# Uso de Complex Event Processing em sistemas de supervisão elétrica

Clayton Tolotti, Marcelo Trindade Rebonatto
Ciência da Computação – Universidade de Passo Fundo (UPF) – Passo Fundo – RS – Brasil
claytontolotti@gmail.com, rebonatto@upf.br

**Resumo: O processamento de eventos complexos combina dados de várias fontes para inferir eventos ou padrões que sugerem circunstâncias mais complicadas. A plataforma Protegemed contribui para segurança elétrica em procedimentos cirúrgicos por meio do registro de eventos que podem gerar periculosidade ao paciente. O objetivo desse trabalho é explorar o processamento de eventos complexos no Protegemed. Para a realização dos testes foi utilizado o sistema de gerenciamento de regras Drools para o processamento de eventos oriundos do Protegemed. Os resultados obtidos comprovaram a importância do uso de processamento de eventos complexos no Protegemed. O sistema de regras de produção do Drools se mostrou eficiente para detectar e inferir corretamente as regras definidas para as situações em que os pacientes podem estar expostos, aumentando a detecção de periculosidade em eventos simultâneos em 28,33%.**

***Palavras-chave:*** CEP, Drools, Processamento de eventos, Protegemed.

***Abstract****: Complex event processing combines data from multiple sources to infer events or patterns that suggest circumstances more complicated. The Protegemed platform contributes to electrical safety in surgical procedures by recording events that may cause danger to the patient. The purpose of this paper is to explore the processing of complex events in Protegemed. In order to perform the tests, the Drools rules management system was used to process events from Protegemed. The results obtained proved the importance of using complex event processing in Protegemed. The Drools production rules system proved efficient in correctly detecting and inferring rules defined for situations in which patients may be exposed, increasing the detection of dangerousness in simultaneous events in 28,33%.*

***Keywords****: CEP, Drools, Event processing, Protegemed.*

## 1 Introdução

Estamos na era em que a área computacional está em rápida transformação, presente nos mais diversos segmentos, como indústria, comércio, saúde, finanças, entre outros. As organizações geram um grande volume de dados, ao mesmo tempo em que precisam armazenar o que está relacionado ao seu interesse, aliado a necessidade de transformar dado em informação, esse cenário implica processar essa grande quantidade de dados disponível em seus ambientes, para buscar padrões, analisar comportamentos, encontrar informações que possam agregar mais qualidade na entrega de produtos e serviços. Uma delas que pode ser recuperada e quantificada, é o risco que pode ocorrer durante um determinado processo. O que pode ser aceito como tolerável para uma organização, pode representar uma situação catastrófica para outra, principalmente setores ligados à área da saúde, onde diariamente trabalha-se para salvar e zelar vidas humanas.

O projeto Protegemed foi concebido para fornecer suporte a equipes médicas e profissionais ligados à área médica. O sistema é composto por uma plataforma de hardware e software para realizar supervisão elétrica de equipamentos eletromédicos. Inicialmente aplicado exclusivamente a Equipamentos Eletromédicos, hoje está alcançando outros tipos de equipamentos elétricos. O Protegemed é composto de três unidades principais, uma que realiza o monitoramento da rede elétrica e captura dados, denominado módulo; outra que recebe os dados obtidos e os armazena em um banco de dados, chamada de servidor; e a última que consulta o banco de dados e exibe as informações capturadas em um software web, conhecida como software de apoio.

O Protegemed realiza o armazenamento dos dados de forma tradicional, e através do software de apoio realiza a recuperação dos dados, gerando informações. Esse processo se mostra confiável e eficaz, mas apresenta limitação frente a determinadas necessidades, por acrescentar um certo atraso entre a captura do dado realizada pelo módulo, até a visualização da informação, isso é natural devido a arquitetura envolvida. Para essas situações

surgiu o *Complex Event Processing* (CEP – Processamento de Eventos Complexos), que apresenta uma nova perspectiva sobre o poder de transformação de dados ou eventos como são chamadas em informações.

A tecnologia CEP possibilita que determinados riscos durante uma operação, possam ser detectados com um tempo de atraso muito menor utilizando processamento de eventos. O tempo é um fator determinante para que riscos possam ser evitados ou minimizados, uma vez que, um ambiente hospitalar, a operação é essencialmente dependente de uma fonte de energia elétrica, expondo assim, pacientes a possíveis ondas de micro choques durante procedimentos médicos, que através de uma aplicação CEP, podem ser identificadas o mais rápido possível e aplicando o devido tratamento, evitando ou minimizando a exposição do paciente a um determinado grau de periculosidade.

O objetivo desse trabalho é explorar o uso de CEP na plataforma Protegemed, melhorando o processo de detecção da periculosidade de eventos ao paciente, melhorando a detecção de micro choques. O restante do texto está dividido em 7 capítulos. No capítulo 2 está descrito a importância do Protegemed e seu funcionamento. No capítulo 3 é abordado o que é o Processamento de Eventos Complexos, o que é um evento simples e evento complexo, como isto está relacionado ao Protegemed, descrição dos diferentes motores CEP existentes no mercado, qual será utilizado para o Protegemed e os motivos da escolha. No capítulo 4 está descrito os trabalhos relacionados a utilização de CEP em projetos, com foco principal a área médica ou hospitalar, área do Protegemed. No capítulo 5 é exposto o modelo de detecção de periculosidade utilizando o sistema de regras de produção do Drools para o Protegemed. No capítulo 6 é descrito a implementação do modelo de regras no software para processar os eventos oriundos do Protegemed. No capítulo 7 é realizado análise dos resultados obtidos através do processamento de eventos com CEP, relacionados a forma da detecção dos mesmos eventos definidos em [3]. Por fim, no capítulo 8 consta as considerações finais do autor.

## 2 Protegemed

O Protegemed é um projeto composto por *hardware*, *firmware*, *software* de apoio e banco de dados, que foi desenvolvido para detectar falhas elétricas de equipamentos eletromédicos (EEM) usados durantes procedimentos cirúrgicos. O objetivo é diminuir os riscos de choques elétricos, em específico micro choques que os pacientes podem receber durante um procedimento. O micro choque é aquele de pequeno valor e que ocorre interno ao corpo humano. No Protegemed, o micro choque é caracterizado pela passagem de uma corrente pelo corpo humano com um valor menor do que 2,0 miliamperes [1].

Os micros choques resultam normalmente das correntes de fuga, que são correntes que seguem um fluxo diferente do padrão normal, podendo atingir um indivíduo em contato com um equipamento elétrico. Os EEM são uma fonte das correntes de fuga. Para identificar os micros choques, o Protegemed utiliza componentes para detecção das correntes e um computador para processar as informações desta corrente [1].

O Protegemed consegue capturar Forma de Onde (FO) de fase e fuga, gerar alertas de risco de micro- choque, calcular a periculosidade deste risco utilizando: valor eficaz da corrente, valor eficaz da corrente para cada frequência entre 60 Hz e seus harmônicos até 720 Hz e análise da similaridade das formas de onda. A captura dos dados é realizada por um componente embarcado (módulo) que possui um *firmware* (conjunto de instruções) para monitorar os valores de corrente elétrica. Os alertas, o cálculo da periculosidade e a análise da similaridade são executados em um software de apoio [1].

A captura dos dados é realizada pelo módulo MBED [2], uma plataforma que contém um microcontrolador para realizar o processamento das instruções contidas no *firmware*. Após o recebimento, ele as envia para a interface responsável pela comunicação com o banco de dados (BD), o MBED utilizado no Protegemed, não implementa um relógio de tempo real para enviar juntamente um *timestamp*, informando a hora em que a corrente de fuga aconteceu. Esta tarefa hoje está a cargo do BD que gera um timestamp no momento da inserção no banco.

O MBED realiza a captura de uma FO em um intervalo de tempo de 16,6 ms, a cada ciclo de análise, o valor do *Root Mean Square* (RMS – Valor Médio Quadrático) é calculado e comparado com o seu limite definido, caso o valor atual for maior que o definido, então este evento será contabilizado. Há um contador neste processo, para que, falsos positivos sejam detectados, minimizando a detecção de variações normais da corrente alternada, por exemplo, se em apenas um ciclo o RMS de uma FO for maior que o definido. Quando o contador atingir seu limite, um evento significativo será gerado e enviado ao servidor, gravando o registro no BD [3].

Na versão atual, o software de apoio comunica com o BD em um intervalo de tempo de 1 s, em busca de eventos significativos que, se encontrado, ocorre o processo de reconstrução da FO, através da Transformada

Discreta de Fourrier (DFT-1), analisando a periculosidade, considerando o RMS definido, e após a corrente por frequência. Em seguida ocorre uma busca no BD para buscar qualquer registro simultâneo entre a FO [3].

As escalas de periculosidades usadas no Protegemed são importantes para identificar se um evento relativo ao micro choque pode representar risco real ao paciente ou não. Nas escalas de periculosidade com base nos valores de corrente, quanto maior o valor, maior a periculosidade. Os micros choques representam potencial de risco proporcional ao valor da corrente e mesmo valores baixos podem ser nocivos, uma vez que durante procedimentos cirúrgicos a proteção natural da pele não pode ser totalmente considerada.

As escalas de periculosidade possuem três classificações da escala: Normal, Atenção e Perigo. Na escala com base no valor nominal da corrente, valores até 0,1 mA são considerados na faixa 'Normal'. Valores de corrente maiores que esse e menores que 0,5 mA ficam na faixa de 'Atenção' e valores maiores ou iguais a 0,5 mA são considerados na faixa de 'Perigo', devendo receber atenção imediata da equipe de engenharia clínica [3].

A escala de periculosidade com base no valor de cada harmônica do espectro de frequência também avalia o valor da corrente, porém apenas a corrente consumida em cada uma das 12 primeiras harmônicas. Com uma frequência base de 60 Hz, são consideradas frequências até 720 Hz. Caso seja usada uma frequência base de 50 Hz, são consideradas frequências de até 600 Hz. A fundamentação que sustenta essa escala é que, correntes em maior frequência podem ter valores maiores de corrente por frequência, porém não indiscriminadamente, uma vez que a corrente age no corpo humano de forma diferente. No trabalho [3], Tabela 6 são mostrados os limites de corrente por frequência para faixas de periculosidade 'Normal', 'Atenção' e 'Perigo'.

