UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE MATEMÁTICA

FILIPE XAVIER TRINDADE DOS SANTOS
LUCAS SIMÕES DE SOUSA ARNAUD
THALES TEIXEIRA PIRES

Automatização de Processos com Redmine e Activiti BPM

Profª. Silvana Rossetto
Orientador

Rio de Janeiro, Fevereiro de 2017

Automatização de Processos com Redmine e Activiti BPM

Filipe Xavier Trindade dos Santos

Lucas Simões de Sousa Arnaud

Thales Teixeira Pires

Projeto Final de Curso submetido ao Departamento de Ciência da Computação do Instituto de Matemática da Universidade Federal do Rio de Janeiro como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Apresentado por:

_____

Filipe Xavier Trindade dos Santos

_____

Lucas Simões de Sousa Arnaud

_____

Thales Teixeira Pires

Aprovado por:

_____

Prof[a]. Silvana Rossetto

_____

Prof[a]. Valeria Menezes Bastos

_____

Prof[a]. Maria Luiza Machado Campos

RIO DE JANEIRO, RJ - BRASIL

Fevereiro de 2017

Sumário

RESUMO

Automatização de Processos com Redmine e Activiti BPM

Filipe Xavier Trindade dos Santos , Lucas Simões de Sousa Arnaud e Thales Teixeira Pires

Fevereiro/2017

Orientador: Silvana Rossetto
IM/DCC - UFRJ

A eficiência de uma empresa está diretamente relacionada à forma como são conduzidos os seus processos internos. Quanto maior o tamanho da organização, mais importante se torna a sua capacidade de gestão para garantir que eles sejam executados com correção e dentro dos prazos esperados. Uma abordagem efetiva para atingir a eficiência organizacional é a automatização de processos de negócio.

O objetivo deste trabalho é apresentar duas diferentes tecnologias já existentes utilizadas para automatização de processos, o Redmine e os chamados BPMS, e suas principais características, e propor uma forma de integração entre elas, aproveitando as qualidades de ambas para oferecer maior qualidade de gestão, padronização e otimização de recursos na execução de processos de negócio.

ABSTRACT

Automatização de Processos com Redmine e Activiti BPM

Filipe Xavier Trindade dos Santos , Lucas Simões de Sousa Arnaud and Thales Teixeira Pires

Fevereiro/2017

Advisor: Silvana Rossetto
IM/DCC - UFRJ

The efficiency of a company is directly related to the way its internal processes are conducted. The larger the size of the organization, the more important its management capacity becomes to ensure that they are executed with correctness and within the expected time frames. An effective approach to achieving organizational efficiency is the automation of business processes.

The objective of this work is to present two different technologies used for automation of processes, Redmine and the so-called BPMS, and their main characteristics, and to propose a way of integration between them, taking advantage of the qualities of both to offer higher management quality, standardization and optimization of resources in the execution of business processes.

Lista de Figuras

Lista de Tabelas

Lista de Códigos

Lista de Abreviaturas e Siglas

BPM	Business Process Management
BPMS	Business Process Management System
BPMI	Business Process Management Initiative
BPMN	Business Process Management Notation
XML	Extensible Markup Language
OMG	Object Management Group
REST	Representational State Transfer
JVM	Java Virtual Machine
HTML	HiperText Markup Language
CSS	Cascading Style Sheets
ESB	Enterprise Service Bus
CSC	Centro de Serviços Compartilhados
ERP	Enterprise Resource Planning

Capítulo 1

Introdução

1.1 Processos de negócio

As organizações, sejam elas de grandes dimensões ou não, são constituídas por recursos humanos e não-humanos que interagem entre si, como por exemplo na troca de informações operacionais entre diferentes áreas de uma empresa. De acordo com a definição apresentada em BPM CBOK[60], processo é uma agregação de atividades e comportamentos executados por humanos ou máquinas para alcançar um ou mais resultados.

Organizações que adotam processos de negócio bem definidos estão melhor habilitadas a operar sua rotina e encontram mais facilidade na identificação de pontos de falha, oportunidades de melhorias na execução e planejamento desses processos. Isso acaba por viabilizar o aumento da qualidade na entrega de produtos e serviços para seus clientes ao longo do tempo.

Os processos de negócio, quando realizados de forma desorganizada e despadronizada, levam à ineficiência organizacional pelo simples fato da sua execução não ser otimizada. Isso ocasiona elevação dos custos, aumento no retrabalho, insatisfação dos colaboradores e a consequente diminuição na qualidade dos serviços relacionados. Portanto, torna-se altamente necessário uma boa gestão dos processos, a fim de que esses problemas sejam evitados ou mitigados a tempo e não tornem-se um

câncer corporativo.

1.2 Automatização de processos de negócio

Uma boa gestão de processos busca padronizar as etapas de um processo, e garantir um acesso mais rápido e eficiente às informações, estabelecendo melhores condições para a execução e controle de processos de negócio. Para atingir este objetivo, conta-se com o suporte de sistemas de informação que auxiliem na execução das etapas do processo de acordo com os padrões definidos. Estes sistemas também precisam acompanhar a flexibilidade da constante mudança dos processos, bem como fornecer ferramentas ao gestor para monitorar e controlar os processos, o que termina por aperfeiçoar a tomada de decisão no nível estratégico organizacional.

A automatização de processos de negócio também reduz a dependência de atuação humana na execução de algumas tarefas mecânicas e repetitivas que passam a ser executadas pelos sistemas. No entanto, são raros os casos em que a interação humana é totalmente extinta, sendo necessário uma atenção especial para que essas tarefas sejam executadas da maneira e sequência lógica mais otimizada possível.

1.3 Exemplo de processo de negócio com falha

Para ilustrar nossa discussão, descrevemos uma falha real ocorrida em 2013 com uma gigante do mercado de varejo e apresentaremos uma sugestão para o estabelecimento de um processo que busca solucionar, padronizar e controlar a execução dessa atividade para que o mesmo erro não seja observado. Vale notar que sem a adoção de um sistema que automatize a execução das tarefas, é praticamente inviável o correto estabelecimento deste processo.

Em um cenário de operação de um portal de *e-commerce* é muito importante a gestão dos preços praticados pela empresa nas diferentes categorias que compõem o conjunto de produtos ofertados ao consumidor. Erros cometidos por uma precificação manual indevida podem levar a altos prejuízos financeiros e ações na justiça por

parte dos consumidores, além de afetar a imagem da corporação perante o mercado.

O caso ocorrido com a gigante Walmart no portal brasileiro em dezembro de 2013, exemplifica bem o cenário descrito no parágrafo anterior. Neste caso, um computador que custava R$2.398,00 reais foi anunciado por R$580,00, o que representava um diferença de 75% a menos em relação ao valor original. Este caso foi noticiado pela imprensa na época, e também houve bastante repercussão nas redes sociais, deixando muitos clientes insatisfeitos pelo cancelamento da compra[64].

Nesse caso, torna-se necessária a implantação de um processo de precificação manual que envolva a avaliação de regras que limitem a troca de preço sem a aprovação de alçadas superiores. Essa nova gestão tem por objetivo trazer mais qualidade e segurança nas mudanças de preço solicitadas pelos diferentes precificadores das categorias do site, alinhando assim a gestão de preços do portal e evitando trocas de preço equivocadas ou não autorizadas pelos gestores.

O processo inicia com a decisão do analista de uma determinada categoria de produtos do site pela mudança de preço de um produto. O novo preço do produto é definido pelo analista e enviado para aprovação.

A solicitação enviada pelo analista é avaliada quanto a regra de alçada definida pelos gestores. Caso possua uma variação de preço acima de 30%, por exemplo, deverá passar pela aprovação de dois gestores. Em um cenário real de negócios, certamente a regra seria mais complexa, como por exemplo variando em função da categoria do produto. Entretanto, assumiremos a regra da variação de 30% nas demonstrações como forma de simplificar o entendimento e aplicação das regras de negócio estabelecidas para o processo.

Após a avaliação da variação de preço, caso tenha sido acima de 30%, o processo deve ser encaminhado para a aprovação de dois gestores. Caso a variação seja menor ou igual a 30%, a mudança não requer aprovação. Após isso, a troca de preço será efetuada no portal.

A Figura 1.1 descreve de forma visual o fluxo do processo descrito anteriormente:

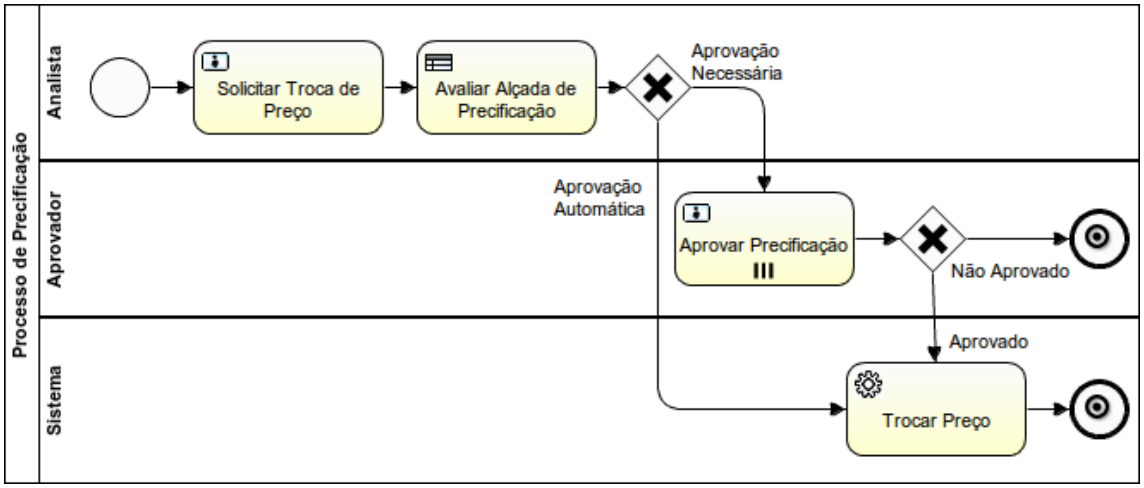

Figura 1.1: Processo de Precificação Manual

1.4 Solução para automatização e gestão de processos

Os sistemas de BPM[10], conhecidos como BPMS[14], possibilitam a implementação de estratégias de operacionalização organizacional por meio de processos, fazendo uso de motores de automação e ferramentas que suportam o ciclo de vida dos processos. Melhorias na visibilidade da operação, facilidades para medir e controlar as etapas de um processo através de sua padronização, adoção de tecnologias de integração de sistemas e automatização de tarefas repetitivas são alguns dos pontos positivos identificados com a adoção desse tipo de solução.

A adoção de soluções BPM trazem também alguns desafios. Por se tratarem de sistemas robustos desenvolvidos para gerenciar processos complexos, sua implantação e manutenção demandam um alto custo: mudanças nos processos irão exigir adaptações nas soluções implantadas, o que pode demandar alto custo e tempo; processos podem ter sua execução impactada devido a situações não previstas durante sua modelagem; além disso, é importante a manutenção de profissionais capacitados para gerenciar e suportar essa nova estrutura tecnológica. Portanto, é necessária uma avaliação criteriosa para que a implantação e automatização seja feita na medida correta.

Considerando a complexidade na adoção de ferramentas BPMS para a gestão e automatização de processos, buscamos uma alternativa capaz de suprir e contemplar fluxos de trabalho menos complexos, garantindo um nível satisfatório de gestão e automação de processos comuns em ambientes corporativos.

Para suprir esta demanda, propomos a utilização do Redmine, ferramenta *open-source* criada inicialmente para a gestão de projetos. Apesar de não ser o objetivo para o qual foi criado, o Redmine se mostra bastante aplicável para a gestão de processos quando consideramos os recursos oferecidos pelo módulo de gerenciamento de tarefas, os quais seguem um fluxo configurável, bem parecido com os fluxos de processos. As possibilidades de cadastrar campos personalizados e de desenvolver novos plugins também contribuem significantemente para que este software se encaixe bastante bem como gestor de processos em contextos com processos simples e que dependem da interação de pessoas. Ressalta-se ainda o baixo custo de implantação para automatização de processos e a possibilidade de manutenção dos fluxos de trabalho por usuários sem experiência com programação.

Entretanto, a utilização do Redmine como gerenciador de processos mostra-se limitada quando a demanda é automatizar um processo um pouco mais complexo, com a ocorrência de fluxos paralelos, ou a automatização de etapas por outros sistemas.

1.5 Objetivos do trabalho

1.5.1 Objetivos gerais

Neste trabalho temos como objetivo o desenvolvimento de uma ferramenta que permita a condução de processos simples e complexos que necessitam de interação humana. Essa ferramenta deverá ser de rápida implantação, simples configuração, possuir uma interface amigável com os atores do processo, e possibilitar rápidas customizações no fluxo dos processos simples, assim como uma condução confiável de processos complexos. As tecnologias envolvidas, bem como a solução final deverão ser de código aberto.

1.5.2 Objetivos específicos

Com o objetivo de estender o Redmine para o atendimento de fluxos de trabalho mais complexos, identificamos a necessidade do desenvolvimento de um motor de processos mais robusto que possibilitasse novas opções e configurações de fluxos na ferramenta. Entretanto, ao avaliarmos a quantidade de possibilidades de fluxos, o desenvolvimento de uma solução desse porte acabou por convergir com soluções praticadas por ferramentas BPMS.

Nesse sentido, torna-se interessante propor uma solução integrada que aproveite a capacidade de modelagem e automatização de fluxos complexos de um BPMS, unida a uma interface amigável e customizável conforme provido pelo Redmine. Nasce assim, a ideia de propor a integração do Redmine com um BPMS. A capacidade de extensão e desenvolvimento de novas funcionalidades do Redmine abriu portas para esta ideia. Alia-se a isto, o fato de que a maioria dos BPMS possuem APIs[7] bem documentadas para integração com outros sistemas.

Para esta integração com o Redmine, buscamos um BPMS que atendesse a determinados critérios. Esta ferramenta deveria ser gratuita, de código-fonte aberto, leve, extensível, contar com uma API[7] REST[8] de fácil integração e possuir uma boa documentação. Ao avaliar as opções disponíveis, identificamos uma ferramenta que se encaixava bastante bem nos critérios adotados: o Activiti BPM. O Activiti BPM é desenvolvido na linguagem de programação Java[56] e é facilmente integrável com aplicações existentes por sua leveza e API[7] amigável, além de utilizar a notação BPMN 2.0[13] para a modelagem dos processos.

Como principal desafio desta proposta de uma ferramenta integrada de automatização e gestão de processos, identificamos a definição de um modelo de execução que fosse capaz de aliar o benefício de ambas as plataformas, sem acrescentar muita complexidade durante a implantação de uma solução deste porte. Além disso, o desafio da sincronização de dados e estados entre as ferramentas também assumiu papel fundamental nas decisões de modelagem durante o desenvolvimento desta integração.

1.6 Organização do texto

O restante deste texto está organizado da seguinte forma:

No capítulo 2 apresentamos como o Redmine pode ser utilizado como gestor de processos. No capítulo 3 introduzimos informações importantes sobre o contexto de gestão de processos de negócio. No capítulo 4 detalhamos a solução proposta para automatização de processos de negócio complexos através da integração do Redmine com um sistema de gerenciamento de processos. No capítulo 5 avaliamos a proposta apresentada e discutimos melhorias futuras. No capítulo 6 apresentamos as conclusões do trabalho.

Capítulo 2

Redmine

2.1 Introdução

Criado em 2006 por Jean-Philippe Lang, o Redmine[38] pode ser definido como uma aplicação *web* desenvolvida para o gerenciamento de projetos. Desenvolvido na linguagem de programação Ruby[61], utiliza a *framework* Rails[57] para suportar sua arquitetura *web*. Conforme apresentado no livro Mastering Redmine[58], este sistema pode ser considerado um dos carros-chefes em soluções para a gestão de projetos no mundo *open-source*.

Neste capítulo vamos explicar como o Redmine, uma ferramenta criada inicialmente para a gestão de projetos, foi utilizada para gestão de processos. Vamos apresentar também, a capacidade extensiva desta ferramenta através do desenvolvimento de *plugins*. Por último, vamos abordar as limitações do Redmine que nos motivaram a buscar novas alternativas, e com isso atingir um estágio mais avançado na automatização de processos.

2.2 Estrutura básica do Redmine

A estrutura básica de gerenciamento de projetos no Redmine é composta por 6 principais elementos. São eles:

2.2.1 Projetos

Projetos são o objeto central do Redmine. Eles são compostos por diversos módulos que acrescentam diferentes dimensões para o seu gerenciamento, como gerenciamento de tarefas, planejamento de etapas e marcos no projeto, acompanhamento do progresso das tarefas em um diagrama de *Gantt*, *Wiki* para organização do conhecimento, entre outros.

2.2.2 Tarefas

São as unidades básicas de execução de trabalho dos projetos (2.2.1). Elas contêm os dados relevantes para o seu gerenciamento (e.g, tipo (2.2.3), situação (2.2.5)), e a sua execução (e.g, título, descrição) e dados adicionais que podem variar entre os projetos e tipos de tarefa, os quais são cadastrados como campos personalizados (2.2.6). É possível criar uma pequena hierarquia de tarefas, que chamaremos de sub-tarefa, quando uma tarefa é relacionada a outra.

2.2.3 Tipos de tarefa

Tipos de tarefa definem o fluxo de trabalho para a realização de atividades similares. De acordo com o tipo, variam as informações necessárias para a execução da tarefa, a sequência de passos para sua conclusão e as ações que cada membro do projeto pode desempenhar em cada etapa.

2.2.4 Papéis de usuários

Os papéis de usuários definem quais permissões um usuário possui, como, por exemplo, visualizar, adicionar e editar tarefas, editar *Wiki*, adicionar e editar documentos. Os papéis são atribuídos aos usuários em cada projeto que ele participa, portanto, ele pode possuir permissões diferentes dependendo do projeto de que ele é membro. A Figura 2.1 ilustra a tela aonde a configuração das permissões de um papel é feita.

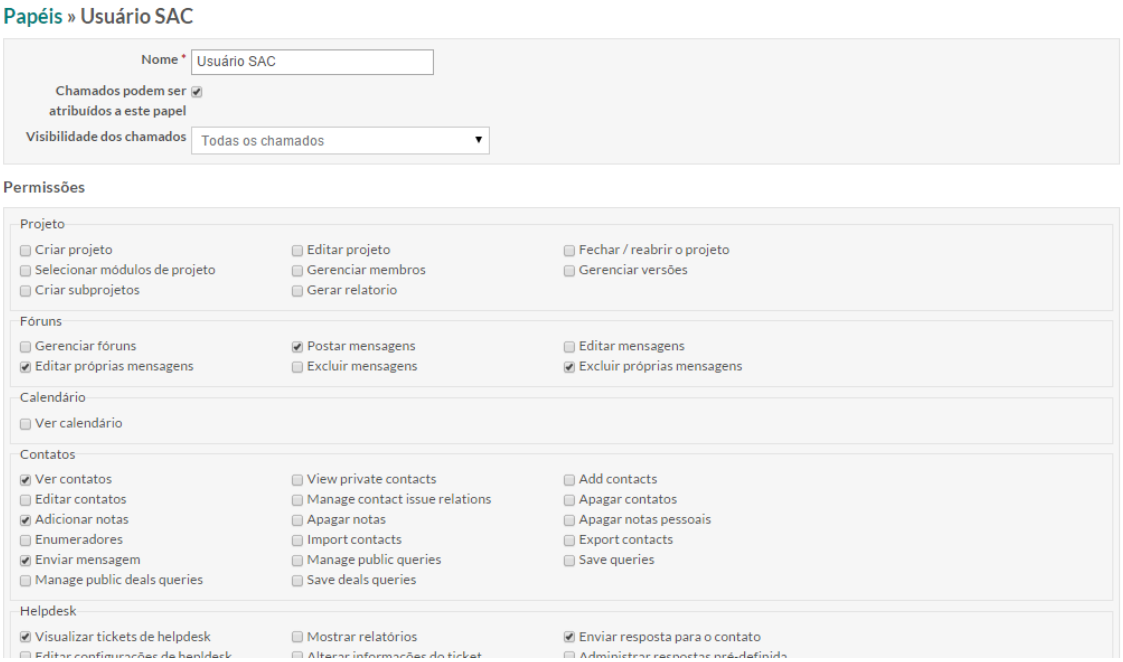

Figura 2.1: Tela de configuração das permissões de um papel

2.2.5 Situação das tarefas

Situação indica em qual etapa do processo uma tarefa se encontra, como "Novo", "Em andamento", "Concluído". Na tela de fluxo de trabalho é possível configurar, para cada papel de usuário, tipo de tarefa e situação, quais campos do formulário podem ser visualizados, ou editados.

