Pós-Graduação em Ciência da Computação

Rafael Isaias Rodrigues Coêlho

Web-REFlex: uma solução para evitar Context Tunneling na execução de processos de negócio declarativos

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
http://cin.ufpe.br/~posgraduacao

RECIFE
2017

Rafael Isaias Rodrigues Coêlho

Web-REFlex: uma solução para evitar Context Tunneling na execução de processos de negócio declarativos

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Orientador: Ricardo Massa Ferreira Lima
Coorientadora: Renata Medeiros de Carvalho

RECIFE
2017

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

C672w Coêlho, Rafael Isaias Rodrigues
Web-REFlex: uma solução para evitar Context Tunneling na execução de processos de negócio declarativos / Rafael Isaias Rodrigues Coêlho. – 2017.
83 f.:il., fig., tab.

Orientador: Ricardo Massa Ferreira Lima.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2017.
Inclui referências e apêndices.

1. Engenharia de software. 2. Modelagem de processos de negócios. I. Lima, Ricardo Massa Ferreira (orientador). II. Título.

005.1 CDD (23. ed.) UFPE- MEI 2017-228

Rafael Isaias Rodrigues Coêlho

Web-REFlex: uma solução para evitar Context Tunneling na execução de processos de negócios declarativos

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação

Aprovado em: 10/03/2017.

BANCA EXAMINADORA

_____

Prof. Dr. Adriano Lorena Inácio de Oliveira
Centro de Informática / UFPE

_____

Prof. Dr. César Augusto Lins Oliveira
Banco Interamericano de Desenvolvimento

_____

Prof. Dr. Ricardo Massa Ferreira Lima
Centro de Informática / UFPE
(**Orientador**)

À minha mãe e meu pai.

AGRADECIMENTOS

Eu dedico este trabalho aos meus pais, Manoel e Maria, que sempre conspiraram a favor do meu sucesso profissional e plenitude como ser humano. Eles, sem dúvida, foram o principal elemento motivador durante toda a minha jornada.

Ao meu querido irmão, Ricardo, pelo amor, apoio e amizade.

As minhas doces avós pela ternura sempre presente, pelos grandes exemplos morais que sempre seguirei.

A minha companheira, Denise, pelo imenso amor e pela contribuição essencial durante todo o processo.

Ao meu orientador prof. Ricardo Massa pelas contribuições e o vital apoio no momento mais difícil desta pós-graduação.

A minha co-orientadora Dr.ª Renata Carvalho por ser parte fundamental deste trabalho. Eu não tenho dúvidas que este não existiria sem a sua presença constante e compreensão humana extraordinária. Minha eterna gratidão por toda colaboração.

Ao Dr. Nelson Souto Rosa, pelo início desta jornada e, seguramente, pelos ensinamentos no início de minha caminhada como pesquisador.

Aos avaliadores Dr. Adriano Lorena e Dr. César Oliveira pelos relevantes comentários e a gentil atenção a este trabalho.

A esta universidade, seu corpo docente, direção e administração que oportunizaram a janela que hoje vislumbro um horizonte superior.

"I have yet to see any problem, however complicated, which, when looked at in the right way, did not become still more complicated."

(Poul Anderson)

RESUMO

Os sistemas de gerenciamento de processos de negócio (BPMS) tradicionais frequentemente incluem restrições no compartilhamento de informações contextuais, problema conhecido como *Context Tunneling*. Este ocorre quando o contexto necessário para controlar a execução de um processo apenas é visível para seus respectivos atores, portanto, um usuário conhece as atividades que foram destinadas a ele, mas não consegue ter uma visão geral de como a execução de suas atividades pode impactar no restante do processo. Dessa forma, o usuário não pode decidir por fazer atividades mais impactantes antes, ou que acelerariam o encerramento do processo. Esta visão restrita geralmente resulta em erros e ineficiências. Essa limitação contextual possui maior impacto para processos nos quais múltiplos participantes estão envolvidos na execução das atividades para alcançar um objetivo comum, tais como cuidados médicos, desenvolvimento de sistemas e organização de eventos. Esses processos representam a maioria dos processos do mundo real e são conhecidos como processos colaborativos. Nesse tipo de processo, a comunicação e a colaboração tornam-se características importantes, visto que a decisão tomada por um usuário pode influenciar o fluxo restante do processo, assim como as ações disponíveis (ou não) para os outros usuários desse processo de colaboração. A abordagem *Case Management* expressa o gerenciamento do processo baseando-se nas informações do contexto no qual está inserido, com o objetivo de mitigar o problema de *Context Tunneling*. Dessa forma, este trabalho propõe a extensão do framework REFlex, introduzindo ao cenário declarativo já existente, características próprias do *Case Management*. Para isso, apresentamos uma solução web (Web-REFlex) capaz de executar várias instâncias e gerenciar informações de contexto para processos de negócio colaborativos. O Web-REFlex registra dados baseado em decisões tomadas previamente por usuários e, através de estatísticas e da técnica *Analytic Hierarchy Process (AHP)*, disponibiliza sugestões de atividades e informações globais sobre o processo. Desse modo, é possível permitir que os usuários tenham uma visão ampla, e não somente restrita ao seu contexto, para que tirem proveito dessas informações a fim de tomar melhores decisões sobre quem deve realizar e/ou qual etapa deve ser executada para satisfazer os objetivos atuais do processo.

Palavras-chaves: Processos de negócio colaborativo. Processos declarativos. *Case Management*. *Context Tunneling*. MCDM. AHP.

ABSTRACT

Traditional Business Process Management Systems (BPMS) often comprise contextual sharing information constraints, creating a problem known as Context Tunneling. It may happen when the context information needed to control the process execution is only visible for its respective actors and, although a user knows his immediate activities, he may not be aware of the overall process, which can severely impact on future results. In this way, the user cannot decide to do more impacting activities before, or that would accelerate the conclusion of the process. This narrow view often results in errors and inefficiencies. Activities that require multiple participants per goal are more vulnerable and yet the most popular in real-world collaborative processes, for instance in medical care, system development and event organization. In this type of process, communication and collaboration become relevant features, since the decision made by a user can alter the process flow, as well as the actions available (or not) to other users. The Case Management approach expresses the process management based on the context information in which it is inserted, so as to mitigate the Context Tunneling problem. This work proposes the REFlex framework extension, introducing specific characteristics of Case Management to the existing declarative scenario. To do this, we present a web-based solution (Web-REFlex) capable of running multiple instances and managing context information for collaborative business processes. Web-REFlex records data based on decisions previously made by users and, through statistics and the AHP technique, provides comprehensive suggestions and information about the process. Furthermore it is possible to allow users to take a broad view, not only in local context, to take advantage of this information in order to make better decisions about who should perform and/or which steps should be performed to meet current goals of the process.

Key-words: Collaborative Business Processes. Declarative Process. Case Management. Context Tunneling. MCDM. AHP.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AHP Analytic Hierarchy Process.

API Application Programming Interface.

BPEL Business Process Execution Language.

BPM Business Process Management.

BPMN Business Process Model and Notation.

BPMS Business Process Management Systems.

CMMN Case Management Model and Notation.

EPC Event Process Chain.

JPA Java Persistence API.

JSON JavaScript Object Notation.

LTL Lógica Temporal Linear.

MCDM Multiple-Criteria Decision-Making.

OMG Object Management Group.

REST Representational State Transfer.

WFMS Workflow Management Systems.

SUMÁRIO

1 INTRODUÇÃO

Business Process Management (BPM) é frequentemente usado por companhias para descrever seus processos de negócio. Segundo Aalst (AALST et al., 2004), o BPM "permite apoiar processos de negócio utilizando métodos, técnicas e software para conceber, implementar, controlar e analisar processos operacionais envolvendo seres humanos, organizações, aplicações, documentos e outras fontes de informação". O gerenciamento de um processo de negócio tem início com a definição de um modelo, através de uma linguagem de modelagem, que permite a representação gráfica do processo.

Workflow é uma abordagem tradicional e amplamente utilizada para a automação de processos de negócio. Nessa abordagem, um processo é descrito a partir de um fluxo de controle rígido. Assim, para que não haja falhas na execução do processo, é imprescindível definir: 1) o conjunto de atividades do processo; 2) em que ordem essas atividades devem ser executadas e 3) quem deve realizar essas atividades.

Buscando atribuir mais flexibilidade na execução desses processos, a comunidade de BPM propôs soluções alternativas. Nesse contexto, surgiram os sistemas de gerenciamento de processos declarativos. O objetivo de um processo de negócio declarativo é descrever ações que devem ser executadas sem descrever, no entanto, como, quem ou em que ordem devem ser executadas (PESIC, 2008). Para isso, os modelos declarativos definem um conjunto de atividades (ações) e um conjunto de regras de negócio (restrições) associadas a essas atividades. A execução de qualquer sequência de atividades é permitida desde que não viole nenhuma das restrições definidas.

Os processos declarativos são úteis para empresas cuja lógica de negócio é dinâmica ou está em constante mudança. Além disso, alguns autores (EIJNDHOVEN; IACOB; PONISIO, 2008) acreditam que o entendimento do fluxo do processo a partir das regras de negócio é mais natural para os profissionais envolvidos, que normalmente possuem poucas habilidades técnicas de TI.

O REFlex (CARVALHO, 2015) é uma ferramenta que permite a definição de processos de negócio declarativos. Sua estrutura é baseada em grafos, nos quais os nós representam as atividades e as arestas, as regras de negócio. Quando uma atividade do processo é executada, o REFlex interpreta dinamicamente o estado atual e as regras de negócio para determinar o próximo estado do processo.

Embora propicie maior flexibilidade na definição e gerenciamento dos processos de negócio, a abordagem declarativa apresenta limitações em relação à gestão de processos de negócio colaborativos. Esses processos são aqueles em que vários participantes estão envolvidos na execução das atividades para alcançar um objetivo comum, fato que representa quase todas as práticas de trabalho no mundo real, tais como cuidados médicos, desenvolvimento de sistemas e organização de eventos.

Nesse tipo de processo, a comunicação torna-se uma característica importante, visto que a decisão tomada por um usuário pode influenciar o fluxo restante do processo, assim como as ações disponíveis (ou não) para os outros usuários desse processo de colaboração. Como exemplos podemos citar: 1) a restrição na execução de algumas tarefas que devem ser feitas por atores escolhidos com base em relações sociais ou de trabalho; 2) uma ação do usuário pode habilitar (ou desabilitar) sub-fluxos do processo e/ou 3) a dependência entre ações envolvendo diferentes usuários podendo indicar, por exemplo, que a ação X do usuário A não pode ser executada antes da ação Y do usuário B.

De acordo com Bazán (BAZÁN, 2015), na maioria dos sistemas de BPM (*BPMS*), os participantes possuem uma visão limitada dos processos em que estão inseridos, perdendo informações importantes relacionadas ao contexto global como, por exemplo, o conhecimento sobre as outras pessoas envolvidas e sobre as histórias e resultados de execuções anteriores. Essa visão limitada está associada ao problema de *Context Tunneling*, que de acordo com Aalst et al. (AALST; WESKE; GRÜNBAUER, 2005) ocorre quando os dados necessários para controlar a execução do processo são visíveis apenas para seus respectivos atores, portanto, o contexto global para a instância do processo não está diretamente disponível.

O conceito de software social, que tornou-se um fenômeno no cenário atual de TI, favorece a integração da equipe e a colaboração, pois permite atravessar as barreiras de comunicação entre os membros da equipe e/ou de outras organizações (BAZÁN, 2015). Assim, a introdução de características inerentes aos softwares sociais no cenário de gerenciamento de processos colaborativos facilita o relacionamento entre os participantes do processo, permitindo alcançar melhores resultados em sua execução.

Outro conceito importante é o paradigma *Case Handling* (AALST; WESKE; GRÜNBAUER, 2005) que surgiu como uma eficiente abordagem para apoiar o gerenciamento de processos de negócio colaborativos. Essa abordagem enxerga o processo como um caso, por exemplo, a solicitação de um crédito pessoal em uma operadora ou a reivindicação do pagamento de seguro. Esse caso, ou contexto, inclui informações gerais sobre o processo em questão tais como atividades, pessoas envolvidas, os relacionamentos tanto entre pessoas quanto dessas com as atividades, conhecimento dos participantes.

Dessa forma, ao utilizar o paradigma *Case Handling*, as organizações podem obter uma visão mais transparente de seus processos de negócio, permitindo que elas analisem o processo de ponta a ponta em vez de examiná-lo passo a passo (ECKHARDT, 2015). Ou seja, a utilização do paradigma *Case Handling* facilita o acesso aos dados do processo (*Case Data*) e, dessa forma, fornece aos participantes uma visão geral das informações do contexto no qual o processo está inserido. Além disso, possibilita a distribuição das atividades entre os participantes do processo, assim, o fluxo das atividades não fica restrito às decisões de um único usuário pois a ordem de execução das atividades pode ser atribuída diretamente ao ator responsável por sua execução.

1.1 Motivação

Na indústria de software atual não é possível encontrar muitos BPMS declarativos que se aproveitam de informações contextuais dos seus processos de negócio. Os sistemas atuais mantêm em suas características a execução de processos com baixa capacidade para absorver informações relativas ao contexto global em que estão inseridos.

Nesse sentido, o sistema declarativo REFlex, que será utilizado como base para a ferramenta de extensão proposta nesse trabalho, possui suas limitações: (1) o sistema apenas pode ser executado localmente, limitada a um usuário por vez; (2) não é permitida a definição de cargos, dessa forma não é possível determinar quem é responsável pela execução de cada atividade, e assim, qualquer participante pode executar qualquer tarefa; (3) a criação e execução de múltiplas instâncias de um processo não é suportada; (4) nenhum tipo de histórico é armazenado, assim, as informações e lições aprendidas de outras instâncias não são aproveitadas para novos processos.

Ou seja, embora o REFlex apresente melhorias substanciais em relação a outros trabalhos, principalmente para processos usados no mundo real, possui problemas que limitam o gerenciamento de processos de negócio colaborativos. Esses problemas motivam a proposta desse trabalho e serão discutidos com mais detalhes ao longo desse documento.

1.2 Objetivos da Proposta

Objetivo principal

Este trabalho propõe uma extensão para ao framework REFlex que permita mitigar o problema de *Context Tunneling* em processos de negócio. Para isso, serão incorporados ao REFlex conceitos do paradigma *Case Handling* e características relacionadas a colaboração, comunicação e cooperação dos softwares sociais. Assim, a ferramenta resultante, Web-REFlex, deverá contribuir com o gerenciamento de processos de negócio ao permitir aos atores envolvidos acesso a informações gerais do contexto de uma instância de processo, permitindo que os usuários possam fazer uso dessas informações para decidir quem e/ou qual atividade deve ser executada para satisfazer a demanda do processo de negócio em execução.

Objetivos específicos

Os seguintes objetivos específicos foram enumerados para atingir o nosso principal objetivo:

- permitir acesso simultâneo do Web-REFlex através de diferentes tipos de dispositivos;
- permitir o controle de múltiplos usuários;
- permitir o controle de múltiplas instâncias de processos;

- computar métricas estatísticas baseadas em decisões tomadas previamente para um cenário, e disponibilizar o conhecimento dessas para ajudar aos usuários na tomada de decisões;
- priorizar atividades de acordo com seu nível de criticidade.

1.3 Resumo das Contribuições

A introdução de características dos softwares sociais agrega melhorias às tecnologias e metodologias BPM atuais. A modelagem de processos de negócio obtém vantagens das técnicas de software social, permitindo integrar e compartilhar o conhecimento de todos os envolvidos no processo. Além disso, a utilização desses conceitos sociais permite coletar e incorporar informações valiosas para a melhora contínua do processo.

A principal contribuição desse trabalho refere-se à incorporação de características sociais e colaborativas em processos de negócio inseridos em contextos distribuídos. O trabalho proposto constitui uma extensão da ferramenta REFlex, que foca na modelagem e execução de processos de negócio utilizando uma abordagem declarativa para definição das regras de negócio associadas ao processo em execução.

Em essência, a nova ferramenta web proposta (Web-REFlex) traz melhorias para execução de processos colaborativos, permitindo que os usuários tirem proveito das informações referentes ao contexto fornecidas pelo Web-REFlex a fim de tomar decisões que satisfaçam a real necessidade do processo de negócio em questão.

1.4 Organização da Dissertação

Para melhor compreensão, este trabalho foi organizado em 6 capítulos. Os capítulos seguintes descrevem:

- Capítulo 2 - Gerenciamento de processos de negócio: apresenta a base conceitual para o desenvolvimento do trabalho e relaciona os principais conceitos sobre as áreas envolvidas na temática de pesquisa.
- Capítulo 3 - REFlex: descreve o REFlex, um framework para modelar processos de negócio declarativos. A ferramenta proposta atua na fase de execução de um processo de negócio declarativo, utilizando o REFlex para gerenciar e atualizar o status das atividades.
- Capítulo 4 - Web-REFlex: apresentação da ferramenta proposta por este trabalho.
- Capítulo 5 – Caso de uso da ferramenta: com o intuito de validar este trabalho este capítulo apresenta um estudo de caso simulado na ferramenta

- Capítulo 6 – Considerações Finais: apresenta os desafios, contribuições, trabalhos futuros e conclusões.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta um referencial teórico sobre os principais conceitos abordados nessa pesquisa. A seção 2.1 apresenta os conceitos de BPM, seu ciclo de vida e o processo de automatização desse gerenciamento. A seção 2.2, apresenta conceitos relacionados à abordagem declarativa para o gerenciamento de processos de negócio. A seção 2.3 descreve o problema de *Context Tunneling*. A seção 2.4 apresenta os conceitos de software social e como esses são importantes para os processos de negócio. A seção 2.5 define o paradigma *Case Management*, suas principais características e como essa abordagem consegue resolver o problema de *Context Tunneling*. Por fim, a seção 2.6 define os conceitos de computação em nuvem.

2.1 Processos de Negócio

Segundo McCormack (MCCORMACK KEVIN P., 2001), processo é um grupo específico de atividades e tarefas subordinadas que resultam em um produto de valor. Sob a ótica do gerenciamento de processos, Martyn (OULD MARTYN A. E OULD, 1995) define processo como um conjunto coerente de atividades conduzido por um grupo de colaboradores para atingir um objetivo.

Para as organizações empresariais, entender como funcionam seus processos é importante para determinar como eles devem ser gerenciados a fim de obter o máximo desempenho como resultado (GONCALVES, 2000). Por isso, atualmente muitas empresas focam em uma abordagem orientada a processos para gerenciar seu negócio. Esta ideia denota o conceito de *BPM*.

BPM é uma metodologia que fornece à equipe de gerenciamento e à organização como um todo uma clara compreensão da forma como o trabalho é realmente realizado (WURTZEL, 2012). O BPM busca mapear e melhorar os processos de negócio da empresa, para isso utiliza uma abordagem definida por um ciclo de vida (MUIJRES, 2011). Para ilustrar esse ciclo, utilizamos o modelo definido pela ABPMP (CBOK, 2009), que possui um total de 6 fases como pode ser visto na Figura 1:

- Planejamento: o ciclo de vida BPM inicia com o planejamento do processo. Nessa fase são estabelecidas as estratégias que serão utilizadas e os objetivos que a empresa deseja alcançar com o processo.
- Análise: durante esta fase é realizada uma análise da organização em que o processo será inserido e da estrutura do processo. A partir disso, é possível aliar os resultados dessa análise com as estratégias e objetivos definidos na fase de planejamento, e assim definir os requisitos necessários.

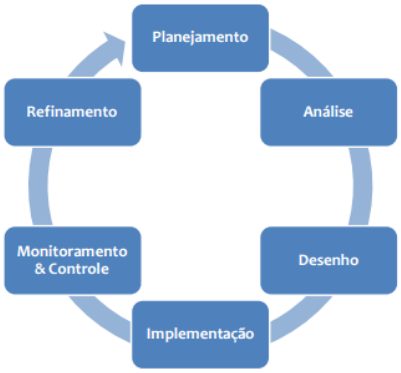


Figura 1 – Ciclo de vida do BP (ABPMP, 2009)

- Desenho: os requisitos identificados na fase anterior são o ponto de partida dessa etapa. As características do processo, os recursos, a ordem das atividades e os aspectos organizacionais são determinados. Esses requisitos são documentados com a ajuda de modelos de processos de negócio, que descrevem graficamente as informações do processo.
- Implementação: os modelos desenhados na fase anterior são executados ou automatizados. Nesta fase são alinhadas as relações entre o mundo real e o modelo do processo.
- Monitoramento: o monitoramento dos processos no sistema é importante para reconhecer desvios ou problemas numa fase inicial. O desempenho pode ser medido manualmente ou por software e os resultados devem estar em conformidade com as expectativas previamente definidas para o modelo.
- Refinamento: nesta fase os resultados da fase de monitoramento são analisados. Com base nas medições de desempenho, melhorias podem ser propostas para o modelo do processo de negócio através de novos requisitos. Estes serão encaminhados para dar início a um novo ciclo a fim de alterar os processos e modelos correspondentes. Assim, o desempenho global dos processos de negócio é continuamente melhorado.

Percebe-se, a partir das fases do ciclo de vida do BPM, que os modelos de processos de negócio desempenham um papel importante para esse fluxo. De acordo com KO (KO, 2009), pessoas tendem a compreender melhor um processo através de modelos. A modelagem dos processos permite que as pessoas consigam identificar ou entender problemas, perceber áreas de potencial melhoria e definir os papéis e responsabilidades dentro da organização. Assim, se o modelo não é bem definido na fase de modelagem, a qualidade das próximas fases é afetada negativamente (SCHREPFER, 2010).

Um modelo de processo pode ser um grafo muito simples ou uma estrutura complexa e detalhada. Para a representação das informações necessárias, os modelos são desenvolvidos utilizando uma linguagem de modelagem ou notação. Várias linguagens podem ser

utilizadas para este propósito, tais como *Business Process Model and Notation (BPMN)* (WHITE, 2004), *Business Process Execution Language (BPEL)* (WEERAWARANA et al., 2005), redes de Petri (AALST, 1998), *Event Process Chain (EPC)* (MENDLING, 2008), fluxogramas, etc.

Para automatizar o processo de gerenciamento desses modelos de negócio foram desenvolvidos sistemas, conhecidos como *BPMS* ou Sistemas de Gerenciamento de Processo de Negócio, que oferecem suporte a todas as fases do ciclo de vida do BPM, monitorando as atividades e o fluxo de dados durante toda a execução do processo.

2.2 BPMS

BPMS, do inglês *Business Process Management Suites / System*, são as ferramentas/sistemas de softwares responsáveis pela automação, execução, controle e monitoração das etapas das atividades e tarefas realizadas em uma organização.

Os BPMS são resultado do desenvolvimento dos sistemas de informação de apoio às tarefas para a gestão de processos, como os sistemas de modelagem de processos utilizando ferramentas gráficas e analíticas que têm por objetivo registrar, representar, compreender e analisar desenhos de processos.

BPMS tem suas raízes em ferramentas de *workflow*. O *Workflow* é uma abordagem tradicional e amplamente utilizada. A principal característica dessa abordagem é o rígido controle para determinar o fluxo de execução das atividades. Um modelo que usa a abordagem de *workflow* deve identificar, no entanto, o conjunto de atividades do processo, o ator responsável pela execução de cada atividade, e os possíveis fluxos de execução.

No entanto, sistemas *workflow* eram limitados para prover a velocidade e flexibilidade necessárias pelas práticas atuais do BPM (CARRARA, 2011). Com a evolução, os sistemas tornaram-se capazes de refletir a operação de negócio e, com a adição de motores de regras e geradores de aplicação no início dos anos 2000, começaram a trilhar um caminho diferente – evoluíram para ambientes de operação com aplicações geradas e executadas dentro dos BPMS (WILLEMANN, 2017).

Segundo Chang (CHANG, 2016) , o maior avanço dos BPMS sobre os sistemas tradicionais *workflow* é a capacidade de suporte de integrações centradas em processos, ou seja, a integração de pessoas, sistemas e dados.

Para Pessôa e STORCH (PESSÔA; STORCH, 2006), um BPMS deve possuir ferramentas para:

- Modelagem;
- Definição de regras de negócios;
- Integração com outros sistemas;
- Controle e execução dos processos;

- Interação com usuários participantes dos processos;
- Monitoramento dos processos.

Em resumo, um BPMS proporciona um novo nível de automação por meio da criação e execução de aplicações que combinam a lógica definida nos modelos de negócio com regras e dados conectados às atividades. Essa capacidade de definir e gerar aplicações de negócio a partir de modelos e regras permite que o BPMS ofereça um gerenciamento avançado de fluxo de trabalho, proporcionando melhorias significativas para as empresas que utilizam essas ferramentas.

Dentre os benefícios advindos do uso de BPMs, Cláudio (WILLEMANN, 2017) destaca:

- Velocidade na modelagem e geração da aplicação;
- Qualidade por meio da capacidade de exteriorizar regras e testá-las individualmente e em grupos;
- Flexibilidade através de interação rápida entre os participantes.

2.3 Processos Declarativos

Os modelos de processos de negócio tradicionais utilizam uma abordagem imperativa para definir o fluxo de execução dos processos. Para permitir uma maior flexibilidade na automatização do fluxo de atividades no processo, abordagens declarativas têm recebido crescente interesse e sugerem uma maneira fundamentalmente diferente de descrever processos de negócio (PESIC, 2008), como será detalhado a seguir.

A abordagem declarativa foca apenas na lógica que direciona as interações entre as ações no processo, descrevendo as atividades que podem ser realizadas, bem como restrições que proíbem o comportamento indesejado (ZUGAL et al., 2015). Ou seja, nessa abordagem apenas as características essenciais são descritas, para isso são definidas restrições que limitam as possibilidades de execução de um processo (REIJERS; SLAATS; STAHL, 2013).

O conceito de processos de negócio declarativos surgiu da adoção do paradigma de programação declarativa, que expressa a lógica de execução do programa sem descrever seu fluxo de controle. A ideia principal da modelagem de processos de negócio declarativa é que um processo é visto como uma trajetória em um espaço de possíveis estados (atividades) e que restrições (ou regras de negócio) são usadas para definir os movimentos válidos nesse espaço de estado (GOEDERTIER; VANTHIENEN; CARON, 2015). O foco da abordagem declarativa é identificar um conjunto mínimo de regras que devem ser respeitadas no fluxo de controle durante seu processo de execução, permitindo um maior grau de flexibilidade do que os fluxos fixos tradicionais (MONTALI et al., 2010).