Quando uma corrente elétrica indesejável for detectada, torna-se necessário avaliar mais informações sobre esta corrente, a FO da corrente diferencial é uma dessas informações obtidas dos EEM ligados ao paciente durante o procedimento cirúrgico. Caso as FO dos EEM apresentem alta similaridade, existe a possibilidade de que o paciente esteja recebendo parte desta corrente [3].

Rebonatto [3] relata as formas de calcular a similaridade entre ondas como, o Coeficiente de *Spearman*, Coeficiente de *phi*, Distância de *Bhattacharyya*, Coeficiente de *Pearson* e o *Root Mean Square Deviation*. A Correlação de *Pearson* é usada para calcular a relação linear entre duas variáveis. O Coeficiente de *phi* foi proposto por *Karl Pearson*, como o Coeficiente de *Pearson*, sendo usado para mensurar a variação entre duas variáveis binárias (Sim/Não, Morto/Vivo, Verdadeiro/Falso). A Distância de *Bhattacharyya* é uma medida de semelhança entre duas distribuições, pode ser usada para o conhecimento de padrões, determinar a proximidade de duas amostrar, medida de divergência entre variáveis, entre outras [3]. O grau de associação entre duas variáveis é determinado pelo Coeficiente de *Spearman*. O Desvio Quadrático Médio (RMSD) é utilizado para calcular a diferença entre os valores estimados e os valores reais [3].

A escala de periculosidade é definida a partir de uma escala de similaridade entre FO de corrente elétrica. Os métodos de *Pearson* e *Spearman* foram os que mostraram os melhores resultados, ao final de todos os testes, o método de *Spearman* obteve um resultado melhor na média, caso a funcionalidade seja embarcada. A Tabela 1 apresenta a proposta da escala de similaridade para as FO de corrente elétrica [3].

Tabela 1: Escala de Similaridade entre as FO.

| **Periculosidade** | **Similaridade** | **Intervalo** |
|---|---|---|
| Perigo | Máxima | [1,000; 1,000] |
| | Alta | [0,950;0,999] |
| Atenção | Média | [0,850; 0,949] |
| Normal | Baixa | [0,500; 0,849] |
| | Mínima | [0,001; 0,499] |

Adaptado de Rebonatto [3]**.**

Os valores da coluna Intervalo determina o grau de similaridade para a FO, que podem variar numa escala de cinco níveis possíveis, os valores são obtidos a partir do coeficiente em módulo, podendo ser utilizada para cálculos baseados nas escalas de *Spearman* e *Pearson*, considerando o deslocamento no tempo de uma das ondas [3].

## 3 Processamento de eventos complexos

O processamento de eventos complexos contempla a demanda de novos desafios, que exigem tomadas de decisões rápidas sobre uma mudança detectada a partir da análise dos dados. Ele possibilita a análise de dados em tempo próximo do real, em menor tempo se comparado ao atraso do processo tradicional, onde é usado o processamento a partir de BD. CEP permite análise e processamento contínuo de um alto volume de dados, permitindo o relacionamento de eventos a partir de diferentes fontes de dados [4].

Um evento pode ser definido como um registro de uma atividade em um sistema computacional, para fins de processamento ou uma ocorrência de uma atividade relevante dentro de um período de tempo. Isso pode indicar que o estado de domínio de uma aplicação sofreu uma alteração [4].

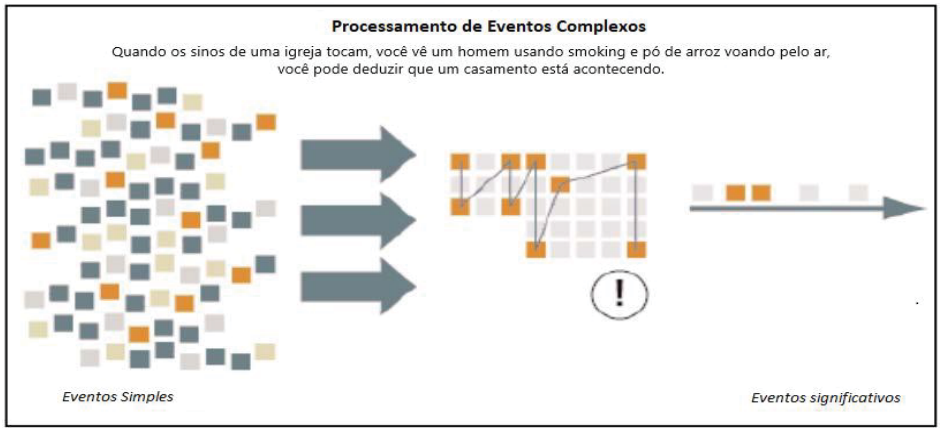
Os eventos auxiliam na detecção de comportamentos da aplicação, sendo benéficos para situações onde a ocorrência de alguma severidade é detectada e necessita receber imediatamente a intervenção humana ou de máquina. Os eventos dividem-se em dois tipos: eventos simples e eventos complexos. Neste trabalho vamos usar a letra e minúsculo para identificar eventos simples e maiúsculo para identificar eventos complexos.

Um evento simples é um registro atômico, uma ocorrência no sistema que pode ser vista de forma isolada. Pode ser representado por um conjunto de atributos e = (id, x, t), onde id é um identificador único, x = {$x_1$, $x_2$, … $x_n$} com n > 0, um conjunto de atributos e t é o tempo da ocorrência do evento [4]. Como exemplo, pode-se citar uma compra em um sistema online, usando o cartão de crédito como forma de pagamento. Sendo a compra representada por um identificador único, neste caso o id, x contém os atributos da compra como, quantidade de itens, valor total da compra, forma de pagamento entre outros, e t é representado pela data em que a compra ocorreu. De forma análoga, no Protegemed, um evento possui um identificador único, representado por seu código e possui diversos atributos (x), como o identificador do equipamento, o tipo de evento liga/desliga para corrente de fase ou início/fim de fuga para corrente diferencial, i valor e características da corrente. O t, nesse caso será a data e hora da ocorrência do evento.

Um evento complexo é composto por um conjunto de eventos: simples ou por outros eventos complexos, utilizando construtores como disjunção, conjunção e sequência [4]. Construtores ou operadores de eventos são utilizados para compor essa relação entre os eventos, gerando informações significativas. Eles podem representar relações lógicas, de tempo ou espaço [5]. Como exemplo de relação de espaço e tempo, podemos considerar não sendo possível uma compra por um mesmo usuário, em dois estabelecimentos, efetuada fisicamente com cartão de crédito, dentro de um intervalo de 5 minutos, sendo a distância entre elas é de 2000 km. As relações lógicas podem expressar uma dependência de existência entre os eventos, como no exemplo acima, para garantir que a compra não seja possível, é necessário além de avaliar o tempo entre as compras e a distância em que elas ocorreram.

Um evento complexo pode ser representado por um conjunto de atributos E = (id, x, c, ti, tf), tf >= ti, onde id é um identificador único, x = {x1, x2, … xn}, onde n > 0, um conjunto de atributos, c {e1, e2, y, en}, n > 0, é um vetor que contém os eventos simples ou complexos, ti e tf representam o início e o fim em tempo, ou seja, a duração de um evento complexo [4]. A Figura 1 ilustra a representação de processamento de eventos complexos a partir de eventos simples.

Figura 1: Funcionamento do CEP, adaptado de Amaral [5], 2011.



Na Figura 1 pode-se identificar uma grande quantidade e diferentes tipos de eventos simples que podem ser recebidos pelo motor CEP, representados pelos pequenos quadrados à esquerda. Então, eles são analisados e relacionados baseados em regras previamente definidas. Neste caso, esta etapa atua como um filtro, produzindo apenas os eventos que representem significado, baseado nas condições de aceitação das regras. A partir desse relacionamento é que podem ser construídos eventos complexos, com informação significativa, representados pelos eventos que estão na flecha de saída, a direita.

O Protegemed terá eventos simples e complexos, sendo exemplo de evento complexo a detecção de periculosidade de eventos por similaridade, ou seja, a ocorrência de dois eventos de fuga de corrente, em que no instante i0 é detectado um evento de início de fuga em uma tomada x e no tempo i1 é detectado na tomada y o início de outro evento de início de fuga, sem ocorrer um evento de final de fuga na tomada x, além da restrição de que tais eventos precisam ser oriundos na mesma sala. Neste momento, está caracterizado um evento complexo com a ocorrência simultânea de fugas de correntes.

### 3.1 Motores CEP

A tarefa para realizar o processamento de eventos, demanda de softwares com alta tecnologia e bom poder de processamento, visto que em um ambiente CEP, o volume de dados é um fator importante e crítico em que a aplicação deve suportar tal demanda [6].

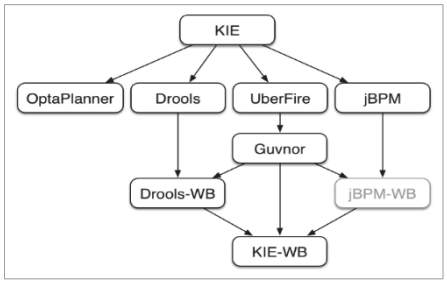
Hoje no mercado há uma diversidade de frameworks que atendem as necessidades do ramo empresarial para realizar o processamento de eventos complexos, entre eles está o Apache Flink, WSO2 Complex Event Processing, JRule Engine, Oracle CEP e o Drools.

O Apache Flink é uma plataforma *open source*, usada para processamento de fluxo de dados distribuídos, é altamente escalável e tolerante a falhas. Os programas podem ser desenvolvidos em Java, SQL, Python e Scala [7]. O WSO2 Complex Event Processing é uma ferramenta open source e trabalha com a linguagem SQL-like para realizar consultas, permite a manipulação dos dados tanto em memória, quanto em banco de dados relacional, possui suporte para processamento paralelo [8]. O JRule Engine é um motor de regras CEP baseado no Requisito de Especificação Java 94, release 1.1. As regras podem ser carregadas por um arquivo XML, ou através de APIs do próprio JRule Engine, dessa forma, as regras podem ser armazenadas externamente, em um banco de dados [9]. O Oracle CEP é chamado de Weblogic Event Server, servidor Java orientado a eventos, possui o Service Engine Oracle CEP, um ambiente rico e declarativo baseado no Oracle Continuous Query Language (Oracle CQL), linguagem de consulta baseada em SQL [10]. O Drools é um Sistema de Gerenciamento de Regras de Negócio (BRMS). A ferramenta realiza o encadeamento e relação de eventos de entrada, baseando-se no seu motor de regras para gerar uma saída. Drools é um projeto open source desenvolvido pela comunidade JBoss.org, e mantido pela empresa Red Hat [6]. Ele utiliza a linguagem Java como desenvolvimento, muito difundida no meio de desenvolvimento de aplicações, além de suportar o processamento de regras temporais necessárias ao Protegemed. Este framework já foi utilizado em alguns projetos no grupo de pesquisa ComPaDi, com os ex-alunos Lange [11] e Júnior [12].