2.2.6 Campos personalizados

Todas as tarefas de projetos possuem dados em comum, como data de início, data de término e descrição. No entanto, dependendo das especificidades de um projeto ou tipo de tarefa, podem ser necessárias informações adicionais. Para esses casos existem os campos personalizados. Eles permitem criar novos campos e adicioná-los às tarefas, estendendo as configurações padrão da ferramenta.

2.3 Gestão de processos com o Redmine

A estrutura de projetos do Redmine é altamente configurável. Todos os elementos explicados na seção 2.2 são cadastrados pelos usuários administradores do sistema que são responsáveis por configurá-los e personalizá-los para atender às demandas de cada projeto.

Todas as configurações e personalizações citadas no último parágrafo são feitas exclusivamente pela interface da ferramenta, sem necessidade de alterações no código da aplicação ou arquivos de configuração, o que confere aos administradores capacidade para modelar a estrutura da ferramenta da forma que for mais conveniente para a necessidade dos usuários.

Além disso, o módulo de gerenciamento de tarefas permite ao usuário interagir muito bem com o projeto, tendo total controle de suas tarefas, e ainda submetido a um controle de acesso bem detalhado e flexível.

As vantagens citadas acima, e ainda o fato de o Redmine ser *open-source*, com uma grande comunidade de usuários e desenvolvedores foram motivadoras para a utilização deste como um gerenciador de processos.

2.4 Modelando processos com o Redmine

O primeiro passo para modelar um processo é identificar como representá-lo na estrutura do Redmine.

Nesta seção, descreveremos como utilizar o potencial de configuração do Redmine para utilizá-lo fora do contexto de gerenciamento de projetos e aplicá-lo como ferramenta de automatização de processos. Explicaremos, também, como definir os atores de um processo, as ações permitidas a cada um deles, especificar os fluxos de trabalho e como gerenciar os dados relevantes ao seu contexto. Estes passos serão ilustrados na implementação do caso de uso descrito na seção 1.3

2.4.1 Modelando processos

Para modelar um processo no Redmine, utilizamos os tipos de tarefa (2.2.3) para estruturar os diferentes processos que podem ser iniciados. Durante a criação do tipo de tarefa, define-se qual situação (2.2.5) é a padrão para aquele tipo, em quais projetos ele é utilizado e quais campos (2.2.6) ele utiliza para guardar informações. Os projetos são utilizados para agrupar processos da forma desejada. As tarefas (2.2.2) são utilizadas para representar instâncias de processo, que significam a materialização de um processo em andamento, ou finalizado. Para iniciar um processo, deverá ser criada uma nova tarefa do tipo de tarefa correspondente ao processo que se deseja iniciar. As situações (2.2.5) representam a etapa em que o processo se encontra.

2.4.2 Fluxo de trabalho

A configuração do fluxo de trabalho é onde se orquestra o processo e define-se como os usuários podem manipular os dados dos chamados. Essa configuração é divida em duas partes, permissão de campos e transição de estados, e é feita separadamente para cada papel.

As permissões de campos ditam de que forma um usuário pode interagir com os campos dos chamados que pode editar dependendo da situação atual do chamado. Baseado nessas permissões um campo pode ficar disponível somente para leitura, editável, ou obrigatório. A Figura 2.2 apresenta a tela de configuração de permissão dos campos no Redmine.

Figura 2.2: Tela de configuração de permissão de campos do fluxo de trabalho

A transição de estados define para quais situações um chamado pode ser alterado de acordo com o estado atual, determinando para quais etapas de um processo um usuário consegue conduzir a tarefa. A Figura 2.3 apresenta a tela de configuração das transições de estados no Redmine.

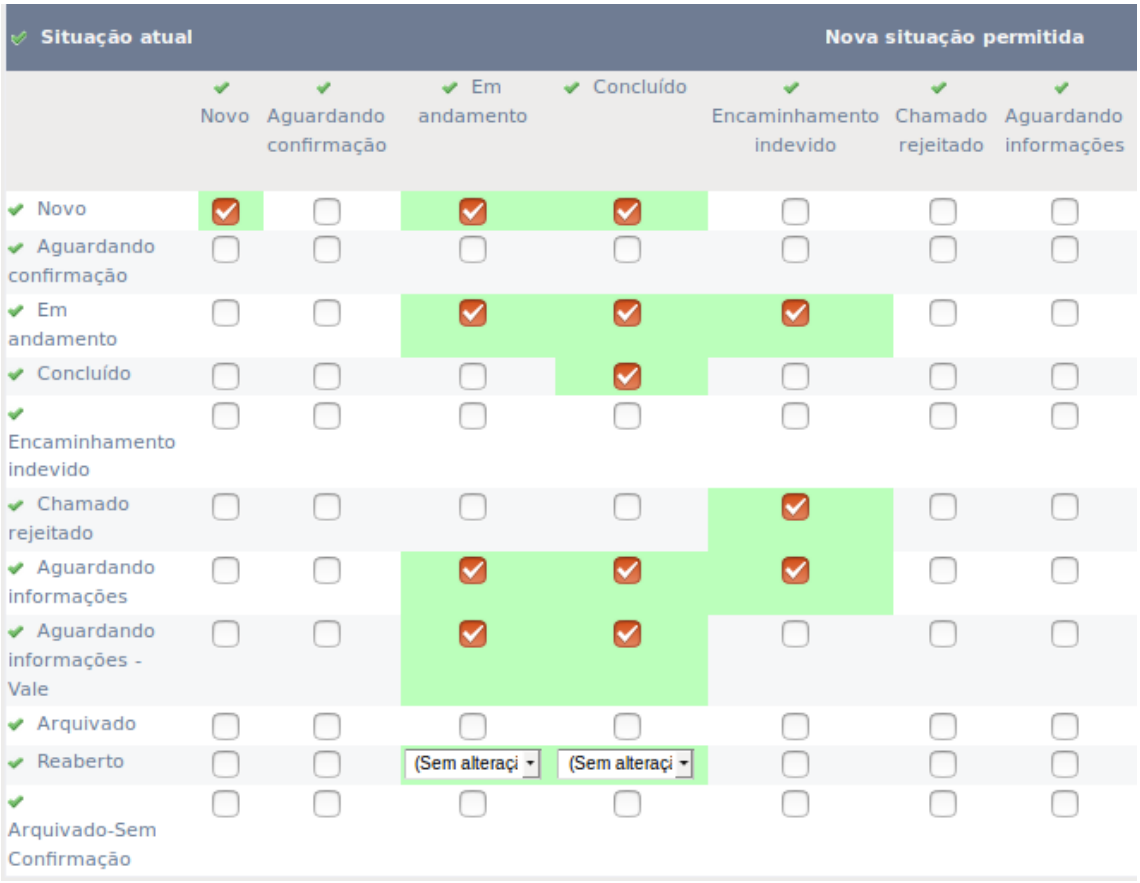

Figura 2.3: Tela de configuração das transições de situação do fluxo de trabalho

2.4.3 Definição dos atores

2.4.3.1 Escolha dos atores

Para definir que um usuário poderá ser um ator de determinado processo, o mesmo deverá ser adicionado ao projeto ao qual o tipo de tarefa correspondente ao processo está relacionado. Esta operação exige que sejam também definidos os papéis que o usuário vai exercer nos processos deste projeto.

O papel que é dado ao usuário agrupa diferentes permissões. Dentre elas, é possível escolher se um usuário pode ter tarefas atribuídas a ele, quais tarefas ele consegue visualizar e quais ações ele é capaz de realizar sobre as tarefas, como ver, editar, criar, ou adicionar notas. As transições de estado de uma tarefa, bem como o controle de acesso aos campos das tarefas também estão incluídos neste conjunto

de permissões, através do Fluxo de trabalho (2.4.2), que é configurado dentro do papel.

2.5 Precificação manual

Para implementar o caso de uso descrito na seção 1.3 no Redmine, foi criado um tipo de tarefa chamado Precificação manual. Também foram criadas as situações Novo, Solicitar troca de preço, Aguardando aprovação, Não aprovado, Trocar preço, Encerrado.

As situações acima representam cada etapa do processo. A transição de situação é feita pelo responsável pela tarefa naquele momento, que também altera para o novo responsável na próxima etapa, caso necessário. A etapa de troca de preço consiste da operação de um sistema externo, e assim que concluída, o responsável pela etapa atual, manualmente altera a situação da tarefa que representa o processo para "Encerrado", o que representa a conclusão da etapa de "Trocar Preço", e a conclusão do processo.

Esta implementação mostra como processos simples podem ser modelados no Redmine, se aproveitando de toda a sua estrutura de tarefas. Desta forma é possível que um usuário acompanhe processos, execute etapas manuais no processo, preenchendo formulários, e tomando decisões para as próximas etapas, como a etapa de aprovação.

A ferramenta escolhida se mostrou uma solução barata, de rápida implementação, e que permite que o próprio usuário administrador seja o responsável pela manutenção da mesma. Muitos cenários problemáticos como o de precificação manual, apresentado nesta seção, são simples e para que fluam melhor requerem uma padronização e automação mínima, que a utilização do Redmine pode oferecer.

2.6 Cenário com aprovação paralela

Apesar do grande potencial de configuração do Redmine que possibilita seu uso para automatização de processos simples, não é possível utilizá-lo em situações de maior complexidade. No exemplo utilizado, modelagem de processos é baseada em uma máquina de estados cujas transições são definidas apenas pelo estado atual do processo e do perfil de acesso dos usuários. Portanto, fluxos mais complexos que possuem, por exemplo, atividades executadas em paralelo ou regras de negócio dependentes de variáveis do contexto de cada execução de um processo não são possíveis de ser modelados apenas com as configurações padrão do Redmine.

Suponhamos que o processo descrito na Figura 1.1 se tornasse um pouco mais complexo e a troca de preço precisasse ser aprovada por três gestores em paralelo, sendo que a reprovação de qualquer um deles reprovasse a precificação. Este caso já não é suportado pelo Redmine. No entanto, este sistema foi desenvolvido de forma a permitir a criação de funcionalidades complementares.

O Redmine foi projetado para ser extensível por meio de *plugins*. Uma funcionalidade da ferramenta pode ser modificada, ou uma nova funcionalidade pode ser criada sem precisar alterar o código original do Redmine. Os *plugins* são desenvolvidos em *Rails*, o mesmo *framework* de programação do Redmine.

Para possibilitar extensões de funcionalidades que envolvem enxertar pedaços de código no meio de uma classe ou de uma tela, o Redmine disponibiliza *hooks* em diversas partes da ferramenta. Hooks são *tags* com um identificador da parte do código em que estão inseridas. Para utilizar um *hook* basta incluir um *hook listener* num *plugin*, e direcionar qual arquivo ou método um determinado *hook* vai disparar.

Muitos plugins desenvolvidos pela comunidade estão disponíveis e podem ser usados para aumentar o poder de modelagem de processos, tornando o Redmine uma ferramenta ainda mais flexível. Esta facilidade nos permite estender o Redmine para possibilitar a implementação de processos mais complexos.

Existem plugins que permitem definir regras de aprovação, regras de mudança automática de etapas mediante eventos, e outros. No entanto, todos estes plugins

são limitados, e consideram um contexto para o qual foram desenvolvidos. Para conduzir processos complexos contando apenas com o Redmine e diversos plugins, conforme a necessidade do processo, seria necessário um esforço muito grande para evoluir os plugins existentes, e criar alguns novos para demandas como disparar uma etapa automatizada que é realizada por outro sistema.

O esforço de evoluir os plugins já existentes não é algo pequeno, pois são desenvolvidos por vários desenvolvedores da comunidade, e muitos daqueles não tem um código organizado ou legível, ou até mesmo não passam nos testes automatizados do Redmine.

A criação de novos plugins também é algo bastante custoso, não só pelo desenvolvimento do código em si, mas porque envolve garantir que cada novo plugin não quebre nenhum teste do Redmine, não gere nenhum problema para uma funcionalidade já existente, e também funcione perfeitamente bem em conjunto com os outros plugins.

O último ponto citado no parágrafo anterior nos leva ao principal obstáculo para que utilizemos apenas o Redmine com plugins para conduzir processos complexos: cada novo plugin adicionado ao conjunto de código original do Redmine adiciona um risco a mais para o funcionamento perfeito do sistema, podendo deixar a ferramenta cada vez mais instável. Portanto, não é indicado que uma mesma instância do Redmine rode em produção com uma quantidade muito grande de plugins, que é o que aconteceria se tivéssemos diversos plugins para atender a necessidade de processos complexos.

Pelas razões citadas acima, preferimos considerar sistemas consolidados que já foram desenvolvidos para gerenciamento de processos complexos, que além de atenderem as nossas necessidades, seriam bastante utilizados, e portanto bem testados. O capítulo 3 então introduzirá o conceito de BPM e os sistemas BPMS.

Capítulo 3

Gerenciamento de Processos de Negócios

3.1 Introdução

Neste capítulo introduziremos o conceito de Gerenciamento de Processos de Negócios, e a notação utilizada para modelar processo de forma padronizada. Falaremos, ademais, dos sistemas que permitem a gestão e condução de processos de forma automatizada. Destacaremos as vantagens e desvantagens deste tipo de sistema, e como esta avaliação contribuiu para a construção de uma solução para a gestão de processos complexos com bastante interação humana.

3.2 BPM

BPM[10] é o acrônimo em inglês para *Business Process Management*, ou em português *Gerenciamento de Processos de Negócio*. De acordo com van der Aalst[63], o BPM é definido como "suporte para processos de negócio, utilizando métodos, técnicas e softwares para projetar, legitimar, controlar e analisar processos operacionais envolvendo humanos, organizações, aplicações, documentos e outras fontes de informação".

Um processo de negócio é definido por um conjunto de atividades coordenadas, relacionadas entre si, que envolvem diferentes pessoas, procedimentos, áreas e tecnologias com o objetivo de gerar valor para a empresa, seja em forma de produtos ou serviços, internos ou externos.

Diferentemente de métodos tradicionais, que são focados no desempenho das unidades funcionais de uma empresa, a adoção do BPM como disciplina de gestão concentra-se no controle e melhoria contínua dos processos funcionais que, na maioria das vezes, permeiam diferentes áreas de negócio.

A aplicação do BPM não deve necessariamente envolver sistemas. Na prática, diversas melhorias de processos podem ser alcançadas sem a utilização de tecnologia ou sistemas de informação. Por exemplo, através da análise e mapeamento de processos de logística, muitas vezes é possível implementar melhorias no processo através do sequenciamento otimizado das tarefas que o constituem, implicando em menos tempo despendido para sua execução.

Entretanto, quando avaliamos a aplicação de BPM em processos críticos, complexos ou até mesmo processos simples mas executados em maior escala, a aplicação de tecnologia torna-se fundamental. A melhoria e automatização de processos através da aplicação de tecnologia é comumente vista na utilização de sistemas de informação para suportar a execução de diferentes tipos de processos, como ocorre normalmente em um CSC[16] (Centro de Serviços Compartilhados).

3.3 BPMN

BPMN[12] é o acrônimo em inglês para *Business Process Management Notation*, ou em português *Notação de Gerenciamento de Processos de Negócio*. Foi criada para representar processos de negócio em forma de diagrama, através de uma notação padronizada e de fácil entendimento por diferentes profissionais, sejam eles desenvolvedores, analistas de negócio ou gestores. Foi criada inicialmente pelo BPMI[11] (Business Process Management Initiative) em 2004, entretanto é mantida atualmente pela OMG[34] (Object Management Group). Sua versão mais atual é a

BPMN 2.0[13], publicada em 2011.

A notação foi concebida sob a perspectiva de cobrir a falta de entendimento entre diferentes departamentos e organizações a cerca de um determinado processo ou conjunto de processos, algo muito frequente no ambiente corporativo. Além disso, através de sua notação padronizada em XML[52] (Extensible Markup Language), diferentes ferramentas podem fazer uso de meta-dados para informatizar a orquestração de processos de negócio.

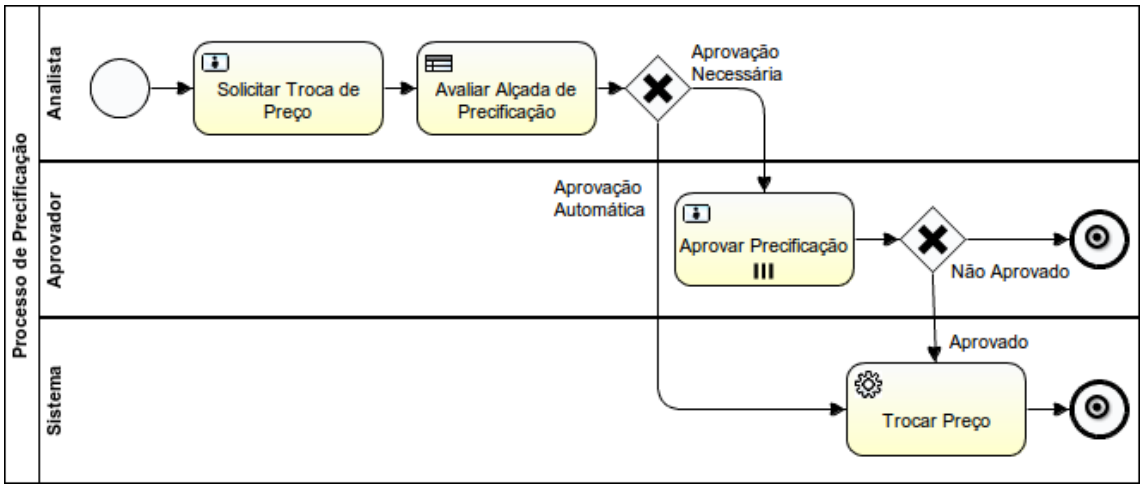

Figura 3.1: Processo representado em BPMN

A Figura 3.1 mostra um exemplo de processo modelado na ferramenta Activiti Designer que será abordada em mais detalhes na seção 4.4.1 e representa o cenário descrito na seção 1.3. Os elementos do modelo serão explicados ao longo deste capítulo.

3.3.1 Objetos

A notação define quatro grupos distintos de objetos para permitir a diagramação de um fluxo de negócio. Os objetos são classificados em objetos de fluxo, artefatos, agrupadores e conectores. São utilizadas figuras geométricas, como retângulos e círculos, além de linhas pontilhadas e tracejadas, entre outros elementos gráficos para representar cada um dos objetos que constituem a notação.

3.3.1.1 Objetos de fluxo

Os objetos de fluxo são os principais elementos do BPMN pois constituem os elementos chave na execução do fluxo de trabalho. Eles são divididos em 3 principais grupos que serão detalhados a seguir: eventos, atividades e decisões.

1. Eventos

   Objetos utilizados para representar que algo aconteceu durante a execução do fluxo. São exemplos de eventos: "chamada de sistema externo recebida", "envio de cancelamento do processo recebido", "a cada 1 minuto". A notação BPMN 2.0 define mais de 60 tipos distintos de eventos. A Figura 3.2 mostra alguns dos diferentes tipos de eventos e suas notações.

