Universidade Federal
do Rio de Janeiro
Escola Politécnica

IDENTIFICAÇÃO DE NÃO CONFORMIDADES EM PROCESSOS INTENSIVOS EM CONHECIMENTO

Guilherme Magalhães Almeida

Projeto de Graduação apresentado ao Curso de Engenharia de Computação e Informação da Escola Politécnica, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Engenheiro.

Orientador: Toacy Cavalcante de Oliveira

Rio de Janeiro

Dezembro de 2017

IDENTIFICAÇÃO DE NÃO CONFORMIDADES EM PROCESSOS INTENSIVOS EM CONHECIMENTO

Guilherme Magalhães Almeida

PROJETO DE GRADUAÇÃO SUBMETIDO AO CORPO DOCENTE DO CURSO DE ENGENHARIA DE COMPUTAÇÃO E INFORMAÇÃO DA ESCOLA POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO ELETRÔNICO E DE COMPUTAÇÃO

Autor:

Guilherme Magalhães Almeida

Orientador:

Prof. Toacy Cavalcante de Oliveira, D.Sc.

Examinador:

Prof. Claudia Maria Lima Werner, D.Sc.

Examinador:

Prof. Renata Mesquita da Silva Santos, M.Sc.

Rio de Janeiro

Dezembro de 2017

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
Escola Politécnica - Departamento de Eletrônica e de Computação
Centro de Tecnologia, bloco H, sala H-217, Cidade Universitária
Rio de Janeiro - RJ CEP 21949-900

DEDICATÓRIA

Opcional.

AGRADECIMENTO

Aos meus pais, a quem eu devo tudo na minha vida, eu sou muito grato por tudo que fizeram e fazem por mim. Foram os primeiros professores e amigos da minha vida, obrigado pelo apoio e incentivo para que eu chegasse até aqui. A minha irma, pelo companheirismo, risadas e seu habitual bom humor.

Agradeço à minha namorada Camilla por sempre estar ao meu lado, me dando forças para continuar e acreditando nos meus sonhos.

Aos meus avós e tios, por serem maravilhosos e terem me recebido em suas casas e pelo apoio contínuo.

Agradeço ao meu orientador Toacy Cavalcante de Oliveira e aos professores da UFRJ que fizeram parte dessa minha jornada.

Quero agradecer a sociedade brasileira pela oportunidade de estudar numa grande universidade pública e de participar do programa ciência sem fronteiras pela CAPES, que foi um ano de grande crescimento e aprendizado.

Gostaria também de agradecer aos meus amigos, nossos caminhos se cruzaram das mais diversas maneiras. Alguns por um pequeno período, outros, amigos pela vida inteira. Guardo um pouco de cada um, todos são muito importantes para mim.

RESUMO

Muitas tarefas e processos possuem uma estrutura bem definida e uma clara sequência de ações que devem ser realizadas. Esses processos podem ser automatizados e gerenciados levando a um aumento na qualidade do processo, na produtividade e uma maior monitoração e entendimento sobre o processo. Outros processos, chamados de processos intensivos em conhecimento, não podem ser modelados em sua completude. Esse tipo de trabalho é pouco estruturado e depende fortemente do conhecimento, colaboração e experiência dos trabalhadores do conhecimento, profissionais altamente especializados. Esses trabalhadores possuem autonomia para decidir o que deve ser feito em cada caso e possuem grande importância para suas organizações e para sociedade de maneira geral. Eles realizam trabalhos difíceis ou impossíveis de serem automatizados, porém é importante prover suporte para facilitar o reúso de conhecimento, colaboração e boas práticas advindas da experiência.

Apesar dos processos intensivos em conhecimento não serem estruturados, podem existir regras que balizam a execução do processo e boas práticas podem ser formuladas a partir da experiência com execuções anteriores. Esse trabalho visa o desenvolvimento de um plugin que permite identificação de não conformidades entre o modelo e o plano de execução. A identificação de não conformidades pode auxiliar na identificação de más praticas, possíveis pontos de melhoria no modelo, ou ser usado para simulação e aprendizado.

Palavras-Chave: trabalho intensivo em conhecimento, trabalhador de conhecimento, gestão de processos, gerenciamento de casos.

ABSTRACT

A lot of tasks and processes have a well defined structure and a clear sequence of actions that must be performed. These processes can be automated and managed leading to an increase in the quality of the process, in productivity and to a higher monitoring and understanding about the process. Other processes, called knowledge intensive processes, cannot be modeled in their completeness. This kind of process is loosely structured and heavily dependent on the collaboration, experience and knowledge of knowledge workers, highly specialized professionals. These workers possess autonomy to make decisions on a case by case basis and they have great value to their organization and to society in general. The work they perform are difficult or impossible to automate, but it is important to support their work by facilitating reuse of knowledge, collaboration and good practices extracted from experience.

Although knowledge intensive processes are not structured, it might have restrictions that drive the execution of the process and best practices can be formulated from prior experience. The development of a plugin that identifies non-compliance between the model and the execution plan is the goal of this work. The identification of non-compliance can reveal bad practices, potential points of improvement of the model, or it can be used for simulation and learning purposes.

Keywords: Knowledge intensive process, knowledge workers, process management, case management.

SIGLAS

BPM - *Business Process Management*

BPMN - *Business Process Model and Notation*

CMMN - *Case Management Model and Notation*

KIP - *Knowledge Intensive Processes*

CMMI - *Capability Maturity Model Integration*

OMG - *Object Management Group*

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Tema

Soluções tecnológicas e produtos de software estão cada vez mais presentes na nossa sociedade e alteram a maneira pela qual as pessoas se relacionam, consomem, trabalham e produzem cultura. Desde a infraestrutura das cidades até tarefas cotidianas como o uso do GPS, *smartphone*, redes sociais, a tecnologia permeia quase todas as atividades diárias, sendo perceptível ou não. A tecnologia também traz mudanças na natureza do trabalho e como ele é desempenhado.

Grande parte dos trabalhos repetitivos foram automatizados e, segundo Ivan Marques, quando isso ocorre não somente existe diminuição do número de postos de trabalho, mas ocorre uma migração de postos de trabalho de materialização para postos de "informacionalização" [6]. A tendência é de aumento na automatização de trabalhos repetitivos e consequente migração dos postos para área de "informacionalização", na qual os profissionais trabalham sobre conceitos, concepções e projetos. Esses trabalhadores podem ser chamados de trabalhadores do conhecimento e geram grande valor para economia atual [6].

Processo é um conjunto de atividades estruturadas que respondem a eventos e geram algum valor. A modelagem de processos de negócios permite um maior controle sobre a execução de um negócio, trazendo benefícios como aumento de produtividade e maior qualidade do trabalho [7].

Outro artefato que visa organizar o projeto é o plano de projeto. Ele fornece uma visão compartilhada do que o projeto deve atingir, com clareza das responsabilidades dos membros, e delineação das atividades a serem realizadas. O plano de projeto auxilia no controle de qualidade e na definição de orçamento e prazos realistas [8].

Muitos processos podem ser analisados, modelados completamente na fase de design, pois são bem estruturados e com uma sequência de execução bem definida. O *Business Process Model and Notation* (BPMN) é uma notação que permite a modelagem de workflow bem estruturado.

Para os trabalhadores do conhecimento, nem sempre é possível realizar essa gestão de processos na sua abordagem tradicional, mas é necessário dar um suporte para esses trabalhadores. Esse suporte pode ser uma ferramenta que auxilie o trabalhador a tomar decisões, a gerenciar o conhecimento ou a verificar se seu trabalho está em conformidade com algumas regras existentes de boas práticas.

Os processos intensivos em conhecimento, realizados pelos trabalhadores do conhecimento, não são bem estruturados. Esses processos dependem muito da colaboração, conhecimento e experiência dos seus participantes e não possuem uma sequência bem definida de tarefas, uma vez que sua execução é dependente do julgamento do participante para cada caso específico. Em 1968, Peter Drucker já ressaltava a importância de suportar o trabalho desse grupo de trabalhadores: "Melhorar a performance dos trabalhadores do conhecimento é tarefa econômica mais importante da nossa era" [9].

Fornecer suporte aos trabalhadores do conhecimento continua sendo um desafio da nossa era, assim como a readaptação dos trabalhadores que tiveram seus postos retirados ou migrados. Existem diversas propostas para auxiliar a modelagem e execução dos processos intensivos em conhecimento e o BPMN não é adequado, pois sua rigidez não é suficientemente adequada para capturar a flexibilidade inerente a tais processos. Outras propostas levam em conta a flexibilidade necessária na execução, a autonomia dos trabalhadores ou ainda a integração do campo de gestão do conhecimento com gestão de processos.

Uma das propostas é o *Case Management Model and Notation* (CMMN), cuja primeira especificação foi liberada em 2014 para suportar modelagem dos processos intensivos em conhecimento. O CMMN é orientado a casos e permite adições de tarefas e modificações em tempo de execução, sendo uma modelagem muito mais flexível.

Nesse cenário de processos intensivos em conhecimento, é complicado garantir aderência rígida ao processo pela própria natureza flexível de sua execução. Porém, com a experiência e diversas execuções, algumas partes do processo podem possuir algumas regras, restrições e boas práticas. Esses fragmentos do processo terão algumas de suas regras verificadas pelo *Plugin* desenvolvido nesse trabalho.

A motivação desse trabalho é auxiliar os trabalhadores do conhecimento atráves do desenvolvimento de um *Plugin*, chamado KIP Plan Builder que permite a identificação de não conformidades entre o plano de projeto e a definição do processo. A proposta é desenvolver o *Plugin* para Camunda [10], uma plataforma de execução e gerenciamento de processos. Nesse *Plugin*, o usuário pode criar um plano de projeto e gerar um relatório de não conformidades. A identificação de não conformidades pode auxiliar na identificação de más práticas, na questão de melhoria do modelo ou ser usada em simulações.

1.2 Metodologia

Primeiro foi realizado uma revisão da literatura sobre os tópicos relevantes do trabalho: trabalhadores do conhecimento; processos intensivos em conhecimento; gerenciamento de casos; CMMN.

A segunda etapa foi definir os requisitos da ferramenta, tanto funcionais quanto técnicos, e os casos de uso. Nesse etapa que ficou decidido que seria um *Plugin* para Camunda, pois Camunda era uma ferramenta *open source* bastante utilizada e com comunidade participativa.

A terceira etapa foi o desenvolvimento da ferramenta em Java e Angular e foi realizada a integração com o Camunda. Foram executados alguns estudos de caso

para apresentação dos resultados.

A última etapa foi a escrita e múltiplas revisões desse documento.

1.3 Organização do Trabalho

Este trabalho tem a seguinte estrutura, o Capítulo 2 apresenta a fundamentação teórica do trabalho, onde são abordados os seguintes temas: Trabalhadores do conhecimento, processos intensivos em conhecimento, gerenciamento de casos e uma comparação entre BPMN e CMMN para processos intensivos em conhecimento.

No capítulo 3, são apresentados exemplos do uso da ferramenta desenvolvida, assim como a interface, funcionalidades e explicação das tecnologias utilizadas.

O capítulo 4 apresenta a conclusão do trabalho e possíveis trabalhos futuros.

Capítulo 2

Fundamentação Teórica

2.1 Trabalhador do Conhecimento

Os trabalhadores do conhecimento (*knowledge workers*) realizam atividades que são difíceis ou impossíveis de serem automatizadas. Esses trabalhadores possuem maior autonomia e liberdade para tomar decisões baseadas no seu julgamento e aplicar caso a caso. Segundo Davenport, "trabalhadores do conhecimento são pessoas com alto nível educação, ou experiência, cuja principal função envolve a criação, distribuição ou aplicação de conhecimento." [11][12].

Professores são um exemplo de trabalhadores do conhecimento, cujo foco reside na distribuição do conhecimento a outras pessoas. O trabalho é transferir parte de seu conhecimento para seus alunos, mas como isso será realizado fica a cargo do professor. O professor tem um objetivo, mas não possui um passo a passo do que fazer. Cada professor tem a liberdade de ser criativo nos seus métodos e abordagens com alunos, vendo o que funciona para diferentes turmas ou alunos individuais, pois cada aluno tem seu tempo, suas dificuldades e facilidades. O trabalhador do conhecimento pode, por exemplo, ter que inovar e fazer algo que nunca foi feito antes.

Técnico de futebol é outra atividade intensiva em conhecimento. Um técnico deve ter conhecimento sobre o jogo em si, sobre sua equipe e as equipes adversárias. Ele trabalha com diversos outros profissionais experientes, como médicos, preparadores físicos, auxiliares, psicólogos para preparar seu time da melhor maneira possível.

É impossível dizer, passo a passo, como ele deve fazer o seu trabalho, ou o que deve fazer. Ele tem objetivos, marcos a serem atingidos, ele pode ser avaliado, mas cada treinador terá sua metodologia, seus princípios, sua maneira de lidar com os jogadores. Para cada partida, ele deve se adaptar ao contexto e tomar as decisões de quais jogadores utilizar, qual estratégia, que treinos específicos vai realizar para atingir seu objetivo.

Nos Estados Unidos, 48% dos trabalhadores podem ser considerados trabalhadores do conhecimento, Davenport estima que esse número esteja entre 25 e 50% nas economias desenvolvidas. Ainda mais relevante é o impacto desses trabalhadores e das empresas que possuem um maior nível de trabalho do conhecimento. Tanto esses trabalhadores, quanto essas empresas, são os que mais se destacam e geram mais valor para economia [13][11][14].

Muito dos trabalhos repetitivos já foram automatizados e alguns até eliminados. Os trabalhadores hoje gastam menos tempo em tarefas rotineiras e mais em tarefas que precisam pensar. Houve uma mudança de foco na natureza do trabalho e de como é gerenciado, o desafio de hoje é como suportar as atividades dos trabalhadores do conhecimento. Em alguns trabalhos dessa natureza, é até difícil delimitar o horário de trabalho, quando ele encerra, pois as ideias podem surgir a qualquer momento. Um funcionário de marketing pode passar o dia inteiro no seu escritório sem ter nenhuma ideia boa e essa ideia surgir no seu tempo livre.

A imprevisibilidade do trabalho do conhecimento não significa apenas que possui variabilidade. Muitos processos possuem variabilidade, mas não são intensivos em conhecimento, como a compra de passagens pela internet. Pois nesses casos, a variabilidade não altera a sequência de ações, o processo não é alterado [15]. A imprevisibilidade dos processos intensivos em conhecimento reside na impossibilidade de se prever que ações serão realizadas pelo trabalhador. Em cada caso, ele irá executar ações que julgar apropriadas para atingir seu objetivo, as tarefas podem não estar definidas, ele não tem um conjunto de tarefas da qual ele pode escolher.