### 3.2 Conhecimento é tudo.

O *Knowledge is Everything* (KIE – Conhecimento é tudo) é um projeto open source para automação e gerenciamento de processos empresariais. O projeto foi criado para abrigar e unificar diversas tecnologias como o Business Process Management (jBPM – Gestão de Processos de Negócios), OptaPlanner, Drools, entre outras. Muitos dos recursos e implementações presentes no KIE, posteriormente são acrescentados na versão empresarial do Red Hat JBoss BRMS [6]. Na Figura 2 pode-se visualizar a estrutura do projeto.

Figura 2: Estruturação do projeto KIE.



As tecnologias relacionadas funcionam de forma independente, mas compartilham os recursos entre si. O jBPM é usado para modelar processos de negócios através de fluxogramas, dando maior visibilidade e agilidade para a lógica de negócio [17]. O OptaPlanner auxilia na otimização do planejamento para realizar maior volume de negócios com menos recursos [18], enquanto o UberFire é uma plataforma para ajudar na rápida criação de aplicações para banco de dados ou aplicações console para a web [19]. O Drools é uma plataforma que, unifica as regras de fluxos de trabalho e o processamento de eventos complexos [6].

O Drools é um módulo presente no KIE, com foco no processamento de eventos complexos, criado e mantido pela Red Hat, que conta com a participação colaborativa da comunidade JBoss.org, contribuindo nas diversas melhorias e implementações a cada nova versão da ferramenta [6].

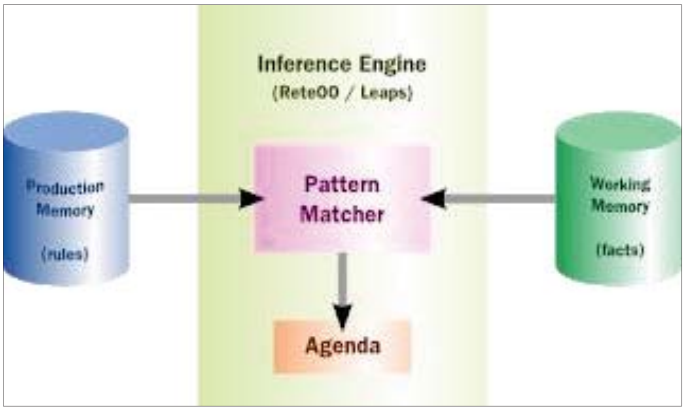
Um sistema BRMS é um software para definir, implantar, executar e monitorar regras de negócios, é um conceito muito difundido no setor empresarial. Drools é uma solução BRMS, que pode ser usada tanto para o desenvolvimento das regras de negócio, como o processamento de eventos complexos, suporta o desenvolvimento através de IDE's, suporte nativo à linguagem Java, disponibiliza uma interface web, onde pode ser realizado a modelagem de regras para o processamento de eventos complexos.

A plataforma apresenta diversas características, como suporte intrínseco a regras temporais, visto que é um requisito essencial em sistemas CEP, e uma das características que distingue eventos é a forte relação temporal, baseado no intervalo com duração de no mínimo 0 (zero), além de suportar a ausência de eventos, (ex. Eventos de negação). Suporte a múltiplos fluxos assíncronos de dados, por onde os eventos são recebidos a partir das diversas fontes de dados.

O Drools, encontra-se na versão estável 7.14, disponibilizada em 8 de novembro de 2018, e conta com a disponibilização de atualizações regulares, através de um canal do site Jboss. A mudança com maior impacto desde o seu lançamento, ocorreu na versão 5.6 para a 6.0, onde o projeto sofreu uma unificação, anteriormente, os componentes eram vistos de forma isolada e independente, mas a partir da versão 6.0, passou a ser unificada pelo projeto KIE.

Neste projeto temos muitos recursos importantes para o desenvolvimento de uma aplicação CEP, dentre eles podemos destacar alguns módulos como o Drools Workbench, o Drools Expert e o Drools Fusion. O Drools Workbench é uma ferramenta baseada em uma interface *web*, onde é possível criar regras, modelar eventos complexos, gerenciar repositórios, realizar o gerenciamento de usuários e grupos, modelar entrada de dados usados para validação das regras ou definir fontes de dados, como conexão a um banco de dados [6]. O Drools Fusion é o módulo responsável por adicionar capacidade de processamento de eventos na plataforma, possui um conjunto de definições para realizar o processamento de eventos complexos de forma adequada, algumas delas: realizar a detecção, correlação, agregação e a composição de eventos, apoiar restrições temporais para modelar as relações temporais entre eventos, suportar o volume de necessário de eventos, escopo de sessão de relógio unificada [20]. O Drools Expert é um motor de regras de inferência, o qual é baseado no algoritmo de *Rete*, chamado de *ReteOO* (*Rete Oriented Object*). As regras de produção são armazenadas na *Production Memory*, e os fatos (dados) são armazenados no *Working Memory* [21]. *Pattern Matcher* é a execução de regras para processamento de eventos, sobre os fatos novos ou existentes. A Figura 3 representa o funcionamento abstrato o motor de regras de inferência.

Figura 3: Funcionamento do sistema de regras de inferência do Drools Expert.




Os fatos são confrontados com o *Working Memory*, que em determinadas situações podem disparar várias regras para o mesmo fato, deixando-as em conflito. A *Agenda* é um componente responsável pela resolução de conflitos e controla a ordem dessas execuções [21].

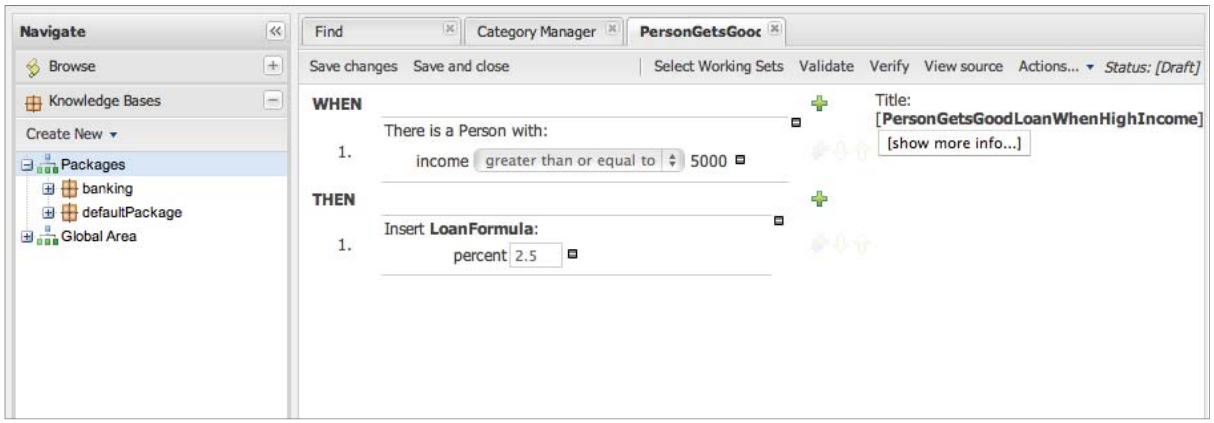
Segundo a JBoss [6], uma aplicação CEP compartilha de várias características comuns e distintas, baseado neste conceito, o Drools fornece uma série de recursos, que atende à necessidade para diferentes problemas, como a semântica de eventos e dois modos de processamento de eventos.

Na semântica de eventos, estes normalmente representam fatos imutáveis, um registro de algo que já aconteceu, porém, o motor não impõe imutabilidade ao modelo de objetos, uma vez que o uso mais comum é o enriquecimento de dados de eventos. Fortes restrições temporais, onde normalmente exigem a correlação de um evento com múltiplos eventos, determinando assim, um acontecimento em determinado ponto de tempo. Os modos de processamento de eventos são dois. O modo nuvem, onde não há noção de fluxo de tempo, embora cada evento tenha um *timestamp*, não existe conceito de "agora". No modo fluxo, a ordenação dos eventos pelo tempo deve ser realizada, assim, o motor entende o conceito de agora, baseado na leitura atual do timestamp da *Session Clock*. Estes recursos estão presentes visto que o mecanismo de regras não impõe restrições como os dados devem ser apresentados, a não ser quando o tempo real ou quase real, tornasse variável importante do processo.

### 3.3 Criando regras

Guvnor é módulo com uma interface *web* presente no Drools até a versão 5.6, possui funcionalidades como: desenvolvimento e gerência de regras de negócio, gerenciamento de usuários e grupos, entre outros. Ele é recomendado quando há necessidade de vários usuários editarem as regras de negócios, como analistas de negócios, desenvolvedores de regras e administradores. Por abranger muitos recursos, a partir da versão 6.0, criou-se o módulo Drools Workbench, onde foram portados todos os recursos no que diz respeito ao desenvolvimento para processamento de eventos complexos, o Guvnor ainda pode ser instalado e usado com o mesmo objetivo, porém sua manutenção para desenvolvimento de regras para eventos foi descontinuada. A Figura 4 mostra a interface disponível via web para a manipulação e criação de regras.

Figura 4: Modelagem de uma regra de processamento de evento complexo pela interface do Guvnor.




A ferramenta web possui um recurso chamado de editor guiado, onde é possível criar as regras para que sejam disparadas sobre determinadas entradas de eventos, a partir da restrição de determinado valor definido nos atributos de um evento complexo.

## 4 Trabalhos Relacionados.

O processamento de eventos complexos já vem sendo estudado desde a década de 90 na Universidade de Cambridge, no Instituto de Tecnologia da Califórnia e na Universidade de Stanford [13] e, desde então pode-se encontrar diversos trabalhos de experimentos com aplicações CEP em diversas áreas do conhecimento e em conjunto com outras tecnologias. Neste trabalho serão citadas algumas áreas de aplicação de CEP como, saúde, *Business Intelligent* (BI) e segurança de sistemas, sendo que a maior parte é voltada para a saúde, área de aplicabilidade do Protegemed.