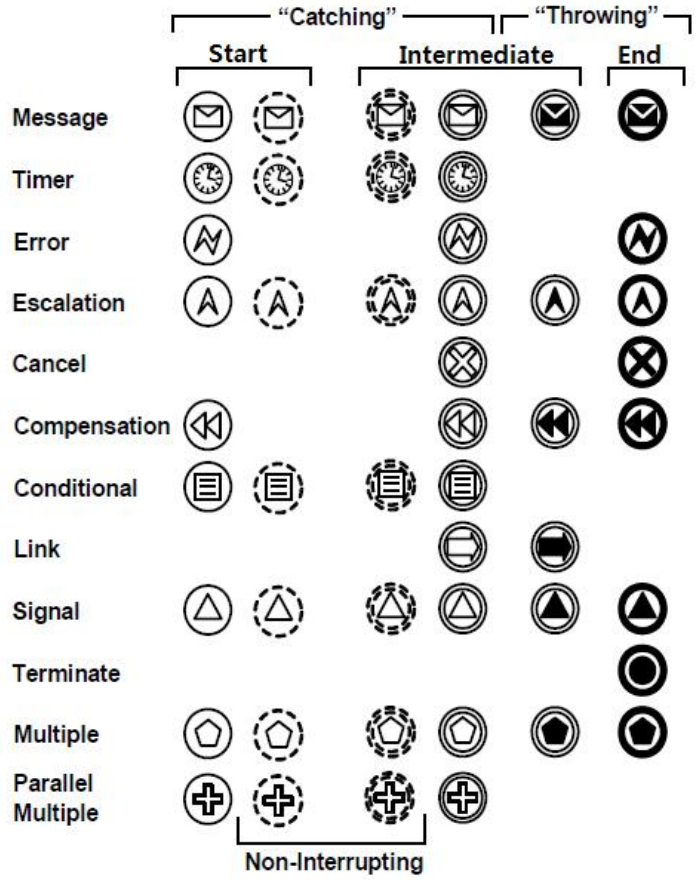

Figura 3.2: Tipos de Eventos[48]

Na dimensão horizontal da Figura 3.2, os eventos estão classificados de acordo com o momento do processo em que podem se manifestar. Os elementos

start são os eventos que podem criar uma nova instância de processo. Os elementos *intermediate* são os eventos que ocorrem durante a execução do processo. Finalmente, os elementos *end* são os eventos que indicam o fim do processo, seja por sucesso ou erro.

Na dimensão vertical da Figura 3.2, os eventos estão classificados de acordo com seu tipo, conforme descrito a seguir:

- Os eventos *message* são utilizados para indicar um ponto do processo em que ocorre comunicação com um agente externo ou outro processo.
- Os eventos *timer* são utilizados para indicar que o processo deverá parar naquele ponto e aguardar até que a condição de tempo parametrizada se torne verdadeira, como por exemplo, uma data específica ou um período de tempo.
- Os eventos *error* são utilizados para indicar que ocorreu um erro durante o fluxo de execução do processo.
- Os eventos *escalation* são utilizados para indicar que um outro participante do processo deve executar determinada ação para que o processo continue seu fluxo.
- Os eventos *cancel* indicam que o processo e suas atividades devem ser canceladas.
- Os eventos *compensation* são utilizados para desfazer ações que já haviam sido completadas e não são mais desejadas e necessitam ser revertidas.
- Os eventos *conditional* são utilizados para pausar o processo, até que uma determinada regra de negócio se torne verdadeira.
- Os eventos *link* representam uma conexão entre pontos distantes do processo, sendo bastante utilizados em processo com elevado número de atividades para facilitar a visualização do processo.
- Os eventos *signal* são utilizados para comunicação entre processos, porém, diferentemente dos eventos *message* que possuem um destinatário específico, os eventos *signal* executam o modelo *broadcast* para os receptores que dão sequência aos seus fluxos.

- Os eventos *terminate* são utilizados para finalizar completamente a execução de um processo.
- Os eventos *multiple* são utilizados para indicar diversos eventos em um único símbolo com a semântica XOR, ou seja, quando houver ocorrência de qualquer um dos eventos.
- Os eventos *parallel multiple* são utilizados para indicar diversos eventos em um único símbolo com a semântica AND, ou seja, somente quando houver ocorrência de todos os eventos.

2. Atividades

Objetos utilizados para representar uma unidade de trabalho a ser realizada no processo. Existem dois tipos básicos de atividades: tarefas ou subprocessos. As tarefas podem ser executadas por humanos ou por algum tipo de serviço, como um seviço web ou mesmo a execução de algum código interno no processo. Já os subprocessos são atividades utilizadas para agrupar outros objetos, encapsulando fluxos de maior complexidade.

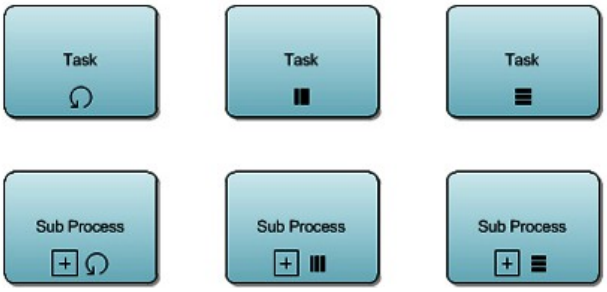

Figura 3.3: Tipos de Atividades[45]

A mesma atividade pode ser executada uma ou mais vezes, e a Figura 3.3 mostra as notações utilizadas para indicar multiplicidade. A seta circular nas atividades mais à esquerda indica que estas atividades serão executadas até determinada condição ser satisfeita. Já as linhas paralelas nas outras atividades da figura indicam execuções múltiplas, iterando em cada item de uma coleção previamente definida no processo. As linhas verticais representam que essa iteração ocorre paralelamente, ou seja, cada uma das instâncias da

atividade ocorre concomitantemente e sem ordem de execução definida. Já as linhas horizontais significam que a execução das atividades é sequencial, onde uma só será iniciada após o término da anterior.

3. Decisões

   São objetos utilizados para controlar o fluxo de trabalho, possibilitando o direcionamento do processo para a escolha de um único sentido, ou para controlar a divergência e convergência de fluxos paralelos. Essas decisões podem ser feitas com base nos dados inerentes ao fluxo do processo ou em eventos relacionados à sua execução.

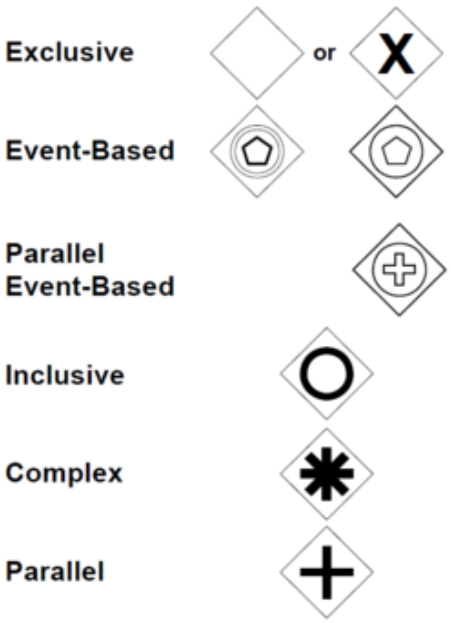

Figura 3.4: Tipos de Decisões[47]

A Figura 3.4 mostra a notação dos diferentes tipos de decisões que podem ser aplicadas ao contexto de um projeto. Elas podem ser exclusivas (apenas um fluxo é seguido, notações exibidas na primeira linha), inclusivas (um ou mais fluxos podem ser executados, notação mostrada na quarta linha), complexas (notação que prevê um comportamento específico que envolve uma lógica mais complexa, mostrada na quinta linha), paralelas (todos os fluxos são executados de maneira independente, notação exibida na última linha) e baseadas em eventos, em vez de dados do processo, as quais também podem ser exclusivas (como as notações da segunda linha) ou

paralelas (como as notações da terceira linha).

3.3.1.2 Agrupadores

São objetos utilizados para organizar visualmente a distribuição dos demais objetos do processo em contêineres que representam a responsabilidade de determinado ator do processo.

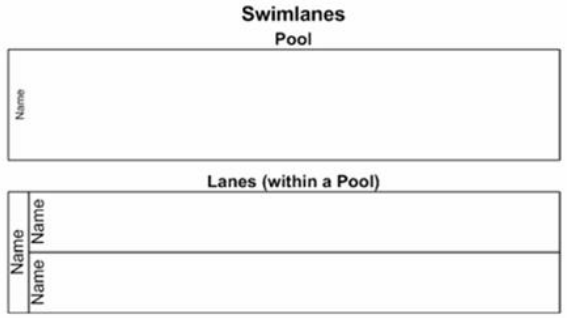

Figura 3.5: Tipos de Agrupadores[43]

A Figura 3.5 mostra os dois tipos de elementos utilizados para esta finalidade. *Lane* é usada para agrupar tarefas do mesmo usuário, ou grupo de usuários que possuem as mesmas responsabilidades dentro de um processo, já a *pool* é utilizada para agregar diferentes *lanes* do processo.

3.3.1.3 Artefatos

São utilizados para acrescentar informações adicionais à modelagem do processo.

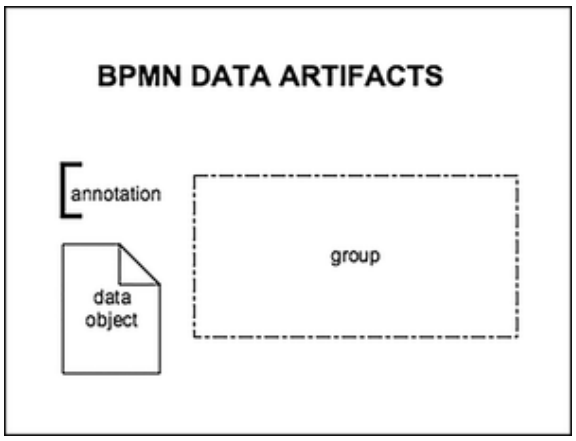

Figura 3.6: Tipos de Artefatos[44]

A Figura 3.6 exemplifica os diferentes tipos de artefatos. O primeiro são as anotações, usadas para adicionar comentários e/ou informações adicionais para facilitar o entendimento do processo, não possui nenhuma influência na execução do fluxo. O segundo é o objeto de dados (data object), usado para listar dados inerentes à execução do processo, como um arquivo de entrada, ou um XML[52] que é gerado como saída de uma etapa e utilizado em outro momento. Estes dados também podem ser variáveis, que serão usadas e manipulados pelos objetos de fluxo, por exemplo, servindo como critério para os objetos de decisão (3.3.1). O terceiro tipo de artefato é o grupo, um retângulo utilizado para englobar objetos de fluxo de forma a melhorar a organização do modelo – assim como as anotações, possui apenas caráter visual, auxiliando a compreensão do processo através de recursos gráficos.

3.3.1.4 Conectores

São utilizados para interligar objetos de fluxo. São classificados em conectores de sequência, mensagem ou associação.

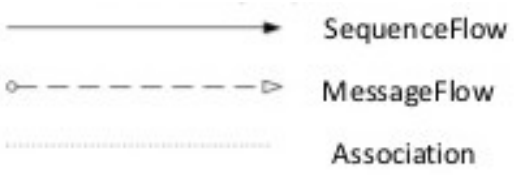

Figura 3.7: Tipos de Conectores[46]

A Figura 3.7 mostra a notação utilizada para os três diferentes tipos de conectores. O primeiro é o conector de sequência, que liga os objetos de fluxo, determinando a ordem de execução de cada um deles. O segundo é o conector de mensagem, que indica quando há troca de mensagens entre diferentes atividades, como uma chamada a um serviço *web*, por exemplo. E, por último, a associação simples, que serve para relacionar logicamente diferentes elementos do modelo, mas não altera a execução do fluxo, por exemplo, relacionando um comentário a uma tarefa.

3.4 BPMS

BPMS[14] é o acrônimo em inglês para *Business Process Management Suites/System*, ou em português *Sistemas de Gerenciamento de Processos de Negócio*. Os softwares BPMS são sistemas especialistas na modelagem, execução, controle e monitoramento de processos de negócio.

A notação BPMN é utilizada para a modelagem dos processos, e interpretada pelo motor do BPMS que a utiliza para a orquestração do fluxo de trabalho. Devido a sua abordagem focada em processos modelados por uma notação padronizada, o entendimento acerca de um processo de negócio entre analistas de negócio e desenvolvedores acaba sendo facilitado.

Dentre algumas das vantagens obtidas pelas organizações através da adoção de sistemas BPMS, podemos citar melhorias na capacidade de gestão e monitoramento de processos, melhorias na visibilidade e entendimento de processos, e maior rapidez na introdução de mudanças e ajustes nos processos de negócio.

Dentre alguns aspectos que caracterizam os sistemas BPMS, podemos citar a flexibilidade e facilidade na construção de fluxos de trabalho complexos através de modelagem gráfica, a possibilidade de integração com outros sistemas de informação, os recursos para gestão dos processos e a interface do usuário com as atividades do processo.

Ao inspecionarmos o ecossistema de aplicações BPMS, podemos encontrar diversas opções variando desde soluções caríssimas até soluções gratuitas e de código-

fonte aberto. Como exemplo de soluções pagas, podemos citar o SAP NetWeaver BPM[41] e Oracle BPM[36]. Como exemplo de soluções com versões gratuitas ou de código-fonte aberto, podemos citar o Activiti BPM[1], Bonita BPM[9] e Intalio BPM[25].

A maioria das aplicações BPMS mais robustas oferecem algum tipo de portal para o usuário. Neste portal, o usuário tem a possibilidade de interagir com os processos, atuar em tarefas a ele designadas, iniciar ou finalizar processos, entre outras atividades relacionadas a gestão dos processos. Além do portal, também costumam oferecer algum tipo de API, o que possibilita a utilização de diferentes interfaces com o usuário e integração com outros sistemas. A disponibilidade de uma API permite a extensão e integração do BPMS para outros sistemas, como por exemplo ao permitir a execução de tarefas através de aplicativos em *smartphones*, como na criação de um portal totalmente customizado para atender às necessidades de uma corporação, ou até mesmo na possibilidade de embutir o motor de processos em um sistema para facilitar a orquestração de processos.

Apesar das vantagens e diversos recursos oferecidos por suites BPM, os altos custos deste tipo de solução, além da complexidade para sua implantação e customização de interfaces para o atendimento de diferentes processos, acabam por inviabilizar sua aplicação em projetos corporativos. Neste sentido, buscamos soluções intermediárias, que ofereçam recursos simples e intuitivos para a criação de processos menos complexos, mas que também sejam capazes de escalar para processos mais complexos quando necessário. Além desses aspectos, seria interessante que esta solução não necessitasse da compra de licenças para sua utilização, uma vez que a necessidade de *hardware* e equipe especializada para mapeamento e automatização de processos já incluem custos consideráveis no orçamento de projetos deste tipo.

No próximo capítulo, apresentaremos uma solução desenvolvida neste projeto para atender os requisitos apresentados anteriormente. Um sistema para automatização e gestão de processos, fácil e intuitivo para o usuário, com uma pequena curva de tempo na implantação de processos simples, mas igualmente capaz de suportar processos complexos quando necessário, sem a necessidade de licenças por utilizar

softwares gratuitos, com uma interface amigável, geração de formulários automáticos, além de uma gestão no controle de acesso a tarefas e processos.

Capítulo 4

Integração Redmine e Activiti BPM

4.1 Introdução

Considerando as dificuldades que muitas empresas enfrentam na sua gestão surge a necessidade de implementação de uma boa gestão de processos, através da metodologia de BPM para melhorar a eficiência, reduzir retrabalho, tarefas manuais repetitivas, e custo.

Em muitos destes casos se faz necessário aplicar tecnologia para automatizar processos e auxiliar na condução e padronização dos processos, bem como permitir que robôs passem a realizar algumas tarefas mecânicas e repetitivas antes executadas por pessoas.

A solução a ser proposta neste capítulo tem por objetivo disponibilizar uma ferramenta para automatização de processos que possibilite a execução de processos simples e processos complexos. Esta ferramenta permitirá a condução de processos, incluindo a interação dos atores com o processo e a gestão e acompanhamento pelos gerentes e responsáveis pelo processo. Esta ferramenta deverá envolver apenas tecnologias de código aberto, bem como fornecer uma opção de interface amigável ao usuário e uma implementação fácil e rápida, assim como fácil manutenção.

No capítulo 2 mostramos como utilizamos o Redmine para gerenciar processos. Esta estratégia para condução de processos simples foi um sucesso. No entanto,

encontramos dificuldades para automatizar processos mais complexos utilizando este sistema. Isto seria possível apenas com o desenvolvimento de plugins ou mudanças no código-fonte principal, ambas as opções custosas, arriscadas e imanuteníveis a longo prazo. Além disso, as funcionalidades que teríamos que desenvolver no Redmine para possibilitar a condução de processos mais complexos já existem nos BPMS, criados exatamente para este tipo de demanda.

Utilizar apenas um BPMS contudo não se mostrou a melhor opção, pois perderíamos o excelente controle de permissões do Redmine, bem como a geração automática de formulários e a facilidade de criar um processo simples. Neste capítulo será apresentada a solução para os problemas citados acima: a integração entre o Redmine e o Activiti BPM, o BPMS escolhido para esta implementação.

4.2 Motivos para integração

A utilização do Redmine como gerenciador de processos torna-se limitada, como foi descrito no Capítulo 2, já que ele não possui efetivamente um motor de processos BPM. Ainda assim, a possibilidade de extensão e desenvolvimento de novas funcionalidades no Redmine através de plugins sugere que o mesmo possa ser aperfeiçoado para o atendimento de processos mais complexos, ao mesmo tempo em que suas vantagens de portal e atendimento de processos mais simples são mantidos.

Com o objetivo de estender o Redmine para atender fluxos de trabalho mais complexos, torna-se necessário o desenvolvimento de um motor de processos mais robusto que possibilite novas opções e configurações de fluxos de trabalho na ferramenta. Entretanto, uma solução deste porte acaba por convergir em problemas solucionados por ferramentas BPMS, que permitem a modelagem e execução de processos complexos.

A utilização de BPMS para a automatização de processos de negócio traz diversas vantagens, conforme apresentado na seção 3.4. Entre elas, utilizar um sistema que implementa as funcionalidades previstas na notação BPMN, que permitem a condução de processos complexos, e por isto reduz a necessidade do desenvolvimento

de funcionalidades específicas.

Boa parte das aplicações BPMS mais robustas oferecem algum tipo de portal, onde ocorrem basicamente as interações com as tarefas humanas, e também APIs para manipulação dos processos, o que permite a integração com diferentes interfaces e aplicações. Entretanto, nem sempre esses portais possuem uma interface tão amigável para o usuário, ou são adaptados a dispositivos móveis como *smartphones* ou *tablets*, ou são de fácil customização. Essas são características positivas encontradas no Redmine, que conta com mais de 50 temas prontos para uso na lista oficial[40], além de possuir uma estrutura bem organizada para customização e criação de novos *layouts*.

O Redmine também possui um excelente suporte à geração automática de formulários pela associação de campos customizados tipados às tarefas, como por exemplo datas, caixas de seleção e texto. Comparativamente, a implantação de processos mais simples pelo Redmine acaba sendo mais rápida e mais fácil de ser mantida do que em ferramentas BPMS, já que não é necessário nenhuma modelagem em notação específica, instalação de processos ou desenvolvimento de formulários; tudo é feito facilmente pela interface web.

Neste sentido, torna-se interessante propor uma solução que aproveite todas as vantagens de portal e implantação de processos de negócio mais simples do Redmine, mas que também ofereça todo o poder e flexibilidade de modelagem de processos mais complexos oferecidos por ferramentas BPMS. Surge assim a proposta de uma solução integrada para a automatização de processos de negócio, através da junção do Redmine com uma ferramenta BPMS.

4.3 Integração genérica

Pensando na possibilidade de utilizar qualquer plataforma BPMS integrada ao Redmine, decidimos avaliar a utilização de ferramentas ESB[53] de forma a facilitar a construção de uma interface única de comunicação com o Redmine. Conforme definido em publicação do O'Reilly[26], ESB é uma plataforma de integração baseada

em padrões que combina mensagens, serviços web, transformação de dados e roteamento inteligente para conectar de forma confiável e coordenar a interação de um número significativo de aplicações e serviços. Assim, bastaria o trabalho de integrar o BPMS escolhido com este barramento, e assim o Redmine estaria capacitado a orquestrar fluxos de trabalho mais complexos utilizando qualquer BPMS de mercado. A Figura 4.1 apresenta o modelo de arquitetura proposto para a integração genérica do Redmine com BPMS.