"Imprevisível, significa que a sequência de ações relevantes dos trabalhadores não pode ser determinada *a priori* e o curso varia de caso para caso em maneiras inesperadas. Muitas vezes, o trabalhador tem que dar sequência ao trabalho mesmo tendo informações parciais e com chegada de novas informações ele pode ter que mudar os planos" [15]. Como o exemplo do professor, que tem que se adaptar as turmas, a reação dos alunos e as dificuldades de cada um e pode alterar completamente a sua maneira de dar aula. O trabalho é responsivo ao contexto, o trabalhador possui a discrição para tomar decisões não previstas.

A modelagem de processos possui diversos propósitos como: Entendimento do processo; design; treinamento e educação; simulação e otimização; suporte a execução de uma instância do processo. Modelar o trabalho como um processo bem definido, no qual existe uma clara sequência de tarefas tem sido uma abordagem muito utilizada para suportar, analisar e aperfeiçoar o trabalho realizado. O foco dessas abordagens é no que deve ser feito. Uma premissa muito aceita é que a qualidade do produto está correlacionada com a qualidade do processo. [11][16]

Essa visão tradicional de processos é difícil de ser implementada no universo dos trabalhadores de conhecimento, pois eles participam de atividades colaborativas, criativas, que envolvem tomada de decisões e não possuem uma estrutura bem definida. A sequência de eventos também não é bem definida, pode evoluir durante a execução e variar de caso a caso e até as tarefas não precisam ser bem definidas em tempo de design. O trabalhador tem autonomia para executar tarefas que não tinham sido previstas ou não executar algumas tarefas, baseado na sua experiência e julgamento do que precisa ser feito para atingir um objetivo. O foco é no que pode ser feito e não no que deve ser feito [11][12].

Apesar da dificuldade de se auxiliar os trabalhadores do conhecimento, segundo Davenport, é possível ter uma orientação a processos que seja benéfica aos trabalhadores de conhecimento, mantendo sua liberdade para ser criativo, tomar decisões e improvisar quando necessário. É importante também o envolvimento do trabalhador na abordagem de processos, pois esses trabalhadores não gostam que outros o digam como realizar seu trabalho. Um paradigma que auxilia esse trabalhador, pois

suporta processos flexíveis e intensivos em conhecimento é o de gerenciamento de casos [17]. O foco não deve ser na automatização, que pode ser impossível nesses casos, mas sim no suporte a tomada de decisões, na colaboração, no reúso, compartilhamento de conhecimento. [18][15]

Davenport classifica os trabalhos de conhecimento de acordo com o tipo de atividade de conhecimento e cada tipo tem uma possível orientação a processo:

- Criação: criação de conhecimento é visto como tarefa criativa, idiossincrática que é difícil de gerenciar em processos, mas é possível ter avanços. Uma abordagem usada por empresas nos anos 80 e 90 era dividir o processo em diversas fases. O objetivo era permitir a avaliação do novo conhecimento criado na transição de uma fase para outra (*stage gates*) para ver se determinados critérios eram atingidos, podendo ser descartada ou prosseguir para próximas fases. Muitas organizações de pesquisa usam esse modelo para processo de inovação, pois permite bastante liberdade dentro das fases.

- Distribuição: Profissões como professores, serviço ao cliente, jornalistas são alguns exemplos. Existem várias pesquisas que sugerem que grupos que compartilham conhecimento possuem performance melhor do que os que não compartilham. Gerenciar as circunstâncias pelas quais o conhecimento é transferido é mais importante do que gerenciar o processo em si.

- Aplicação: Engenharia, programação, médicos, contador são exemplos de profissões que aplicam o conhecimento para encontrar alguma solução. Nesses casos facilitar o reuso de soluções e conhecimento pode aumentar a performance desses trabalhadores. Na Engenharia de Software, reúso de software é uma área de pesquisa própria e traz muitos benefícios como a aderência a padrões e aumento da confiabilidade do produto, pois o software já foi usado e testado [11].

Ivan Marques também discute sobre importância do trabalho sobre a informação, o que ele chama de trabalho de "informacionalização", que são os trabalhos que geram mais valor na nossa sociedade, e que o número de postos de trabalho de "informacionalização" é grande e está concentrado nas sedes. Enquanto os trabalhos

repetitivos, que recebem menores salários, estão diminuindo com a automatização, que ele chama de trabalho de "materialização" [6].

Ivan Marques destaca a importância dos trabalhos de "informacionalização" para o Brasil e outros países que querem diminuir as diferenças para os países desenvolvidos. Ele define 3 tipos de investidura informacional: 1) Investidura de uso: Faz se uso da tecnologia, mas importando a concepção, projeto e os produtos. 2) Investidura de materialização: Parte do trabalho responsável por montar, fabricar um produto. 3) Investidura de virtualização: Trabalho sobre a informação, criação dos projetos, conceitos e dos processos de produção [6].

Com esses conceitos, ele mostra que, nos países sede, o principal tipo de investidura é o de virtualização, onde teria mais postos de trabalho, que geram mais valor e recebem melhores salários. E, nos países hospedeiros das transnacionais, o principal tipo de investidura é o de materialização.

Considerando a relevância do trabalhador do conhecimento para economia, destacada por Davenport e Drucker, um dos desafios do Brasil para diminuir sua dependência é de gerar postos de trabalho na área de virtualização ou do trabalho do conhecimento. [6]

2.2 Processos Intensivos em Conhecimento

DiCiccio definiu processos intensivos em conhecimento como "Processos intensivos em conhecimento são processos cuja condução e execução dependem fortemente dos trabalhadores de conhecimento realizando diversas tarefas interconectadas de decisão e intensivas em conhecimento. KIPs são genuinamente centrados em conhecimento, informação e dados e requerem uma flexibilidade considerável em tempo de design e execução" [1]

Di Ciccio classifica os processos de negócios ao longo de um espectro baseado na estrutura e previsibilidade do processo. Essa classificação pode ser vista na Figura 2.1 [1].

- Processos estruturados: Processos cuja sequência de atividades e exceções são conhecidas, todas as tarefas e decisões podem ser modeladas e previstas. Representa trabalhos rotineiros com pouca flexibilidade e evolução, as interações entre usuários e os eventos são conhecidos e controlados.

- Processos estruturados com exceções ad hoc: Similar aos processos estruturados, mas é um pouco mais flexível, pois pode ter ocorrência de exceções que não foram previstas.

- Processos não estruturados com segmentos predefinidos: Não se sabe a sequência exata de ações e tarefas que serão executadas, mas existem políticas e regulamentos que balizam a execução de alguns segmentos do processo. Esses trechos estruturados podem ter a forma de procedimentos prescritivos ou *templates* e diretrizes.

- Processos fracamente estruturados: Processos que não há uma prescrição do que deve ser feito e como, mas possui algumas regras e políticas que limitam a ação do usuário, quais comportamentos não são aceitos ou desejados.

- Processos não estruturados: Em grande parte, representados pelos processos intensivos em conhecimento no qual os trabalhadores do conhecimento confiam na sua experiência e conhecimento para atingir um determinado objetivo. São processos pouco previsíveis e com muita flexibilidade, conforme a execução do processo evolui e o contexto se altera, os trabalhadores tomam as decisões e realizam tarefas necessárias para levar o processo até o fim.

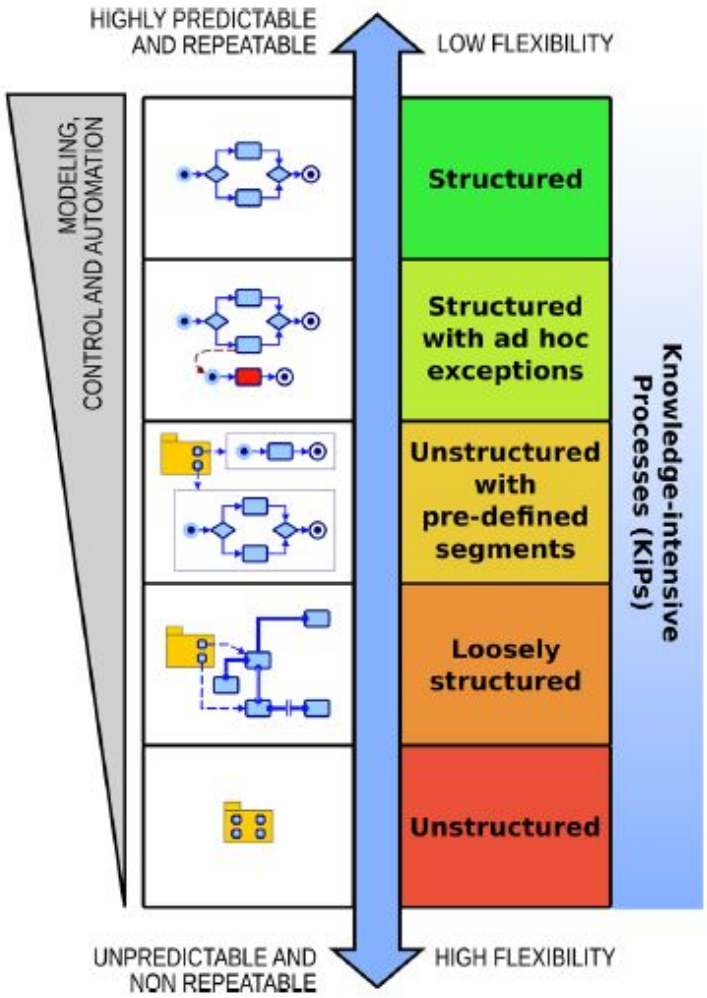

Figura 2.1: Classificação da estrutura de processos. Fonte: [1].

Uma abordagem tradicional para aumentar a competitividade é a adoção de um sistema de gestão de processos que permite a empresa definir, monitorar e melhor gerenciar seus processos. O BPM é uma dessas metodologias e é focado na melhoria continua dos processos [19][20][7].

Muitos processos por mais complexos que sejam, podem ser analisados e modelados em sua completude. São os processos bem estruturados, localizados no topo da Figura 2.1. Processos que possuem tarefas repetitivas, cujo fluxo de ações seja conhecido. A gestão de processos é, em grande parte, realizada sobre esses tipos de processos. Exemplos de processos estruturados incluem processos administrativos e de produção [1].

Já as atividades dos trabalhadores de conhecimento não possuem um fluxo bem definido de ações e não podem ser gerenciadas dessa maneira. A abordagem clássica ao BPM tem limitações que não o tornam adequado a esses processos, que envolvem um analista criando o modelo completo do processo. Isso leva a uma disparidade entre o que é modelado e o que é de fato executado. Também não é um ambiente apropriado para processos que mudam dinamicamente. Porém, a maturidade das metodologias de gerenciamento de processos com a inclusão da dimensão do conhecimento, da relevância do componente humano e suas interações fornece um suporte a esses cenários desafiadores. Assistência médica, gerenciamento de casos, pedido de seguro, governança de TI, marketing, planejamento estratégico são exemplos desses cenários que são processos não estruturados, não previsíveis e colaborativos. Esse tipo de trabalho tem quatro características: incerteza, orientação a objetivos, emergência, base crescente de conhecimento [1][20][21].

Com o crescente número de postos de trabalho intensivos em conhecimento e muitos serviços governamentais se tornando intensivos em conhecimento, existe a necessidade de readaptação organizacional que leve em conta as características desse tipo de trabalho. Esses trabalhadores sabem o que estão fazendo e há uma mudança do poder do gerente para o trabalhador do conhecimento. Aumentar a performance desses trabalhadores envolve uma série de questões: Criação de um ambiente colaborativo que facilite o compartilhamento de conhecimento; um ambiente onde exista confiança entre trabalhadores; preferencialmente uma organização menos hierárquica e mais horizontal; incluir os trabalhadores na modelagem do processo; linguagem de modelagem de processos que seja flexível; também deve-se considerar a dimensão do conhecimento, o fluxo do conhecimento nesses processos: como o conhecimento é criado, transformado, utilizado e compartilhado.[22][11][1].

DiCiccio define 8 principais características dos processos intensivos em conhecimento, que serão analisadas no contexto do exemplo utilizado do professor:

- Focado no conhecimento: A execução e progressão do processo depende do conhecimento do trabalhador e de suas interações com objetos de dados e conhecimento. O trabalho do professor é focado no conhecimento e motivado pelo desejo de transmitir parte desse conhecimento para seus alunos.

- Colaborativo: Normalmente, o processo ocorre num ambiente colaborativo no qual os participantes trocam e criam conhecimento e desempenham diferentes papéis. Davenport ressalta a importância de um ambiente propício à colaboração de conhecimento [11]. Os professores interagem com seus alunos, diretores e outros colegas e, para cada turma e escola, a interação será única, levando a aulas diferentes. Um professor certamente poderia se beneficiar da colaboração com outros colegas de metodologias que funcionaram, experiências diferentes e compartilhamento de notas e materiais.

- Imprevisível: Ao chegar a uma nova turma, mesmo com experiência, aulas planejadas e uma metodologia definida, a reação e a interação com aqueles alunos pode fazer com que professor se adapte, alterando sua aula. O professor pode escolher aulas não convencionais, ao ar livre, visita a museus, pontos turísticos. Essa característica representa que não se sabe ao certo que atividades serão executadas, como as coisas serão feitas.

- Emergente: Diretamente ligado a imprevisibilidade, o curso de ações é definido conforme a execução avança e mais informações estão disponíveis.

- Orientação à objetivos: Muitas vezes, o trabalhador só conhece seu objetivo e possivelmente algumas regras, procedimentos e recomendações de boas práticas. O processo pode ser avaliado, medido por marcos e objetivos intermediários, atingir ou não esses marcos pode determinar o restante do fluxo. Por isso, o foco no suporte a tomada de decisões é mais relevante que a automação. O professor tem um objetivo de transferir seu conhecimento a outros.

- Responsivo a eventos: Eventos alteram o contexto e, possivelmente, o fluxo de execução do processo. Esses eventos podem ser únicos e impossíveis de serem previstos, dependendo do trabalhador e sua autonomia para determinar o impacto do evento e o que deve ser feito. Esses eventos podem afetar as decisões do trabalhador.

- Direcionado por regras e limitações: Muitas vezes não se sabe como as coisas devem ser feitas, ou ter um passo a passo detalhado, mas possivelmente existem regras que limitam o que não pode ou não deveria ser feito. No caso

dos professores, agredir fisica ou verbalmente um aluno, certamente não seria adequado.

- Não repetível: A instância do processo é única, como não são processos estruturados, mas sim dependentes do conhecimento e autonomia das pessoas, a execução irá levar em conta as particularidades de cada profissional e de suas interações entre si. A maneira de lidar com o caso varia de trabalhador para trabalhador, levando em conta o conhecimento, experiência, personalidade de cada um. Dois professores dando aula para a mesma turma, no mesmo colégio, terão interações completamente distintas e, provavelmente, abordagens distintas.