Os primeiros quatro trabalhos abordam a aplicabilidade na área da saúde, Lange [11] propõe melhorias para o processamento de eventos, buscando garantias do tempo real de processamento. Yao [4] propôs um modelo de hospital inteligente, usando CEP para monitorar em tempo próximo do real, equipamentos, pacientes e médicos. Weber et al. [14] relata sobre a aplicabilidade do uso de CEP em uma emergência hospitalar e Meister [15] relaciona o uso da tecnologia na Telemedicina. Os dois trabalhos seguintes relacionam BI, onde Amaral [5],

demonstra que o uso de CEP possibilitou a automatização de determinadas tarefas, elevando o grau de confiabilidade do processo operacional. Palmer [13] destaca que as organizações necessitam monitorar, analisar e agir sobre determinadas condições de mudanças comerciais. Para finalizar, Lars et al. [16] propõe uma nova abordagem de combater e detectar ataques em sistema computacionais através de CEP.

Lange [11] em seu trabalho abordou o processamento de eventos com Drools relacionado aos desafios baseados em tempo real. O tempo real é uma mensuração realizada através de um dispositivo físico, ou seja, quantitativa. Em software se utiliza a noção de tempo qualitativa (quando, depois, antes). O modelo de tarefas para CEP de Tempo Real apresentado por Lange [11], identificado como CEPRT, procura processar eventos de monitoramento de sinais vitais de um paciente, porém, com modificações em relação ao modelo tradicional CEP. O CEPRT é a execução do CEP, adicionado de uma fila de prioridades para os eventos. Lange [11] cita ainda a possibilidade de realizar a ordenação desses eventos, definindo a importância de cada um, sendo possível processar os eventos dentro do motor de regras, seguindo a prioridade dada para cada evento.

Yao [4], utiliza um sistema de *Radio Frequency Identification* (RFID – Identificação por radiofrequência) para a tarefa de processamento de eventos complexos em conjunto com a plataforma Drools, com foco em um ambiente hospitalar. Ele descreve um hospital como um ambiente grande, muito ocupado e ao mesmo tempo caótico, onde diariamente médicos cuidam e salvam vidas, pacientes são internados, recebem altas, são movimentados para ambulatórios diferentes, sempre sob cuidados técnicos especializados, mas não monitorados em tempo real. Este cenário é muito motivador para o uso de sistemas RFID com CEP, auxiliando na identificação e localização de equipamentos, pacientes e médicos. O trabalho de Yao propõe um ambiente hospitalar inteligente, onde pacientes, médicos e equipamentos recebem tags RFID. O CEP foi introduzido para processar e correlacionar dados gerados pelo RFID, que a partir de múltiplos fluxos, pode extrair eventos simples em uma grande quantidade de dados, e correlacioná-los para gerar eventos complexos.

Weber et al. [14] descreve a utilização do processamento de eventos complexos em uma emergência hospitalar, para suportar as necessidades de avaliação clínica e de translado dos pacientes, implantados em duas áreas de atendimento. Primeiro, na triagem de pacientes que sofreram mordida de cães, a fim de identificar níveis de riscos de infecção, e assim ter um melhor direcionamento para os pacientes que necessitem de tratamento mais urgente. Segundo, em um cenário mais complexo, estuda a viabilidade de identificar o tamanho da amostra e a capacidade de identificar pacientes com lesões cerebrais traumáticas na anticoagulação. Os dados foram obtidos a partir de diversas fontes como, preenchimento de formulários, resultado de testes laboratoriais, exames radiológicos, sendo processados em tempo próximo do real.

Meister [15] aborda o uso da Telemedicina como forma de superar alguns problemas na assistência da saúde em áreas rurais, diferenças demográficas e falta de estrutura econômica. Ao mesmo tempo em que a Telemedicina pode resultar em um novo problema: a geração excessiva de informação. Para estes desafios, foi feito o uso de ILOG (Sistema de gerenciamento de regras de negócio) com CEP. ILOG atua como um filtro de informações, evitando a geração de informação desnecessária, só então faz envio para o CEP. Os testes são aplicados no monitoramento de pacientes com insuficiência cardíaca através da telemetria, considerando dois tipos de eventos, peso e pressão sanguínea. O aumento de peso pode ser um indicador para a retenção de água nos pulmões. O aumento significativo da pressão sanguínea, especialmente combinada ao ganho de peso, pode representar insuficiência cardíaca grave.

Amaral [5] na sua tese de doutorado utiliza sistemas de *middleware* RFID com CEP para realizar o processamento de eventos oriundos a partir de dispositivos embarcados, uma vez que o *middleware* RFID dispõem de recursos básicos para gerenciamento de dados, seguindo as especificações *Application Level Event Standart*, que são eventos no nível da aplicação, e não suporta o tratamento para eventos complexos.

Palmer [13] aborda a importância do CEP para a tomada de decisão inteligente nas organizações como, no uso de algoritmos de negociações comerciais, rastreamento de logística e controle de estoque. Diferenças entre as tecnologias CEP e *Event Stream Processing*, ferramentas destinadas a usuários finais, com dashboard, gráficos, como o *Business Activity Monitoring*. *Data Stream Processing*, onde os dados são armazenados na ordem cronológica em que ocorrem, sendo úteis para o desenvolvimento de futuros novos cenários.

Lars et al. [16] expõe a crescente dificuldade encontrada por softwares do tipo *Intrusion Detection System* e *Intrusion Prevention System* em analisar e prevenir ataques ou invasões cada vez mais complexos em sistemas computacionais. O trabalho propõe a implementação do ACCEPT (*Anomaly Management in Computer Systems Through Complex Event Processing Technology*) em um ambiente virtualizado, onde é utilizado CEP para realizar a agregação dos eventos monitorados. Os logs gerados pelas aplicações em execução na (s) máquina (s) virtual

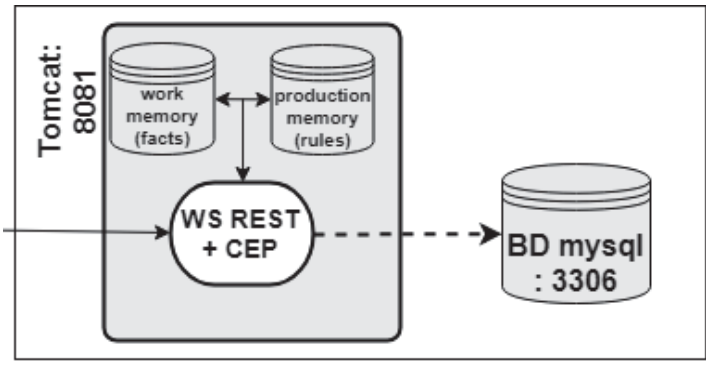
(is) é processado em uma máquina virtual separada, chamada de ACCEPT-VM, que a partir da análise, se necessário, poderá desencadear ações para que anomalias de segurança sejam interrompidas.

A partir do estudo dos trabalhos citados acima, pode-se constatar diversas vantagens na utilização de CEP nas várias áreas em que foi empregado, já que em todas elas, a tecnologia em questão agregou mais confiabilidade e segurança no processo, principalmente os trabalhos relacionados a área da saúde, mesmo nicho de atuação do Protegemed. Isto é um fator de motivação para adotar esta abordagem, visto que, análises preliminares indicam que CEP poderá trazer melhor segurança elétrica para procedimentos cirúrgicos.

## 5 Detecção de periculosidade usando cep.

A ferramenta Drools contempla as necessidades requisitadas para a detecção e quantificação para o projeto, além da modelagem das regras para cálculos numéricos, a ferramenta suporta o processamento de regras temporais, recurso indisponível na versão anterior do Protegemed, porém, necessário. Antes de implementar uma solução para detecção de periculosidade de eventos do Protegemed usando CEP, um modelo para detecção foi desenvolvido. As informações dos eventos serão recebidas e tratadas com a adição de um motor CEP. Na figura 5 é possível identificar o modelo proposto.

Figura 5: Diagrama principal para o tratamento de eventos à nova versão.




Os eventos recebidos são inseridos no BD e no motor CEP, que por sua vez acessa o working memory e o production memory para realizar o match das regras com os fatos, somente atualizando as informações no BD quando necessário.

Eventos recebidos pela são inferidos pelas regras desenvolvidas. São previstos 5 tipos de regras: detecção de eventos órfãos, cálculo da periculosidade da corrente, periculosidade da corrente por frequência, periculosidade por similaridade de FO, detecção de similaridade entre eventos concorrentes e término de um evento. Para quantificar as regras de periculosidade, foram utilizados apenas eventos do tipo 1, início de fuga de corrente e tipo 6, final de fuga de corrente. Os valores de periculosidade são os definidos na seção 2.

Para eventos órfãos, foram considerados registros sem o seu "par" correspondente com início ou fim de fuga de corrente ou alimentação (fase). Nestes casos, o valor para a duração será -1, além da remoção do registro errôneo da base do Drools. Esse tratamento se fez necessário, uma vez que foi identificado na aplicação anterior que os MBED's enviavam o mesmo tipo de evento oriundo da mesma tomada e sala consecutivas vezes.

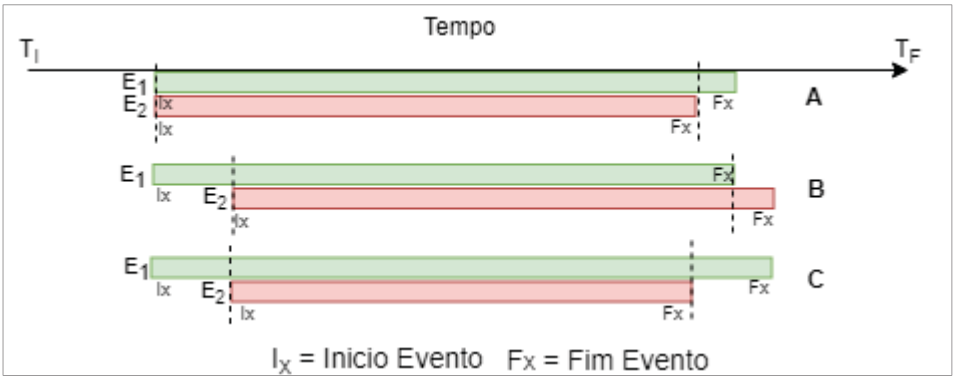
O cálculo da periculosidade por corrente avalia o valor eficaz, que é comparado com os limites citados em [3]. O cálculo da periculosidade de corrente por frequência faz a inferência da periculosidade a partir da decomposição das doze primeiras harmônicas, onde gerará um valor, comparado então com os limites definidos em[3]. Qualquer um dos valores que representar um grau 3 ("perigo") interrompe as demais avaliações, pois uma situação de perigo já foi identificada.

