UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE SISTEMAS DE INFORMAÇÃO

VALDIR ALVES MENDES JUNIOR

Estudo da Aplicação de Sistemas de Gestão de Processos de Negócios em diferentes Modelos de Desenvolvimento de Software

FLORIANÓPOLIS
2007

UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE SISTEMAS DE INFORMAÇÃO

VALDIR ALVES MENDES JUNIOR

Estudo da Aplicação de Sistemas de Gestão de Processos de Negócios em diferentes Modelos de Desenvolvimento de Software

Trabalho de Conclusão de Curso submetido à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Bacharel em Sistemas de Informação

Prof. José Leomar Todesco
Orientador

FLORIANÓPOLIS
2007

VALDIR ALVES MENDES JUNIOR

Estudo da Aplicação de Sistemas de Gestão de Processos de Negócios em diferentes Modelos de Desenvolvimento de Software

Trabalho de Conclusão de Curso submetido à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Bacharel em Sistemas de Informação

_____

Prof. José Leomar Todesco

Orientador

_____

Prof. Roberto Carlos dos Santos Pacheco

Banca Examinadora

_____

Prof. José Francisco Salm Jr.

Banca Examinadora

Dedico este trabalho a minha família,
que deu a oportunidade de chegar até onde estou.
A minha namorada, pelo incentivo e
apoio nos momentos mais difíceis.
Aos meus amigos, que acreditaram em mim e
não me deixaram desistir.

Agradecimentos

A Deus, por tudo.

À minha família, por tudo que fizeram durante minha vida.

À minha namorada, Aline, pela compreensão e paciência.

Ao meu orientador, que apesar de pouco tempo que conseguimos conciliar nossas agendas, pôde direcionar meus esforços e mostrar o caminho que deveria ser seguido.

À Universidade e aos professores, que deram toda a estrutura necessária para meu crescimento pessoal e profissional.

Aos meus amigos, principalmente ao Juliano e Cassiano, pela amizade e momentos de descontração durante a faculdade.

A todas as pessoas que ajudaram no meu aprendizado, que foram fundamentais para o sucesso deste trabalho.

RESUMO

Quando se fala em ganho de competitividade nas organizações, logo se relaciona com quem proporciona serviços e/ou produtos com rapidez, qualidade e custos mais baixos.

Conseqüentemente, na indústria de desenvolvimento de software isso não é diferente. Com o crescimento acelerado pela procura de soluções tecnológicas que trouxessem tais melhorias, muitas empresas de software surgiram ou aumentaram sua demanda afim de suprir tais expectativas. Assim como em outros nichos de mercado, essas empresas devem buscar melhorias em seus processos, podendo trazer com isso um bom lugar no mercado de soluções tecnológicas.

Dentro desse cenário, este trabalho traz um estudo comparativo das metodologias de desenvolvimento de software para a utilização de tecnologias de Gestão de Process de Negócios (BPM), um conceito que une gestão de processos de negócios com tecnologia de informação voltada à melhoria de processos. Com uma análise conceitual, o foco desta pesquisa será a estruturação dos processos e a implementação das atividades, utilizando os modelos de desenvolvimento propostos.

Conclui-se com a indicação do uso dessa tecnologia para empresas que possuam ou queiram adotar um modelo de processo de desenvolvimento de software. Para isso, apresentam-se os principais impactos na sua utilização, variando a cada tipo de modelo avaliado.

Palavras-chave: BPM, Workflow, Processos de negócio, Modelos de desenvolvimento de software

ABSTRACT

When competitiveness comes to rise in organizations, soon appears who provides services and / or products with speed, quality and lower costs.

As a result, the industry of software development is not different. With the accelerated growth of demand for technology solutions that bring such improvements, many software companies emerged or increased their demand in order to meet such expectations. As in other niche markets, these companies must seek improvements in its processes, enabling it to reach a good place in the market for technology solutions.

In this scenario, this work brings a study about the application of software development methodologies at the point of view of Business Process Management (BPM), a concept that unifies management of business processes with information technology, focusing on the processes improvement. With a conceptual analysis and criticism, the focus of this research is the structure of the processes and the implementation of the activities, using the models of development proposed.

It accomplishes the indication of this technology to companies that might have or wish to adopt a model of process development for software. Therefore, it is presented the main impacts on their use, varying with each type of model evaluated.

Palavras-chave: BPM, Workflow, Business process, Software development models

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE REDUÇÕES

BPEL – *Business Process Execution Language*

BPM – *Business Process Management*

BPMI – *Business Process Management Iniative*

BPMN – *Business Process Management Notation*

BPMS – *Business Process Management Systems*

EAI – *Enterprise Application Integration*

IDE – *Integrated Development Environment*

JBPM – *JBoss Business Process Management*

JPDL – *jBpm Process Definition Language*

SOA – *Service Oriented Architecture*

UML – *Unified Model Language*

XPDL – *eXtensible Process Definition Language*

TI – *Tecnologia da informação*

YAWL – *Yet Another Workflow Language*

SUMÁRIO

1. INTRODUÇÃO

1.1. Apresentação

O ganho de competitividade e produtividade, buscadas constantemente pelas empresas, estão relacionadas com agilidade nos processos e otimização dos recursos, fornecendo qualidade e mantendo a segurança para os clientes. Esses desafios são enfrentados no dia-a-dia de muitas empresas que almejam o sucesso na busca por melhores resultados.

Três grandes mudanças mundiais fizeram com que o ambiente empresarial alterasse sua forma de organização: a globalização, a transformação em economias industriais em economias baseada em conhecimento, e as mudanças organizacionais (Laudon & Laudon, 1998).

A globalização trouxe consigo o aumento da quantidade de oferta, aumentando as expectativas e exigências do cliente. A concorrência entre empresas de diversos países fez com que o mundo corporativo exigisse transformações para que muitas empresas pudessem manter sua competitividade.

Estamos nos ingressando em uma nova era em que a fonte mais abrangente de riqueza passa ser a informação e o conhecimento. Isso faz com que economias baseadas em indústrias e capital sejam transformadas em organizações mais flexíveis, orgânicas e maleáveis, constituindo equipes multifuncionais ou rede sociais, e não mais departamentos com estruturas organizacionais elementares (Chiavenato, 1993).

Grande parte dessas transformações organizacionais está voltada para a tecnologia da informação. A automação de processos de negócio está cada vez mais procurada através dos sistemas de informação, a fim de otimizar seus recursos humanos e fluxos de trabalho. Com a tecnologia da informação, o sucesso de muitas áreas das organizações foi alavancado por softwares que atendessem suas expectativas, sendo fundamental no desenvolvimento dos produtos e serviços.

Para garantir essas melhorias e automatizar os processos é necessário antes identificar e mapear todos os processos que necessitam de melhorias a fim de poder modelar conforme a otimização necessária para o negócio. Logo após essa modelagem, pode-se então, automatizar e aplicar as tecnologias disponíveis no mercado nesses processos.

Dentro dos tipos de software encontrados para a automação de processos, estão os sistemas de gestão de processos de negócios. Essas ferramentas vêm ao encontro das estratégias de redesenho e otimização de processos de negócio através da automatização oferecida nos fluxos de trabalho. Com esse auxílio, as organizações tornam-se mais ágeis, seguras, confiáveis e por enquanto proporcionam um diferencial entre outras organizações (Usirono, 2003). Além disso, permite que outros sistemas possam ser integrados e que a monitoração dos processos seja mais efetiva na tomada de decisão na empresa.

Através desse contexto, este trabalho irá aprofundar, avaliar e discutir, através de uma ótica de gestão de processos, os efeitos causados desses sistemas aos modelos de processos de software, assim como sua prática em um sistema de *workflow* utilizando os métodos estudados.

1.2. Objetivos

1.2.1. Objetivo Geral

Este trabalho tem como objetivo principal realizar a aplicação de sistemas de gestão de processos nos principais modelos de desenvolvimento de software, desde as mais tradicionais até as ditas “ágeis”, com foco na identificação de ganhos de qualidade, gestão de prazos e redução de custos, entre outros.

1.2.2. Objetivos Específicos

Para se chegar ao objetivo principal, este trabalho foi organizado em:

- Apresentar conceitos relacionados a processos de negócios e reengenharia de processos, mostrando a sua relação com a atual gestão de processos de negócios.

- Identificar os componentes de uma solução de gestão de processos as tecnologias envolvidas para sua implementação, apresentando algumas linguagens e notações utilizadas atualmente.
- Apresentar os principais modelos de projetos em tecnologias de informação na atualidade, assim como uma breve descrição de cada uma delas.
- Analisar, modelar e realizar implementações para alguns modelos de gestão de projetos apresentados, bem como a sua utilização prática em uma organização que possua ou almeje um modelo de projeto de software.

1.3. Justificativas

Conforme os objetivos apresentados, a principal contribuição deste trabalho está em avaliar até que ponto a implementação da tecnologia de sistemas de gestão de processos e workflow na automação de processos de negócios afeta suas dimensões de desempenho em projetos de desenvolvimento de software.

Além disso, procura analisar e avaliar os inibidores e catalisadores inerentes à implementação de um sistema workflow, e como eles influenciaram no desempenho obtido. À medida que são feitas as análises e avaliações, a pesquisa desenvolve uma comparação entres os modelos de processos de software.

Por ser útil no gerenciamento de processos, abre discussões para que empresas do ramo de software não apenas forneçam tecnologias mas também a utilizem em seus processos de desenvolvimento de sistemas. Com a automação de seus processos, o uso dessas tecnologias ajudaria no crescimento desse tipo de organização e aumentaria seu destaque no mercado.

Com diversas experiências bem sucedidas e um nível tecnológico maduro, as tecnologias de BPM e *Workflow* trazem benefícios estratégicos na mudança automática nos processos de uma empresa, podendo diminuir custos e eliminar

rotinas desnecessárias em tempo real. Além disso, trazem grandes vantagens no processo transacional da empresa, maior qualidade e confiabilidade nos trabalhos executados, melhorias na comunicação de seus funcionários e clientes, redução de tempo para aplicar uma estratégia de negócio, além de estar prontamente apto para uma oportunidade de mercado (Fruscione, 1996).

Foi pensando nessas possibilidades e na sua aplicabilidade nos processos de desenvolvimento de software que este trabalho foi elaborado, conduzindo um aumento ainda maior de produtividade nos modelos de desenvolvimento propostos.

1.4. Organização dos Capítulos

Este trabalho foi divido em duas partes, uma com a revisão literária, demonstrando os principais conceitos necessários para o trabalho; e outra parte relacionada ao estudo dos casos de gestão como proposta do documento.

A parte um deste trabalho, destinada a revisão da literatura, contém as definições e conceitos necessários para o entendimento de processos, sua contextualização e como são as empresas tratam processos para sua otimização. Abrange os seguintes tópicos:

- Conceitos relacionados a processos de negócios;
- A reengenharia de processos como inovação de administração;
- A gestão por processos de negócios com visão da tecnologia de informação;
- Modelagem de processos, bem como as formas de notação;
- Sistemas de gestão de processos e *Workflow*, assim como automatização e otimização de processos;

Em seguida, a segunda parte do trabalho é iniciada com o conceito de processo de desenvolvimento de software. Logo após, é descrito os principais componentes do ciclo de vida de um software para que possam ser entendidas as principais atividades de sua construção. Além disso, essa parte do trabalho trata os principais modelos de desenvolvimento, dividindo em três categorias. Com isso, são colocados os seguintes tópicos:

- Demonstração do ciclo de vida de um software
- Modelos de processos tradicionais
- Modelos de processos iterativos e incrementais
- Modelos de processos ágeis

Já na terceira parte, é apresentado a análise e os resultados obtidos com a pesquisa. Também serão abordados os métodos de pesquisa e comparação e a descrição do estudo nas aplicações do trabalho, bem como seus resultados e discussões. Em conseqüência, é possível demonstrar:

- Os critérios de avaliação
- O mapeamento dos processos
- A implementação de um sistema
- Análise individual e comparativa dos processos

2. Processos de Negócios

Por volta de 1990, empresas do ramo adotaram o termo “Reengenharia” como uma tentativa de ganhar a competitividade que tinham perdido durante os anos anteriores. Naquela época, a principal motivação era a Reengenharia de processos de negócios com foco principal nos processos de negócios.

Segundo Davenport (1993), processo de negócio é um conjunto de atividades projetadas para produzir uma saída específica para um cliente em particular ou negócio. Isso tem uma ênfase maior em como o trabalho será feito em uma organização, tendo como foco o produto ou serviço da empresa. O autor ainda conclui que o processo é especificado através do tempo e espaço, com um começo e um fim, e entradas e saídas claramente definidas, ou seja, uma estrutura para a ação.

Já Hammer & Champy’s (1993) definem esse conceito como uma coleção de atividades que tem uma ou mais espécies de entradas diferentes criando uma saída que adiciona valor ao cliente do processo.

Johansson et al. (1993) define processo como um conjunto de atividades ligadas que tem uma entrada e transforma isso em uma saída. O ideal é que a transformação ocorrida deveria adicionar valor para a entrada e criar uma saída mais útil e eficaz ao receptor do processo.

Para Rummler & Brache (1995), um processo de negócio é uma série de etapas projetadas para produzir um produto ou serviço. Os processos que o cliente externo é envolvido são chamados de *processos primários*, já os processos que são invisíveis ao cliente externo, mas essenciais à gerência eficaz do negócio, são denominados *processos de suporte*.

Estas definições têm certa homogeneidade em seus conceitos como entradas e saídas bem definidas; organizadas pelo tempo e espaço; deve haver um receptor para o processo, um cliente; as transformações ocasionadas devem adicionar valor ao recipiente; o processo não existe sozinho, deve acompanhar o negócio da organização; além de poder medir diversas funções.

Para o trabalho, iremos adotar a definição do processo como sendo um conjunto de atividades ou procedimentos executados de forma seqüencial ou paralela, que juntos realizam os objetivos específicos no contexto do negócio, definido através de regras e relacionamentos.

Nos últimos anos, tornou-se uma tendência incluir o termo nas ferramentas de desenvolvimento de software, surgindo em seguida novos termos como “orquestração de processos de negócios”, “gestão de processos de negócios”, etc. A verdade é que começou a se utilizar esses significados envolvendo muito pouco sobre os conceitos de processos de negócios nas ferramentas. Por exemplo, um fornecedor de ferramentas de integração de sistemas sugere que a orquestração de uma série de *web-services* com um único *web-service* de alta complexidade constitui um processo de negócios. A maioria dos engenheiros de processos imploraria em contestar tal concepção.

2.1. Reengenharia de Processos

A reengenharia de processos traz consigo uma visão diferente nas empresas. Ela procura substituir a velha interpretação dos processos voltada para a empresa por uma visão mais voltada para o cliente e o produto final. Com isso, as organizações passam a olhar suas atividades de outra forma e determinar como podem reconstruir esses processos para melhorar a condução do negócio. Essa reconstrução leva a melhorias na eficiência e na eficácia dos processos atuais, trazendo consigo a otimização dos fluxos de trabalho.

Morris & Brandon (1994) conceituam reengenharia como o reprojeto do trabalho e implementação de novos projetos. Davenport (1993) adiciona que deve existir a busca por mudanças radicais.

Para Hammer & Champy (1993), autores do termo, a reengenharia de processos é repensar as atividades fundamentais e reestruturar os processos radicalmente, para conseguir melhorias em indicadores críticos de desempenho, como a redução de custos, atendimento, velocidade e qualidade. Dessa definição decorrem quatro palavras chaves:

- Fundamental: a reengenharia busca fazer unicamente o essencial;

- Radical: desconsidera todas as estruturas anteriores e reinventa a maior parte;
- Drástica: joga tudo fora e busca a substituição de algo novo;
- Processos: orientada para processos e não para as tarefas, tendo visão mais abrangente.

Através desses conceitos nota-se que há um enfoque na reconstrução, no novo, no recomeçar, enfim, derrubar a casa antiga. Chiavenato (1993) difere isso das outras metodologias como a melhoria contínua e gestão de qualidade total, pois essas tratam do reaproveitamento dos processos antigos, porém seguem a mesma abordagem e dão importância aos processos.

