma ferramenta de simulação de BPD. A modelagem e a simulação foram realizadas usando esse protótipo, e o estudo de caso mostra a correspondência com as necessidades de mudanças dinâmicas nos negócios, bem como as possibilidades de modelagem e simulação da BPD.

O trabalho de (MOREIRA et al. 2015a) [17] destaca que inúmeras aplicações de TIC com mecanismos para detectar situações foram desenvolvidas para apoiar o gerenciamento de desastres (GD) nos últimos anos, sendo um campo de grande importância econômica e social. Essas aplicações situation-awareness (SA) tentam perceber e compreender, quase em tempo real, algum tipo de situação de interesse possibilitando projetar uma reação a situação detectada, por exemplo: epidemias de doenças, pessoas doentes isoladas etc. O autor descreve que, um obstáculo para a modelagem de aplicações de SA é a falta de construções estruturais e temporais bem fundamentadas, o que é inerente às técnicas convencionais de projeto.

Em seu outro trabalho (MOREIRA et al. 2015b) [11] descreve um framework com o objetivo de melhorar a interoperabilidade e a produtividade no desenvolvimento de aplicativos sensíveis a situações para o gerenciamento de desastres, sendo necessários mecanismos e diretrizes apropriados, abordando a falta de semântica na modelagem de situações de emergência. Além disso, a natureza sempre mutável e imprevisível dos cenários de desastres apresenta desafios para o processamento e a colaboração de informações. O *framework* combinar os seguintes elementos: (i) Uma ontologia fundacional para a conceitualização temporal; (ii) Especificações bem fundamentadas de modelos estruturais e comportamentais; (iii) Um mecanismo *Complex Event Processing* (CEP) baseado em uma plataforma distribuída baseada em regras para o gerenciamento de situações; (iv) Uma abordagem baseada em modelos.

É mostrado em (BAJEC; KRISPER; 2005) [18] que as regras de negócios são evidentemente importantes para as organizações, pois descrevem como estão fazendo negócios. Seu valor também foi reconhecido dentro do domínio do sistema de informação (SI), principalmente por causa de sua capacidade de tornar os aplicativos flexíveis e passíveis de alterações. O trabalho propõe uma metodologia que ajuda os empresários e desenvolvedores a manter as regras de negócios no nível comercial alinhadas com as regras implementadas no nível do sistema. Em contraste com várias abordagens existentes que se concentram principalmente nas regras de negócios no escopo de um aplicativo, a metodologia proposta aborda todo o SI de uma organização. Descreve também os requisitos para um suporte de ferramenta que seria apropriado para apoiar a metodologia. Em ambos os trabalhos o autor engloba conceitos de integração entre *situation-awareness* e *behavior model* (BPMN) para a descrição de estruturas e comportamentais.

A Tab. 2 traz uma amostra de trabalhos que buscam integrar sensibilidade ao contexto e BPM. Para efeitos de avaliação, os seguintes requisitos foram observados: Sensibilidade a Contexto com Processos de Negócio (R1); Suporte a Situações (R2); Definição de Workflows (R3); Adaptabilidade em Tempo de Execução (R4); parte destes requisitos foram usados como base para a definição da arquitetura conceitual proposta neste artigo. O símbolo "+" significa que o trabalho atende completamente ao requisito avaliado, o "–" indica que o trabalho atende apenas parcialmente, enquanto que o símbolo "•" representa o não atendimento ao requisito.

Tabela 2: Comparação de trabalhos que integram sensibilidade ao contexto e BPM

Trabalhos	R1	R2	R3	R4
(ABBASI;SHAIKH;2009)	+	•	+	•
(BUCCHIARONE;MEZZINA;PISTORE2013)	+	•	•	-
(VASILECAS;KALIBATIENE;LAVBIC; 2016)	+	•	+	+
(MOREIRA et al.,2015)	+	+	+	-
(BAJEC; KRISPER;2005)	-	•	+	-

Conforme visto na Tab. 2, a maioria dos trabalhos analisados não atendem completamente aos requisitos R2 e R4, mostrando que existe uma carência de soluções integradas para o desenvolvimento de aplicações em ambientes BPM que suportem análise de contexto e situações, adaptabilidade de processos e regras em tempo de execução, e que, adicionalmente, promovam a separação de papéis.

O objetivo deste trabalho é apresentar uma arquitetura conceitual que una conceitos de análise de situações contextuais e processos de negócio com foco em facilitar a resolução de problemas organizacionais que dependem de análise de dados sensíveis.

3 ARQUITETURA PROPOSTA

A solução proposta adota como princípio geral de modelagem o uso de uma abordagem que privilegia a 'separação de papéis' ou 'separação de preocupações' ("*separation of concerns*") [19]. A separação de papéis é amplamente conhecida na comunidade de Engenharia de Software, sendo uma estratégia muito usada quando se trata de lidar com um ambiente com problemas a serem trabalhados em diferentes níveis de abstração. Isso torna possível o desacoplamento das camadas de soluções, evitando que os usuários de níveis mais altos precisem se preocupar com os detalhes técnicos de níveis inferiores.