Os conceitos de trabalhadores do conhecimento e processos intensivos em conhecimento estão intrinsecamente conectados em si e ao conceito de gerenciamento de conhecimento. O foco do conceito de trabalhador de conhecimento é nas pessoas, quais as características do seu trabalho, o que diferencia de outros trabalhadores e sua relevância para as organizações. Já na definição de processo intensivo em conhecimento, o foco é no processo em si, quais suas características e requisitos para que o suporte a esse ambiente seja mais produtivo.

A área de desenvolvimento de software é intensiva em conhecimento, pois é um esforço coletivo, criativo e complexo, dependente da experiência, conhecimento de seus participantes. Ela é dependente da interação dos indivíduos e organizações e do uso de tecnologia. "O processo de desenvolvimento de software tem sido considerado um sistema sociotécnico, no qual aspectos humanos e organizacionais possuem um papel fundamental e precisam ser suportados pela tecnologia de uma maneira que seja orientada ao humano e organizacional."[23][16].

O entendimento da importância da qualidade do processo para a qualidade final do software levou a criação de padrões e a busca pela melhoria contínua do processo. O CMMI é um guia que classifica o nível de maturidade dos processos. Nesse tipo de classificação, os níveis são: Inicial, gerenciado, definido, quantitativamente gerenciado e otimizado. O nível inicial reflete um processo ad-hoc e a empresa avança nos níveis quando atinge objetivos específicos em diversas áreas de processo,

na sua busca pela melhora contínua de seus processos para se obter processos bem caracterizados e entendidos, com métricas para qualidade e performance [24].

Existem disparidades entre o modelo e o processo que é de fato executado e o processo evolui e possui diversas razões para ser modificado. Portanto, um dos desafios da área de processo de software é ter linguagens de modelagem de processos flexíveis e tolerantes, que devem permitir a modelagem incompleta, informal e parcial com a possibilidade de melhorias incrementais. Essa necessidade se relaciona com as características e requisitos dos processos intensivos em conhecimento e o CMMN permite essa modelagem flexível e adaptação em tempo de execução. [16][25].

Levando-se em consideração a natureza intensiva em conhecimento do desenvolvimento de software, um sistema sociotécnico e colaborativo, o foco passa da automação e da tentativa de aderência rígida ao processo para suporte de decisões, colaboração entre trabalhadores de conhecimento e convergência do processo. É importante que as exceções possam ser monitoradas e gerenciadas.[1][16].

O desenvolvimento ágil na engenharia de software incorpora algumas dessas características de processos intensivos em conhecimento, focando na colaboração e auto-organização dos times de desenvolvimento e numa abordagem iterativa de desenvolvimento.

O desenvolvimento ocorre nas *sprints*, período de tempo, nas quais algum incremento ou funcionalidade deve ser produzida. A *sprint* tem objetivos, reuniões diárias e uma avaliação no fim da *sprint*. Uma *sprint* pode ser vista na Figura 2.2. Outra característica do desenvolvimento ágil é que o time de desenvolvimento é auto-organizável e pode atuar em diversas áreas, possuindo as habilidades necessárias para realizar o incremento[2].

O desenvolvimento ágil, portanto, tem objetivos e marcos, mas não tem fluxo completamente definido de tarefas. O time de desenvolvedores tem autonomia para se organizar e desempenhar suas funções.

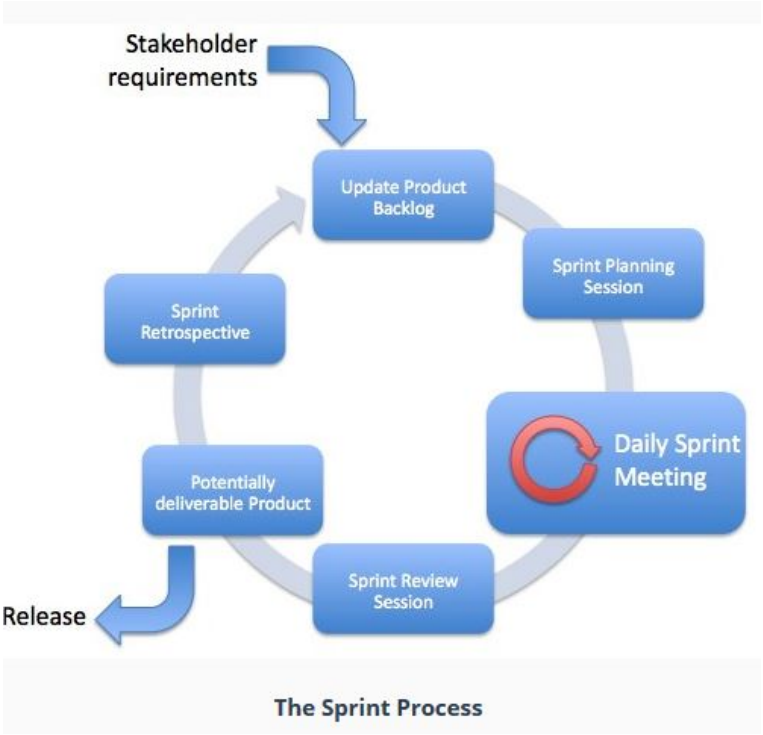

Figura 2.2: Scrum Sprint. Fonte: [2].

2.3 Gerenciamento de Casos

A gestão de processos tradicional, focado no gerenciamento de *workflows*, não é indicada para processos intensivos em conhecimento. Existe um outro paradigma que se mostra mais adequado para esses processos flexíveis e pouco estruturados. O gerenciamento de casos leva em consideração as características dos *KIPs*: flexibilidade; orientado a objetivos e foco no que pode ser feito pelos trabalhadores. O caso tem uma analogia com uma pasta na qual informações e o contexto são armazenados conforme evolução do caso, casos legais são bons exemplos. O papel do gerenciamento de casos é dar assistência a tomada de decisões e não guiar o trabalho, além disso, nesse paradigma os dados são cidadãos de primeira classe [3][17].

No passado, os casos eram pastas com informações que iam sendo compartilhada entre trabalhadores e departamentos. Esses trabalhadores se comunicavam para tomar as decisões e atingir seus objetivos, eles possuíam conhecimento das regras da empresa e e experiência com casos passados para tomar as decisões. Já o gerenciamento de casos atual procura dar um suporte tecnológico para esses trabalhadores.

[18]

"Um caso é um procedimento que envolve as ações realizadas em relação a um sujeito em uma determinada situação para atingir um objetivo[...] Os casos podem ser resolvidos de maneira completamente ad-hoc dependendo apenas do discernimento do trabalhador do conhecimento, mas o ganho de experiência permite que boas práticas e procedimentos possam ser definidos para se administrar um caso de forma mais eficiente" [3].

Outra definição dada por Marin é: "Gerenciamento de caso é uma prática para processos intensivos em conhecimento com o caso como principal repositório, onde o curso de ação para se atingir o objetivo é altamente incerto e a execução gradualmente emerge de acordo com a base de conhecimento disponível e a expertise do trabalhador de conhecimento." [26]

Essas definições ressaltam o suporte ao trabalhador do conhecimento e algumas características de kips definidas por DiCiccio, como a emergência, imprevisibilidade e orientação a objetivos. Como Davenport ressaltou, o auxílio na tomada de decisões e na colaboração são fatores mais importantes do que um guia ou busca por automação. O trabalhador tem conhecimento para decidir o que fazer naquelas circunstâncias para atingir seu objetivo e o gerenciamento de casos provê esse suporte aos trabalhadores, sem tirar sua autonomia.

Algumas características de gerenciamento de casos são: [18]

- Intensivo em conhecimento
- Variabilidade
- Complexidade de informação
- Colaboração
- Diversos participantes e papéis
- Inter relacionamento entre casos: um exemplo é a aplicação para cidadania de um indivíduo pode ser afetada pelo sucesso da aplicação de um parente.

O gerenciamento de workflows apresenta problemas conhecidos: são muito restritivos; pouco flexíveis, apresentam problemas para lidar com exceções inesperadas e a se adaptar a mudanças; o foco no fluxo de controle, deixando o contexto em segundo plano, isso é chamado de *context tunneling*. Uma pessoa com determinado papel só verá informações limitadas para uma determinada tarefa e somente as tarefas que foram designadas a ela, ignorando o resto das informações que podem ser relevantes. O trabalhador do conhecimento deve ter acesso a todo o contexto do caso. Devido às características de imprevisibilidade e emergência, os modelos de workflow não conseguem capturar todo o conhecimento dos experts e as atividades que devem ser executadas. O gerenciamento de casos evita alguns dos problemas do gerenciamento de *workflows* como o *context tunneling* [17][20].

O caso é o principal conceito do gerenciamento de casos e não as atividades ou o fluxo de sequência. Quais os objetivos e o contexto daquele caso, o que pode ser feito para atingir o objetivo dadas as circunstâncias e informações disponíveis. Gerenciamento de casos inclui a dimensão do conhecimento, os dados são cidadãos de primeira classe, podem ser editados e acessados a qualquer momento e desempenham papel importante no estado do caso. Os trabalhadores precisam lidar com fluxo de dados e conhecimento em suas duas dimensões: conhecimento explícito e conhecimento tácito [17][22].

O conhecimento explícito pode ser formalizado, definido e representado em documentos, diretrizes e outros artefatos. Esse conhecimento pode, portanto, ser facilmente distribuído, modificado e utilizado pelos trabalhadores do conhecimento. Já o conhecimento tácito dificilmente pode ser externalizado, ele é dependente do conhecimento, experiência, intuição dos indivíduos e, portanto, mais difícil de ser compartilhado. O trabalhador interage com ambos os tipos de conhecimento e ambos são essenciais na tomada de decisões. O conhecimento tácito está embutido nas ações dos trabalhadores do conhecimento [27][12].

Os dados que os trabalhadores interagem podem ser estruturados ou não estruturados. Nathaniel Palmer levanta a transição do mundo relacional, focado nas bases de dados em tabelas, para o mundo do *big data* e a explosão da computação na

nuvem. O gerenciamento de casos deve lidar com essas mudanças acessando dados de variadas fontes e dados não estruturados para fornecer a informação desejada no momento certo, facilitado o *insight* do trabalhador. Os dados não estruturados que capturam também a semântica e os metadados e não se prende a relacionamentos predeterminados [28].

Uma característica fundamental do gerenciamento de casos é a capacidade de se planejar em tempo de execução. Essa adaptabilidade e autonomia que são um dos grandes diferenciais do gerenciamento de casos. É necessário uma notação e modelagem flexível que não tire a autonomia do trabalhador do conhecimento. O trabalhador deve poder selecionar tarefas em tempo de execução, adicionar tarefas não previstas, colaborar com outros trabalhadores de maneira ad-hoc [3].

Um caso tem duas fases: fase de design, na qual a parte estruturada e conhecida do processo é modelada; fase de execução, na qual o trabalhador de fato realiza o trabalho, executando as tarefas na ordem que achar apropriado e aplicando, criando ou compartilhando conhecimento. Possivelmente adicionando tarefas que não tinham sido previstas, as tarefas discricionárias podem ser adicionadas pelos trabalhadores de acordo com seu discernimento [3]. As duas fases podem ser vistas na Figura 2.3.

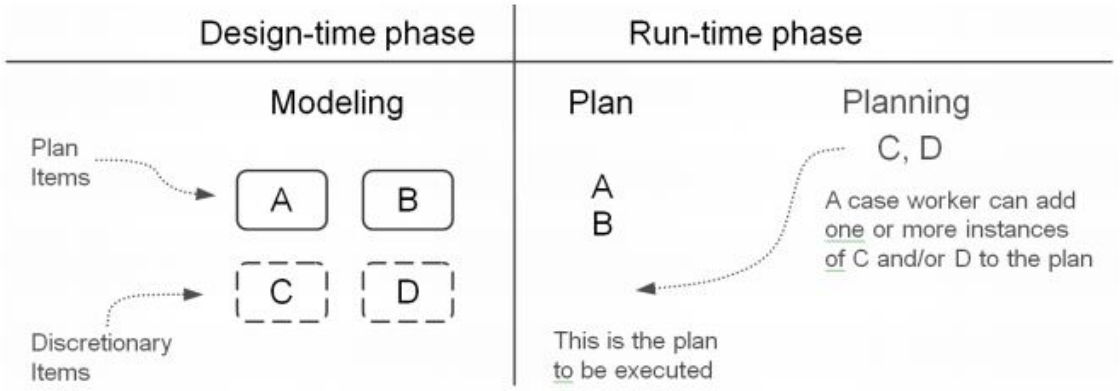

Figura 2.3: Planejamento nas fases de design e execução. Fonte: [3].

Swenson define duas abordagens para gerenciamento de casos: *Adaptive case management*; *production case management* [3][29]:

- *Adaptive Case Management* (ACM): Os próprios trabalhadores do conhecimento são responsáveis pela criação e aprimoramento do caso. Eles podem

criar e modificar o caso a qualquer momento. [29].

- *Production Case Management* (PCM): A grande distinção entre PCM e ACM é que ,no PCM, há separação entre a fase de design e a de execução. O trabalhador seleciona que tarefas executar, mas não tem autonomia de decidir que tarefas estarão disponíveis. É uma aplicação de gerenciamento desenvolvida por especialistas quando existem certos padrões na execução do caso.[29]

2.4 BPMN vs CMMN em Processos Intensivos em Conhecimento

BPMN e CMMN são complementares, uma vez que o BPMN é indicado para modelar processos estruturados enquanto o CMMN é adequado para processos flexíveis como KIPS. Os modelos de processo em BPMN são modelados inteiramente em tempo de design e são modificados entre uma execução e outra. O CMMN permite a adaptação em tempo de execução. BPMN foca na sequência de atividades e no que deve ser feito, CMMN foca no contexto do caso e no que pode ser feito [17]. Alguns problemas de se usar BPMN para modelar kips:

- BPMN introduz ordenamento das tarefas.
- Sem tarefas opcionais.
- Alternativas Limitadas
- Visão limitada dos dados(*context tunelling*)

[4]

Alguns desses problemas são eliminados com uso do BPMN estendido que tem elementos extras, diminuido a dificuldade do BPMN para suportar processos intensivos em conhecimento. Um exemplo é o elemento *Sub-Processos ad-hoc*, esse sub-processo é mais flexível pois as tarefas não são ordenadas nem obrigatórias, o trabalhador tem autonomia pra executar como quiser. Isso mitiga os problemas vistos anteriormente como ordenamento das tarefas, falta de tarefas opcionais. Porém

não permite a mesma expressividade que CMMN permite com seus decoradores. [4][30]

O problema de *context tunneling* persiste, o trabalhador continua tendo visão limitada dos dados e tarefas. Um outro problema é que o uso de sub-processos ad-hoc pode dificultar o entendimento do modelo e muitas engines ainda não o suportam [4].