Para citar um exemplo, suponha uma empresa que elimine os processos redundantes ou que não agregam valor ao processo. Ela está praticando a melhoria contínua. Isso pode diminuir as funções desnecessárias, diminuir prazos de entrega e aumentar a satisfação do cliente (Davenport, 1993).

Já outra empresa do ramo que esteja empregando reengenharia de processos poderia oferecer atendimento on-line em tempo real, disponibilizar seus produtos em outras plataformas, buscar novos nichos de mercado, dar autonomia e poder de delegação ao pessoal de linha de frente. Através disso, a empresa consegue novas oportunidades, aumento do número de clientes, participação nas mudanças organizacionais, além dos benefícios já citados na melhoria contínua.

Para aplicar a reengenharia, muitos autores se diferem ligeiramente entre nomes e passos nas diferentes metodologias. Mesmo assim, eles dividem os mesmos princípios e elementos básicos, como simplificar processos (assim como no princípio de Sun Tzu em “Dividir para conquistar”), reaproveitar e recombinar atividades, redefinir as tecnologias de informação, entre outros.

Porém, a mudança radical dos processos pode se tornar difícil para uma empresa. É preciso de pessoas com vontade e dispostas a mudar, assumindo responsabilidades, trabalho em equipe, etc. Chiavenato (1993) diz que a tecnologia pode ajudar e facilitar essa mudança.

A maioria dos autores sugere que a tecnologia da informação desempenha um papel fundamental na reengenharia de processos. Ela é considerada a principal forma de construir novas formas de trabalho e colaboração dentro da empresa e pelo mercado afora. Recentemente, os sistemas gerenciadores de *workflow* tornaram-se contribuintes significativos no melhoramento da eficiência dos processos. Esses sistemas serão abordados em capítulos seguintes mostrando a importância deles neste trabalho.

2.2. Gestão de Processos de Negócios

Os modelos de gestão estão mudando constantemente e muitas vezes não é possível reaproveitar os benefícios dos modelos anteriores. Como visto nos capítulos anteriores, é preciso mudar o paradigma e abolir a divisão por departamentos, adotar uma gestão voltada para processos. Quando uma empresa analisa, modela e implementa os seus processos, permite que estes tenham seus tempos de execução reduzidos, enquanto busca eliminar os principais problemas.

Gestão por processos de negócios, em inglês *Business Process Management* (BPM), vem solucionar esses problemas sendo a intersecção entre gestão e tecnologia. Esse modelo abrange métodos, técnicas e ferramentas para o auxílio ao projeto, controle e análise de processos de negócios, envolvendo recursos humanos, organizações, aplicações, documentos e outras fontes de informação (Aalst, Hofstede & Weske, 2003).

BPM se difere de reengenharia de processos, pois não visa mudanças revolucionárias nos processos de negócios, mas sua evolução contínua. Recentemente, BPM ganhou maior atenção no mundo corporativo e pode ser considerado o sucessor da Reengenharia de Processos, pois dirige seu foco na eficiência de processos através da tecnologia de informação e negligenciando os aspectos pessoais de mudanças (Aalst, Hofstede & Weske, 2003). Diferentemente da reengenharia, que significa iniciar tudo de novo, a gerência de processos constrói sobre e transforma o que já existe.

A mudança de paradigma desse modelo de gestão reside em adquirir soluções abrangentes para o negócio, partindo do desenho do processo por especialistas no assunto, utilizando ferramentas integradas pela equipe de desenvolvimento e que também permitirá a monitoração de eficiência nas atividades, a simulação de alterações e modificações no processo com uma mínima descontinuidade funcional (Gurley, 2003).

Gurley (2003) considera BPM como um novo modelo de gestão baseado em tecnologias de informação, trazendo consigo aplicações baseadas em *browsers* (navegadores de internet), e-mail (correio eletrônico), conectividade global e integração entre sistemas (Enterprise Application Integration – EAI). Com toda essa estrutura, pode-se disponibilizar uma solução muito poderosa, focada no negócio da empresa. Basicamente, uma mistura entre *workflow*, sistemas integrados e o desenvolvimento de aplicações, provendo processos mais fáceis de serem desenvolvidos, automatizando sua execução, monitorando a performance atual e fazendo com que os fluxos de trabalho possam ser mudados em tempo de execução.

O autor ainda sugere seis tópicos-chave para a existência de uma solução BPM:

- **Ferramentas integradas de desenvolvimento (IDE)**: abrange ferramentas de modelagem de processos e o ambiente de programação, tendo como finalidade modelar processos. Acredita-se ou não, muitas companhias aumentam consideravelmente seus ganhos apenas pelo simples fato de sentar e refletir sobre seus processos através de ferramentas que possibilitem modelar seus processos. Uma das características de aplicativos BPM, é poder modelar processos e isso se refletir automaticamente na execução dos mesmos sem a interferência da equipe de TI;
- ***Engine* de processo (motor)**: responsável pela coreografia dos processos, controla os estados e variáveis para as atividades de milhares de instâncias de processos. Para um sistema BPM, isso é de fundamental importância;

- **Papéis de usuários**: assim como em qualquer aplicativo, um administrador de sistemas define o papel e o nível de autorização para cada usuário. Em BPM, isso pode ser ajustado para cada tarefa de processo, definindo o responsável por cada atividade a ser executada em um processo. Através desse recurso, podem-se redefinir os executores de cada tarefa em tempo de execução do aplicativo, facilitando o remanejamento de pessoal nos processos;
- ***Workflow***: define a infra-estrutura de comunicação entre tarefas apropriadas para cada indivíduo. Normalmente, definido graficamente por ferramentas de modelagem (Ver Item 2.4.1);
- **Monitoração de processos**: compreende o rastreamento e medição de cada processo. Útil na identificação de pontos críticos e inconsistências, além de facilitar na melhoria contínua de processos;
- **Integração entre sistemas**: fator crítico em BPM. Permite que processos de negócios acessem dados de outros sistemas da organização. Aqui a adoção de *web-services* minimizaria a quantidade de esforço necessária para integrar sistemas legados e recém criados.

Recentemente, muitos fornecedores de aplicativos colocam o termo BPM em sistemas de integração com *web-services* ou sistemas *workflow* em seus produtos simplesmente pelo apelo de *marketing*. Através dos conceitos de Gurley percebe-se que para se ter uma solução BPM, não basta ter apenas um dos componentes desse modelo, e sim implementar todos ou a maioria de seus componentes, afim de realmente suprir todas a perspectivas da gestão de processos.

2.3. Modelagem de Processos

Em um mundo cada vez mais competitivo, as organizações de maior destaque são aquelas que proporcionam produtos ou serviços de forma rápida, com custos baixos, agregando qualidade e segurança para o cliente. Para atingir esses quesitos, a empresa deve buscar o aumento da eficiência, da satisfação do cliente e da lucratividade procurando identificar novos fatores competitivos,

explorando cada vez mais novos métodos de operação e entrega (Usirono, 2003). Dentro desse cenário, é preciso ter maior qualidade na forma de se pensar e agir, a fim de abrir novas portas no mercado.

Denominada modelagem ou mapeamento de processos, a representação dos processos atuais e futuros de uma organização, de modo que esses processos possam ser analisados e melhorados, podem aumentar a eficiência e qualidade do negócio (Hammer, 1993).

Para Hammer (1993), a aplicação de modelagem de processos é a força motriz para promover conhecimento na organização, impulsionando a redução do abismo de conhecimento entre desenvolvedores de sistemas, a gerência do negócio e o relacionamento com o cliente procurando um entendimento comum entre as partes.

Já para May (2003), construir modelos através dos cenários atuais do negócio podem ajudar na tomada de decisão, no planejamento e otimização da performance do processo. Isso é muito poderoso para calcular o suporte do processo, fazer mudanças no volume de produtos/serviços, simular preços de venda, avaliar recursos utilizados nas atividades, analisar reparos e obter indicadores de produtividade.

Hunt (1996) sugere que os fluxos de trabalho possam ser representados graficamente e textualmente para garantir o entendimento dos processos de negócio.

Havey (2005) complementa que os processos devem ser representados de forma simples e intuitiva através de diagramas elaborados por ferramentas de apoio e implementados por um modelo executável, utilizado na automação dos processos e gestão de *workflows*.

Com isso, temos duas formas de representar um processo: através de linguagens de marcação – com a qual serão abordadas nos próximos capítulos – e notações gráficas – explicadas a seguir. Cada tipo de representação tem seu mérito, e uma boa arquitetura requer a integração das duas.

Havey (2005) ainda propõe que, sendo um algoritmo, o processo bem modelado pode ser executado por qualquer *engine* de processos. À medida que

as atividades possam ser expressas de forma sintaticamente e semanticamente não ambíguas graficamente, uma linguagem de programação ou outro tipo de interpretador pode aceitá-los de forma que existam entradas e direcionem seu fluxo para determinado fim. Foi através desse conceito que foi possível à construção de *workflows* e automação de processos através de ferramentas gráficas, já que utilizam notações concisas para a geração de código ou execução de fluxo.

A partir daí, os elementos gráficos participaram não só da fase de construção de um sistema, mas também de sua execução. As notações gráficas de processos tornaram-se cada vez mais importantes na maneira de administrar sistemas.

Muitos autores empregam suas próprias representações de processos, alguns até de maneira específica para um determinado problema. Através dessas notações, foram criadas ferramentas que auxiliam na construção de modelos e diagramas. Para isso, precisavam padronizar em uma forma única de representar os processos, surgindo notações para o desenho de fluxos de trabalho. Para isso, têm-se exemplos como a Business Process Modeling Notation (BPMN), específico para a modelagem de processos e a Unified Modeling Language (UML), representando de forma geral a engenharia de software.

2.3.1. Unified Modeling Language

A UML é uma linguagem de especificação de modelo de objetos, tendo como principal objetivo ser uma notação gráfica para criar um modelo abstrato de um sistema. Com a UML, podemos ter poderosas técnicas para expressar muitas ocasiões durante a fase de análise e planejamento de um projeto (McLeod, 1998).

A UML inclui um diagrama de atividades, também chamado de diagrama de eventos. Essa notação é capaz de mostrar processos em um fluxo semelhante à notação BPMN, já que também possui suporte ao paralelismo, eventos e sincronização. Um diagrama de atividade é semelhante a um fluxo de

trabalho, mostra o controle de uma atividade para outra. Ele pode ser usado para o entendimento do negócio, para descrever sistemas com muitas atividades concorrentes e descrever algoritmos seqüenciais complexos. Em UML, esse diagrama se encaixa melhor à modelagem de processos de negócios (Jacobson, 1999).

McLeod (1998) sugere que outros diagramas da UML também podem ser utilizados para a modelagem de processos. Diagramas de componentes e implantação podem especificar a escolha de uma implementação específica, similar à escolha de uma tarefa no BPMN. Segundo Jacobson (1999), a utilização de diagramas de casos de uso também pode ser utilizada, mas carece na falta de semântica empregada para facilitar o entendimento.

2.3.2. Business Process Modeling Notation

A BPMN é uma notação gráfica para representação de processos de negócios em fluxos de trabalho. Essa notação foi desenvolvida pelo consórcio Business Process Management Initiative (BPMI) levando mais de dois anos de esforço do grupo. O primeiro esforço era fornecer uma notação que fosse fácil de entender por todos seus usuários: para que os analistas de negócios pudessem criar esboços iniciais dos processos, aos colaboradores técnicos responsáveis implementassem a tecnologia, e finalmente as pessoas que controlam e monitoram os processos. BPMN é a ponte do abismo entre planejamento e a implementação de processos de negócios (White, 2004).

Os diagramas produzidos por essa notação são constituídos de um conjunto de elementos gráficos. Esses elementos devem ser simples de produzir um diagrama do negócio e ao mesmo tempo poder alcançar a complexidade inerente dos processos.

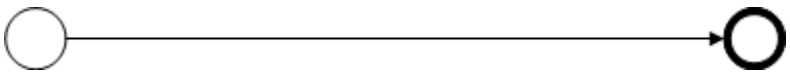


Figura 1: Início e Fim de um processo (Fonte: Autor)

Desse modo, existe um pequeno conjunto de categorias para facilitar o reconhecimento dos elementos e entender o diagrama. Dentro de cada

categoria, existem variações e informações adicionais que pode ser adicionado nos requisitos complexos sem modificar dramaticamente a visualização do diagrama (White, 2004).

As quatro categorias básicas são:

- Objetos de fluxo: compreende as tarefas, eventos, junções, bifurcações e sub-processos;

Figura 2: Objetos de Fluxo (Fonte: Autor)

- Objetos de conectividade: representa as transições, mensagens e associações;

Figura 3: Objetos de conectividade (Fonte: Autor)

- *Swimlanes* (raias): separa os objetos por funcionalidade ou por participante no processo;

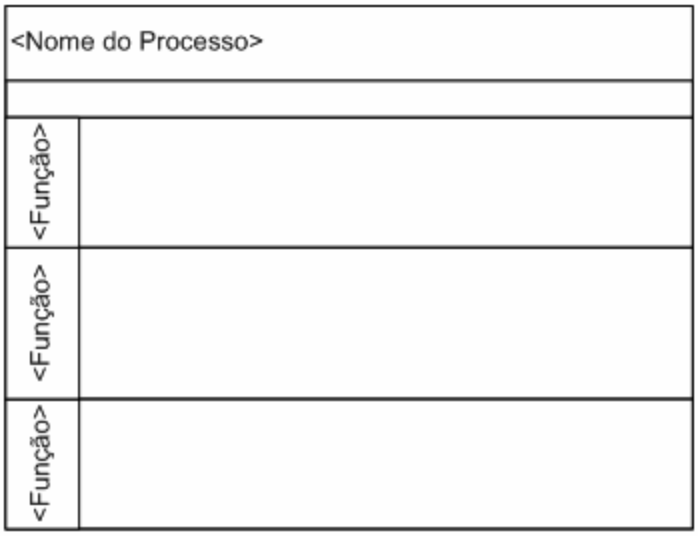


Figura 4: *Swinlanes* ou Raias (Fonte: Autor)

- Artefatos: fornecem informações adicionais ao processo.

Figura 5: Artefatos (Fonte: Autor)

Diante dessas possibilidades, essa notação será utilizada para a representação gráfica dos processos analisados deste trabalho.

Esta pesquisa não tem como finalidade descrever todos os elementos gráficos do BPMN, sendo que cada categoria apresentada possui suas particularidades na execução, permitindo um conjunto muito variado para representação de processos organizacionais. Para mais detalhes dos elementos gráficos da notação utilizados no trabalho, veja o Apêndice A.

A seguir, um simples exemplo de BPMN demonstrando um processo de compra, realizado com cartão bancário.