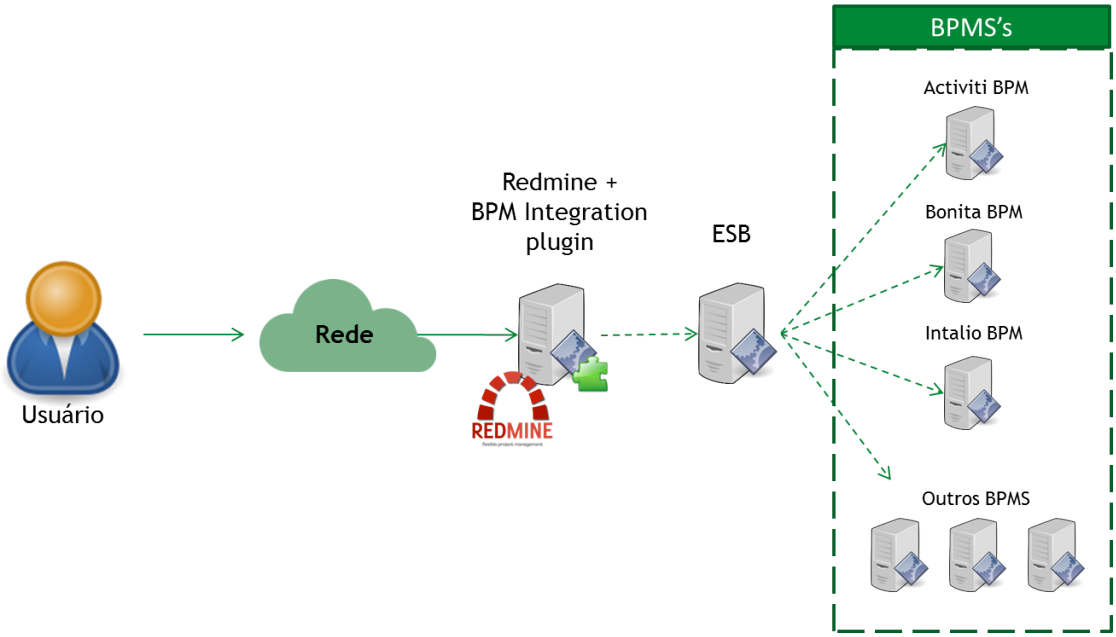

Figura 4.1: Arquitetura proposta de integração genérica entre Redmine e BPMS's

Iniciamos o desenvolvimento deste barramento através da escolha de um ESB e pelo menos dois BPMS para validar a construção desta interface. Para o ESB, optamos pelo Mule ESB[32] que possui uma versão gratuita e está posicionado como uma das melhores ferramentas do seguimento de integração empresarial, segundo o quadrante mágico anual da Gartner[62]. Para o BPMS, optamos por duas ferramentas com versões gratuitas, desenvolvidas na linguaguem Java e com uma boa interface REST[8]: Activiti BPM[1] e Bonita BPM[9].

Possuir uma boa API REST[8] é um ponto importante, pois é uma interface simples e muito usada, o que facilita utilizar a mesma estrutura para comunicação

do ESB com os BPMS, além de ampliar o número de BPMS que poderíamos incluir posteriormente, pois muitos deles possuem uma interface REST.

O primeiro serviço escolhido para modelagem no barramento foi o responsável por disparar um novo processo no BPMS. Consistiria basicamente na disponibilização de um serviço REST, com parâmetros comuns às APIs de ambos os BPMS para que um novo processo fosse iniciado na plataforma escolhida. Apesar de um esforço considerável para a construção deste primeiro serviço, o mesmo foi desenvolvido com sucesso.

Entretanto, este piloto foi suficiente para identificarmos logo no início que haveria um grande esforço para a construção deste barramento. Para implementar o serviço citado acima, foi necessário estudar a fundo os serviços implementados pela interface REST de cada BPMS e percebemos que haviam muitas diferenças entre eles. Devido a estas diferenças percebemos que não seria possível construir uma comunicação genérica entre o ESB e os BPMS, e exigiria uma alta complexidade. Percebemos fundamentalmente que esta solução exigiria um altíssimo esforço de desenvolvimento e modelagem, e que nos afastaríamos do objetivo central deste trabalho, que é prover um motor de processos mais complexo para o Redmine. Além disso, se tivéssemos que desenvolver uma estrutura nova de comunicação para cada BPMS, a camada de interface do ESB deixava de fazer sentido.

Em razão disso optamos por uma solução de menor complexidade, mas que provesse o ganho almejado com a integração do BPMS ao Redmine.

4.4 Integração Redmine e Activiti BPM

Dada a complexidade de criação de uma interface genérica de integração do Redmine com todos os BPMS, decidimos implementar a integração do Redmine com um BPMS específico de mercado.

Para a escolha do BPMS de mercado, estabelecemos alguns critérios. O motor deveria ser gratuito e de código aberto, estando assim alinhado com a proposta open-source do Redmine. Além disso, seria fundamental que a ferramenta fosse leve,

extensível e com uma API de fácil utilização e boa documentação, possibilitando assim um rápido entendimento da plataforma e abertura para qualquer modificação ou extensão necessária na ferramenta. Seria desejável também que a ferramenta estivesse em constante evolução e fosse suportada por uma comunidade ativa e que pudesse oferecer suporte para dúvidas encontradas durante o processo de integração.

Ao avaliarmos as possibilidades existentes, identificamos uma ferramenta que se encaixava bastante nos critérios adotados, o Activiti BPM. O Activiti BPM é uma ferramenta de código aberto, compartilhado no GitHub[18], possui extensa documentação[5] e dispobiliza uma API REST, arquitetura oficialmente adotada pelo *framework* Ruby on Rails desde sua versão 2.0, lançada em dezembro de 2007[54], facilitando a comunicação com o Redmine, que é escrito nesta linguagem, além das razões citadas na seção 4.3.

4.4.1 Activiti BPM

Criado em 2010 por ex-integrantes do projeto jBPM[28], o Activiti BPM[1] é um projeto de código aberto sob a licença Apache[6], que proporciona um motor BPM leve, estável e de fácil integração. O Activiti BPM é desenvolvido na linguagem de programação Java[56] e é facilmente integrável com aplicações existentes por sua leveza e API[7] amigável, além de utilizar a notação BPMN 2.0 para a modelagem dos processos.

Nesta seção, vamos apresentar os componentes do Activiti BPM, os requisitos de software para seu funcionamento e informações sobre o ambiente de execução e modelagem do processo.

4.4.1.1 Componentes

O Activiti BPM é constituído por diversos componentes que possibilitam o aproveitamento do ecossistema BPM. Com exceção do *Activiti Engine*, que constitui o módulo principal da ferramenta, todos os demais módulos são opcionais e dependem do cenário de utilização do BPMS.

A Figura 4.2 apresenta os componentes do Activiti BPM: *Activiti Modeler*, *Activiti Designer*, *Activiti Kickstart*, *Activiti Engine*, *Activiti Explorer* e *Activiti REST*. Os componentes são classificados nas categorias *Modeling* (Modelagem), *Runtime* (Tempo de Execução) e *Management* (Gerenciamento).

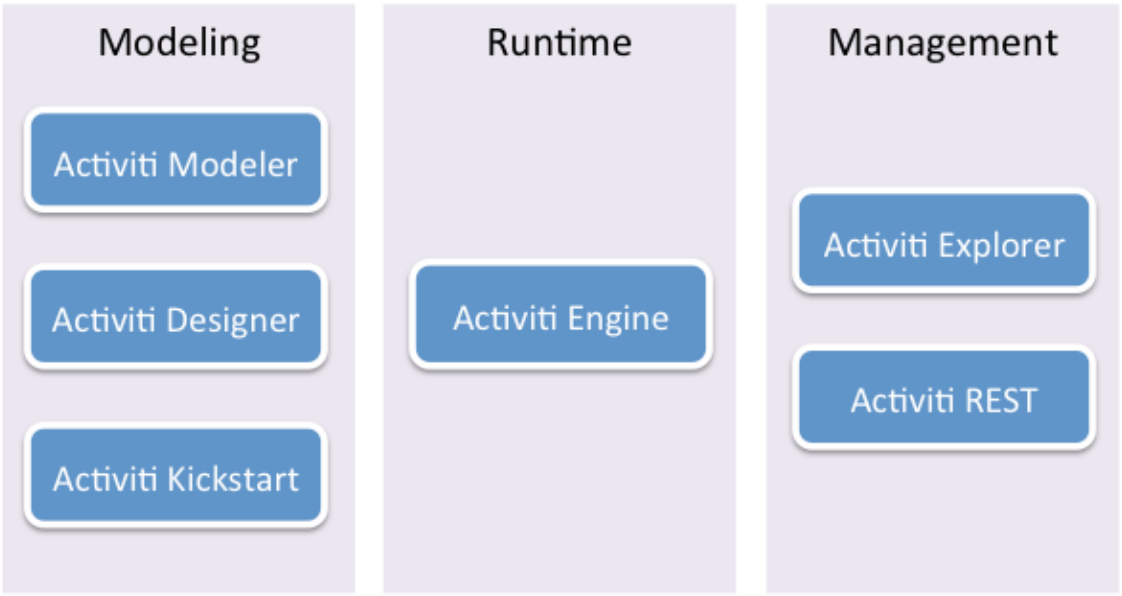

Figura 4.2: Componentes do Activiti BPM[15]

4.4.1.1.1 Activiti Modeler

O *Activiti Modeler* possibilita a modelagem de processos BPMN 2.0 utilizando o browser. O Activiti Modeler não está mais em desenvolvimento ativo pela equipe principal, mas permanece disponível como parte da aplicação web Activiti Explorer.

4.4.1.1.2 Activiti Designer

O *Activiti Designer* é um plugin criado para a interface de desenvolvimento Eclipse. Ele permite ao analista ou desenvolvedor modelar processos na notação BPMN, e gera automaticamente a notação do processo em XML. A Figura 4.3 mostra a IDE[24] *Eclipse* já com o plugin *Activiti Designer* instalado para a modelagem do processo.

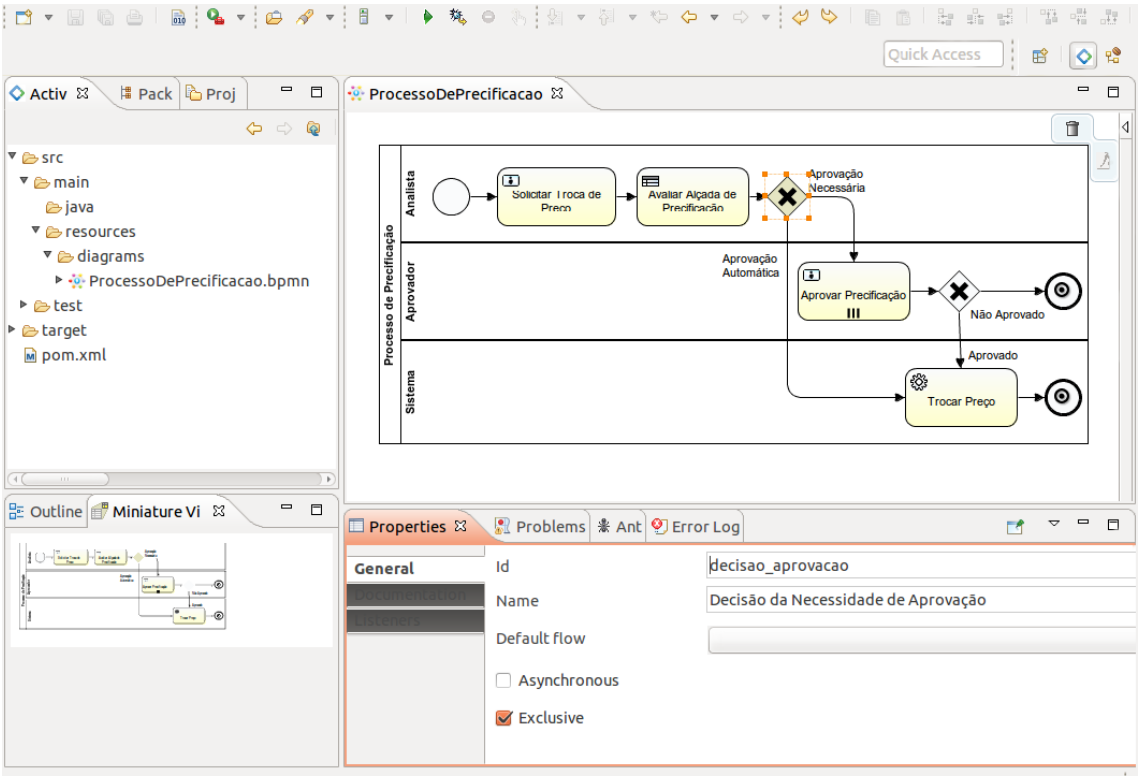

Figura 4.3: Activiti Designer

4.4.1.1.3 Activiti Kickstart

O *Activiti Kickstart* foi criado para facilitar a criação de processos *adhoc* (sub-processos cujas tarefas podem ser executadas em qualquer ordem, várias vezes, ou mesmo puladas) através de uma interface web simples e intuitiva, sem que o usuário necessite conhecer BPMN, já que são utilizados tabelas e formulários simples para a definição do fluxo de trabalho. Seu objetivo é oferecer um meio rápido para construção de processos mais simples, reduzindo assim o custo e tempo investido na automatização desses processos.

4.4.1.1.4 Activiti Engine

O *Activiti Engine* é o componente principal do Activiti BPM pois nele estão presentes o motor de funcionamento do BPM e as APIs para acesso e controle da ferramenta. Este componente é disponibilizado através de um simples arquivo

JAR[27] (modelo de arquivo padrão para bibliotecas da linguagem Java). Sendo assim, o motor BPM pode ser facilmente utilizado em diferentes projetos Java através da inclusão dessa biblioteca como dependência.

4.4.1.1.5 Activiti Explorer

O *Activiti Explorer* é a interface web padrão com o usuário, disponibilizada para o controle e execução de processos. Através dessa interface, o usuário pode instalar novas definições de processos, iniciar ou cancelar processos, finalizar ou delegar tarefas. A Figura 4.4 mostra a janela do Activiti Explorer.

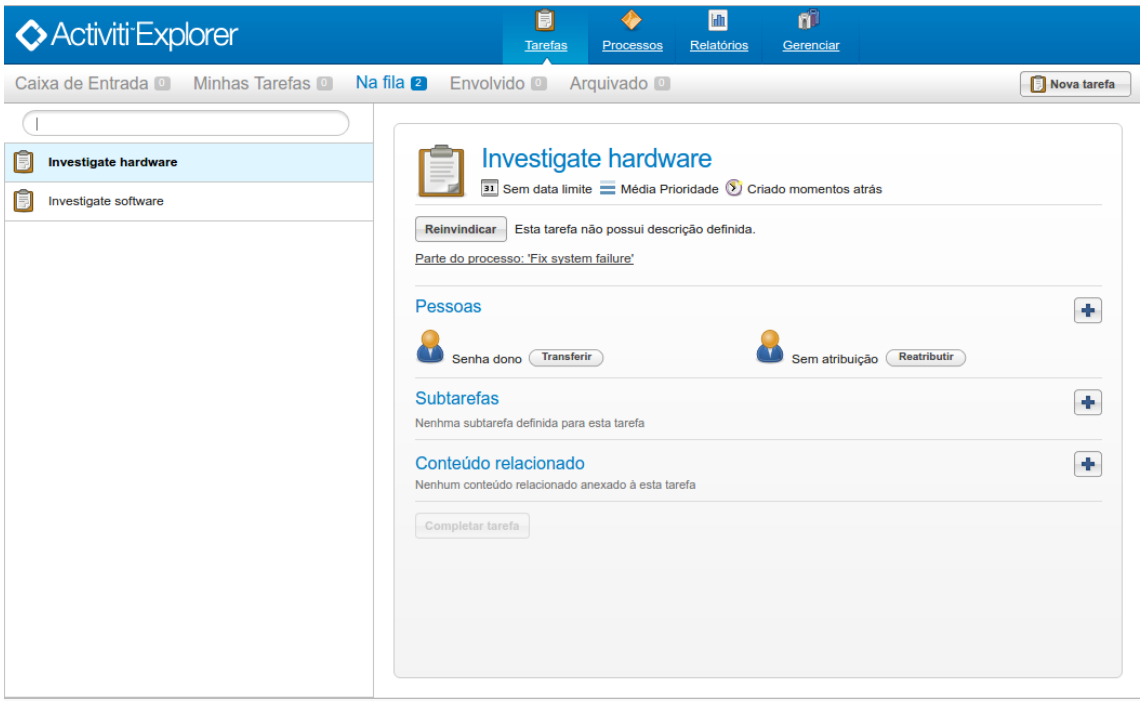

Figura 4.4: Activiti Explorer

4.4.1.1.6 Activiti REST

O *Activiti REST* disponibiliza o acesso às funcionalidades principais do Activiti através de uma interface REST. Isso simplifica o acesso às suas funcionalidades por diferentes linguagens de programação e facilita a integração com diferentes aplicações de interface com o usuário, como por exemplo aplicações para dispositivos móveis.

4.4.1.2 Requisitos de Software

O Activiti BPM requer um ambiente Java JDK versão 6 ou superior para seu funcionamento. Além do ambiente Java configurado, também é necessário um banco de dados para suportar a persistência das informações dos processos. Uma variedade de bancos é suportada pelo BPMS, são eles: H2[21], MySQL[33], Oracle[35], PostgreSQL[37], DB2[23] e SQL Server[42].

4.4.1.3 Ambiente de Execução

Existem basicamente duas possibilidades principais para configurar o ambiente de execução do Activiti BPM. A primeira opção é a inclusão da dependência do Activiti BPM em um projeto Java web, e a instalação do pacote WAR[51] para execução em um servidor Java web. Outra opção, mais simples, é a utilização do pacote WAR já pronto para funcionamento, disponibilizado diretamente no site da ferramenta[3].

O arquivo disponibilizado no portal da ferramenta já inclui o *Activiti Engine* e o *Activiti Explorer* num único pacote pré-configurados com um banco de dados em memória H2[22] que dá suporte ao seu funcionamento e não requer nenhuma configuração adicional, já que é um banco de dados *in-memory*[22]. Para iniciar a execução do BPMS, é necessária a instalação do artefato em um servidor Java como o Apache Tomcat 7[49].

Após a instalação do artefato e inicialização do Tomcat, o portal pode ser acessado através do endereço *http://127.0.0.1:8080/activiti-explorer*, dado que o Tomcat tenha sido instalado na sua configuração padrão. A Figura 4.5 demonstra a tela de autenticação do *Activiti Explorer*.

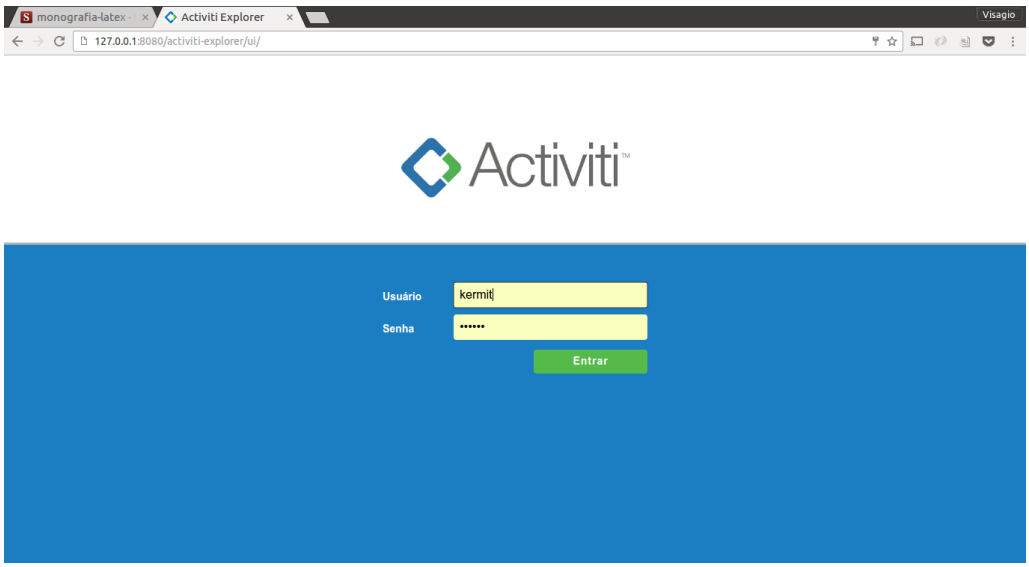

Figura 4.5: Login do Activiti Explorer

4.4.1.4 Modelagem do processo

Para a modelagem do processo, é necessária a instalação do plugin *Activiti Designer* na ferramenta de desenvolvimento *Eclipse*[17]. Mais detalhes sobre como proceder com a instalação do plugin estão disponíveis no guia de instalação disponível no site[2] do Activiti BPM.