No caso particular da arquitetura proposta, esta separação de papéis se reflete nas seguintes escolhas: (i) uso de uma abordagem de alto nível para a definição de situações, regras e modelos de processos de negócio; e (ii) uso de uma abordagem de baixo nível, orientada a serviços, que opera em tempo de execução e cuja característica é a flexibilidade e promoção do desacoplamento via interfaces padronizadas [20, 21].

No projeto da arquitetura, foram estabelecidos os seguintes requisitos: (R1) Definição de regras e situações em *design-time* (tempo de modelagem); (R2) Definição de modelos de processos de negócio em *design-time*; (R3) Nó de execução de situações, regras e modelos de processos em *runtime* (tempo de execução). Estes requisitos servem de base para alicerciar os principais pontos relevantes pertinentes a separação de responsabilidades e estruturas implementacionais dos projetos derivados da arquitetura proposta. Os níveis de abstração que caracterizam os níveis de *design-time* e *runtime* serão discutidos com mais detalhes na seção seguinte.

3.1 Arquitetura Conceitual

A arquitetura conceitual proposta é baseada nos trabalhos [10, 11, 22], que apresentam arquiteturas conceituais para a representação de situações integradas a modelos de processos de negócio.

A arquitetura é dividida em dois níveis: (i) nível de *design-time* (tempo de modelagem), para especialistas de domínio e desenvolvedores; (ii) nível de *runtime* (tempo de execução), no qual o nível de runtime é separado em aplicações sensíveis a situações;

O nível de *design-time* para os *developers*, por sua vez, abstrai o conhecimento de baixo nível no qual as aplicações estão rodando, e permite ao desenvolvedor de sistemas se preocupar apenas com a definição dos tipos de situações, regras de situações etc. Este nível se estende também aos BPM Experts, que modelam processos de negócios mais fiéis aos interesses dos stakeholders. A definição dos tipos de situação ("*situation types*"), regras de situação ("*situation rules*"), processos de negócio ("*business process*"), fluxos de regra ("*ruleflows*"), dentre outros, são as preocupações nesta etapa.

O nível de *Runtime* de aplicações sensíveis a situação é organizado em oito componentes, cujas funcionalidades se baseiam nas estruturas das plataformas Drools [5, 23, 24], jBPM [10, 25] e Scene [26]. Estes componentes permitem que as estratégias de análise de contexto, sensibilidade a situações e uso de modelos de processos de negócio sejam suportadas pela arquitetura.

A Fig. 1 mostra a arquitetura conceitual proposta. Os componentes assinalados em vermelho foram implementados e serão melhor abordados nos capítulos seguintes. Os nomes dos componentes são descritos Língua Inglesa com o objetivo de manter o padrão adotado por Mica Endsley [27], que é a autora pioneira em pesquisas de *situation-awareness*, e nas estruturas propostas por [10, 11, 22] que se baseiam este trabalho. Uma descrição sucinta dos principais módulos da arquitetura é feita a seguir:

***Situation Type*:** Define os tipos de situações com base nas variáveis do domínio de interesse, no qual que possuem representatividade para o sistema [26].

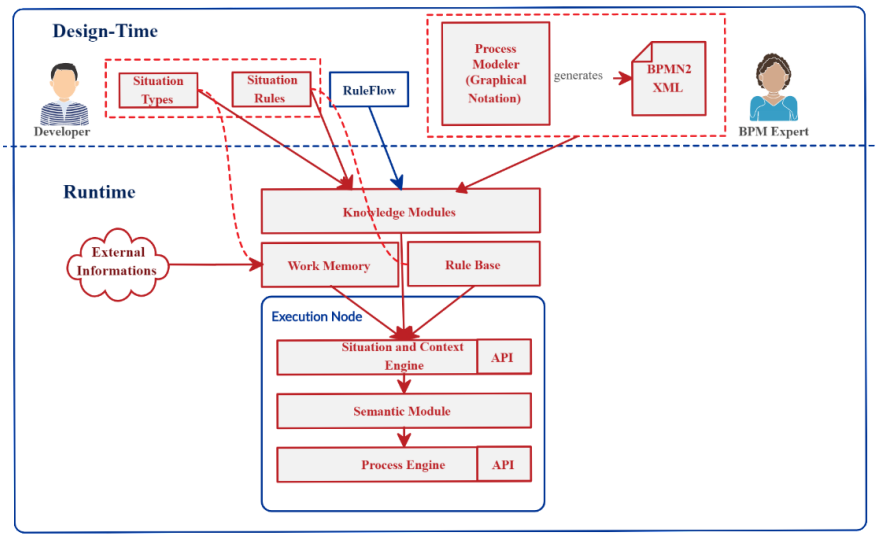

Figura 1: Arquitetura conceitual

***Situation Rule*:** Analisa os dados provenientes do componente *External Information* (Informações Externas) para realizar o casamento de padrões (*pattern match*); após isso, uma ação é disparada, podendo ou não ser um processo de negócio.