Dessa forma, um modelo declarativo captura um processo sob uma suposição de que tudo é permitido a menos que seja explicitamente proibido por uma regra. Neste contexto, uma regra pode assumir a forma de uma relação binária entre pares de tarefas e deve ser satisfeita em cada execução de um processo, como por exemplo "a tarefa B só pode ser realizada se a tarefa A tiver sido realizada anteriormente" (GIACOMO et al., 2015).

Nessa abordagem baseada em regras, um modelo de processo é composto por basicamente dois elementos: atividades e restrições. As restrições são as regras de negócio e são usadas para definir aspectos das atividades no processo de execução do processo. As regras descrevem as atividades que podem ser realizadas, bem como definem restrições que impedem comportamentos indesejados.

Há uma série de conceitos diferentes, utilizados pelos pesquisadores, para definir uma regra de negócio. Para Gomez (GOMEZ; FRANCO, 2010), "uma regra de negócio é uma declaração que define ou restringe algum aspecto do negócio, e que não pode ser decomposta em regras mais detalhadas". O *Business Rule Group* (HAY et al., 2000) define o termo "regras de negócio" sob duas perspectivas diferentes: (1) do ponto de vista do negócio como "uma diretiva destinada a governar, orientar ou influenciar o comportamento empresarial" e (2) como "um pedaço atômico de lógica de negócio reutilizável, que é especificado declarativamente", da perspectiva de TI.

Para ilustrar a modelagem de um processo declarativo, será descrito um processo simples de pagamento numa loja virtual. Em um sistema online de vendas, por exemplo, existe algumas formas de pagamento disponíveis: cartão de crédito, cartão de débito, boleto bancário e PayPal. A utilização das regras dos processos declarativos pode definir e limitar muitos aspectos da execução desse processo. Caso nenhuma regra seja definida, no caso do processo de pagamento em questão, os usuários do sistema poderão realizar os pagamentos da forma como acharem melhor. Entretanto, se um conjunto de regras for adicionado, as opções de fluxo serão limitadas. A seguir, alguns exemplos de regras que poderiam ser utilizadas para definir o fluxo de pagamento do sistema:

- Regra de Negócio 1: se os produtos selecionados estiverem em promoção, os clientes deverão usar cartão de débito como forma de pagamento.
- Regra de Negócio 2: apenas se o cliente selecionar as opções boleto bancário ou Paypal, o desconto de 10% no valor a ser pago será concedido.
- Regra de Negócio 3: caso o cliente selecione a forma de pagamento cartão de crédito, será necessário checar a identidade através de documento com foto.

Atualmente existe um conjunto de ferramentas disponíveis tanto na comunidade acadêmica quanto para uso comercial. Algumas delas são:

- DCR Graphs (HILDEBRANDT; MUKKAMALA, 2011) baseia-se em grafos e usa lógica gráfica para expressar os estados em tempo de execução do processo. A ferramenta, disponível em *http://dcrgraphs.com/the-tool/*, é utilizada principalmente na academia, porém inúmeras empresas em todo o mundo já integraram gráficos DCR em seus sistemas de TI (DCR, 2011).
- A ferramenta DECLARE (BURATTIN et al., 2012), ao contrário dos sistemas convencionais que usam linguagens de modelagem semelhantes a grafos (por exemplo, redes de Petri), baseia-se na lógica para modelar e executar processos de negócios. Utiliza Lógica Temporal Linear (LTL) como formalismo para a representação dos processos de negócio e combina isso com uma linguagem gráfica extensível, conhecida como ConDec (PESIC, 2008). A ferramenta está disponível em *www.win.tue.nl/declare/*
- O REFlex (CARVALHO, 2015), que será usado como base para o desenvolvimento da ferramenta proposta nesta dissertação e será melhor detalhado no capítulo 3, é um mecanismo baseado em grafos que verifica a conformidade das regras durante a execução do processo.

2.4 O problema de Context Tunneling

A expressão *Context Tunneling* foi definida por Aalst (AALST; WESKE; GRÜNBAUER, 2005) e ocorre quando apenas as informações necessárias para controlar a execução do processo são disponibilizadas ao usuário, ficando limitada a visão do contexto geral da instância em que o processo está inserido.

De acordo com Dustdar (DUSTDAR, 2007), *Tunneling* é baseado em três noções de contexto (Fig. 2):

- Contexto individual: refere-se às relações entre indivíduo e atividade.
- Contexto das atividades: refere-se às relações entre as atividades do processo.
- Contexto da equipe: refere-se ao grupo de usuários envolvido no processo de negócio.

Context Tunneling pode ser encontrado por usuários na maioria dos *BPMS*. Esses sistemas fornecem aos usuários uma visão limitada dos processos nos quais participam, não levando em consideração dados sobre o contexto geral do processo. Por exemplo, informações sobre pessoas que possam contribuir para um processo e o histórico de execuções de instâncias anteriores são úteis em processos colaborativos.

Essa visão limitada, sem atenção para toda a informação contextual, não é aceitável em processos de negócio colaborativos. Múltiplas organizações e trabalho em equipe compõem este tipo de processo e o rastreamento de atividades é particularmente importante para entender o progresso da colaboração. Dada uma situação médica específica, por exemplo, médicos e enfermeiros podem realizar um atendimento mais adequado caso

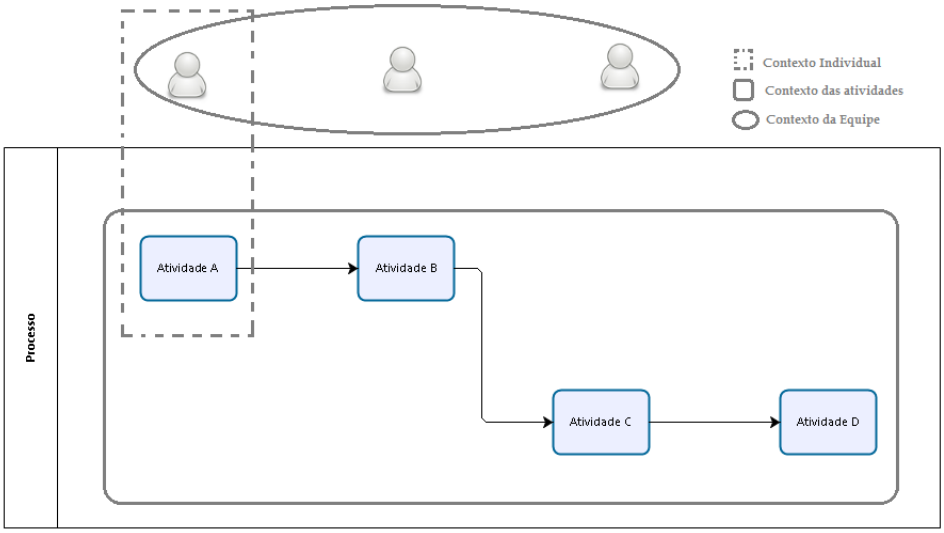


Figura 2 – Noções de contexto

tenham acesso ao registro médico completo do paciente. No caso de uma realização de cirurgia, é fundamental que o cirurgião tenha conhecimento de varias informações do paciente e não apenas àquelas diretamente relacionadas ao procedimento cirúrgico.

No entanto, controlar a execução dos processos de negócio colaborativos torna-se complexo devido à cardinalidade dos processos de negócio, número de participantes envolvidos e pela correlação entre as instâncias (LIU; LI; ZHAO, 2009). Por isso, o *Context Tunneling* representa um dos principais problemas dos *BPMS* atuais.

Embora a abordagem declarativa forneça várias características positivas, nenhuma das ferramentas atuais (DCR Graphs (HILDEBRANDT; MUKKAMALA, 2011), DECLARE (PESIC; SCHONENBERG; AALST, 2007), REFlex (SILVA et al., 2013)) oferece suporte para processos de negócio colaborativos. No DCR Graphs e Declare é possível atribuir um usuário à execução da atividade, mas o problema de *Context Tunneling* continua existindo, dado que o usuário só vai ver as atividades referentes a si próprio (sem nenhuma outra informação).

Para evitar o *Context Tunneling* algumas iniciativas podem ser tomadas para expandir esta visão: (1) integração de contextos, para dar uma compreensão completa do processo, e (2) reutilização de informações contextuais em diferentes instâncias do processo para ajudar os usuários a tomar decisões.

2.5 Software Social

Na última década as redes sociais tornaram-se um fenômeno na área de tecnologia da informação, sendo a internet um meio de longo alcance capaz de permitir a integração social. As organizações modernas trazem em sua natureza o relacionamento social nos processos de trabalho, especialmente no que diz respeito ao conhecimento dos envolvidos

no processo, que precisam compartilhar informações e ideias, construindo assim, valores de forma colaborativa e interativa (RASHID, 2009).

Como determinamos na seção anterior, nos processos de negócio, os participantes envolvidos fazem parte de um time e necessitam discutir e colaborar com os outros membros da equipe. Além disso, é importante que os envolvidos no processo possam conhecer as informações sobre execuções anteriores para modificar e melhorar suas ações no processo.

O software social favorece a integração da equipe e a colaboração, características inerentes aos processos de negócio, pois permite atravessar as barreiras de comunicação entre os membros da equipe ou de outras organizações (BAZÁN, 2015). Essas características proveem uma melhor integração entre todos os envolvidos no ciclo de vida dos processos de negócio, oferecendo novas possibilidades para uma modelagem mais efetiva e flexível.

Nesse sentido, surgiu o conceito de *Groupware*, que se refere a programas que permitem times trabalharem juntos. Esse conceito, no Modelo 3C de Colaboração, originalmente proposto por Ellis (ELLIS; GIBBS; REIN, 1991), analisa a colaboração em três dimensões (Fig. 3): comunicação, colaboração e coordenação.

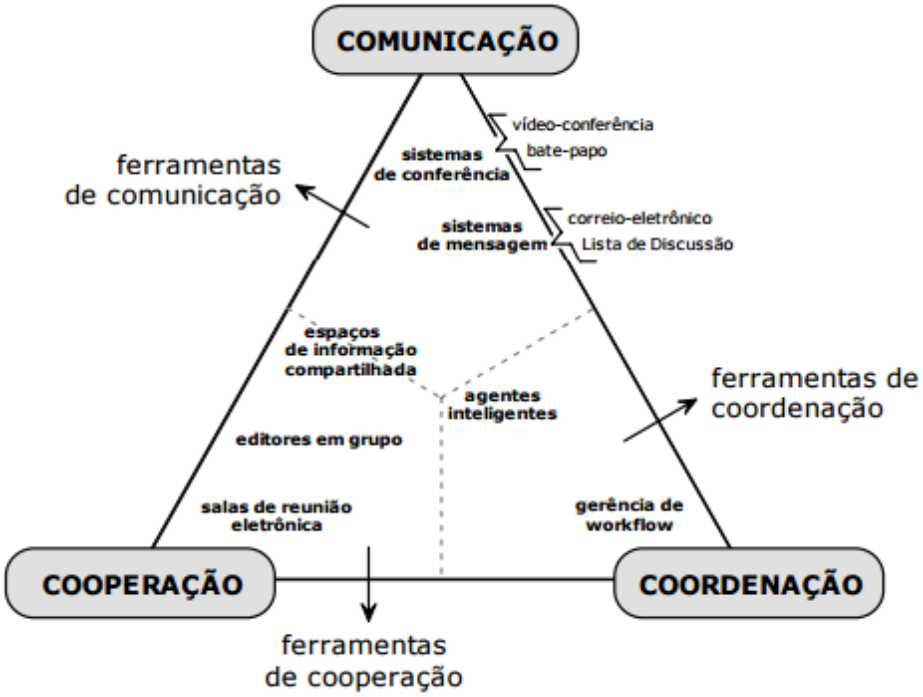


Figura 3 – Classificação 3C dos sistemas colaborativos (PIMENTEL et al., 2006)

O modelo 3C de colaboração é baseado na concepção de que, para colaborar, os membros de um grupo comunicam-se, coordenam-se e cooperam-se. Assim, esse modelo é frequentemente usado pela literatura para classificar os sistemas colaborativos, tal como os processos de negócio colaborativos.

A comunicação se refere à ação, pois enquanto se comunicam, as pessoas negociam e tomam decisões. Já a coordenação trata do gerenciamento de pessoas e/ou organização

das atividades e recursos. Na cooperação existe a ação conjunta para produzir algo e existe a necessidade de renegociação, o que demanda comunicação e coordenação para reorganizar as tarefas (GOUVÊA et al., 2016).

Relacionar os conceitos tanto de *BPM* quanto de software social possibilita refletir a realidade da maioria dos processos de negócio colaborativos das organizações atuais.

2.6 Case Management

Enquanto a maioria dos sistemas de processos de negócio são orientados a atividades (*Workflow Management Systems (WFMS)*) ou regras (declarativos), *Case Management* é um tipo de abordagem recente que baseia-se na definição de um caso para descrever um processo. Essa mudança na forma de enxergar o processo diminui a importância de uma única atividade em prol de um contexto mais amplo.

O caso é o conceito principal e define a instância de um processo, descrevendo todos os dados e informações relevantes ao contexto. Um caso pode ser definido como uma coleção de tarefas, ações, processos e qualquer outra informação importante para um processo de negócio específico (BUKHSH, 2015). As informações de contexto do caso e as ações tomadas pelos participantes são os direcionadores do fluxo de execução de todo o processo.

IBM define *Case Management* como um processo orientado a objetivos onde as pessoas devem, em tempo real, tomar decisões complexas baseadas nas mudanças da informação, muitas vezes trabalhando de forma interativa com outros participantes da mesma ou de outras organizações, para obter resultados mais eficazes (ZHU et al., 2015).

O *Case Management* coordena o fluxo do processo em resposta às ações do usuário, concentrando-se em definir o que pode em vez do que deve ser feito. Nesse cenário, os participantes tornam-se responsáveis por alcançar o objetivo do negócio. Para apoiar essa prática, a abordagem oferece acesso a todas as informações relevantes para o contexto, fornecendo aos envolvidos no processo capacidade para atuar na execução e compreender como o processo evolui (MARIN, 2016). O conhecimento dessas informações permite ao usuário visualizar o contexto global da situação, iniciar atividades e processos e manter registros de histórico globais (MOTAHARI-NEZHAD; SWENSON, 2013).

A forma como essa abordagem lida com a atribuição das atividades aos participantes também é muito mais flexível se comparada com outras existentes. O *Case Management* assume que os participantes têm uma visão completa do processo em que estão inseridos, entretanto, dependendo do papel a ele atribuído no processo, poderá processar e visualizar certas informações.

Case Management foi originalmente desenvolvido para ajudar a gerenciar o trabalho social e áreas de aplicação relacionadas (MARIN; HULL; VACULÍN, 2012). Atualmente, tornou-se um eficiente padrão para modelagem de processos de negócio colaborativos, que integram pessoas, processos e regras em uma única definição de instância.

Case Management como um paradigma para suporte de processo de negócio foi inicialmente proposto por Aalst (AALST; STOFFELE; WAMELINK, 2003) e ficou amplamente conhecido como *Case Handling Paradigm*. De acordo com outro trabalho de Aalst (AALST; WESKE; GRÜNBAUER, 2005), as principais características desse paradigma são:

- Evitar o *Context Tunneling* fornecendo todas as informações de contexto disponíveis. Ou seja, propõe apresentar o processo como um todo ao invés de mostrar apenas fragmentos;
- Definir quais atividades deverão ser realizadas com base nas informações de contexto disponíveis e não apenas de acordo com as atividades previamente executadas;
- Separar a distribuição das atividades entre os participantes do processo, permitindo adicionar novos tipos de papéis e não apenas o papel em execução;
- Permitir que os trabalhadores visualizem dados antes ou depois que as atividades correspondentes tenham sido executadas.

Além desses pontos, a inclusão dos conceitos de *Case Management* permite coletar e analisar informações sobre a execução de processos semelhantes ou atividades previamente executadas, permitindo aos usuários tirar vantagem em casos semelhantes.

2.6.1 *CMMN*

O *CMMN* foi recentemente adotado pela *Object Management Group (OMG)* como um padrão de modelagem para processos de negócio. Este define um meta-modelo e notação padronizada para modelar um caso, possibilitando, assim, a utilização desses modelos em diferentes ferramentas.

O CMMN é composto por um conjunto de artefatos que são utilizados para armazenar as informações de um dado contexto do processo. Esses artefatos são utilizados no *Case Management* para determinar o fluxo de controle da execução das atividades. Para Aalst (AALST; WESKE; GRÜNBAUER, 2005) e Vanderfeesten (VANDERFEESTEN; REIJERS; AALST, 2006), esse controle envolve também um fluxo lógico como o fluxo baseado nas atividades dos modelos tradicionais, para direcionar a sequência de atividades. Todavia, os participantes envolvidos no processo podem modificar essa sequência utilizando as informações de contexto dos artefatos disponíveis.

Existe duas fases para um modelo CMMN. Na fase de design, o modelador define um conjunto de atividades, regras que devem ser respeitadas durante a execução do caso, e itens discricionários que seão decididos em tempo de execução pelo trabalhador. Na fase de execução o trabalhador executa o modelo seguindo um fluxo predefinido, mas decidindo as atividades e sua ordem de acordo com seu conhecimento e necessidades. Além

disso, também é possível "ajustar" o caso, planejando incluindo os itens discricionários na execução.

Allah (BUKHSH, 2015) afirma que os elementos centrais do CMMN são definidos em três tipos:

1. *Case Model Element*: que introduz a noção do caso como um diretório, no qual são definidos todos os elementos a ele relacionado.

2. *Information Model Element*: arquivos que contém todos as informações de um caso específico.

3. *Plan Model Elements*: inclui eventos, tarefas, estágios, entre outros.

As definição das regras é feita a partir da definição de uma expressão. Entretanto, CMMN não define uma linguagem específica para essas expressões, nem especifica a riqueza/poder dessa linguagem. Em sua versão mais atual, o CMMN não inclui uma semântica formal em seu escopo.

Em relação à linguagem ConDec, usada pelo DECLARE, esta utiliza Lógica Temporal Linear (LTL) como formalismo para a representação interna dos processos de negócio, onde cada modelo é mapeado para uma fórmula LTL. Devido a esta formalização, uma verificação em tempo de execução pode ser feita com foco em violações das regras devido a múltiplas restrições nos modelos do processo de negócio (CARVALHO et al., 2016).

Adicionalmente, o REFlex usa a linguagem Alloy (JACKSON, 2012) sem necessitar gerar todos os possíveis estados de execução de um processo, mas apenas o estado atual de execução, que é atualizado em tempo de execução. Isto evita autômatos que crescem exponencialmente em tamanho para modelos realistas, resultando em ineficiência e elevado uso de memória. Além disso, o REFlex pode inferir estados futuros e evitar estados inconsistentes, garantindo uma representação precisa do modelo e que os processos de negócio serão executados corretamente (CARVALHO et al., 2016). Esta menor capacidade em representar regras de negócio na modelagem pelo CMMN, nos faz permanecer com a escolha inicial utilizada pelo REFlex.

2.7 Multiple-Criteria Decision-Making

Tomar decisões complexas é, de modo geral, uma tarefa difícil de ser realizada, porém, fundamental para alcançar o sucesso nos processos de negócio dentro de uma organização empresarial. No contexto dos processos de negócio colaborativos, a tomada de decisões torna-se ainda mais complexa pois envolve diversos atores, cada um deles com o seu sistema de valor e objetivos que algumas vezes representam interesses individuais.

A tomada de uma decisão em uma empresa deve levar em consideração todos os seus possíveis impactos no resultado final de um processo. Quando analisamos o processo de

atendimento de um paciente em uma unidade de urgência, por exemplo, deve-se considerar todos os possíveis fatores que tenham impacto no objetivo do processo. Nesse caso, a análise deve considerar o aspecto rapidez no atendimento, pois existem critérios subjetivos que podem afetar direta ou indiretamente o resultado final. Assim, a utilização de métodos que levem em consideração aspectos objetivos e também aspectos subjetivos é fundamental para a obtenção de um resultado confiável.

O problema de escolher dentre várias alternativas, qual delas satisfaz melhor um conjunto de critérios, é um problema típico de *Multiple-Criteria Decision-Making (MCDM)*. Para a *International Society on Multiple Criteria Decision Making*, MCDM pode ser definido como o estudo dos métodos e processos pelos quais as considerações sobre vários critérios conflitantes podem ser formalmente incorporados ao processo de planejamento de gestão (FILHO, 2011).

O processo decisório não deve ser realizado de maneira intuitiva, e sim estar apoiado em métodos multicritério como instrumento de apoio a tomada de decisão. Esses métodos são um importante conjunto de ferramentas para abordar decisões complexas em organizações porque auxiliam os gestores, em situações de incertezas, complexidade e objetivos conflitantes (WANG, 2010).

O *AHP* é uma técnica MCDM, originalmente introduzida por Thomas Saaty na década de 1970, que leva em conta ambos os critérios objetivos e subjetivos para fornecer um ranque das alternativas (FILHO, 2011). Para Favrett (FAVRETTO; NOTTAR, 2016), AHP é um método eficaz para a tomada de decisão e que possibilita ao decisor identificar a melhor opção dentro das múltiplas alternativas possíveis, ajudando-o na determinação das prioridades. Por sua vez, Ubando (UBANDO; GUE; AGUILAR, 2016) cita AHP como uma abordagem amplamente utilizada com bons resultados em vários campos do conhecimento. Este método é baseado em três princípios:

- Construção de hierarquias: o problema de escolha é decomposto em níveis hierárquicos, como forma de buscar uma melhor compreensão e avaliação do mesmo;
- Priorização: fundamenta-se na habilidade do ser humano de perceber o relacionamento entre os elementos, comparando-os em pares em relação a um determinado foco ou critério;
- Consistência lógica: é possível avaliar o modelo de priorização construído quanto à sua consistência.

2.7.1 Construção da hierarquia

A primeira etapa do método AHP consiste na decomposição do problema/decisão em uma hierarquia. A Figura 4 apresenta a estrutura hierárquica padrão do AHP, que deve ser composta, no mínimo, de:

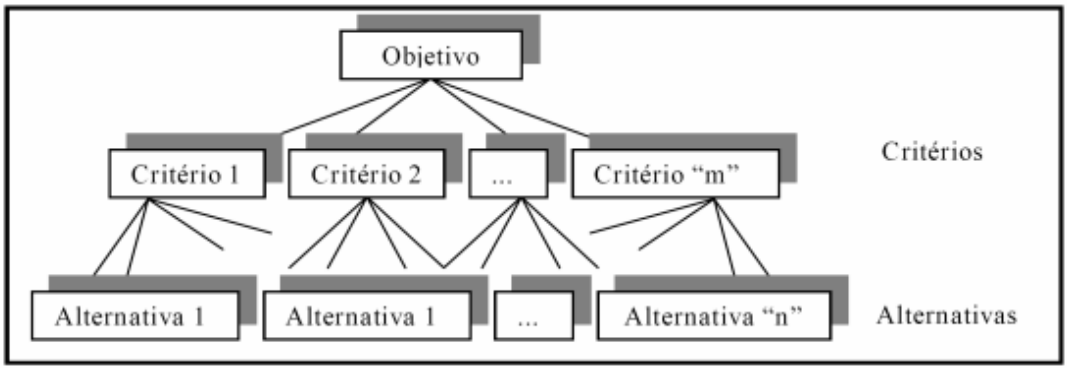


Figura 4 – Hierarquia AHP (Fonte: MARIN et al. (MARINS; SOUZA; BARROS, 2009))

- um objetivo, que se localiza no nível mais alto da hierarquia e representa a meta que se pretende alcançar;
- critérios (rapidez no atendimento, conforto do cliente, entre outros), que são definidos nos níveis intermediários, influenciam no processo para que seja possível alcançar o objetivo final;
- e por fim, possíveis alternativas disponíveis dentre o universo das variedades disponíveis.

Essa estruturação permite que os usuários possam decompor seu problema em uma hierarquia de subproblemas, mais facilmente compreendidos, sendo que cada qual pode ser analisado independentemente.

2.7.2 Priorização

A segunda etapa consiste em estabelecer prioridades entre os elementos para cada nível da hierarquia. Para isso, a técnica AHP realiza comparação par a par entre critérios do mesmo nível, onde são avaliadas as preferências relativas entre cada elemento.

O primeiro ponto a ser considerado é a determinação de uma escala de valores para essas comparações. Na AHP, a forma mais utilizada para definir os pesos e realizar as comparações entre os elementos é a partir da escala de relativa importância entre duas alternativas proposta por Saaty (SAATY, 2005). Atribuindo valores que variam entre 1 e 9, a escala determina a importância relativa de um elemento com relação ao outro, conforme apresentado na Tabela 1.

Em geral, a comparação entre os elementos segue a lógica de: "quanto o elemento X é importante quando comparado com o elemento Z". Essas comparações normalmente tomam a forma de uma matriz, conhecida como matriz de comparação, que como podemos ver na Tabela 2 compara dois critérios e atribui a eles os valores da Escala Saaty.

Essa análise deve ser feita para cada nível da hierarquia, ou seja, os subcritérios existentes para cada um dos critérios considerados também devem passar pela mesma forma

Tabela 1 – Escala de relativa importância de Saaty (Adaptado: Vargas (VARGAS; IPMA-B, 2010))

ESCALA	AV. NUMÉRICA	RECÍPROCO
Igualmente preferido	1	1
Moderadamente preferido	3	1/3
Fortemente preferido	5	1/5
Muito fortemente preferido	7	1/7
Extremamente preferido	9	1/9
2, 4, 6 e 8	São valores intermediários	-

Tabela 2 – Matriz comparativa AHP (Fonte: Favretto e Nottar (FAVRETTO; NOTTAR, 2016))

	CRITÉRIO 1	CRITÉRIO 2
Critério 1	1	Avaliação Numérica
Critério 2	1/Avaliação Numérica (Recíproco)	1

de comparação, com a mesma escala de valores. Para obter a prioridade relativa de cada critério é necessário:

1. normalizar os valores da matriz de comparações – tem por objetivo igualar todos os critérios a uma mesma unidade, para isto cada valor da matriz é dividido pelo total da sua respectiva coluna (Tabela 3 e Tabela 4);

2. obter o vetor de prioridades – também conhecido como vetor de Eigen, tem por objetivo identificar a ordem de importância de cada critério, para isto é calculado a média aritmética dos valores de cada linha da matriz normalizada obtida no item anterior (Tabela 5).