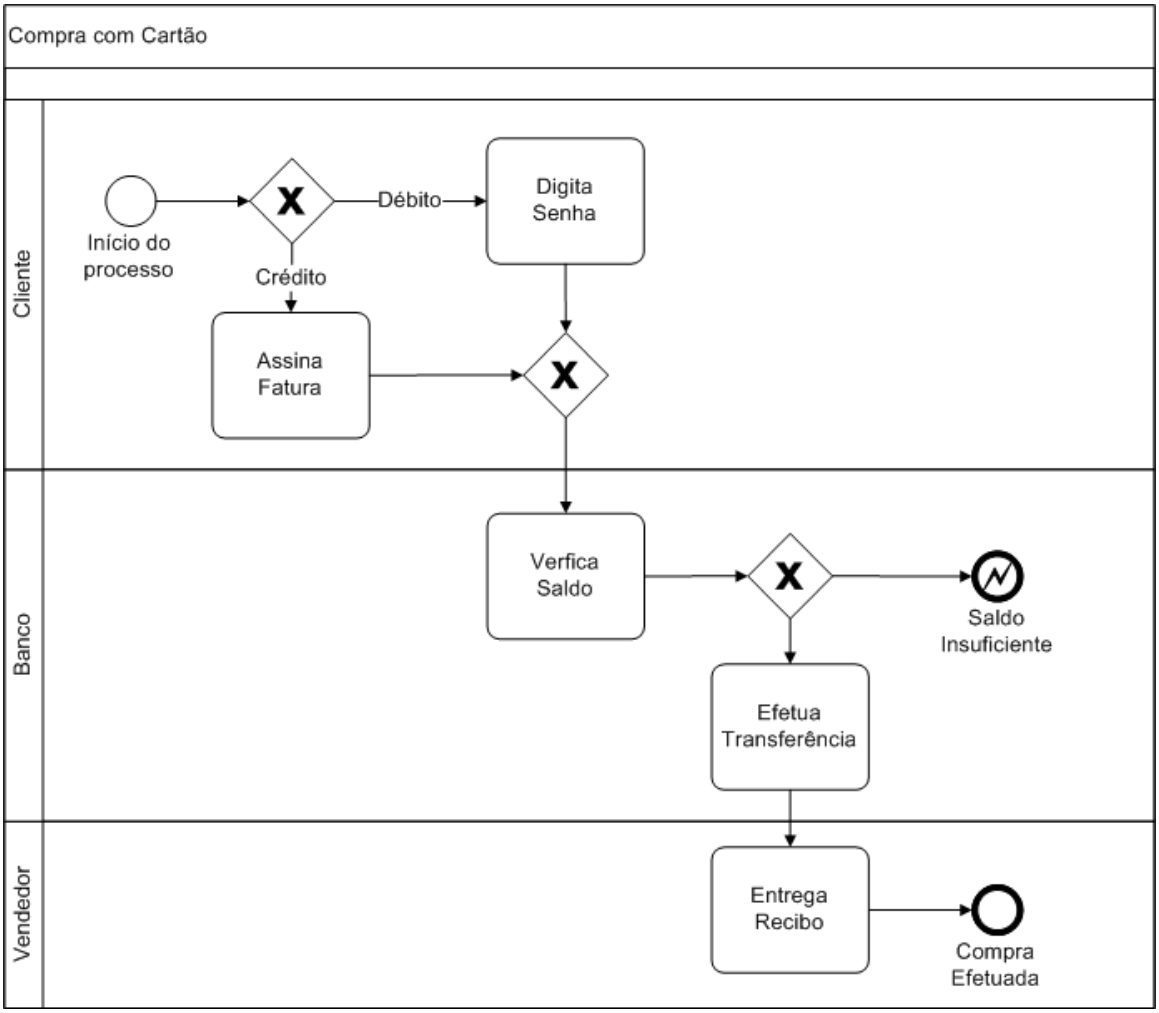


Figura 6: Exemplo BPMN - Compra com Cartão (Fonte: Autor)

2.4. Sistemas de Gestão de Processos de Negócios

Reis (2006) conceitua os sistemas de gestão de processos de negócios (BPM) como uma solução que permite a geração e controle dos processos de negócio da empresa, proporcionando rápida tomada de decisão e realinhamento dos processos de forma mais agilizada. Com isso, percebe-se a importância de sistemas BPM não só como uma maneira de colocar os processos em um software e automatizá-los, mas sim uma maneira de poder monitorá-los e redefini-los em tempo de execução, surtindo efeito nos fluxos de trabalho automaticamente.

Um sistema BPM cobre a execução de atividades das organizações para controlar e, se necessário, melhorar os processos de negócio. Enquanto um objetivo da empresa for difícil ou não estiver em conformidade, esse tipo de ferramenta faz com que a redefinição das atividades seja feita de maneira mais rápida e mais barata, já que não será preciso alterar o sistema atual com a ajuda da equipe de TI. Os sistemas de BPM monitoram a execução dos processos de negócio de modo que os gerentes possam analisar e transformar os processos em resposta aos dados (Aalst, Hofstede & Weske, 2003).

Embora o foco inicial desses sistemas seja o mecanismo de automatização dos processos, isso foi aperfeiçoado para que processos possam ser mais bem interagidos entre as pessoas, especialmente na questão de gerência da empresa.

Reis (2006) alerta ainda sobre a confusão entre BPMS, *Workflows*, EAI e SOA. Existem empresas se apropriando de termo BPM pelo simples fato desse conceito ser novo e aumentar as chances de vendas nas ferramentas, mas sem se enquadrar nesta categoria. Isto pode acontecer por desconhecimento ou até por má fé entre fornecedores. Caso seja optado desenvolver uma ferramenta que atenda os requisitos de uma solução BPMS, devem-se adquirir todos os conceitos sobre tal solução e definir se compreende todos os quesitos para a utilização do termo no produto.

2.4.1. Workflow

As soluções adotadas por empresas atualmente visam suprir os requisitos de uma forte aderência às regras de negócio, grande flexibilidade para acompanhar as constantes mudanças e o fundamento de um ambiente colaborativo que propicie uma gestão de conhecimento contínuo.

Sendo um tipo de sistema de gestão de processos de negócios, os sistemas de *workflow* representam uma solução para atender essa demanda, pois se adéquam muito bem ao conhecimento do negócio, possuindo estratégias e um dinamismo que apóiam os quesitos indispensáveis para o sucesso da automação dos processos (Gurley, 2003).

Um gerenciador de *workflow* representa um componente de software que tem como entrada uma descrição formal dos processos de negócios mantendo os estado de execução desses processos, delegando informações ou tarefas de um colaborador para outro (Baeyens, 2004).

O conceito de *workflow* surgiu da noção de processos organizacionais, que se relacionavam a manufatura e ao escritório. Estes processos surgiram desde a industrialização e eram criados para buscar melhorias na eficiência das atividades empenhadas. Normalmente, o trabalho era dividido em tarefas bem definidas, regras, papéis e procedimentos que guiavam a maior parte do trabalho realizado nas atividades de manufatura e escritório (Georgakopoulos et. al., 1995).

O termo *workflow* é definido por diversos autores da literatura. Em Georgakopoulos et. al. (1995) é dito que há pouco consenso sobre o que trata um *workflow* e quais a funcionalidades que um sistema de *workflow* deve ter. Através disso, o autor conceitua diversas definições fornecidas por empresas que produzem ou fazem uso de *workflow*, tais conceitos como:

- Para um representante da empresa PeopleSoft Inc., um *workflow* é o mecanismo que possibilita a implementação das práticas de reengenharia de processos (visto no item 2.1 deste trabalho);
- Na Recognition Internal Inc., na descrição de um produto encontra-se a definição de *workflow* como o processo através do qual atividades independentes se unem para completar uma transação, ou um processo do negócio, dentro da empresa.

Através desses estudos, Georgakopoulos et. al. (1995) define que *workflow* é a coleção de tarefas organizadas para executar um determinado processo de negócio. Estas tarefas podem ser executadas por um ou mais sistemas além de uma ou mais pessoas, podendo haver diversas combinações de ambos. Além disso, é destacado que um *workflow* define a ordem das atividades a serem executadas sob um conjunto de condições necessárias afim de obter a sincronização e finalização do processo.

Para Kappel (2000), não há distinção entre *workflow* e processos de negócios. O autor detalha que os mesmos tipicamente são constituídos de várias atividades que devem ser executadas em ordem definida, envolvendo certo número de colaboradores em um ambiente distribuído para alcançar um objetivo comum na organização.

Já Hollingworth (1995), o *workflow* pode facilitar ou automatizar o processo de negócio, totalmente ou parcialmente. Ele ainda detalha que o objetivo do *workflow* é a automação (ou computadorização) dos procedimentos cujos documentos, informações ou tarefas são passadas entre os participantes seguindo um conjunto definido de regras, contribuindo para o objetivo final do negócio.

Com essa definição, os sistemas de *workflow* se baseiam na reengenharia de processos de negócios, a qual trata de análise, modelagem, definição e implementação dos principais processos de negócios da organização.

Pode-se considerar também que para se ter um *workflow* não será necessário um sistema de gerenciamento de *workflows*, apenas um sistema que completamente define, gerencie e executa as atividades de um processo de negócio em um software o qual a ordem de execução esteja guiada pela representação gráfica do *workflow* (Hollingsworth, 1995). Porém, não tendo um sistema de gerenciamento *workflow* adequado, a qual não permita a remodelagem gráfica das atividades, o software deve ser reimplementado toda vez que a ordem das atividades muda, tornando o sistema inflexível para adaptar-se aos processos da empresa.

2.4.1.1. Características

Os sistemas de gerenciamento de workflow devem estar preparados para executar uma grande variedade de processos de negócio. Para isso, estes sistemas têm que tratar as diversas variáveis envolvidas nos fluxos de trabalho, como por exemplo o tempo de duração de um processo, a execução paralela de atividades, o ambiente de execução a qual irá ser implementado, entre outras.

Apesar dessa variabilidade, os sistemas de gerenciamento de workflow possuem um conjunto de características comuns. Este conjunto fornece uma base para o desenvolvimento de tecnologias que permitam a integração e a interoperabilidade destes sistemas (Hollingsworth, 1995).

Ainda em Hollingsworth (1995), são definidas as principais características comuns a estes sistemas são:

- Possibilidade de definir e modelar os fluxos de trabalho dos processos e suas atividades;
- Funções de controle voltadas ao monitoramento dos processos durante sua execução, bem como a ordenação das atividades que constituem os processos gerenciados;
- Interações com usuários e aplicações responsáveis por executar as atividades definidas.

2.4.1.2. Principais Benefícios

Foram apontados diversos benefícios na literatura sobre a implementação de sistemas de gerenciamento de *workflows*. Em WfMC (2006) pode-se encontrar a seguinte lista de benefícios:

- Melhoria na eficiência: Durante a automação dos processos, ocorre a eliminação de diversas atividades antes necessárias;
- Maior controle do processo: Advindo da implementação dos sistemas de *workflow*, o gerenciamento dos processos torna-se simplificado, resultado da padronização das metodologias de trabalho e da disponibilidade de auditorias geradas pelo sistema;
- Melhoria na qualidade de atendimento aos clientes: Com *workflow*, aumenta o grau de consistência dos processos, o que permite uma maior previsibilidade relativa aos níveis de resposta aos clientes;
- Flexibilidade: Através do gerenciamento dos fluxos graficamente, o redesenho dos processos podem ser feitos enquanto o sistema ainda esteja rodando, atendendo as mudanças das necessidades do negócio em tempo real;

- Melhoria nos processos de negócio: Através dos controles citados, o foco nos processos pode simplificar ou dinamizar o fluxo das atividades.

A flexibilidade é considerada um dos maiores benefícios para empresas possuem constantes mudanças em seus processos. Com a dinâmica do mercado, diversas organizações precisam estar prontas para as mudanças no ambiente de negócios afim de atender os requisitos do mercado e a legislação vigente.

Para o escopo deste trabalho, isso será de grande valia para as metodologias de processo de software, visto que podem existir diversas mudanças no projeto devido a alterações nos requisitos do cliente ou na arquitetura utilizada. Caso isso ocorra, o sistema de *workflow* utilizado pode corrigir rapidamente o processo das atividades, alterando as definições no *workflow*.

2.4.2. Linguagens de Definição de Processos

Para um modelo de processos, tanto o analista de negócios quanto os desenvolvedores da tecnologia requerem um alto nível de padronização para garantir a qualidade de implementação. Havey (2005) diz que existem duas regras para impulsionar a qualidade e manutenção de programação: manter os processos suficientemente pequenos para poder ajustar em uma tela de computador ou folha de papel (mover complexidade para sub-processos) e manter os processos na maior granularidade possível (menos detalhes por fluxo).

Através da importância da padronização de definições de processos, foram elaboradas algumas linguagens de marcação para tal fim. O maior objetivo desse propósito é ter interoperabilidade e integração entre ferramentas e *engines* de processos, garantindo a componentização e comunicação entre diversas plataformas diferentes.

Pode-se citar algumas linguagens como o XPDL, uma linguagem prática de desenvolvimento; BPEL, que tem suporte nativo para *web services*; YAWL,

que garante conceitos básicos de processos; ou jPDL, uma solução que mistura algumas técnicas de programação declarativa.

2.4.2.1. Process Definition Language

XPDL é uma linguagem de formato padronizado pelo intercâmbio de diversos produtos de *workflows* existentes como ferramentas de modelagem e *engines* de processos. Um documento XPDL representa uma ou mais definições de processos, cada qual com um conjunto de atividades e transições (Havey, 2005).

Essa linguagem foi especialmente desenvolvida para padronizar as ferramentas e *engines* de processos. A idéia é que uma ferramenta de modelagem possa exportar um documento XPDL com a definição de um processo e poder carregar essa definição em outra ferramenta distinta, com a qual suporte o formato da linguagem. Além disso, um dos propósitos é mover a definição criada por uma ferramenta gráfica de modelagem de processos diretamente para a execução do processo em uma *engine* de workflow.

2.4.2.2. Business Process Execution Language

A BPEL, como próprio nome sugere, é a linguagem de definição e execução de processos de negócio. Seu foco principal está na programação e a possibilidade da utilização de *web-services* na orquestração de processos.

Essa especificação também pode ser considerada como uma extensão da utilização de serviços *web* comuns, aumentando a interações entre serviços e facilitando na implementação desses sistemas (Dietzen, 2004). Com BPEL, pode-se tanto definir como orquestrar *web services* como um *workflow.*

De acordo com Bortolini (2006), BPEL nasceu da união de padrões de grandes empresas como o XLANG da Microsoft e o WSFL da IBM. Por essa influência, muitos fornecedores de tecnologias de gestão de processos ficaram atentos a essa linguagem, adotando-a como padrão.

O autor ainda fala sobre sua pouca utilização no mercado e aplicação prática. Não que essa tecnologia suporte poucos casos de utilização, mas pelo fato de não haver uma organização que suporte a linguagem com sua plena

potencialidade. Devido a isso, foi relatado que BPEL está sendo mais utilizado na criação e orquestração de *web-services* do que no controle e execução de processos de negócios.

2.4.2.3. Yet Another Workflow Language

A YAWL é uma linguagem de baseada em padrões de projeto de *workflow.* A idéia da criação da linguagem foi o suporte diversificado de padrões de *workflow* existentes em apenas uma definição que compreenda todos os padrões.

Sem uma unificação de conceitos, cada fornecedor oferecia uma solução – ou padrão – para um determinado problema mas que não era suportada por outros sistemas ou não continha o suporte ao padrão completamente. O resultado disso era desencorajador: cada fornecedor tinha uma solução diferente para cada padrão, sendo que muitos não suportavam a metade dos problemas (Havey, 2005). Pior ainda, os fornecedores não tinham consistência teórica, cada produto possuía sua própria notação, definição e semântica.

YAWL é uma das possíveis soluções para a unificação de padrões de *workflow*, já que seu objetivo principal é fornecer um conjunto de definições que garanta a abrangência em várias perspectivas diferentes.

2.4.2.4. jBPM Process Definition Language

A jPDL é uma linguagem de processos baseada em notações gráficas como a UML. O propósito de sua definição é executar modelos de processos através de sua *engine*: o jBPM, um componente de execução de processos como alternativa de infra-estrutura de software.

A linguagem é centralizada no conceito de estado do processo. Uma das razões principais é que a maioria das linguagens de programação também suporta o conceito de estado: “um programa está funcionando ou não está” (Jboss, 2007), simplificando na construção de transições, decisões e ações em paralelo. Com isso, a filosofia do jPDL é estender a linguagem Java para a gestão de estados de processos e ter menos dificuldades na fase de

programação. Para a equipe Jboss (2007), a linguagem diferencia-se das demais pelo simples fato de ajudar nessa implementação.

Nesta linguagem, as funcionalidades básicas do *workflow* implementado são simplesmente empacotadas com se fosse uma biblioteca Java. Esta biblioteca possui os serviços necessários para gerenciar e executar os processos em um servidor de aplicações com um banco de dados (Figura 7).