O cálculo de periculosidade por similaridade avalia dois eventos através da Correlação de *Spearman*, que resulta num coeficiente de correlação entre as formas de onda de dois eventos relacionados. A regra para detecção de eventos concorrentes com similaridade restringe que devam ser com o tipo 1, ocorrer em tomadas diferentes, estejam na mesma sala e inclui a restrição temporal, ou seja, este evento é caracterizado do tipo complexo. A figura 6 ilustra as restrições para sua detecção.

Na relação identificada com a letra A o Evento 1 ($E_1$) tem Tempo Inicial ($I_x$) igual ao do Evento 2 ($E_2$), indicando que iniciam no mesmo instante, essa é a única situação que considera a ocorrência de uma fuga de

corrente concomitante em [3]. Neste caso, a análise da similaridade das formas de onda para calcular a periculosidade dos eventos de fuga se faz necessária. Na relação identificada pela letra B, mesmo que o $E_2$ tem o seu $F_x$ maior que o $F_x$ do $E_1$, o $E_2$ tem o $I_x$ antes do $F_x$ do $E_1$, durante esse intervalo de tempo em que a sobreposição dos eventos, ocorre uma fuga de corrente concomitante. A mesma relação de concorrência é identificada na letra C o $E_1$ tem o $I_x$ antes do $E_2$, o $E_2$ tem seu $I_x$ antes do $F_x$ do $F_1$. As linhas verticais em pontilhado identificam a duração do total do evento complexo. As situações identificadas pelas letras B e C foram possibilitados de detecção com a adição do motor CEP.

Figura 6: Detecção de similaridade entre duas fugas de correntes, caracterizando um evento do tipo complexo no Protegemed.

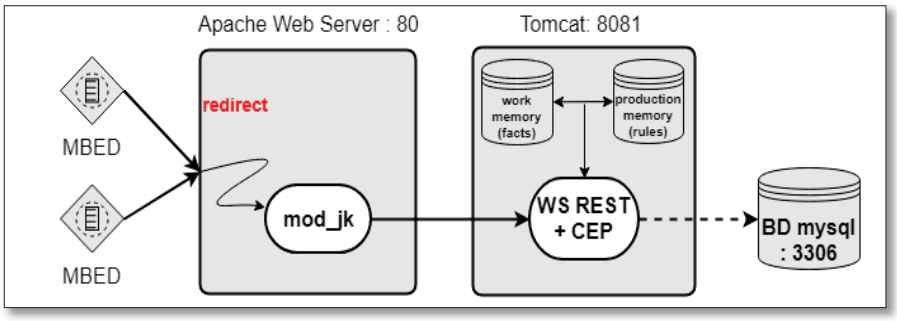



Por fim, foi desenvolvido as regras para detectar o final dos eventos, responsável por identificar eventos do tipo 5 e 6 depois da ocorrência de eventos 4 e 1, correlacionando com a sala e tomada de ocorrência, calculando assim a duração total dos eventos com periculosidade, adicionalmente, foi utilizado a informação do fim do evento de alimentação de um equipamento (fase) para calcular o tempo em que o equipamento ficou ligado. Após a detecção as regras farão a remoção do *working memory*, finalizando o ciclo do evento. No apêndice I é possível verificar o código fonte de todas as regras desenvolvidas.

## 6 Implementação da detecção de periculosidade

O modelo de implementação de periculosidade com CEP substituirá parte da aplicação atual que é executada em código PHP, a funcionalidade da versão anterior era capturar os eventos recebidos pelo servidor Apache, inseri-los no BD e atualizar o tempo de uso dos eventos quando era possível determiná-los. A nova aplicação contempla este recebimento acrescido do Drools que irá possibilitar a análise de periculosidade dos eventos em tempo próximo do real. Para atender as novas necessidades optou-se pelas seguintes tecnologias, linguagem Java, Drools, web service desenvolvido no padrão *A Representational State Transfer* (Transferência de Estado Representacional – REST) e no servidor de aplicação Tomcat, onde será executado a aplicação. Na Figura 7 está representada a estrutura da aplicação desenvolvida.

Figura 7: Diagrama do funcionamento com a alteração proposta.




Pode-se observar que a compatibilidade entre a comunicação final dos MBED's, responsáveis por enviar os eventos e o motor Drools permanece de forma similar como na versão anterior. Isto foi viável com a adição do componente *mod_jk*, presente no Apache *Web Server*, que fará todos os redirect's solicitados para a aplicação protegemed, que está presente no Tomcat.

Foi implementado web service REST, que disponibiliza ações em métodos HTTP [22] sendo que neste trabalho será usado apenas o método POST, uma vez que é o método utilizado na comunicação entre os dispositivos MBED's e a interface PHP.

O fluxo de execução da aplicação com CEP terá início no recebimento dos dados pelo servidor Apache, porta default 80, redirecionando para a porta 8081 do Tomcat, esse redirect é feito pelo componente *mod_jk* presente no Apache. O *web service* REST irá receber a requisição, instanciando um objeto do tipo CapturaAtual (classe Java), que corresponde a um evento a ser inserido no *working memory*. As informações de valor eficaz, ganho, valor médio, seno e cosseno são enviadas no formato hexadecimal, sendo necessário realizar a conversão para o tipo ponto flutuantes. Para cada evento recebido, haverá uma ação do motor CEP, disparando todas as regras presentes no *production memory* sobre todos os fatos presentes no *working memory.*

A modelagem das regras de inferência será implementada no formato *Domain Rule Language* (Linguagem de Regra de Domínio – DRL). As DRL's conterão as restrições para a identificação das escalas de periculosidade da corrente, frequência, por similaridade, detecção de eventos periculosos em janelas de tempos simultâneas, além dos eventos órfãos e finais de eventos do tipo 5 e 6. Os eventos detectados a partir desse conjunto de regras, serão armazenados no banco de dados. A figura 8 mostra a sintaxe de escrita para regras no formato DRL usado em uma das regras, as demais constam no Apêndice I.

Figura 8: Regra para detectar eventos de similaridade concorrentes.

```
rule "rule-detected-similaridade-1"
    salience 3 //regra será disparada antes da relação de início e fuga.
    when
        $eventOneBefore : CapturaAtual( getEventos().getCodEvento() == 1)
        $eventOneAfter : CapturaAtual( getEventos().getCodEvento() == 1
                    && $eventOneBefore.getTomada().getCodTomada() != getTomada().getCodTomada()
                    && $eventOneBefore.getSalaCirurgia().getCodSala() == $eventOneAfter.getSalaCirurgia().getCodSala()
                )
        $outlet : CapturaAtual(
                    this.getEventos().getCodEvento() == 6
                    && this.getTomada().getCodTomada() != $eventOneBefore.getTomada().getCodTomada()
                    && this.getTomada().getCodTomada() == $eventOneAfter.getTomada().getCodTomada()
                    && this.getSalaCirurgia().getCodSala() == $eventOneAfter.getSalaCirurgia().getCodSala()
                    ,this after[0s] $eventOneAfter && $eventOneAfter after[0s] $eventOneBefore
                )
    then
        logger = Logger.getLogger(CapturaAtual.class);
        logger.info(drools.getRule().getName());
        new CapturaAtualDAO().updateSimilaridade($eventOneBefore, $eventOneAfter);
end;
```

Um dos principais recursos disponíveis no Drools são os operadores temporais, usados para determinar a relação de tempo entre eventos. Nesta regra, foi usado o operador after, utilizando como condição 0s, mesmo que um evento comece depois ou no mesmo instante que outro, a regra será disparada. Fez-se necessário o uso do recurso *salience*, possibilitando definir prioridade de execução entre as regras, já que há a necessidade de que a regra de detecção de órfãos por exemplo, execute antes de todas as outras. No Apêndice II consta um diagrama com a ordem de prioridade de todas as regras.

Houve a necessidade de alterações no banco de dados como, criação de tabelas para armazenar os limites das periculosidades por frequência, corrente e similaridade, bem como tabela para armazenar futuras alterações nas versões desses limites, permitindo rastreabilidade e histórico, novas informações como o valor das periculosidades passaram a ser armazenadas em novas colunas da tabela capturaatual.

# 7 Resultados

## 7.1 Descrições dos eventos analisados

Para fins de análises e testes reais, foram obtidos dados em uma base com 493.052 registros entre os meses setembro de 2014 e abril de 2018, deste total foram considerados 493.052 registros, entre os meses de abril de 2015 e maio de 2016, essa restrição por um período específico foi considerada em virtude da uniformidade dos dados presente nestes meses, eliminando lacunas, contabilizando um período de 14 meses. Os registros foram formatados e desenvolvidos scripts para então serem enviados a aplicação CEP, desta forma, passariam pela análise das regras.

Dentre os 14 meses buscados, temos a seguinte distribuição por tipo de evento identificados na tabela 1, pode-se observar que a maior parte são eventos 4 e 5, que identificam respectivamente o início da fase, ou seja, quando o equipamento é ligado e eventos do tipo e fim de fase, quando o equipamento é desligado. Foi realizado uma média no período de 14 meses de análise, baseado nos valores da tabela 1, em média, 34.943 eventos serão recebidos pelo motor CEP.

Tabela 1: Distribuição dos eventos por tipo.

| Evento | Total | % | Media mês |
|---|---|---|---|
| 1 | 11.133 | 2,26% | 795 |
| 4 | 317.388 | 64,37% | 22.671 |
| 5 | 160.556 | 32,56% | 11.468 |
| 6 | 3.975 | 0,81% | 284 |

Os valores para a periculosidade da corrente por frequência e periculosidade por corrente foram calculados com CEP, obteve-se os mesmos resultados definidos em [3].

A partir da análise dos dados da tabela 1 é possível verificar que os eventos com maior ocorrência são aqueles que indicam o início de um evento, independente qual seja o seu tipo, ou seja, os eventos do tipo 1 e 4. Nota-se também que os eventos de alimentação (1) superam em muito os eventos de fuga de corrente (6). Além disso, pela disparidade dos pares de eventos 1-6 e 4-5, existem muitos eventos órfãos. A distribuição agrupada por tipo de evento e detalhado mês a mês está no Apêndice III.