Tabela 3 – Processo de normalização: matriz comparativa original

	CRITÉRIO A	CRITÉRIO B	CRITÉRIO C
Critério A	1	1/2	3
Critério B	2	1	4
Critério C	1/3	1/4	1
Somatório	10/3	7/4	8

2.7.3 Consistência lógica

Na sequência, faz-se necessário verificar a consistência lógica dos resultados obtidos. A verificação visa captar se os tomadores de decisão foram consistentes nas suas opiniões para a tomada de decisão. No entanto, o índice de inconsistência somente permite a

Tabela 4 – Processo de normalização: matriz comparativa normalizada

	CRITÉRIO A	CRITÉRIO B	CRITÉRIO C
Critério A	3/10	2/7	3/8
Critério B	3/5	4/7	1/2
Critério C	1/10	1/7	1/8

Tabela 5 – Matriz normalizada com vetor de prioridades

	CRITÉRIO A	CRITÉRIO B	CRITÉRIO C	PRIORIDADE RELATIVA
Critério A	3/10	2/7	3/8	0,3202
Critério B	3/5	4/7	1/2	0,5571
Critério C	1/10	1/7	1/8	0,1226

$$CI = \frac{\lambda_M ax - n}{n - 1}$$

Figura 5 – CI é o índice de consistência, n é o número de critérios avaliados e $\lambda_M ax$ que é o número principal de Eigen

avaliação da consistência e a regularidade das opiniões dos tomadores de decisão e não avalia se essas opiniões são as mais adequadas para o contexto organizacional (VARGAS; IPMA-B, 2010).

O cálculo do índice de consistência, definido por Saaty ((SAATY, 2005)), é dado pela seguinte equação:

O índice de consistência tem como base o número principal de Eigen. Ele é calculado através do somatório do produto de cada elemento do vetor de Eigen (Tabela 5) pelo total da respectiva coluna da matriz comparativa original (Tabela 3). A Tabela 6 apresenta o cálculo do número principal de Eigen.

Tabela 6 – Cálculo do número principal de Eigen

	CRITÉRIO A	CRITÉRIO B	CRITÉRIO C
Vetor Eigen	0,3202	0,5571	0,1226
Total	10/3	7/4	8
Número principal de Eigen	1,0673	0,9749	0,9808

Por fim, visando verificar se o valor encontrado do índice de consistência (CI) é adequado, Saaty ((SAATY, 2005)) propôs o que foi chamado de taxa de consistência (CR). Ela é determinada pela razão entre o valor do índice de consistência (CI) e o índice de consistência aleatória (RI), conforme podemos ver na equação a seguir:

$$CR = \frac{CI}{RI}$$

O resultado da matriz só será considerado consistente se a taxa de consistência for menor que 10%. O valor de RI é fixo e tem como base o número de critérios avaliados, conforme Tabela 7.

Tabela 7 – Tabela de índices RI (Fonte: Vargas (VARGAS; IPMA-B, 2010))

Dimensão da matriz (n)	1	2	3	4	5	6	7	8	9	10
Valor de RI	0	0	0,58	0,9	1,12	1,24	1,32	1,41	1,45	1,49

2.7.4 Avaliação das alternativas prioritárias

A partir do momento em que todas as comparações foram efetuadas e os pesos relativos entre os critérios a serem avaliados foram estabelecidos, a prioridade relativa de cada uma das alternativas é calculada. Da mesma forma que foi realizada para a priorização dos critérios, as alternativas são confrontadas duas a duas em relação a cada um dos critérios estabelecidos. A Tabela 9 representa a matriz de comparação de todas as possíveis alternativas definidas na hierarquia, em relação a um critério específico.

Tabela 8 – Matriz comparativa - Alternativas versus Critério em análise

	ALTERN. 1	ALTERN. 2	ALTERN. 3	ALTERN. 4
ALTERN. 1	1	1/3	1/7	9
ALTERN. 2	3	1	1/5	3
ALTERN. 3	7	5	1	1/3
ALTERN. 4	1/9	1/3	3	1

Assim como nos cálculos para critérios, o processo de normalização e da prioridade relativa são realizados para cada uma das alternativas. Por fim, o cruzamento entre todas as alternativas em todos os critérios determina a prioridade final de cada uma delas em relação à meta. O cálculo da prioridade final é determinado pelo somatório dos produtos entre o peso de prioridade da alternativa e o peso do critério (GOMEDE; BARROS, 2012).

A Tabela 9 mostra o exemplo de um cálculo para a determinada ALTERNATIVA X, baseando-se nos pesos dos critérios e nos pesos resultantes das comparações entre essa alternativa e seu critério, já encontrado anteriormente.

Tabela 9 – Avaliação final para a ALTERNATIVA X

CRITÉRIO	PESO CRITÉRIO	PESO ALTERNATIVA	PRODUTO
CRITÉRIO A	0,3202	PA1	0,3202*PA1
CRITÉRIO B	0,5571	PA2	0,5571*PA2
CRITÉRIO C	0,1226	PA3	0,1226*PA3

Essa prioridade determina a probabilidade que a alternativa tem de atender a meta estabelecida. Quanto maior a probabilidade, mais aquela alternativa contribui para a meta final.

2.8 Trabalhos Relacionados

Esta seção tem como objetivo apresentar os trabalhos que estão diretamente relacionados ao tema abordado neste trabalho. Foram descritos três trabalhos que se relacionam e suas comparações:

- *FLOWer* (ATHENA, 2002) é uma ferramenta para gerenciamento de casos. Os principais componentes dessa ferramenta são: *FLOWer Studio*, *FLOWer Management* e *FLOWer Case Guide*. Os dois primeiros são utilizados para definição dos processos e da organização no qual está inserido. Esses relacionamentos permitem associar as tarefas aos perfis de acesso definidos na organização, o que permite limitar quais trabalhadores podem ou não realizar uma tarefa. O compartilhamento de informações contextuais é realizado durante o gerenciamento das atividades através do *FLOWer Case Guide*. Nele o usuário tem uma visão geral do caso e pode preencher arquivos de dados (fomulários, texto livre) com informações relevantes ao caso.

- *DCR Graphs* (HILDEBRANDT; MUKKAMALA, 2011) possui uma versão web que possibilita simular o fluxo de atividades dos gráficos DCR. Característica essencial para verificar a precisão dos modelos de processo. A simulação inclui usuários controlados por computador, que podem receber diferentes papéis, sendo possível simular de forma individual ou colaborativa. É possível definir múltiplas instâncias e compartilhamento com outros usuários, além de fornecer mecanismos de troca de mensagens entre os participantes. Possui ferramentas de análise dos resultados da simulação dos modelo, que poderão ser utilizados pela equipe para posteriores melhorias. o reaproveitamento dessa análise, entretanto não é disponível em tempo de execução.

- *SocialFlow* (BAZÁN, 2015) é uma aplicação web que permite a execução de processos distribuídos com suporte à interação entre os participantes. Para isso, realiza o controle do fluxo dos processos e suas distintas instâncias segundo a ordem e estado das atividades que o compõem e as permissões de acesso dadas aos usuários envolvidos no processo. Para controle do fluxo a aplicação utiliza a ferramenta *Bonita Open Solution* (SOLUTION, 2009) como motor de processos. Em relação a integração entre os participantes, o *SocialFlow* propõe a inclusão de aspectos sociais, e para isso, permite incorporar comentários a cada instancia dos processos, manter conversas entre os participantes, etiquetar participantes e notificar os participantes sobre a ocorrência dos eventos.

- *IBM Case Manager on Cloud* (IBM, 2016) é uma plataforma construída baseada no conceito de casos, uma coleção de informações e tarefas, através do conhecimento dos trabalhadores para alcançar o objetivo desejado. Ela fornece informações contextuais através de análises em tempo real e aprimora o trabalho da equipe por meio de funcionalidades de colaboração. Uma forma de contexto utilizada é o armazenamento de arquivos (documentos, emails, imagens e vídeos) para cada caso, onde esses dados ficam salvos após a conclusão do caso, para fins de auditoria. O histórico é baseado nas operações já realizadas no caso, como: *upload* de arquivos e *log* de execução de atividades.

2.8.1 Quadro Comparativo

Diante dos trabalhos relacionados acima, é possível verificar a diferença entre estes e a proposta deste trabalho na Tabela 10.

Através da nossa análise, verificamos que o *FLOWer* não disponibiliza acesso concorrente para seus usuários, característica que julgamos essencial para permitir o compartilhamento de informações e o trabalho colaborativo. Embora todas apresentem Informações Contextuais, as ferramentas *FLOWer*, *DCR Graphs* e *SocialFlow* não reutilizam essas informações (Realimentação Contextual) para ajudar os usuários durante o processamento de instâncias similares. Também, nenhuma delas utiliza informações contextuais para recomendar a execução de uma tarefa, ou seja, priorizar uma atividade em relação a outra. Cada ferramenta apresentada possui características particulares e podem contribuir com as necessidades dos processos de negócio colaborativos.

Tabela 10 – Quadro comparativo entre os trabalhos relacionados

	Modelagem do Processo	Acesso Concorrente	Múltiplas instâncias	Informação Contextual	Realimentação Contextual	Recomendação de tarefas
FLOWer	Workflow	Não	Não	Sim	Não	Não
IBM Case Manager	Workflow	Sim	Sim	Sim	Sim	Não
SocialFlow	Workflow	Sim	Sim	Sim	Não	Não
DRC Graphs	Declarativa	Sim	Sim	Sim	Não	Não
Web-REFlex	Declarativa	Sim	Sim	Sim	Sim	Sim

2.9 Síntese do capítulo

De maneira geral, esse capítulo apresentou as informações necessárias para compreensão dos conceitos abordados nesse trabalho e do *background* teórico da dissertação. O conhecimento desses conceitos são necessários ao entendimento da ferramenta proposta. Na seção 2.1, apresentamos os conceitos sobre a importância da utilização dos *BPMS* para as organizações empresariais. Na seção 2.2, apresentamos como processos de negócio declarativos trazem maior flexibilidade aos BPMS tradicionais. A seção 2.3 introduz o problema de *Context Tunneling* para os processos de negócio. A seção 2.4 aborda conceitos de software social, que introduz a comunicação, colaboração e coordenação em sistemas de software. Na seção 2.5 apresentamos o paradigma *Case Management*, um conceito que permitiu analisar os processos baseados em casos, permitindo que os envolvidos no sistema tenha uma visão geral da instância do processo e de execuções anteriores para processos semelhantes. A seção 2.6 apresenta detalhes sobre o MCDM e a técnica implementada por este trabalho, a *AHP*. E, por fim, a seção 2.7 apresenta os trabalhos que estão diretamente relacionados ao tema abordado neste trabalho.

O Web-REFlex, uma ferramenta ambientada na nuvem, utiliza o paradigma *Case Handling* aliado a características do software social para atuar na fase de execução de um processo de negócio declarativo. Nesse contexto, a ferramenta faz uso do REFlex, uma *engine* para processos declarativos, que será detalhado no próximo capítulo, para gerenciar e atualizar o status de atividades e instâncias de processos.

3 *REFLEX*

Neste capítulo apresentamos o REFlex (CARVALHO et al., 2013), framework utilizado por este trabalho para gerenciar a execução de processos declarativos. Este é composto por um motor de regras (*Rule Engine*) e um orquestrador de serviços declarativos capaz de chamar serviços externos para executar tarefas automatizadas, como serviços Web (*Orchestrator*).

O REFlex possui quatro objetivos principais: 1) permitir a execução de atividades que não violem restrições de regras; 2) desativar a execução de atividades que temporariamente violam restrições de regras; 3) bloquear a execução de atividades que violem permanentemente restrições de regras; e 4) impossibilitar a finalização do processo até que todas as atividades obrigatórias ao processo tenham sido executadas (CARVALHO, 2015).

3.1 REFlex Rule Engine

O motor de regras é responsável pela validação das regras de negócio e deve garantir que todas elas sejam respeitadas até o final da execução. Dessa forma, evita que o usuário execute atividades que violam as exigências do processo e obriga a execução das atividades pendentes para a finalização do processo. Também garante que o processo nunca alcance um estado de *deadlock*, em que atividades pendentes não possam ser executadas. Este motor controla a execução de processos declarativos e faz a verificação das regras do processo sem a geração de todos os caminhos de execução de atividades possíveis, evitando o problema da explosão de estados.

O mecanismo de regras do REFlex é composto basicamente por 2 fases: modelagem e execução. A figura 6 apresenta o esquema de mecanismo de regra REFlex para gerenciar e atualizar o status das atividades.

Durante a fase de modelagem, o usuário especifica um processo de negócio definindo as atividades e regras de negócio. Este modelo é então compilado em uma linguagem gráfica, por meio de grafos, que é utilizada pela Engine para interpretar as regras.

Uma vez que o REFlex inicia a execução do processo, ele interage com o usuário para mostrar as atividades disponíveis no momento inicial da execução. Quando o usuário executa uma atividade, o REFlex interpreta dinamicamente o estado atual e as regras de negócio para determinar o próximo estado do processo, apresentando na interface do usuário os novos status das atividades.

3.1.1 Semântica e Estrutura de Grafos

A definição das regras de negócio é baseada em grafo para representação do estado atual do processo e são verificadas em tempo de execução. O estado do grafo é atualizado

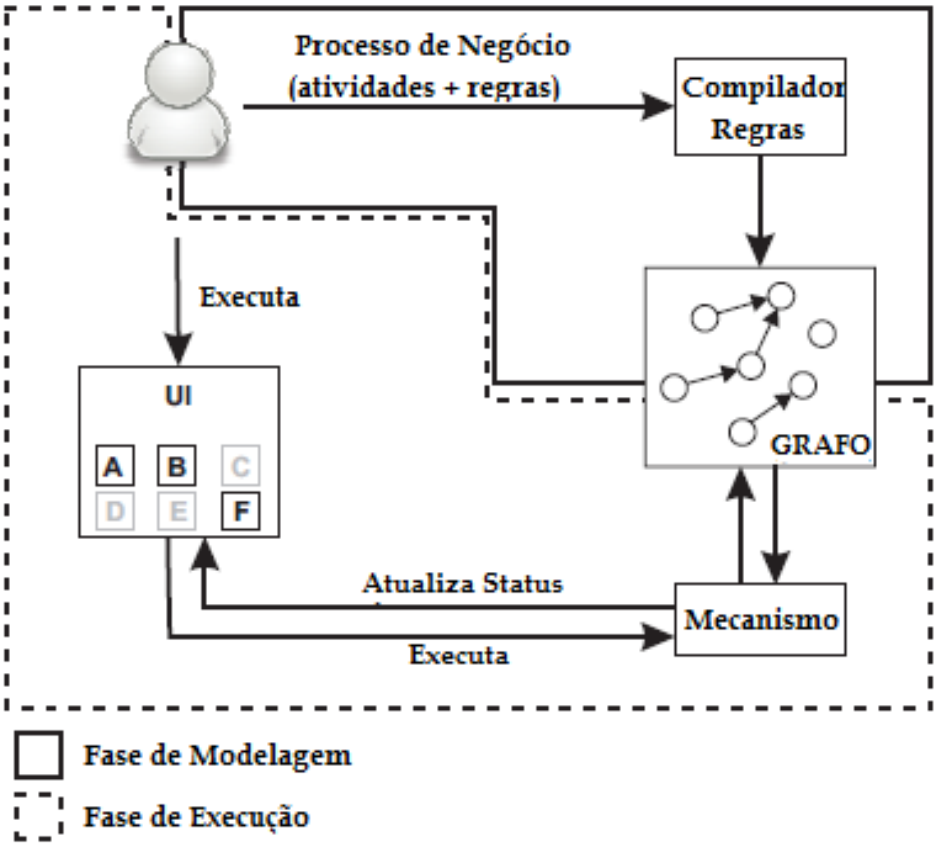


Figura 6 – Esquema de funcionamento do mecanismo de regras do REFlex (Adaptado: Renata et al. (CARVALHO et al., 2013))

dinamicamente para garantir que todas as regras sejam cumpridas e represente o estado do processo no momento atual da execução.

Os grafos do REFlex têm dois elementos básicos: nós, representam as **atividades**, e arestas, definem as **regras (relacionamentos)** entre as atividades.

3.1.2 Atividades

Cada atividade do sistema é representada em REFlex como um nó do grafo. A representação de uma atividade também incluem algumas propriedades, como pode ser visto nas próximas subseções.

3.1.2.1 Estados

A cada atividade do processo é atribuído um estado, representados graficamente na figura 7. Durante o tempo de execução, os estados das atividades são atualizados dependendo das ações dos usuários.

O estado das atividades pode ser:

1. **enabled**, quando não viola nenhuma das regras de negócio definidas. Podendo ser executadas no momento corrente do fluxo do processo.

2. **disabled**, quando uma atividade viola temporariamente uma das regras de negócio. Essa atividade poderá ser reativada posteriormente de acordo com as regras definidas para os próximos passos da execução do processo.

3. **blocked**, quando a atividade viola permanentemente uma das regras de negócio. Essa atividade não poderá ser desbloqueada no decorrer do fluxo restante.

Os estados das atividades podem estar associados a propriedade **obliged**, que determina que a atividade deve ser obrigatoriamente executada para que o processo seja finalizado. Essas atividades fazem parte de um conjunto de atividades chamado de "atividades pendentes".

Na representação gráfica, uma atividade obrigatória possui um círculo extra envolvendo-a para indicar essa obrigatoriedade.

(a) (b) (c) (d) (e)

Figura 7 – Representação gráfica dos estados das atividades: (a) *Enabled*, (b) *Disabled*, (c) *Blocked*, (d) *Obliged and Enabled* e (e) *Obliged and Disabled* (Adaptado: Renata et al. (CARVALHO et al., 2013))

3.1.2.2 Condição de existência

As atividades de um processo podem estar associadas a uma condição de existência que determina se uma atividade está: (1) ativa, quando é levada em consideração no processo e pode assumir alguns dos estados da atividade (enabled, disabled, blocked, obliged); ou (2) inativa, quando a atividade é ignorada na execução do processo e todas as regras associadas a ela também devem ser desativadas.

A condição de existência é resultante da condição ***"activity IF predicate"*** (CARVALHO, 2015), onde o predicado é uma expressão que pode assumir os valores verdadeiro ou falso.

3.1.3 Regras

No REFlex cada aresta representa uma relação (regra) entre os nós (atividades), como na Figura 8. Tais relações expressam as regras que definem o comportamento do processo.

Uma única regra pode ser expressa por mais de uma aresta. Assim como as atividades, os relacionamentos possuem propriedades que serão detalhadas no restante desta seção.

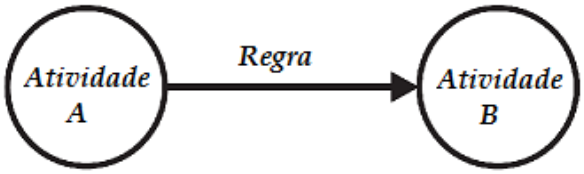


Figura 8 – Representação gráfica do relacionamento entre atividades e regras no REFlex

3.1.3.1 Atividade inicial e destino

Cada aresta tem exatamente uma atividade inicial (fonte) e uma atividade de destino. A relação expressa pela aresta expressa o efeito resultante da aplicação da regra nessas atividades. No caso da Figura 8, a **Atividade A** representa a atividade inicial e a **Atividade B** a atividade destino.

3.1.3.2 Tipos de regra

Cada relacionamento possui um tipo de regra associado, e cada um deles caracteriza um comportamento diferente. A Figura 9 apresenta a representação gráfica para todos esses tipos de regras disponíveis no REFlex.

Obligation (Fig. 9(a)): A regra de obrigação define que sempre que uma determinada atividade A for executada, a atividade B também deve ser eventualmente executada depois de A. Essa obrigação é válida para todas as execuções de A que não são seguidas de pelo menos uma execução de B.

Temporary Obligation (Fig. 9(b)): Esta regra estabelece que, se uma determinada atividade A é executada, a atividade B também deve ser executada, antes ou depois de A. Diferentemente do tipo anterior, essa obrigação só precisa ser validada uma vez durante toda a execução do processo.

Blocking (Fig. 9(c)): Define que se uma determinada atividade A é executada, a atividade B não pode mais ser executada. Em outras palavras, a execução da atividade A bloqueia a execução da atividade B.

Temporary Blocking (Fig. 9(d)): As regras de bloqueio temporário indicam que, enquanto a atividade A não seja executada, a atividade B também não poderá ser executada. Neste caso, a execução de A é uma pré-condição para a execução de B.

At Least (Fig. 9(e)): Essa regra possui um valor n associado, que indica a quantidade de vezes mínima que essa regra deve ser executada. Uma atividade A deve ser executada pelo menos n vezes. Quando A é executado: se o valor de n é maior que 1, o valor de n é decrescido de 1 e a atividade A mantém-se como obrigatória; caso contrário, o estado da relação é definido como bloqueado e a atividade A deixa de ser obrigatória.

At Most (Fig. 9(f)): Assim como o tipo anterior, essa regra possui um valor n associado. Porém esse valor agora indica a quantidade máxima de vezes que essa regra deve ser executada. Uma atividade A deve ser executada no máximo n vezes. Quando a atividade A é executada: se n for maior que 1, o valor de n é decrescido de 1; caso contrário, o estado de A é definido como bloqueado.

Precedent Obliged (Fig. 9(g)): Essa regra é a única em que seu comportamento depende do estado da atividade inicial (precedente). Define que enquanto uma determinada atividade A é obrigatória, a atividade B deve permanecer como desabilitada. Sendo liberada apenas quando a atividade A for executada.

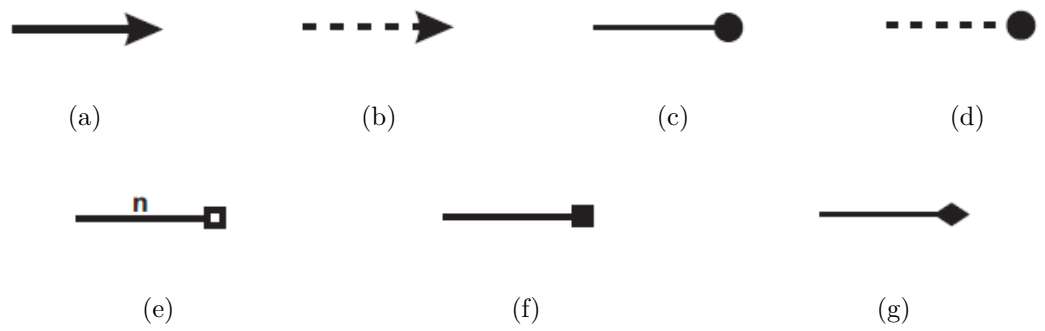


Figura 9 – Representação gráfica dos tipos de regras: (a) *Obligation*, (b) *Temporary Obligation*, (c) *Blocking*, (d) *Temporary Blocking*, (e) *At Least*, (f) *At Most*, (g) *Precedent Obliged* (Adaptado: Renata et al. (CARVALHO et al., 2013))

3.1.3.3 Peso

Para cada regra (aresta) definida é possível associar um valor, chamado de peso, que é representado por um número inteiro. Esse valor é usado para quantificar qualquer aspecto que a regra precisa gerenciar. Para exemplificar, uma aresta específica pode ter seu peso diminuído toda vez que um determinado evento é executado no processo, sendo removido quando seu peso atinge zero.

3.1.3.4 Existence condition

Essa condição de existência determina quando a regra deve ser considerada ou não no processo. É determinada por uma expressão ***"rule IF predicate"***, onde o predicado é definido por uma expressão que pode ser verdadeira ou falsa.

3.1.3.5 Estados

A condição de existência define os possíveis estados de uma regra, que podem ser:

- Uma relação *enabled* faz parte do processo e deve ser verificada. Ocorre quando a condição de existência não existe ou é verdadeira.

- Um relacionamento *disabled* não faz parte do processo, porém pode eventualmente ser ativado novamente. Nesse caso a condição de existência é falsa.
- Um relacionamento *blocked* não faz parte do processo e nunca poderá ser reativado até o final do processo. Independe da condição de existência e ocorre quando uma regra já foi finalizada.

3.2 Caso de Exemplo

Para entender o funcionamento do mecanismo de regras REFlex apresentamos o caso representado pelo gráfico da Figura 10. É importante ressaltar que o estado do gráfico muda durante a execução do processo. Portanto, o estado atual do gráfico está relacionado a um momento específico na execução.

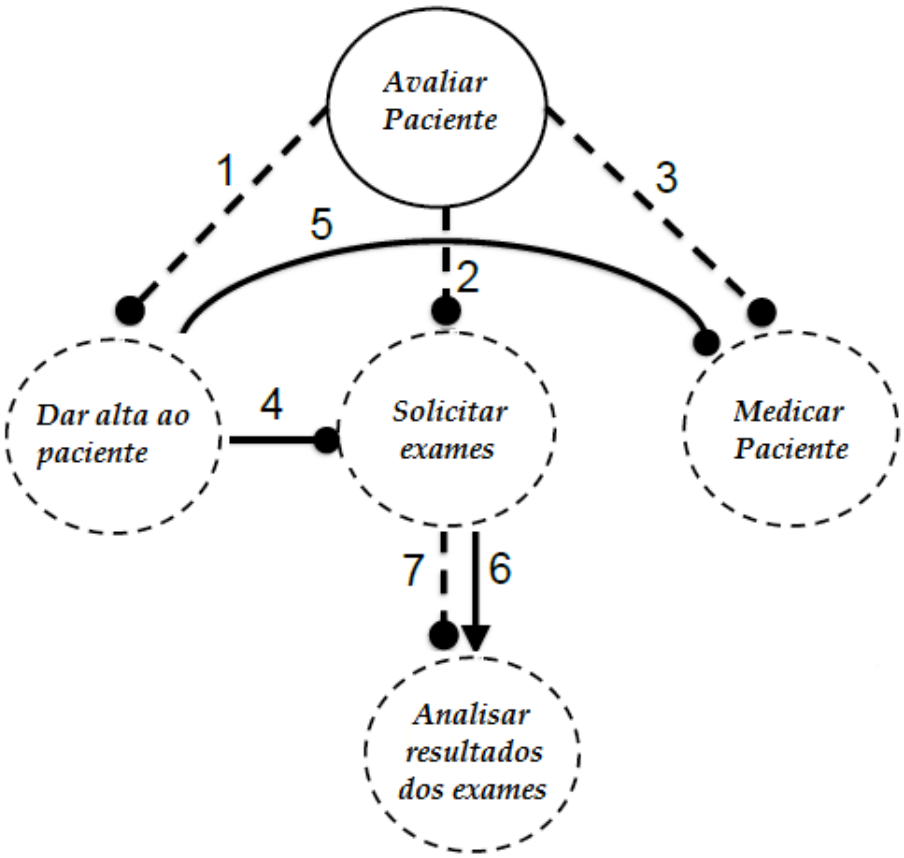


Figura 10 – Exemplo de modelo no REFlex (Adaptado: Silva (SILVA, 2014))

Esse modelo representa as atividades que poderão ser executadas por um médico no processo de atendimento a um paciente. As arestas (enumeradas) representam as regras de negócio associadas a essas atividades. Conhecendo a representação gráfica do modelo podemos entender facilmente o fluxo de execução desse processo.

