Uma Abordagem Baseada em Modelos para Gerenciamento de Situações em CEP

Rafael Simonassi Amorim, Patrícia Dockhorn Costa, Roberta Lima Gomes

Programa de Pós-Graduação em Informática, Universidade Federal do Espírito Santo (UFES)
Av. Fernando Ferrari, s/n, Vitória, ES, Brasil

simonassiamorim@inf.ufes.br, pdcosta@inf.ufes.br, rgomes@inf.ufes.br

Abstract. *This paper proposes a methodology to support situation awareness in complex event processing (CEP) systems, both at design time and runtime. At design time, the graphical language SML has been used, allowing developers to model situations at a high level of abstraction. At runtime, transformations were implemented from SML models to Java code, allowing the target platform (Esper) to detect and process situations described in SML. To illustrate the proposed methodology, an application scenario for monitoring influenza epidemics has been developed according to the recommendations described by the World Health Organization.*

Resumo. *Este trabalho propõe uma metodologia para apoiar o uso de situações em sistemas de processamento de eventos complexos (CEP), tanto em tempo de design quanto em tempo de execução. Em tempo de design foi utilizada a linguagem gráfica SML, permitindo que desenvolvedores modelem situações num alto nível de abstração. Em tempo de execução, foram implementadas transformações de modelos SML para código Java, permitindo que a plataforma alvo (Esper) seja capaz de detectar e processar as situações descritas em SML. Para exemplificar a metodologia proposta, foi desenvolvida uma aplicação voltada para o cenário de detecção de Influenza, segundo as recomendações descritas pela Organização Mundial de Saúde.*

1. Introdução

Sistemas de informação atuais utilizam, cada vez mais, eventos como forma de comunicação e interação. Eventos podem ser entendidos como "algo que aconteceu ou está contemplado para acontecer no domínio" [Etzion and Niblett 2010]. Essa demanda levou à necessidade de gerenciar e manipular eventos de forma automática, sistemática e em tempo hábil [Eckert and Bry 2009]. Sistemas CEP (*Complex Event Processing*) [Luckham 2002] surgiram com o intuito de suprir essa demanda, permitindo a manipulação de uma enorme quantidade de dados (*i.e*, eventos) heterogêneos com tempo de resposta (quase) real. Esses sistemas provêm mecanismos para a criação de eventos a partir da composição ou agregação de outros eventos.

Sistemas CEP visam suprir a lacuna entre eventos que ocorrem no ambiente e os estados (padrões) particulares de interesse sobre os quais o sistema deseja agir (ou reagir). Sistemas com essas características, muitas vezes referido como *context-awareness* [Dey 2001], têm como objetivo primário prover uma interação eficaz entre os usuários,

adaptando automaticamente o comportamento da aplicação de acordo com a situação atual (ou prevista) do usuário. [Costa 2007] define uma situação como "um estado particular de interesse para aplicações cujos constituintes podem ser uma combinação de entidades e de suas condições contextuais". O conceito de situação permite aos usuários (desenvolvedores, projetista, etc.) criar abstrações a respeito das entidades que compõem uma determinada situação. Em vez de manipular entidades de baixo nível de abstração (ex. dados sensoriais que representam a temperatura de um individuo), é possível concentrar-se em padrões de alto nível de abstração construídos a partir dessas entidades de nível de abstração inferior (ex. a situação "febre de um individuo").

Para usufruir dos benefícios provenientes da utilização de situações em aplicações *context-awareness*, é necessário prover suporte adequado (i) em tempo de *design* [Sobral 2015], [Costa 2012], [Costa 2007]; e (ii) em tempo de execução [Rizzi 2014], [Pereira 2013]. Em tempo de *design* é necessário prover suporte para a especificação de situações num alto nível de abstração, permitindo ao desenvolvedor definir situações complexas (*i.e*, compostas) em termos de situações definidas previamente. Em tempo de execução é necessário prover suporte para detectar e manter informações sobre as situações, permitindo ao sistema reagir adequadamente à ocorrência de cada situação.