CMMN foi criado para gerenciamento de casos e suporte a processos intensivos em conhecimento e os problemas citados para BPMN não ocorrem no CMMN. A flexibilidade de adaptação, planejamento em tempo de execução com seleção de tarefas e adição de tarefas discricionárias elimina os 3 primeiros problemas. O fato de ser orientado a objetivo e apresentar o contexto completo para trabalhador elimina o problema do *context tunelling* [4].

Existem processos que podem ser modelados tanto no BPMN quanto no CMMN, mas o quão adequada será essa modelagem, se será fácil de ler e entender, ou um modelo confuso dependerá da estrutura do processo. A grande diferença é que o BPMN é imperativa, apropriado para processos estruturados, enquanto que CMMN é declarativo, apropriado para processos intensivos em conhecimento [4]. Interessante notar que CMMN, por ser mais novo, possui compatibilidade com BPMN e pode ter tarefas que chamam modelos BPMN.

Devido aos motivos citados acima, a aplicação desenvolvida irá usar CMMN para modelar um processo intensivo em conhecimento, extrair suas regras, gerar um relatório de não conformidade e realizar *deploy* do processo.

Capítulo 3

KIP Plan Builder

3.1 Funcionalidades

O KIP Plan Builder é um plugin para Camunda, uma plataforma de gerenciamento de processos e *workflows*, que permite a identificação de não conformidades entre o plano de projeto e a definição de um processo.

O KIP Plan Builder trabalha em cima de processos não estruturados com segmentos predefinidos, sobre esses segmentos que o *Plugin* atua. Com o ganho de experiência, alguns procedimentos e boas práticas podem ser definidas mesmo em processos intensivos em conhecimento.

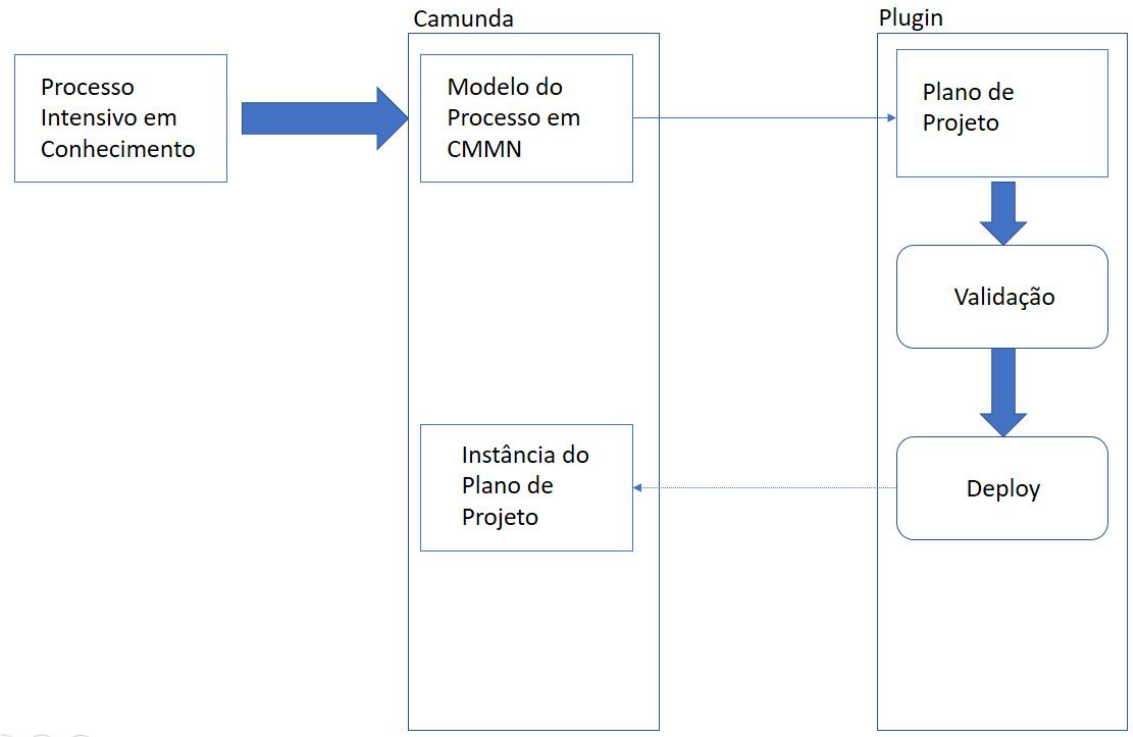

Figura 3.1: Wbs-Builder.

A Figura 3.1 mostra o comportamento geral do *Plugin* desenvolvido. Inicialmente, o usuário modela um processo intensivo em conhecimento usando CMMN e realiza *deploy* na engine do Camunda. Após isso, ele pode usar o *Plugin* para criar seu plano de projeto, esse plano tem como base uma determinada definição de processo. Dessa definição de processo são extraídas regras e um conjunto de tarefas das quais o usuário pode selecionar para criação das atividades do seu plano.

Depois da criação do plano de projeto, o usuário pode validar esse plano gerando o relatório de não conformidades. O *Plugin* faz um batimento entre as atividades planejadas pelo usuário e as regras extraídas da definição de processo. No final, o usuário pode fazer o *deploy* do plano de projeto para engine do camunda, criando uma instância do seu plano e gerenciar as tarefas que criou no Camunda. O *Plugin* apenas informa o trabalhador das não conformidades, não impedindo de que esse realize o *deploy* mesmo na existência de não conformidades. A autonomia continua com o trabalhador.

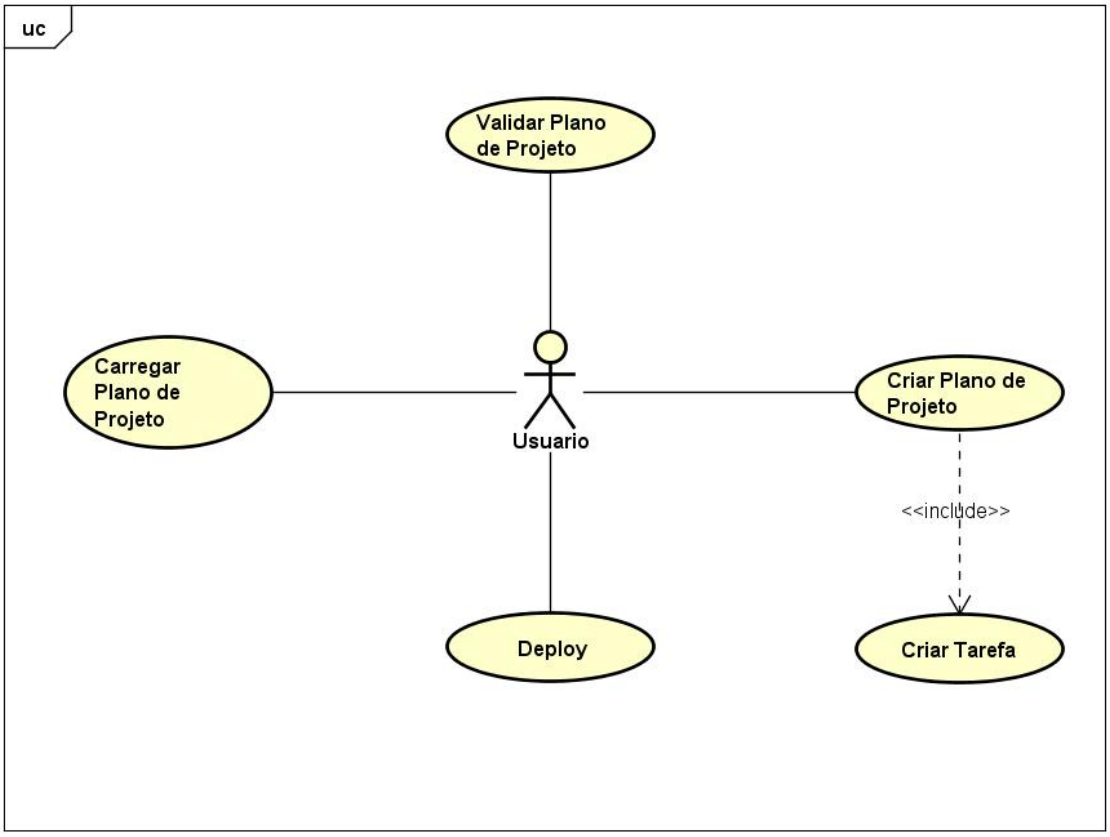

Figura 3.2: Diagrama de Casos de Uso.

Single Page Application é uma aplicação que interage com usuário atráves de uma única página em vez de carregar diversas páginas do servidor. O *Plugin* funciona como uma *Single Page Application*, toda interação ocorre na tela de *Cockpit* do Camunda.

O *Plugin* possui os seguintes casos de uso que podem ser vistos na Figura 3.2: Criar Plano de Projeto; Carregar Plano de Projeto; Criar e Remover Tarefas; Validar Plano de Projeto; *Deploy* do plano de Projeto.

- Criar Plano de Projeto: Na tela inicial, o usuário poderá criar um novo plano de projeto e associar esse plano à uma definição de processo. O usuário deverá informar o nome do plano e salvar. Esse caso de uso é demonstrado nas Figuras 3.4 e 3.6.

- Carregar Plano de Projeto: Na tela inicial, o usuário pode escolher um plano de projeto já definido de uma lista, esse caso de uso pode ser visto na Figura 3.5.

Cada quadrado apresenta um plano de projeto previamente criado contendo seu nome e o modelo do processo base. Quando usuário seleciona, esse plano é carregado e aberto para edição.

- Criar Tarefas: Com um plano de projeto carregado ou criado, o usuário seleciona "criar nova tarefa" que abre um popup com formulário. Nesse formulário o usuário precisa informar: 1) Nome da Tarefa; 2) Tipo da tarefa; 3) Data planejada de começo; 4) Data planejada de fim. Isso pode ser visto na Figura 3.8.

- Validar Plano de Projeto: Com um plano de processo carregado, o usuário pode validar esse plano. O sistema irá verificar as não conformidades e exibir um relatório para o usuário. As não conformidades que serão verificadas são as relativas a ordem das tarefas e a obrigatoriedade de tarefas.

- *Deploy* do plano de Projeto: Após validação, o usuário pode decidir fazer o *deploy* de seu plano de projeto para executar aquelas tarefas no servidor do Camunda.

Foi necessário estender o xml do CMMN com algumas propriedades para o funcionamento do plugin, um exemplo pode ser visto na Figura 3.3. Essas propriedades são utilizadas para associar o plano de projeto e as tarefas criadas pelo usuário com a definição de processo e as tarefas disponíveis. Cinco propriedades foram adicionadas:

- wbsbuilder:id: contém id da definição do processo do qual as regras e tarefas são extraídas.

- wbsbuilder:dataPlanejadaInicio: Data informada pelo usuário de quando a tarefa deverá ser começada.

- wbsbuilder:dataPlanejadaFim: Data informada pelo usuário de quando a tarefa deverá terminar.

- wbsbuilder:tipoTarefa: Contém o id da tarefa original da definição do processo base.

- wbsbuilder:nickname: Contém o nome da tarefa original da definição do processo base.

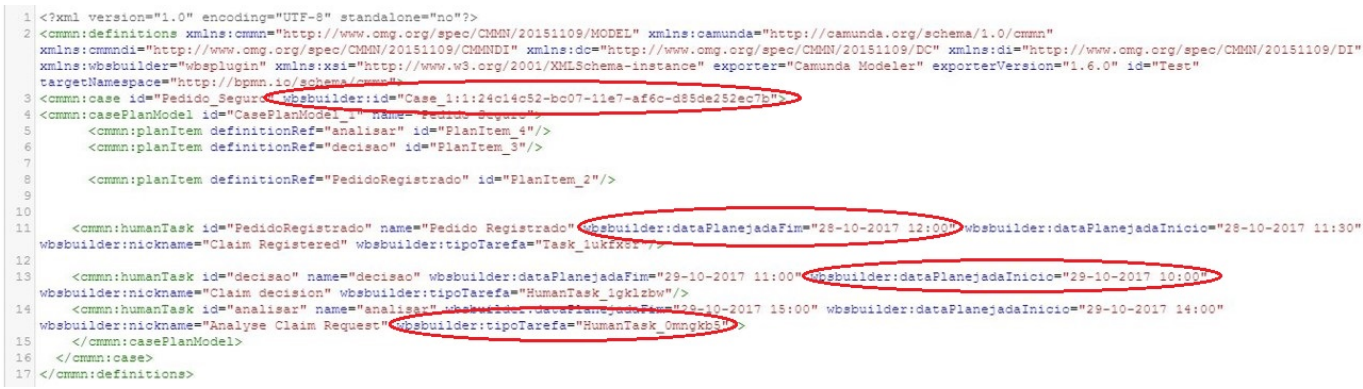

Figura 3.3: Extensão XML.

As duas não conformidades tratadas pelo KIP Plan Builder são:

- Obrigatoriedade: As tarefas no CMMN podem ser marcadas como obrigatórias. A ausência de uma tarefa obrigatória no plano do trabalhador é uma não conformidade.

- Ordem: No CMMN, algumas tarefas podem ter critérios de entrada, isso indica relação de dependência entre tarefas. Uma determinada tarefa só fica disponível para execução quando sua tarefa predecessora atinge um estado final. É entendido como não conformidade quando uma tarefa é planejada para começar antes de sua tarefa predecessora ter terminado ou quando a tarefa predecessora não é planejada.

A extração das regras da definição do processo base é feita de duas maneiras. As tarefas obrigatórias são buscadas pelos *plan Itens* que contém a *tag* "*cmmn:RequiredRule*" e as dependências entre algumas tarefas são buscadas pelos *sentries*. A explicação da notação CMMN pode ser encontrada no Anexo A. No final da extração das regras, o plugin possui uma lista com os ids das tarefas obrigatórias e um dicionário de dependências, onde a chave é o id da tarefa que possui tarefas predecessoras e o valor é uma lista dos ids das tarefas predecessoras.

Para gerar o relatório, o plugin faz o batimento entre as regras extraídas e as tarefas planejadas pelo usuário. Usando a propriedade wbsbuilder:tipoTarefa, o plugin consegue identificar que tarefas da definição original foram planejadas. O programa faz uma iteração sobre a lista de tarefas obrigatórias e procura por elas

na lista de tarefas planejadas, se alguma não for encontrada, uma não conformidade é identificada.

Para verificar as não conformidades de ordem, o programa examina o dicionário de dependências. Para cada tarefa que possui predecessores, *Plugin* verifica se essa tarefa foi planejada e se tiver sido planejada ele verifica se seus predecessores foram planejados. Usando as informações wbsbuilder:dataPlanejadaInicio e wbsbuilder:dataPlanejadaFim, o plugin consegue identificar a ordem planejada das tarefas e se há alguma inconsistência com a definição.