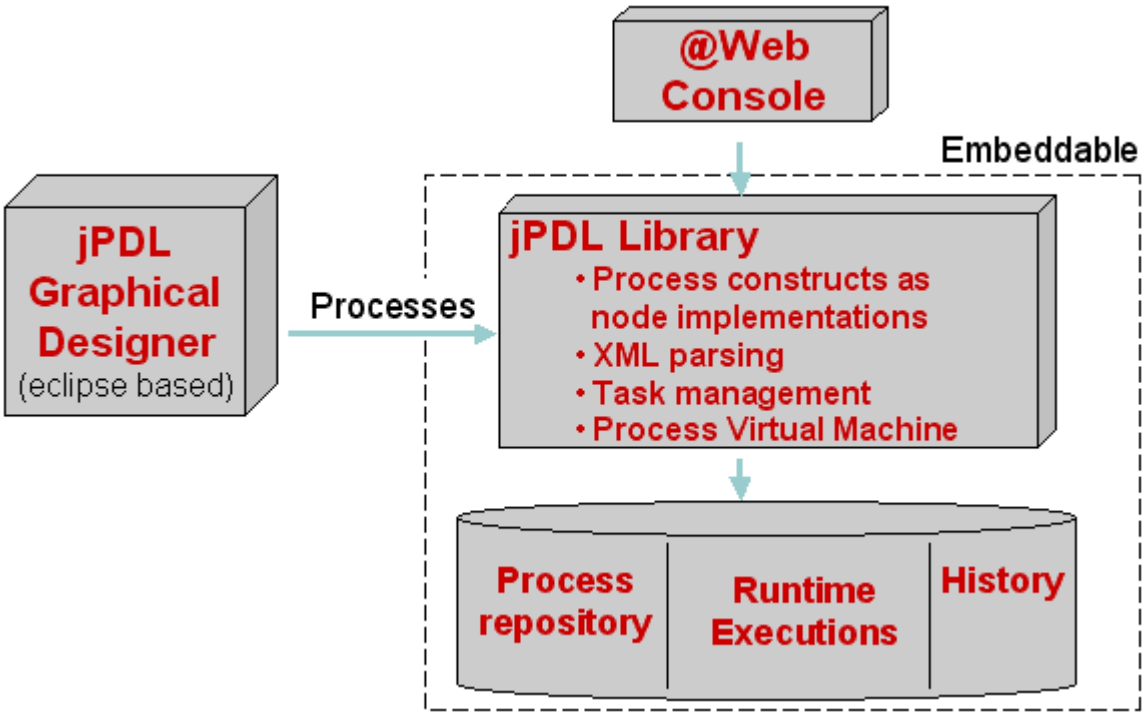


Figura 7: Componentes do jPDL (Fonte: JBoss, 2007)

Além disso, o jPDL combina a facilidade de desenvolvimento das aplicações de fluxo de trabalho, fluxos de processos corporativos e fluxos de página de aplicações web com um mecanismo de processos flexível e escalável. Integrado com o jBPM, permite a rápida e fácil implementação de aplicações que operam em conjunto, podendo oferecer uma estrutura flexível para a criação, coordenação e monitoramento dos processos corporativos (JBoss, 2007).

Para que isso aconteça, o jBPM inclui vários nós de processos jPDL que são utilizados para criar gráficos de processos de forma declarativa. Esses nós oferecem as funcionalidades necessárias para a criação de processos análoga à notação BPMN. Esses nós possibilitam a execução fundamental dos processos,

assim como permitem a presença de código Java para ampliar a lógica do processo e atender as necessidades específicas da aplicação.

3. Processo de desenvolvimento de software

A utilização de processos de software tem sido um fator importante para o sucesso de empresas produtoras de sistemas de informação e fábricas de software. Esses processos podem melhorar significantemente a qualidade desses sistemas, podendo medir e avaliar o desempenho dos setores relacionados ao desenvolvimento.

Um processo de software pode ser entendido como um conjunto estruturado de atividades com a finalidade e objetivo de desenvolver um sistema com qualidade. Assim, são tarefas e resultados associados que produzem um software (Sommerville, 1995).

Para Pressman (2004), processo de software são passos preditos – ou um guia – com o objetivo de ajudar a criar resultados de alta qualidade em tempo otimizado. O autor ainda complementa sobre a importância do processo devido ao aumento de estabilidade, controle e organização para as atividades que, se não controladas, podem se tornar caóticas.

Além disso, o autor sugere que para que possamos obter uma compreensão sobre o que é um software, devemos examinar as suas características do processo que o tornam diferente do restante das coisas que são construídas pelo restante da indústria. Ele ilustra isto com um comparativo entre o processo de construção do software e do hardware: enquanto que com o hardware as nossas idéias são traduzidas em formas físicas (assim como circuitos elétricos, impulsos magnéticos, etc.), com o software isto não é possível, uma vez que ele é um elemento de um sistema lógico.

Dentro deste cenário, Pressman cita três características do processo de construção de um software que demonstra esta diferença e as dificuldades relativas a este processo:

- **O software é desenvolvido ou projetado por engenharia, não manufaturado no sentido clássico**: Para atingir um bom nível de qualidade na construção de um software, assim como nos demais produtos manufaturados, um bom projeto é essencial. Entretanto,

quando começa a construção do software, as características da produção tornam-se diferentes de outras manufaturas. No desenvolvimento do software, os custos estão concentrados no trabalho de engenharia, fazendo com que nos projetos de software a gestão de projetos não seja da mesma forma como se fossem projetos de manufatura.

- **O software não se desgasta**: Diferente do hardware, que sofre com os efeitos do ambiente no qual se encontra, o software não se desgasta com o passar dos anos. O que acontece no software pode ser denominado de “deteriorização”, já que ao longo de sua existência o sistema passa por diversas modificações. Com esse conceito, a partir de cada modificação, outros problemas e falhas podem ser encontrados, fazendo com que o software passe por inúmeras evoluções.
- **A maioria dos softwares é feito sob medida, em vez de ser montada a partir de componentes existentes**: Ao contrário da construção e montagem do hardware, que possui diversos componentes disponíveis para manufatura, a maioria dos softwares não possui, salvo exceções, um catálogo de componentes que possam ser simplesmente combinados, tendo como produto final um programa. Pressman ainda conclui que “Ainda que muita coisa tenha sido escrita sobre reusabilidade de software, estamos apenas começando a ver as primeiras implementações bem-sucedidas do conceito”.

Pressman sugere que processo de software pode ser caracterizado como mostrado na Figura 8, definindo um pequeno número de tarefas que são aplicadas a todos os processos de software, não importando seu tamanho ou complexidade. O autor ainda explica que essas tarefas são relacionadas ao desenvolvimento do software, entrega do projeto, pontos de qualidade, podendo ser adaptada pela característica do software ou pela equipe de projeto.

Finalmente, Pressman dá importância às atividades de apoio – ou “guarda-chuva”, como gerência de configuração e gestão de projetos para o sucesso do processo.

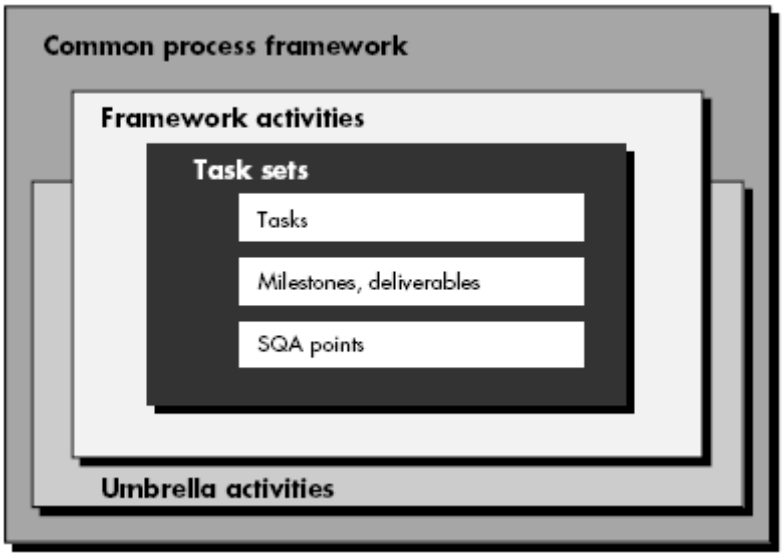


Figura 8: Arquitetura Comum para Processo de Software (Fonte: Pressman 2004)

3.1. Ciclo de vida de Software

Para Mitchell (2000), ciclo de vida de software representa as fases que um projeto do software possui a fim de definir e/ou projetar as exigências do software, executar as mudanças do programa, testar o sistema para assegurar-se de que as mudanças sejam exatas, e para instalar as mudanças em sistemas que também já estejam rodando.

Segundo Pressman (2004), para cada atividade do ciclo de vida existem tarefas básicas e objetivos definidos. Estas atividades fazem parte do conjunto mínimo para conceber um produto de software.

A busca por uma mistura otimizada de processos tem resultado em diferentes padrões na história do desenvolvimento de software. Um dos primeiros a serem publicados foi o padrão ISO/IEC 12207, que foi proposto em 1988 e publicado em Agosto de 1995. Foi criado para estabelecer uma arquitetura comum e internacional para adquirir, fornecer, desenvolver, operar, e manter sistemas.

De acordo com a norma, os ciclos de vida podem ser agrupados em três classes:

- Primário: primeiros a serem acionados;
- Suporte: processos de suporte para funções especializadas;
- Organizacional: pode ser usada para estabelecer, controlar e fornecer um processo organizacional.

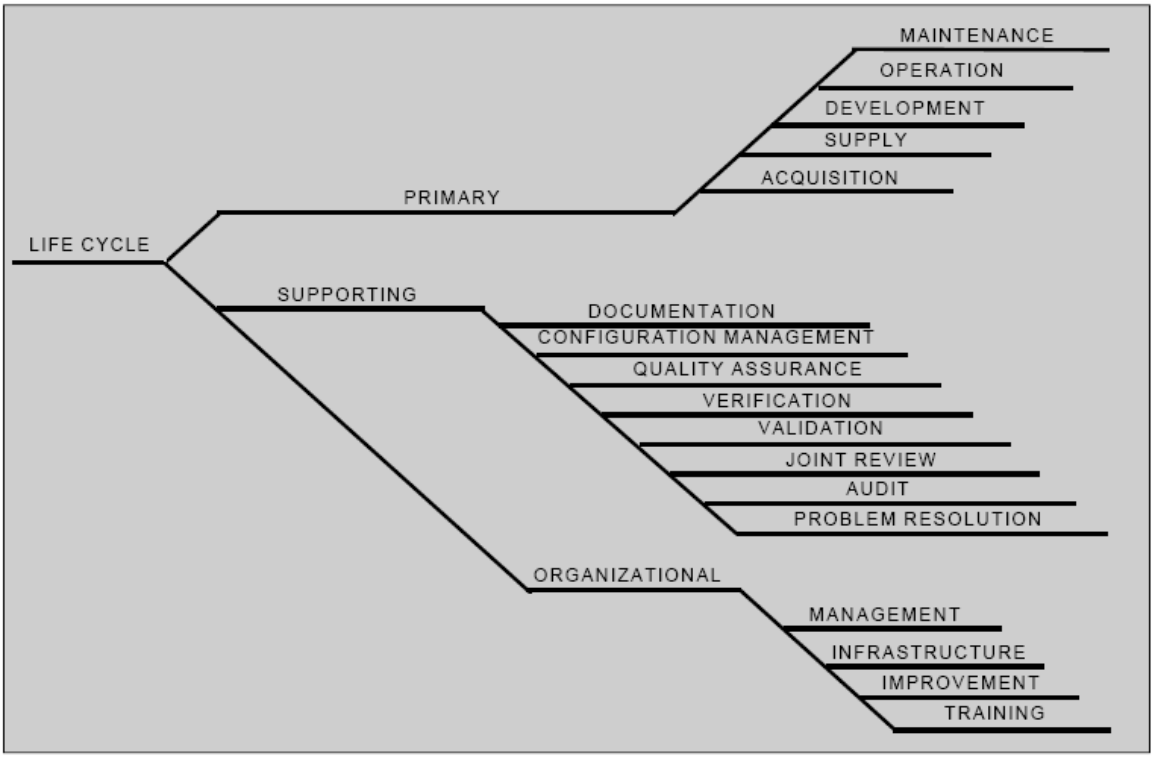


Figura 9: Ciclo de vida de Software (Fonte: ISO/IEC 12207)

Cada fase dentro dos ciclos de vida pode ser dividida em diferentes atividades. Devido ao nível de detalhamento desse padrão, este tópico tem como objetivo explicar apenas as atividades mais comuns para o processo de desenvolvimento de software.

Apesar de parecer com uma seqüência linear e obrigatória, muitos autores apontam que não existe uma ordem simultânea para cada atividade do processo como a própria realidade pode demonstrar.

3.1.1. Análise de requisitos

A extração dos requerimentos de um sistema é, normalmente, a primeira tarefa de sua criação. Sua natureza é geralmente iterativa devido às constantes mudanças de requisitos no projeto. No desenvolvimento de um sistema que possui interações entre pessoas, hardware ou componentes de software, a atividade de análise de requisitos descreve todos os requerimentos que o sistema deve ter, inclusive a alocação de funcionalidades para essas pessoas, hardware ou componentes de software (IEEE 1074 (1997)).

Para a norma ISO/IEC 12207, a análise de requisitos descreve a fase de entendimento dos requisitos e solicitações do sistema. Com isso, pode-se obter e definir os requerimentos, solicitações e necessidades do cliente através de sua solicitação direta ou através de outras formas como revisão da proposta de negócio, objetivos operacionais, ambiente de hardware e outros documentos.

Além disso, a norma diz que é imprescindível entender as expectativas do cliente, assegurando que tanto o cliente quanto o fornecedor entendam os requisitos da mesma forma. É necessário obter um acordo dos requisitos analisados e delegar as tarefas entre as equipes que irão desenvolver o trabalho em relação aos requisitos do cliente.

De acordo com Pressman (2004), até as atividades de análise de requisitos devem estar presentes na gerência de configuração para controlar todas as mudanças feitas nos requisitos do cliente. Através desse controle, assegura-se que o resultado de mudanças tecnológicas e de necessidades do cliente serão identificados e os impactos de introdução dessas mudanças serão avaliados.

Embora o cliente, provavelmente, acredite saber o que software deva fazer, esta tarefa exige habilidade e experiência tanto no negócio quanto em engenharia de software, afim de reconhecer a falta de informações e inconsistências nos requisitos.

3.1.2. Especificação

De acordo com Pressman (2004), especificação – também conhecida como design ou projeto – significa a representação de alguma coisa que pode ser construída. Isto pode ser traçado pelos requisitos do cliente e, ao mesmo tempo, avaliado por um conjunto de critérios de “boas práticas”.

Já na norma ISO/IEC 12207, desenvolve-se uma especificação de software para cada componente do sistema. Esses componentes devem ser refinados em níveis mais baixos, contendo unidades de software que possam ser codificadas, executadas e testadas. A especificação das interfaces deve permitir a implementacão sem a necessidade de informação adicional. Durante essa fase, se for necessário, deve ser feita a modificações e atualizações na documentação do usuário. É importante definir e documentar o planejamento de testes e o cronograma para testar unidades de software.

A norma também diz sobre as necessidades de avaliação deste detalhamento, considerando os critérios listados a seguir:

- Rastreabilidade para os requisitos do item de software;
- Consistência externa com o projeto da arquitetura;
- Consistência interna entre os componentes e unidades de software;
- Adequação dos métodos e padrões de projeto utilizados;
- Viabilidade dos testes;
- Viabilidade da operação e manutenção.

Para a Jacobson (1999), as finalidades dessa atividade são:

- Transformar os requisitos em um design do sistema a ser criado.
- Desenvolver uma arquitetura sofisticada para o sistema.
- Adaptar o design para que corresponda ao ambiente de implementação, projetando-o para fins de desempenho.

3.1.3. Arquitetura de Software

A arquitetura de um sistema representa uma representação abstrata do software. Ela está relacionada à garantia de que o sistema irá ao encontro de requisitos do produto, como também assegurar que futuros requisitos possam

ser analisados e atendidos. A etapa da arquitetura também direciona as interfaces entre os sistemas e outros produtos de software, como também com o hardware básico ou com o sistema operacional.

A arquitetura de software é um conceito de fácil compreensão e que a maioria dos engenheiros entende de modo intuitivo, especialmente quando se tem um pouco de experiência. No entanto, é difícil defini-lo com precisão. Em particular, é difícil desenhar uma linha bem definida entre a atividade de especificação e a arquitetura — a arquitetura é um aspecto da especificação que se concentra em alguns recursos específicos.