O presente trabalho tem como objetivo propor uma metodologia para auxiliar a especificação, detecção e processamento de situações em sistemas de CEP. Com a utilização de situações em sistemas de CEP, espera-se proporcionar os seguintes benefícios: (i) incorporar a lógica de detecção e processamento de situações ao serviço CEP, a fim de retirar tal complexidade e responsabilidade das aplicações cliente, (ii) diminuir o tráfego de eventos desnecessários na rede e (iii) garantir um acordo semântico entre os clientes. A metodologia proposta baseia-se na transformação de modelos de situações, especificados na linguagem SML [Costa 2012], para um conjunto de regras a serem executadas em uma plataforma CEP. A fim de exemplificar tal metodologia, o trabalho propõe implementar transformações de SML para a plataforma Esper, utilizando a ferramenta Acceleo[1]. É proposta também uma aplicação voltada para o cenário de detecção de *Influenza* com base nas especificações e recomendações apresentadas pela Organização Mundial da Saúde[2].

O presente trabalho está dividido da seguinte forma: a Seção 2 discute a noção de situação e os requisitos necessários para uma abordagem utilizando tais conceitos; a Seção 3 descreve o cenário de detecção de Influenza, utilizado como motivação pelo trabalho; a Seção 4 apresenta uma visão geral da abordagem; a Seção 5 discute trabalhos relacionados e, por fim, a Seção 6 apresenta as conclusões e trabalhos relacionados.

2. Situações

Situações são entidades compostas cujos constituintes são outras entidades, suas propriedades e as relações em que estão envolvidos [Costa 2007]. Exemplos de situações são: "John está com febre", "John teve uma febre intermitente durante os últimos seis meses", *etc*. Um *situation type* (tipo de situação) [Kokar 2009] nos permite

[1] http://www.acceleo.org/

[2] http://www.who.int

considerar as características gerais de situações de um tipo particular, capturando os critérios de unidade e identidade de situações desse tipo. Um mesmo *situation type* pode ser instanciado múltiplas vezes e pode ser definido em termos de *situation types* mais simples. Um exemplo de *situation type* é "paciente está com febre". Instâncias dessa situação são criadas à medida que instâncias de "paciente" ("John", "Paul", *etc*.) estejam na condição de "está com febre", como por exemplo, a instância "John está com febre".

Para que situações possam ser detectadas (*i.e* instâncias de um *situation type*) é necessário identificar se existem instâncias das entidades envolvidas na situação cujas propriedades satisfazem as restrições capturadas no *situation type*. Enquanto as restrições capturadas no *situation type* forem satisfeitas, dizemos que a situação está ativa. A partir do momento em que uma das restrições definidas pelo *situation type* não é mais satisfeita, essa situação deixa de existir e, nesse caso, é dita uma situação passada. Com base nas características de situações apresentadas, é possível destacar os seguintes requisitos básicos para uma abordagem baseada em situações:

1. *Situation types* devem ser definidos em tempo de *design*, e suas instâncias devem ser detectadas em tempo de execução;

2. *Situation types* devem ser definidos em relação às entidades, bem como as restrições sobre as propriedades e relações dessas entidades;

3. As propriedades temporais das situações devem ser consideradas (ex. tempo inicial).

3. Cenário de Aplicação e SML

Anualmente, em todo o mundo estima-se que epidemias de *Influenza* resultam em cerca de 3 a 5 milhões de casos graves da doença, e cerca de 250.000 a 500.000 mortes[3]. Dentre os possíveis sintomas, estão incluídos: alta súbita de febre, tosse, dor de cabeça, dor muscular e articular, mal-estar, dor de garganta e coriza. A *Influenza* espalha-se com facilidade, tornando-se de extrema importância monitorar as ocorrências de seus casos. O monitoramento visa minimizar o impacto da doença, fornecendo informações úteis para as autoridades responsáveis pela saúde pública. Essas informações possibilitam um melhor planejamento sobre as medidas de controle e intervenção apropriadas.