***Process Modeler*:** Conhecido também como componente de modelagens de processos de negócio que utiliza abordagem drag-and-drop; permite ao BPM expert gerar processos em formato compreensível aos suites BPM.

***Knowledge Modules*:** É o módulo de conhecimento responsável por prover a compreensão de módulos de alto nível junto ao sistema, esta abordagem visa a integração com o baixo nível da arquitetura. Alguns desses módulos são: *Situation Types*, *Situation Rules*, *RuleFlow*; entretanto, há outros componentes em nível de *design-time* que são suportados, no qual é responsabilidade deste módulo prover um entendimento comum a todo sistema a respeito das funcionalidades inerentes aos módulos.

***Work Memory*:** Também conhecido como memória de trabalho, possui como atribuição armazenar os dados capturados provenientes do domínio de interesse; para que posteriormente sejam testadas as informações da work memory com as regras de negócios, esperando um casamento de padrões no *Execution Node*. A arquitetura denota informações provenientes do contexto como uma nuvem, no qual representa as informações inseridas na *Work Memory* advindas de um meio externo.

***Rule Base*:** Componente responsável por armazenar os *Situation Types* e *Situation Rules*, no qual são definidos no nível de *design-time*. As setas pontilhadas utilizadas na arquitetura conceitual representam a relação direta entre estes dois componentes.

***Execution Node*:** O *Execution Node* (nó de execução) pode ser descrito como uma caixa preta para o alto nível; seus componentes internos serão descritos a seguir:

(i) *Situation and Context Engine e API:* Este módulo junta os dados da *Work Memory* e as regras de negócios definidas na *Rule Base*; este módulo possui um submódulo denominado API que é um conjunto de funções de baixo nível que permite uma interação direta com o(s) gatilho(s) de situações e contexto; a compreensão de módulos de processos, sinais de eventos, gerenciamento de concorrência de situações e/ou contextos, associações a processos de negócio também fazem parte das funções descritas neste módulo.

(ii) *Semantic Module:* Denominado módulo semântico é responsável por realizar o tratamento dos modelos de processos de negócio. Os processos de negócio neste nível sofrem um *parser*, sendo assim, destrinchados em tags que passem significado para o *(*Process Engine) (Motor de Processos); esta tradução permite que a estrutura interna do *(*Process Engine) possa executar o processo.

(iii) *Process Engine e sua API:* A *Process Engine* carrega e executa modelos de processos de negócio; instâncias de processos de negócio são criadas e seu fluxo interno é seguido; o submódulo API esta disponível para a interação com outros componentes externos, interação com classes, serviços internos e externos; este módulo também é capaz de prover uma compreensão a nível de execução dos processos no qual retorna o atual estado do processo e permite o gerenciamento da sua forma de execução, podendo ser síncrono ou assíncrono.

***External Informations*:** São as informações externas, ou seja, dados que possuem representatividade para o domínio. Este módulo captura as informações que fazem sentido a aplicação e permite que o conjunto de dados seja analisado; isso permite que ocorra a ativação de situações caso um particular conjunto de eventos ocorra. Os dados que não são relevantes para a aplicação, podem ser armazenados para análises futuras, ou descartadas.

Este conjunto de componentes representa os principais componentes da arquitetura conceitual proposta, a seguir será apresentado a arquitetura de implementação baseada nos componentes que foram descritos acima.

3.2 Arquitetura de Implementação

A arquitetura de implementação é mostrada na Fig. 2. A abordagem de desenvolvimento adotada, possui como foco a separação de papéis no qual o especialista de processos de negócio inicialmente

constrói os modelos de processos para solucionar os problemas de um dado domínio. O desenvolvedor, por sua vez, abstrai os detalhes da modelagem dos processos, e por meio de um editor de regras e situações define as regras pertinentes ao negócio.

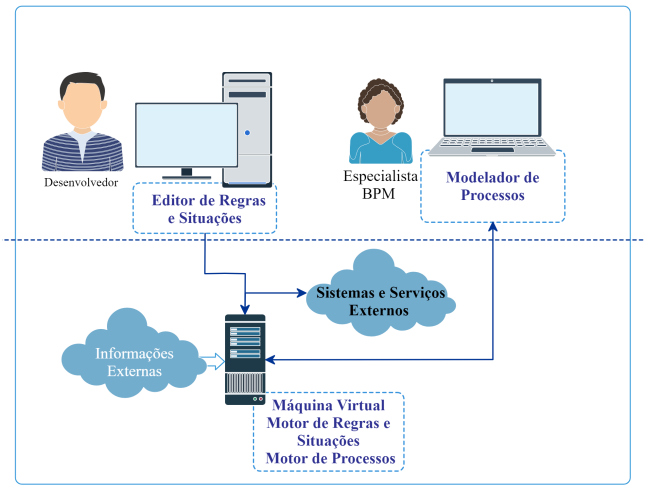

Figura 2: Arquitetura de Implementação

O editor gráfico de modelos de processos de negócio escolhido neste trabalho é o *BPMN2 Diagram Editor*. O requisito básico de uma ferramenta de edição de modelos para uso na arquitetura proposta é que ela seja capaz de salvar modelos de processos de negócio em um formato compreensível por outros editores de modelos e engines de processos de negócio. No editor os modelos são salvos em padrão XML, permitindo importações futuras em outros editores de modelos de BP ou mesmo serem consumidos por engines de processos. A escolha deste editor se mostrou adequada, satisfazendo ao requisito R1 da arquitetura conceitual.