David Garlan e Mary Shaw (1994) sugerem que a arquitetura de software é um nível de projeto voltado para questões que vão: “além dos algoritmos e das estruturas de dados da computação. A projeção e a especificação da estrutura geral do sistema emergem como um novo tipo de problema. As questões estruturais incluem organização total e estrutura de controle global; protocolos de comunicação, sincronização e acesso a dados; atribuição de funcionalidade a elementos de design; distribuição física; composição de elementos de design; escalonamento e desempenho; e seleção entre as alternativas de design.”

Já no artigo da IEEE (1998) define a arquitetura como “o conceito de nível mais alto de um sistema em seu ambiente”. Ele também abrange à integridade do sistema, às restrições econômicas, às preocupações estéticas e ao estilo. Ele não se limita a um enfoque interno, mas leva em consideração o sistema como um todo em seu ambiente de usuário e de desenvolvimento, ou seja, um enfoque externo.

Para Jacobson (1999), a arquitetura de um sistema de software (em um determinado ponto) é a organização ou a estrutura dos componentes significativos do sistema que interagem por meio de interfaces, com elementos constituídos de componentes e interfaces sucessivamente menores.

3.1.4. Implementação

A atividade de implementação é a transformação de um projeto detalhado para um código de uma linguagem de programação. Esta atividade produz o

código executável, o banco de dados (se aplicável) e a documentação necessária para a manutenção, integrando-os para o funcionamento do sistema. Vale ressaltar que a fase de implementação deve ter, durante toda sua concepção, padrões apropriados de código software (IEEE 1074 (1997)).

Já a norma ISO/IEC 12207 complementa que a atividade de implementação consiste na definição ou seleção de um modelo de ciclo de vida de software adequado ao escopo, magnitude e complexidade do projeto e na documentação dos resultados, de acordo com:

- O processo de documentação;
- Colocação dos resultados sob o processo de gerência de configuração;
- A execução do controle de alterações, de acordo com ele;
- A documentação e resolução de não-conformidades e problemas encontrados no software e nas tarefas, de acordo com o processo de resolução de problema;
- Execução dos processos de apoio, conforme especificado no contrato;
- Seleção, adaptação e utilização de padrões, métodos, ferramentas e linguagens de programação de computador;
- Desenvolvimento dos planos para conduzir as atividades do processo de desenvolvimento.

De acordo com Jacobson (1999), esta atividade pode assumir objetivos adicionais:

- Definir a organização e padronização do código em termos de subsistemas de implementação organizados em camadas;
- Implementar o código em termos de componentes (arquivos-fonte, binários, executáveis e outros);
- Testar os componentes desenvolvidos como unidades;
- Integrar os resultados produzidos por implementadores individuais (ou equipes) ao sistema executável.

3.1.5. Teste

Testes são especialmente feitos onde tenham sido codificados por dois ou mais engenheiros trabalhando juntos. A realização de testes de funcionalidades verifica a presença de erros e comportamentos inadequados nas funções e componentes básicos do sistema.

Baseado na comparação entre os resultados atuais e esperados, além da concordância com os critérios necessários de funcionamento do sistema, a atividade de teste pode ser iterativa, pois muitos testes são realizados durante o ciclo de vida do software (IEEE 1074, 1997). A norma ainda diz que cada anomalia ocorrida durante a fase de testes deve ser reportada, além do impacto desta na validade do teste.

Para Jacobson (1999), a atividade de teste atua em vários aspectos como uma provedora de serviços para as outras atividades. Os testes enfatizam principalmente a avaliação da qualidade do produto, realizada através de várias práticas centrais, tais como:

- Localizar e documentar defeitos na qualidade do sistema;
- Avisar de forma geral sobre a qualidade observada no software;
- Validar as suposições feitas nas especificações de design e requisito através de demonstração concreta;
- Validar as funcionalidades do software conforme projetadas;
- Verificar se os requisitos foram implementados de forma adequada.

Jacobson (1999) ainda traz uma diferença interessante entre a fase de teste e as outras atividades: enquanto as outras enfatizam a abrangência, o teste enfatiza a deficiência. O teste desafia as suposições, os riscos e as incertezas inerentes ao trabalho de outras atividades, tratando essas questões por meio de uma demonstração concreta e uma avaliação imparcial. O desafio é evitar dois extremos potenciais: uma abordagem que não avalie o software de forma adequada e efetiva, expondo seus problemas e pontos fracos, e uma abordagem que seja inapropriadamente negativa ou destrutiva. Com a adoção de uma abordagem tão negativa, talvez a fase de teste nunca considere

aceitável a qualidade do software, e provavelmente alienará o esforço de teste aplicado.

3.1.6. Documentação

Uma importante tarefa é a documentação do projeto de software para futuras manutenções e evolução do sistema.

Assim como a atividade de teste, a documentação do software deve abranger todo o ciclo de vida do sistema sendo um conjunto de documentos gerados pelas outras atividades do processo. O propósito desta atividade é fornecer as informações necessárias para os desenvolvedores e usuários do sistema (IEEE 1074, 1997).

3.1.7. Suporte e Manutenção

Essa fase tem como foco a mudança no sistema associada às correções dos erros, adaptações necessárias para o ambiente que o envolve e alterações devido às mudanças de requisitos do cliente.

Pressman (2004) salienta que nesta fase são reaplicados os passos de definição e desenvolvimento dentro do contexto do software existente. O autor ainda sugere quatro tipos de mudanças encontradas durante a fase de suporte:

- **Correção**: Mesmo tendo a melhor qualidade nas atividades vistas, é comum que o cliente encontre defeitos no software. Manutenções corretivas são feitas para solucionar os defeitos.
- **Adaptação**: Depois de um tempo, o ambiente original (ex.: CPU, Sistema Operacional, regras de negócio, produtos externos) onde o sistema se encontra está passível de mudanças. Manutenções adaptativas resultam em modificações para o software acomodar essas novas mudanças no ambiente.
- **Realce**: Enquanto do software é utilizado, o usuário/cliente reconhecerá funções adicionais que fornecerão novos benefícios para o sistema. Manutenções perfeccionistas estendem o sistema além dos originais requerimentos funcionais.

- **Prevenção**: O software pode se deteriorar devido a algumas mudanças, e por causa disso, manutenções preventivas devem ser conduzidas para servirem as necessidades dos usuários finais. Em essência, a manutenção preventiva faz alterações em programas que podem ser mais fáceis de serem corrigidos, adaptados e realçados.

A manutenção e melhoria de software lidam com a descoberta de novos problemas e requisitos. Não somente pode ser necessário adicionar códigos que combinem com o projeto original, mas determinar como o software trabalhará em algum ponto depois da manutenção estar completa, pode requerer um significativo esforço por parte de um engenheiro de software. A maioria das atividades de manutenção é para ampliar os sistemas para novas funcionalidades, as quais, de diversas formas, podem ser consideradas um novo trabalho. De acordo com diversos autores, cerca de dois terços do tempo de desenvolvimento de software é destinado para manutenção.

3.2. Modelos de processo de software

Há mais de uma década, vem se tentando encontrar um processo ou metodologia previsível e repetível que melhore a produtividade e qualidade. Alguns tentaram sintetizar e formalizar a tarefa aparentemente incontrolável de escrever um software. Outros aplicaram técnicas de gestão de projetos na construção de sistemas. Sem o gerenciamento de projeto, o processo do projeto de software pode facilmente sofrer atraso ou estourar o orçamento.

Segundo Pressman (2004), para solucionar problemas que a indústria de software vem passando, muitas metodologias foram criadas com a finalidade de otimizar o processo de desenvolvimento de software, assim como sua produtividade.

Como um grande número de projetos de software não atende as expectativas em termos de funcionalidades, custos, ou cronograma de entrega, ainda não existe um modelo de processo perfeito para todas as aplicações. Mesmo assim, aconselha-se seguir pelo menos um modelo do que nenhum.

Conceitualmente, um modelo de processo é escolhido baseado na natureza do projeto e a aplicação, métodos e ferramentas utilizadas, no cronograma, além da cultura da própria empresa (Pressman, 2004).

Nas seguintes seções, são discutidos três modelos de processos para a tecnologia de informação. Cada um representa uma tentativa de trazer uma ordem inerente para as atividades caóticas de desenvolvimento de sistemas. É importante recordar que cada um dos modelos foram aplicados na prática por diversas empresas de software, não se limitando a apenas conceitos acadêmicos.

3.2.1. Modelo tradicional

Alguns modelos de desenvolvimento de software estabeleceram níveis e/ou etapas a serem seguidas, com o intuito de aprimorar a produção de software. O modelo "cascata" foi o modelo precursor na modelagem do processo de produção de software e é considerado um dos maiores exemplos de modelo tradicional para desenvolvimento de sistemas.

De acordo com Pressman (2004), o modelo cascata, ou modelo linear seqüencial, é um processo visto como um fluxo seguindo diretamente para frente, sem interrupções (como uma cascata) através das diversas fases do ciclo de vida de software (Figura 10).

O termo “cascata” foi originado através do artigo de 1970 por W.W. Royce citando-o como o modelo de software mais antigo existente, argumentando seus riscos e propensão a falhas. Ironicamente, esse modelo defeituoso mereceu um destaque muito grande na época, apesar das críticas negativas do autor. Royce teve seus argumentos amplamente ignorados e muitas empresas o adotaram como metodologia oficial, tornando-se um dos modelos mais tradicionais de construção de software.

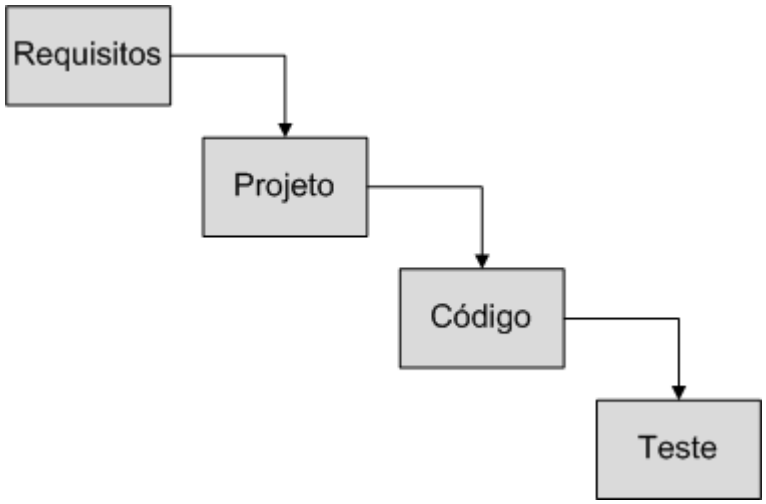


Figura 10: Modelo Cascata (Fonte: Adaptado de W.W. Royce)

Este modelo é muito simples entender seu funcionamento e uso. Royce (1970) caracteriza o modelo cascata como um processo com a qual cada fase deve ser terminada completamente antes da fase seguinte começar. No fim de cada fase, uma revisão ocorre para determinar se o projeto está no trajeto correto, dizendo se o fluxo do processo pode continuar ou não.

Apesar das vantagens de fácil gerenciamento e simples utilização, Pressman (2004) alerta alguns problemas encontrados quando esse modelo seqüencial é aplicado:

- Projetos reais raramente seguem um fluxo seqüencial que o modelo propõe. Embora o modelo linear pudesse acomodar iterações, mudanças podem causar confusões para o prosseguimento das fases.
- Freqüentemente, o cliente não consegue indicar explicitamente todos os requisitos do sistema. O modelo seqüencial linear requer isto além de ter que acomodar a incerteza natural que existe no começo de muitos projetos.
- O cliente deve ter paciência. Uma versão do sistema que esteja sendo fabricada não será avaliada até que o projeto seja entregue. Um erro grande, se não detectado até a entrega do projeto, pode ser desastroso.

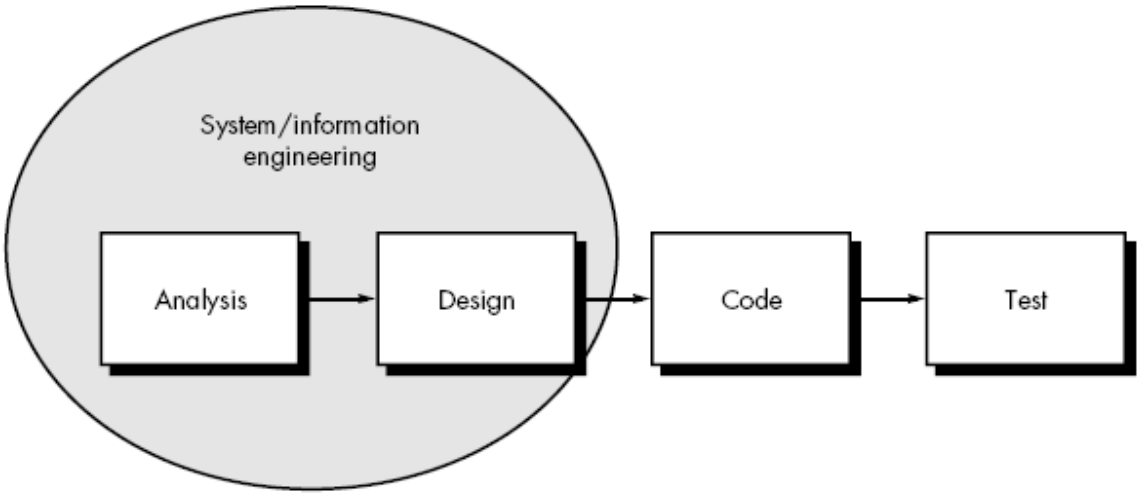


Figura 11: Modelo Sequencial Linear (Fonte: Pressman, Roger)

Para Hurst (2007), o modelo cascata não deve sofrer mudanças de requisitos para obter sucesso. Esta limitação tem conduzido uma censura crítica para esse modelo. O processo de desenvolvimento de sistemas envolve descobertas, além de que diferentes fases normalmente sobrepõem outras.

Muitas variações do modelo cascata foram criadas para que essas desvantagens fossem eliminadas. Essas variações começam a focar a distinção entre o modelo cascata e os processos iterativos, demonstrado a seguir.

3.2.2. Modelo Iterativo e Incremental

O modelo iterativo e incremental é um processo cíclico de desenvolvimento para atender as fraquezas do modelo cascata, o mais tradicional. Quando Royce explicou o modelo cascata em 1970, ele estava realmente escrevendo em defesa do processo iterativo. O modelo iterativo procede usando as lições aprendidas por cada fase para modificar os resultados da fase anterior. O desenvolvimento iterativo é também referenciado como desenvolvimento incremental (Hurst, 2007).

Royce (1970) conceitua o modelo em uma aproximação ao modelo cascata cujos múltiplos ciclos são desenvolvidos, fazendo com que o ciclo de vida seja se parecido com várias instancias do modelo cascata. Cada ciclo é dividido em pequenas iterações para facilitar o gerenciamento do projeto,

passando pelas mesmas fases de requisitos, projeto, implementação e testes do modelo tradicional (Figura 12).

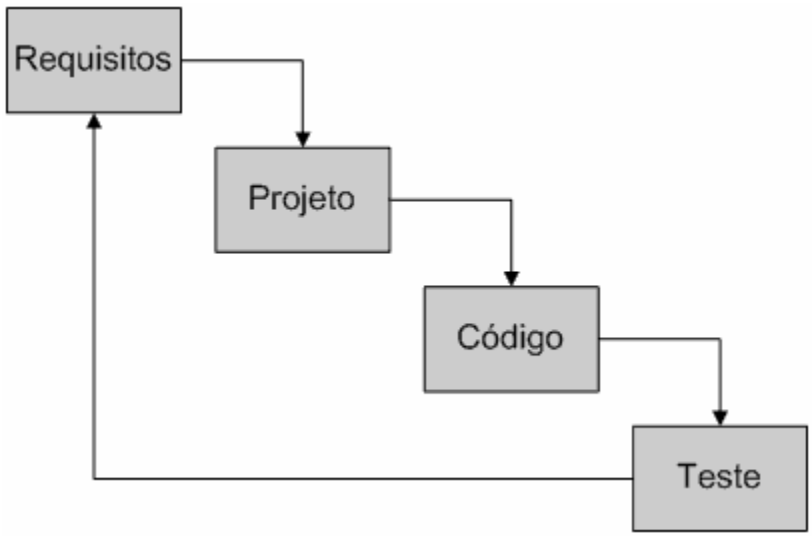


Figura 12: Modelo Iterativo e Incremental (Fonte: Adaptado de W.W. Royce)

Pressman (2004) diz que a idéia básica por atrás da visão iterativa e incremental é desenvolver, incrementalmente, um sistema de software, permitindo que o desenvolvedor tenha como vantagem o que foi aprendido durante o desenvolvimento da iteração anterior. As etapas chaves do processo eram começar com uma execução simples de um subconjunto de requisitos do software e realçar iterativamente na seqüência do desenvolvimento das versões até que o sistema inteiro esteja pronto. Em cada iteração, as modificações do projeto são feitas e as novas funcionalidades são adicionadas.