Baseado nas informações acima, este trabalho propõe a implementação de uma aplicação-cenário que visa o monitoramento de áreas de foco de *Influenza*, a fim de auxiliar no processo de tomada de decisões. É possível que uma pessoa se encaixe em dois possíveis quadros: (i) *Influenza-Like Illness* (ILI) ou (ii) *Severe Acute Respiratory Infections* (SARI). ILI é uma infecção respiratória aguda na qual o paciente apresenta, há pelo menos 10 dias, (i) medição de febre com temperatura maior que 37 graus célsius e (ii) tosse. SARI representa um quadro de ILI que requeira hospitalização. Com base nessas informações, foram definidos três tipos de situações principais:

1. Situação Febre: ocorre se a temperatura de um paciente for superior a 37 graus;

2. Situação ILI: ocorre para todo paciente na situação febre e tosse por 10 dias;

3. Situação SARI: ocorre para todo paciente na situação ILI que requeira hospitalização.

[3] http://www.who.int/mediacentre/factsheets/fs211/en/

A fim de modelar as situações desejadas, o presente trabalho usa a linguagem gráfica SML (*Situation Modeling Language*), que provê um editor gráfico voltado para a modelagem de situações, facilitando a definição de *situation types* em tempo de *design*. A definição de *situation types* em SML é composta por dois modelos: um modelo de contexto e um modelo de *situation types*. O modelo de contexto define as entidades e relacionamentos existentes no domínio. O modelo de *situation types* define as situações do domínio na forma de padrões gerados a partir das entidades e relações especificadas no modelo de contexto. Como o modelo de contexto para o cenário proposto é simplório, composto de apenas um tipo de entidade, Paciente (*Patient*) e seu contexto temperatura (*temperature*), o mesmo será omitido. A partir do cenário descrito, foram modeladas três situações simples: Febre (*Fever*), Tosse (*Cough*) e RequerHospitalização (*RequiresHospitalization*). A Figura 1 ilustra os diagramas SML desenvolvidos. Duas situações compostas foram modeladas a partir do cenário: ILI e SARI (Figura 2).

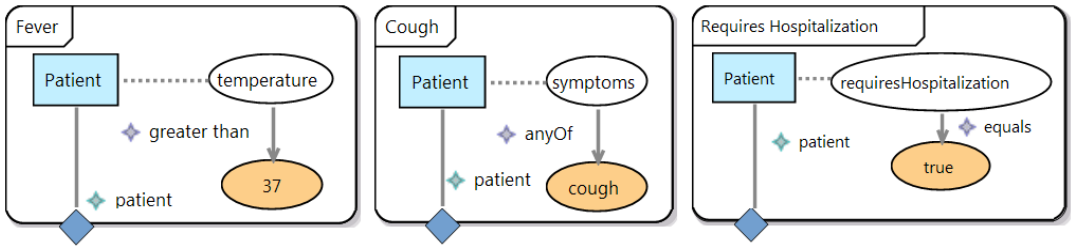

Figura 1. Especificação das Situações Simples do Cenário utilizando SML

Cada retângulo maior representa a especificação de uma situação, composta por participantes do padrão bem como suas relações e atributos. Em SML um participante pode ser uma Entidade, uma Situação ou um *Relator* (representando a ocorrência de um contexto relacional). Mais detalhes sobre SML em [Costa and Mielke 2012].