Quando o processo inicia, as regras de negócio 1, 2 e 3, que são do tipo *Temporary Blocking*, determinam que apenas a atividade *Avaliar Paciente* está disponível ao médico. Ou seja, as atividades *Dar alta ao paciente*, *Solicitar exames* e *Medicar Paciente* só estarão disponíveis após realizada a avaliação do paciente. Enquanto isso, a atividade *Analisar Resultados dos exames* só será ativada (regra 7) e se tornará obrigatória (regra 6) caso a atividade *Solicitar exames* seja executada.

Após executar a atividade *Avaliar Paciente* o médico poderá decidir por executar as 3 atividades que agora estão disponíveis (*Dar alta ao paciente*, *Solicitar exames* e *Medicar Paciente*). Caso o médico opte por *Dar alta ao paciente*, as atividades *Solicitar exames* e *Medicar Paciente* ficarão bloqueadas e não poderão mais ser realizadas. Caso opte por *Medicar Paciente* o médico pode finalizar o processo após realizar a atividade. Porém caso opte por realizar a atividade *Solicitar exames*, o médico será obrigado (regra 6) a *Analisar resultados dos exames* antes de finalizar o processo de atendimento. Isso ocorre porque o usuário não tem permissão para concluir a execução do processo enquanto houver atividades pendentes.

3.3 Liveness-Enforcement

A preocupação com possíveis situações de *deadlock* é comum ao gerenciamento de processos de negócios declarativos. Para garantir que situações inaceitáveis nunca sejam alcançadas, REFlex utiliza um algoritmo *liveness-enforcing* capaz de evitar que o usuário possa conduzir equivocadamente a execução para tais situações.

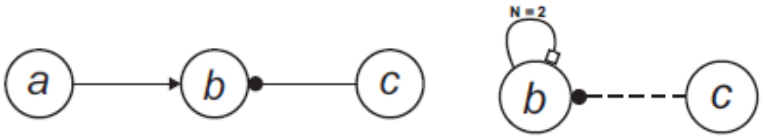
O entendimento de como essas situações são tratadas pelo algoritmo foi muito importante durante o desenvolvimento da ferramenta proposta neste trabalho, pois garante que, mesmo com as novas funcionalidades criadas, a execução do processo nunca chegue a uma situação de *deadlock*.

A seguir as possíveis situações tratadas pelo algoritmo:

- Uma atividade pode, *possivelmente*, chegar a uma situação de *deadlock* se $\exists\ A\ \in$ *atividades* $\mid\ a\ \in$ *obliged e* $A\ \in$ *disabled*.
- Uma atividade está em *deadlock* quando $\exists\ A\ \in$ *atividades* $\mid\ a\ \in$ *obliged e* $A\ \in$ *blocked*. Na figura 11 é possível ver exemplos dessa situação.

Em seu trabalho, Carvalho (CARVALHO, 2015) define todas as situações de *deadlock* existentes na Engine. Para exemplificar, a figura 11 apresenta dois casos.

- Figura 11(a) - se ambas as atividade a e c forem executadas, b será, ao mesmo tempo, *obligated* e *blocked*;
- Figura 11(b) - a atividade b é inicialmente *obligated* devido à regra *At Least* e *disabled* devido à regra *Temporary Obligation*. Nesse caso, se c for bloqueada antes



(a) (b)

Figura 11 – Situações de *deadlock*

de ser executada, b não será mais habilitada impedindo a execução do processo de terminar.

3.3.1 Algoritimo *liveness-enforcement*

O objetivo do algoritimo é evitar todas as situações que tornem, simultaneamente, uma atividade *blocked* e *disabled*. Para isso, algumas restricões são utilizadas para desativar ou bloquear atividades que podem levar o usuário a esse tipo de situação.A execução do modelo sempre ocorre apenas após o algoritmo analisar o modelo e inserir as regras necessárias para remover possíveis *deadlocks*.

3.3.1.1 *Blocking Propagation*

Se uma atividade estiver *blocked*, então todas as atividades que poderão torna-la *obligated* em algum momento da execução também devem ser bloqueadas. Caso contrário, a atividade pode chegar aos estados *blocked* e *obligated* ao mesmo tempo, causando um impasse. Para isso, o algoritmo adiciona uma nova regra *blocking* (Figura 12).

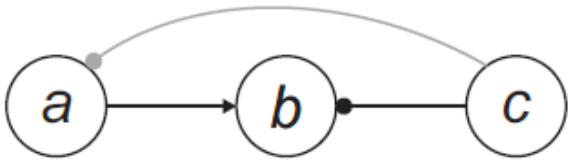


Figura 12 – Ajuste *Blocking Propagation*

3.3.1.2 *Enabling Obliged*

Se uma atividade é *Obligated*, enquanto A continuar *Obligated* todas as atividades cuja execução pode bloqueá-la devem ser desabilitadas. Para isso o REFlex utiliza uma função que define se a atividade está *enabled* ou *disabled*. Para que todas as situações de *deadlock* sejam atingidas a função é:

$$E(A) = r, S : r = (S, A, TEMPORARY BLOCKING, WI, C) \wedge (C \ is \ TRUE)$$

$$r_2, T_2 : r_2 = (A, T_2, BLOCKING, W_{I2}, C_2) \wedge PENDING(T_2) \wedge (C_2\ is\ TRUE)$$

$$r_3 : r_3 = (A, A, ATMOST, W_{I3}, C_3) \wedge (C_3\ is\ TRUE))$$

$$r_4, S_4 : r_4 = (S_4, A, OBLIGATION, W_{I4}, C_4) \wedge W(r_3) = 1 \wedge PENDING(S_4) \wedge (C_4\ is\ TRUE)$$

3.3.1.3 *Enabling Pre-Conditions*

Se uma atividade *a* é *obligated*, todas as atividades que são precedentes à sua execução também devem ser executadas. Assim, tornando as atividades precedentes *obligated* é possível certificar-se de que a execução do processo pode terminar. No exemplo da figura 13(a), se *b* for *obligated*, *c* deve ser executada para remover a regra *temporary obligation* e permitir a execução de *b*. Para isso, o ajuste adiciona uma nova regra de *a* para o destino *c*. O novo arco garante que todas as atividades que podem tornar *b* obrigatória irão também garantir que *c* seja executada pelo menos uma vez.

(a) (b)

Figura 13 – Exemplo do ajuste *Enabling Pre-Conditions*

3.3.2 Regras de Inferência

Como vimos, o algoritmo descrito analisa a estrutura do grafo para decidir quais regras devem ser adicionados ao modelo. O problema é que existem algumas relações indiretas entre as atividades que não são expressas diretamente por suas regras. No entanto, é possível inferir essas relações e adicionar novas regras ao modelo, sem alterar o comportamento original do processo.

3.3.2.1 Inferência 1

Identifica a situação em que uma atividade nunca pode ser *enabled* porque uma de suas pré-condições está *blocked*. No exemplo da figura 14, devido a regra *Temporary Blocking* da atividade *b* para *a*, se *b* for bloqueada, *a* nunca será ativada.

Para evitar situações como estas, é preciso adicionar novas regras *Blocking*. Estas garantem que todas as atividades que podem bloquear *b* também bloquearão *a*.

3.3.2.2 Inferência 2

Identifica a propagação de atividades obrigatórias. Se *a* é *obligated*, então deve ser executada até o final da execução da instância do processo. Além disso, caso a execução desta

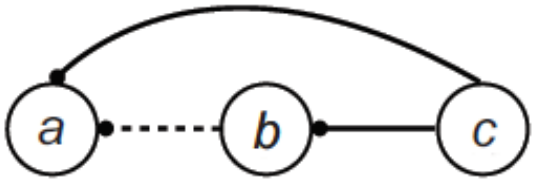


Figura 14 – Exemplo da Inferência 1

atividade torne outras *obligated*, elas também terão que ser obrigatoriamente executadas. Portanto, é uma relação transitiva.

Na figura 15 quando *a* é executada, B torna-se *obligated*. Dessa forma, em algum momento *b* será obrigatoriamente executada e então *c* também será *obligated*. Nessa situação, *c* também poderia ser marcado como *obligated* imediatamente após a execução de *a*. Para garantir essa relação transitiva, é inserida uma nova regra *Obligation* de *a* para *c*.

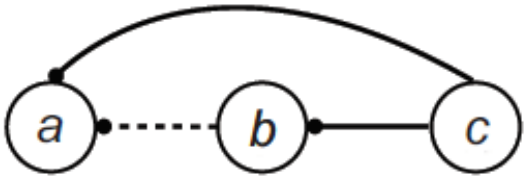


Figura 15 – Exemplo da Inferência 2

3.4 *Data-Aware Rules*

Essas regras permitem incluir condições de existência, definidas pelos usuários, ao conjunto de regras do processo. Essas condições são muito importantes para que seja possível modelar processos de negócios completos, de acordo com as necessidades específicas de cada negócio. Por exemplo, uma atividade Replanejar Orçamento, pode ser necessária apenas se as despesas estiverem excedendo os limites planejados.

Além disso, essas *Data-Aware Rules* podem ser utilizadas para direcionar a execução do processo por caminhos diferentes. Suponha uma atividade *Análise da Proposta* cujo resultado dessa atividade é uma decisão, aceita ou negada. Esta informação pode ser essencial para a avaliação das regras a partir deste momento. Pode ser, por exemplo, que algumas atividades só sejam *obligated* se a proposta foi aceita.

O REFlex permite que as regras sejam ativadas ou desativadas durante a execução do processo. Dependendo dos resultados das atividades já executadas, as regras podem ser inseridas ou removidas dinamicamente do modelo de processo. Portanto, o processo torna-se dependente do contexto e ainda mais flexível, uma vez que o modelo muda durante a execução de acordo com as escolhas e resultados de atividades do usuário.

As expressões *Data-Aware Rules* seguem o padrão:

SE *predicado* **ENTÃO** *regra*

Para exemplificar temos,

SE *data de devolucao* $>$ *data esperada de devolucao* **ENTÃO** *aplicar multa*

A figura 16 mostra a representação gráfica para este tipo de regra, que é composta basicamente por um arco associado a uma expressão. A expressão corresponde ao predicado que é a condição para a regra.

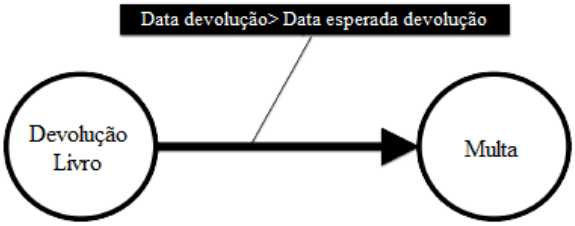


Figura 16 – Exemplo de Regra *Data-Aware*

Entretanto é importante ter ciência que a utilização dessas regras é um desafio para o algoritmo *liveness-enforcement*. A razão é que nem sempre é possível prever quais regras serão ativadas em tempo de execução. Na figura 17 é possível visualizar esse problema.

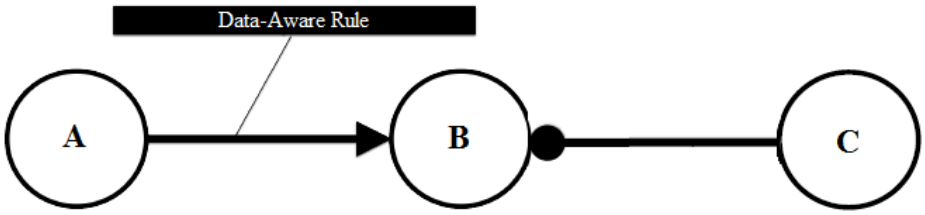


Figura 17 – Exemplo de problema devido ao uso indevido de *Data-Aware Rules*

Se a regra estiver ativa no momento em que A é *obligated*, a engine desabilita C até que A e B sejam executadas. Porém, isso é ignorado caso a regra não esteja ativa naquele momento. Então é possível que o usuário execute a atividade C, que bloqueia B. Nesse cenário, caso a condição de ativação da regra fosse alcançada depois da execução de C, teríamos uma situação de *deadlock*, pois a execução de A tornaria B *obligated* e *blocked*.

Para garantir que esse tipo de situação não ocorra, o *REFlex* limita as variáveis que podem ser afetadas por certas atividades do processo, de acordo com as seguintes instruções:

Uma atividade A não permite alterar o valor de uma variável de processo x se, para qualquer atividade B:

1. O processo contém uma regra *not after*(A, B) e

   (A) existe uma ou mais regras condicionais no processo que dependem de x; (B) as regras condicionadas a x determinam se B é *Obligated* ou não.

2. O processo contém uma regra *At most*(A, 1) e

   (A) as regras cuja existência depende do resultado de x afetam se ela pode configurar A como *Obligated* ou não. .

3.5 REFlex Orchestrator

O módulo de *Orchestrator*, proposto por Silva (SILVA, 2014), apresenta uma extensão ao módulo de mecanismo de regras do REFlex. Esse módulo permite a comunicação entre os dados provenientes do mecanismo de regras e serviços da Web. Dessa forma, é possível que os usuários utilizem esses serviços em tempo de execução de forma flexível, enquanto o sistema garante obediência ao conjunto de regras de negócio que define o processo.

3.5.1 Definição de Processo

A definição do processo é feita pelo usuário. Antes de iniciar a execução ele define, através de um arquivo XML, a instância do processo de negócio que deseja executar. Para que não haja falhas, todas as informações necessárias para a execução do processo devem estar bem definidas no arquivo de entrada.

O arquivo XML definido pelo usuário deve ser capaz de representar tanto a estrutura do processo (o grafo REFlex) como os metadados associados a essa estrutura (tipos de dados e definição de links para conexão com os serviços Web) (CARVALHO, 2015). A Figura 18 delimita todos os elementos necessários para definição do processo no arquivo XML.

A seguir descreveremos brevemente cada um dos elementos definidos na figura:

- Atividades: cada processo possui um conjunto de atividades associadas.
- ServiceBinding: inclui todas as informações necessárias para associar a execução de uma atividade com um serviço da internet.
- DataInputBinding: especifica as variáveis requeridas para cada operação, *Service-Binding*. Para cada *DataInputBinding* de entrada existe um *variableBinding* associado. Para completa definição de uma variável de entrada, o *variableBinding* tem quatro atributos: *variableName*, *global*, *type* e *expression*.

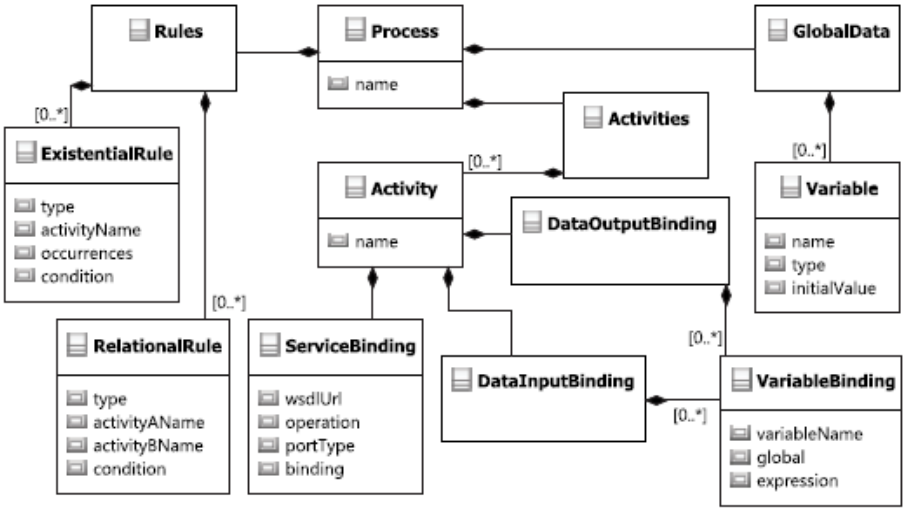


Figura 18 – Elementos XML para definir um processo de negócio (Adaptado: Carvalho (CARVALHO, 2015))

- DataOutputBinding: muito similar ao *dataInputBinding* pois também é composto por *variableBinding*. No entanto, um *dataOutputBinding* representa o retorno de uma operação (*ServiceBinding*). Quando uma operação é chamada, seu resultado retornado é capturado por um *variableBinding* para atualizar variáveis globais do REFlex.

- Regras: contém um conjunto de regras. Cada uma dessas regras possuem um tipo, o nome da atividade, o numero de ocorrências e as condição de existência.

- GlobalData: contém uma lista de variáveis globais. Durante a execução das atividades essas variáveis podem ser acessadas e modificadas.

Após a definição do processo no arquivo XML, o módulo *Orchestrator* será responsável por transformar os dados especificados nesse arquivo em elementos legíveis ao REFlex. Ou seja, nos elementos gráficos próprios do mecanismo de regras do REFlex. Dessa forma, o *Orchestrator* conseguirá interagir com o mecanismo de regras do REFlex.

A próxima seção descreve como é a interação do *Orchestrator* com os outros elementos que compõem ou interagem com o sistema.

3.5.2 Integração: REFlex Engine e REFlex Orchestrator

A Figura 19 dá uma visão geral das interações entre o módulo *Orchestrator*, que interage com a ferramenta (interface) utilizada pelo usuário, os serviços Web disponíveis e o mecanismo de regras do REFlex.

Uma vez que o usuário escolhe uma atividade *enabled* a ser executada, a interface interage com o módulo *Orchestrator* do REFlex informando a atividade escolhida. Após a

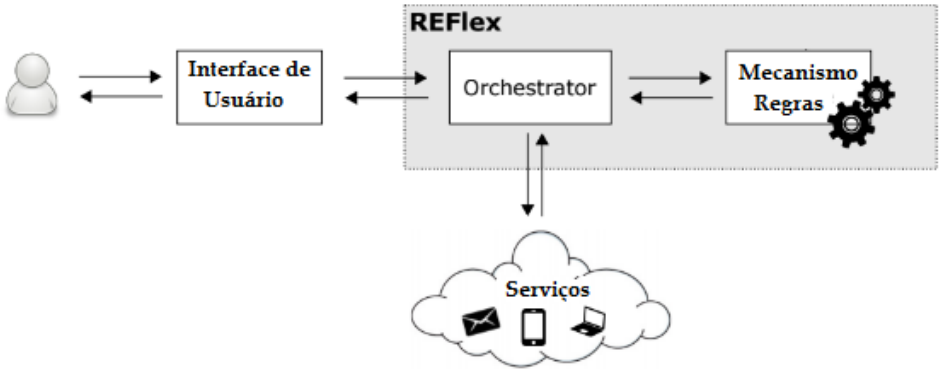


Figura 19 – Esquema de interação do módulo *Oschestrator* (Adaptado: Carvalho (CARVALHO, 2015))

execução de cada atividade, o *Orchestrator* interage com o mecanismo de regras REFlex para verificar e atualizar o status das atividades e regras de negócio do processo.

Quando uma atividade está vinculada a um serviço da Web, o *Orchestrator* é responsável por invocar o serviço correspondente para executar a operação. O REFlex usa a interface WSDL do serviço da Web para construir e interpretar automaticamente mensagens SOAP enviadas e recebidas. Assim que recebe a resposta do serviço Web, seu conteúdo é atribuído às variáveis do processo (SILVA, 2014).

Através da interface de usuário, o usuário é capaz de saber: i) quais atividades estão habilitadas a serem executadas naquele momento; ii) os valores atuais dos dados globais compartilhados pelas atividades (ou serviços da Web), e iii) se a execução do processo pode ser concluída nesse momento (CARVALHO, 2015).

3.6 Síntese do capítulo

Nesse capítulo apresentamos o mecanismo de regras e o módulo *Oschestrator* do REFlex. O primeiro é responsável por determinar e atualizar os status do processo em tempo de execução, enquanto o segundo permite a comunicação com serviços Web.

No próximo capítulo, apresentaremos a ferramenta Web-REFlex, uma extensão da ferramenta REFlex que introduz, dentro da noção de gerenciamento de grupos, uma visão completa da execução do processo para todos os usuários envolvidos em um processo de negócio.

4 WEB-REFLEX

O Web-REFlex é uma solução baseada na nuvem que visa evitar o problema de *Context Tunneling* na execução de processos de negócio declarativos. Os seus recursos permitem o conhecimento de informações contextuais que podem ajudar durante o processo de tomada de decisão, influenciando também os resultados da colaboração entre os usuários. Este capítulo descreve como o mecanismo de contexto do Web-REFlex funciona.

4.1 Arquitetura na Nuvem

O REFlex (CARVALHO, 2015) apresenta melhorias substanciais em relação a outros trabalhos, principalmente para processos usados no mundo real, porém possui problemas que limitam o gerenciamento de processos de negócio colaborativos.

Uma importante limitação do projeto original é apenas permitir uma execução de instância através de comandos em um terminal (ou console) por um único usuário. No terminal, como pode ser visto na Figura 20, ao iniciar o programa, o usuário pode escolher uma das atividades disponíveis e, após processamento pelo REFlex Engine e Orchestrator, receberá as novas tarefas ativas. Esse ciclo ocorre até a finalização do processo.

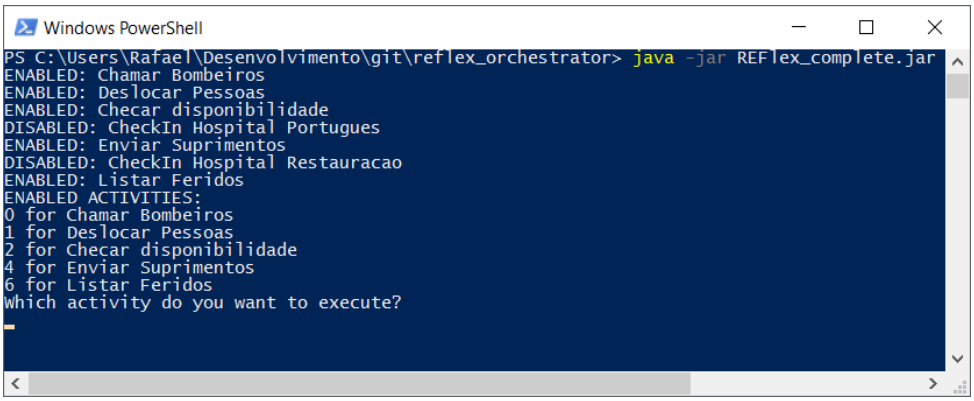


Figura 20 – Projeto original REFlex em execução no terminal

Como implicação desta limitação e ausência de funcionalidades com foco em processos colaborativos no projeto original, destacamos restrições encontradas que não atendem aos requisitos deste tipo de processo:

- o sistema precisa ser instalado localmente para cada computador de acesso, incluindo todas as dependências usadas pelo projeto original;
- cada console pode executar uma instância de um processo, não sendo possível interação entre dois terminais de acesso;

- para um console de acesso, apenas um usuário poderá executar atividade por vez, não sendo possível alternar entre entre instâncias ou processos em um mesmo console;
- não existe registro de usuários no sistema. Desse modo, não existe segurança de acesso ao REFlex e não é possível definir informações pessoais ou profissionais. Consequentemente, não é possível a definição de cargos, tornando infactível determinar quem é responsável pela execução de cada atividade;
- nenhum tipo de histórico é armazenado. As informações e lições aprendidas de outras instâncias não são aproveitadas para novos processos;
- não é oferecido informação contextual que auxilie o usuário antes ou durante a execução de um processo.

Nas próximas seções, nós esclarecemos como este trabalho soluciona as principais limitações do REFlex. Apresentamos os detalhes técnicos envolvidos e como o usuário passa a utilizar o Web-REFlex como um sistema de gerenciamento de processos de negócio (BPMS) para a execução de processos colaborativos em atividades do mundo real.

4.1.1 Tecnologias

Os projetos legados do REFlex foram desenvolvidos com a linguagem de programação Java 1.7 sem o uso de *frameworks* que apoiassem o desenvolvimento. Durante este trabalho foi definido a continuação do uso da linguagem Java, porém, com o intuito de tornar o projeto mais atualizado, foi utilizado a versão 1.8 do *Java Development Kit (JDK)*. Esta decisão inclui a adaptação dos projetos originais para esta nova versão.

Um dos objetivos deste trabalho é tornar o REFlex uma aplicação WEB podendo assim ser utilizado através de diversas plataformas com acesso à internet. Para este propósito foi necessário incluir novas linguagens e *frameworks* que nos auxiliassem durante o desenvolvimento. Dentre eles, de forma macro e que possuem relevância a serem citados, temos:

1. *Front-end*: AngularJS 1.4.7 + HTML5 + CSS3
2. *Back-end*: Java 1.8 + Springboot 1.3.6.RELEASE
3. Segurança: Spring Security
4. Banco de dados: PostgreSQL
5. Versionamento: Git

O Web-REFlex requer apenas uma conta criada no site, e não uma instalação prévia de software para que seja executado um processo de negócio. Feito isso, o usuário pode criar o seu processo de negócio como também executá-lo de qualquer dispositivo capaz de processar HTML5. Isso inclui, além de computadores e notebooks, dispositivos móveis que, em sua maioria, possuem navegador capaz de executar o Web-REFlex.

A versão WEB do REFlex é RESTful, ou seja, utiliza *Representational State Transfer (REST)* como estilo arquitetural focado nos papéis dos componentes, ignorando detalhes da implementação entre as camadas da arquitetura e da linguagem da *Application Programming Interface (API)*. Dessa forma, a camada *front-end* torna-se independente do *back-end*, permitindo que tenhamos serviços implementandos por diferentes projetos e linguagens sendo unificados na camada de visão.