Muitas iterações podem ser necessárias antes que o projeto esteja completo. Como o modelo cascata, o processo iterativo começa com a fase de requisitos seguida pelas fases de projeto, implementação e testes. Depois deste primeiro ciclo, uma fase de avaliação é iniciada para avaliar os sucessos e falhas do trabalho feito. Os *feedbacks* do usuário, os insumos de performance, dificuldades de código, requisitos inconsistentes e inadequados, e ferramentas de analise do programas são usados com um conjunto de objetivos para o próximo ciclo de modelo. Depois de essas fases terem sido completadas pela segunda vez, mais uma fase de avaliação começa. Este processo se repete até que o projeto esteja finalizado (Hurst, 2007).

Hurst (2007) complementa que esta aproximação de fazer correções, mudar o projeto e ter novas descobertas dentro do processo, refletem mais como um software comercial é desenvolvido na prática. Projetistas podem perder requisitos e cometer enganos. Clientes, às vezes, têm apenas uma vaga noção do que eles realmente querem. Desenvolvedores freqüentemente acham que o projeto inicial não é adequado para refletir no sistema, como problemas de *hardware*, complexidade e necessidades do usuário. Fazendo com que as modificações sejam possíveis, o processo iterativo adiciona flexibilidade e melhores resultados dentro do processo de desenvolvimento do software.

Muitos modelos de desenvolvimento podem ser considerados iterativos. Neste trabalho será abordado o modelo em espiral e o processo unificado para explicar melhor o processo de desenvolvimento iterativo e incremental.

3.2.2.1. O Modelo em Espiral

O modelo em espiral pode ser considerado um modelo iterativo e incremental, com mais ênfase na análise de riscos. O modelo sugere uma espiral em que o componente angular representasse o progresso e o radiano da espiral representasse os custos das atividades.

Boehm (1988) foi o precursor do modelo em espiral na década de 80. Através de o modelo iterativo ser bem utilizado antes deste modelo, o autor foi o primeiro a explicar porque iterações são importantes para produzir um sistema para as expectativas do cliente.

De acordo com Pressman (2004), o modelo pode ser dividido dentro de um numero de atividades, também chamadas de regiões de tarefas. Tipicamente, existem de três a seis regiões dependendo, podendo variar de acordo com o autor ou projeto aplicado. Com isso, Pressman sugere seis possíveis regiões de tarefas, também mostrado na Figura 13:

- **Comunicação com o cliente**: esta tarefa estabelece uma comunicação efetiva entre desenvolvedores e o cliente;
- **Planejamento**: requer a (re) definição dos recursos, cronogramas, e outras informações relatadas ao projeto;

- **Análise de risco**: avalia os riscos técnicos e gerenciais;
- **Engenharia**: constrói uma ou mais representações da aplicação;
- **Construção e versão**: constrói, testa, instala e fornece suporte ao usuário (incluindo a documentação e treinamento);
- **Avaliação do Cliente**: obtém *feedback* do cliente baseado nas avaliações das representações criadas durante o estágio de engenharia e implementado durante o estágio de instalação.

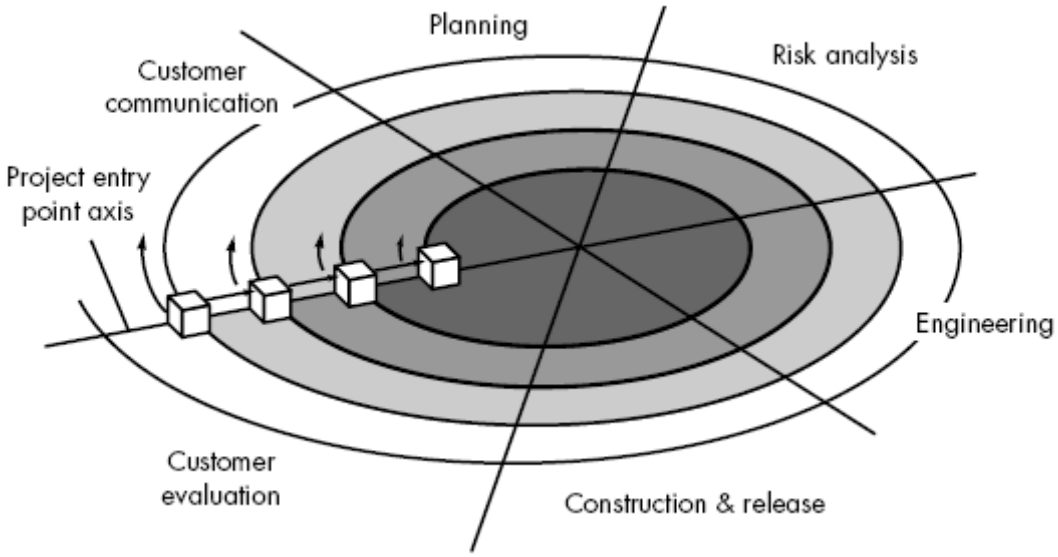


Figura 13: Modelo Espiral (Fonte: Pressman, Roger 2004)

3.2.2.2. Processo Unificado

O processo de desenvolvimento unificado ou denominado apenas de processo unificado é um popular conjunto de padrões de processo de desenvolvimento iterativo e incremental, afim de transformar requisitos do usuário em um sistema de software (Jacobson, 1999). Como ele é um processo baseado em componentes, o sistema pode ser construído em componentes interconectados via interfaces muito bem definidas.

O processo unificado não é apenas um processo como próprio nome diz, mas sim, um conjunto de padrões extensíveis com a qual deve ser customizado para as necessidades específicas de uma organização ou projeto.

O termo processo unificado é geralmente usado para descrever processos genéricos, incluindo elementos comuns para refinamentos do modelo de projeto. Foi usado pela primeira vez no livro *The Unified Software Development Process* publicado em 1999 por Ivar Jacobson, Grady Booch e James Rumbaugh. Desde então, vários autores afiliaram-se ao termo para descrever esses processos genéricos.

Jacobson (1999) define os principais aspectos que distinguem o processo unificado em: o direcionamento a casos de uso, a centralização da arquitetura e possuir o comportamento iterativo e incremental. Além disso, o autor complementa que o processo consiste na repetição de uma série de ciclos durante a vida de um sistema, como mostrado na Figura 14. Cada ciclo é concluído com uma versão do produto pronta para distribuição. Essa versão é um conjunto relativamente completo e consistente de artefatos, possivelmente incluindo manuais e um módulo executável do sistema, que podem ser distribuídos para usuários internos ou externos.

Figura 14: Modelo de Processo Unificado (Fonte: Jacobson, 1999)

A partir de uma perspectiva de gerenciamento, o ciclo de vida de software deste processo é dividido em quatro fases seqüenciais (Iniciação, Elaboração, Construção e Transição), cada uma concluída por um marco principal, ou seja, cada fase é basicamente um intervalo de tempo entre dois marcos principais. Em cada final de fase é executada uma avaliação para determinar se os

objetivos da fase foram alcançados. Uma avaliação satisfatória permite que o projeto passe para a próxima fase (Jacobson, 1999).

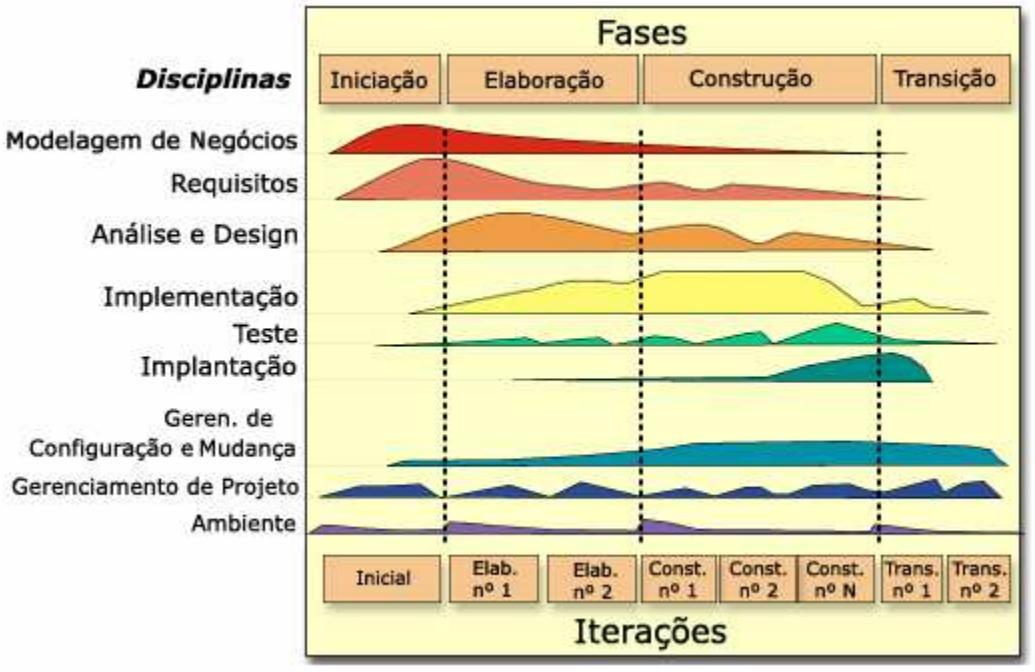


Figura 15: Fases e disciplinas do Processo Unificado (Fonte: Jacobson, 1999)

A figura acima lista na coluna à esquerda as atividades que devem ser realizadas em cada fase (requisitos, análise, projeto, implementação, testes, etc). As curvas não devem ser interpretadas literalmente, mas representam uma aproximação do esforço despendido com cada atividade em cada fase (Ericsson, 1998).

A seguir, são demonstrados alguns conceitos de Jacobson (1999) sobre as fases do processo unificado. Além disso, o autor diz que para um ciclo de evolução, as fases de iniciação e de elaboração seriam bem menores. Ferramentas que automatizam parte do esforço da fase de construção podem amenizar isso, tornando a fase de construção muito menor do que as fases de iniciação e de elaboração juntas (Figura 16).

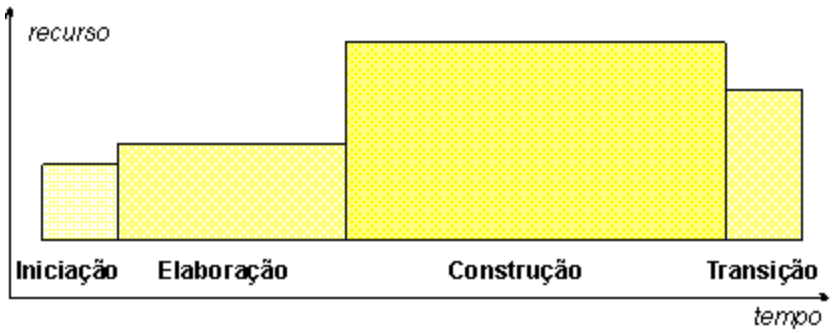


Figura 16: Gráfico das fases do processo unificado (Fonte: Jacobson, 1999)

Na fase de Iniciação, um modelo de casos de uso simplificado, que contenha os casos de uso mais críticos, poderá responder o que o sistema fará, bem como para cada tipo de usuário. Nesta fase a arquitetura é experimental, tipicamente apenas um esboço contendo os subsistemas mais cruciais. Os riscos mais importantes são identificados e priorizados, a fase de elaboração é planejada em detalhes e o projeto como um todo é estimado a grosso modo.

Durante a fase de elaboração, a maioria dos casos de uso do produto é especificada em detalhes e a arquitetura do sistema é projetada. A arquitetura é expressa em forma de visões de todos os modelos do sistema, implicando na existência de visões arquiteturais do modelo de casos de uso, do modelo de análise, do modelo de projeto, do modelo de implementação e do modelo de distribuição. Durante esta fase, os casos de uso mais críticos são realizados. No final da fase de elaboração, o gerente do projeto está em condição de planejar as atividades e estimar os recursos necessários para completar o projeto.

Já na fase de construção, o produto é efetivamente construído, começando a implementação definitiva do software. Embora a arquitetura do sistema encontre-se estável, os desenvolvedores podem descobrir maneiras melhores de estruturá-lo e podem sugerir pequenas mudanças aos arquitetos. No final desta fase, o produto contém todos os casos de uso que gerentes e clientes acordaram desenvolver nessa versão. Entretanto, é possível que ele não esteja totalmente livre de defeitos. Mais defeitos poderão ser descobertos e corrigidos durante a fase de transição.

Finalmente, a fase de transição cobre o período durante o qual o produto fica em versão beta. Nessa versão, um pequeno número de usuários experientes utiliza o produto e relata defeitos e deficiências. Desenvolvedores os corrigem e incorporam algumas das melhorias sugeridas. Esta fase envolve atividades como, treinamento de clientes, fornecimento de assistência on-line e correção de defeitos encontrados depois da distribuição.

3.2.3. Modelo de desenvolvimento ágil

O processo de desenvolvimento ágil foi criado durante os anos 90 com uma reação aos modelos pesados que estavam sendo usados naquela época. O processo originou-se da conclusão de que o modelo em cascata era burocrático, lento e contraditório comparado as formas usuais com que os engenheiros de software realizavam seus trabalhos.

Inicialmente, métodos ágeis eram conhecidos como métodos leves. Em 2001, membros proeminentes da comunidade se reuniram e adotaram o nome métodos ágeis. Embora cada envolvido tivesse suas próprias práticas e teorias preferidas, todos concordavam que, em suas experiências prévias, os projetos de sucesso tinham em comum um pequeno conjunto de princípios.

Com base nisso, eles criaram o *Agile Manifesto* (Manifesto para o Desenvolvimento Ágil de Software), freqüentemente chamado apenas de “manifesto ágil”. O termo desenvolvimento ágil identifica metodologias de desenvolvimento que adotam os princípios do manifesto ágil. Estes princípios são os seguintes:

- Indivíduos e interação entre eles mais que processos e ferramentas
- Software em funcionamento mais que documentação abrangente
- Colaboração com o cliente mais que negociação de contratos
- Responder a mudanças mais que seguir um plano

Os métodos ágeis representam uma família de métodos de desenvolvimento, melhor do que uma única representação ou metodologia. Eles enfatizam a comunicação entre todos os membros do projeto e incentivam a disponibilizar a equipe inteira em apenas um lugar. Clientes de desenvolvimento

ágil podem ser clientes reais, gerentes de projeto ou analistas de negócios. Em todos os casos, a participação do cliente é bem vinda e encorajada (Hurst, 2007).

De acordo com o Agile Manifesto (2001), os métodos ágeis tentam minimizar os riscos do projeto dividindo o projeto em pequenas iterações (assim como no modelo iterativo e incremental) que duram normalmente uma semana ou um mês. Cada iteração representa um projeto em miniatura, com planejamentos, requisitos, projeto, codificação, testes e documentação. O objetivo de um projeto ágil é liberar o sistema no fim de cada iteração, mesmo se essa versão não estiver completa. Além disso, as prioridades são reavaliadas no fim de cada ciclo de iterações.

Através desses conceitos, os métodos ágeis tentam satisfazer os clientes com rapidez, entregas contínuas de um sistema utilizável, entregues em semanas ao invés de meses. A principal medida de progresso é na construção do software. Uma comunicação próxima, face a face, é esperada entre desenvolvedores e as pessoas do negócio. O projeto deve adaptar-se as mudanças, em que a cada mudança tardia nos requisitos são bem vindas (Teles, 2005).