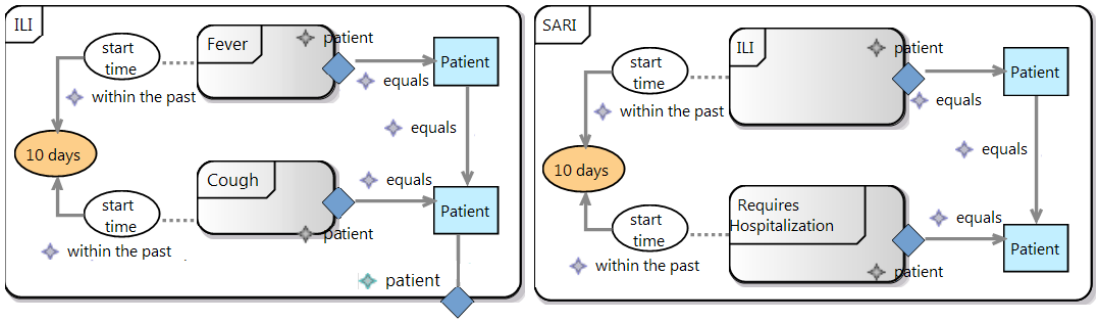

Figura 2. Especificação das Situações Compostas do Cenário utilizando SML

Cada retângulo menor representa uma entidade. Tomemos por exemplo a situação *Fever* (Figura 1). A entidade apresentada corresponde ao tipo *Patient*. *Patient* está associado a um atributo, representado por uma elipse branca, com nome "*temperature*". Esse atributo, por sua vez, está associado a um literal, representado por uma elipse preenchida, de valor igual a 37. O atributo e o seu respectivo valor (literal) são relacionados por meio de uma seta. A seta representa a ocorrência de uma *relação formal*. Na situação Febre, a relação formal utilizada corresponde à relação "maior que" (*greater than*). Considere que todos os elementos utilizados nos diagramas, bem como suas relações, foram definidos previamente no modelo de contexto.

Dessa forma, o diagrama referente à situação Febre descreve que a mesma ocorre quando uma entidade do tipo Paciente possui um atributo de nome "temperatura" com valor maior que 37. Além disso, é possível ver que a entidade Paciente possui uma associação denominada *patient*, conectada à borda da situação por meio de um losango. Essa relação indica que a entidade Paciente envolvida na situação pode ser referenciada por outras situações por meio do nome *patient.* É importante observar que as situações ILI e SARI são compostas por situações mais simples. ILI é composta pelas situações Febre e Tosse, enquanto SARI é compostas pelas situações ILI e RequerHospitalização. Ambas as situações que compõem as situações ILI e SARI estão associados por uma janela temporal, representada pela relação formal "*whitin the past*" e o atributo tempo inicial (*start_time*), presente em toda situação. Essa relação indica que a situação deve estar ativa pelos últimos 10 dias. As regras e padrões SML utilizados na especificação de situações serão abordadas em mais detalhes nas seções 4.3 e 4.4.

4. Abordagem Orientada a Modelos

4.1. Esper

Esper[4] é um componente *open source* para CEP, distribuído nas linguagens Java e .NET, capaz de lidar com diferentes fluxos de dados (eventos). A manipulação de eventos em Esper é comparada como um banco de dados invertido: em vez de armazenar os dados e executar *queries* sobre estes dados, Esper armazena *queries* e as executa sobre os fluxos de dados. O modelo de execução é contínuo: *queries* são avaliadas sempre que um novo dado é recebido, e não apenas uma vez como ocorre tradicionalmente em banco de dados. Esper provê uma linguagem denominada Esper EPL (*Event Pattern Language*). Essa linguagem é similar à linguagem SQL (*Structured Query Language*), tanto no uso de clausulas *select* quanto *where*. Porém, ao invés de utilizar tabelas, EPL faz uso do conceito de *views*. *Views* são análogas às tabelas SQL, definindo quais dados estão disponíveis para consultas (*queries*). *Views* podem representar um fluxo inteiro de dados ou uma janela (temporal ou espacial) aplicada sobre um dado fluxo. Em uma *view* é possível aplicar ordenação (*order by*), agrupamento (*group by*), manipular propriedades dos eventos, dentre outros, da mesma forma como em SQL. A espeficiação de *views* é realizada por meio da cláusula *from*.

Um exemplo de *query* EPL pode ser dado por "*select avg(price) from StockTickEvent.win:time(30 sec)*". Essa *query* computa a média de preço (*price*) dos eventos do tipo *StockTickEvent* recebidos nos últimos 30 segundos. Para cada novo evento do tipo *StockTickEvent* recebido pela *engine* Esper, a *query* é reavaliada. Nesse exemplo, a *view* é representada por uma janela temporal, *win:tim(30 sec)*, aplicada sobre um fluxo de dados (eventos do tipo *StrockStrickEvent*). Para essa *query*, assume-se que existe uma classe (Java) *StockTickEvent* cujo um dos atributos tem nome igual à *price*.