3.2 Interface

A aplicação Camunda possui diversos pontos de extensão, de forma a facilitar o desenvolvimento de *plugins*. Um desses pontos fica localizado na tela inicial de seu *cockpit*, onde está o *plugin* desenvolvido nesse trabalho. Na Figura 3.4, o *plugin* está com borda azul para identificação.

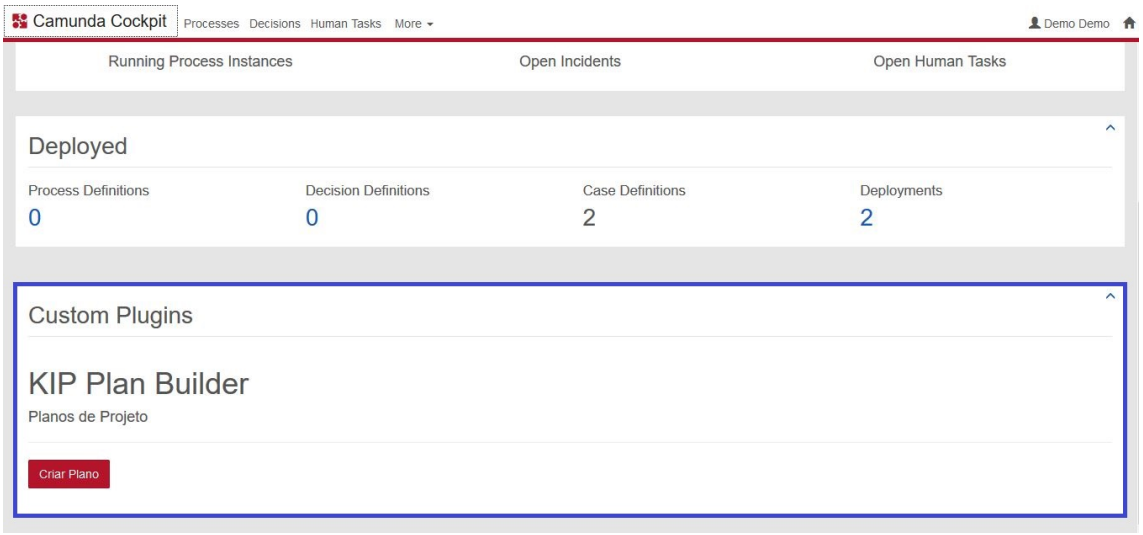

Figura 3.4: Tela Inicial.

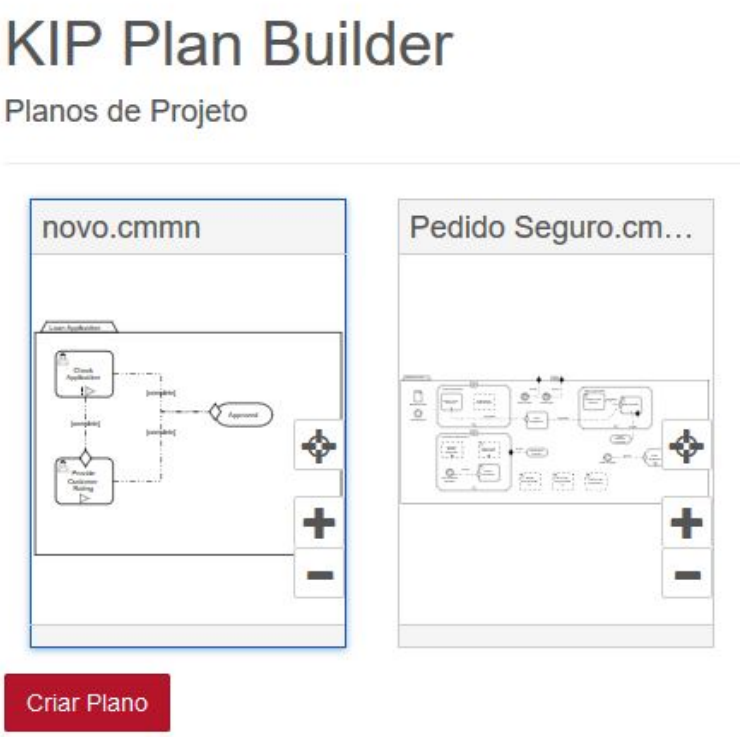

Figura 3.5: Planos na Tela Inicial.

Na tela representada pela Figura 3.5, o usuário pode acessar os planos de projeto previamente criados, ou criar um novo, usando *popup* e selecionando qual a definição de caso que será a base como vista na Figura 3.6.

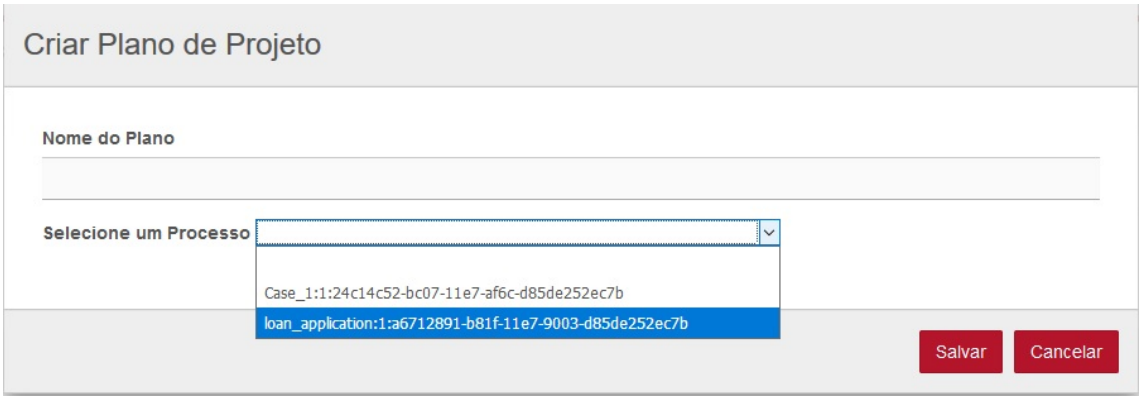

Figura 3.6: Criação de Plano.

Ao criar ou selecionar um plano, a tela mostra novas opções que podem ser vistas na Figura 3.7. Nessa tela, o usuário pode criar e deletar tarefas, gerar o relatório de não conformidades e fazer o *deploy* do seu plano de projeto para executar as tarefas

que ele criou. O *popup* para criar tarefas pode ser visto na Figura 3.8.

Figura 3.7: Menu Criação de tarefas

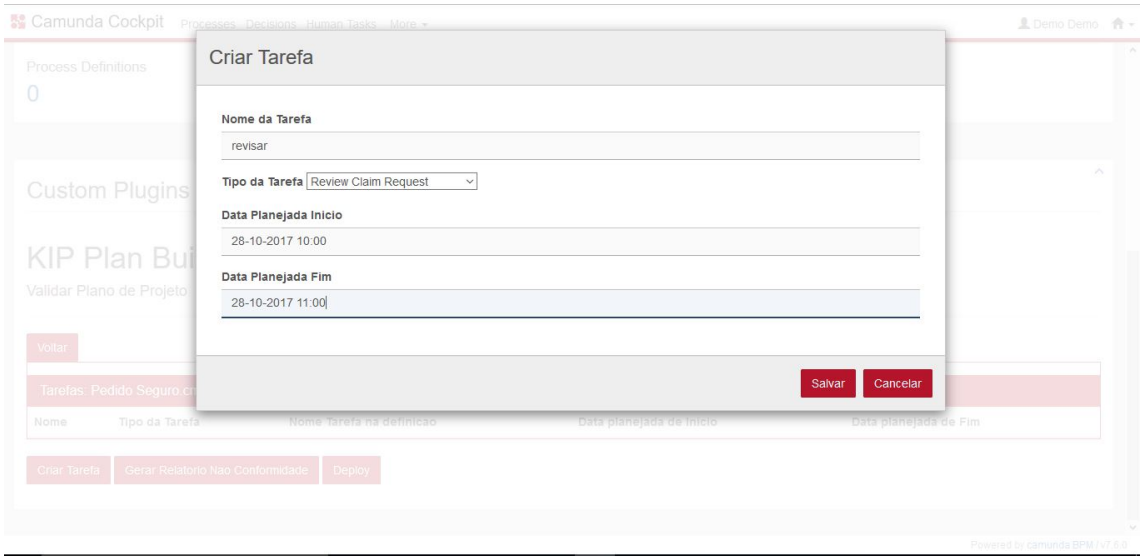

Figura 3.8: Popup Criação de tarefas

3.3 Camunda

Camunda é uma plataforma *open source*, desenvolvida em Java e Angular, para gerenciamento de processos e *workflows* com suporte para BPMN, CMMN E DMN. O usuário pode fazer o *deploy* de processos e aplicações para o servidor do Camunda e, a partir disso, gerenciar esse processo pelo Camunda. Ele pode criar diversas instâncias do processo, executar as atividades e monitorar o processo.

Camunda oferece duas maneiras de interagir com sua engine: API Rest, que faz uso de métodos HTTP como *GET*, *POST*, *DELETE* para interagir com a Engine

do Camunda e retorna uma mensagem indicando se a requisição teve sucesso ou não. Pode ainda retornar mensagem em um formato como xml, ou json, com propriedades relevantes, como o id do processo.

Api Java, que é o método mais utilizado para acessar a engine. O plugin desenvolvido utiliza as duas apis, mas majoritariamente faz chamadas a API Java. A engine oferece diversos serviços para interagir com os modelos e processos que estão rodando em seu servidor. Os serviços relevantes e usados no desenvolvimento do KIP Plan Builder foram:

- *RepositoryService*: Serviço para interagir com as definições de processos e os deployments, vários recursos dos processos podem ser acessados, como suas propriedades, modelo xml, a representação em classe java do processo. Esse serviço também permite realizar o *deployment* de definições.
- *RuntimeService*: Serviço responsável pelas instâncias dos processos e onde pode iniciar novas instâncias. Quando a instância é terminada, ela não pode mais ser acessada por esse serviço, terá de ser acessada no *HistoryService*.

Camunda oferece diversos pontos de *plugin*, onde é possível acrescentar funcionalidades. Esses pontos são predefinidos e o plugin deve indicar onde ele irá aparecer. Um *plugin* é um componente de software para adicionar uma funcionalidade a um programa sem precisar alterar o programa principal. O programa principal é independente dos *plugins* e oferece serviços para o plugin consumir e uma maneira para plugin se registrar. O *plugin* desenvolvido estende a funcionalidade do Camunda, permitindo que usuário valide seu plano de projeto em cima das regras de uma definição de caso (CMMN). Após a validação, o sistema exibe um relatório de não conformidades e o usuário pode fazer o *deploy* desse plano de projeto.

O *plugin* é um projeto Maven .jar, que precisa ser incluído nas bibliotecas de dependência das aplicações do *Cockpit*. No lado do servidor, pode se usar classes JAX-RS (Java API for RESTful web services) para estender a api do Camunda e expor os dados e serviços para o cliente. No lado do cliente, será usado Html e

AngularJS para exibir páginas customizadas e serviços. O *plugin* precisa ser registrado em um documento para ser reconhecido pelo *Cockpit*. Mais detalhes sobre o desenvolvimento estão no Anexo B.

3.4 Resultados

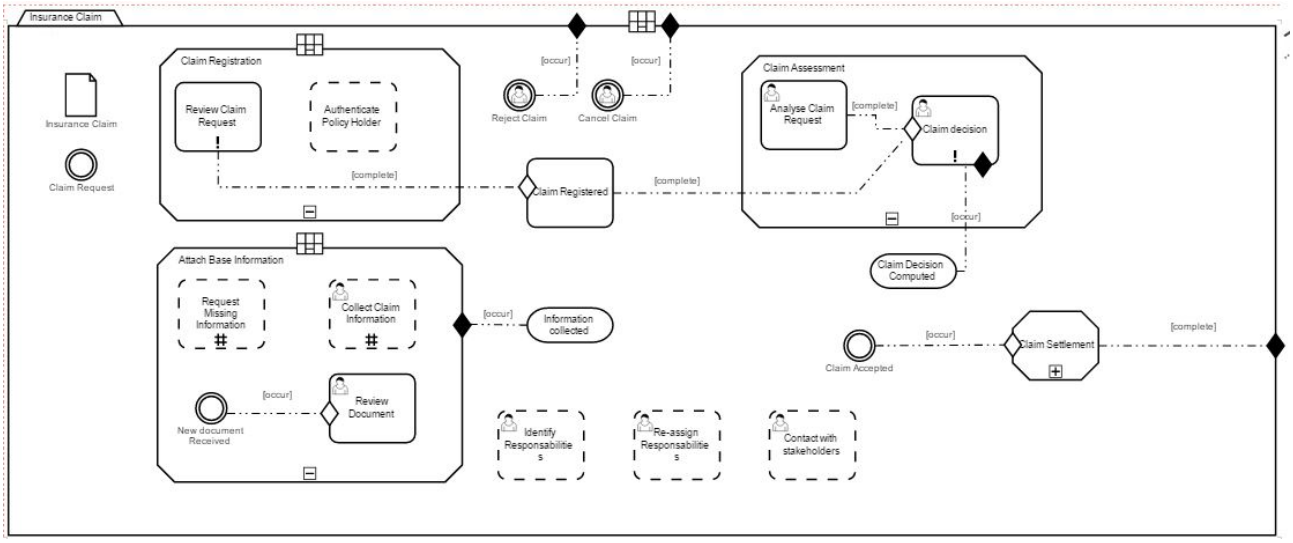

Figura 3.9: Modelo de Pedido de seguro. Adaptado de:[4].

A Figura 3.9 mostra o modelo CMMN escolhido para realizar os testes e demonstrar o funcionamento do plugin. É um processo de pedido de seguro, um processo intensivo em conhecimento, e possui duas tarefas obrigatórias: *claim decision*, *review claim request*. O modelo também apresenta dependência entre algumas tarefas: *claim Registered* depende de *review claim request*; *Claim decision* depende mutuamente de *analyse claim request* e *claim registered*.

Inicialmente, o usuário irá criar um novo plano de projeto utilizando o popup da Figura 3.6 e associar esse plano a definição do processo de pedido de seguro. Após essa criação, o usuário pode utilizar o popup de 3.8 para planejar a execução de suas tarefas.

Dois exemplos foram executados em cima dessa definição do pedido de seguro exibido na Figura 3.9. Para o caso 1 foram criadas quatro tarefas que podem ser

vistas na Figura 3.10. As duas tarefas obrigatórias foram planejadas: *claim decision*, *review claim request*. Porém existe uma não conformidade de ordem. A tarefa decisão(*Claim decision*) é planejada para execução anteriormente a tarefa de analisar(*analyse claim request*). O relatório de não conformidades pode ser visto na Figura 3.11.