Ao criar um novo diagrama através da opção de criar um novo *Activiti Diagram*, o *Activiti Designer* cria um arquivo *.bpmn* e disponibiliza duas visões para a modelagem do processo. Uma aba visual, contendo o desenho do processo, e outra aba textual, com a notação em XML. Ambas as abas estão interligadas e podem ser utilizadas para a modelagem, sendo a visual mais simples para o desenvolvimento do processo.

Para a modelagem do processo na paleta visual, é necessário arrastar os componentes visuais dispostos na aba lateral a fim de obter a modelagem desejada do processo na notação BPMN.

Ao final da modelagem, o arquivo *.bpmn* é gerado para ser utilizado na instalação do processo no motor de processos. A Figura 4.6 representa a modelagem do processo ilustrado na introdução deste trabalho utilizando o *Activiti Designer* e já descrito

na notação BPMN 2.0. O código gerado em XML pode ser consultado no apêndice A.1.

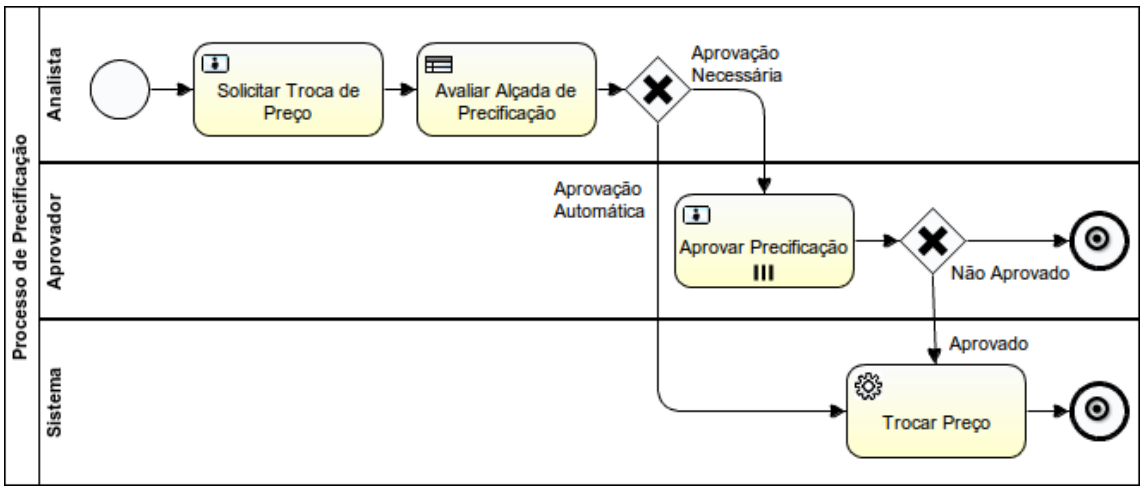

Figura 4.6: Processo modelado no Activiti BPM

4.5 Customização do Redmine

Para atingir nosso objetivo de integração do Redmine com o Activiti BPM, criamos um plugin para o Redmine que possibilita a comunicação com o Activiti BPM. Chamamos este plugin de BPM Integration. Nas próximas sessões, descreveremos o processo de construção deste plugin e como utilizá-lo. Neste trabalho utilizamos a versão 3.1 do Redmine[39].

4.5.1 Arquitetura da solução

A integração proposta tem por objetivo centralizar o máximo de funcionalidades no Redmine, deixando transparente tanto para o usuário comum como o gestor a existência de um motor BPM por trás dele.

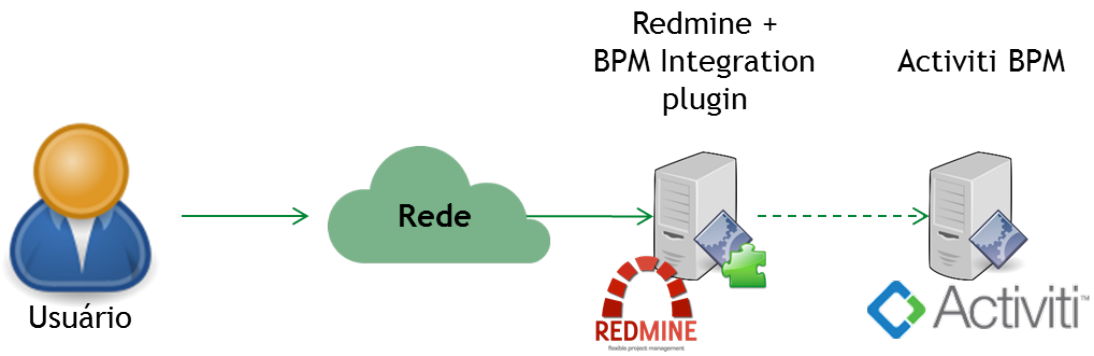

Figura 4.7: Arquitetura da integração entre Redmine e Activiti

A Figura 4.7 ilustra a arquitetura da solução de integração proposta. O usuário continua a interagir diretamente com o Redmine da mesma maneira que fazia antes da criação do *plugin*. O Redmine, então, se comunica com o Actitivi pela sua API disponibilizada através de um Web Service REST.

A abordagem utilizada visou concentrar todo o esforço de integração do lado do Redmine, sem a necessidade de customização do Activiti BPM, tornando transparente para ele, também, que o Redmine está sendo usado como sua interface com os usuários finais.

4.5.2 Linguagens

O plugin desenvolvido para o Redmine foi implementado em Ruby[61], utilizando a framework Rails[57]. As modificações realizadas no Activiti BPM foram implementadas em Java, a linguagem em que este foi desenvolvido.

4.5.3 Banco de dados

O banco de dados utilizado neste trabalho foi o MySQL 5.7 por ser o banco de dados mais utilizado e testado com o Redmine pela comunidade.

4.5.4 Comunicação entre o Redmine e Activiti

Para possibilitar a integração entre o Activiti BPM e o Redmine, foi necessário construir no Redmine uma estrutura de dados que represente o modelo de dados do BPMS. Também foi preciso garantir a atualização das informações relativas ao andamento dos fluxos dos processos de forma que eles possam ser desempenhados corretamente através do Redmine. Assim, fez-se necessário o desenvolvimento de um mecanismo de sincronização entre as duas aplicações.

Toda a comunicação entre as duas ferramentas foi desenhada de forma que o Redmine funcionasse como uma interface do Activiti BPM. Portanto, o Redmine consome a API REST do BPMS para disparar ações ou ler informações do mesmo.

No restante desta seção apresentamos os detalhes da integração proposta: o tópico 4.5.4.1 descreve a utilização do Redmine para a criação de novas definições de processos no Activiti BPM e como os novos dados são mapeados de volta para o Redmine; os tópicos 4.5.4.2 e 4.5.4.4 mostram, respectivamente, como o Redmine é utilizado para iniciar novas instâncias de processos (relativos às definições previamente configuradas) e finalizar as tarefas em andamento. Por fim; os tópicos 4.5.4.3 e 4.5.4.5 explicam como é feita a sincronização dos dados presentes no Activiti BPM para o Redmine, criando as novas tarefas e atualizando o *status* dos processos até serem concluídos no BPMS.

4.5.4.1 Definição de processo

O primeiro passo necessário para disponibilizar um processo a ser conduzido no Activiti BPM é desenhar a modelagem do processo no padrão BPMN 2.0. Isto é feito através do plugin Activiti Designer a ser instalado no Eclipse, como explicado na seção 4.4.1.4.

Para efetivamente disponibilizar este processo para ser iniciado, é necessário fazer a instalação do arquivo .bpmn contendo a modelagem do processo. Quando utiliza-se apenas o Activiti BPM para conduzir processos, é comum usar o Activiti Explorer (4.4.1.1.5) para instalar fazer esta instalação.

Portanto, para o Redmine assumir a função de interface principal do Activiti BPM foi desenvolvida uma funcionalidade do Activiti Explorer que permite ao usuário fazer a instalação de um processo através diretamente através do Redmine, uma ação que envia o arquivo da modelagem BPMN e disponibiliza o processo para ser iniciado. Esta funcionalidade dispara um serviço que acessa a API REST do Activiti BPM, executando uma requisição POST que efetivamente realiza a instalação, adicionando a modelagem do processo em questão à lista de definições de processos ativos que podem ser iniciados. Após executada a requisição POST, é iniciada uma rotina que busca a definição do processo recém criado. As informações do processo que são recuperadas consistem das tarefas humanas definidas, campos de formulário, variáveis de processo e outros detalhes. Esta ação é realizada pelo usuário administrador da ferramenta, e deve ser realizada apenas uma vez para cada tipo de processo. A Figura 4.8 ilustra a sequência de ações descritas acima.

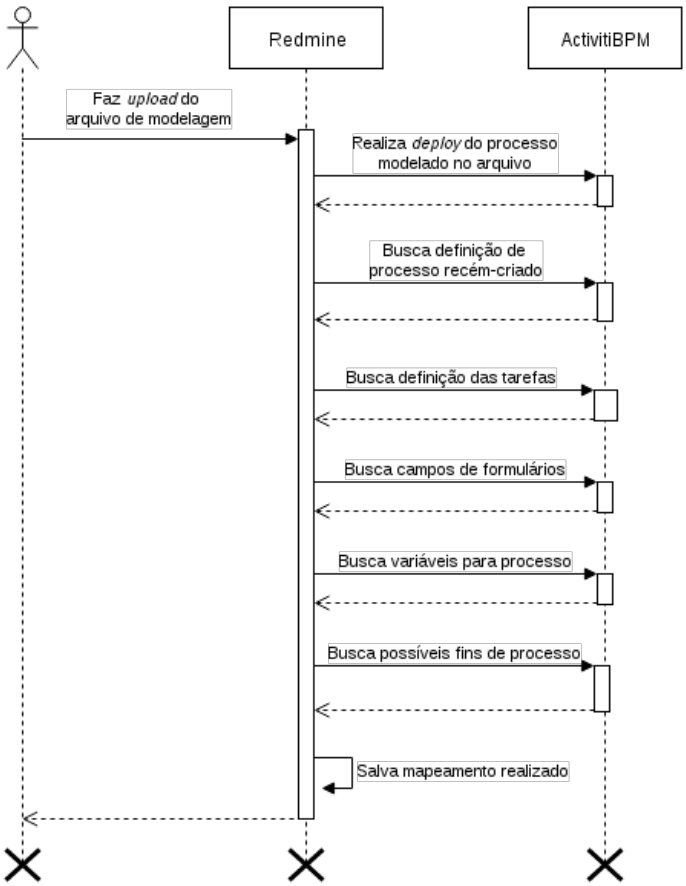

Figura 4.8: Diagrama de sequência de instalação (deploy) de processo

Na Figura 4.9 é possível observar a lista de definições de processo apresentada no Redmine. Cada linha desta tabela representa um processo diferente que pode ser iniciado no Activiti BPM, resultando na criação de uma tarefa no Redmine associada ao respectivo tipo de tarefa configurado para o processo.

Figura 4.9: Lista de definições de processo disponíveis no Activiti BPM exibida na tela do plugin do Redmine

4.5.4.2 Inicialização de um processo

Sempre que uma tarefa vinculada a um processo BPM é criada (configuração provida pelo plugin), é disparado um procedimento de sincronização ilustrado na Figura 4.10. Esta vinculação é feita através da uma configuração na tela que pode ser acessada ao clicar em cada definição de processo da lista na tela exibida na Figura 4.9. Este procedimento executa uma requisição à API REST do Activiti BPM que inicia uma nova instância de um processo no mesmo. A requisição carrega as informações da tarefa recém-criada, inclusive os valores dos campos customizados que foram mapeados para algum campo de formulário da definição de processo do BPMS. O status da tarefa do Redmine é, então, atualizado a partir do retorno do serviço do Activiti e é disparado o procedimento de sincronização de tarefas descrito na seção 4.5.4.3. Esta ação é realizada por um usuário comum da ferramenta, que necessita iniciar um determinado processo.

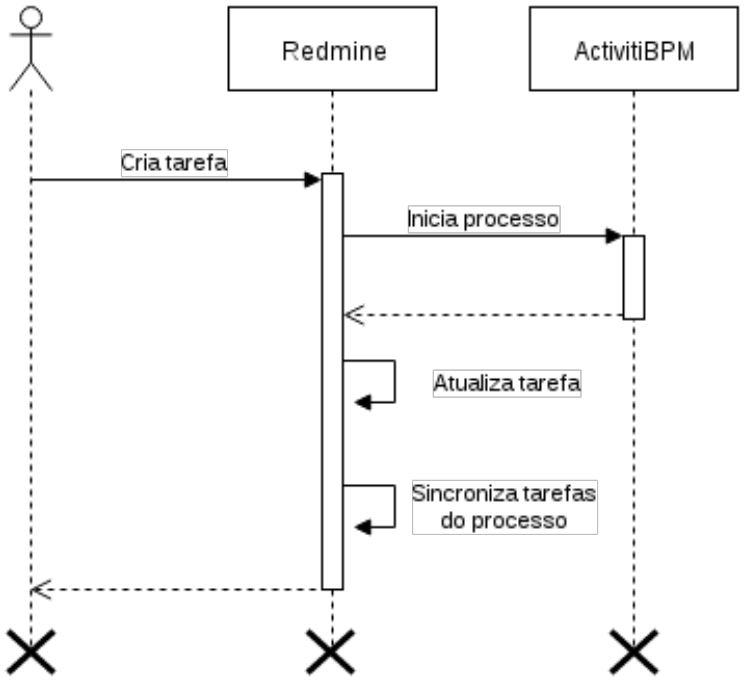

Figura 4.10: Diagrama de sequências ilustrando criação de processo no ActivitiBPM a partir do Redmine

A Figura 4.11 mostra a tela de criação de tarefas no Redmine responsável por disparar um novo processo no Activiti BPM do tipo "Emissão cartão suprimentos"usado como exemplo.

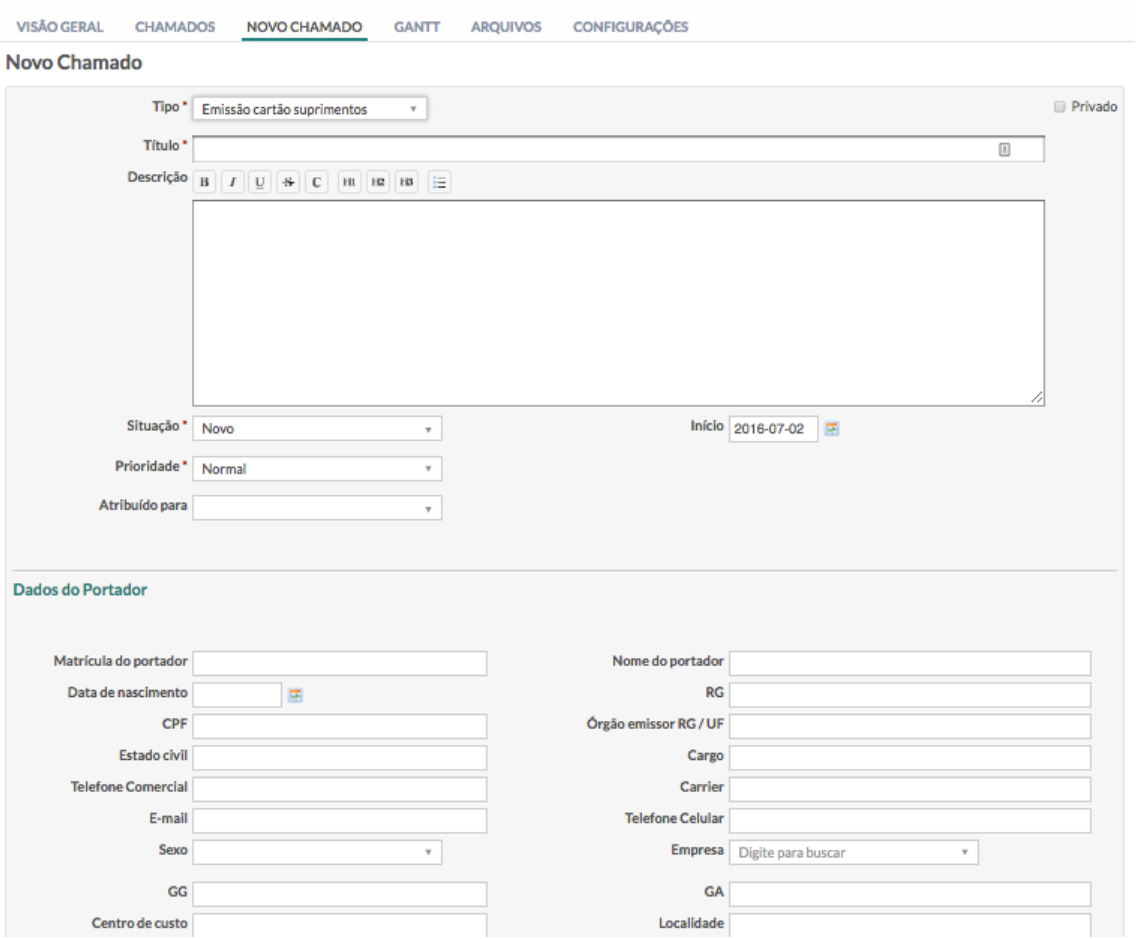

Figura 4.11: Tela de criação de tarefas no Redmine

Os valores preenchidos para os campos padrão do Redmine são gravados no processo iniciado no Activiti BPM, bem como os campos personalizados que são configurados. Foi desenvolvida uma funcionalidade que permite ao usuário mapear cada campo personalizado para um campo presente no modelo do processo. A Figura 4.12 mostra a tela onde esta configuração é feita. Nesta mesma tela são configurados variáveis do processo, estados de conclusão do processo e mensagens de conclusão e estados iniciais das tarefas do processo no Redmine.

Os campos citados acima são declaradas no XML de modelagem do processo na notação BPMN. O plugin BPM Integration lê estas variáveis através da API REST do Activiti BPM e monta esta tela de configuração. As variáveis de processo são dados constantes utilizados ao longo do processo que são inicializadas com o valor

preenchido nesta tela. O tipo de valor a ser preenchido para cada váriável depende de sua declaração, podendo ser uma situação (2.2.5), um usuário, um grupo ou um campo personalizado (2.2.6) cadastrado no Redmine ou um texto livre. Na configuração dos estados de conclusão o usuário deve preencher a situação correspondende a cada evento (3.3.1.1) de conclusão previsto no processo. Sempre que um processo é finalizado no Activiti BPM, o Redmine identifica e transiciona a situação para a situação configurada nesta tela. Também é possível configurar uma mensagem de conclusão padrão, que será preenchida como nota na tarefa do Redmine junto à mudança de situação. Em "Status inicial das tarefas do processo"o usuário configura a situação inicial para cada sub-tarefa que é criada durante o processo. Na seção 4.5.4.3 explicaremos como as sub-tarefas são usadas para representar etapas do processo.