O BRMS - *Drools IDE (Integrated Development Environment for Drools)* [23, 24] foi escolhido para dar suporte a definição das regras e situações. O Drools permite que regras sejam criadas no formato .drl para serem utilizadas posteriormente pela *Situation and Context Engine*. Esta *IDE* possibilita a integração com a API do motor de processos (*Process Engine*); neste trabalho, a ferramenta jBPM possibilita a integração com serviços e aplicações internas e externas. A API do motor de processos permite definir gatilhos de processos dentro das regras e situações no qual os processos podem ser instanciados de forma síncrona ou assíncrona de acordo com o domínio em que está inserido.

O Drools permite que regras de negócios sejam definidas de forma mais natural e declarativa; Promove a criação de aplicações dinâmicas e adaptativas, com regras que frequentemente mudam diante dos cenários do domínio; além disso, possui sua integração com o jBPM é realizada de forma fácil, o que torna a API do motor de processos mais acessível; sendo assim, os pontos destacados levaram a escolha da *Drools IDE*, se mostrando bastante adequada na arquitetura de implementação e satisfazendo ao requisito R2 da arquitetura proposta.

Diante a deficiência do Drools em analisar situações ao longo do tempo, (Pereira; 2013)[26] desenvolveu uma plataforma chamada Scene que permite tal análise. Esta abordagem proposta por Pereira foi incorporada ao Drools para solucionar as questões pertinentes a análise de situações. Como uma plataforma de gerenciamento de situações o Scene suporta o desenvolvimento de aplicações *situation-awareness*, oferecendo design de artefatos para a especificação e gerenciamento do ciclo de vida de situações que inclui a detecção de situações; esta abordagem envolve o reconhecimento de padrões de situação compostas. O Scene aprimora as funcionalidades da *Drools Engine* de forma a suportar nativamente a percepção da situação baseada em regras.

A definição de *Situation Types* e *Situation Rules* são suportados pelo Scene [6] de forma a representar os domínios de aplicação. As características inerentes a cada domínio são especificadas por meio dos *Situation Types*. Os *Situation Rules* descrevem as regras de negócios que devem ser satisfeitas ao longo do tempo, permitindo que uma determinada ação seja tomada. Nesta abordagem, as ações iniciam processos de negócio previamente definidos na aplicação, e os executam assinconamente. Esta abordagem permite que os processos sejam executados paralelamente a aplicação, no qual novos dados sejam coletados e/ou ações sejam tomadas.

Como já antecipado, a *engine workflow* adotada na arquitetura de implementação é o jBPM [10, 25]. O jBPM é um ambiente baseado em Máquina Virtual de Processos (*PVM – Process Virtual Machine*), com vistas a suportar múltiplas linguagens de processos de forma nativa, por exemplo, a notação BPMN 2.0. O jBPM tem como função dar suporte à implementação de modelos de processos de negócio, definição de serviços, integração entre serviços e integração com o Drools, permitindo que regras sejam disparadas mesmo dentro de tarefas definidas em uma instância de processos de negócio.

O jBPM realiza os *parsers* dos modelos de processos de negócios e definidos nos arquivos de regras por meio do componente de módulo semântico; o módulo semântico realiza a separação de todos os componentes descritos no diagrama gráfico que representa o processo, discriminando para o motor de processos todos os elementos ali inseridos, e dando sentido a cada um deles; as funções do motor de processos tornam-se necessárias ao realizar a execução dos processos; esta é capaz de realizar instâncias de processos, persistências de dados de processos e o gerenciamento dos processos.

As tecnologias descritas neste capítulo preenchem o requisito R3 da arquitetura, no qual tangem aspectos de execução de situações, regras e compatibilidade com modelos de processos de negócio em *runtime*; possibilitando que um nó de execução seja definido. As regras e situações deste nó executam modelos de processos de negócio como gatilhos definidos a partir da ocorrência de situações e regras descritas em *design-time*.

4 ESTUDO DE CASO

O cenário escolhido para o estudo de caso é o de uma Clínica de Estética que necessita de um sistema para monitorar o armazenamento do medicamento Botox após a sua chegada à empresa. Este produto, bastante sensível às condições de temperatura ambiente, precisa ser mantido em um compartimento adequado, sob determinado intervalo de temperatura, para que não se deteriore e tenha condições de uso. Após a diluição com soro conforme bula e uso do protocolo adotado pelo médico, o produto deve ser mantido somente

no freezer por um período de no máximo de 3 dias. Neste cenário, uma série de situções que colocam o produto em risco podem ser especificadas e que, na sua ocorrência, requerem ações imediatas, como as notificações aos professionais responsável ou a atuação de algum mecanismo alternativo de controle de temperatura.