4.2. Visão geral da abordagem

A abordagem aqui proposta se apoia na geração automática de código a partir de modelos descritos em SML. O código gerado permite que a plataforma alvo seja capaz

[4] http://www.espertech.com/esper/index.php

de detectar as situações descritas no modelo SML. Para ilustrar a abordagem, foi utilizada a plataforma Esper. A Figura 3 apresenta uma visão geral da abordagem. São aplicadas duas transformações, uma para cada modelo SML. A primeira transformação (T1) é responsável por gerar código a partir dos modelos de contexto. O código gerado corresponde às classes Java que representam as entidades descritas no modelo. A segunda transformação (T2) é responsável por gerar código a partir dos modelos de situações. O código gerado corresponde a dois módulos: (i) um módulo contendo classes Java que representam as situações modeladas e (ii) um módulo contendo *queries* EPL correspondentes aos padrões de ativação e desativação das situações.

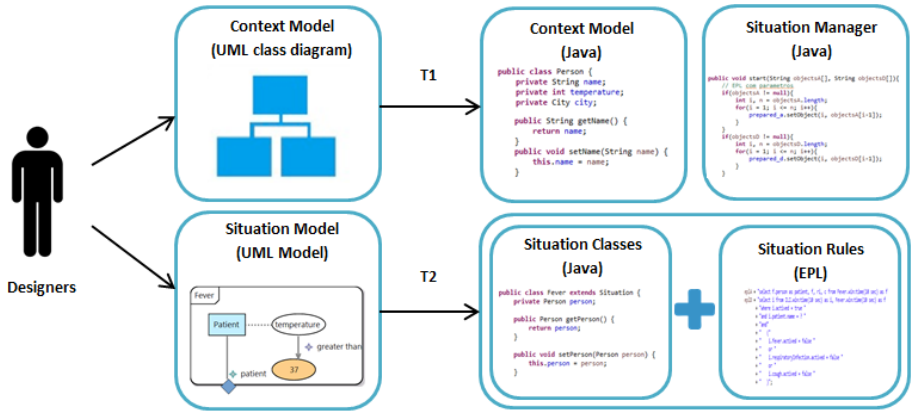

Figura 3. Visão Geral da Abordagem

A transformação (T1) também gera o *Situation Manager*, que consiste numa camada de abstração desenvolvida sobre o Esper, sendo uma das contribuições do presente trabalho. O *Situation Manager* permite ao Esper detectar e gerir o ciclo de vida das situações. Para cada diagrama SML modelado, duas *queries* são geradas, uma referente à regra de ativação e outra referente à regra de desativação da situação. O *Situation Manager* gerencia o ciclo de vida das situações, impedindo a execução de regras de ativação referentes à situações já ativas e a execução de regras de desativação referentes à situações já desativadas. O código correspondente ao *Situation Manager* é fixo, pois seu comportamento independe das situações modeladas para o domínio.

Além do suporte à gerência de situações, o *Situation Manager* visa auxiliar o processo de composição de situações. Utilizando o *Situation Manager* é possível, por exemplo, definir uma situação Febre Intermitente como a ocorrência de um evento de Febre desativada seguida de um evento de uma Febre ativa. É responsabilidade do *Situation Manager* verificar se cada situação participante encontra-se ativa ou desativada. Sem tal suporte, seria necessário definir a situação em função de eventos primitivos: um evento de temperatura maior que 37, seguido de um evento de temperatura menor ou igual a 37, seguido de um evento de temperatura maior que 37.

4.3 Geração de uma regra de situação

Cada *situation type* modelado dá origem a duas regras EPL, uma responsável pela ativação da situação e uma responsável pela desativação da situação. Nessa seção serão apresentadas as regras EPL geradas com base nas situações modeladas nas Figura 1 e Figura 2. Por motivos de simplicidade serão apresentadas apenas as regras EPL referentes às situações Febre e ILI, uma vez que as demais seguem os mesmos padrões.