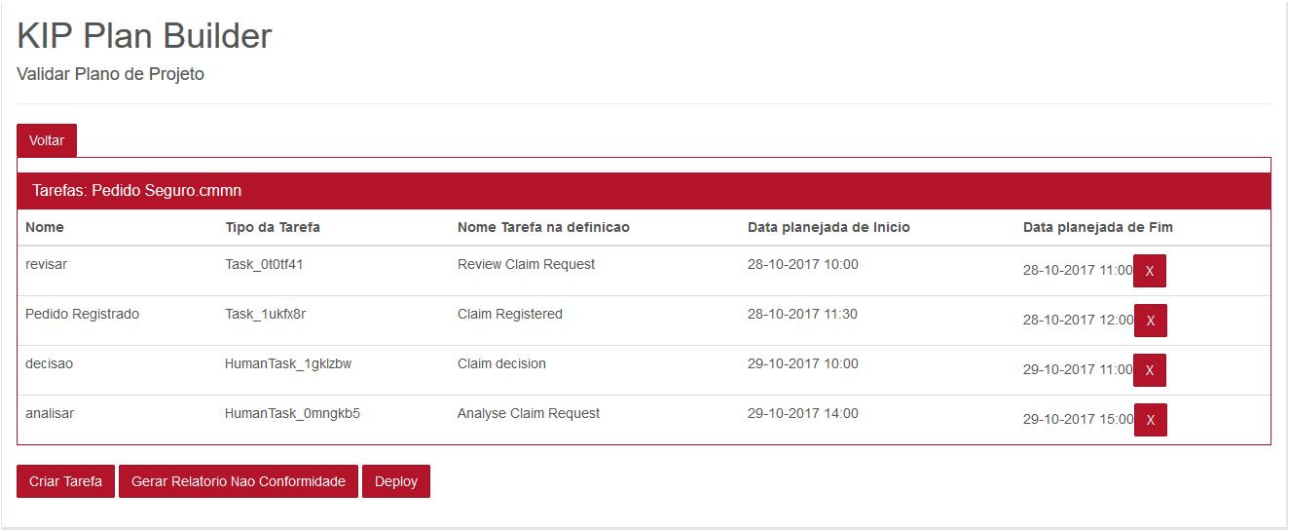

Figura 3.10: Caso 1.

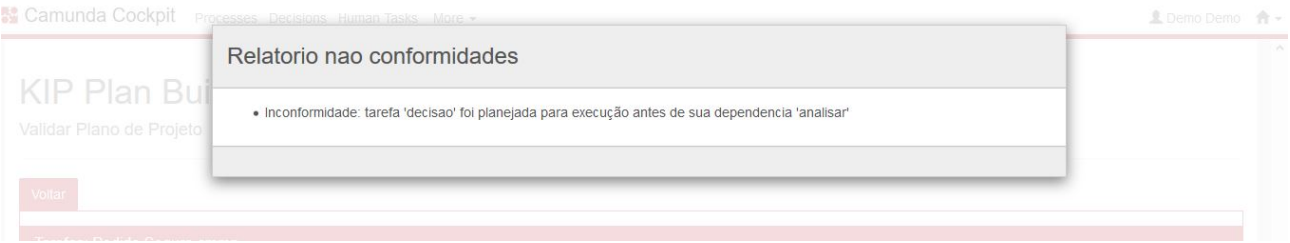

Figura 3.11: Inconformidades Caso 1.

O *Plugin* examina o dicionário de dependências, cujo conteúdo do caso 1 é exibido na Tabela 3.1.

Tabela 3.1: Conteúdo Dicionário de Dependências

Chave	**Valor**
Claim Registered	*[Review Claim Request]*
Claim Decision	*[Analyse Claim Request, Claim Registered]*

Primeiro, o *Plugin* irá verificar para tarefa *Claim Registered* se a sua tarefa predecessora foi executada. No exemplo do Caso 1, a tarefa *Review Claim Request* será encontrada e então ira verificar a ordem planejada. *Review Claim Request* foi planejada para terminar as 28/10/2017 as 11:00 e *Claim Registered* só foi planejada para começar na mesma data as 11:30. Portanto, nenhuma não conformidade identificada até então.

O mesmo procedimento é realizado para a segunda chave, representada pela tarefa *Claim Decision*. Porém aqui existe uma não conformidade: A tarefa *Claim Decision* tem seu inicio, 29/10/2017 10:00, anterior ao término da tarefa que deveria ser sua predecessora, *Analyse Claim Request* só termina as 29/10/2017 15:00. Nesse caso específico, *Claim Decision* é completada antes do início da sua dependência, como pode ser visto na linha temporal exibida na Figura 3.12

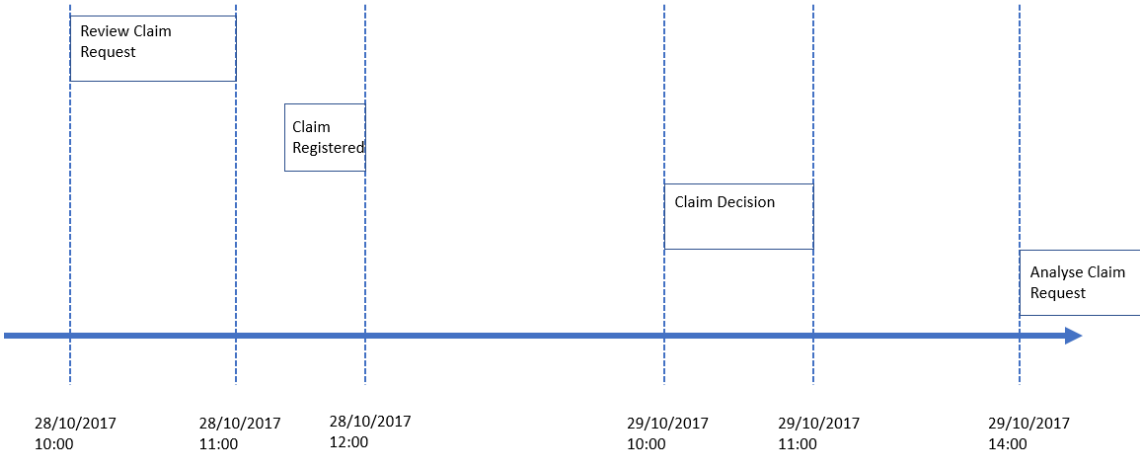

Figura 3.12: Linha Temporal Caso 1.

O caso 2 possui 3 tarefas que podem ser vistas na Figura 3.13. Nesse exemplo, as tarefas obrigatórias não foram planejadas e, portanto, existem duas não conformidades. As inconformidades são exibidas na Figura 3.14.

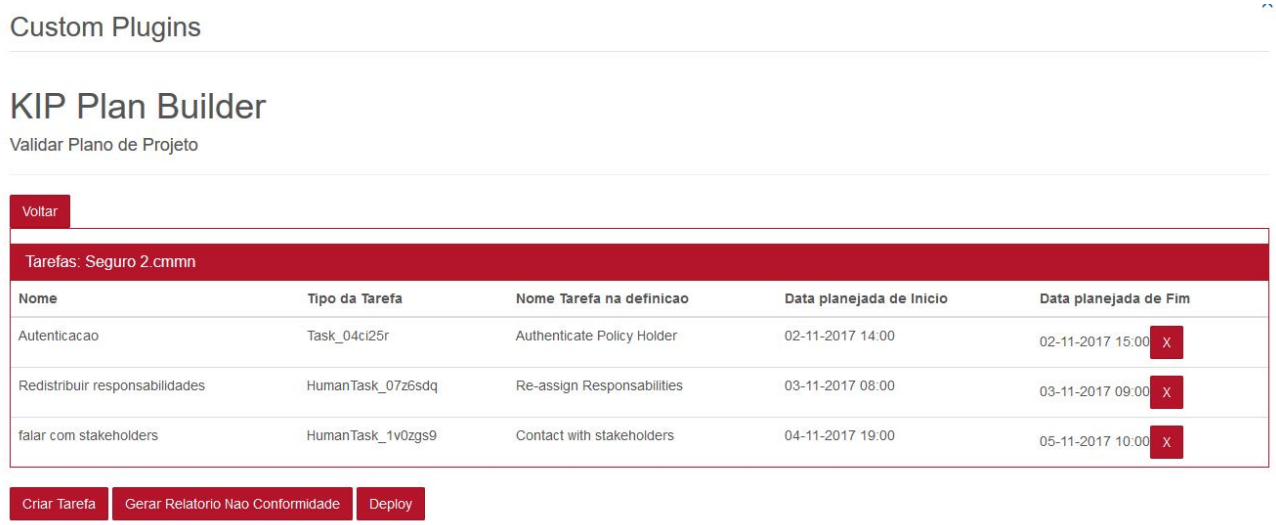

Figura 3.13: Caso 2.

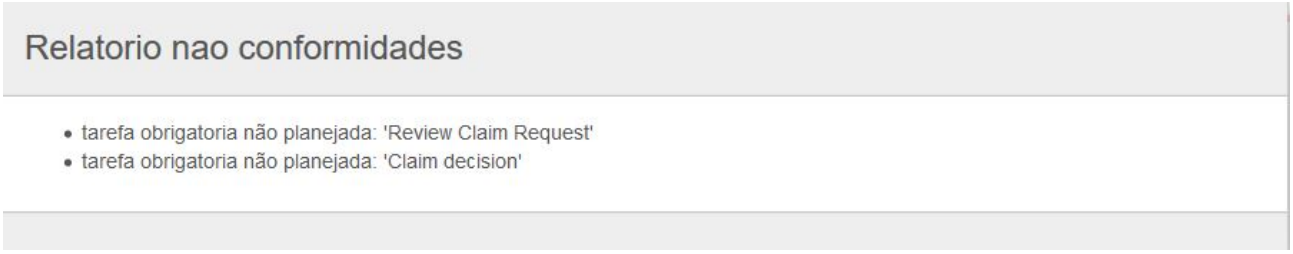

Figura 3.14: Inconformidades Caso 2.

O programa percorre a lista de tarefas obrigatórias (*claim decision*, *review claim request*) e procura no plano de projeto. No plano ele encontra as seguintes tarefas: *Authenticate Policy Holder*, *Re-assign Responsabilities*, *Contact with Stakeholders*. Como ele não encontra as tarefas obrigatórias no plano de projeto, ele identifica essas duas não conformidades.

Esses dois estudos de caso exemplificam os tipos de não conformidades identificadas pelo KIP Plan Builder desenvolvido nesse trabalho.

Capítulo 4

Conclusão

O trabalhador do conhecimento desempenha um papel importantíssimo na sociedade, por isso existe a necessidade de fornecer suporte a esse trabalho intensivo em conhecimento. Esta não é uma tarefa fácil devido às características dos processos intensivos em conhecimento, mas existem várias propostas para realizar isso. Gerenciamento de casos é um paradigma poderoso e um das principais soluções usadas para suportar kips, CMMN é uma linguagem para modelar casos.

Um ambiente de suporte aos trabalhadores de conhecimento deve focar em facilitar a criação, organização, reuso e compartilhamento de conhecimento entre os trabalhadores, deve fornecer contexto e informações relevantes para auxiliar a tomada de decisões e criação de conhecimento. Deve ser um ambiente adaptável no qual o modelo do processo possa ser alterado em tempo de execução pelos trabalhadores que possuem autonomia e conhecimento para isso.

Apesar dessa flexibilidade e emergência, é possível com tempo e experiência criar procedimentos, boas práticas e regras para fragmentos do processo. É importante que o trabalhador possa identificar as não conformidades sobre esses fragmentos para saber se seu plano está de acordo com as regras do processo, se ele está seguindo as boas práticas.

O KIP Plan Builder auxilia nessa identificação de não conformidades, sem tirar a autonomia do trabalhador. Ele pode, mesmo com não conformidades, realizar o deploy e gerenciar seu processo no Camunda se decidir que é adequado. O KIP Plan

Builder apenas informa o trabalhador, a decisão continua em suas mãos.

Os processos intensivos em conhecimento são sistemas sociotécnicos, centrados em pessoas, e portanto além das considerações tecnológicas, um ambiente ideal deve levar em consideração fatores da organização e relacionamentos pessoais.

Foi decidido criar um Plugin para Camunda, pois o Camunda é uma plataforma open source, bem estabelecida e com uma comunidade bem participativa. Criar um Plugin, em vez de criar uma aplicação do zero, trouxe muitos beneficios: Acesso aos serviços do Camunda de gerenciamento de processos; Maior possibilidade dessa ferramente ser utilizada ou estendida por outras pessoas.

KIP Plan Builder foi desenvolvido, principalmente, em Java e Angular. Tem em torno de 1500 linhas de código e é exibido na tela de Cockpit do Camunda. O Plugin trabalha sobre definições de casos em CMMN e trata dois tipos de não conformidades, obrigatoriedade e ordem das tarefas, e exemplos podem ser vistos na Seção 3.4.

Uma proposta de trabalho futuro é a possibilidade do usúario usar várias definições de processos diferentes para criar seu plano de projeto e gerar o relatório de não conformidades nesse ambiente multiprocessos. Outra melhoria seria a possibilidade do usuário modelar outros elementos, além de tarefas como marcos, sentries, etc.

Referências Bibliográficas

[1] DI CICCIO, C., MARRELLA, A., RUSSO, A., "Knowledge-Intensive Processes: Characteristics, Requirements and Analysis of Contemporary Approaches", *Journal on Data Semantics*, v. 4, n. 1, pp. 29–57, Mar 2015.

[2] SCHWABER, K., SUTHERLAND, J., "The Scrum Guide", 2001.

[3] OMG, "Case Management Model and Notation V1.1", http://www.omg.org/spec/CMMN/1.1, 2016, (Acesso em 28 Agosto 2017).

[4] BUKHSH, Z. A., *BPMN Plus: A Modelling Language for Unstructured Business Processes*. M.Sc. dissertation, University of Twente, Agosto 2015.

[5] CAMUNDA, "CMMN 1.1 Implementation Reference", https://docs.camunda.org/manual/7.7/reference/cmmn11, (Acesso em 20 Agosto 2017).

[6] MARQUES, I., *Brasil e a Abertura dos Mercados, O*. CONTRAPONTO, 2002.

[7] EUAX, "O que é BPM (Business Process Management)?", http://www.euax.com.br/2015/06/bpm-business-process-management-o-que-e/, Junho 2015, (Acesso em 28 Agosto 2017).

[8] HOBERECHT, B., "The Importance of a Project Plan", http://www.pinnacleprojects.com/index.php/the-project-plan-sp-1739577267, (Acesso em 5 Dezembro 2017).