Esse caminho tende a facilitar a integração do Web-REFlex com futuras implementações, sejam estas de novas técnicas de *MCDM* ou de outros módulos que possam ser desenvolvidos em outras linguagens.

O servidor é composto por um conjunto de serviços REST sem estado que se comunica com a camada de visão através de mensagens no formato *JavaScript Object Notation (JSON)*. O cliente (*browser*), através do AngularJS, utiliza um MVC que pode ser separado em uma camada de visão, um controlador e os serviços *front-end* (Figura 21). Diferentemente do MVC utilizado no *back-end*, o que é implementando pelo AngularJS tem como objetivo manipular a lógica de apresentação do dados no *browser*, e não a lógica de negócio. Em resumo, as camadas do *front-end* são:

- Camada de visão: composto pelo código HTML5, CSS3 e as diretivas do AngularJS;
- Controlador: responsável por fazer a ligação entre os dados vindos do *back-end* via JSON e a camada de visão, incluindo o *two-way data binding*. Outra responsabilidade é realizar validações para o cliente;
- Serviços *front-end*: serviços providos pelo AngularJS que permite a integração com o *back-end* e é injetado no controlador.

No caso do *back-end*, existem três camadas principais:

- Camada de roteamento: gerencia as requisições e pontos de entrada dos serviços oferecidos pela API, incluindo os parâmetros lidos das requisições HTTP;
- Camada de serviços: contém toda a lógica responsável pelo processamento do Web-REFlex;
- Camada de persistência: faz o mapeamento dos modelos com o banco de dados e suas transações.

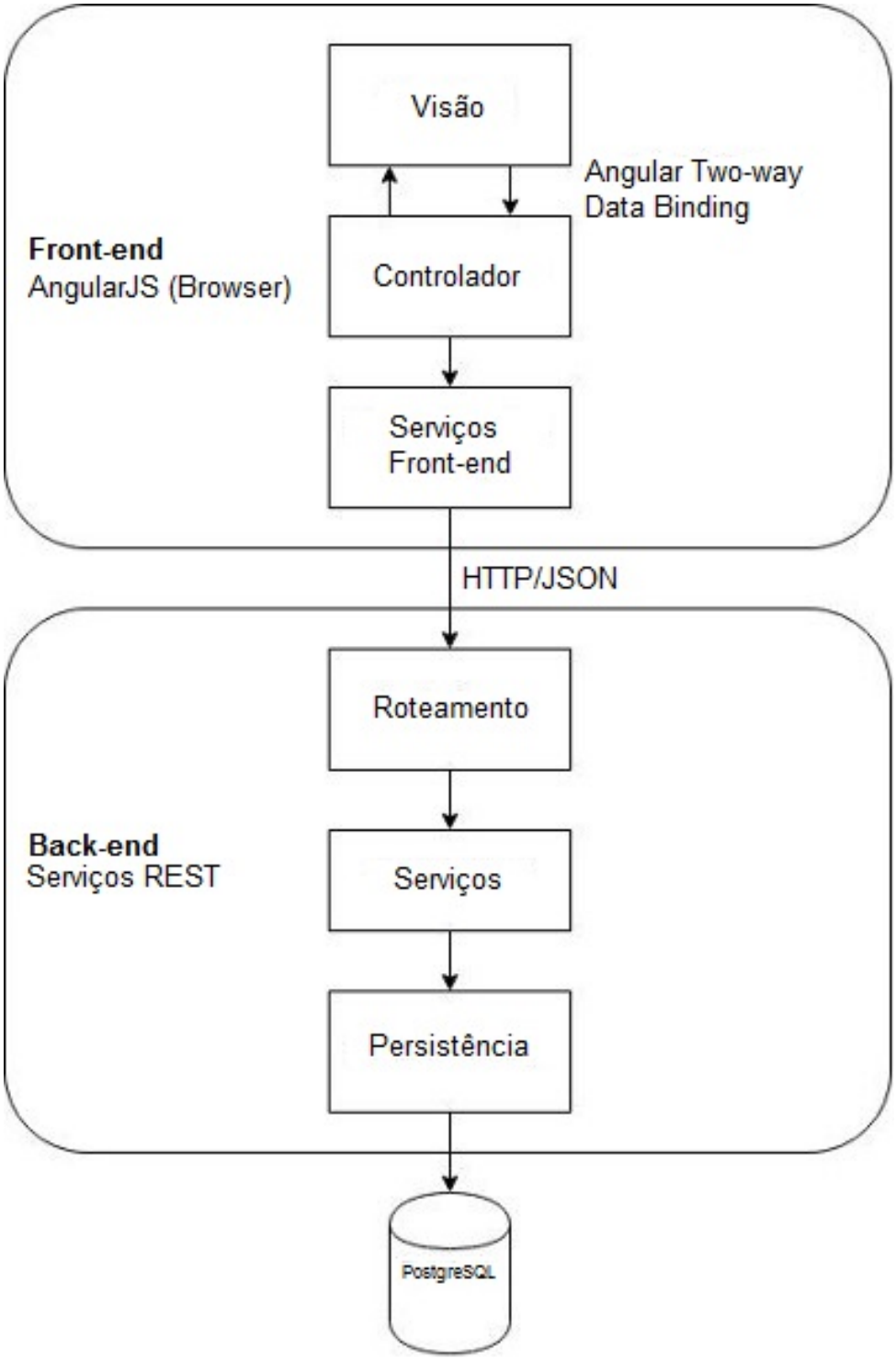


Figura 21 – Arquitetura WEB - Spring MVC e AngularJS (Adaptado: Novik (NOVIK, 2015))

Adicionalmente, no *back-end*, utilizamos o Spring Security, de forma a se beneficiar do *framework* Spring Boot utilizado neste trabalho, para implementar a segurança de acesso dos usuários e toda a comunicação entre *front-end* e *back-end*.

4.1.2 Banco de dados

Os projetos originais, REFlex Engine e Orchestrator, são responsáveis pela checagem de regras e execução remota de *webservices* associados às atividades. Todo o processo é realizado via console através de linhas de comando e, devido a isso, a execução do programa está condicionada ao seu processo no sistema operacional. Caso este seja terminado, todo progresso feito é perdido e não é possível retomar à instância do ponto de parada, sendo assim necessário executar novamente todos os passos. Essas características somadas não permitem que um processo possa ser pausado ou retomado de outro dispositivo.

O Web-REFlex resolve essas limitações com a portabilidade dos projetos legados para a WEB, usando o REFlex Engine e Orchestrator como base na manipulação dos processos de negócio e um banco de dados relacional (PostgreSQL). Este armazena informações dos processos que passa a permitir pausar e retomar instâncias a qualquer momento, em qualquer dispositivo, como também gerar estatísticas que ajudam na tomada de decisão. É importante salientar que isso foi possível pois o REFlex Engine não depende da geração de todos os estados alcançáveis (CARVALHO, 2015), o que permite salvar apenas o estado atual de uma instância, sem o *trace* completo, e retomar à sua execução sem perda de informações.

O banco relacional é criado automaticamente através da API programada em Java com o auxílio do Hibernate e *Java Persistence API (JPA)*. O banco de dados pode ser dividido entre elementos para a manipulação dos usuários, para controle dos processos e para geração de informações contextuais. Na próxima seção, detalhamos as funcionalidades do Web-REFlex e sua interação com o banco de dados.

4.2 Conceitos do Web-REFlex

A seguir serão especificadas características funcionais do Web-REFlex, que permitem expandir a visão contextual dos usuários e, assim, enriquecer o processo de execução das atividades.

4.2.1 Hierarquia de usuários

O Web-REFlex possui três níveis de hierarquia de usuários: Super Administrador, Administrador e Usuário. Estes usuários são agrupados em organizações que têm como finalidade unificar regras, configurações e processos sobre eles. As principais funções de cada perfil de usuário são:

- O Super Administrador foi planejado para um controle de alto nível do Web-REFlex. Possui permissões para gerenciar funcionalidades administrativas e são atributos únicos desse perfil, por exemplo: a criação e exclusão de organizações, a criação e exclusão de outros Super Administradores e o acesso a recursos globais do Web-

REFlex independente de organização. Esse usuário entretanto, não possui permissões para criação e manipulação de processos/instâncias/atividades, dessa forma, não tem poder diretamente sobre os processos das organizações.

- O Administrador é o perfil com maior autoridade sobre os processos, instâncias e atividades de uma organização. Algumas de suas atribuições são: criação e exclusão de processos, criação dos perfis de usuários participantes do processo, visualização de informações globais de sua organização e mudança nas propriedades de processos e atividades. Ele também possui todos os privilégios de um perfil Usuário.
- O Usuário tem permissões básicas nos processos e instâncias, como a criação de novas instâncias e a execução de atividades de uma instância, desde que tenha alguma atividade a ele atribuída. Os usuários desse perfil não podem, entretanto, fazer alterações nas configurações de artefatos de uma organização ou de um processo.

4.2.2 Visão alto nível do Web-REFlex

Como mencionado na seção 4.1.2, a estrutura do banco de dados é criada na primeira execução do Web-REFlex através das entidades programadas no *back-end*. Todas as tabelas, campos e relacionamentos são criados, porém, nenhum dado é inserido automaticamente.

Para se obter o primeiro acesso é necessária a execução de um script (Apêndice A) para que seja criado um Super Administrador e suas permissões de acesso diretamente no banco de dados. Feito esta operação, o recém-criado Super Administrador pode acessar o Web-REFlex, usando suas credenciais, para iniciar o gerenciamento do sistema.

Na tela inicial após login, este perfil possui permissão para visualizar estatísticas globais sobre todas as organizações, todavia, vale salientar que este não tem acesso a informações confidenciais: os dados dos processos, as instâncias e as atividades são restritas aos usuários com perfil Administrador da empresa. Como pode ser visto na Figura 22, algumas das informações obtidas são:

1. o número total de processos, independente de organização, criados no Web-REFlex;
2. a quantidade de instância iniciadas a partir dos processos citados no item anterior;
3. número de organizações criadas no Web-REFlex;
4. total de usuários cadastrados no sistema.

Juntamente com estas informações, incluímos uma barra de progresso para representar o crescimento do atributo ao longo de um período definido de tempo. Este intervalo foi definido a partir da volatilidade de cada informação, por exemplo, para o número de instâncias que possui elevada variação diária, usamos a escala de uma semana. Para o caso das organizações, que possui nível de crescimento menor, o período mensal se torna mais apropriado.

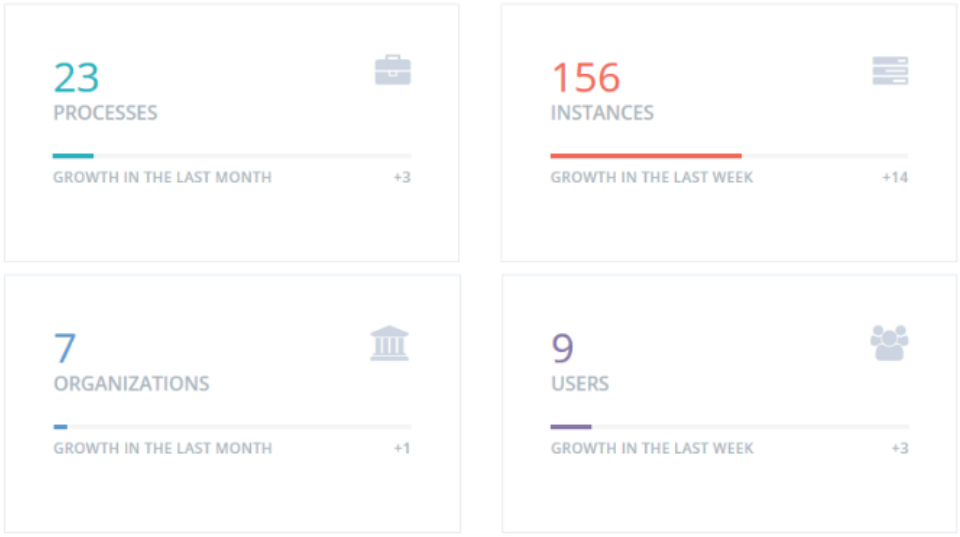


Figura 22 – Estatísticas globais do Web-REFlex

4.2.3 Organizações

Uma organização no Web-REFlex é composta dos relacionamentos entre usuários, processos e instâncias. A criação de uma organização, como mencionamos anteriormente, é de responsabilidade única do Super Administrador. Apenas após a criação da organização, um usuário com perfil Administrador poderá criar processos para essa organização.

A Figura 23 apresenta o formulário para criação de uma organização. Esta é composta por um nome e um conjunto ilimitado de cargos que posteriormente poderão ser atribuídos a seus integrantes.

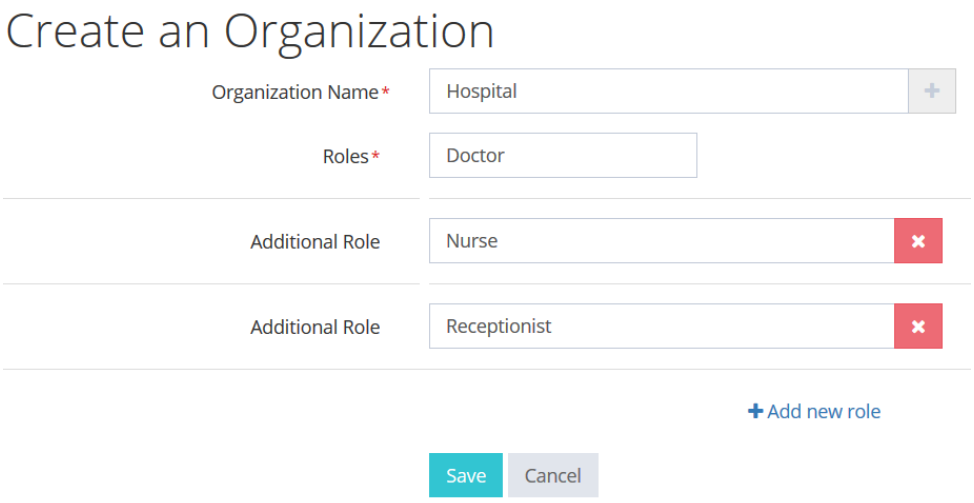


Figura 23 – Formulário para criação de uma organização

A Figura 24 lista o conjunto de organizações criadas no Web-REFlex, incluindo a recém-criada pelo formulário da Figura 23.

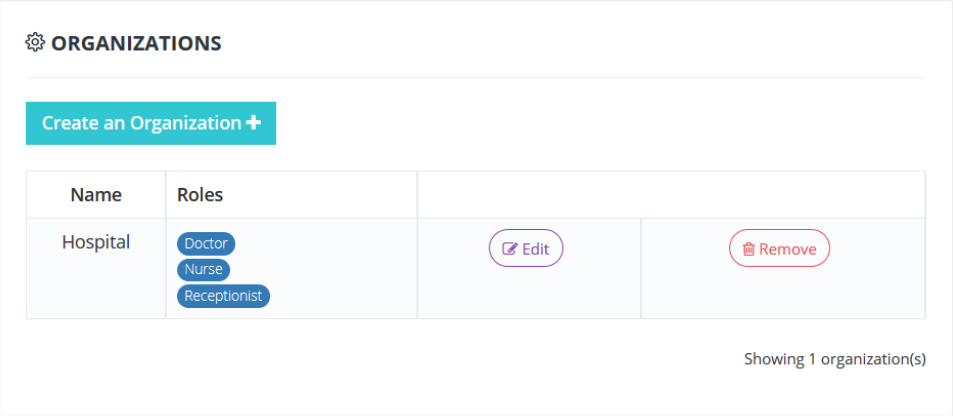


Figura 24 – Lista de organizações criadas no Web-REFlex

4.2.4 Usuários

O gerenciamento dos usuários integrantes de uma organização é permitido tanto para o perfil Super Administrador quanto para o Administrador. Entretanto, este último só pode manipular os usuários da organização a qual pertence.

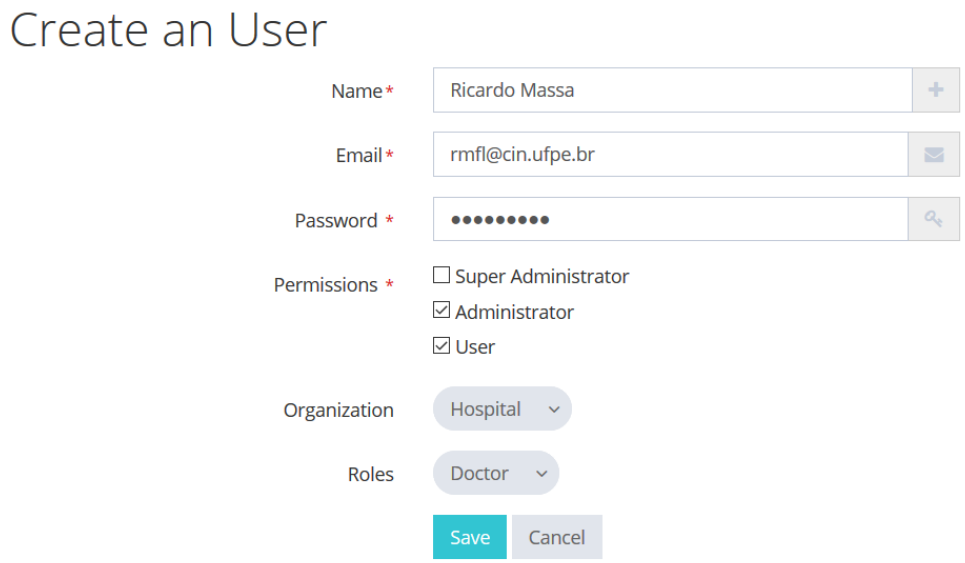


Figura 25 – Formulário para criação de um usuário

Para cada usuário é possível definir um conjunto de atributos que irão determinar permissões de visualização e execução de atividades. A Figura 25 exibe o formulário para criação de um usuário e apresenta os seguintes campos:

- Nome e e-mail: informações pessoais para identificar o usuário;
- Password: senha de acesso ao sistema;
- Permissões: cada usuário pode ter um conjunto de permissões que define seus privilégios no Web-REFlex;

- Organização: define a organização a qual o usuário pertence;
- Cargo: define a função do usuário na organização. A atribuição do cargo determina quais atividades poderão ser realizadas pelo usuário em um processo.

Após a criação, é possível visualizar todos os usuários criados. A Figura 26 exibe a lista na visão do Super Administrador. Caso essa listagem fosse vista por um Administrador, apenas os integrantes da sua organização estariam listados.

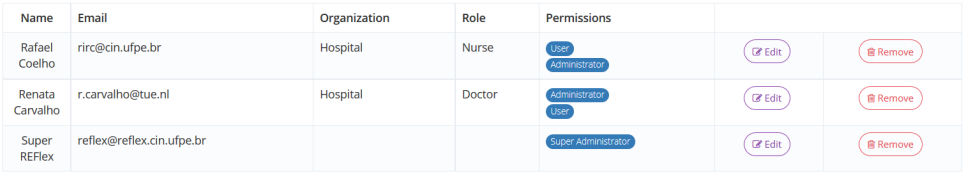


Figura 26 – Listagem de usuários (visão de um Super Administrador)

4.3 Múltiplos Processos e Instâncias

O Web-REFlex permite a definição de múltiplos processos e criação de instâncias relativas a estes. A Figura 27 exibe a tela inicial do sistema após o usuário realizar o login de acesso. Na parte superior da tela são listados os processos criados para a organização a qual o usuário pertence. Na parte inferior, seção *Resume an Instance*, são listadas as instâncias dos processos que estão ativas para o usuário logado.

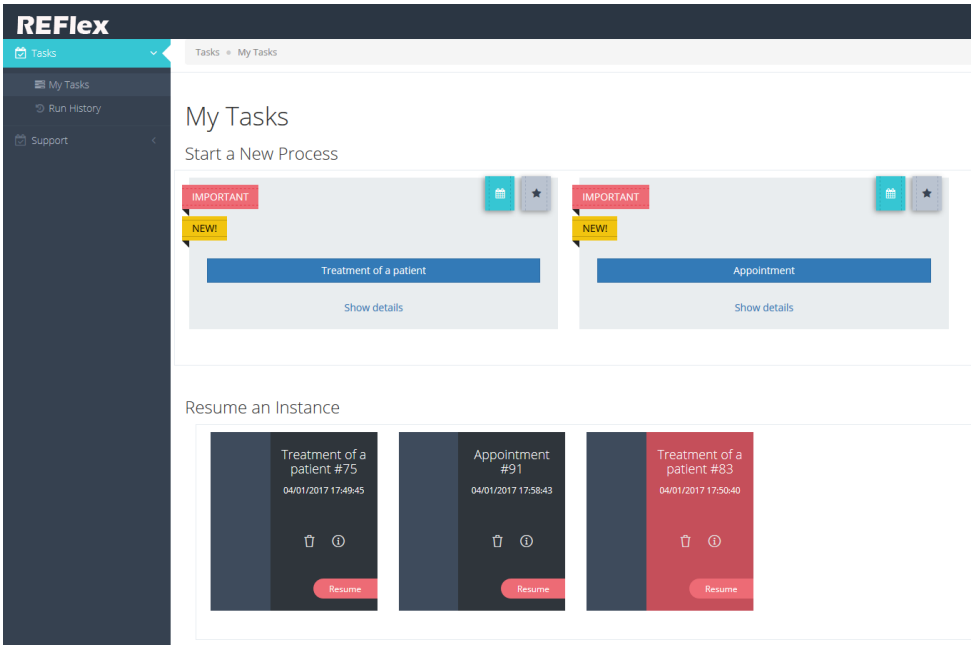


Figura 27 – Tela inicial após login

Para garantir uma melhor experiência de usabilidade da ferramenta, introduzimos algumas técnicas de apoio visual muito utilizadas. A tabela 11 apresenta alguns dos elementos utilizados na nossa proposta juntamente com os objetivos pretendidos com seu uso.

Elemento	Utilização	Objetivos Pretendidos
Cores	Identificação itens críticos	Facilitar a visualização, chamando atenção do usuário para as tarefas mais críticas
Ordenação	Ordem de acordo com a data de execução da última atividade	Permitir que o usuário possa perceber quais instâncias estão há mais tempo em espera
TAGs	Nos elementos gráficos dos processos e nas atividades	Dar destaque a informações relevantes, como: importância do processo, cargos atribuído a uma atividade, recomendação de atividades, etc.

Tabela 11 – Elementos de Usabilidade da Ferramenta

Essa organização permite que os usuários tenham uma ampla visão de todos os seus processos. Podendo pausar ou retomar suas atividades de forma prática e rápida. Nas próximas seções, 4.3.1 e 4.3.2, as características de todos os elementos que compõe a interface da ferramenta serão detalhadas.

4.3.1 Processos

O Web-REFlex utiliza um arquivo XML, descrito na seção 3.5.1, para definição de um processo, o apêndice B disponibiliza um XML de exemplo. Desse modo, a definição de um processo não é feita graficamente no Web-REFlex. A Figura 28 exibe o formulário para criação do processo.

Figura 28 – Formulário para criação de um processo

Além do upload do arquivo, é possível definir alguns atributos do processo. O *Minimum Importance Rating* determina o limite de criticidade em que algumas atividades do processo passarão a ser priorizadas. Na seção 4.3.1.1 este conceito será abordado com maior profundidade. Esse atributo não é obrigatório e o usuário pode optar por não utilizá-lo, assumindo o valor *No importance rule*. Existem quatro níveis diferentes e cada um deles é representado por uma cor para facilitar a visualização na interface da ferramenta. São eles, do mais ao menos importante: *Critical* (preto), *Important* (vermelho), *Moderate* (amarelo) e *Low* (verde).