A situação Febre deve ser ativada quando é detectado um paciente com temperatura superior a 37. A Figura 4 ilustra a regra de ativação correspondente à situação Febre. Na cláusula *from* o tipo *Patient* indica o fluxo de dados a ser capturado pela *view*. Sobre essa *view* é aplicada a restrição, "*temperature* > 37", indicando que apenas eventos do tipo *Patient,* com um atributo *temperature* cujo valor seja superior a 37, devem ser filtrados. Aos objetos filtrados é aplicado um *label*, cujo nome dado é *Patient*, através da palavra reservada "*as*". Esse *label* é utilizado na cláusula *select*, indicando que os objetos filtrados devem ser retornados pela *query*. A figura referente à regra de desativação gerada para essa situação é análoga e, por isso, será omitida. A diferença é dada pela negação da restrição utilizando o operador *not*, alterando o filtro para os eventos do tipo *Patient* com atributo *temperature* com valor menor ou igual a 37.

```
select Patient
from Patient(temperature > 37) as Patient
```

Figura 4. Regra de ativação da situação Febre

A situação ILI é uma situação composta e sua ativação depende da ativação de duas outras situações: *Fever* e *Cough*. Ambas as situações participantes estão associadas ao mesmo *Patient* e devem estar ativas por um período de 10 dias. A Figura 5 ilustra as regras de ativação e desativação correspondentes à situação ILI. Na cláusula *from* da regra de ativação são especificados dois tipos, *Fever* e *Cough*, indicando fluxos de eventos (*views)* diferentes. Em Esper, é necessário especificar uma janela temporal sempre que duas ou mais *views* são aplicados sobre uma mesma *query*. Para especificar uma janela, foi aplicada a função *win:time(10 day)* sobre cada *view*, indicando uma janela temporal com duração de 10 dias. Assim como na situação Febre, ambas as *views* receberam um *label*, devidamente utilizado pela cláusula *select*.

Regra de Ativação	Regra de Desativação
select Fever.Patient as Patient, Fever, Cough from Fever.win:time(10 day) as Fever, Cough.win:time(10 day) as Cough where Fever.actived = true and Cough.actived = true and Fever.Patient. id = Cough.Patient.id	select ILI from ILI.win:time(10 sec) as ILI, Fever.win:time(10 sec) as Fever, Cough.win:time(10 sec) as Cough where ILI.actived = true and (not (ILI.Fever.actived = true) or not (ILI.Cough.actived = true) or not (ILI.Fever.Patient.id = ILI.Cough.Patient.id))

Figura 5. Regras de ativação e desativação da situação ILI

Na clausula *where* da regra de ativação foram aplicadas as restrições necessárias: ambas as situações devem: (i) estar ativas e; (ii) fazer referencia ao mesmo paciente. A primeira restrição é garantida através do atributo *actived,* enquanto a segunda é garantida pela comparação entre os atributos *id,* identificando unicamente cada entidade. A regra de desativação correspondente é similar à de ativação, com três diferenças: (i) o tipo de evento ILI inserido na cláusula *from* como um novo fluxo de eventos; (ii) as restrições negadas e; (iii) a substituição dos operadores *and* pelo operador *or*, indicando que a situação deve ser desativada caso uma das restrições não seja mais satisfeita.

4.4. Generalização dos mapeamentos

Baseados nos estudos da Seção 4.3 foram definidos padrões básicos (atômicos) em SML. Por simplicidade, será apresentado apenas um subconjunto dos mapeamentos

implementados por este trabalho, conforme as necessidades do cenário proposto na Seção 3. O processo de transformação consiste em decompor padrões complexos em ocorrências dos padrões básicos, dividindo o problema em partes menores com solução conhecida. A Figura 6 ilustra a transformação de um padrão de *Situation Type* em SML envolvendo Entidade (*Entity*), Relação Formal (*relation*), Atributo (*attribute*) e Valor (*value*). Entidades representam tipos de objetos que desejamos filtrar. O tipo de objeto mapeado pela Entidade é utilizado na clausula *select*, permitindo que a aplicação faça uso dos objetos filtrados, e na clausula *from,* especificando as *views*. Atributos e Relações geram restrições sobre os objetos, nas clausulas *from*. O valor associado a um atributo é convertido num valor literal de acordo com o tipo de dado especificado para o atributo no modelo de contexto. A *query* EPL de desativação para o diagrama da Figura 6 é análoga à *query* EPL de ativação, com sua restrição negada pelo operador *not* e, por simplicidade, será omitida.

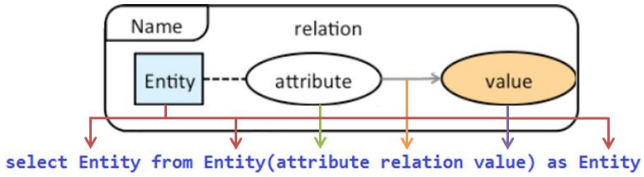

Figura 6. Regra de transformação para Entidade, Relação Formal e Atributo.