[9] DRUCKER, P. F., *Age of Discontinuity*. Butterworth-Heinemann Ltd, 1969.

[10] "Camunda", https://camunda.org/, Acesso em 10 Agosto 2017.

[11] DAVENPORT, T. H., *Handbook on Business Process Management 1. International Handbooks on Information Systems*, chapter Process Management for Knowledge Work, Berlin, Springer, pp. 17–35, 2010.

[12] GRONAU, NORBERT, *Modeling and Analyzing knowledge intensive business processes with KMDL, Comprehensive insights into theory and practice*. GITO mbH, 2012.

[13] DAVENPORT, T. H., *Thinking for a Living: How to Get Better Performances And Results from Knowledge Workers*. Harvard Business Review Press, 2005.

[14] ZUMBRUM, J., "The Rise of Knowledge Workers Is Accelerating Despite the Threat of Automation", https://blogs.wsj.com/economics/2016/05/04/the-rise-of-knowledge-workers-is-accelerating-despite-the-threat-of-automation/, Maio 2016, (Acesso em 28 Agosto 2017).

[15] SWENSON, K. D., "White Paper State of the Art In Case Management", http://www.omg.org/spec/CMMN/1.1, MarÃ§o 2013, (Acesso em 28 Agosto 2017).

[16] FUGGETTA, A., "Software Process: A Roadmap". In: *Proceedings of the Conference on The Future of Software Engineering*, ICSE '00, pp. 25–34, New York, NY, USA, 2000.

[17] VAN DER AALST, W. M. P., WESKE, M., "Case Handling: A New Paradigm for Business Process Support", *Data Knowl. Eng.*, v. 53, n. 2, pp. 129–162, May 2005.

[18] WHITE, M., "Case Management: Combining Knowledge With Process", https://www.bptrends.com/case-management-combining-knowledge-with-process/, 2009, (Acesso em 20 Agosto 2017).

[19] DAVENPORT, T. H., *Process Innovation-Reengineering Work Through Information Technology,*. Harvard Business School Pres, 1993.

[20] STAVENKO, Y., KAZANTSEV, N., GROMOFF, A., "Business Process Model Reasoning: From Workflow to Case Management", *Procedia Technology*, v. 9,

n. Supplement C, pp. 806 – 811, 2013. CENTERIS 2013 - Conference on ENTERprise Information Systems / ProjMAN 2013 - International Conference on Project MANagement/ HCIST 2013 - International Conference on Health and Social Care Information Systems and Technologies.

[21] MARJANOVIC, O., FREEZE, R., "Knowledge Intensive Business Processes: Theoretical Foundations and Research Challenges". In: *2011 44th Hawaii International Conference on System Sciences*, pp. 1–10, Jan 2011.

[22] MLADKOVA, L., "Knowledge Management for Knowledge Workers", *Eletric Journal of Knowledge Management*, v. 9, pp. 248–258, 2011. Available at: www.ejkm.com.

[23] FUGGETTA, A., DI NITTO, E., "Software Process". In: *Proceedings of the on Future of Software Engineering*, FOSE 2014, pp. 1–12, New York, NY, USA, 2014.

[24] TEAM, C. P., *CMMIÂ® for Development, Version 1.3*, Report, Carnegie Mellon University, Nov. 2010.

[25] MONTANGERO, C., *The Software Process: Modelling and Technology*, Berlin, Heidelberg, Springer Berlin Heidelberg, pp. 1–13, 1999.

[26] MARIN, M. A., HAUDER, M., MATTHES, F., *Case Management: An Evaluation of Existing Approaches for Knowledge-Intensive Processes*, Cham, Springer International Publishing, pp. 5–16, 2016.

[27] GRUDZIÅSKA-KUNA, A., "Supporting knowledge workers : case management model and notation (CMMN)", *Information Systems in Management*, v. Vol. 2, No. 1, pp. 3–11, 2013.

[28] PALMER, N., *Case Management Megatrends*, Future Strategies Inc., pp. 101–109, 2012.

[29] SWENSON, K. D., *Case Management: Contrasting Production vs. Adaptive*, Future Strategies Inc., pp. 109–117, 2012.

[30] DIAS, F., "BPMN: Modelando processos de negócio com elementos avanccados (Parte II)", http://blog.iprocess.com.br/2013/01/bpmn-modelando-processos-de-negocio-com-elementos-avancados-parte-ii/, Janeiro 2013, (Acesso em 28 Agosto 2017).

[31] BREITENMOSER, R., KELLER, T., "Case Management Model And Notation - A Showcase", *European Scientific Journal*, v. 11, 2015.

[32] MARIN, M. A., "The Case Management Model and Notation (CMMN) version 1.0", http://cmmn.byethost4.com, 2016, (Acesso em 20 Agosto 2017).

[33] SIGNAVIO, "Signavio Process Manager User Guide", https://docs.signavio.com/userguide/editor/en/SignavioUserManual.pdf, (Acesso em 20 Agosto 2017).

[34] HINKELMANN, K., "Case Management Model And Notation", Slideshow, (Acesso em 20 Agosto 2017).

[35] ORACLE, "Creating Extensible Applications", https://docs.oracle.com/javase/tutorial/ext/basics/spi.html, (Acesso em 10 Outubro 2017).

[36] HADLEY, M., "The Java API for RESTful Web Services (JAX-RS) – Rapidly Build Lightweight Web Services", http://www.oracle.com/technetwork/articles/java/jax-rs-159890.html, Julho 2010, (Acesso em 10 Outubro 2017).

[37] ORACLE, "The Java EE 6 Tutorial", https://docs.oracle.com/javaee/6/tutorial/doc/gilik.html, 2013, (Acesso em 10 Outubro 2017).

Apêndice A

CMMN

A OMG (Object Management Group) foi fundada em 1989 e é uma organização internacional que aprova padrões abertos para aplicações orientadas a objeto. OMG foi responsável pela especificação do BPMN e lançou a primeira especificação CMMN em maio de 2014.

BPMN(Business Process Model and Notation) foi adotado como padrão para modelagem de processos de negócios. Esses Modelos são bons para processos predefinidos, completamente especificados e repetíveis, ou seja, processos bem estruturados que possuem uma sequência bem definida de tarefas.

Alguns processos não são bem definidos, como os processos intensivos em conhecimento, e a necessidade de modelar essas atividades que se adaptam às circunstâncias e dependem das decisões dos trabalhadores de conhecimento gerou a especificação do CMMN.

O CMMN contém menos elementos, que podem ser vistos na Figura , e é mais simples que BPMN. Além disso tem outro foco, CMMN foca no suporte a processos não estruturados e BPMN é mais adequado a processos bem estruturados.

O caso (Case) é um conceito de alto nível que combina todos elementos que constituem o modelo do caso. É analogo a uma pasta, que contém todo o contexto da execução. O *case file* é um container para os objetos de dados que os trabalhadores criam e editam que no CMMN são representados pelo *case file items*. [31].

O comportamento completo do caso sendo modelado é descrita no *case plan*, mas nem todo trabalho será modelado, a colaboração do trabalhador e sua interação com objetos de dados. Dados podem ser editados, adicionados e removidos a qualquer momento desde que trabalhador tenha permissão. *Case plan items* são os elementos do *case plan* e podem ser: *tasks*, *stages*, *milestones*, *event listeners* [32].

Stage é uma forma de agrupamento para melhor organização do modelo e pode conter outros *plan items* como *tasks* ou outro *stage*. O *case plan* é um *stage* mais abrangente do modelo [5]. "São containers similares a sub-processos[...]. São usados para gerenciar a complexidade do modelo decompondo em partes trabalháveis" [32]

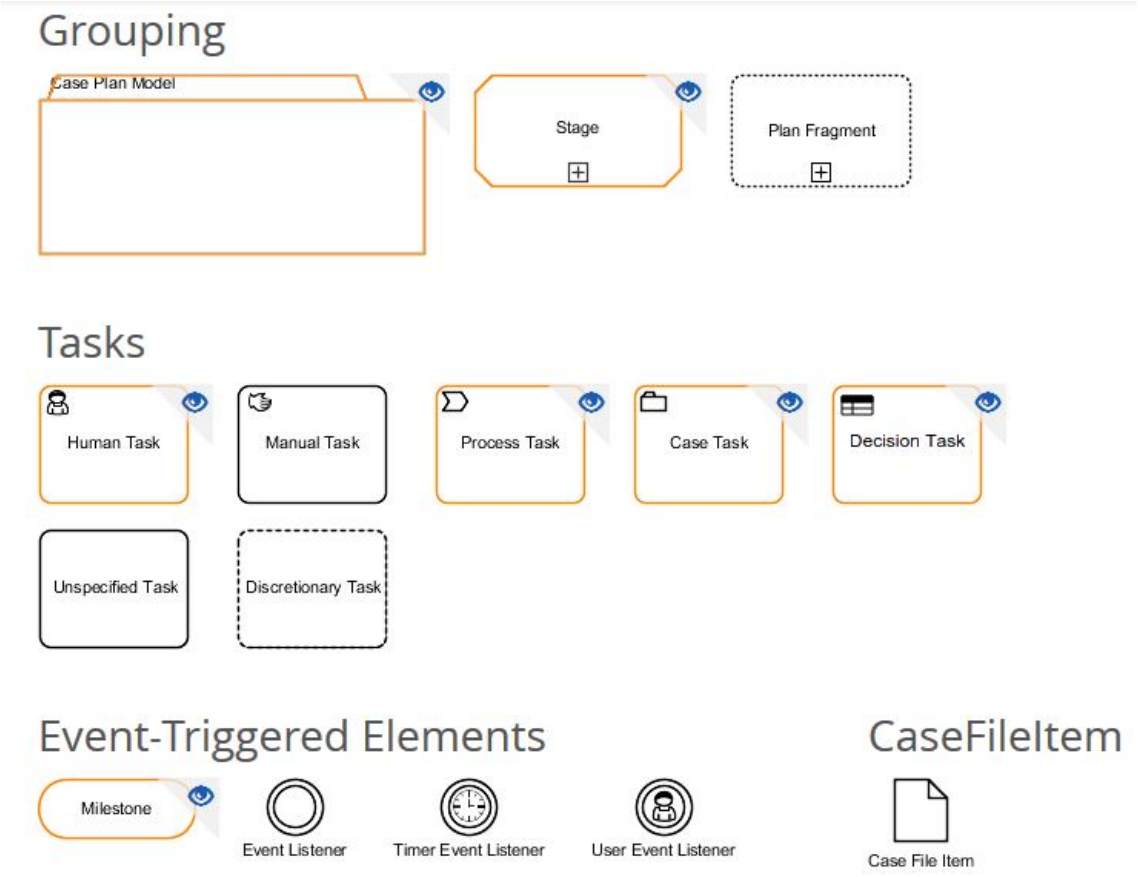

Figura A.1: Elementos CMMN. Fonte: [5].

Task é uma unidade de trabalho, representa a execução do trabalho propriamente dita. Existem 4 tipos de tarefas:

- *Human Task*: Tarefa executa por um trabalhador do caso. Pode ser uma tarefa que interrompe o fluxo de execução representado pela pessoa na borda

da tarefa ou que não interrompe a sequência de execução representado pela mão na borda.

- *Process Task*: É usado para invocar um processo BPMN de dentro de um case.
- *Decision Task*: É relacionado a um diagrama DMN.
- *Case Task*: Tarefa que referencia outro diagrama CMMN.

As tarefas discricionárias, representadas pela borda tracejada, são tarefas que podem ser adicionados em tempo de execução pelo usuário de acordo com seu julgamento.

Transições e as dependências em um caso cmmn podem ser modeladas com o uso de *sentries*. Essas transições podem ser de entrada ou saída representadas por critérios:

- *Entry Criteria*: Condições que determinam quando um *plan item* está habilitado para execução, uma determinada tarefa pode depender da execução de outra tarefa ou de se atingir determinado marco. Representado por um diamante vazio mostrado na Figura A.3.
- *Exit Criteria*: Condições que determinam quando *plan item* pode passar para um estado final. Representado por um diamante preenchido.

Os *sentries* podem ter duas partes: *OnPart*, um ou mais eventos como gatilhos, podem ser tarefas, marcos atingidos ou eventos que aconteceram; IfPart, condições representadas por expressões booleanas.

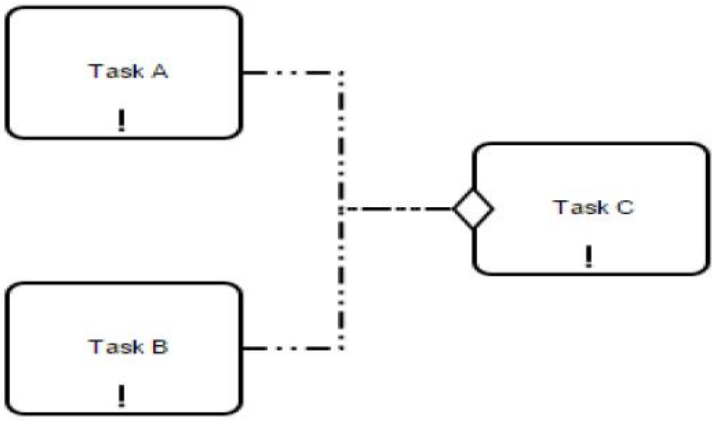

Figura A.2: Dependência entre tarefas. Fonte: [3].

Um exemplo do uso de sentry na imagem A.2. A tarefa C possui um sentry de entrada e pode ser ativada se uma das tarefas A ou B for completada.

Milestones: São marcos que podem representar objetivos e condições intermediárias e são usados para determinar o progresso do caso. Não atingir o marco pode indicar a falha de uma determinada instância de um caso e o caso ser encerrado.

Event Listeners: Esperam a ocorrência de eventos para disparar alguma atividade. Existem 3 tipos:

- *Event Listener.*
- *Timer event listener.*
- *User event listener.*

Type	Marker						
	Planning Table	Entry Criterion	Exit Criterion	AutoComplete	Manual Activation	Required	Repetition
Case Plan Model	✖		✔	✔			
Stage	✖	✔	✔	✔	✔	✔	✔
Task	✖	✔	✔		✔	✔	✔
Milestone		✔				✔	✔
EventListener							
CaseFileItem							
PlanFragment							

Figura A.3: Decoradores CMMN. Fonte: [5].

Decoradores auxiliam na expressividade do modelo cmmn e podem ser adicionados de acordo com a imagem A.3.