O atributo *Set Goal*, quando selecionado, permite que o usuário utilize a técnica AHP para que, através do objetivo que se pretende alcançar com o processo, o Web-REFlex possa sugerir a atividade que deve ser priorizada em uma instância. Nesse caso, para fazer uso dessa técnica, o Administrador é redirecionado para um formulário interativo com quatro passos, inspirado na hierarquia da técnica (seção 2.7.1). As quatros etapas são definidas a seguir:

- a primeira etapa (Fig. 29) é definir o objetivo do processo, o topo da hierarquia. O Administrador deve descrever as metas do processo e esta informação serve de guia e lembrete, durante a execução do processo, para os integrantes da organização;

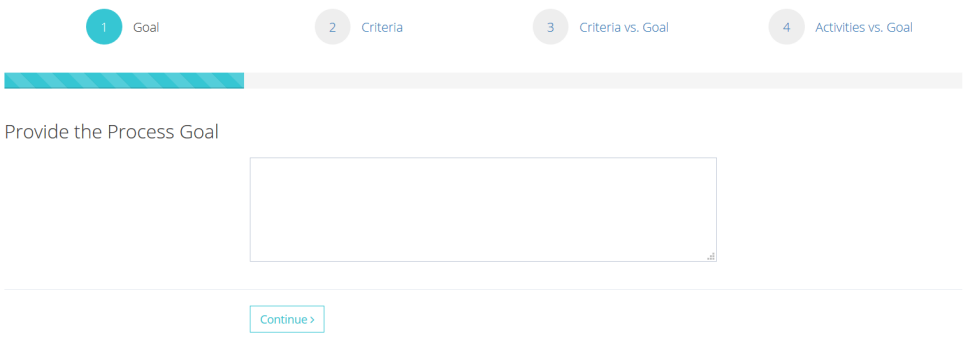


Figura 29 – Etapa 1: definição do objetivo do processo

- a segunda etapa (Fig. 30) determina os critérios utilizados pelo Web-REFlex para avaliar as atividades do processo. Os critérios são definidos livremente pelo Administrador e devem representar as qualidades que melhor avaliam uma atividade do processo;
- na terceira etapa (Fig. 31), os critérios são emparelhados por pares e o Administrador define qual elemento do par é mais importante através de uma escala numérica. As relações definidas nesta etapa são o que determinam a importância de um critério sobre outro. Assim, por exemplo, caso o critério Custo seja muito mais importante do que o Tempo, é neste passo que conseguimos deixar indicado. A partir deste momento, o Web-REFlex é capaz de montar a matriz comparativa demonstrada na seção 2.7.2;

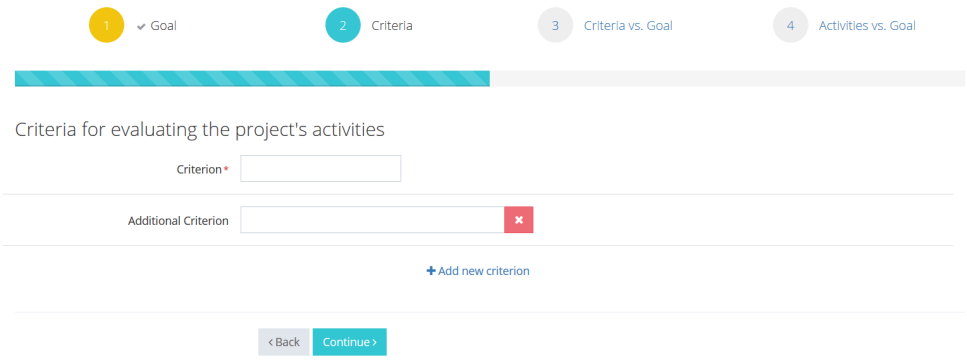


Figura 30 – Etapa 2: definição dos critérios de avaliação

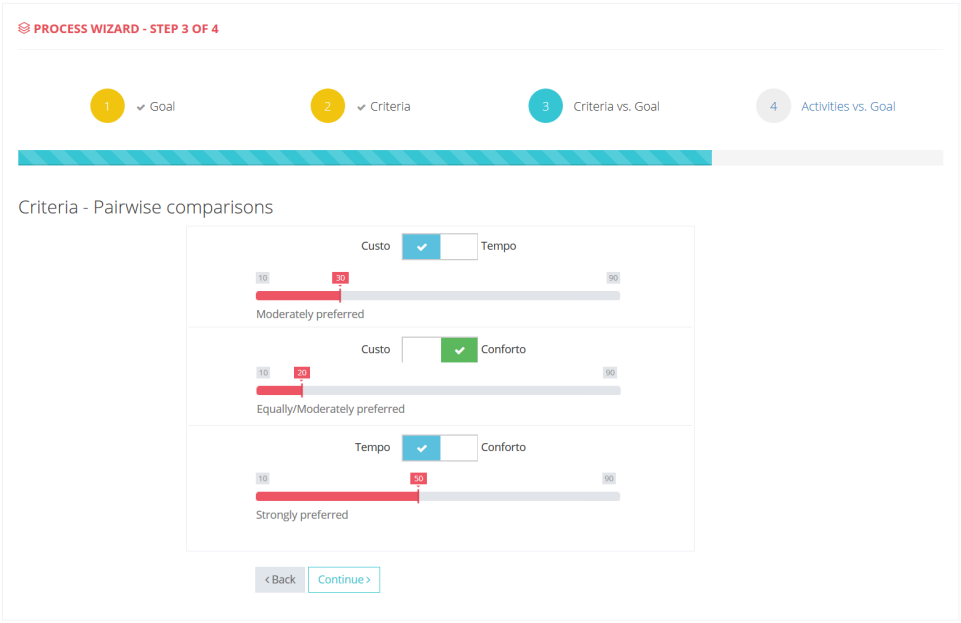


Figura 31 – Etapa 3: comparação entre os critérios de avaliação

- na quarta etapa (Fig. 32), as atividades (representadas como *Alternativas* na hierarquia) são comparadas por pares em relação a cada um dos critérios para que se possa definir a significância relativa entre estas tarefas. Similarmente ao passo anterior, a partir desta etapa é possível montar a matriz comparativa demonstrada na seção 2.7.4.

Após o cadastro de todas as informações referentes a técnica AHP, o Web-REFlex passa a indicar a atividade disponível que mais se encaixa com o objetivo do processo. Essa indicação é feita em tempo de execução e recalculada sempre que as atividades disponíveis para o usuário são atualizadas. Para sinalizar qual atividade deve ser priorizada naquele momento, o Web-REFlex utiliza a etiqueta *Best choice*, como visto na Figura 33.

Cada análise é individual para o cargo do usuário e para as atividades possíveis naquele momento, dessa forma, o Web-REFlex pode indicar uma atividade como *Best choice*

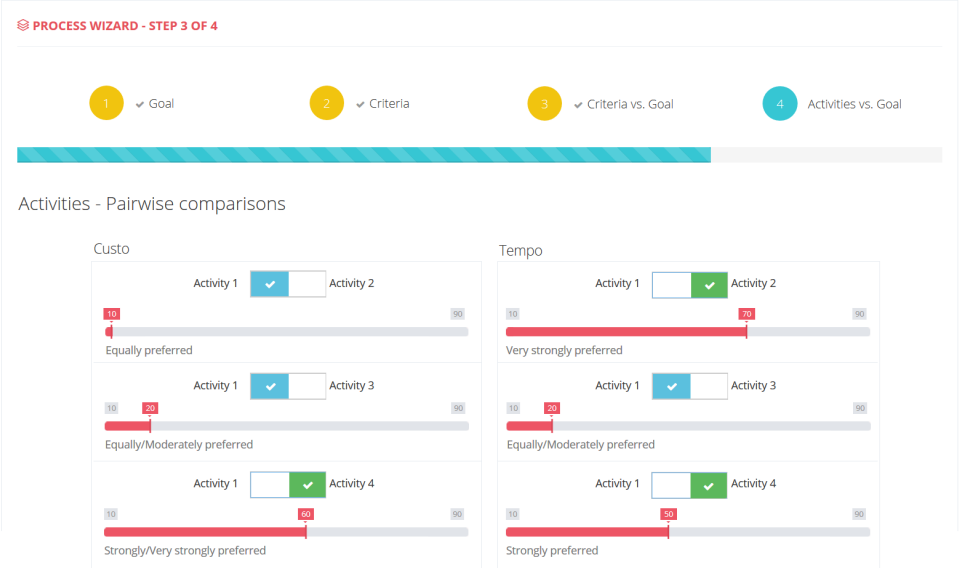


Figura 32 – Etapa 4: comparação entre as atividades do processo por critério de avaliação

diferente para cada situação de acordo com os pesos definidos para os critérios do processo.

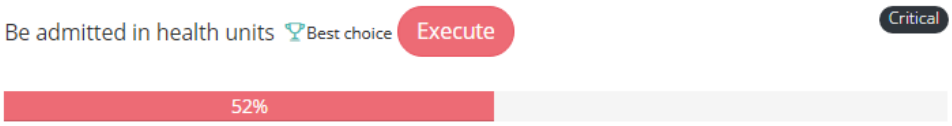


Figura 33 – Etiqueta *Best choice* representada na atividade *Be admitted in health units*

4.3.1.1 Representação Gráfica

O elemento da Figura 34 representa um processo no Web-REFlex. Este componente pode receber etiquetas personalizadas com o intuito de repassar detalhes sobre o processo de forma simples e prática. O usuário pode ter acesso a mensagens como *Important*, *New* ou ícones que tenham significado para a organização, como o de um calendário que pode representar processos com prazo de duração.

O link *Show details* direciona o usuário para uma nova tela com todas as informações do processo, essa tela é apresentada na Figura 35. Os elementos da parte superior da tela contabilizam, da esquerda para a direita, o numero total de instâncias criadas, de execuções e de usuários desse processo. Além disso, a barra de progresso em cada um dos elementos representa a taxa de crescimento do elemento em um período de tempo específico, semanas ou meses, dependendo da volatilidade da informação. Essas informações podem ser importantes para que o usuário tenha controle sobre como o processo está sendo utilizado.

Na parte inferior temos o painel de atividades. Cada processo é composto por um conjunto de atividades e para cada uma dessas atividades é atribuído um privilégio (*Privilege*

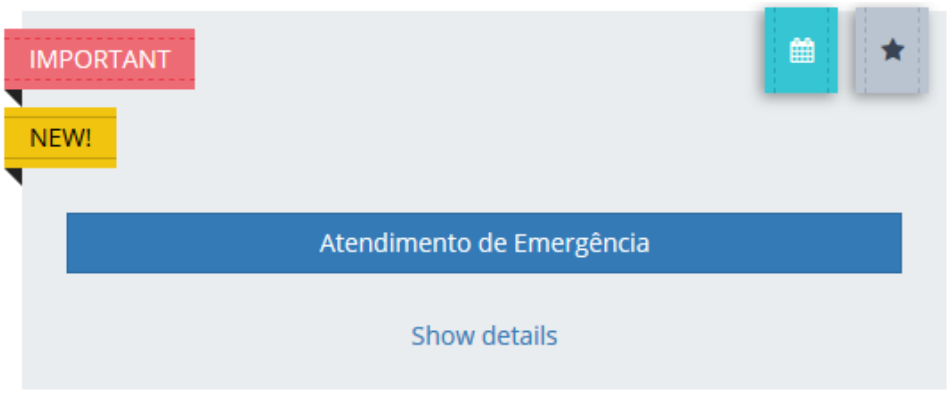


Figura 34 – Representação gráfica de um processo

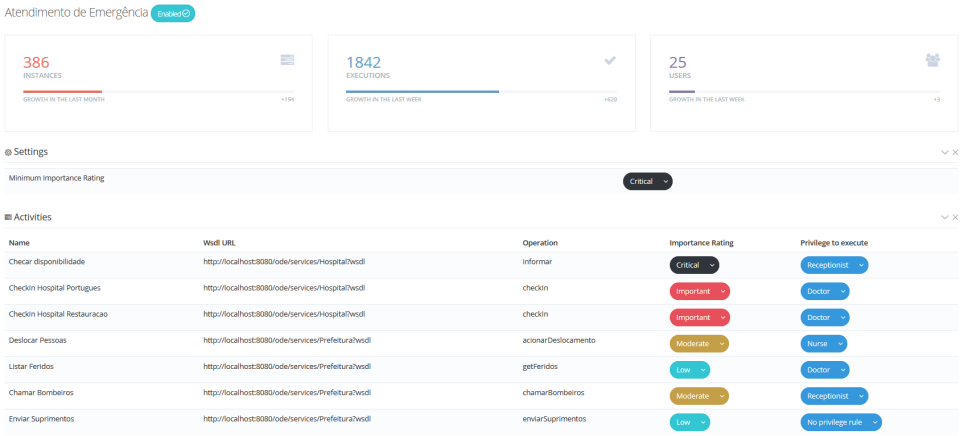


Figura 35 – Visualização dos detalhes de um processo

to execute) e um nível de criticidade (*Importance Rating*). Esses atributos são definidos a seguir:

- O atributo *Privilege to execute* é restrito ao conjunto de cargos da organização e sua função é restringir a execução de uma atividade por usuários que não possuem o cargo apropriado para executá-la. Uma atividade pode estar unicamente ligada a um cargo ou não ter um cargo definido e, portanto, pode ser realizada por qualquer funcionário da organização. Este tipo de restrição tem valor prático, uma vez que no mundo real quase todas as atividades envolvem múltiplos participantes, com papéis distintos, e suas relações de cooperação e dependências.

- O atributo *Importance Rating* define a criticidade de uma atividade. Este atributo nos permite definir um ranking de prioridade entre as tarefas, auxiliando os usuários durante a execução de uma atividade. Os níveis de criticidade são definidos através do banco de dados e a configuração padrão tem quatro níveis, assim como os definidos para determinar o atributo *Minimum Importance Rating* de um processo, que são: *Critical* (preto), *Important* (vermelho), *Moderate* (amarelo) e *Low* (verde). É

importante esclarecer, que a criticidade de uma atividade não viola as regras de processo. Durante a execução das atividades, o ranking de prioridades e o bloqueio das atividades é realizado apenas entre as atividades habilitadas no momento. Assim, para o caso em que uma atividade *a* (*important*) seja precedente para *b* (*critical*), não haveria problemas pois a atividade *b* só entraria para o ranking depois que *a* fosse executada.

O Web-REFlex realiza comparações entre o *Importance Rating* de uma atividade com o atributo *Minimum Importance Rating* que, como vimos anteriormente, é definido durante a criação do processo. Nesta comparação, realizada em tempo de execução, o *Minimum Importance Rating* é utilizado como limitador para definir quais atividades poderão ser realizadas. Por exemplo, para um processo que tenha o *Minimum Importance Rating* definido como *Important*, sempre que houver disponibilidade de atividades com pelo menos essa grandeza (neste caso, *Important* ou *Critical*), todas as outras atividades menos importantes serão desabilitadas e o usuário precisará executar algumas das atividades de urgência. Somente no caso de haver apenas atividades definidas com nível de criticidade inferior (*Moderate* ou *Low*), o usuário é livre para escolher e executar qualquer atividade disponível.

4.3.2 Instâncias

Uma instância é uma representação concreta de um processo. Esta é composta por um conjunto de atividades com seus respectivos cargos e níveis de criticidade definidos no processo.

No Web-REFlex, o usuário pode iniciar uma instância através da representação gráfica do processo. Como visto na Figura 34, o usuário deve clicar no botão azul central para que uma instância seja criada. Um integrante apenas pode criar instâncias de processos da organização a que pertence e possa executar, no mínimo, uma atividade do processo alvo. Dessa forma, o processo só fica visível para um usuário caso exista alguma atividade disponível para ele. Após a criação da instância, o usuário pode dar início à execução das atividades.

4.3.2.1 Representação Gráfica

A representação gráfica de uma instância pode ser vista na Figura 36. Além do nome, é possível destacar quatro itens importantes desse elemento:

- Cor de fundo: apresenta a cor da atividade mais importante disponível para o usuário nesta instância em particular. Usuários com diferentes funções podem ter diferentes atividades disponíveis e, consequentemente, a cor pode mudar de acordo com a criticidade da atividade de maior prioridade. Essa informação pode ser diferencial no

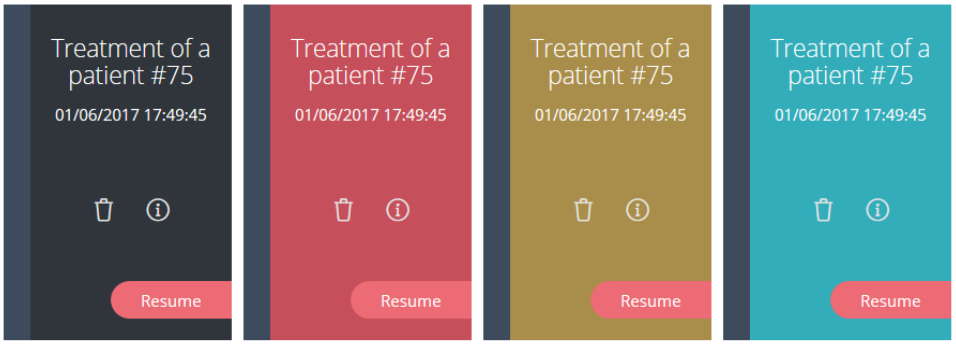


Figura 36 – Representação gráfica de uma instância de acordo com as quatro possíveis cores do atributo *Importance Rating*

momento em que o usuário precisa decidir qual instância deve atender, permitindo que ele opte pela instância mais importante naquele momento.

- Data da última execução de uma atividade: sempre que uma atividade é executada em uma instância, o Web-REFlex armazena a hora atual. Na seção *Resume an Instance*, as instâncias são organizadas de acordo com esta data, fazendo com que aquelas pendentes a mais tempo tenham preferência sobre as mais recentes. Esse padrão de organização permite que o usuário possa priorizar as instâncias que estão a mais tempo em espera.

- Botão de informação: abre uma caixa de diálogo com informações relevantes sobre a instância, por exemplo, tempo desde que a última atividade foi executada, número de execuções da mesma atividade nessa instância e uma classificação que mostra da atividade mais frequente para a menos comum.

- Botão *Resume*: direciona o usuário para a tela em que são apresentadas as atividades do processo. Esse gerenciamento será detalhado na próxima seção deste capítulo.

Com esses atributos, o usuário tem informações que o permitem decidir qual instância é mais relevante para ser retomada. O utilizador, que é o especialista no negócio da organização, é capaz de analisar rapidamente a relevância das instâncias e tomar alguma decisão.

4.4 Gerenciamento de Atividades

O fluxo de execução das atividades no Web-REFlex é feito utilizando o motor de regras do REFlex, as informações de contexto do Web-REFlex e as escolhas dos usuários. O ciclo para definição das atividades disponíveis, que é repetido sempre que um usuário executa uma atividade do processo, é representado pela Figura 37, e é composto por 3 etapas:

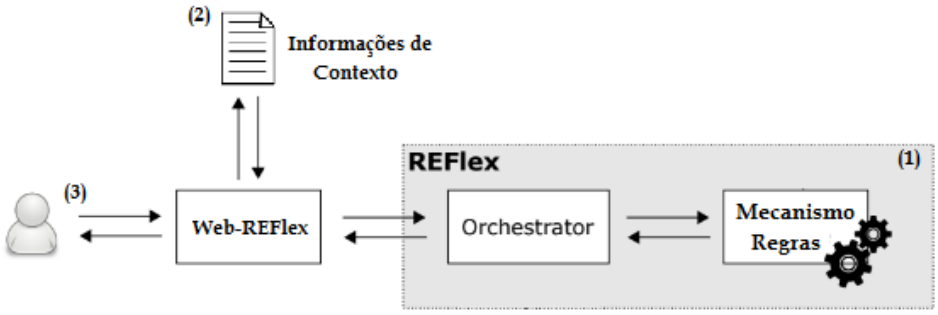


Figura 37 – Ciclo de atualização das atividades.

1. Etapa (1): quando uma instância é iniciada ou reiniciada, o Web-REFlex primeiramente comunica-se com o REFlex que, baseado nas regras de negócio definidas no arquivo XML, disponibilizará um conjunto de atividades disponíveis para execução naquele momento. O mecanismo de regras do REFlex não precisou ser atualizado, os atributos/regras inseridos por esse trabalho são armazenados em banco de dados e são processados pelo Web-REFlex em conjunto com os provenientes do *REFlex Engine*.

2. Etapa (2): em seguida, o Web-REFlex filtra, dentre as atividades de entrada, quais são as atividades que estarão habilitadas para o usuário. Para isso, utiliza as informações de contexto, como os atributos *Privilege to execute* e *Importance Rating*.

   Por exemplo, o atributo *Privilege to execute* é responsável por restringir a execução de atividades que não pertencem ao cargo do usuário. Essas atividades são mostradas na visualização da instância, mas ficam bloqueadas para o profissional. Como pode ser visto na Figura 38, um funcionário com o cargo *Doctor* não pode executar a atividade *Undertake health examination*, pois essa é de responsabilidade de um *Nurse*.

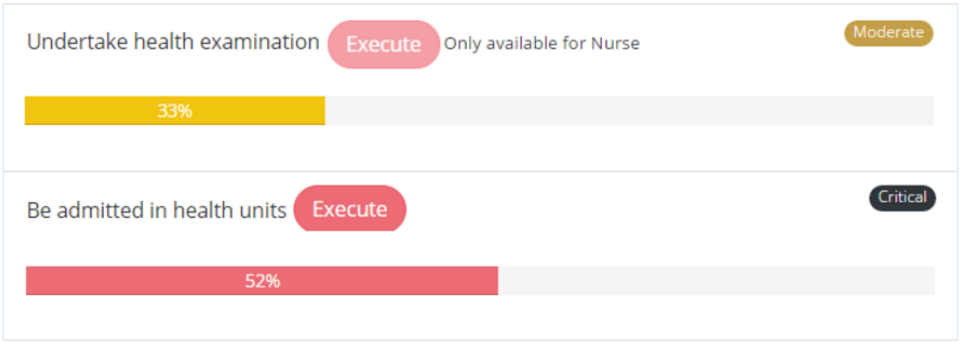


Figura 38 – Proporção em que uma atividade foi executada em cenários anteriores

Esse recurso também é relevante para minimizar o problema de *Context Tunneling*, uma vez que o *Doctor* tem acesso a todas as atividades disponíveis naquele momento,

e não apenas às relativas ao seu papel, permitindo, assim, uma visualização global das atividades do processo.

Além disso, a informação contextual *Importance Rating* da atividade também pode ser utilizada para filtrar as atividades de acordo com o *Minimun Importance Rating* do processo, desde que este tenha sido definido durante a sua criação.

Como já mencionamos na seção anterior, comparações são realizadas entre esses dois atributos. Dessa forma é possível verificar se há atividades que devem ser priorizadas, que exigem execução imediata no conjunto de atividades disponíveis para o cargo do usuário. Assim, o Web-REFlex as torna habilitadas e outras menos importantes serão desabilitadas para execução (Figura 39). Esta abordagem, apesar de reduzir a flexibilidade durante a execução, explicita quais as normas da organização devem ser priorizadas.

Na instância apresentada, as atividades com *Importance Rating* inferior a *Important* estão todas desativadas, pois o *Minimun Importance Rating* do processo é *Important*. Esse recurso pode ser utilizado quando um administrador da organização quiser obrigar que os participantes da organização priorizem atividades mais importantes quando estas estiverem disponíveis.

3. Etapa (3): representa a escolha do usuário. Todas as vezes que um usuário executa uma atividade, um novo ciclo se inicia.

4.5 Processo de execução das atividades

Para executar uma atividade, o usuário deve clicar na instância do processo na qual deseja atuar e assim poderá ter acesso as atividades disponíveis para ele. A Figura 39 mostra a visão de um usuário em relação às atividades do processo.

Cada atividade é representada por um nome, uma *tag* que representa seu *Importance Rating*, uma barra de progresso com um valor estatístico vinculado e, caso esta seja a alternativa mais bem avaliada pelo AHP, a etiqueta *Best choice*. O valor estatístico é definido como uma porcentagem e representa a proporção de vezes que aquela atividade foi executada em cenários idênticos anteriormente, quando o mesmo conjunto de atividades estava disponível. A Figura 38 apresenta duas atividades e suas proporções correspondentes.

Esse reaproveitamento das informações de instâncias anteriores é outra característica importante para minimizar o *Context Tunneling*. Com essa informação o usuário pode utilizar a experiência de outros especialistas para tomar a decisão de qual atividade é mais apropriada para ser executada no cenário em que se encontra.

O botão *Execute* aciona a execução da atividade. A partir disso, um novo ciclo de atualização das atividades é iniciado. Esse esquema continua até que o processo seja finalizado.

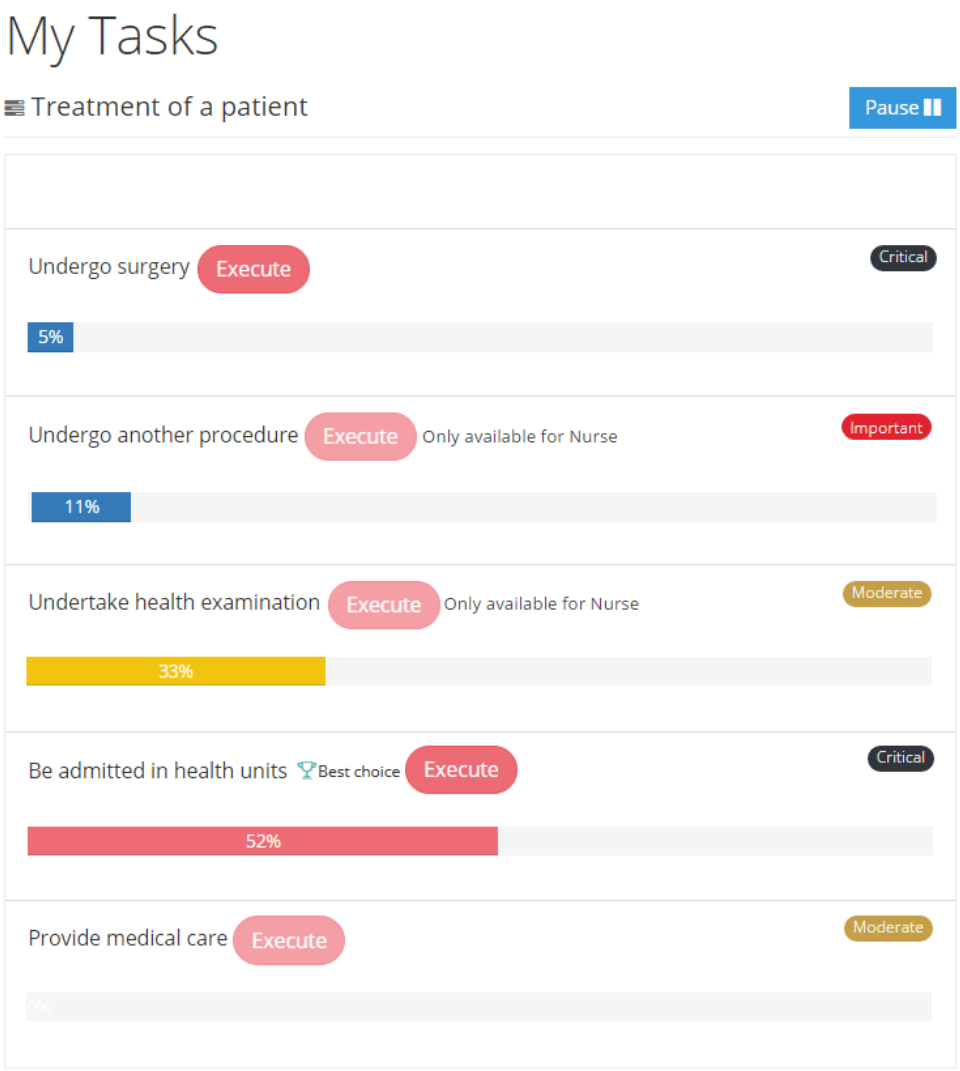


Figura 39 – Atividades de uma instância

4.6 Sincronização

Quando uma atividade é executada, um novo ciclo de atualização das atividades determinará quais serão as novas atividades disponíveis que, como podem ser diferentes das atuais, podem tornar imprópria alguma atividade que estava habilitada para execução. Assim, o conjunto de usuários que terá alguma atividade disponível varia de acordo com o estado atual da instância que é atualizada a cada execução de uma atividade.

Para evitar que as instâncias cheguem a um estado incorreto em tais situações, o Web-REFlex controla a concorrência entre os usuários usando um mecanismo de comparação de datas. Um atributo importante para a solução deste problema é a data da última execução de uma atividade de uma instância. Quando o profissional tenta retomar uma instância ou executar uma atividade, esta data é enviada para o Web-REFlex e ele identifica se o usuário tem uma cópia atualizada da instância.

Quando a instância está atualizada, o usuário será capaz de executar a operação e

o Web-REFlex atualiza a data de execução da atividade. Caso contrário, o profissional é informado do erro (Figura 40) e redirecionado para o conjunto mais atualizado de instâncias disponíveis. Essa dinâmica acontece até que a instância possa ser concluída com êxito.