O desenvolvimento ágil permite que os envolvidos na construção do sistema sejam tratados como trabalhadores do conhecimento e conseqüentemente são estimulados a aprender durante todo o processo do projeto de software além de poder tomar decisões melhores com base nesse aprendizado. Sendo assim, aumenta-se a liberdade de expressão no trabalho pois não existe um processo rígido que impõe o que pode ou não pode fazer, desestimulando a criatividade da equipe.

No desenvolvimento tradicional, os especialistas do negócio conversam com os analistas, que abstraem e passam as informações adequadas aos programadores, que estão limitados a apenas escrever o código fonte do software. Esta especialização de atividades não é aconselhável pois é completamente desprovida de *feedback*. O analista tem toda a responsabilidade de criar o modelo de domínio, baseado somente nas informações fornecidas pelos especialistas do negócio. Eles não têm a oportunidade de aprender com o

desenvolvimento ou ganhar experiência com as versões iniciais do software (Evans, 2004).

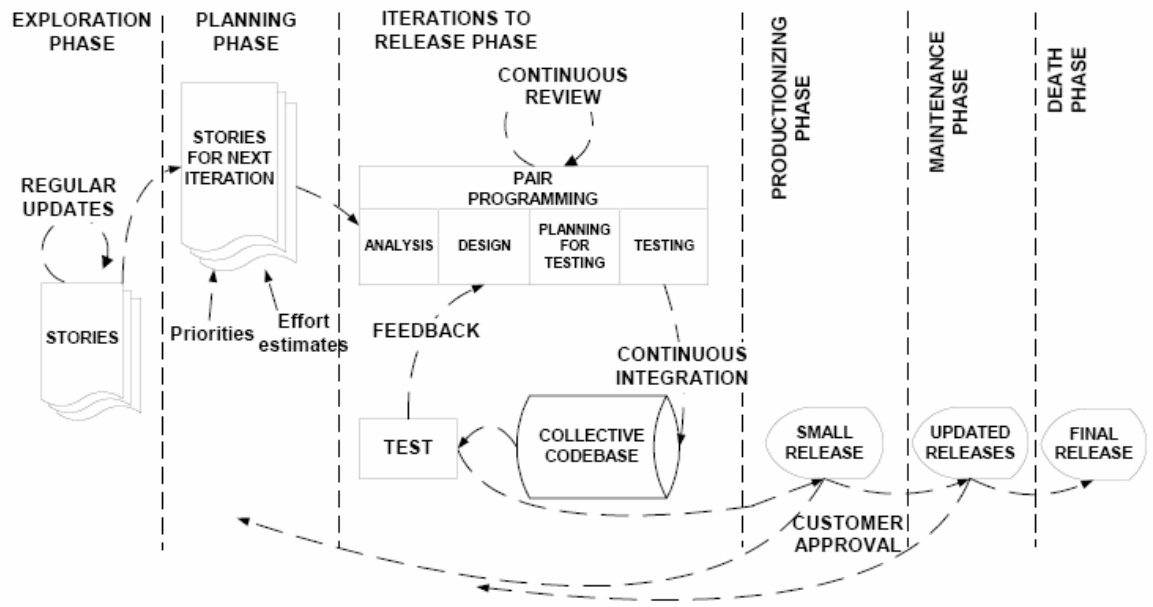


Figura 17: Ciclo de vida de um processo ágil (Fonte: Abrahamsson, 2002)

Devido a estas circunstâncias, esse modelo de desenvolvimento é direcionado intensivamente as pessoas. Evans (2004) conclui que trabalhadores do conhecimento rendem mais e melhor em ambientes que estimulam o uso intensivo da criatividade e do conhecimento. Escrever programas de computador é um trabalho tão criativo quanto escrever um livro, um artigo ou uma monografia. É uma atividade tipicamente intelectual, muito comum ocorrer idas e vindas, já que o aprendizado adquirido torna possível aos envolvidos perceberem maneiras cada vez melhores de fazerem seus trabalhos.

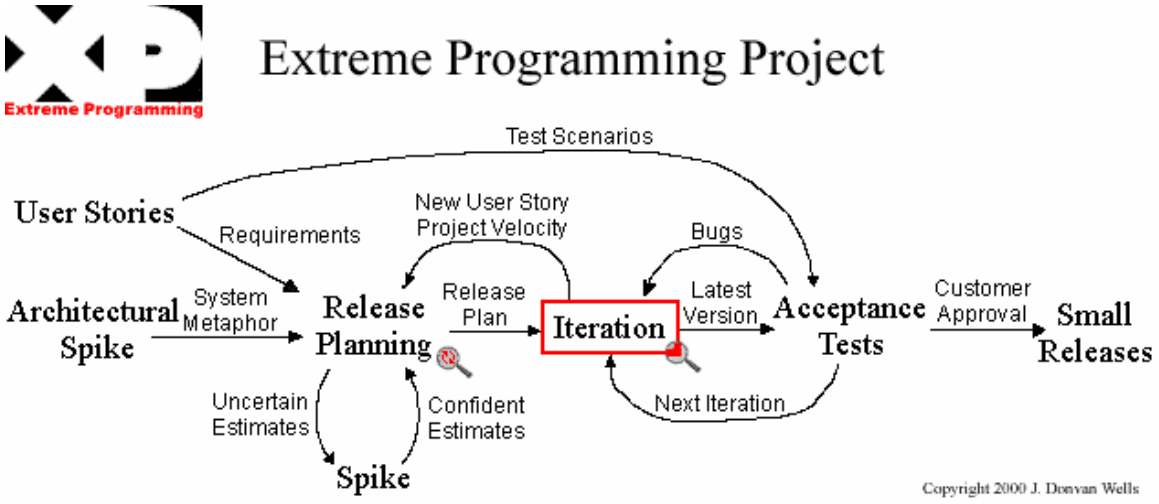


Figura 18: Exemplo de Projeto Ágil (Fonte: XP, 2000)

4. Análise e Resultados

Este capítulo tem como objetivo obter um estudo comparativo dos modelos de processos vistos, baseando-se nos critérios encontrados e analisados, afim de nortear a avaliação dos processos de desenvolvimento de software.

Para demonstrar a análise de alguns critérios de avaliação, foi necessária a implementação de um sistema de gestão de processos para desenvolvimento de software afim de conseguir um resultado mais próximo da realidade de empresas fornecedoras de sistemas de informação.

Com isso, Georgakopoulos et. al. (1995) identifica três etapas para a implementação de um sistema de gerenciamento de processos:

- Modelagem de processos e definição do workflow
- Reengenharia de processo
- Implementação do workflow

Segundo o autor, as etapas realizadas para o desenvolvimento e implementação deste tipo de sistema são semelhantes ao processo de desenvolvimento de um sistema de informações tradicional. Entretanto, é enfatizada neste caso a ênfase dada para a modelagem do processo através de técnicas de reengenharia de processos ou de melhorias incrementais antes da implementação da automação.

Deste modo, serão utilizados os conceitos de Georgakopoulos (1995) de implementação para a construção de um sistema simples de gestão de processos de software. Entretanto, para uma visão comparativa entre as metodologias dos processos na aplicação desse sistema, a parte de reengenharia será pouco explorada, afim de manter esses processos mais íntegros com os modelos originais.

4.1. Critérios de avaliação

Segundo Abreu & Abreu (2003), um sistema de informação gerencial deve ser avaliado através do contexto organizacional da empresa e da qualidade

técnica adotada. Sendo assim, diversas métricas poderiam ser usadas para determinar se o sistema avaliado realmente obteve sucesso. Algumas delas podem ser: a quantidade de acesso diário ao sistema, os objetivos propostos atingidos, o retorno financeiro obtido, satisfação do usuário, entre outros.

Para avaliar as metodologias de desenvolvimento de software com base em BPMS, foi feita uma análise dos critérios necessários para direcionar o estudo comparativo deste trabalho. Dessa forma, foi possível comparar os modelos citados na aplicação dos modelos de processo explorados na seção anterior.

Segundo Campos (1994), é possível classificar os critérios de avaliação de processos de negócios de duas formas diferentes: uma voltada para o resultado final do produto ou serviço e a outra no resultado do processo. De acordo com o autor, a primeira forma tende a ser medida pelas necessidades do cliente e suas expectativas. Já a segunda, relaciona como as atividades no processo foram elaboradas para o processo ser finalizado corretamente.

Dentro deste cenário, foram utilizados os seguintes critérios para a medição dos processos de desenvolvimento de software, segundo os conceitos de Campos (1994) e adaptados para o contexto do trabalho:

- Qualidade agregada: Medição da quantidade de falhas ou erros no produto, no caso o software. Além disso, pode-se medir na qualidade os impactos de satisfação e motivação dos colaboradores do processo para a conclusão do mesmo;
- Custos: Custos das atividades realizadas pelos colaboradores em cada uma das tarefas definidas pelo modelo de processos. Incluem-se também custos relacionados pela falta de qualidade como re-trabalho de tarefas, correções de erros, inspeção de código, etc.;
- Entrega: Tempo executado por cada tarefa do desenvolvimento, incluindo os atrasos e as horas adicionais trabalhadas.

Comparando-se as metodologias de processo de software, Smith (2001) diz que a maioria dos processos possui diversos elementos comuns que faz com

que uma comparação seja possível. Os fatores de avaliação propostos pelo autor podem ser divididos nas seguintes dimensões:

- Alocação de tempo e esforço por tarefa: Discute como cada tarefa é organizada temporalmente e alocada pelo nível de esforço para sua conclusão;
- Artefatos: Produtos gerados a partir das tarefas realizadas;
- Atividades: Discute o meio em que cada artefato do processo deve ser produzido;
- Papéis: Explora as posições de cada colaborador no processo;
- Fluxo de trabalho para as tarefas: Delineia as principais áreas de interesse de cada metodologia para a engenharia de software.

4.2. Modelagem dos Processos

Nas subseções seguintes será abordada a modelagem dos processos para cada modelo de software citado.

4.2.1. Modelo tradicional

Com base nos conceitos propostos por Royce (1970), a modelagem dos processos de desenvolvimento tradicionais obteve características muito semelhantes ao modelo proposto pelo autor. Um dos motivos para que isso aconteça foi a sua simplicidade e visão demasiadamente superficial, fazendo com que o diagrama se pareça bastante com o modelo cascata de desenvolvimento.

Foi atribuída na modelagem a utilização de sub-processos nas fases do modelo para que fique a cargo de qualquer organização implementar sua própria concepção nestes sub-processos.

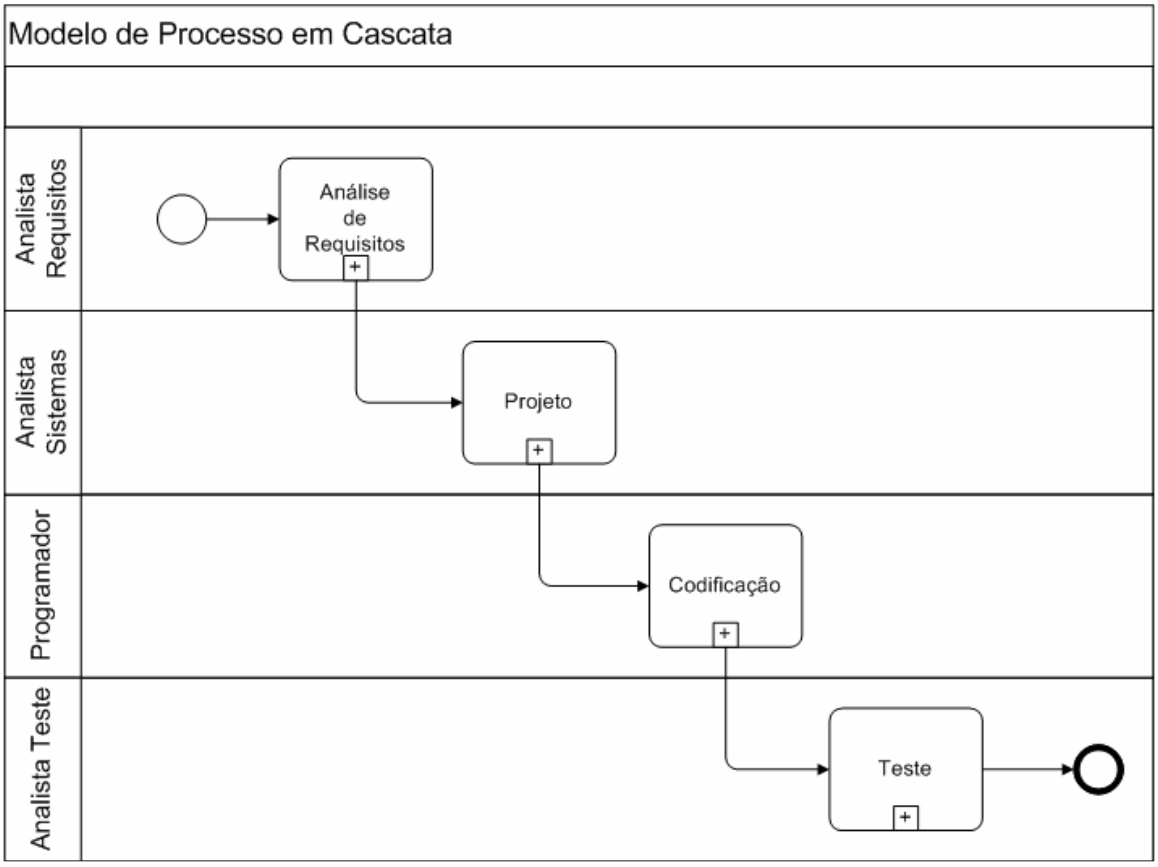


Figura 19: Modelagem do processo tradicional de software (Fonte: Autor)

4.2.2. Modelo Iterativo e Incremental

Para o modelo iterativo e incremental, a base para a modelagem foi o processo unificado, citado neste trabalho.

Com as fases de processos bem definidas pelo modelo unificado de processo proposto por Jacobson (1999), decidiu-se aplicar um macroprocesso abrangendo todas as fases citadas pelo autor. Por ser um modelo iterativo, cada sub-processo tem como característica a execução do mesmo diversas vezes, ocasionando um *looping* e agregando valor a cada iteração, já que também é considerado incremental.

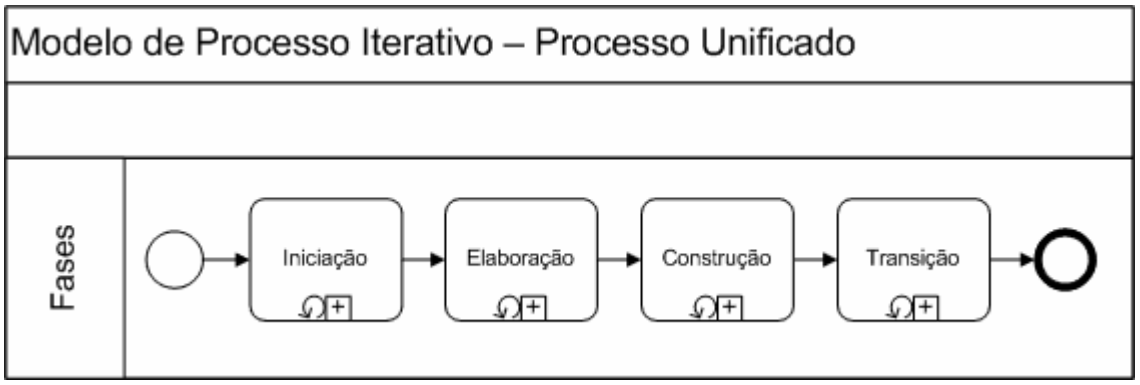


Figura 20: Modelagem do processo iterativo (Fonte: Adaptado de Jacobson, 1999)

Logo após a definição do macroprocesso, para cada fase foi feita uma modelagem específica, contendo um maior nível de detalhes.