### 7.2 Processamento de eventos órfãos

A partir da existência de eventos órfãos detectados, foi buscado quantificar os mesmos. A Tabela 2 contém a distribuição dos eventos órfãos por tipo de evento. No apêndice IV consta a relação detalhada de todos os órfãos.

Tabela 2: Quantidade total de registros órfãos com seu respectivo percentual.

| Evento | Total | Órfãos | % |
|---|---|---|---|
| 1 | 11.133 | 7.159 | 64,30% |
| 4 | 317.388 | 156.841 | 49,42% |
| 5 | 160.556 | 9 | 0,006% |
| 6 | 3.975 | 1 | 0,025% |

A possibilidade de realizar o tratamento de eventos órfãos com a adição do motor CEP é importante pois, eventos do tipo 1 indicam o início de uma fuga de corrente, conhecer o início correto do evento é essencial para determinar a sua duração quando o fim for detectado, ou possibilitar o disparo de algum alerta em casos que o paciente possa estar em risco.

### 7.3 Análises da periculosidade por similaridade de eventos concorrentes

A análise de quantidade de detecção de eventos ocorridos simultaneamente envolveu a comparação com a detecção desses mesmos eventos com a aplicação sem CEP com a nova versão utilizando motor CEP. Em [3] será considerado eventos simultâneos somente se dois ou mais eventos tiverem a mesma data inicial, restringindo a data até segundos, na nova versão com motor CEP, isto pode ser estendido com a precisão de *timestamp*, considerando milissegundos. Na tabela 3 é possível observar o aumento na detecção desse tipo de evento.

Tabela 3: Comparativo de detecção de casos de eventos simultâneos.

| | Casos detectados | Eventos | Percentual |
|---|---|---|---|
| **Sem CEP** | 426 | 852 | 7,65% |
| **Com CEP** | 2.003 | 4.006 | 35,98% |
| **Total** | - | 11.133 | |

Comparando os dois percentuais, nota-se que a aplicação CEP teve um aumento de 28,33% na detecção de eventos simultâneos em relação a versão anterior, considerando ainda que a aplicação com CEP detectou todos os eventos simultâneos. Isso pode ser explicado pela expansão da simultaneidade, descrita na Figura 6. A Tabela 4 ilustra uma distribuição de eventos simultâneos pela periculosidade calculada. No Apêndice V consta a distribuição mensal.

Tabela 4: Quantidade de casos de eventos simultâneos agrupados por escala de periculosidade.

| Periculosidade | Sem CEP | Com CEP | Aumento |
|---|---|---|---|
| **Normal** | 413 | 1.765 | 1.352 |
| **Atenção** | 4 | 177 | 173 |
| **Perigo** | 9 | 61 | 52 |

Na tabela 4 é possível perceber a importância da adição do motor CEP no Protegemed, na coluna "Aumento" pode-se observar o aumento na quantidade de eventos simultâneos detectados. Nota-se um grande aumento na quantidade de eventos com periculosidade Normal.

Os eventos com maior periculosidade encontrada foram destacados na Tabela 5, incluindo o valor de corrente de cada um deles.

Tabela 5: Relação de eventos concorrentes com maior similaridade.

| | Evento 1 | | | Evento 2 | | |
|---|---|---|---|---|---|---|
| **Similaridade** | **Código** | **Eficaz** | **Início** | **Código** | **Eficaz** | **Início** |
| 1,000 | 574367 | 0,97 | 14/03/2016 21:10:54 | 574371 | 0,93 | 14/03/2016 21:17:05 |
| 0,995 | 161535 | 0,21 | 12/05/2015 14:13:46 | 161536 | 0,24 | 12/05/2015 14:13:47 |
| 0,992 | 552946 | 0,23 | 10/03/2016 08:46:11 | 552947 | 1,89 | 10/03/2016 08:46:11 |
| 0,991 | 320273 | 0,24 | 20/10/2015 16:30:05 | 320275 | 0,23 | 20/10/2015 16:30:06 |
| 0,991 | 574390 | 0,28 | 14/03/2016 21:47:03 | 574394 | 0,41 | 14/03/2016 21:47:09 |

A partir da coluna "Início do Evento" é possível inferir que, na versão anterior do Protegemed, apenas os eventos com data inicial igual a 10/03/2016 08:46:11 seriam identificados como eventos concorrentes, uma vez que, o critério é ambos terem a data inicial iguais, ou seja, dos cinco eventos detectados pela aplicação com CEP,

A periculosidade da corrente é baseada no valor da eficaz. Na tabela 6 pode-se observar os cinco maiores eventos baseados no valor da eficaz.

Tabela 6: Eventos com maior valor de eficaz.

| | Evento 1 | | Evento 2 | | |
|---|---|---|---|---|---|
| **Valor Eficaz** | **Código** | **Início** | **Código** | **Início** | **Similaridade** |
| 1,90 | 552993 | 10/03/2016 09:37:26 | 552994 | 10/03/2016 09:37:26 | 0,965 |
| 1,89 | 552957 | 10/03/2016 08:46:32 | 552958 | 10/03/2016 08:46:32 | 0,968 |
| 1,89 | 552946 | 10/03/2016 08:46:11 | 552947 | 10/03/2016 08:46:11 | 0,992 |
| 1,87 | 552987 | 10/03/2016 09:37:02 | 552989 | 10/03/2016 09:37:02 | 0,990 |
| 1,87 | 552952 | 10/03/2016 08:46:28 | 552953 | 10/03/2016 08:46:28 | 0,959 |

Todos os eventos descritos na tabela 6 possuem relação de ocorrência simultânea com algum outro evento, isto pode ser observado na relação entre as colunas "Evento 1" e "Evento 2", pela coluna "Início" pode-se constatar que todos os eventos iniciam no mesmo instante, situação ilustrada na Figura 6, letra A.

## 8 Considerações finais

A adição do Drools, bem como seus recursos ao projeto Protegemed trouxe inúmeros benefícios, supriu as necessidades da detecção da periculosidade por similaridade entre ondas de fuga de corrente ocorrendo em momentos diferentes, mas concorrentemente, possibilitou identificar os eventos órfãos gerados pelos MBED'S, determinar a duração total de um evento, além de possibilitar a implementação do tempo total que um determinado equipamento permanece ligado. Vale destacar que um dos principais recursos do Drools que possibilitaram esses benefícios foi o uso de operadores temporais, recurso chave para uma aplicação CEP. Isso gerará uma confiabilidade maior sobre os eventos de riscos identificados, diminuirá o tempo de resposta em uma possível situação de risco para o paciente, pois, os eventos serão analisados pelo motor CEP em tempo próximo do real em que ocorrem.

A partir da adição do Drools como motor CEP ao Protegemed para gerenciar as regras de produção será possível expandir a aplicação deste modelo de detecção para outras necessidades identificadas futuramente. Os dados adquiridos podem passar por uma análise mais detalhada, pois possibilitará novas descobertas sobre os eventos, uma vez que o motor CEP agregou mais detalhes e identificou situações antes indetectáveis.

## Referências


[1] M. A. Schmitz, “Comunicação Bidirecional para Plataforma Embarcada no Protegemed. ”, *Dissertação de Mestrado*, Universidade de Passo Fundo, Passo Fundo, 2017.

[2] MBED, “MBED LPC 1768”. [Online]. Available at: https://developer.mbed.org/platforms/mbed-LPC1768. [Acessado: 12-out-2017].

[3] M. T. Rebonatto, “Métodos para análise de correntes elétricas de equipamentos eletromédicos em procedimentos cirúrgicos e detecção de periculosidade aos pacientes”, Porto Alegre, 2015.

[4] W. Yao, C.-H. Chu, e Z. Li, “Leveraging complex event processing for smart hospitals using RFID”, *J. Netw. Comput. Appl.*, vol. 34, p. 799–810, 2011.

[5] L. A. Amaral, “Elevando A Capacidade De Integração De Sistemas De Middleware Rfid Através Do Processamento De Eventos Complexos Distribuídos Entre Diferentes Organizações E Negócio”, Porto Alegre, p. 96, 30-ago-2011.

[6] JBoss “Drools”. [Online]. Available at: https://docs.jboss.org/drools/release/7.2.0.Final/drools-docs/html_single/index.html#_welcome. [Acessado: 19-ago-2017].

[7] A. Foundation, “Apache Flink”, 2017. [Online]. Available at: https://flink.apache.org/index.html. [Acessado: 02-ago-2017].

[8] WSO2, “Complex Event Processing”, 2017. [Online]. Available at: http://wso2.com/products/complex-event-processor/. [Acessado: 02-ago-2017].

[9] Java, “JRule Engine”, 2017. [Online]. Available at: http://jruleengine.sourceforge.net/. [Acessado: 03-ago-2017].

[10] Oracle, “Oracle CEP”. [Online]. Available at: https://docs.oracle.com/cd/E17904_01/doc.1111/e14476/overview.htm#CEPGS107. [Acessado: 14-out-2017].

[11] H. L. Severino, “Processamento de eventos complexos em tempo real: Implementação de um modelo de tarefas genérico”, *Trabalho de Conclusão de Curso*, Universidade de Passo Fundo, Passo Fundo, 2013.

[12] V. C. O. Junior e M. T. Rebonatto, “CEP embarcada em SoC com tolerância a falhas”. 2013.

[13] M. Palmer e M. Duzmuran, “An Introduction to Event Processing”, 2010.

[14] S. Weber, H. J. Lowe, S. Malunjkar, e J. Quinn, “Implementing a Real-time Complex Event Stream Processing System to Help Identify Potential Participants in Clinical and Translational Research Studies”, *AMIA Annual Symposium Proceedings Archive*, p. 472–476, nov-2010.

[15] S. Meister e O. Koch, “Information Overload in Telemedicine: Using Complex Event Processing and Context for Intelligent Information Filtering and Supply”, *IFAC Proc. Vol.*, vol. 45, nº 4, p. 7–12, 2012.

[16] L. Baumgärtner, C. Strack, B. Hoßbach, M. Seidemann, B. Seeger, e B. Freisleben, “Complex Event Processing for Reactive Security Monitoring in Virtualized Computer Systems”, *The 9th ACM International Conference on Distributed Event-Based Systems*, Oslo, p. 22–33, 29-jun-2015.

[17] JBoss, “jBPM – Open Source Business Process Management – Process engine”. [Online]. Available at: https://www.jbpm.org/. [Acessado: 09-set-2017].

[18] JBoss, “OptaPlanner – Constraint satisfaction solver (Java™, Open Source)”. [Online]. Available at: https://www.optaplanner.org/.