A Fig. 3 representa o domínio de interesse analisado.

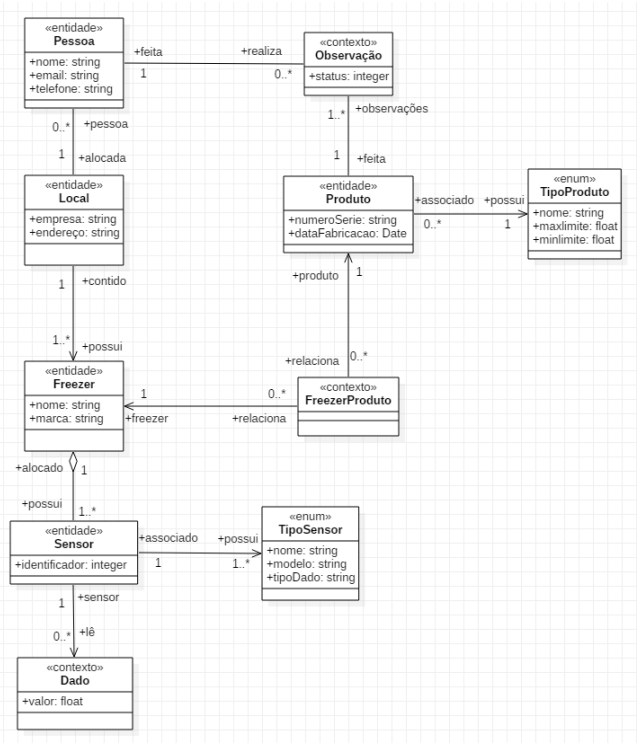

Figura 3: Diagrama UML de representação das entidades e contextos do estudo realizado

O objetivo desta seção é mostrar a visão do Especialista BPM (*BPM Expert*) e a do desenvolvedor de sistemas na criação de uma possível solução para este problema usando para o armazenamento deste medicamento a infraestrutura proposta, tendo em vista, que este domínio possui uma entidade sensível a temperaturas, ou seja, o medicamento e, o local aonde esta entidade é armazenada, interage frequentemente com ações humanas que permitem a elevação de temperatura do local, ou seja, o freezer. Com isso, este estudo de caso foi realizado capturando dados reais de temperatura de uma clínica de estética na região metropolitana de Vitória-ES. Os dados de temperatura foram utilizados para análise de ocorrências de situações de interesse, que são modelados pelos desenvolvedores, com o objetivo de explorarem situações de interesses com base no domínio e, os especialistas BPM modelarem em alto nível os critérios com o qual os *stakeholders* serão notificados, caso hajam elevações de temperaturas.

O domínio é descrito basicamente por pessoas alocadas em uma Clínica Médica, esta clínica possui locais para o armazenamento de medicamentos, estes locais possuem sensores de medição de temperatura e luminosidade. As pessoas alocadas por sua vez, realizam observações de um conjunto de produtos que são armazenados em locais refrigerados.

4.1 Visão do Especialista de Processos de Negócio

Considere um processo de negócio que executa um cenário similar ao descrito anteriormente, é apresentado na Fig. 4. Este processo inicialmente analisa se o freezer está ligado e, após isso, realiza uma leitura do dado de temperatura no serviço que se comunica com os sensores de temperatura. Caso a temperatura esteja acima de -5 graus Celsius, as partes interessadas são notificadas sobre esta elevação. Caso a temperatura persista acima da prevista, ocorre uma nova notificação após 30 minutos.

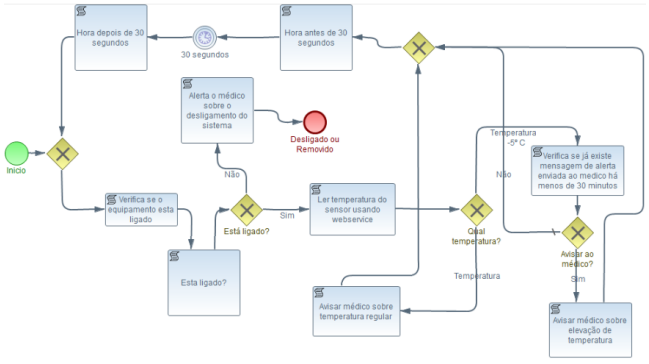

Figura 4: Modelo de processos automatizado puramente em BPMN sem regras de negócio.

Na Fig. 5, é descrito um modelo de processos de negócio para notificar aos responsáveis pelo produto mediante à ocorrência de algum problema. O mecanismo de notificação é disparado mediante a detecção de uma situação particular de interesse previamente especificada. O estado do freezer define se há sensores defeituosos ou se o mesmo parou de funcionar.