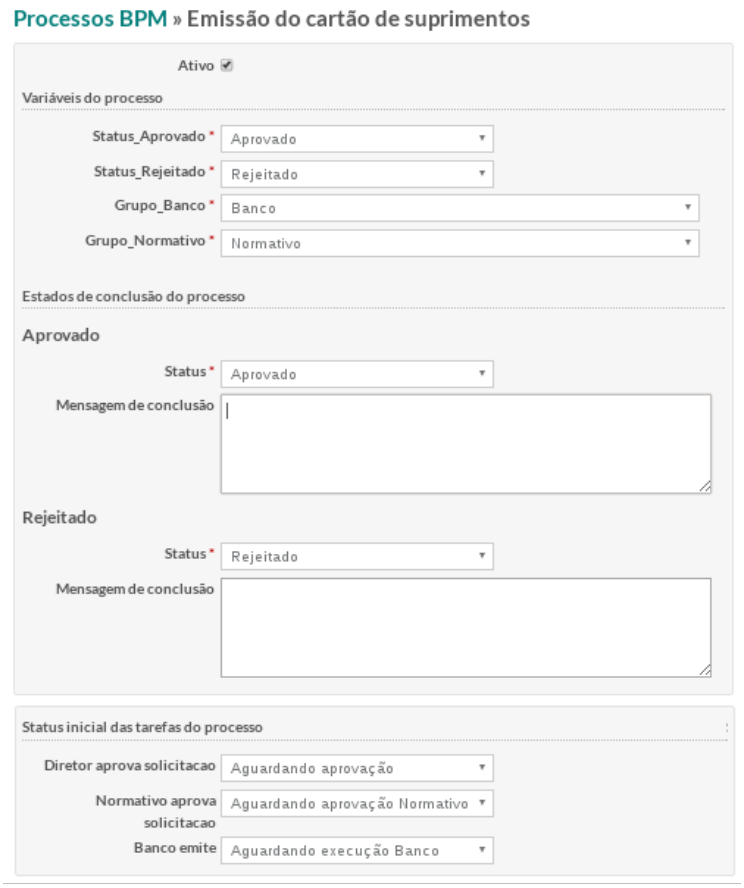

Figura 4.12: Tela de configurações do processo - Variáveis

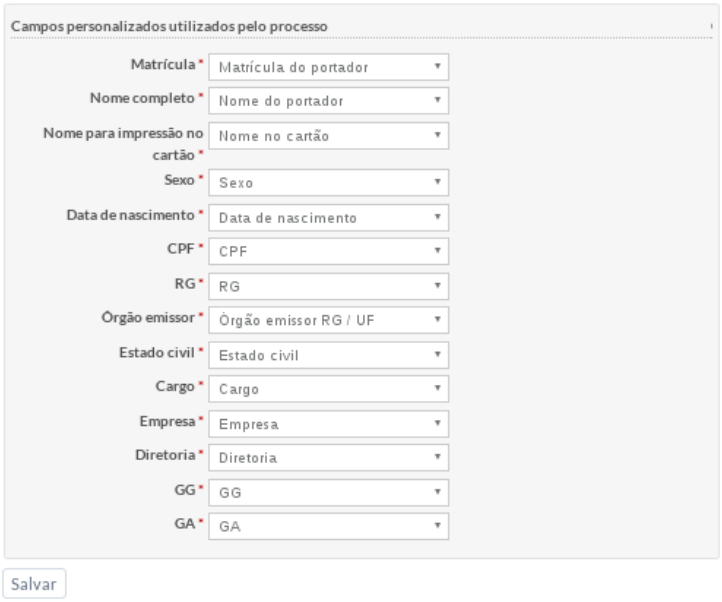

Figura 4.13: Tela de configurações do processo - Campos personalizados

4.5.4.3 Tarefas humanas

Chamaremos as tarefas que são realizadas por humanos, citadas na seção 3.3.1.1, tarefas humanas. Quando uma atividade deste tipo está presente na modelagem de um processo, significa que a etapa em questão deve ser realizada por uma pessoa. A interação do responsável pela tarefa com o processo é um dos pontos fortes do Redmine. Utilizando o plugin BPM Integration, as tarefas humanas do Activiti BPM são representadas no Redmine por sub-tarefas das tarefas que representam processos. Uma sub-tarefa nada mais é que uma hierarquia de tarefas, como mencionado na seção 2.2.2. Quando uma tarefa humana é criada no decorrer do processo, o Redmine precisa refletir isto, de modo que esta tarefa possa ser executada pelo ator correto e o processo siga seu curso. O espelhamento desta tarefa no Redmine é totalmente automatizado através de uma rotina do plugin BPM Integration que é executada periodicamente e verifica se existem tarefas humanas no Activiti BPM que não foram espelhadas no Redmine, e cria uma nova sub-tarefa. Os valores dos campos do processo de uma tarefa são populados nos campos padrão e campos personalizados conforme configurados na tela exibida na Figura 4.13. Esta rotina também é executada quando um novo processo é iniciado ou uma tarefa finalizada.

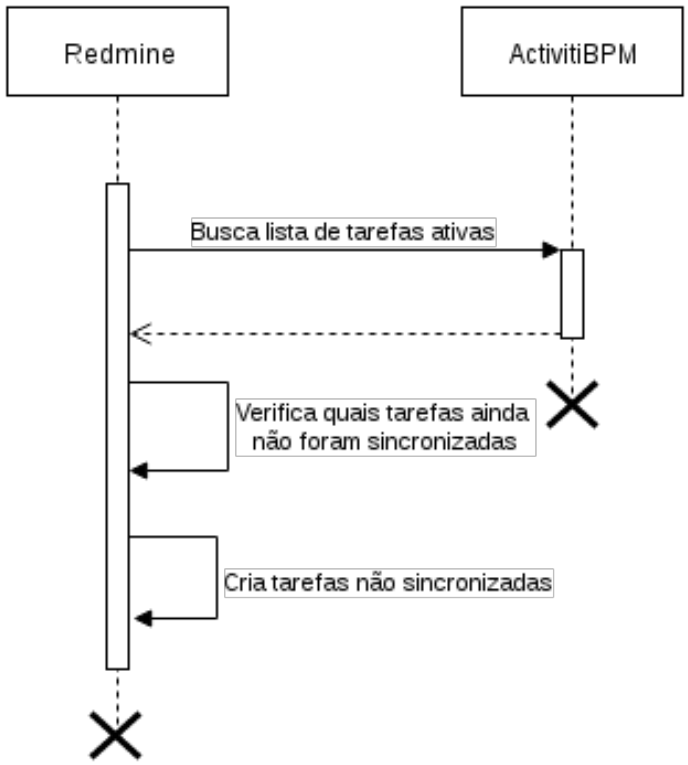

Figura 4.14: Diagrama de sequência de criação de tarefas do Activiti no Redmine

A Figura 4.14 descreve este procedimento: primeiramente, o Redmine realiza uma consulta à API REST do Activiti para buscar as tarefas que estão em andamento, então verifica quais ainda não foram mapeadas para si, e, por fim, cria as novas tarefas, guardando referência da tarefa original do Activiti para que não sejam criadas novamente em uma futura reexecução do processo.

4.5.4.4 Finalização de tarefas

Quando uma tarefa do Redmine é finalizada, é necessário concluir a sua contraparte no Activiti, quando houver, dando seguimento ao fluxo do processo no motor BPM. A Figura 4.15 mostra a sequência de processamento realizada com a conclusão da tarefa no Redmine.

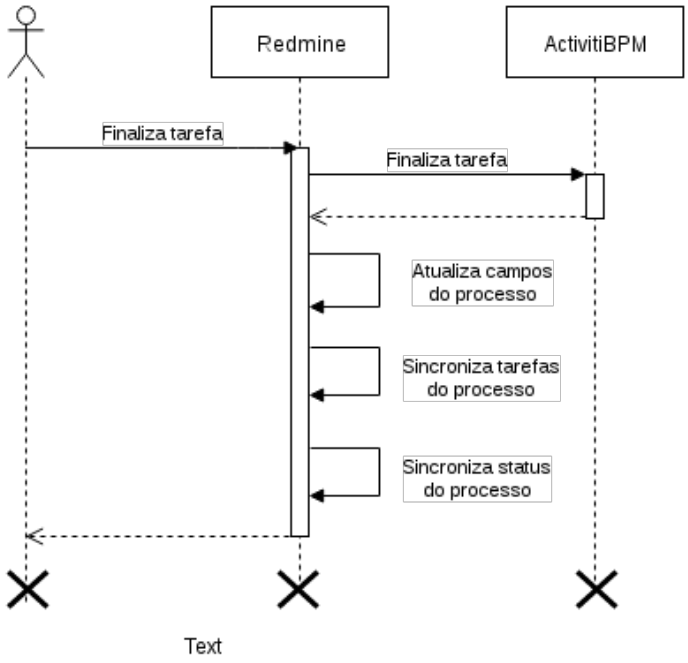

Figura 4.15: Diagrama de sequência de finalização de tarefas do Activiti pelo Redmine

Após o usuário concluir a tarefa (transicionar para uma situação que representa a sua conclusão), o Redmine utiliza a API REST do Activiti para finalizar a tarefa correspondente no BPMS. Uma vez a tarefa concluída, os campos da tarefa principal do Redmine que representa o processo do Activiti são atualizados com as alterações realizadas na tarefa e são executados, em sequência, os *jobs* de sincronização de tarefas e de processos, descritos nas seções 4.5.4.3 e 4.5.4.5, respectivamente. Esta ação é realizada pelo usuário comum da ferramenta, que é ator de alguma etapa, e possui uma tarefa atribuída para ele.

4.5.4.5 Atualização de processos

O último tópico que necessita ser sincronizado é o status do processo no Redmine para refletir o andamento do fluxo do processo no BPMS. Este procedimento é executado periodicamente para manter as tarefas do Redmine sempre atualizadas em relação ao Activiti.

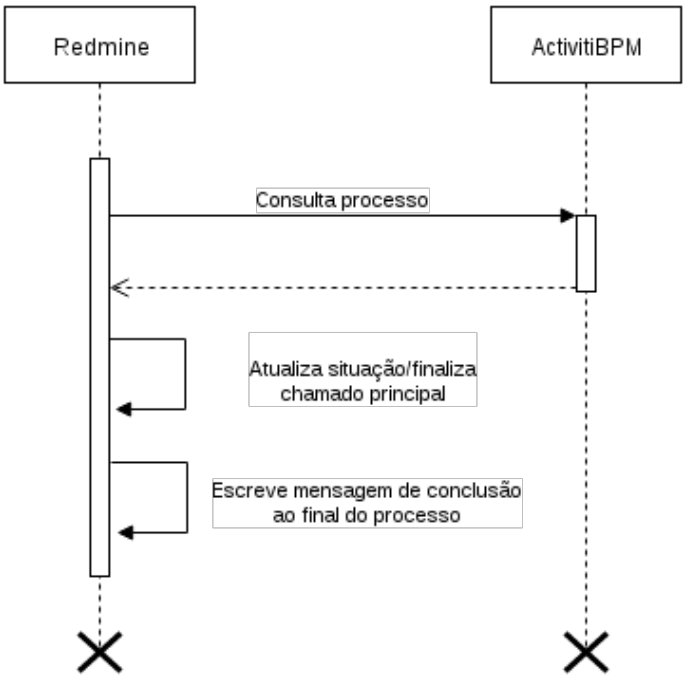

Figura 4.16: Diagrama de sequência de sincronização de status do processo

A Figura 4.16 mostra a sequência de execução desta sincronização. O Redmine consulta a API REST do Activiti para buscar os dados do andamento do processo. A partir dos dados obtidos é possível determinar se houve alteração na situação do processo ou ele foi terminado. Neste último caso, além de atualizar a situação do chamado no Redmine, também é escrita a mensagem de conclusão configurada no *plugin*.

4.6 Customização do Activiti BPM

A integração entre o Redmine e o Activiti BPM foi desenvolvida sob o aspecto de direcionar a maior parte das customizações para o lado do Redmine, uma vez que esta é apenas uma das possibilidades de interface com o motor de processos. Num cenário mais amplo de processos mais complexos, outros tipos de dispositivos ou sistemas poderiam realizar uma comunicação direta com os processos.

A API REST padrão oferecida pelo Activiti BPM foi utilizada para a integração entre as ferramentas, uma vez que é bem completa e oferece a maioria dos serviços

necessários para a comunicação. Também contou a facilidade de chamadas a APIs REST pelo Ruby on Rails.

Entretanto, identificamos a ausência de um serviço fundamental para integração entre as ferramentas. Esse serviço deveria retornar uma lista contendo as definições de tarefas contidas em um determinado processo, incluindo os campos disponíveis nos formulários das tarefas. Essa definição nos permitiria estabelecer a interface para o mapeamento dos campos do processo com os campos das tarefas do Redmine.

Sendo assim, foi necessário estendermos a API REST do Activiti BPM através da criação de uma classe Java representando um novo serviço, semelhante aos serviços existentes no seu código-fonte. Esse serviço consistiu no consumo de uma API Java já disponibilizada pelo Activiti BPM, mas ausente na API REST. O código-fonte pode ser consultado no apêndice B.1.

Capítulo 5

Avaliação

5.1 Introdução

Neste capítulo apresentaremos um caso real de utilização da ferramenta como demonstração do resultado obtido no desenvolvimento deste projeto. Além disso, apresentaremos trabalhos relacionados a alternativas que visam facilitar a gestão de processos.

5.2 Caso real

Em um cliente da Visagio, onde o Redmine já era utilizado para a gestão de diversos processos na área de Suprimentos, observou-se a necessidade da implantação de fluxos de aprovação mais complexos do que o Redmine poderia oferecer por padrão. Nesse cenário, observamos uma excelente oportunidade para avaliar na prática o funcionamento da solução descrita ao longo deste trabalho.

A Figura 5.1 demonstra o processo para solicitação de criação de cartão de crédito corporativo. Este processo foi configurado para ser conduzido no Activiti BPM e Redmine através do plugin do Redmine *BPM Integration* desenvolvido neste trabalho. O processo pode ser criado pelo funcionário que precisa utilizar um cartão, ou pelo seu superior, que abrirá em seu lugar. O processo é disparado mediante

a abertura de um chamado no Redmine, onde são preenchidos dados pessoais do solicitante, endereço para entrega do cartão e diretor aprovador. Após a criação do processo, o Activiti BPM imediatamente cria a próxima tarefa: *Workflow Diretor*. Através do mecanismo de sincronização, uma sub-tarefa é criada no Redmine, representando a aprovação do diretor. Esta tarefa tem o responsável definido pelo preenchimento do aprovador na tarefa anterior, sendo encaminhado através do fluxo no Activiti BPM ao Redmine definindo o responsável pela tarefa. Como é possível observar no diagrama, o processo segue por mais um nível de aprovação a ser executado pelo Normativo, área que é responsável por validar as informações preenchidas. Após aprovado, o processo segue para emissão, representada pela etapa *Workflow Banco*, a ser realizada pelo atendente do banco, que concluirá a mesma assim que o cartão for emitido e enviado ao solicitante. Caso algum dos aprovadores reprove a solicitação do cartão, o chamado é encerrado e assume o status Reprovado.

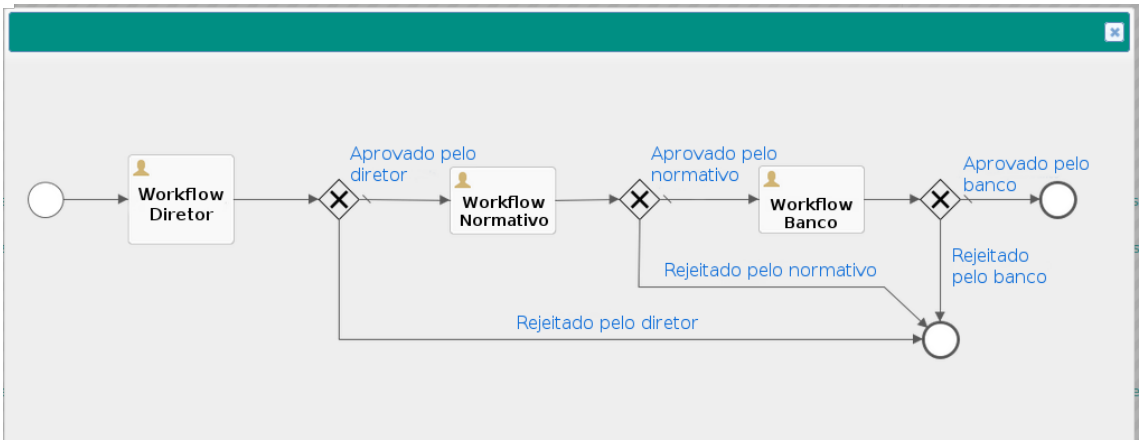

Figura 5.1: Modelo de processo de criação de cartão corporativo

No caso do processo descrito, a implementação no Activiti BPM permitiu que as tarefas de aprovação fossem atribuídas automaticamente, e que a reprovação de uma etapa resultasse no encerramento do processo com o status Reprovado. Esse comportamento não seria possível de maneira simples no Redmine, diferentemente do Activiti BPM, onde a modelagem permite esse tipo de fluxo de trabalho.

No mês seguinte a primeira experiência do uso da integração entre o Redmine

e o BPMS, novos processos foram implantados utilizando a mesma solução, dessa vez para orquestrar fluxos de cadastro de fornecedores da companhia. A Figura 5.2 representa a modelagem de um desses fluxos, em que é necessária a aprovação da área normativa do processo da empresa antes que o cadastro seja realizado.

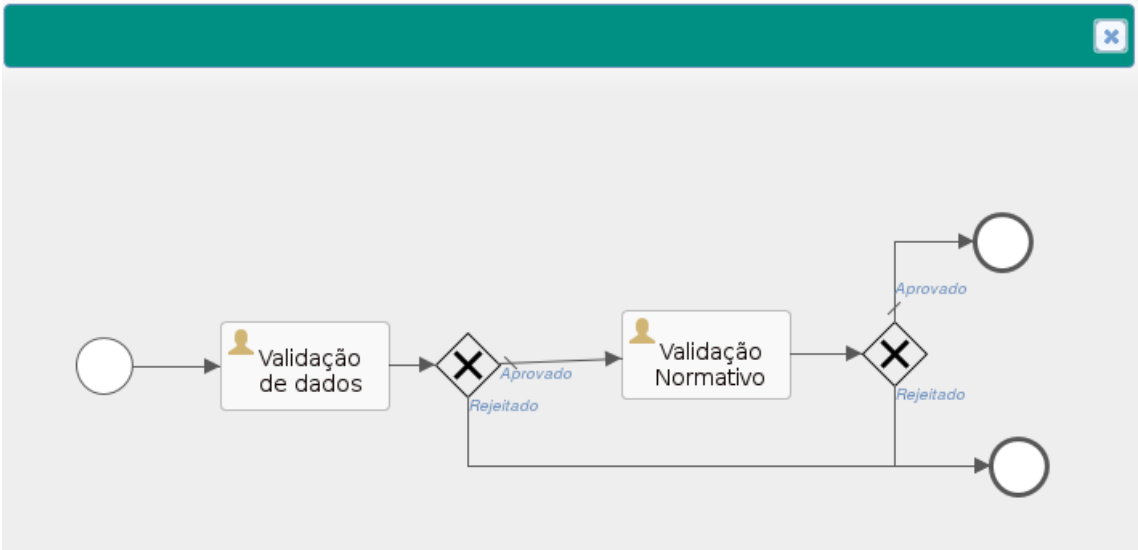

Figura 5.2: Modelo de processo de cadastro de fornecedores

Por fim, após 6 meses de uso do *plugin* objeto deste trabalho para o processo de cadastro de fornecedores, um novo e mais complexo grupo de processos foi incorporado à plataforma, o Recebimento Fiscal. As Figuras 5.3 e 5.4 ilustram o fluxo do processo de recebimento de Nota Fiscal de Serviços, que foi dividido em 2 partes para melhor visualização.