[19] JBoss, “UberFire”. [Online]. Available at: http://www.uberfireframework.org/docs/whoUses/drools.html. [Acessado: 25-set-2017].

[20] JBoss, “Drools Fusion”. [Online]. Available at: http://docs.jboss.org/drools/release/5.3.0.Final/drools-fusion-docs/html_single/index.html. [Acessado: 21-ago-2017].

[21] JBoss, “Drools Expert”. [Online]. Available at: https://docs.jboss.org/drools/release/5.2.0.Final/drools-expert-docs/html_single/. [Acessado: 21-ago-2017].

[22] J. Mogul, J. Gettys, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, “RFC 2616 - Hypertext Transfer Protocol -- HTTP/1.1,” 1999. [Online]. Available: https://tools.ietf.org/html/rfc2616. [Accessed: 25-Jan-2018].

Apêndice I.

Regra para detecção dos arquivos órfãos do tipo 1 e 6.

```
1. rule "rule-detect-orphan-event-one"
2. salience 10
3. when
4.      $typeOne : CapturaAtual(this.getEventos().getCodEvento()==1)
5.    $currentOne : CapturaAtual(
6.         getEventos().getCodEvento()==$typeOne.getEventos().getCodEvento()
7.         && getTomada().getCodTomada()==$typeOne.getTomada().getCodTomada()
8.         && getCodCaptura() != $typeOne.getCodCaptura())
9.      $delete : CapturaAtual( $typeOne after[0s] $currentOne)
10.  then
11.      new CapturaAtualDAO().updateCapturaAtualOrfao($typeOne);
12.      retract($typeOne);
13. end
14. rule "rule-detect-orphan-event-six"
15. salience 10
16. when
17.       $typeSix : CapturaAtual( this.getEventos().getCodEvento()==6)
18.       $currentSix:CapturaAtual(
19.               getEventos().getCodEvento()==$typeSix.getEventos().getCodEvento()
20.               && getTomada().getCodTomada()==$typeSix.getTomada().getCodTomada()
21.               && getCodCaptura()!=$typeSix.getCodCaptura())
22.       $delete : CapturaAtual( $typeSix after[0s] $currentSix)
23. then
24.             new CapturaAtualDAO().updateCapturaAtualOrfao($typeSix);
25.       retract($typeSix);
26. end
```

Regra para detecção dos arquivos órfãos do tipo 4 e 5.

```
1. rule " rule-detect-orphan-event-four"
2. salience 9
3. when
4.    $ typeFour : CapturaAtual(this.getEventos().getCodEvento()==4)
5.    $ currentFour : CapturaAtual(
6.         getEventos().getCodEvento()==$typeFour.getEventos().getCodEvento()
7.         && getTomada().getCodTomada()==$typeFour.getTomada().getCodTomada()
8.         && getCodCaptura() != $ typeFour.getCodCaptura())
9.    $delete : CapturaAtual( $ typeFour after[0s] $ currentFour)
10.  then
11.    new CapturaAtualDAO().updateCapturaAtualOrfao($typeOne);
12.    retract($typeOne);
13. end
14. rule " rule-detect-orphan-event-five"
15. salience 9
16. when
17.    $typeSix : CapturaAtual( this.getEventos().getCodEvento()==5)
18.    $currentSix:CapturaAtual(
19.       getEventos().getCodEvento()==$typeSix.getEventos().getCodEvento()
20.       && getTomada().getCodTomada()==$typeSix.getTomada().getCodTomada()
21.       && getCodCaptura()!=$typeSix.getCodCaptura())
22.    $delete : CapturaAtual( $typeSix after[0s] $currentSix)
23. then
24.    new CapturaAtualDAO().updateCapturaAtualOrfao($typeSix);
25.    retract($typeSix);
26. end
```

Regra para calcular a periculosidade da corrente para eventos do tipo 1.

```
1. rule "rule-calcule-corrente"
2. salience 6
3. when
4.    $captura : CapturaAtual( getEventos().getCodEvento() == 1)
5. then
6.    Integer $periculosidadeCorrente = EscalaCorrente.getStatusCorrente($captura);
7.    $captura.setPericulosidadeCorrente($periculosidadeCorrente);
8.    new CapturaAtualDAO().updateCorrente($captura);
9. end
```

Regra para calcular a periculosidade da corrente para eventos do tipo 1.

```
1. rule "rule-frequencia-corrente"
2. salience 5
3. when
4.    $captura : CapturaAtual( getEventos().getCodEvento() == 1 )
5. then
6.     Integer $periculosidadeFrequencia = EscalaFrequencia.getStatusFrequencia($captura);
7.    $captura.setPericulosidadeFrequencia($periculosidadeFrequencia);
8.    new CapturaAtualDAO().updateFrequencia($captura);
9. end
```

Regra para detectar a ocorrência de eventos simultâneos.

```
1. rule "rule-detected-similaridade-1"
2. salience 3 //regra será disparada antes da relação de início e fuga.
3. when
4.    $eventOneBefore : CapturaAtual( getEventos().getCodEvento() == 1)
5.    $eventOneAfter : CapturaAtual( this.getEventos().getCodEvento() == 1
6. && $eventOneBefore.getTomada().getCodTomada() != this.getTomada().getCodTomada()
7. && $eventOneBefore.getSalaCirurgia().getCodSala()==this.getSalaCirurgia().getCodSala())
8.    $outlet : CapturaAtual(
9.        this.getEventos().getCodEvento() == 6
10.  && this.getTomada().getCodTomada() != $eventOneBefore.getTomada().getCodTomada()
11.  && this.getTomada().getCodTomada() == $eventOneAfter.getTomada().getCodTomada()
12.  && this.getSalaCirurgia().getCodSala()==$eventOneAfter.getSalaCirurgia().getCodSala()
13.       ,this after[0s] $eventOneAfter && $eventOneAfter after[0s] $eventOneBefore )
14. then
15.    logger = Logger.getLogger(CapturaAtual.class);
16.    String $capturas;
17.    Double $res;
18.    Integer $periculosidadeSimilaridade;
19.    $capturas = $eventOneBefore.getCodCaptura() + "|" + $eventOneAfter.getCodCaptura();
20.    $res = Calculos.calcularSpearman($eventOneBefore, $eventOneAfter);
21.    $eventOneBefore.setSpearman($res);
22.    $eventOneAfter.setSpearman($res);
23.
24.    if ($res >= 0.95) {
25.       $periculosidadeSimilaridade = 3;
26.    } else if ($res <= 0.84900) {
27.       $periculosidadeSimilaridade = 1;
28.    } else {
29.       $periculosidadeSimilaridade = 2;
30.    }
31.    logger = Logger.getLogger(CapturaAtual.class);
31.    $eventOneBefore.setPericulosidadeSimilaridade($periculosidadeSimilaridade);
32.    $eventOneAfter.setPericulosidadeSimilaridade($periculosidadeSimilaridade);
33.    new CapturaAtualDAO().updateSimilaridade($capturas,$eventOneBefore,$eventOneAfter);
34. end;
35.
36. rule "rule-detected-similaridade-2"
37. salience 2 //regra será disparada antes da relação de início e fuga.
38. when
39.    $eventOneBefore : CapturaAtual( getEventos().getCodEvento() == 1)
40.    $eventOneAfter : CapturaAtual( this.getEventos().getCodEvento() == 1
41.    && $eventOneBefore.getTomada().getCodTomada()!=this.getTomada().getCodTomada()
42.&& $eventOneBefore.getSalaCirurgia().getCodSala()==this.getSalaCirurgia().getCodSala())
43.    $outlet : CapturaAtual(
44.        this.getEventos().getCodEvento() == 6
45.  && this.getTomada().getCodTomada() == $eventOneBefore.getTomada().getCodTomada()
46.  && this.getTomada().getCodTomada() != $eventOneAfter.getTomada().getCodTomada()
47.  &&this.getSalaCirurgia().getCodSala()==$eventOneBefore.getSalaCirurgia().getCodSala()
48.       ,this after[0s] $eventOneAfter && $eventOneAfter after[0s] $eventOneBefore)
49. then
50.    logger = Logger.getLogger(CapturaAtual.class);
51.    String $capturas;
52.    Double $res;
53.    Integer $periculosidadeSimilaridade;
54.    $capturas = $eventOneBefore.getCodCaptura() + "|" + $eventOneAfter.getCodCaptura();
55.    $res = Calculos.calcularSpearman($eventOneBefore, $eventOneAfter);
56.    $eventOneBefore.setSpearman($res);
57.    $eventOneAfter.setSpearman($res);
58.
59.    if ($res >= 0.95) {
60.       $periculosidadeSimilaridade = 3;
61.    } else if ($res <= 0.84900) {
62.       $periculosidadeSimilaridade = 1;
63.    } else {
64.       $periculosidadeSimilaridade = 2;
65.    }
66.    $eventOneBefore.setPericulosidadeSimilaridade($periculosidadeSimilaridade);
67.    $eventOneAfter.setPericulosidadeSimilaridade($periculosidadeSimilaridade);
68.    new CapturaAtualDAO().updateSimilaridade($capturas,$eventOneBefore,$eventOneAfter);
69. end;
```

Regra para detectar o fim do ciclo de um evento de fuga de corrente.