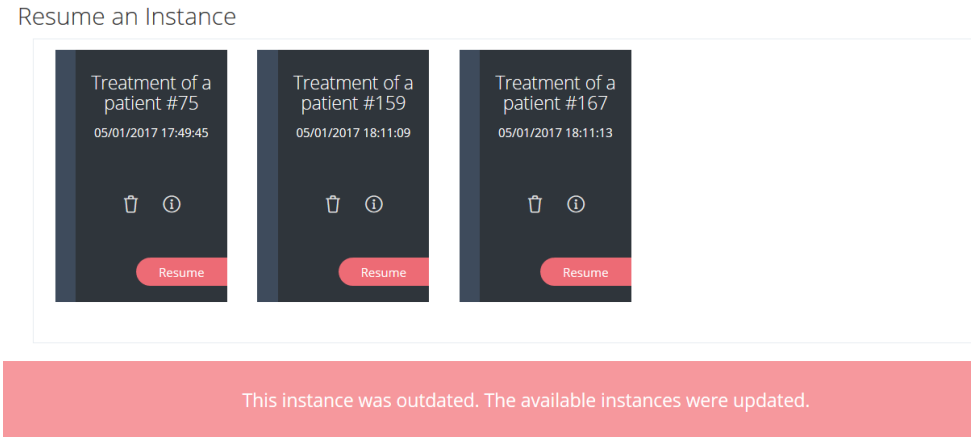


Figura 40 – Erro de instância desatualizada

4.7 Síntese do capítulo

Como demonstramos nesse capítulo, nossa solução para o problema de *Context Tunneling*, Web-REFlex, é uma ferramenta que relaciona conceitos de gerenciamento de grupos com o compartilhamento de informações contextuais para que os usuários envolvidos tenham uma visão completa do processo de negócio.

Para isso, nós incorporamos conceitos de gerenciamento de processos de negócio colaborativos ao *framework* REFlex. A partir do Web-REFlex, através da portabilidade para a nuvem, é possível acessar este *BPMS* a partir de qualquer dispositivo com acesso a internet, que possua leitor de HTML5, sem a instalação de programas adicionais. Esta mudança permite que múltiplos usuários possam trabalhar simultaneamente em variados processos e instâncias.

Passamos a coletar dados de uso e, com isso, fornecer estatísticas de como os usuários tem interagido com os processos. Permitimos que as atividades tenham níveis de importância e, aliado com a técnica *AHP*, estes saibam quais atividades são mais impactantes e tenham maior chance de cumprir o objetivo do processo mais rapidamente.

O Web-REFlex, com propósito de ser testado em organizações reais, possui segurança através de credenciais e regras de acesso a áreas distintas conforme o perfil do usuário cadastrado. Também utilizamos controle de concorrência para evitar que as instâncias alcancem algum estado incorreto ou que seja necessário retrabalho pelos integrantes da organização.

No capítulo a seguir, apresentamos a execução de um processo colaborativo através do Web-REFlex. Exploramos situações em que os usuários possam tirar proveito das informações contextuais, oferecidas pela ferramenta, para tomar melhores decisões durante a execução de um processo.

5 CASO DE EXEMPLO DA PROPOSTA

Esta seção apresenta um caso de exemplo para demonstrar o uso do Web-REFlex como uma solução para o processamento de informações contextuais para processos colaborativos. Demonstramos como usar nossa ferramenta através de um processo de negócio vinculado a um atendimento hospitalar para o atendimento de pacientes em emergência. Para obter melhores resultados, previamente, realizamos simulações a fim do banco de dados conter informações sobre execuções de instâncias anteriores.

5.1 Problemática - Processo de atendimento em um hospital

O processo analisado neste caso de exemplo caracteriza o tratamento de pacientes em um hospital (organização). O processo se inicia com a chegada do paciente à organização e finaliza quando o paciente recebe alta. Para cada novo paciente, é iniciada uma nova instância de processo que representa o início do atendimento. A Figura 41 apresenta o modelo REFlex para nosso caso de exemplo.

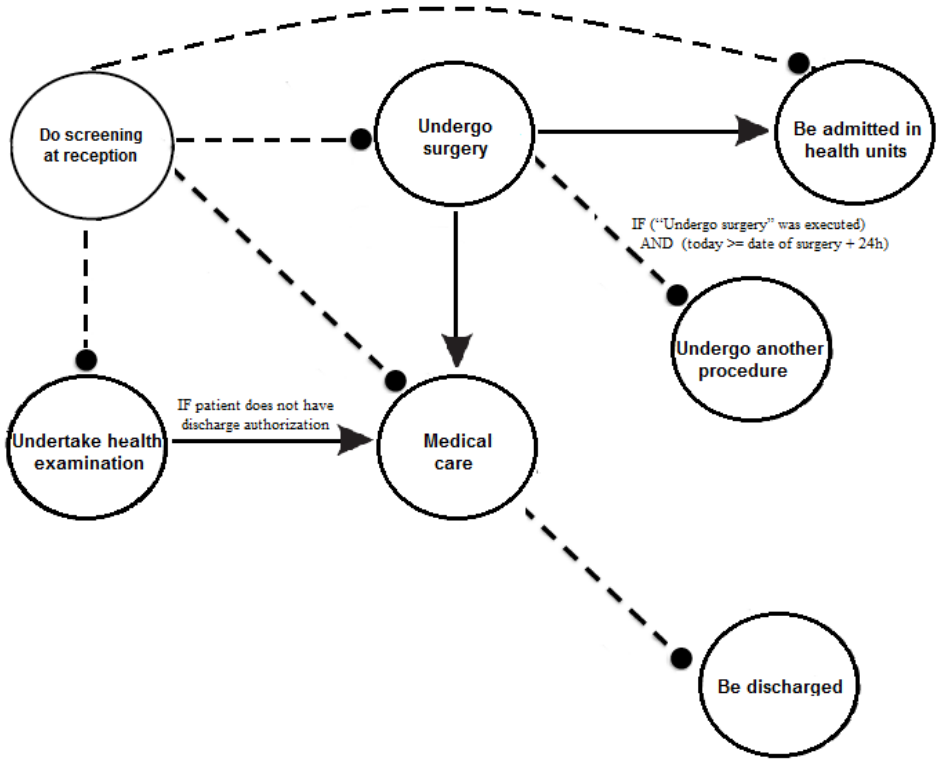


Figura 41 – Modelo REFlex para representação do caso de exemplo

Na tela *Create a Process* (Figura 42) definimos o processo através do upload do arquivo *emergencia.xml*. O *Minimum Importance Rating* atribuído foi *Important* e esse valor

será utilizado posteriormente para definir quais atividades deverão ser priorizadas pelos colaboradores do processo durante a execução das atividades.

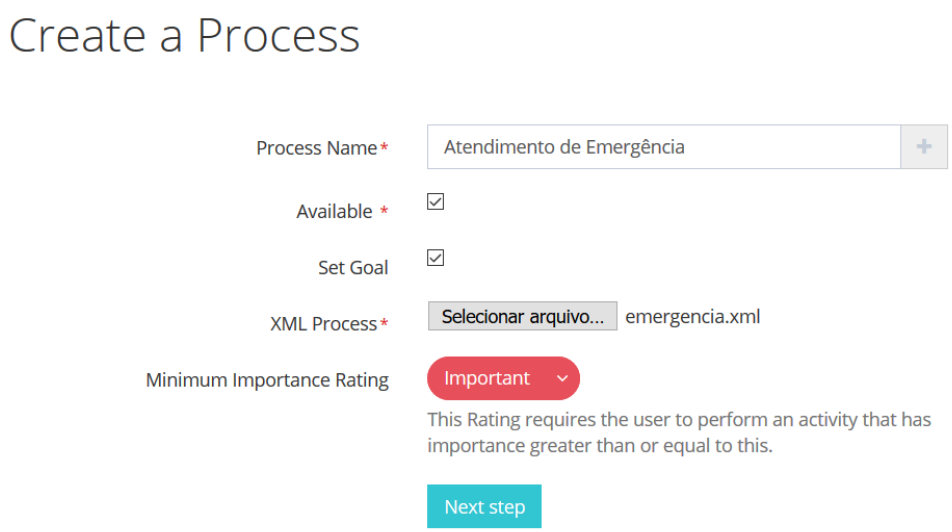


Figura 42 – Formulário de criação do processo do caso de exemplo

A Figura 43 apresenta a lista de atividades após criação do processo. Como padrão, os atributos *Importance Rating* e *Privilege to execute* recebem inicialmente os valores default *Low* e *No privilege rule*. No entanto, através do painel de controle (Figura 43) do processo, esses valores foram alterados para atribuir cargos às atividades e definir as prioridades de cada uma. Além disso, o hospital contém sete serviços web (*WSDL URL*) no qual cada um está relacionado a uma atividade do processo de negócio.

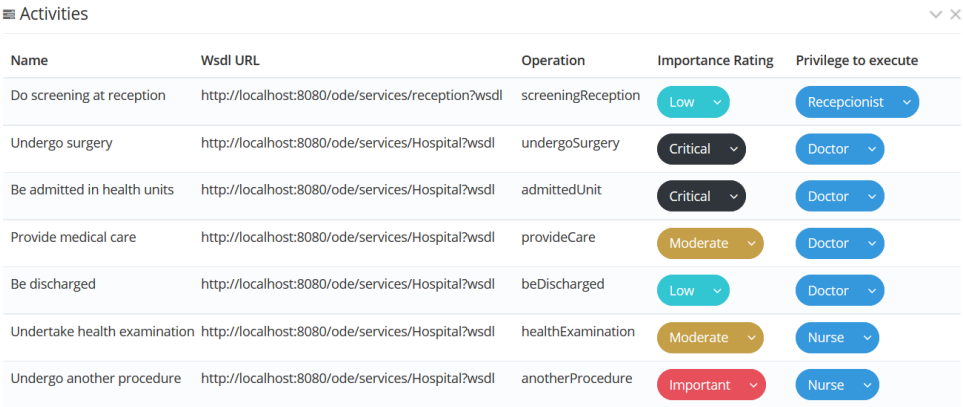


Figura 43 – Lista das atividades e seus atributos

Todas as atividades do processo foram distribuídas entre três cargos previamente definidos pelo usuário Administrador (*Receptionist*, *Nurse* e *Doctor*). Dessa forma, qualquer usuário da organização que possua um desses três cargos associados ao seu perfil terá permissão para realizar as tarefas do cargo correspondente. A Tabela 12 lista os cargos e relaciona-os com as atividades permitidas.

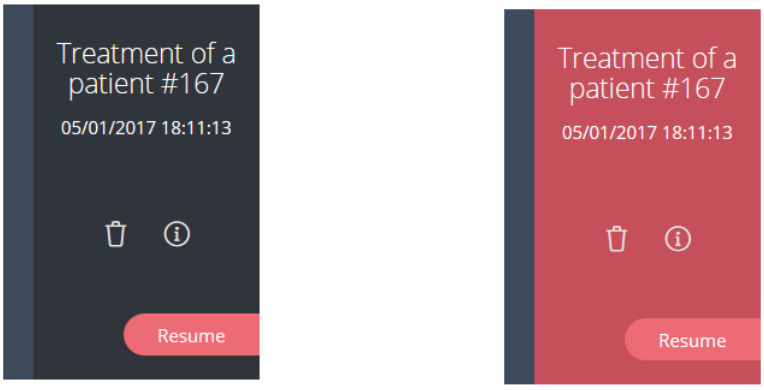
Tabela 12 – Distribuição das atividades de acordo com os cargos

CARGO	ATIVIDADES
Receptionist	*Do screening at reception*
Nurse	*Undertake health examination* *Undergo another procedure*
Doctor	*Undergo surgery* *Provide medical care* *Be discharged* *Be admitted in health units*

5.1.1 Execução do processo

A fim de exemplificar como funciona a ferramenta, mostramos a execução do processo **Atendimento de Emergência** sob o ponto de vista de dois usuários de cargos distintos: *Doctor* e *Nurse*. Esses dois usuários compartilham a mesma instância de processo, essa decisão foi tomada para elucidar o controle de concorrência entre usuários que trabalham em uma mesma instância. Para facilitar o entendimento, admitimos que a atividade *Do screening at reception* já foi realizada por um usuário *Receptionist*.

No cenário atual, após execução da atividade pela recepcionista, a instância possui cinco atividades disponíveis: *Undergo surgery*, *Provide medical care*, *Undertake health examination*, *Undergo another procedure* e *Be admitted in health units*. Como há atividades disponíveis para *Nurses* e *Doctors*, a instância do processo fica visível para ambos os usuários. No entanto, eles observam a instância no Web-REFlex de forma diferente, conforme podemos observar na Figura 44.



(a) (b)

Figura 44 – Visão da mesma instância por dois usuários com cargos distintos: (a) *Doctor*, (b) *Nurse*

5.1.2 Escolha da Instância

A instância do *Doctor* é colorida com a cor preta (*Critical*) e a *Nurse* com a cor vermelha (*Important*). Isso acontece porque no conjunto de atividades disponíveis para o *Doctor*, naquele instante da execução, existe pelo menos uma atividade com *Importance Rating* definido como *Critical* (*Undergo surgery* e *Be admitted in health units*), enquanto para o usuário *Nurse*, a atividade mais importante é *Undergo another procedure* que tem *Importance Rating* igual a *Important*.

A utilização das cores torna-se importante quando o usuário possui em execução várias instâncias de um mesmo processo (Figura 45). Nesse caso, o usuário pode identificar qual instância possui a atividade mais crítica disponível já em sua lista de instâncias na tela inicial. Dessa forma, ele pode optar por realizar primeiro a atividade da instância na qual o nível de importância da sua ação é maior.

(a) (b) (c)

Figura 45 – Lista das instâncias disponíveis para um usuário

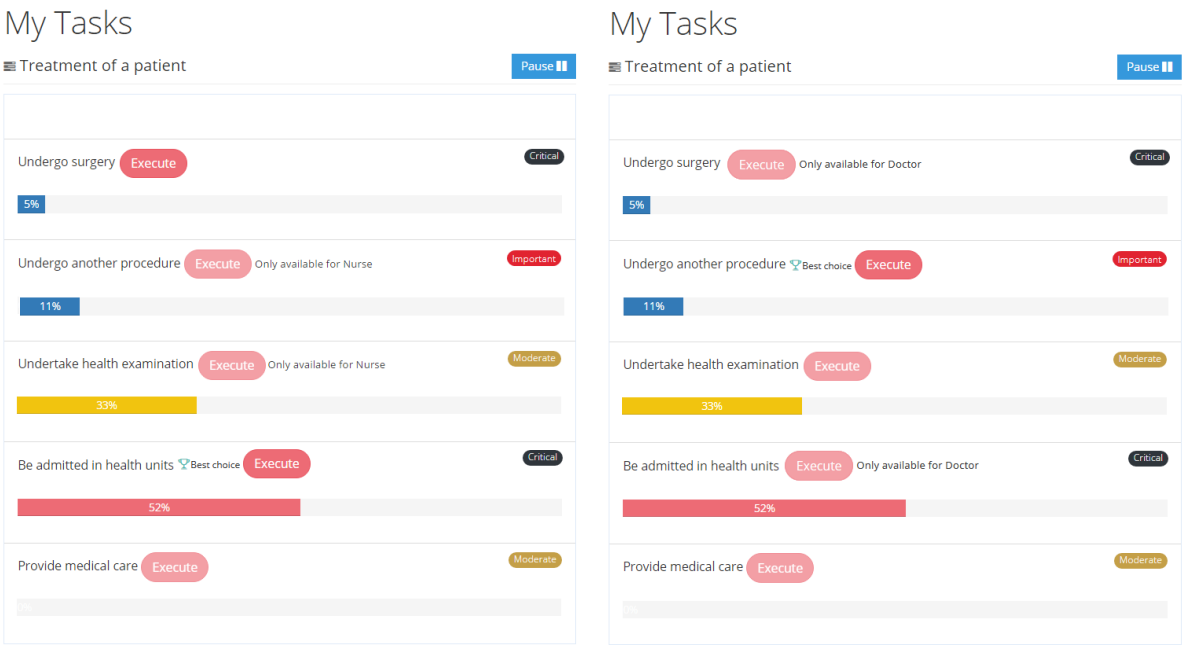
Entretanto, o usuário pode considerar outro detalhe importante para decidir qual a melhor opção. A data apresentada na instância representa o momento da última execução de uma atividade. Assim, é possível saber há quanto tempo a instância está em espera.

No cenário do nosso caso de exemplo, o usuário poderia optar por escolher tanto a instância que está há mais tempo parada (Fig. 45(c)), quanto a instância em que sua atividade possui maior criticidade (Fig. 45(a)), ou até mesmo balancear esses dois atributos para decidir a melhor opção. Essas informações permitem ao usuário tirar proveito de informações contextuais da instância/atividade e, assim, optar por fazer a atividade mais impactante antes ou que pode acelerar o encerramento do processo.

5.2 Execução das atividades de uma instância

Ao acessar uma instância (botão *Resume*), o usuário tem acesso ao conjunto das atividades disponíveis e, associado a cada atividade, uma barra de progresso que representa

a frequência em que esta atividade foi executada em instâncias anteriores, quando o cenário era semelhante ao cenário atual da perspectiva do usuário. Além disso, é atribuído informações sobre qual o cargo responsável por sua execução e o nível de importância de cada uma delas, como apresentado na Figura 46.



(a) (b)

Figura 46 – Atividades disponíveis para diferentes cargos: (a) *Doctor*, (b) *Nurse*

O botão *pause* permite pausar a execução da instância e o usuário volta para a tela principal da ferramenta, onde pode retomar a execução quando desejar clicando no elemento correspondente a instância.

As tabelas 13 e 14 resumem a figura 46 com a lista de todas as atividades. Para que o usuário possa executar uma atividade ela deve estar *enabled*, autorizada para usuários com seu cargo e ter *Importance Rating* maior ou igual ao *Minimum Importance Rating* do processo, que para o caso em questão é *Important*. Por isso, embora as atividades *Undertake health examination* e *Provide medical care* estejam habilitadas, não podem ser executadas por seus respectivos cargos *Nurse* e *Doctor* pois possuem *Importance Rating* menor que o *Minimum Importance Rating* do processo.

Tabela 13 – Distribuição das atividades *Doctor*

ATIVIDADES	*CARGO*	*IMPORTANCE*	PODE SER EXECUTADA?
Undergo surgery	*Doctor*	*Critical*	Sim
Undergo another procedure	*Nurse*	*Important*	Disponível apenas para *Nurse*
Undertake health examination	*Nurse*	*Moderate*	Disponível apenas para *Nurse*
Be admitted in health units	*Doctor*	*Critical*	Sim
Provide medical care	*Doctor*	*Moderate*	*Importance Rating* inferior ao do processo

Tabela 14 – Distribuição das atividades *Nurse*

ATIVIDADES	*CARGO*	*IMPORTANCE*	PODE SER EXECUTADA?
Undergo surgery	*Doctor*	*Critical*	Disponível apenas para *Doctor*
Undergo another procedure	*Nurse*	*Important*	Sim
Undertake health examination	*Nurse*	*Moderate*	*Importance Rating* inferior ao do processo
Be admitted in health units	*Doctor*	*Critical*	Disponível apenas para *Doctor*
Provide medical care	*Doctor*	*Moderate*	Disponível apenas para *Doctor*

Nas subseções a seguir, demonstramos como essas informações contextuais podem ser utilizadas como critérios para limitar as escolhas dos usuários ou para auxiliá-los na tomada de decisão sobre qual atividade deve executar.

5.2.1 Limitação das atividades disponíveis

Na Figura 46 é possível observar que algumas atividades ficam bloqueadas para os usuários. Na Figura 46(a), por exemplo, o usuário *Doctor* pode realizar apenas as atividades *Undergo surgery* e *Be admitted in health units*. A seguir, listamos o motivo de exclusão para cada atividade não disponível para execução por um usuário deste cargo:

- *Undergo another procedure* e *Undertake health examination*: estas atividades estão disponíveis apenas para *Nurses*, pois foram definidas para esse cargo no momento da definição do processo;
- *Provide medical care*: essa atividade possui *Importance Rating* igual a *Moderate* que é menor do que o *Minimum Importance Rating* do processo (*Important*). Nesse caso, a atividade torna-se desabilitada pois a instância possui uma atividade com *Importance Rating* maior ou igual que *Important*.

Neste mesmo cenário, mas sob a visão de um usuário *Nurse*, as atividades disponíveis são diferentes, como visto na Figura 46(b). Apenas a atividade *Undergo another procedure* pode ser executada. As razões que impedem a execução das outras atividades são:

- *Undergo surgery* e *Be admitted in health units*: atividades disponíveis apenas para *Doctor*;
- *Undertake health examination* e *Provide medical care*: possuem *Importance Rating* igual a *Moderate* que é menor do que o *Minimum Importance Rating* do processo (*Important*). Nesse caso, as duas atividades ficam desabilitadas pois a instância possui uma atividade com *Importance Rating* maior ou igual que *Important*.

5.2.2 Tomada de decisão baseada em estatísticas anteriores

Na Figura 46(a) e na Figura 46(b), os usuários podem ver a proporção em que cada atividade já foi realizada anteriormente para cenários semelhantes. Com essa informa-

ção, é possível que o profissional entenda como a organização se comporta em cenários particulares e também cobrar a execução de uma atividade por um usuário de outro cargo.

No nosso caso de exemplo, a atividade *Be admitted in health units* foi executada em 52% dos casos para o cenário em questão. Assim, esta foi opção mais escolhida e serve de indicativo ao *Doctor* de como o hospital costuma funcionar. Este tipo de informação é especialmente útil para funcionários que não estão habituados com os costumes da empresa. Todavia, baseado em seu conhecimento profissional, o usuário pode julgar que a atividade não é a mais apropriada para o caso em particular.

É importante ressaltar que essa informação estatística não limita, de nenhuma forma, a execução de uma atividade. Ela serve apenas como guia auxiliar ao usuário baseado em lições aprendidas de execuções anteriores.

5.2.3 Tomada de decisão baseada no resultado da técnica de priorização AHP

No nosso caso de exemplo, selecionamos a opção *Set Goal* no momento da criação do processo. Essa opção permite introduzir os resultados da técnica AHP para priorização das atividades. Dessa forma, durante a execução de uma instância, o usuário pode utilizar a informação contextual *Best choice* que, baseada no resultado da utilização da técnica AHP, indica qual atividade é a mais apropriada para alcançar o objetivo do processo.

Em se tratando de um atendimento de emergência, utilizamos como objetivo principal o atendimento imediato ao paciente. Dessa forma é apropriado que o critério *Tempo* possua maior importância e tenha um peso maior quando comparado a outros critérios, por exemplo, como *Custo*. Assim, atividades que estejam mais ligadas ao critério *Tempo* tem maior chance de serem indicadas ao usuário.

Tomando como exemplo a Figura 47, temos duas atividades disponíveis para o usuário: *Undergo surgery* e *Be admitted in health units*. Dentre estas atividades e baseado nos critérios definidos pelo Administrador, a atividade que mais se aproxima do objetivo do processo (obteve a maior prioridade resultante da AHP) é a *Be admitted in health units*, sinalizada pela etiqueta *Best choice*.

5.2.3.1 Atualização das Atividades e Acesso Concorrente

Como vimos, uma instância pode ter várias atividades disponíveis distribuídas entre usuários com diferentes funções dentro da organização. A cada atividade executada, o REFlex Engine define quais serão as atividades disponíveis e quais delas estarão habilitadas no próximo passo da execução.

Para evitar que as instâncias cheguem a um estado incorreto, o Web-REFlex controla a concorrência entre os usuários e, somente quando a instância está atualizada, o usuário poderá executar suas atividades sem impendimentos. Caso contrário, o profissional é informado que a sua cópia da instância está desatualizada e é redirecionado para o conjunto

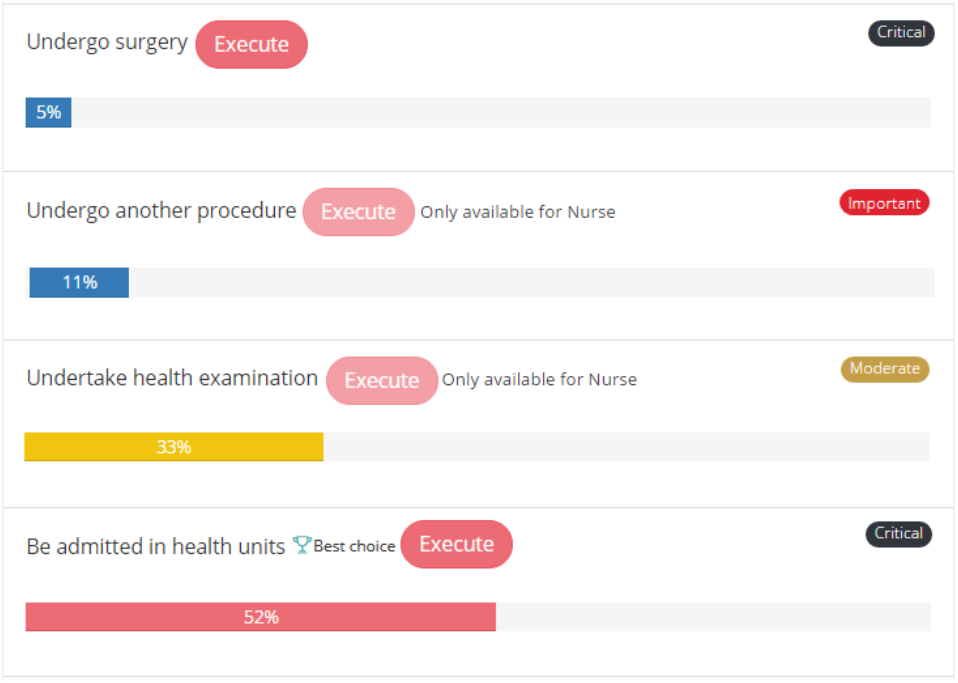


Figura 47 – Atividades avaliadas pela abordagem AHP

mais atualizado de instâncias disponíveis. Essa dinâmica ocorre até que o processo possa ser concluído com êxito. No nosso caso, quando o paciente recebe a alta médica.

5.3 Análise e Síntese do Capítulo

O objetivo deste capítulo foi apresentar a execução de um processo, tipicamente colaborativo, utilizando a abordagem definida neste trabalho. A solução proposta é um meio complementar para auxiliar o gerenciamento dos processos do *framework* REFlex, agregando, para isso, informações gerais do contexto do processo e permitindo a distribuição do processo para acesso concorrente a múltiplos usuários.