A Figura 7 ilustra a regra de transformação para a *query* EPL de ativação referente a uma situação composta. Cada situação participante (*SituationA* e *SituationB*) é composta por suas próprias entidades (*EntityA* e *EntityB*). Por sua vez, cada entidade é composta por seus próprios atributos (*attributeA* e *atributeB*).

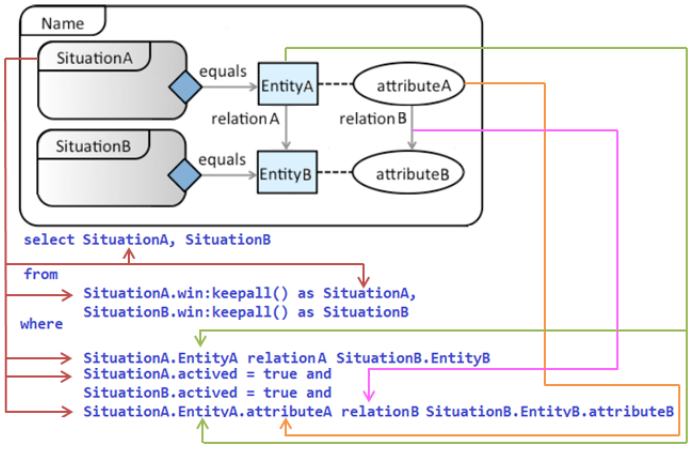

Figura 7. Regra de transformação para EPL de ativação de situação composta

Como as entidades participantes são também situações (retângulos de bordas arredondas), deve-se levar em consideração o *status* de cada situação participante. Em SML essa distinção é feita pela cor utilizada no retângulo que representa a situação participante. A cor cinza (Figura 7) indica uma situação ativa, enquanto que a cor branca indica uma situação passada. Na cláusula *from* a função *win*:*keepall*() é aplicada em cada situação. Como a Figura 7 não especifica janelas para as *views*, por padrão é aplicada a função *win*:*keepall*(), indicando que todos os eventos do tipo *SituationA* e *SituationB* recebidos ao longo do tempo devem ser avaliados.

5. Trabalhos Relacionados

Segundo [Margara 2014], a utilização adequada de tecnologias CEP depende de uma modelagem correta e precisa a respeito do domínio em questão. Porém, esse aspecto tem sido pouco explorado na literatura, uma vez que a maioria dos trabalhos encontrados visa prover processamento eficiente de eventos, e não focam em aspectos de modelagem de eventos e regras. Este trabalho visa apoiar desenvolvedores de sistemas CEP por meio da utilização de linguagem gráfica para modelagem de situações (SML) e da geração automática de regras EPL.

Em [Bruns 2014] foi desenvolvida uma linguagem específica de domínio baseada em Esper EPL, denominada DS-EPL (*Domain-Specific Event Processing Language*). Essa abordagem é semelhante a adotada no presente trabalho: elementos do domínio devem ser mapeados num modelo similar ao modelo de contexto, permitindo que a linguagem gerada utilize conceitos relacionados ao dominio modelado. Porém, ainda que DS-EPL permita ao desenvolvedor trabalhar num nível mais alto de abstração, o mesmo deve conhecer as regras, padrões e demais restrições que compõem a linguagem DS-EPL gerada. SML também é uma linguagem especifica de domínio, porém permite que o usuário modele situações de forma gráfica, totalmente independente de tecnologia.

Outro trabalhos relacionados também visam a especificação e detecção de situações [Fidler 2005] [Hasan 2011] [Renners 2012], porém sem levar em consideração o ciclo de vida associado às situações. Definir e gerir situações utilizando eventos primitivos são tarefas complexas, principalmente quando se trata de situações com alto nível de composição (situações complexas compostas por demais situações compostas). O presente trabalho defende que a gerência do ciclo de vida das situações deve estar incorporada ao sistema CEP, permitindo ao desenvolvedor trabalhar em um nível mais alto de abstração, definindo situações compostas a partir de situações mais simples, definidas previamente.