- *Manual Activation*: existem duas maneiras de ativar uma tarefa: automática, que é a maneira default; ativação manual, usuário pode decidir se ativa a tarefa ou se desabilita.
- *Required*: Para que o *stage* que contém um item de plano com decorador required seja completado, esse item de plano precisa atingir algum estado terminal.
- *Repetition*: Indica que condições uma tarefa, stage ou milestone precisam para ser repetida.
- *Auto Complete*: indica que o stage será completado quando todos os plano de itens obrigatórios estiverem completados. [33] [34][5]

Apêndice B

Desenvolvendo Plugin

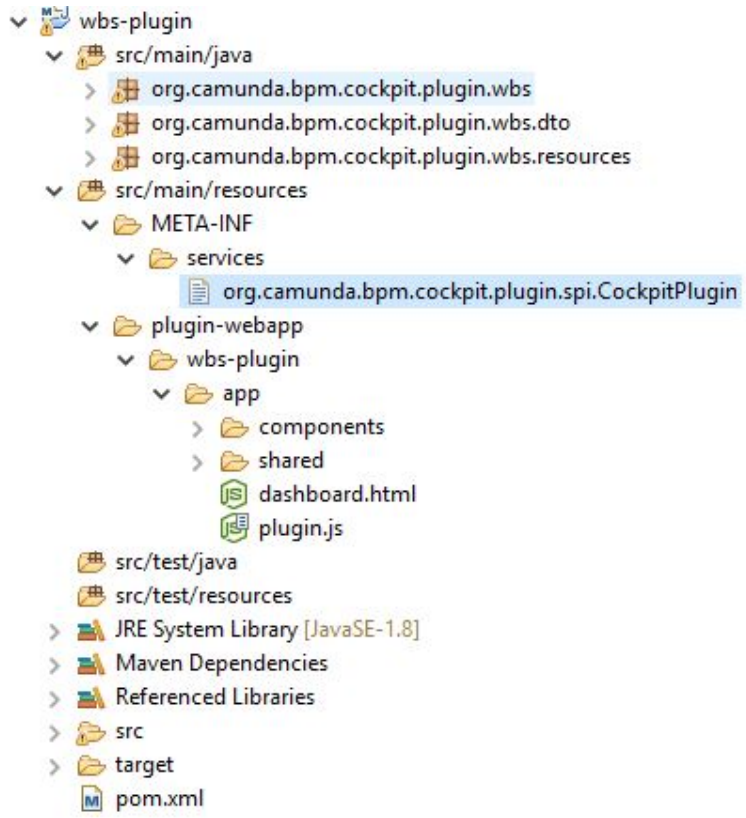

Figura B.1: Estrutura Wbs Builder.

A Figura B.1 mostra a estrutura do código do plugin desenvolvido. Para que Camunda reconheça o plugin, certas configurações precisam ser feitas. Dentro da pasta src/main/java ficam as classes Java que interagem com engine do camunda e as classes de recursos Jax-RS que são referentes a parte do servidor. Dentro da pasta src/main/resources/plugin-webapp fica a parte do cliente implementada em

Html e AngularJS.

A primeira coisa a ser feita é a criação de uma classe principal do plugin que deve estender a classe AbstractCockpit Plugin e pode ser vista na Figura B.2

```
package org.camunda.bpm.cockpit.plugin.wbs;

import java.util.HashSet;

public class WbsPlugin extends AbstractCockpitPlugin{

    public static final String ID = "wbs-plugin";

    public String getId() {
        return ID;
    }

    @Override
    public Set<Class<?>> getResourceClasses() {
        Set<Class<?>> classes = new HashSet<Class<?>>();

        classes.add(WbsPluginRootResource.class);

        return classes;
    }

}
```

Figura B.2: Classe Principal do Plugin.

O plugin deve ser publicado na pasta /META-INF/services para que Camunda reconheça essa extensão. Alguns termos importantes para entender aplicações extensíveis:

- Service: Conjunto de interfaces e um mecanismo para se obter uma implementação que forneça alguma funcionalidade.
- Service Provider Interface (SPI): Conjunto de interfaces públicas e classes abstratas que um serviço define.
- Service Provider: Uma implementação do SPI.

[35]

O service provider registrado é a classe principal do KIP Plan Builder.

JAX-RS é uma api java para facilitar o desenvolvimento de web services com arquitetura REST(Representational State Transfer). Oferece anotações declarativas que permitem: identificação de componentes, roteamento de requisições para métodos de uma classe, extração de dados da requisição para argumentos de um método. [36]

Alguns exemplos de anotações JAX-RS são:

- @GET, @POST, @PUT, @DELETE: Indicam a qual requisição HTTP aquele método irá responder. Devem ser usados em cima de um método.
- @PathParam: Extrai os parametros provenientes da url para argumentos.
- @QueryParam: Extrai os parametros definidos na query string da url para argumentos.

[37]

Na Figura B.3 é possível ver exemplos do uso da notação @Path e @PathParam que são responsáveis pelo mapeamento de urls para métodos e extração de dados da url para argumentos respectivamente. Na Figura B.4 faz se uso da notação @GET e esse método é a implementação do método mapeado na Figura B.3 chamado de getAllProcessDefinitions.

```
@Path("plugin/" + WbsPlugin.ID)
public class WbsPluginRootResource extends AbstractPluginRootResource{

    public WbsPluginRootResource() {
        super(WbsPlugin.ID);
    }

    @Path("{engineName}/process-definition")
    public ProcessDefinitionResource getAllProcessDefinitions(@PathParam("engineName") String engineName) {
        return subResource(new ProcessDefinitionResource(engineName), engineName);
    }
```

Figura B.3: Root Resource.

```java
public class ProcessDefinitionResource extends AbstractPluginResource{

    public ProcessDefinitionResource(String engineName) {
        super(engineName);
    }

    //Retorna lista com ids das definicoes de processos armazenadas na base do camunda
    @GET
    public List<String> getAllProcessDefinitions() {
        ProcessEngine processEngine = ProcessEngines.getProcessEngine(engineName);
        RepositoryService repositoryService = processEngine.getRepositoryService();
        List<CaseDefinition> caseDefinitions = repositoryService.createCaseDefinitionQuery()
                .orderByCaseDefinitionName().asc().list();
        List<String> ids = caseDefinitions.stream().map(CaseDefinition::getId).collect(Collectors.toList());
        return ids;
    }
}
```

Figura B.4: Classe ProcessDefinition.

Na parte do cliente, foi criado um DataFactory em Angular que consome os recursos oferecidos pelo JAX-RS atráves de métodos HTTP. Esse DataFactory é injetado em outros controladores que precisam acessar um desses métodos, um trecho do código do DataFactory é exibido na Figura B.5.

```javascript
ngDefine('cockpit.plugin.wbs-plugin.servicesModule', function(module){

    module.factory('DataFactory', ["$http", "Uri", "$rootScope", "$q", "$location", function($http, Uri, $rootScope, $q, $location){
        var DataFactory ={};

        DataFactory.processDefinitions = [];

        DataFactory.getAllProcessDefinitions = function(){
            return $http.get(Uri.appUri("plugin://wbs-plugin/:engine/process-definition"))
              .success(function(data) {
                DataFactory.processDefinitions = data;
             });
        };

        DataFactory.getProjectPlanNames = function(){
            return $http.get(Uri.appUri("plugin://wbs-plugin/:engine/getProjectPlanNames"))
              .success(function(data){

            });
        };

        DataFactory.getTasks = function(plano){
            console.log(plano);
            return $http.get(Uri.appUri("plugin://wbs-plugin/:engine/tasks?planoProjeto="+plano))
            .success(function(data){
            });
        };

        DataFactory.criarPlanoDeProjeto = function(plano){
            return $http.get(Uri.appUri("plugin://wbs-plugin/:engine/createProjectPlan?IdProcesso=" + plano.IdProcesso + "&NomePlano="
            .success(function(data){

            });
        }
```

Figura B.5: DataFactory.

Como discutido previamente, existem diversos pontos de extensão no Camunda, onde o plugin irá aparecer precisa ser configurado no Angular. Essa configuração é exibida na Figura B.6

```
ngDefine('cockpit.plugin.wbs-plugin.componentsModule', [
    './dashboardController',
    './processesController',
    './tasksController',
    './createPlanController',
    './createTaskCtrl',
    './relatorioPopupCtrl'
],
function(module){

    var Configuration = ['ViewsProvider', function(ViewsProvider) {

        ViewsProvider.registerDefaultView('cockpit.dashboard', {
          id: 'process-definitions',
          label: 'Deployed Processes',
          url: 'plugin://wbs-plugin/static/app/dashboard.html',
          dashboardMenuLabel: 'Sample',
          controller: 'dashboardController',

          // make sure we have a higher priority than the default plugin
          priority: 12
        });
      }];

      Configuration.$inject = ['ViewsProvider'];

      module.config(Configuration);

      return module;
});
```

Figura B.6: Configuração de localização do Plugin.

A Figura B.7 mostra as classes do lado do Servidor. O pacote ...wbs contém a classe principal WbsPlugin.Java e algumas classes auxiliares como XmlParser que é responsável pela extração das regras dos modelos. O pacote ...wbs.dto possui as classes dos objetos de transferência de dados que representam tarefas, plan item, plano de projeto e a definição de processo. ...wbs.resources possui as classes JAX-RS que interagem com a api Java do Camunda para oferecer funcionalidades ao usuário e onde é feito o mapeamento entre urls e os métodos.

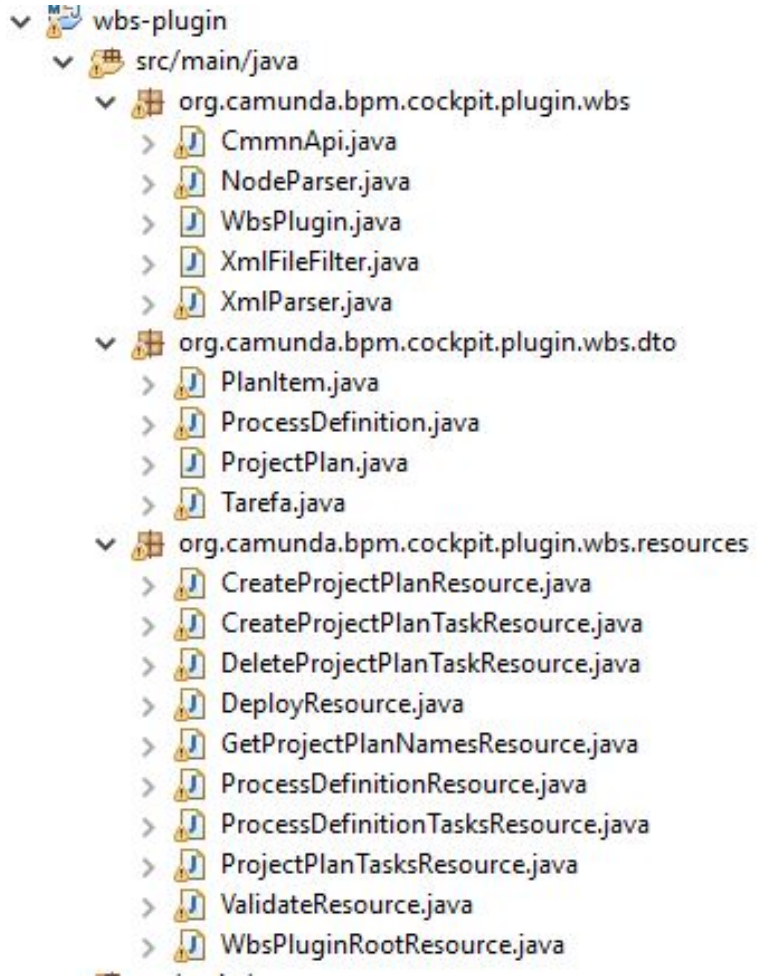

Figura B.7: Lado Servidor.

A Figura B.8 mostra os arquivos do lado do cliente. O arquivo dentro de /META-INF/services é onde o plugin é registrado como service provider para camunda. O arquivo DataFactory é o que consome os recursos oferecidos pelo servidor e as diversas páginas html e controladores em AngularJS são responsáveis pela exibição das páginas, interação com usuário e exibição de popups.

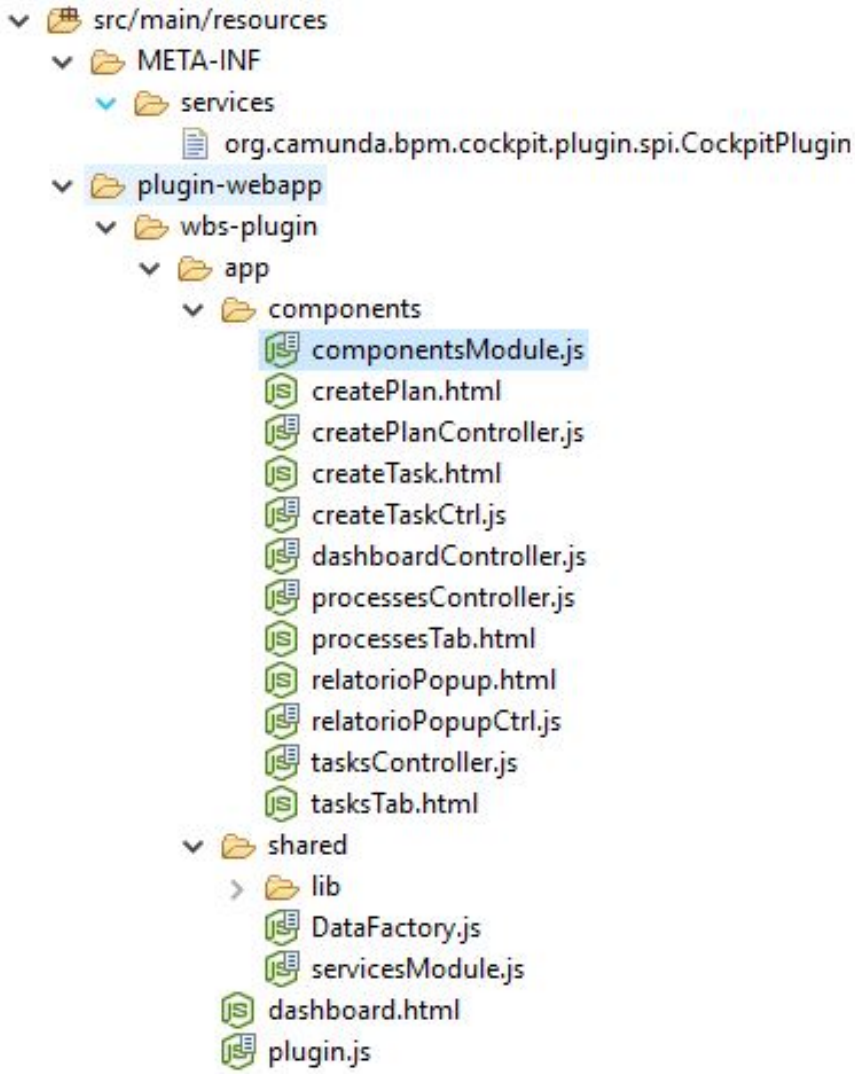

Figura B.8: Arquivos lado Cliente.