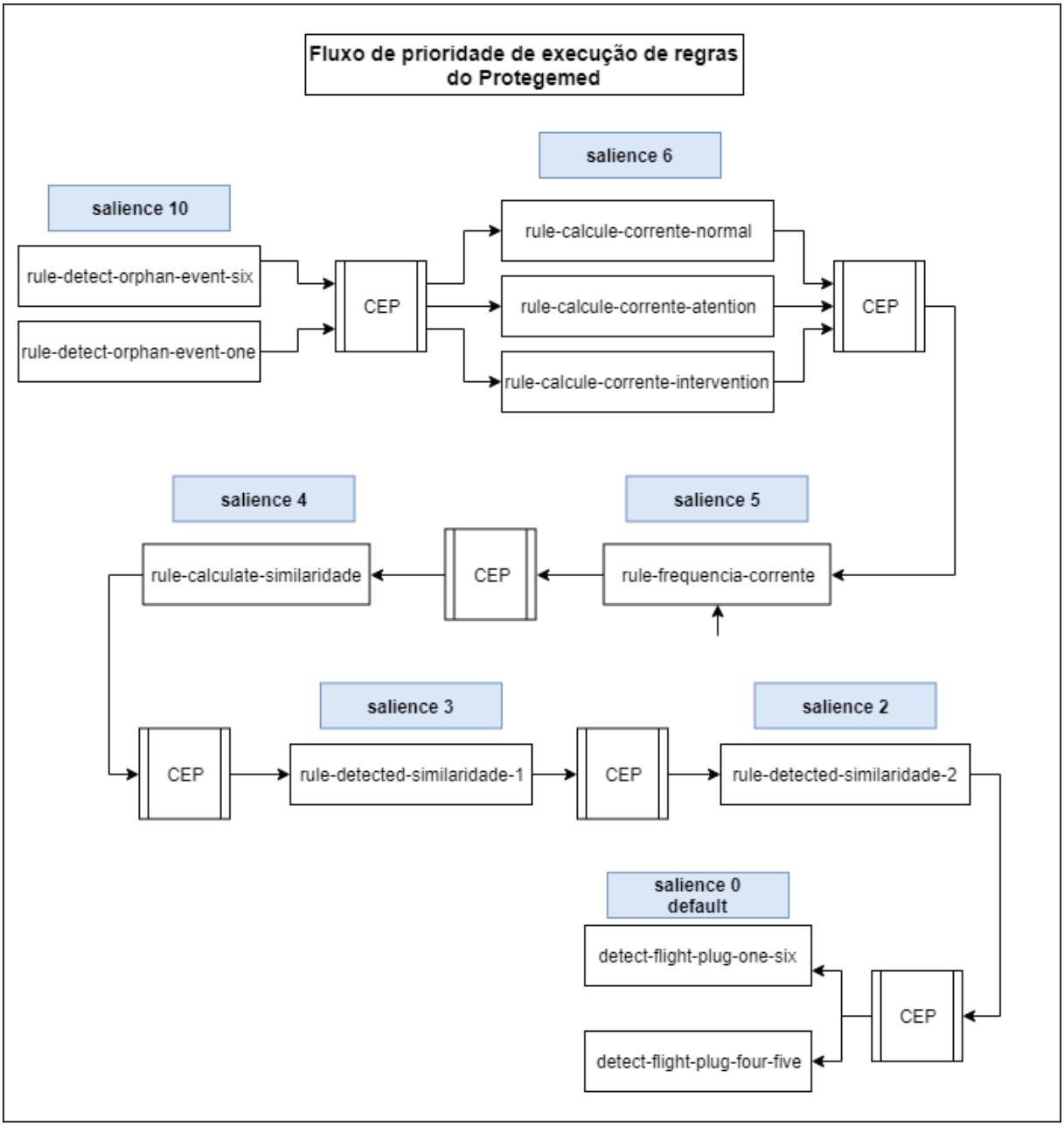
```
1. rule "detect-flight-plug-one-six"
2. when
3.    $eventOne : CapturaAtual( getEventos().getCodEvento() == 1 )
4.    $eventSix : CapturaAtual( getEventos().getCodEvento() == 6 )
5.    $outlet   : CapturaAtual(
6.    $eventSix.getTomada().getCodTomada() == $eventOne.getTomada().getCodTomada()
7.     && $eventSix after[0s] $eventOne)
8. then
9.    long $duracao=CapturaAtual.durationBetweenEvent($eventSix.getData(),
10.       $eventOne.getData());
11.    new CapturaAtualDAO().updateDurationCapturaAtual($eventOne, $eventSix, $duracao);
12.    delete($eventOne);
13.    delete($eventSix);
14. end
```

Regra para detectar o fim do ciclo de um evento de fase, ligamento e desligamento do equipamento.

```
1. rule "Detect flight plug four five"
2. when
3.    $eventFour : CapturaAtual( getEventos().getCodEvento() == 4 )
4.    $eventFive : CapturaAtual( getEventos().getCodEvento() == 5 )
5.    $outlet : CapturaAtual($eventFive.getTomada().getCodTomada()
6.        == $eventFour.getTomada().getCodTomada(),
7.       $eventFive after[0s] $eventFour)
8. then
9.    long $duracao = CapturaAtual.durationBetweenEvent($eventFive.getData(),
10.   $eventFour.getData());
11.   new CapturaAtualDAO().updateDurationCapturaAtual($eventFour, $eventFive,
12.       $duracao);
13.    retract($eventFour);
14.    retract($eventFive);
15. end
```

Apêndice II.

Ordem em que as regras devem ser executadas para a detecção correta dos eventos.



Apêndice III.

Distribuição dos eventos agrupados por tipo durante todo o período analisado.

| Evento | Total | Mês |
|---|---|---|
| 1 | 38 | 04/2015 |
| 4 | 322 | 04/2015 |
| 5 | 323 | 04/2015 |
| 6 | 38 | 04/2015 |
| 1 | 2.386 | 05/2015 |
| 4 | 50.893 | 05/2015 |
| 5 | 13.034 | 05/2015 |
| 6 | 993 | 05/2015 |
| 1 | 3.289 | 06/2015 |
| 4 | 36.339 | 06/2015 |
| 5 | 346 | 06/2015 |
| 6 | 910 | 06/2015 |
| 1 | 1.762 | 07/2015 |
| 4 | 25.089 | 07/2015 |
| 5 | 875 | 07/2015 |
| 6 | 733 | 07/2015 |
| 1 | 1.182 | 08/2015 |
| 4 | 31.168 | 08/2015 |
| 5 | 9.472 | 08/2015 |
| 6 | 337 | 08/2015 |
| 1 | 2 | 09/2015 |
| 4 | 3.941 | 09/2015 |
| 5 | 789 | 09/2015 |
| 6 | 1 | 09/2015 |
| 1 | 69 | 10/2015 |
| 4 | 3.059 | 10/2015 |
| 5 | 1.685 | 10/2015 |
| 6 | 62 | 10/2015 |
| 1 | 80 | 11/2015 |
| 4 | 10.394 | 11/2015 |
| 5 | 5.175 | 11/2015 |
| 6 | 36 | 11/2015 |
| 1 | 8 | 12/2015 |
| 4 | 2.131 | 12/2015 |
| 5 | 902 | 12/2015 |
| 6 | 8 | 12/2015 |

| Evento | Total | Mês |
|---|---|---|
| 1 | 3 | 01/2016 |
| 4 | 16.365 | 01/2016 |
| 5 | 16.359 | 01/2016 |
| 6 | 3 | 01/2016 |
| 1 | 5 | 02/2016 |
| 4 | 67.721 | 02/2016 |
| 5 | 67.689 | 02/2016 |
| 6 | 5 | 02/2016 |
| 1 | 1.958 | 03/2016 |
| 4 | 52.835 | 03/2016 |
| 5 | 39.231 | 03/2016 |
| 6 | 756 | 03/2016 |
| 1 | 325 | 04/2016 |
| 4 | 15.226 | 04/2016 |
| 5 | 2.850 | 04/2016 |
| 6 | 67 | 04/2016 |
| 1 | 26 | 05/2016 |
| 4 | 1.905 | 05/2016 |
| 5 | 1.826 | 05/2016 |
| 6 | 26 | 05/2016 |

Apêndice IV:

Distribuição dos eventos órfãos agrupados por tipo e mês de ocorrência durante o período analisado.

| Evento | Total | Órfãos | % | Mês |
|---|---|---|---|---|
| 5 | 323 | 1 | 0,31% | 04/2015 |
| 1 | 2.386 | 1.393 | 58,38% | 05/2015 |
| 4 | 50.893 | 37.859 | 74,39% | 05/2015 |
| 5 | 13.034 | 3 | 0,02% | 05/2015 |
| 1 | 3.289 | 2.379 | 72,33% | 06/2015 |
| 4 | 36.339 | 35.996 | 99,06% | 06/2015 |
| 1 | 1.762 | 1.028 | 58,34% | 07/2015 |
| 4 | 25.089 | 24.214 | 96,51% | 07/2015 |
| 1 | 1.182 | 846 | 71,57% | 08/2015 |
| 4 | 31.168 | 21.696 | 69,61% | 08/2015 |
| 1 | 2 | 1 | 50,00% | 09/2015 |
| 4 | 3.941 | 3.152 | 79,98% | 09/2015 |
| 1 | 69 | 7 | 10,14% | 10/2015 |
| 4 | 3.059 | 1.371 | 44,82% | 10/2015 |
| 1 | 80 | 44 | 55,00% | 11/2015 |
| 4 | 10.394 | 5.222 | 50,24% | 11/2015 |
| 4 | 2.131 | 1.229 | 57,67% | 12/2015 |
| 4 | 16.365 | 6 | 0,04% | 01/2016 |
| 4 | 67.721 | 35 | 0,05% | 02/2016 |
| 5 | 67.689 | 3 | 0,00% | 02/2016 |
| 1 | 1.958 | 1.202 | 61,39% | 03/2016 |
| 4 | 52.835 | 13.604 | 25,75% | 03/2016 |
| 1 | 325 | 259 | 79,69% | 04/2016 |
| 4 | 15.226 | 12.378 | 81,30% | 04/2016 |
| 5 | 2.850 | 2 | 0,07% | 04/2016 |
| 6 | 67 | 1 | 1,49% | 04/2016 |
| 4 | 1.905 | 79 | 4,15% | 05/2016 |

Apêndice V:

Distribuição dos eventos simultâneos detectados com sua quantidade de ocorrência no mês e seu tipo de periculosidade.

| Periculosidade | Quantidade | Mês |
|---|---|---|
| Normal | 582 | 06/2015 |
| Normal | 17 | 11/2015 |
| Normal | 506 | 07/2015 |
| Normal | 4 | 10/2015 |
| Normal | 3 | 12/2015 |
| Normal | 70 | 05/2015 |
| Normal | 13 | 08/2015 |
| Normal | 49 | 04/2016 |
| Normal | 520 | 03/2016 |
| Normal | 1 | 09/2015 |
| Atenção | 5 | 12/2015 |
| Atenção | 19 | 04/2016 |
| Atenção | 23 | 03/2016 |
| Atenção | 11 | 08/2015 |
| Atenção | 4 | 11/2015 |
| Atenção | 78 | 07/2015 |
| Atenção | 1 | 10/2015 |
| Atenção | 36 | 06/2015 |
| Perigo | 7 | 08/2015 |
| Perigo | 4 | 05/2015 |
| Perigo | 2 | 07/2015 |
| Perigo | 3 | 10/2015 |
| Perigo | 24 | 06/2015 |
| Perigo | 2 | 11/2015 |
| Perigo | 19 | 03/2016 |