Analisando o caso de exemplo, é possível compreender e perceber várias situações em que os usuários podem tirar proveito de informações do contexto para auxiliá-los na tomada de decisões durante o gerenciamento do processo. Entretanto, a utilização do Web-REFlex em um ambiente organizacional real pode trazer uma visão mais ampla das contribuições propostas, que tem como objetivo mitigar o problema de *Context Tunneling*. Da mesma forma, a utilização da ferramenta por usuários reais pode levantar possíveis limitações que não foram percebidas até o momento.

De maneira geral, esse capítulo apresentou um caso de exemplo de utilização da ferramenta para validar a proposta, garantindo a corretude da abordagem apresentada nessa dissertação. Para o próximo capítulo, apresentamos as considerações finais da dissertação, apresentando as conclusões e os trabalhos futuros.

6 CONSIDERAÇÕES FINAIS

A definição de processos, para gerenciar as atividades nas organizações empresariais, consolidou-se como uma prática eficiente no ambiente de negócios atual. Para descrever um processo de negócio, diversas ferramentas podem ser utilizadas. Nesse cenário, destacam-se as ferramentas de gerenciamento declarativas, que definem atividades e regras para modelar o comportamento de um processo. Essas oferecem uma maior flexibilidade na definição de processos que a abordagem imperativa, na qual é necessário modelar todas as possibilidades de interações entre as atividades.

Entretanto, a maioria das ferramentas declarativas existentes possui restrições relacionadas ao controle da execução de processos de negócio colaborativos. A limitação contextual, também conhecida como *Context Tunneling*, ocorre quando os dados necessários para controlar a execução do processo são visíveis apenas para seus respectivos usuários, portanto, o contexto global da instância não está diretamente disponível para consulta. Desta forma, os usuários não conseguem tirar vantagem dessas informações globais para decidir quem e/ou qual passo deve ser tomado para satisfazer as suas exigências atuais do processo.

Este trabalho apresenta uma solução baseada na nuvem, chamada Web-REFlex, para evitar o problema de *Context Tunneling* na execução de processos de negócio colaborativos. Para tal, nossa abordagem propõe uma extensão ao framework REFlex, um motor de regras capaz de executar processos de negócio declarativos, adicionando características da abordagem *Case Management* que ampliam o acesso a informações de contexto do processo ao usuário.

O Web-REFlex utiliza o REFlex para controlar o fluxo de atividades em tempo de execução. A forma de representação de um processo declarativo utilizada pelo REFlex permite salvar apenas o estado atual de uma instância, sem o *trace* completo, e retomar à sua execução sem perda de informações. Assim, devido a tal característica, foi possível pausar ou retomar o fluxo de trabalho do processo com intermédio do banco de dados relacional, permitindo o acesso concorrente a múltiplos usuários.

Para controle dos usuários, o Web-REFlex utiliza a noção de gerenciamento de cargos e funções. Para tanto, atribui a responsabilidade de uma atividade a um determinado cargo e isto permite que determinadas atividades apenas possam ser executadas por um seleto grupo de usuários.

Objetivando a reutilização de informações de contexto, o Web-REFlex calcula métrica da frequência de uso das atividades a partir das decisões tomadas em execuções anteriores do processo, e apresenta essas métricas de forma relevante para ajudar os usuários na tomada de decisões.

Adicionalmente, utilizamos uma técnica de *MCDM*, a *AHP*, para ajudar o usuário

a escolher a atividade que melhor se adeque aos objetivos do processo. Para isso, cada processo possui um objetivo, critérios e as atividades serão as alternativas para alcançá-lo. Cada elemento desta hierarquia tem um peso numérico para que os elementos possam ser comparados e assim o Web-REFlex possa indicar a melhor alternativa para uma instância.

Assim, Web-REFlex pode ser considerada como uma solução completa para o gerenciamento de processos de negócio colaborativos. Em resumo, através da ferramenta, é possível: i) permitir o controle de múltiplos usuários; ii) permitir o controle de múltiplas instâncias de processos; iii) computar métricas estatísticas baseadas em decisões tomadas previamente para um cenário, e disponibilizar o conhecimento dessas para ajudar aos usuários na tomada de decisões; e iv) priorizar atividades de acordo com seu nível de criticidade.

Para demonstrar nossa solução, descrevemos a execução de um processo de negócio colaborativo. O processo faz uso das informações de contexto e do motor de regras do REFlex para simular os resultados. Embora simples, o caso de uso ilustra a utilidade da abordagem para tais aplicações do mundo real.

6.1 Limitações e trabalhos futuros

Embora apresente várias vantagens se comparado com as soluções atuais, a solução ainda apresenta algumas limitações e possibilidades de melhorias que podem ser abordadas em trabalhos futuros. Como exemplo, podemos destacar as seguintes atividades:

- Utilização da ferramenta em um cenário real com usuários em uma organização empresarial. Assim, será possível verificar as melhorias proporcionadas pela ferramenta e identificar suas possíveis limitações.
- Permitir a modelagem gráfica de um processo através do Web-REFlex.
- Dar suporte as fases do ciclo de vida BPM. Juntamente com o tópico anterior, a modelagem gráfica, será possível realizar a modelagem do processo no Web-REFlex.
- Adicionar suporte ao fluxo de dados do processo. Dessa forma, será possível personalizar cada processo com dados da instância, como exemplo, para um processo hospitalar: informações pessoais do paciente, resultados de exames, nome do médico, tratamento prescrito, etc.
- Tornar possível a obtenção de informações sobre as regras do processo. Isto pode ser oferecido de forma gráfica, através da notação REFlex, e de forma textual, durante a execução das instâncias, para usuários leigos.
- Realizar uma análise mais detalhada sobre as técnicas *MCDM* e, possivelmente, a implementação de um novo método a ser inserido no Web-REFlex.

- No âmbito do software social, a implementação de um chat facilitaria a comunicação entre os envolvidos no processo, permitindo a cobrança da execução de atividades e troca de informações de forma mais efetiva.
- Criação de um sistema de notificações para alertar os participantes sobre a ocorrência de um evento, tais como: realização de uma atividade por outro integrante do processo, mudanças do nível de criticidade de uma atividade ou ao receber uma nova mensagem do chat.

REFERÊNCIAS

AALST, W. et al. Business process management: a personal view. *Business Process*, 2004.

AALST, W. M. Van der. The application of petri nets to workflow management. *Journal of circuits, systems, and computers*, World Scientific, v. 8, n. 01, p. 21–66, 1998.

AALST, W. M. Van der; WESKE, M.; GRÜNBAUER, D. Case handling: a new paradigm for business process support. *Data & Knowledge Engineering*, Elsevier, v. 53, n. 2, p. 129–162, 2005.

AALST, W. Van der; STOFFELE, M.; WAMELINK, J. Case handling in construction. *Automation in Construction*, Elsevier, v. 12, n. 3, p. 303–320, 2003.

ATHENA, P. *Flower User Manual*. [S.l.], 2002.

BAZÁN, P. *Implementación de procesos de negocio a través de servicios aplicando metamodelos, software distribuido y aspectos sociales*. Tese (Doutorado) — Facultad de Informática, 2015.

BUKHSH, Z. A. *BPMN Plus: a modelling language for unstructured business processes*. Dissertação (Mestrado) — University of Twente, 2015.

BURATTIN, A.; MAGGI, F. M.; AALST, W. M. van der; SPERDUTI, A. Techniques for a posteriori analysis of declarative processes. In: IEEE. *Enterprise Distributed Object Computing Conference (EDOC), 2012 IEEE 16th International*. [S.l.], 2012. p. 41–50.

CARRARA, A. R. *Implantação de sistema BPMS para a gestão por processos: uma análise crítica.* Tese (Doutorado) — Universidade de São Paulo, 2011.

CARVALHO, R. M. d. Reflex: a graph-based model for declarative business processes. Universidade Federal de Pernambuco, 2015.

CARVALHO, R. M. de; MILI, H.; GONZALEZ-HUERTA, J.; BOUBAKER, A.; LESHOB, A. Comparing condec to cmmn. 2016.

CARVALHO, R. M. de; SILVA, N. C.; LIMA, R. M.; CORNÉLIO, M. L. Reflex: an efficient graph-based rule engine to execute declarative processes. In: IEEE. *2013 IEEE International Conference on Systems, Man, and Cybernetics*. [S.l.], 2013. p. 1379–1384.

CBOK, B. Guide to the business process management common body of knowledge. *Versão 2.0. 2009. Disponível em: www. abpmp. org. Acesso em: 25 nov 2012*, 2009.

CHANG, J. F. *Business process management systems: strategy and implementation*. [S.l.]: CRC Press, 2016.

DCR. *DCR GRAPHS*. 2011. Disponível em: <http://dcrgraphs.com/cases/>.

DUSTDAR, S. Towards context-based autonomic services. In: *WEBIST (1)*. [S.l.: s.n.], 2007. p. 7–8.

ECKHARDT, A. *Implementing A Case-handling System: Supporting the Design Process.* Dissertação (Mestrado), 2015.

EIJNDHOVEN, T. V.; IACOB, M.-E.; PONISIO, M. L. Achieving business process flexibility with business rules. In: IEEE. *Enterprise Distributed Object Computing Conference, 2008. EDOC'08. 12th International IEEE.* [S.l.], 2008. p. 95–104.

ELLIS, C. A.; GIBBS, S. J.; REIN, G. Groupware: some issues and experiences. *Communications of the ACM*, ACM, v. 34, n. 1, p. 39–58, 1991.

FAVRETTO, J.; NOTTAR, L. A. Utilização da metodologia analytic hierarchy process (ahp) na definição de um software acadêmico para uma instituição de ensino superior do oeste catarinense. *Sistemas & Gestão*, v. 11, n. 2, p. 183–91, 2016.

FILHO, E. A. C. *Uma contribuição ao estudo do problema da escolha entre manuntenção própria ou contratada numa empresa de transporte aéreo, tomando-se por base o método Fuzzy Multi-Criteria Decision-Making (MCDM).* Tese (Doutorado) — Universidade Federal do Rio de Janeiro, 2011.

GIACOMO, G. D.; DUMAS, M.; MAGGI, F. M.; MONTALI, M. Declarative process modeling in bpmn. In: SPRINGER. *International Conference on Advanced Information Systems Engineering.* [S.l.], 2015. p. 84–100.

GOEDERTIER, S.; VANTHIENEN, J.; CARON, F. Declarative business process modelling: principles and modelling languages. *Enterprise Information Systems*, Taylor & Francis, v. 9, n. 2, p. 161–185, 2015.

GOMEDE, E.; BARROS, R. Utilizando o método analytic hierarchy process (ahp) para priorização de serviços de ti: Um estudo de caso. *VIII Simpósio Brasileiro de Sistemas de Informação, São Paulo*, p. 408–419, 2012.

GOMEZ, T. H.; FRANCO, A. Business rules extraction from business process specifications written in natural language. *Business Rules Journal*, v. 11, n. 7, 2010.

GONCALVES, J. E. L. As empresas são grandes coleções de processoss. In: *Revista de administração de empresas 40.1.* [S.l.: s.n.], 2000. p. 6–9.

GOUVÊA, M. T. A.; SANTORO, F. M.; CAPPELLI, C.; PIMENTEL, M. Externalização do conhecimento através de group storytelling: um estudo de caso em tutoria online. 2016.

HAY, D.; HEALY, K. A.; HALL, J. et al. Defining business rules-what are they really. *Final Report*, Business Rule Group, 2000.

HILDEBRANDT, T. T.; MUKKAMALA, R. R. Declarative event-based workflow as distributed dynamic condition response graphs. *arXiv preprint arXiv:1110.4161*, 2011.

IBM. *IBM Case Manager on Cloud - Service Definition.* [S.l.], 2016.

JACKSON, D. *Software Abstractions: logic, language, and analysis.* [S.l.]: MIT press, 2012.

KO, R. K. A computer scientist's introductory guide to business process management (bpm). *Crossroads*, ACM, v. 15, n. 4, p. 4, 2009.

LIU, C.; LI, Q.; ZHAO, X. Challenges and opportunities in collaborative business process management: Overview of recent advances and introduction to the special issue. *Information Systems Frontiers*, Springer, v. 11, n. 3, p. 201–209, 2009.

MARIN, M.; HULL, R.; VACULÍN, R. Data centric bpm and the emerging case management standard: A short survey. In: SPRINGER. *International Conference on Business Process Management*. [S.l.], 2012. p. 24–30.

MARIN, M. A. Introduction to the case management model and notation (cmmn). *arXiv preprint arXiv:1608.05011*, 2016.

MARINS, C. S.; SOUZA, D. d. O.; BARROS, M. d. S. O uso do método de análise hierárquica (ahp) na tomada de decisões gerenciais–um estudo de caso. *XLI SBPO*, v. 1, 2009.

MCCORMACK KEVIN P., e. W. C. J. *Business process orientation: Gaining the e-business competitive advantage*. [S.l.: s.n.], 2001.

MENDLING, J. Event-driven process chains (epc). In: *Metrics for Process Models*. [S.l.]: Springer, 2008. p. 17–57.

MONTALI, M.; TORRONI, P.; CHESANI, F.; MELLO, P.; ALBERTI, M.; LAMMA, E. Abductive logic programming as an effective technology for the static verification of declarative business processes. *Fundamenta Informaticae*, IOS Press, v. 102, n. 3-4, p. 325–361, 2010.

MOTAHARI-NEZHAD, H. R.; SWENSON, K. D. Adaptive case management: Overview and research challenges. In: IEEE. *2013 IEEE 15th Conference on Business Informatics*. [S.l.], 2013. p. 264–269.

MUIJRES, G. Master thesis proposal: Business process enactment in healthcare. 2011.

NOVIK, A. *Web App Architecture – the Spring MVC – AngularJs stack*. 2015. Disponível em: <https://www.javacodegeeks.com/2015/01/web-app-architecture-the-spring-mvc-angularjs-stack.html>.

OULD MARTYN A. E OULD, M. A. *Business Processes: Modelling and analysis for re-engineering and improvement*. [S.l.: s.n.], 1995.

PESIC, M. Constraint-based workflow management systems: shifting control to users. 2008.

PESIC, M.; SCHONENBERG, H.; AALST, W. M. van der. Declare: Full support for loosely-structured processes. In: IEEE. *Enterprise Distributed Object Computing Conference, 2007. EDOC 2007. 11th IEEE International*. [S.l.], 2007. p. 287–287.

PESSÔA, M. S. d. P.; STORCH, S. Escolhas tecnológicas para o gerenciamento por processos. *ROTONDARO, RG (Coord.). Gestão integrada de processos e da tecnologia da informação. São Paulo: Atlas*, p. 190–218, 2006.

PIMENTEL, M.; GEROSA, M. A.; FILIPPO, D.; RAPOSO, A.; FUKS, H.; LUCENA, C. J. P. Modelo 3c de colaboração para o desenvolvimento de sistemas colaborativos. *Anais do III Simpósio Brasileiro de Sistemas Colaborativos*, p. 58–67, 2006.

RASHID, K. Social networking and bpm of the future. *BpTrends Column. Setiembre*, 2009.

REIJERS, H. A.; SLAATS, T.; STAHL, C. Declarative modeling–an academic dream or the future for bpm? In: *Business Process Management*. [S.l.]: Springer, 2013. p. 307–322.

SAATY, T. L. *Theory and applications of the analytic network process: decision making with benefits, opportunities, costs, and risks*. [S.l.]: RWS publications, 2005.

SCHREPFER, M. *Modeling Guidelines for Business Process Models*. Tese (Doutorado) — Humboldt - UniversitatNIVERSITAT ZU BERLIN, 2010.

SILVA, N.; CARVALHO, R.; LIMA, R.; OLIVEIRA, C. Integrating declarative processes and soa: A declarative web service orchestrator. In: THE STEERING COMMITTEE OF THE WORLD CONGRESS IN COMPUTER SCIENCE, COMPUTER ENGINEERING AND APPLIED COMPUTING (WORLDCOMP). *Proceedings of the International Conference on Semantic Web and Web Services (SWWS)*. [S.l.], 2013. p. 5.

SILVA, N. C. Reflex: rule engine for flexible processes. Universidade Federal de Pernambuco, 2014.

SOLUTION, B. O. *Bonita Open Solution*. 2009. Disponível em: <http://es.bonitasoft.com/>.

UBANDO, A. T.; GUE, I. H. V.; AGUILAR, K. D. T. Analytical hierarchy process with artificial neural network: A case study of algal biofuel production impact prioritization in the philippines. In: IEEE. *Region 10 Conference (TENCON), 2016 IEEE*. [S.l.], 2016. p. 961–965.

VANDERFEESTEN, I.; REIJERS, H. A.; AALST, W. M. P. *Product based workflow design with case handling systems*. [S.l.]: Beta, Research School for Operations Management and Logistics, 2006.

VARGAS, R. V.; IPMA-B, P. Utilizando a programação multicritério (analytic hierarchy process-ahp) para selecionar e priorizar projetos na gestão de portfólio. In: *PMI Global Congress*. [S.l.: s.n.], 2010. v. 29, p. 31.

WANG, W.-P. A fuzzy linguistic computing approach to supplier evaluation. *Applied Mathematical Modelling*, Elsevier, v. 34, n. 10, p. 3130–3141, 2010.

WEERAWARANA, S.; CURBERA, F.; LEYMANN, F.; STOREY, T.; FERGUSON, D. F. *Web services platform architecture: SOAP, WSDL, WS-policy, WS-addressing, WS-BPEL, WS-reliable messaging and more*. [S.l.]: Prentice Hall PTR, 2005.

WHITE, S. A. Introduction to bpmn. *IBM Cooperation*, v. 2, n. 0, p. 0, 2004.

WILLEMANN, C. *O que é BPM, BPMN e BPMS*. 2017. Disponível em: <http://claudiowillemann.com/o-que-bpm-bpmn-bpms>.

WURTZEL, M. *What Is BPM?* Mcgraw-hill, 2012. ISBN 9780071802253. Disponível em: <https://books.google.com.au/books?id=0ed7Xawz3hsC>.

ZHU, W.-D.; BENOIT, B.; JACKSON, B.; LIU, J.; MARIN, M.; MEENA, S.; OSPINA, J. F.; RIOS, G. et al. *Advanced Case Management with IBM Case Manager*. [S.l.]: IBM Redbooks, 2015.

ZUGAL, S.; SOFFER, P.; HAISJACKL, C.; PINGGERA, J.; REICHERT, M.; WEBER, B. Investigating expressiveness and understandability of hierarchy in declarative business process models. *Software & Systems Modeling*, Springer, v. 14, n. 3, p. 1081–1103, 2015.

APÊNDICE A – SCRIPTS

- Criação do Super Administrador

```
-- Dados do primeiro Super Administrador, Super REFlex.
INSERT INTO tb_user(id, name, avatar, email, password, created_at)
VALUES (nextval('hibernate_sequence'), 'Super REFlex', 'reflex.png',
'reflex@reflex.cin.ufpe.br', SENHA CRIPTOGRAFADA, now());
```

- Criação dos tipos de permissão

```
INSERT INTO tb_permission(id, role, created_at) VALUES
(nextval('hibernate_sequence'), 'ROLE_SUPER', now());

INSERT INTO tb_permission(id, role, created_at) VALUES
(nextval('hibernate_sequence'), 'ROLE_ADMIN', now());

INSERT INTO tb_permission(id, role, created_at) VALUES
(nextval('hibernate_sequence'), 'ROLE_USER', now());
```

- Associação da permissão ROLE_SUPER (Super Administrador) para o usuário recém-criado (Super REFlex)

```
INSERT INTO tb_user_permission(user_id, permission_id)
VALUES ('1', '2');
```

APÊNDICE B – ARQUIVO XML

Listing B.1 – Test

```
<process>
    <globalData>
        <variable name="restauracao" type="STRING" initialValue="
            Restauracao"></variable>
        <variable name="portugues" type="STRING" initialValue="Portugues">
            </variable>
        <variable name="feridos" type="STRING_LIST"></variable>
        <variable name="quantidade_feridos" type="INT" initialValue="0"></
            variable>
        <variable name="result_checkIn" type="STRING"></variable>
        <variable name="result_informar" type="BOOLEAN"></variable>
        <variable name="result_bombeiros" type="BOOLEAN"></variable>
        <variable name="result_suprimentos" type="BOOLEAN"></variable>
        <variable name="zero" type="INT" initialValue="0"></variable>
    </globalData>

    <namespaces>
        <namespace code="ax23" name="http://reflex.cin.ufpe.br/xsd" />
        <namespace code="ns0" name="http://reflex.cin.ufpe.br" />
    </namespaces>

    <activities>
        <activity name="Enviar Suprimentos">
            <serviceBinding operation="enviarSuprimentos"
                wsdlUrl="http://localhost:8080/ode/services/Prefeitura?wsdl"
                portType="PrefeituraPortType" binding="
                    PrefeituraSOAP11Binding" />
            <dataInputBinding>
                <variableBinding variable="hospital" global="false"
                    type="STRING" expression="xpath:/enviarSuprimentos/
                        hospital" />
            </dataInputBinding>
            <dataOutputBinding>
                <variableBinding variable="result_suprimentos"
                    global="true" expression="//ns0:enviarSuprimentosResponse
                        /ns0:return" />
            </dataOutputBinding>
        </activity>

        <activity name="Chamar Bombeiros">
            <serviceBinding operation="chamarBombeiros"
                wsdlUrl="http://localhost:8080/ode/services/Prefeitura?wsdl"
```

```
37              portType="PrefeituraPortType" binding="
                    PrefeituraSOAP11Binding" />
            <dataInputBinding>
39              <variableBinding variable="quantidade" global="false"
                    type="int" expression="xpath:/chamarBombeiros/quantidade"
                        />
41          </dataInputBinding>
            <dataOutputBinding>
43              <variableBinding variable="result_bombeiros" global="true"
                    expression="//ns0:chamarBombeirosResponse/ns0:return" />
45          </dataOutputBinding>
        </activity>
47
        <activity name="Listar Feridos">
49          <serviceBinding operation="getFeridos"
                wsdlUrl="http://localhost:8080/ode/services/Prefeitura?wsdl"
51              portType="PrefeituraPortType" binding="
                    PrefeituraSOAP11Binding" />
            <dataInputBinding>
53              <variableBinding variable="nivel" global="false"
                    type="STRING" expression="xpath:/getFeridos/nivel" />
55          </dataInputBinding>
            <dataOutputBinding>
57              <variableBinding variable="feridos" global="true"
                    expression="//ns0:getFeridosResponse/ns0:return/
                        ax23:feridos" />
59              <variableBinding variable="quantidade_feridos"
                    global="true" expression="//ns0:getFeridosResponse/
                        ns0:return/ax23:quantidade" />
61          </dataOutputBinding>
        </activity>
63
        <activity name="Deslocar Pessoas">
65          <serviceBinding operation="acionarDeslocamento"
                wsdlUrl="http://localhost:8080/ode/services/Prefeitura?wsdl"
67              portType="PrefeituraPortType" binding="
                    PrefeituraSOAP11Binding" />
            <dataInputBinding>
69              <variableBinding variable="Hospital" global="false"
                    expression="xpath:/acionarDeslocamento/hospital" />
71              <variableBinding variable="feridos" global="true"
                    expression="xpath:/acionarDeslocamento/pessoas" />
73          </dataInputBinding>
        </activity>
75
        <activity name="Checar disponibilidade">
77          <serviceBinding operation="informar"
```

```
            wsdlUrl="http://localhost:8080/ode/services/Hospital?wsdl"
                portType="HospitalPortType"
79          binding="HospitalSOAP11Binding" />
        <dataInputBinding>
81          <variableBinding variable="hospital" global="false"
                expression="xpath:/informar/hospital" />
83          <variableBinding variable="quantidade_feridos"
                global="true" expression="xpath:/informar/quantidade" />
85      </dataInputBinding>
        <dataOutputBinding>
87          <variableBinding variable="result_informar" global="true"
                expression="//ns0:informarResponse/ns0:return" />
89      </dataOutputBinding>
    </activity>
91
    <activity name="CheckIn Hospital Restauracao">
93      <serviceBinding operation="checkIn"
            wsdlUrl="http://localhost:8080/ode/services/Hospital?wsdl"
                portType="HospitalPortType"
95          binding="HospitalSOAP11Binding" />
        <dataInputBinding>
97          <variableBinding variable="restauracao" global="true"
                expression="xpath:/checkIn/hospital" />
99          <variableBinding variable="feridos" global="true"
                expression="xpath:/checkIn/pessoas" />
101     </dataInputBinding>
        <dataOutputBinding>
103         <variableBinding variable="result_checkIn" global="true"
                expression="//ns0:checkInResponse/ns0:return" />
105     </dataOutputBinding>
    </activity>
107
    <activity name="CheckIn Hospital Portugues">
109     <serviceBinding operation="checkIn"
            wsdlUrl="http://localhost:8080/ode/services/Hospital?wsdl"
                portType="HospitalPortType"
111         binding="HospitalSOAP11Binding" />
        <dataInputBinding>
113         <variableBinding variable="portugues" global="true"
                expression="xpath:/checkIn/hospital" />
115         <variableBinding variable="feridos" global="true"
                expression="xpath:/checkIn/pessoas" />
117     </dataInputBinding>
        <dataOutputBinding>
119         <variableBinding variable="result_checkIn" global="true"
                expression="//ns0:checkInResponse/ns0:return" />
121     </dataOutputBinding>
```

```
        </activity>
123   </activities>

125   <rules>
        <rule activityA= "CheckIn Hospital Portugues" activityB="CheckIn
            Hospital Restauracao" type="NOT_COEXISTENCE"/>
127     <rule activityA= "Listar Feridos" activityB="Checar
            disponibilidade" type="RESPONSE"/>
        <rule activityA= "Checar disponibilidade" activityB="CheckIn
            Hospital Restauracao" type="PRECEDENCE"/>
129     <rule activityA= "Checar disponibilidades" activityB="CheckIn
            Hospital Portugues" type="PRECEDENCE"/>
        <rule activityA= "Deslocar Pessoas" n="2" type="ABSENCE"/>
131     <rule activityA= "Chamar Bombeiros" n="2" type="ABSENCE">
          <condition template="EQUALS" type="NUMBER" variable1="
              quantidade_feridos" variable2="zero"/>
133     </rule>
      </rules>
135 </process>
```