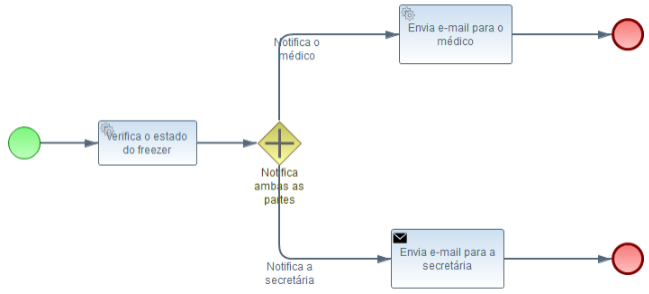

Figura 5: Realiza a notificação das partes interessadas caso haja algum defeito nos sensores ou o freezer pare de funcionar

Na abordagem proposta, tanto serviços internos da implementação da engine de processos quanto serviços externos de e-mail podem ser utilizados para realizar a notificação. De forma comparativa, pode-se perceber que há uma diminuição considerável no

tamanho do processo apresentado na Fig. 5 quando comparado ao processo da Fig. 4. Esta diminuição se dá pois, processos automatizados podem elevar o nível de abstração e diminuir a complexidade de códigos de baixo nível; tornando processos grandes e complexos, em processos menores e menos complexos, permitindo assim uma maior manutenibilidade. Esta abordagem simplifica o entendimento de domínios, dentre eles, os que necessitam de detecção de situações de interesse.

4.2 Visão do Desenvolvedor

A perspectiva do desenvolvedor envolve a modelagem das regras de negócios e a ativação do processo de negócios. Uma definição do Tipo de Situação ou *Situation Type* pode ser visto na Fig. 6, que descreve quais dados são pertinentes ao domínio em questão. Os dados definidos no *Situation Type* devem ser escolhidos com base na natureza e relevância da informação para o domínio, para que seja utilizado no momento de detecção das situações.

```
1  package co.pextra.botox;
2
3  import java.util.List;
4
5  import br.ufes.inf.lprm.scene.model.impl.Situation;
6  import br.ufes.inf.lprm.scene.util.SituationHelper;
7  import br.ufes.inf.lprm.situation.annotations.part;
8  import br.ufes.inf.lprm.situation.model.events.*;
9  import org.kie.api.runtime.process.ProcessInstance;
10
11 declare TemperatureSituation extends Situation
12     sensor: Sensor @part
13     transactions: List @part
14 end
```

Figura 6: Definição do Tipo de Situação

Após a definição do *Situation Type*, é necessário descrever as regras que serão casadas para a detecção de situações. Com isso, caso um determinado conjunto de eventos ocorra, uma situação é detectada e um processo de negócio é iniciado. A Fig. 7 mostra um fragmento de código de definição de *Situation Rules* no cenário descrito.

A Fig. 7 descreve 4 (quatro) regras que tomam como base o *Situation Type* definido como *TemperatureSituation*. A primeira regra analisa se houve mais de 3 (três) elevações de temperatura com o mesmo sensor em um intervalo de 2 (dois) minutos; A segunda regra analisa se houve mais de 5 (cinco) picos de temperatura acima de -5 graus em um intervalo de 2 minutos e a terceira regra analisa se houve mais de 10 (dez) picos de temperatura acima de 0 (zero) grau em um intervalo de 2 (dois) minutos. Por fim, a quarta regra analisa se em um intervalo de tempo de 2 (dois) minutos houve pelo menos 2 (duas) ativações do *TemperatureSituation*; caso isso ocorra, um processo de negócio de notificação às partes interessadas é disparado.

Os *Situation Types* e os *Situation Rules* são interpretados pelo *Knowledge Module*, os *Situation Rules* são armazenados na Rule Base, os dados de temperatura vindos do freezer são armazenados na *Work Memory* e o componente *Situation and Context Engine* realiza o processamento das informações armazenadas tanto na *Work Memory* quanto na *Rule Base*, com o objetivo de realizar um *pattern match*.

```
rule "Ocorreram 3 alterações de temperatura com o mesmo sensor em 2 minutos"
@role(situation)
@type(TemperatureSituation)
    when
       sensor : Sensor();
       transactions: List(size > 3) from accumulate(
          transaction: TemperatureEvent(sensorId == sensor.sensorId)
                over window:time (2m),
          collectList(transaction)
       )
     then
       SituationHelper.situationDetected(drools);
end
rule "Mais de 5 elevações de temperaturas em um intervalo de 2 minutos para um único sensor"
@role(situation)
@type(TemperatureSituation)
    when
       sensor : Sensor();
       transactions: List(size > 5) from accumulate(
          transaction: TemperatureEvent(sensorId == sensor.sensorId, temperature > -5)
                over window:time (2m),
          collectList(transaction)
       )
     then
       SituationHelper.situationDetected(drools);
end
rule "Mais de 10 elevações de temperaturas em um intervalo de 2 minutos para um único sensor"
@role(situation)
@type(TemperatureSituation)
    when
       sensor : Sensor();
       transactions: List(size > 10) from accumulate(
          transaction: TemperatureEvent(sensorId == sensor.sensorId, temperature > 0)
                over window:time (2m),
          collectList(transaction)
       )
     then
       SituationHelper.situationDetected(drools);
end
rule "inicia uma situação quando houver um caso suspeito"
    when
       temperatureSituation: List(size > 2) from accumulate (
         Activation($sit: situation, $sit isA TemperatureSituation.class)
                over window:time (2m);
         collectList($sit)
       )
    then
        //Para 2 (duas) ativações de regras que ocorreram num intervalo de 2 minutos,
        //um processo é iniciado.
        ProcessInstance processInstance = kcontext.getKieRuntime()
                                        .startProcess("co.pextra.botox.notification");
end
```