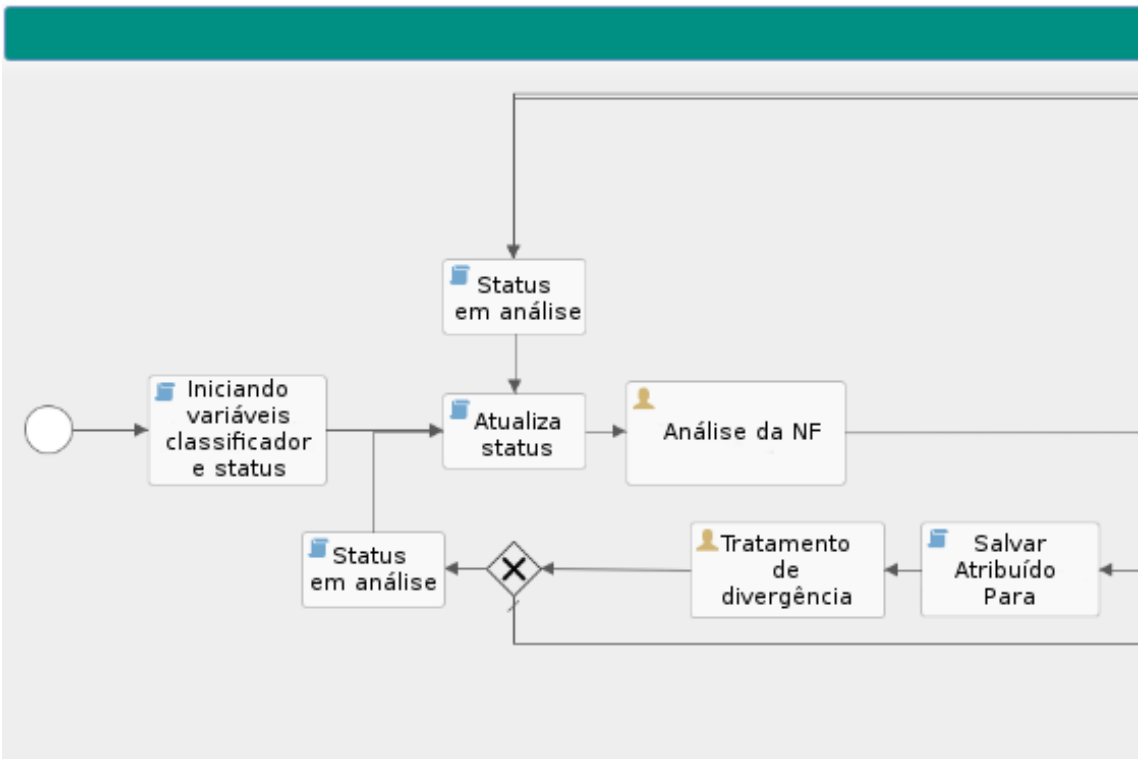

Figura 5.3: Modelo de processo de recebimento fiscal (Parte 1)

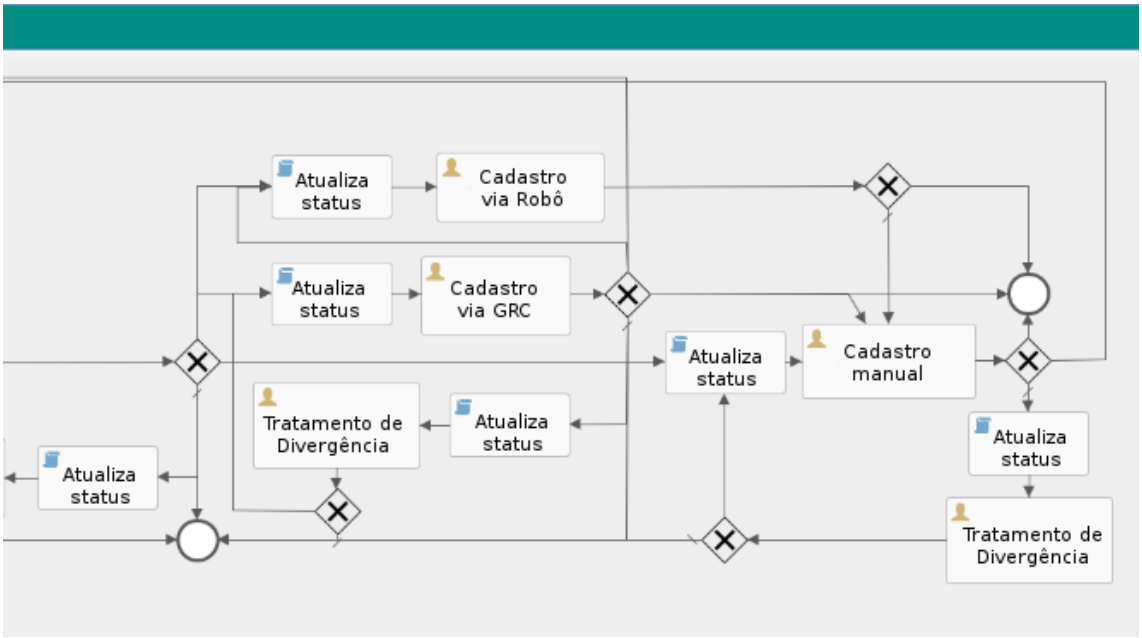

Figura 5.4: Modelo de processo de recebimento fiscal (Parte 2)

Este processo inclui desde o envio da nota fiscal pelo fornecedor, passando pela validação de seus dados pelos analistas da área de recebimento fiscal e tratamento de possíveis inconsistências encontradas, até o cadastro da nota no ERP[55] (*Enterprise Resource Planning*) da empresa.

Este processo envolve vários atores responsáveis pelas diferentes etapas do processo, e a distribuição das tarefas entre eles é feita automaticamente pela camada de integração desenvolvida.

No gráfico da Figura 5.5 é possível verificar o total de processos iniciados (Abertos) e concluídos (Fechados) para os três diferentes grupos de processos. Ao observarmos que todos os processos tem um alto percentual de conclusão, podemos concluir que a integração entre o Redmine e o Activiti BPM funcionou conforme o esperado, e que os processos fluíram normalmente com a interação dos atores do processo através do Redmine. Além disso, não foram observados reclamações importantes referentes ao funcionamento do fluxo de trabalho. No total, foram iniciados 70.162 processos desde Março de 2016, o que representa uma média de 220 processos iniciados por dia até Janeiro de 2017.

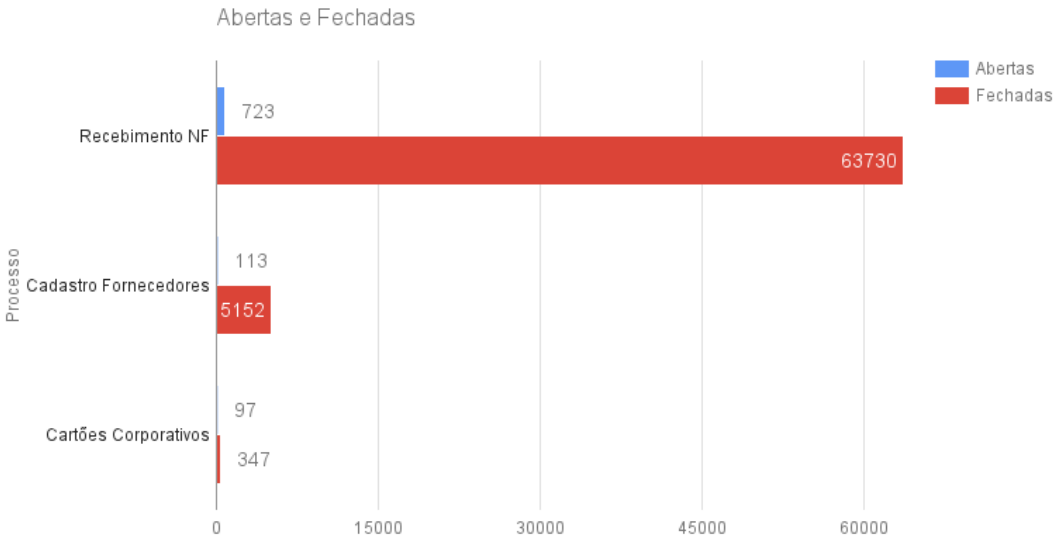

Figura 5.5: Processos utilizando o plugin

5.3 Trabalhos relacionados

A seguir são apresentados alguns trabalhos relacionados ao tema central deste trabalho, que é a utilização de metodologias e ferramentas com objetivo de facilitar e otimizar a gestão e execução de processos.

5.3.1 Melhoria de processos pelo BPM: aplicação no setor público

Este artigo [31] apresenta um relato de aplicação da metodologia BPM, que foi adaptada para o contexto de uma organização pública e utilizada para modernizar o processo de controle de trânsito animal no Brasil. A partir da análise do processo atual foram propostas melhorias a fim de otimizar recursos, melhorar a confiabilidade e aumentar a satisfação de clientes.

5.3.2 JIRA Core

JIRA Software[30] é um sistema de gestão de projetos de *software* que é utilizado para realizar a gestão de erros, melhorias e novas funcionalidades, utilizando fluxos de trabalho personalizados de acordo com a necessidade de cada projeto, como o Redmine.

No entanto, foi percebido pelos responsáveis da ferramenta o seu uso crescente para projetos não técnicos[20], fugindo do público alvo original. Então, para suprir a necessidade desses times foi criado o JIRA Core[29], que é um *software* de gestão de processos de negócio que permite a configuração de diferentes fluxos de trabalho pelos usuários administradores do próprio sistema e fornece mecanismos de controle e acompanhamento da execução desses processos.

A abordagem do JIRA Core foi diferente da proposta deste trabalho. Aquele buscou embutir nele mesmo, principalmente, a configuração de fluxos de processos simples, porém ainda mais complexos do que o JIRA Software permitia. O plugin *BPM Integration*, por sua vez permitiu a configuração de fluxos complexos através da integração do Redmine com o BPM.

Capítulo 6

Conclusão

6.1 Introdução

A automação de processos de negócio vem tornando-se cada vez mais fundamental nas organizações. Conforme citado por Mohapatra[59], "através da automação de processos, podemos desenvolver processos repetíveis, executáveis e gerenciáveis. Dessa forma transparente, as organizações permanecem focadas no seu verdadeiro propósito, naquilo que realmente gera valor ao negócio".

Neste capítulo apresentamos algumas considerações finais a respeito deste projeto, tais como reflexões sobre a motivação, busca da solução e conclusão sobre o seu desenvolvimento. Além disso, sugerimos algumas novas funcionalidades e melhorias como trabalhos futuros, para que este projeto continue sendo desenvolvido e cada vez mais se torne uma alternativa viável na automatização e gestão de processos de negócio.

6.2 Considerações

O combustível principal deste trabalho, fortemente compartilhado por todos os seus integrantes, foi a possibilidade de unir o interesse acadêmico e a incrível experiência vivenciada durante todos os anos de estudos na UFRJ com o objetivo

de gerar valor para o mercado corporativo através da Visagio[50], empresa na qual trabalhamos há mais de três anos e criada por ex-alunos desta instituição.

Tendo iniciado com a necessidade de automatizar e padronizar processos simples e com bastante interação humana, encontramos no Redmine uma ótima solução. Descobrimos que o Gerenciador de Projetos podia também ser utilizado como Gerenciador de Processos. Utilizar este sistema faz sentido pela excelente estrutura de gerenciamento de tarefas, acompanhamento do histórico, facilidade para criação de fluxos de trabalho, campos personalizados, e ótimo controle de permissões.

Contudo, se quisermos guiar processos mais complexos, com fluxos paralelos, atribuições, ou decisões de caminhos automáticas, o Redmine por si só não nos atenderia. Ele não foi desenvolvido originalmente com este objetivo, e não valeria a pena tentar desenvolver plugins para possibilitar a condução de processos complexos no Redmine, pois as funcionalidades que são necessárias para isto, já foram todas implementadas em sistemas chamados BPMS.

Os BPMS foram desenvolvidos exatamente com o objetivo de modelar fluxos complexos, e nos atenderiam muito bem em termos de execução do processo. No entanto, como uma das características dos processos que queremos automatizar é a necessidade de bastante interação humana, todas as funcionalidades que facilitam esta interação do Redmine e que já estávamos habituados seriam abandonados.

A proposta final deste trabalho, portanto, foi o desenvolvimento de um plugin para o Redmine, que chamamos de *BPM Integration*, para integrá-lo com um BPMS. Como descrevemos no Capítulo 4, a abordagem inicial foi utilizar um ESB[53] como uma camada de comunicação entre o Redmine e os BPMS, e desenvolver uma plataforma genérica à qual pudesse ser plugado qualquer BPMS. Devido a algumas dificuldades nesta estratégia, decidimos estabelecer a comunicação com apenas um BPMS e após avaliarmos as possibilidades disponíveis, escolhemos o Activiti BPM[1].

O plugin *BPM Integration* foi desenvolvido em Ruby on Rails, a linguagem na qual o Redmine foi feito. Ademais, a comunicação entre o Redmine e o Activiti BPM foi feita através do consumo da API REST disponibilizada pelo BPMS.

6.3 Melhorias propostas

Em alinhamento com a utilização de soluções de código-fonte aberto que compõem este trabalho, todo o código-fonte da integração desenvolvida está disponibilizado no GitHub da Visagio[19]. O *Redmine BPM Integration Plugin* segue em constante evolução através da implementação desta solução em clientes relevantes no Brasil, o que exige constantes melhorias e avanços no projeto inicial apresentado neste trabalho.

A seguir são apresentadas algumas propostas de melhorias para o seguimento deste trabalho:

6.3.1 Tarefas contínuas

Durante a implantação, um *feedback* foi recebido pelos usuários do sistema. O plugin introduziu um comportamento um pouco diferente na continuidade do processo do que era de costume no Redmine. Como explicado na Seção 4.5.4.3, a representação de uma tarefa humana do Activiti no Redmine é uma sub-tarefa do chamado original que representa o processo. Uma das principais razões desta abordagem foi a necessidade de contemplar processos que existem tarefas humanas que devem ser realizadas em paralelo, de modo que cada ator pudesse executar sua tarefa isoladamente numa sub-tarefa.

Esta modificação, no entanto, tirou um pouco a continuidade de um processo em que várias etapas são executadas sequencialmente por um mesmo ator. Na condução de um processo no Redmine sem o plugin *BPM Integration* os atores sempre interagem com o processo através da tarefa que representa o mesmo no Redmine.

Já na modelagem proposta com o plugin *BPM Integration* o usuário interage com o processo através da sub-tarefa atribuída para ele e altera a situação desta, representando a conclusão daquela etapa. Ao fazer isto o processo segue para a próxima etapa no Activiti BPM e cria a próxima atividade (3.3.1.1) e o plugin *BPM Integration* em seguida cria uma nova sub-tarefa, para permitir a interação do ator

com o processo, através do Redmine.

Quando o usuário conclui uma sub-tarefa, e ele é o responsável pela próxima etapa, ele não visualiza imediatamente na tela a próxima sub-tarefa criada, permitindo que avance as etapas numa mesma tela. Nestes casos é necessário acessar a tarefa pai da sub-tarefa que acabou de ser concluída (a que representa o processo), e à partir daí, acessar a próxima sub-tarefa. Neste cenário, existe possibilidade de melhorias para facilitar a execução das tarefas para o usuário, permitindo que o processo flua melhor quando as tarefas sequenciais são executadas por um mesmo ator.

As Figuras 6.2 e 6.1 mostram como é o acesso às tarefas mencionado acima, exemplificado com o processo de criação de cartões corporativos ilustrado na Figura 5.1. Após concluir a tarefa, o usuário deve clicar no link em amarelo na parte superior da tela exibida na Figura 6.1. Ao fazer isto é redirecionado para a tela da tarefa que representa o processo, exibida na Figura 6.2. Abaixo do título *Sub-chamados* estão as sub-tarefas. Como é possível observar, a primeira sub-tarefa é a que acabou de ser concluída, com a situação *Aprovado*, e a segunda é a próxima etapa a ser realizada, em cujo link o ator deve clicar para acessá-la.

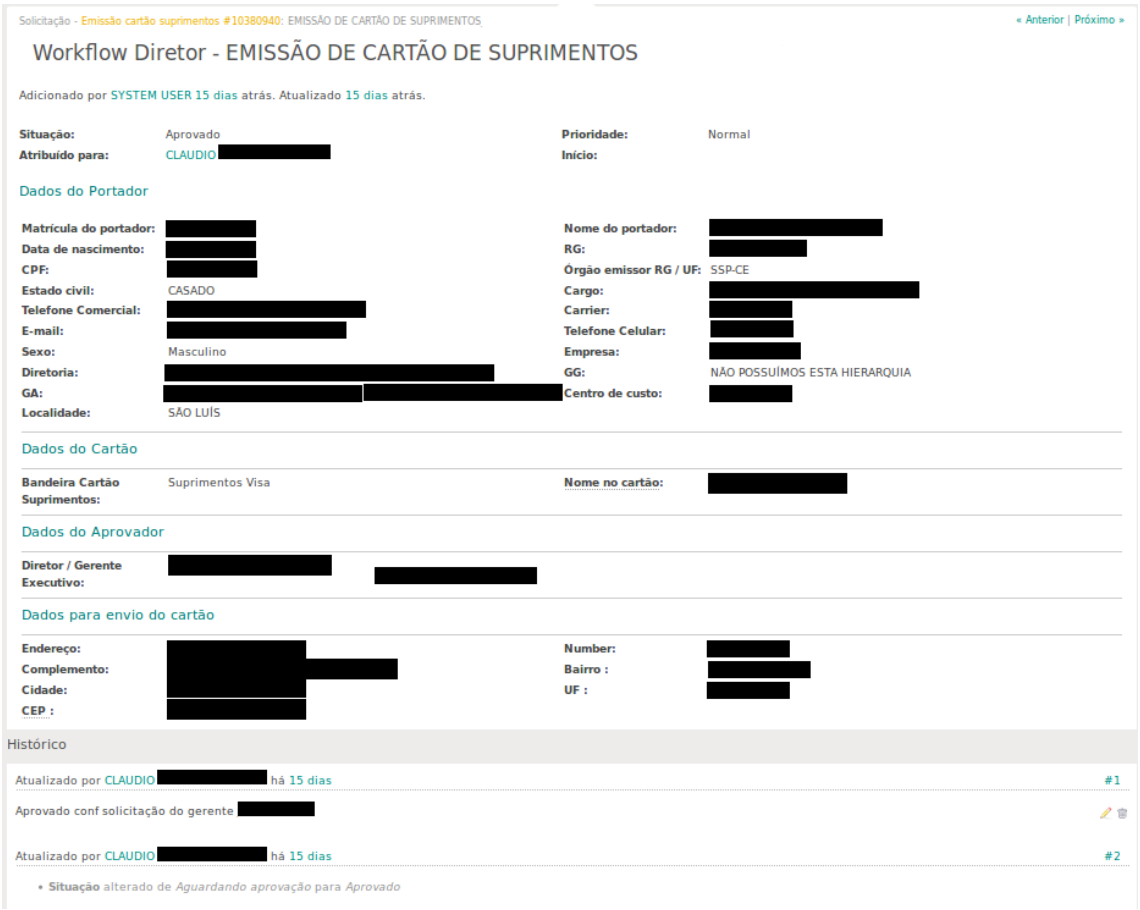

Figura 6.1: Sub-tarefa, representando uma das etapas do processo de criação de cartões corporativos

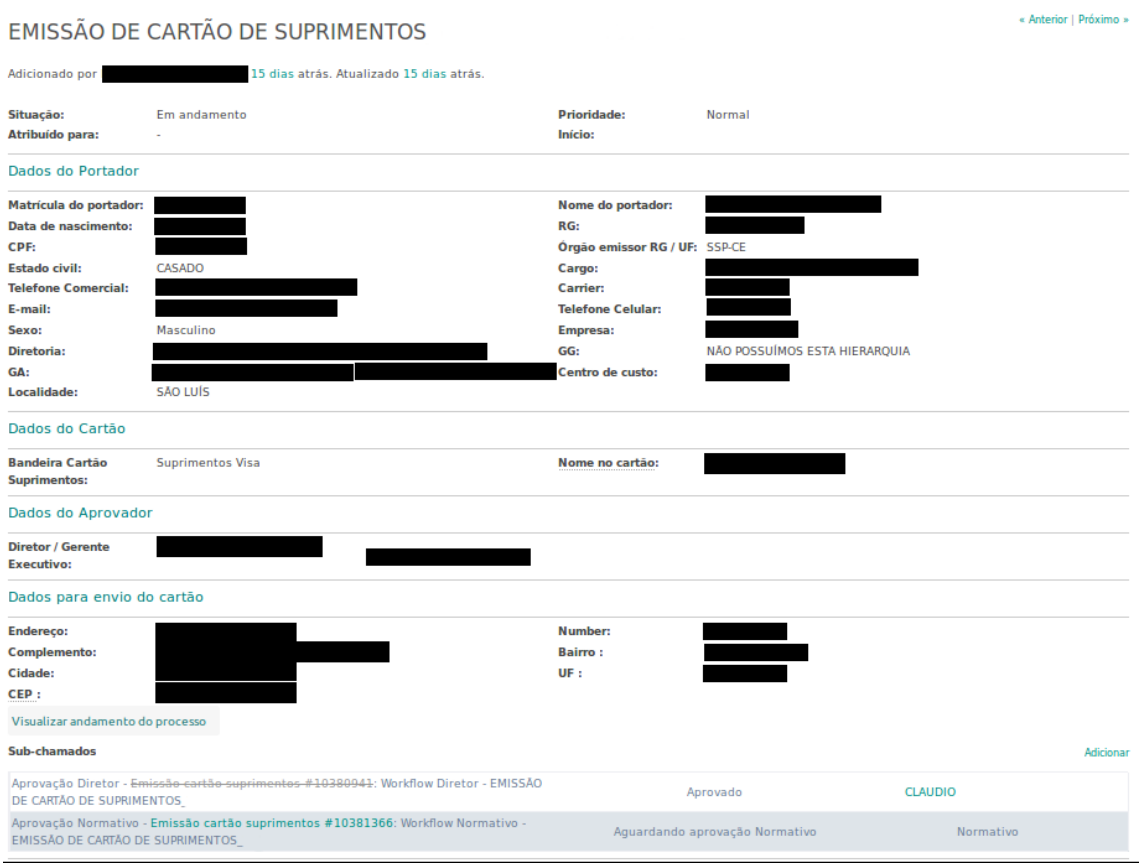

Figura 6.2: Tarefa representando o processo de criação de cartões corporativos

6.3.2 BPMN integrado

Neste trabalho utilizamos o Activiti Designer para a modelagem dos processos que é constituído basicamente do Eclipse com um plugin BPMN. Este cenário apresenta-se pouco prático para usuários sem conhecimento de TI. Nesse sentido, observamos a oportunidade de prover uma modelagem de processos fácil e integrada ao Redmine. Assim, o usuário poderia criar seu processo diretamente na ferramenta, com uma interface simples e online, sem a necessidade de contato com aplicações específicas de desenvolvimento.