6. Considerações Finais e Trabalhos Futuros

Este trabalho propôs uma metodologia para auxiliar a especificação, detecção e processamento de situações em sistemas CEP por meio de uma abordagem de desenvolvimento orientado a modelos. Tal metodologia provê suporte tanto em tempo de *design* quanto em tempo de execução: em tempo de *design* foi utilizada a linguagem gráfica específica de domínio (SML), permitindo ao desenvolvedor especificar situações com um alto nível de abstração; em tempo de execução foi criada uma camada de abstração sobre a plataforma Esper capaz de detectar e gerenciar as situações. Como exemplo, foi apresentado o cenário de *Influenza*, juntamente com os padrões de transformações utilizados para mapear os diagramas SML em código Java referente às *queries* EPL (de ativação de desativação) correspondentes.

Como trabalhos futuros espera-se desenvolver testes de desempenho, utilizando a plataforma Esper, de forma que possa ser analisado o impacto da gerência e detecção de situações sobre a plataforma. Sem uma análise detalhada de desempenho, não foi possível concluir se algumas melhorias esperadas foram alcançadas, como por exemplo, a diminuição no número de eventos disseminados pela rede. Além disso, como a metodologia é independente de plataforma, futuramente espera-se implementar transformações para diferentes plataformas alvos a fim de comprovar tal independência.

Referências

BRUNS, R. et al. DS-EPL: Domain-Specific Event Processing Language. **DEBS '14 Proc. 8th ACM Intl' Conf. on Distributed Event-Based Systems**, 2014. 83-94.

COSTA, P. D. Architectural Support for Context-Aware Applications - From Context Models to Services Platforms. **University of Twente**, 2007.

COSTA, P. D. et al. A Model-Driven Approach to Situations: Situation Modeling and Rule-Based Situation Detection. **2012 IEEE 16th International Enterprise Distributed Object Computing Conference**, 2012. 154-163.

DEY, A. K. Understanding and Using Context. **Personal and Ubiquitous Comp, vol. 5,** 2001.

ECKERT, M.; BRY, F. Complex Event Processing (CEP). **Informatik-Spektrum**, April 2009.

ETZION, O.; NIBLETT, P. **Event Processing in Action**. Manning Publications Co., 2010.

FIDLER, E. et al. The padres distributed publish/subscribe system. **8th Intl' Conf. on Feature Interactions in Telecommunications and Software Systems**, 2005.

HASAN, S. et al. Toward Situation Awareness for the Semantic Sensor Web: Complex Event Processing with Dynamic Linked Data Enrichment. **Proc 4th Intl' Workshop on Semantic Sensor Networks 2011**, 2011.

KOKAR, M. M.; MATHEUS, C. J.; BACLAWSKI, K. Ontology-based situation awareness. **Information Fusion, vol. 10**, 2009. 83-98.

LUCKHAM, D. C. **The Power of Events:** An Introduction to Complex Event Processing in Distributed Enterprise Systems. Boston, USA: Addison-Wesley, 2002.

MARGARA, A.; CUGOLA, G.; TAMBURRELLI, G. Learning from the past: automated rule generation for complex event processing. **DEBS '14 Proc. 8th ACM Intl' Conf. on Distributed Event-Based Systems**, 2014. 47-58.

PEREIRA, I.; DOCKHORN, P. C.; ALMEIDA, J. P. A. A Rule Based Platform for Situation Management. **Intl' Multi-Disciplinary Conf. on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA)**, 2013.

RENNERS, L.; BRUNS, R.; DUNKEL, J. Situation-Aware Energy Control by Combining Simple Sensors and Complex Event Processing. **Proc. Workshop on AI Problems and Approaches for Intelligent Environments**, 2012. 33-38.

RIZZI RAYMYNDO, C. et al. An Infrastructure for Distributed Rule-Based Situation Management. **Intl' Multi-Disciplinary Conf. on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA)**, 2014. 202-208.

SOBRAL, V. M.; ALMEIDA, J. P. A.; COSTA, P. D. Assessing Situation Models with a Lightweight Formal Method. **Intl' Multi-Disciplinary Conf. on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA)**, 2015.