Figura 7: Definição das Regras de Situações

Ao ocorrer o casamento de padrão e devido à ocorrência um conjunto de eventos de temperatura em um intervalo de tempo relevante para as regras de negócio, o processo de negócio é coletado no repositório de processos, passa pelo *Semantic Module*, que realiza o parser de todo o processo XML e o entrega para o *Process Engine* para que este execute o processo forma assíncrona e automatizada.

4.3 Resultados

O estudo de caso consistiu em aplicar o experimento em 2 (dois) freezeres que realizavam a contagem do número de aberturas e eram medidos por dois motes de sensores contendo sensores de temperatura e luminosidade. O freezer 1 realizava a notificação de temperatura aos *stakeholders* e o freezer 2 não realizava a notificação, apenas realizava a captação dos dados.

Ao decorrer do estudo pôde-se notar que durante os 30 dias do experimento o freezer 1 realizou 39 notificações. A média de abertura dos freezeres girou em torno de 25 a 30 aberturas ao dia, ou seja, tomando como base apenas os dias úteis, em torno de 550 a 660 aberturas ao mês. Vale salientar que os freezeres utilizados no experimento armazenavam também outros medicamentos e não apenas botox. A média de notificações ao mês girou em torno de 5,9% a 7,09% tomando como base o número de aberturas do freezer.

Sendo assim, podemos concluir que a utilização de uma abordagem que facilite a modelagem de situações e processos de negócio pode facilitar a aplicabilidade de soluções tecnológicas a ambientes que necessitam de iterações humanas para tomada de medidas preventivas.

5 CONCLUSÕES

Este artigo apresentou uma arquitetura de suporte ao desenvolvimento de aplicações que objetivam integrar análise de situações contexuais e processos de negócio por meio de regras de negócio, a separação de responsabilidades entre especialistas de domínio e desenvolvedores é um ponto relevante devido a necessidade de cada vez mais recorrente de automatização de processos. Na arquitetura proposta, a definição das regras de negócios e o tratamento de eventos associados a estas, são realizadas pelos desenvolvedores, enquanto que os modelos de processos de negócio são descritos pelos especialistas de processos de negócio. Esta abordagem é interessante considerando que a complexidade de implementação de algumas tarefas, permitindo que estas sejam executadas por meio de modelos de processos de negócio que podem ser automatizados, passando a produzir resultados fiéis aos interesses dos *stakeholders*.

As funcionalidades elencadas na arquitetura tomaram como base um mapeamento sistemático da literatura. O trabalho segue, portanto, o movimento recente de concepção de aplicações sensíveis a situações integradas aos modelos BPM em diferentes domínios de aplicações.

Um estudo de caso foi realizado com o objetivo de validar a solução proposta. O foco principal foi a implementação de uma abordagem baseada na arquitetura proposta para a captura de resultados relevantes. O estudo de caso consistiu em aplicar uma implementação baseada na arquitetura em um cenário real com o objetivo de capturar dados relevantes para análises contextuais e realizar suas respectivas notificações por meio de processos de negócio definidos pelos *stakeholders*.

Os grandes desafios deste trabalho foram, elencar requisitos relevantes para alicerçar a arquitetura proposta, integrar as ferramentas de situações contextuais a processos e conseguir um ambiente real que pudesse ser aplicado o experimento.

O resultado do estudo, demonstrou pontos positivos e de avanço ao estado-da-arte na área de pesquisa; acrescentou questões relevantes de melhorias, bem como, novos pontos a serem tratados em trabalhos futuros. Algumas questões importantes não foram contempladas nesta versão da arquitetura, constituindo portanto, temas naturais para trabalhos futuros. Dentre estes trabalhos, destacam-se: (i) expansão da arquitetura proposta para suportar processos de negócio vindos de repositórios de processos de negócio; (ii) realizar um experimento empírico para o recebimento de *feedbacks* de especialistas acerca da arquitetura proposta; (iii) focar em questões específicas da separação de responsabilidades da arquitetura;

REFERÊNCIAS

[1] Martijn Zoet, Johan Versendaal, Pascal Ravesteyn, and Richard J Welke. Alignment of business process management and business rules. In *ECIS*, page 34, 2011.

[2] Wei Wang, Marta Indulska, and Shazia Sadiq. Integrated modelling of business process models and business rules: a research agenda. ACIS, 2014.

[3] Tim Van Eijndhoven, Maria-Eugenia Iacob, and María Laura Ponisio. Achieving business process flexibility with business rules. In *2008 12th International IEEE Enterprise Distributed Object Computing Conference*, pages 95–104. IEEE, 2008.