Já existem iniciativas similares neste sentido, como o Activiti Modeler[4], exibido na Figura 6.3. Portanto, uma melhoria interessante seria acoplar esta ferramenta ao Redmine e avaliar as melhorias necessárias para deixar sua utilização agradável e simples numa única ferramenta.

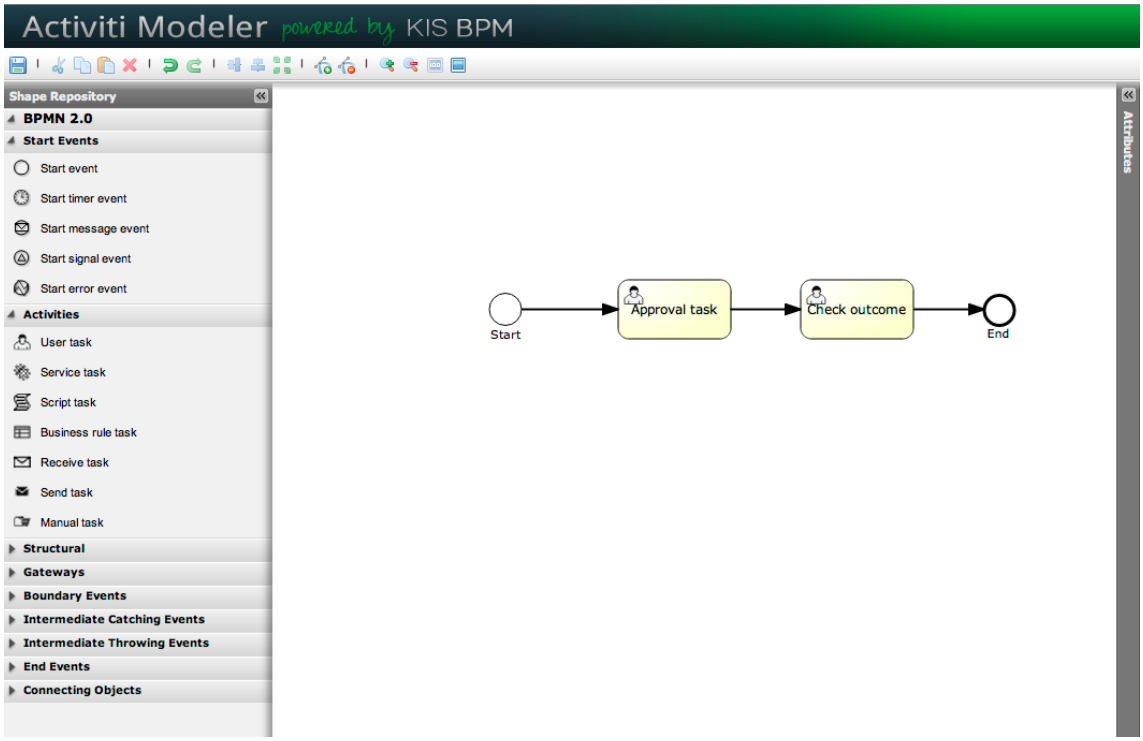

Figura 6.3: Activiti Modeler

6.3.3 BRM integrado

Uma característica bastante comum em sistemas BPMS é a presença de um BRM (*Business Rules Management*) para gestão de regras de negócio nos processos. O Activiti BPM suporta a integração com o Drools, uma aplicação *web* desenvolvida em Java especialista em BRM. O Drools permite a criação de uma tabela de decisão que pode ser configurada pelo usuário para definir uma regra de negócio baseada em intervalo de valores estabelecendo, por exemplo, as alçadas de aprovação em um processo de pagamento.

Uma proposta de evolução no contexto da integração do BPMS com o Redmine seria o desenvolvimento de um plugin BRM que permita a gestão de regras de negócio para processos BPM diretamente no Redmine.

6.3.4 BPMS genérico

Conforme discutido na Seção 4.3, esta melhoria proporcionaria uma camada genérica de integração de qualquer BPMS ao Redmine, flexibilizando assim a escolha do BPMS ou até mesmo a integração com um BPMS que já esteja em uso pela empresa.

Referências

[1] Activiti BPM. http://activiti.org/.

[2] Activiti Designer. https://eclipse.org/.

[3] Activiti Download. http://activiti.org/download.html.

[4] Activiti modeler. http://bpmn20inaction.blogspot.com.br/2012/09/activiti-modeler-getting-started-part-1.html.

[5] Activiti User Guide. http://www.activiti.org/userguide.

[6] Apache License. http://www.apache.org/licenses/LICENSE-2.0.

[7] API. http://www.webopedia.com/TERM/A/API.html.

[8] API REST. http://www.restapitutorial.com/.

[9] Bonita BPM. http://www.bonitasoft.com/.

[10] BPM. http://www.gartner.com/it-glossary/business-process-management-bpm/.

[11] BPMI. http://searchcio.techtarget.com/definition/Business-Process-Management-Initiative-BPMI.

[12] BPMN. http://www.omg.org/bpm/#BPMN.

[13] BPMN 2.0. http://www.omg.org/spec/BPMN/2.0/.

[14] BPMS. http://www.gartner.com/it-glossary/bpms-business-process-management-suite/.

[15] Componentes do Activiti BPM. http://activiti.org/components.html.

[16] CSC. http://www.visagio.com/blog/2012/08/entenda-melhor-a-diferenca-entre-um-csc-e-a-centralizacao-de-servicos/.

[17] Eclipse. https://eclipse.org/.

[18] GitHub. https://github.com/activiti.

[19] GitHub Visagio. https://github.com/visagio/redmine_bpm_integration.

[20] A guide to setting up business workflows using jira core. http://blogs.atlassian.com/2015/11/how-to-set-up-business-workflows-in-jira-core/.

[21] H2. http://www.h2database.com/html/main.html.

[22] H2 In-Memory. http://www.h2database.com/html/features.html#in_memory_databases.

[23] IBM DB2. http://www.ibm.com/analytics/us/en/technology/db2/.

[24] IDE. http://searchsoftwarequality.techtarget.com/definition/integrated-development-environment.

[25] Intalio BPM. http://www.intalio.com/products/bpms/overview/.

[26] Introduction to the Enterprise Service Bus. O'Reilly Media, 2004. Disponível em: ftp://ftp.oreilly.com/pub/summit/ch01.pdf, Aceso em: 15 de jan. de 2017.

[27] JAR file. https://docs.oracle.com/javase/tutorial/deployment/jar/basicsindex.html.

[28] jBPM. https://docs.jboss.org/jbpm/v6.2/userguide/.

[29] Jira core. https://www.atlassian.com/software/jira/core.

[30] Jira software. https://www.atlassian.com/software/jira.

[31] Melhoria de Processos pelo BPM: aplicação no setor público. http://hdl.handle.net/10183/65643.

[32] MuleSoft. https://www.mulesoft.com/.

[33] MySQL. https://www.mysql.com/.

[34] OMG. http://www.omg.org/.

[35] Oracle. https://www.oracle.com/database/index.html.

[36] Oracle BPM. http://www.oracle.com/us/technologies/bpm/overview/index.html.

[37] PostgreSQL. https://www.postgresql.org/.

[38] Redmine. http://www.redmine.org/projects/redmine/wiki.

[39] Redmine 3.1. http://www.redmine.org/news/100.

[40] Redmine Theme List. http://www.redmine.org/projects/redmine/wiki/Theme_List.

[41] SAP NetWeaver BPM. http://scn.sap.com/people/arafat.farooqui/blog/2009/08/05/part-i-an-introduction-to-sap-netweaver-bpm.

[42] SQL Server. https://www.microsoft.com/pt-br/server-cloud/products/sql-server/.

[43] Tipos de agrupadores. http://kb.qpr.com/qpr2012_2/index.html?swimlanes.htm.

[44] Tipos de artefatos. https://www.gliffy.com/blog/2016/02/09/what-is-business-process-model-notation-bpmn/.

[45] Tipos de atividades. http://blog.goodelearning.com/bpmn/common-bpmn-modeling-mistakes-activities/.

[46] Tipos de conectores. https://en.wikipedia.org/wiki/File:Different_Types_of_BPMN_connections.png#/media/File:Different_Types_of_BPMN_connections.png.

[47] Tipos de decisões. http://training-course-material.com/index.php?title=BPMN_2.0_dla_Analitykow_Biznesowych&action=slide.

[48] Tipos de eventos. http://training-course-material.com/training/BPMN_2.0_Events.

[49] Tomcat 7. https://tomcat.apache.org/tomcat-7.0-doc/index.html.

[50] Visagio. http://www.visagio.com/pt/.

[51] WAR file. http://searchsoa.techtarget.com/answer/What-is-a-WAR-file.

[52] XML. http://www.w3schools.com/xml/.

[53] Enterprise service bus (esb). http://searchsoa.techtarget.com/definition/enterprise-service-bus, Outubro 2007. Accessado em: 2016-06-25.

[54] Rails 2.0: It's done! http://weblog.rubyonrails.org/2007/12/7/rails-2-0-it-s-done, Dezembro 2007. Accessado em: 2016-06-25.

[55] de Sousa, J. M. E. DEFINITION AND ANALYSIS OF CRITICAL SUCCESS FACTORS FOR ERP IMPLEMENTATION PROJECTS. 2013. Tese (Doutorado em Psicologia) - Universitat Politècnica de Catalunya. Barcelona, Espanha, 2013. Disponível em http://jesteves.com/Tesis_phd_jesteves.pdf.

[56] Deitel, H. Java: como programar. 6ª edição. Pearson Prentice Hall, 2005.

[57] Hartl, M. Ruby on Rails Tutorial (Rails 5), 4ª Edição, 2016.

[58] Lesyuk, A. Mastering Redmine, 1ª Edição. 2013. Disponível em https://www.amazon.com/Mastering-Redmine-Andriy-Lesyuk/dp/1849519145.

[59] MOHAPATRA, S. Business Process Automation. 2009. Disponível em https://books.google.com.br/books?hl=pt-BR&lr=&id=qvJdpqUquDAC&oi=fnd&pg=PR993&dq=business+process+automation&ots=Dw3x8GZvYU&sig=rs3MforXRu7MTmZ02BqyAEe5CjI#v=onepage&q=business%20process%20automation&f=false.

[60] OF BUSINESS PROCESS MANAGEMENT PROFESSIONALS, A. BPM CBOK, 1ª Edição. 2013. Disponível em http://c.ymcdn.com/sites/www.abpmp.org/resource/resmgr/Docs/ABPMP_CBOK_Guide__Portuguese.pdf.

[61] THOMAS, D., E HANSSON, D. H. Desenvolvimento Web Ágil com Rails. 2ª Edição. Bookman, 2008. 680p.

[62] TSIDULKO, J. Here Are The 5 Leaders In Gartner's 2016 Magic Quadrant For Integration Platform-As-A-Service. CRN, 09/08/2016. Disponível em: http://www.crn.com/slide-shows/cloud/300080296/here-are-the-5-leaders-in-gartners-2016-magic-quadrant-for-integration-platfor htm/pgno/0/4. Accessado em: 15 de jan. de 2017.

[63] VAN DER AALST, W. M. P., T. H. A. H. M., E WESKE, M. Business Process Management: A Survey. 2003. Disponível em http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.90.9525&rep=rep1&type=pdf.

[64] XAVIER, L. Walmart erra preço de produto em site, cancela venda e irrita clientes. http://oglobo.globo.com/economia/defesa-do-consumidor/walmart-erra-preco-de-produto-em-site-cancela-venda-irrita-clientes-11098550, Dezembro 2013. Acessado em: 2016-06-25.

Apêndice A

Processo de Precificação em BPMN

Código A.1: Código do Processo de Precificação em BPMN

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
    "http://www.w3.org/2001/XMLSchema" xmlns:activiti="http://
    activiti.org/bpmn" xmlns:bpmndi="http://www.omg.org/spec/BPMN
    /20100524/DI" xmlns:omgdc="http://www.omg.org/spec/DD/20100524/
    DC" xmlns:omgdi="http://www.omg.org/spec/DD/20100524/DI"
    typeLanguage="http://www.w3.org/2001/XMLSchema"
    expressionLanguage="http://www.w3.org/1999/XPath"
    targetNamespace="http://www.activiti.org/test">
  <collaboration id="Collaboration">
    <participant id="processo_precificacao" name="Processo de
        Precificacao" processRef="process_pool1"></participant>
  </collaboration>
  <process id="process_pool1" name="process_pool1" isExecutable="
      true">
    <laneSet id="laneSet_process_pool1">
      <lane id="analista" name="Analista">
        <flowNodeRef>solicitar_troca_preco</flowNodeRef>
        <flowNodeRef>avaliar_alcada_precificacao</flowNodeRef>
        <flowNodeRef>decisao_aprovacao</flowNodeRef>
        <flowNodeRef>inicio</flowNodeRef>
      </lane>
```

```
14        <lane id="aprovador" name="Aprovador">
15          <flowNodeRef>exclusivegateway2</flowNodeRef>
16          <flowNodeRef>aprovar_precificacao</flowNodeRef>
17          <flowNodeRef>processo_finalizado_nao_precificado</
                flowNodeRef>
18        </lane>
19        <lane id="sistema" name="Sistema">
20          <flowNodeRef>trocar_preco</flowNodeRef>
21          <flowNodeRef>processo_finalizado_precificado</flowNodeRef>
22        </lane>
23      </laneSet>
24      <startEvent id="inicio" name="Inicio"></startEvent>
25      <userTask id="solicitar_troca_preco" name="Solicitar Troca de
            Preco"></userTask>
26      <sequenceFlow id="flow1" sourceRef="inicio" targetRef="
            solicitar_troca_preco"></sequenceFlow>
27      <userTask id="aprovar_precificacao" name="Aprovar Precificacao"
            >
28        <multiInstanceLoopCharacteristics isSequential="false"
              activiti:collection="aprovadores"></
              multiInstanceLoopCharacteristics>
29      </userTask>
30      <serviceTask id="trocar_preco" name="Trocar Preco"></
            serviceTask>
31      <sequenceFlow id="fim_processo_finalizado_precificado"
            sourceRef="trocar_preco" targetRef="
            processo_finalizado_precificado"></sequenceFlow>
32      <exclusiveGateway id="decisao_aprovacao" name="Decisao da
            Necessidade de Aprovacao"></exclusiveGateway>
33      <sequenceFlow id="aprovacao_necessaria" name="Aprovacao
            Necessaria" sourceRef="decisao_aprovacao" targetRef="
            aprovar_precificacao"></sequenceFlow>
34      <sequenceFlow id="aprovacao_automatica" name="Aprovacao
            Automatica" sourceRef="decisao_aprovacao" targetRef="
            trocar_preco"></sequenceFlow>
35      <businessRuleTask id="avaliar_alcada_precificacao" name="
            Avaliar Alcada de Precificacao"></businessRuleTask>
```

```
36      <sequenceFlow id="flow9" sourceRef="solicitar_troca_preco"
            targetRef="avaliar_alcada_precificacao"></sequenceFlow>
37      <sequenceFlow id="flow10" sourceRef="
            avaliar_alcada_precificacao" targetRef="decisao_aprovacao"><
            /sequenceFlow>
38      <exclusiveGateway id="exclusivegateway2" name="Decisao da
            Aprovacao"></exclusiveGateway>
39      <sequenceFlow id="flow11" sourceRef="aprovar_precificacao"
            targetRef="exclusivegateway2"></sequenceFlow>
40      <sequenceFlow id="aprovado" name="Aprovado" sourceRef="
            exclusivegateway2" targetRef="trocar_preco"></sequenceFlow>
41      <sequenceFlow id="fim_processo_finalizado_nao_precificado" name
            ="Nao Aprovado" sourceRef="exclusivegateway2" targetRef="
            processo_finalizado_nao_precificado"></sequenceFlow>
42      <endEvent id="processo_finalizado_nao_precificado" name="Fim">
43        <terminateEventDefinition></terminateEventDefinition>
44      </endEvent>
45      <endEvent id="processo_finalizado_precificado" name="Fim">
46        <terminateEventDefinition></terminateEventDefinition>
47      </endEvent>
48    </process>
49 </definitions>
```

Apêndice B

REST de Definições de Tarefas

Código B.1: REST de Definições de Tarefas

```
1  package com.redminebpm.integration.service.api;
2
3  import java.util.ArrayList;
4  import java.util.List;
5
6  import javax.servlet.http.HttpServletRequest;
7
8  import org.activiti.engine.RepositoryService;
9  import org.activiti.engine.impl.RepositoryServiceImpl;
10 import org.activiti.engine.impl.bpmn.behavior.
       UserTaskActivityBehavior;
11 import org.activiti.engine.impl.pvm.PvmActivity;
12 import org.activiti.engine.impl.pvm.ReadOnlyProcessDefinition;
13 import org.activiti.engine.impl.pvm.delegate.ActivityBehavior;
14 import org.activiti.engine.impl.pvm.process.ActivityImpl;
15 import org.activiti.engine.impl.task.TaskDefinition;
16 import org.activiti.rest.common.api.DataResponse;
17 import org.springframework.beans.factory.annotation.Autowired;
18 import org.springframework.web.bind.annotation.PathVariable;
19 import org.springframework.web.bind.annotation.RequestMapping;
20 import org.springframework.web.bind.annotation.RequestMethod;
21 import org.springframework.web.bind.annotation.RestController;
22
```

```
23 @RestController
24 public class TaskDefinitionService <T extends PvmActivity> {
25
26     @Autowired
27     protected RepositoryService repositoryService;
28
29     @SuppressWarnings("unchecked")
30     @RequestMapping(value="/repository/task-definitions/{
           processDefinitionId}", method = RequestMethod.GET, produces =
            "application/json")
31     public DataResponse getTaskDefinitions(@PathVariable String
           processDefinitionId, HttpServletRequest request) {
32         DataResponse dataResponse = new DataResponse();
33
34         ReadOnlyProcessDefinition processDefinition =
35                 ((RepositoryServiceImpl)repositoryService)
36                 .getDeployedProcessDefinition(processDefinitionId);
37
38         List<TaskDefinition> allActivities = new ArrayList<
               TaskDefinition>();
39         if (processDefinition != null) {
40             List<T> activities = (List<T>) processDefinition.
                   getActivities();
41             addActivities(activities, allActivities);
42         }
43
44         dataResponse.setData(allActivities);
45         return dataResponse;
46     }
47
48     @SuppressWarnings("unchecked")
49     private void addActivities(List<T> activities, List<
           TaskDefinition> taskDefinitions) {
50         for (T activity : activities) {
51             if (! activity.getActivities().isEmpty()) {
52                 addActivities((List<T>)activity.getActivities(),
                       taskDefinitions);
53             } else {
```

```
54                ActivityBehavior activityBehavior = ((ActivityImpl)
                      activity).getActivityBehavior();
55                if (activityBehavior instanceof
                      UserTaskActivityBehavior) {
56                    TaskDefinition taskDefinition = ((
                          UserTaskActivityBehavior)activityBehavior).
                          getTaskDefinition();
57                    taskDefinitions.add(taskDefinition);
58                }
59            }
60        }
61    }
62 }
```