[4] Olegas Vasilecas, Diana Kalibatiene, and Dejan Lavbič. Rule-and context-based dynamic business process modelling and simulation. *Journal of Systems and Software*, 122:1–15, 2016.

[5] Michal Bali. *Drools JBoss Rules 5.0 Developer's Guide.* Packt Publishing Ltd, 2009.

[6] Patrícia Dockhorn Costa, João Paulo A Almeida, Isaac SA Pereira, Marten van Sinderen, and Luís Ferreira Pires. Rule-based support for situation management. In *Fusion Methodologies in Crisis Management*, pages 341–364. Springer, 2016.

[7] Talita da Cunha Mattos, Flávia Maria Santoro, Kate Revoredo, and Vanessa Tavares Nunes. A formal representation for context-aware business processes. *Computers in Industry*, 65(8):1193–1214, 2014.

[8] Juan Ye, Simon Dobson, and Susan McKeever. Situation identification techniques in pervasive computing: A review. *Pervasive and mobile computing*, 8(1):36–66, 2012.

[9] Un Hee Schiefelbein, Diovane Soligo, Vinícius Maran, José Palazzo M de Oliveira, João Carlos Damasceno Lima, and Alencar Machado. Pervasive system based on situation-awareness for feedback of energy efficiency. In *Proceedings of the XIV Brazilian Symposium on Information Systems*, page 17. ACM, 2018.

[10] Mariano Nicolas De Maio, Mauricio Salatino, and Esteban Aliverti. *jBPM6 Developer Guide.* Packt Publishing Ltd, 2014.

[11] João LR Moreira, Luís Ferreira Pires, Marten van Sinderen, and Patricia Dockhorn Costa. Towards ontology-driven situation-aware disaster management. *Applied ontology*, 10(3-4):339–353, 2015.

[12] Weihong Huang and Ting Tao. Adding context-awareness to knowledge management in modern enterprises. In *2004 2nd International IEEE Conference on'Intelligent Systems'. Proceedings (IEEE Cat. No. 04EX791)*, volume 2, pages 393–398. IEEE, 2004.

[13] Leonardo A de C Gatti, Flávia Maria Santoro, and Vanessa T Nunes. An agent-based architecture for knowledge management in context-aware business processes. In *The 2010 14th International Conference on Computer Supported Cooperative Work in Design*, pages 318–323. IEEE, 2010.

[14] Ricardo de Almeida Falbo. Mapeamento sistemático. *Retrieved October*, 7, 2018.

[15] Abu Zafar Abbasi and Zubair A Shaikh. A conceptual framework for smart workflow management. In *Information Management and Engineering, 2009. ICIME'09. International Conference on*, pages 574–578. IEEE, 2009.

[16] Antonio Bucchiarone, C Mezzina, and Marco Pistore. Captlang: a language for context-aware and adaptable business processes. In *Proceedings of the Seventh International Workshop on Variability Modelling of Software-intensive Systems*, page 12. ACM, 2013.

[17] João LR Moreira, Luís Ferreira Pires, Marten van Sinderen, and Patrícia Dockhorn Costa. Developing situation-aware applications for disaster management with a distributed rule-based platform. In *Challenge+ DC@ RuleML*, 2015.

[18] Marko Bajec and Marjan Krisper. A methodology and tool support for managing business rules in organisations. *Information Systems*, 30(6):423–443, 2005.

[19] João Paulo A Almeida. Model-driven design of distributed applications. In *OTM Confederated International Conferences"On the Move to Meaningful Internet Systems"*, pages 854–865. Springer, 2004.

[20] Vladimir Vujović, Mirjana Maksimović, Branko Perišić, and Vladimir Milošević. A graphical editor for restful sensor web networks modeling. In *2014 IEEE 9th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI)*, pages 61–66. IEEE, 2014.

[21] Patrick Th Eugster, Pascal A Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The many faces of publish/subscribe. *ACM computing surveys (CSUR)*, 35(2): 114–131, 2003.

[22] Saymon Castro de Souza, José Gonçalves Pereira Filho, and Eugênio Fraga Spessimille. A rule-base approach for wsn application development in a cloud environment, 2013.

[23] Mauricio Salatino, Mariano De Maio, and Esteban Aliverti. *Mastering jboss drools 6.* Packt Publishing Ltd, 2016.

[24] Mark Proctor, Michael Neale, Peter Lin, and Michael Frandsen. Drools documentation. *JBoss*, 5(05):2008, 2008.

[25] John Koenig. Jboss jbpm white paper. *JBoss Labs*, 2004.

[26] Isaac SA Pereira, Patrícia Dockhorn Costa, and João Paulo A Almeida. A rule-based platform for situation management. In *2013 IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA)*, pages 83–90. IEEE, 2013.

[27] Mica Endsley. Endsley, m.r.: Toward a theory of situation awareness in dynamic systems. human factors journal 37(1), 32-64. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 37:32–64, 03 1995. doi: 10.1518/001872095779049543.