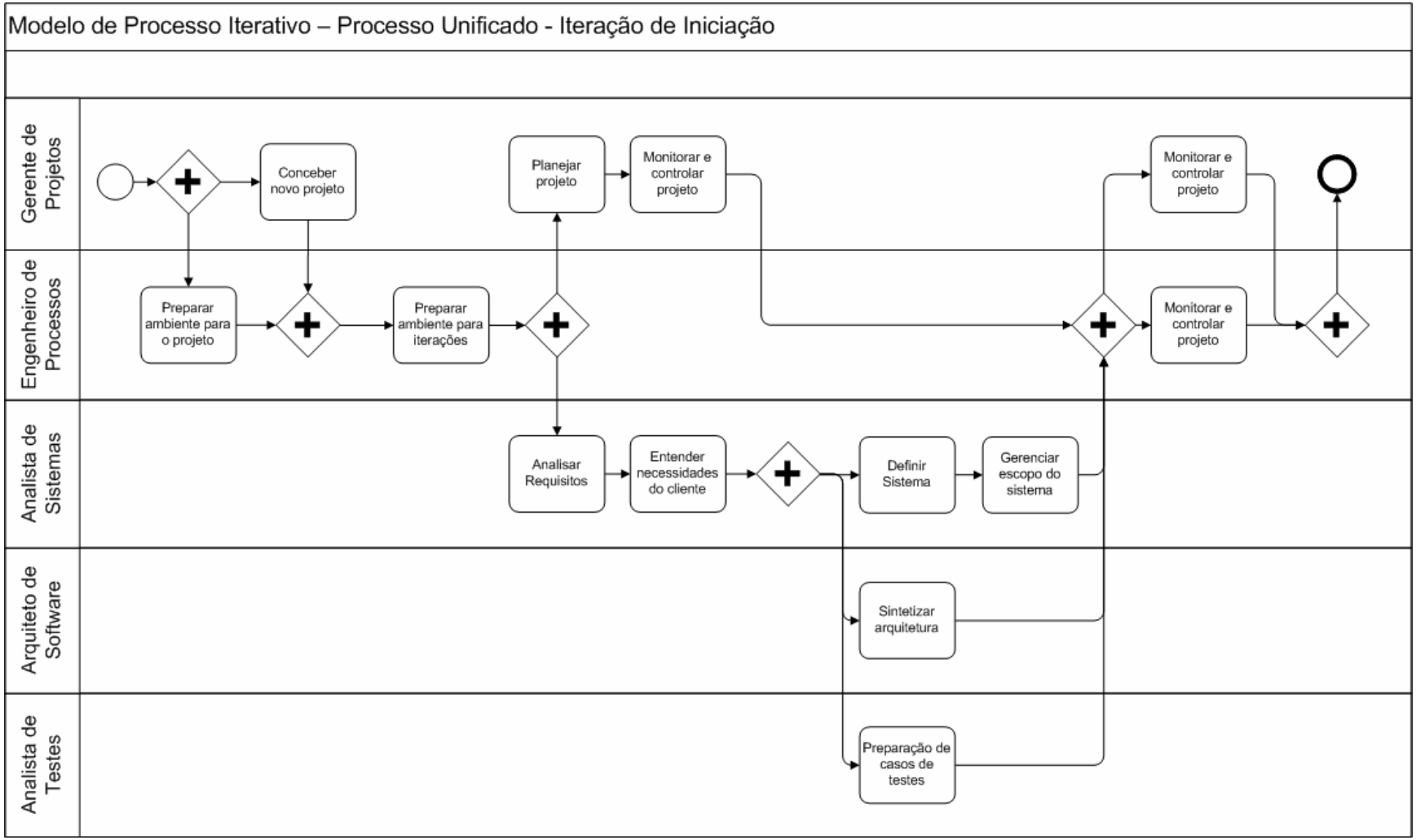


Figura 21: Modelagem do processo iterativo – Iniciação (Fonte: Adaptado de Jacobson, 1999)

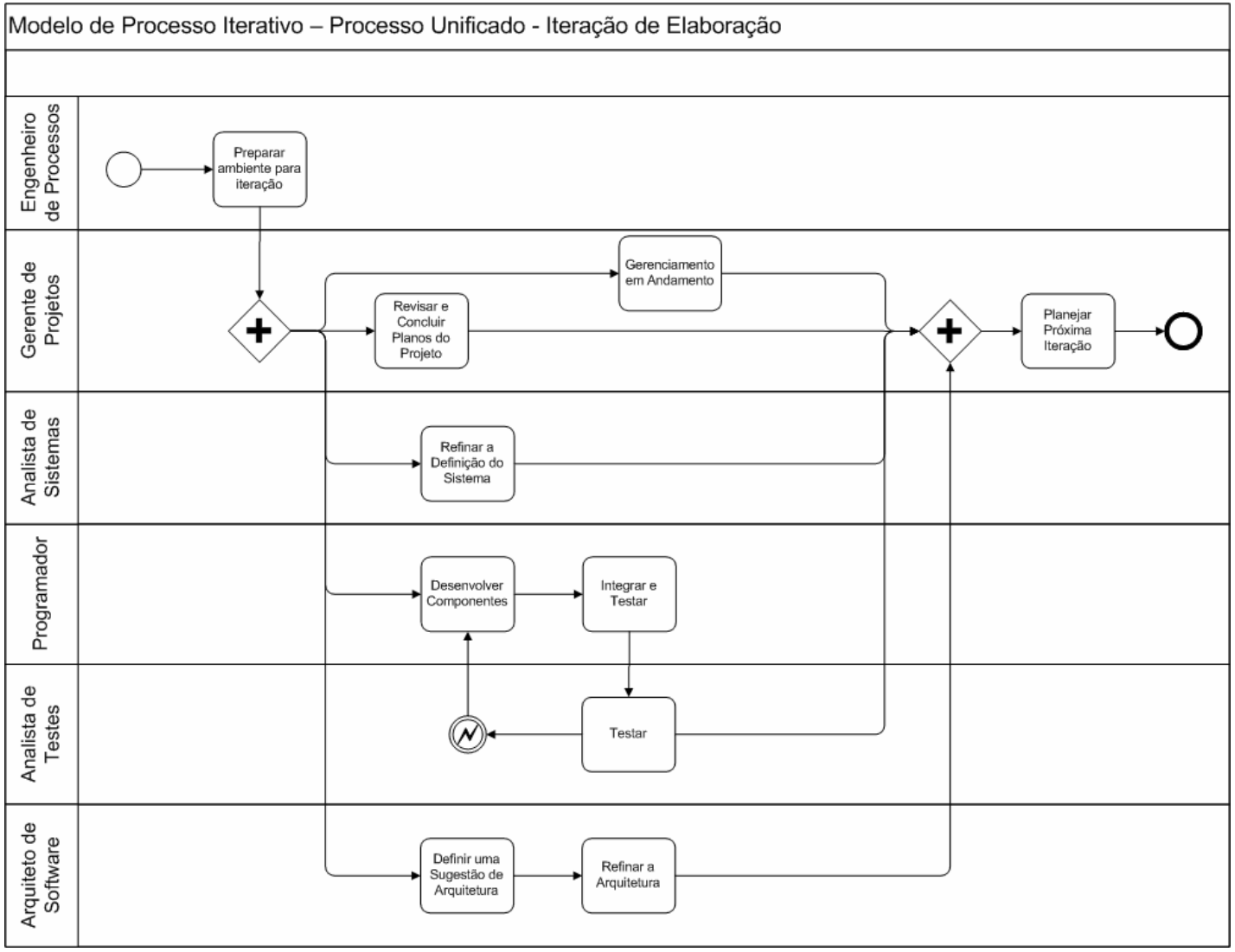


Figura 22: Modelagem do processo iterativo – Elaboração (Fonte: Adaptado de Jacobson, 1999)

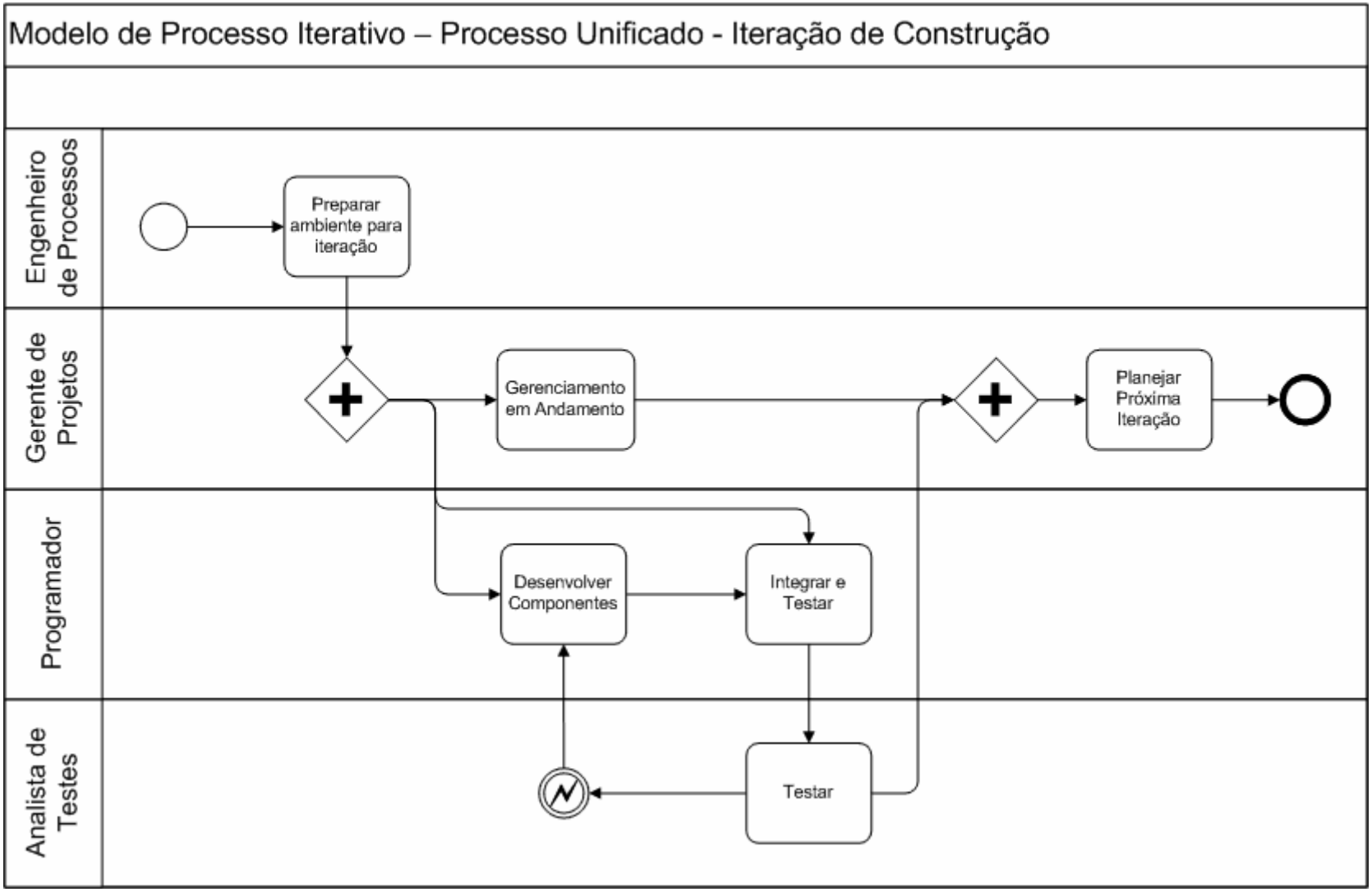


Figura 23: Modelagem do processo iterativo – Construção (Fonte: Adaptado de Jacobson, 1999)

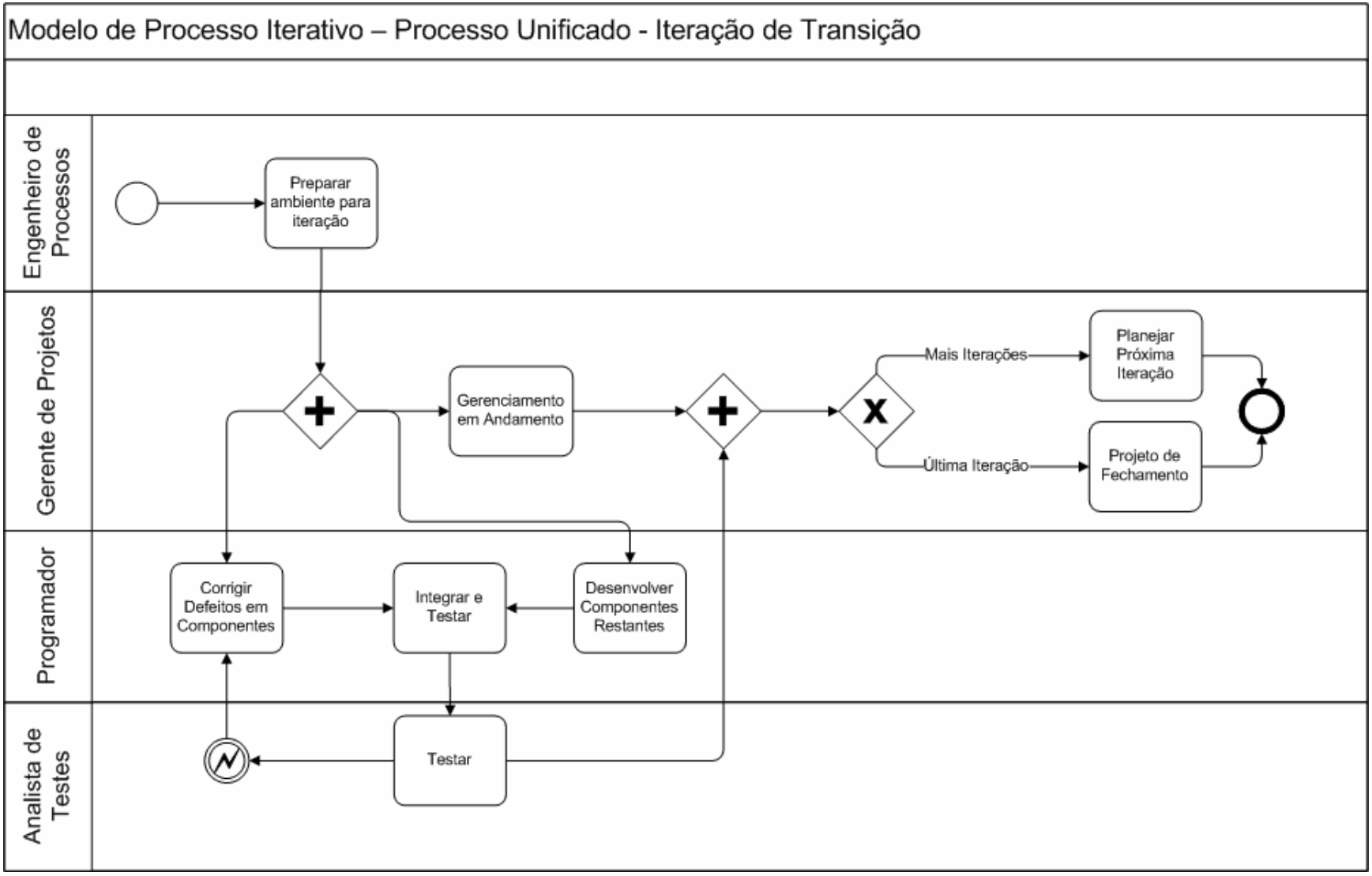


Figura 24: Modelagem do processo iterativo – Transição (Fonte: Adaptado de Jacobson, 1999)

4.2.3. Modelo de desenvolvimento ágil

Para a modelagem do modelo de desenvolvimento ágil, foi constatada a participação do cliente dentro do processo do software, característica marcante do modelo. Além disso, notou-se a característica iterativa e incremental nas atividades de implementação, fazendo com que tenha pequenos e repetitivos *loopings* durante estes procedimentos.

Por ser um modelo voltado para o código gerado, na modelagem percebeu-se o foco na atividade de implementação. Com isso, esta atividade e as atividades de testes podem ocorrer iterativamente podendo adicionar as práticas utilizadas de desenvolvimento ágil como a refatoração de código, integração contínua, programação em pares, etc.

No desenvolvimento ágil, pode existir a participação de um consultor para solucionar e ajudar nas questões técnicas de desenvolvimento. Para que isso exista na modelagem, foi atribuída uma tarefa com compensação, fazendo com que o consultor execute suas atividades e imediatamente após sua finalização, retorne o fluxo para quem solicitou sua participação no processo, no caso o analista ou desenvolvedor (Figura 26).

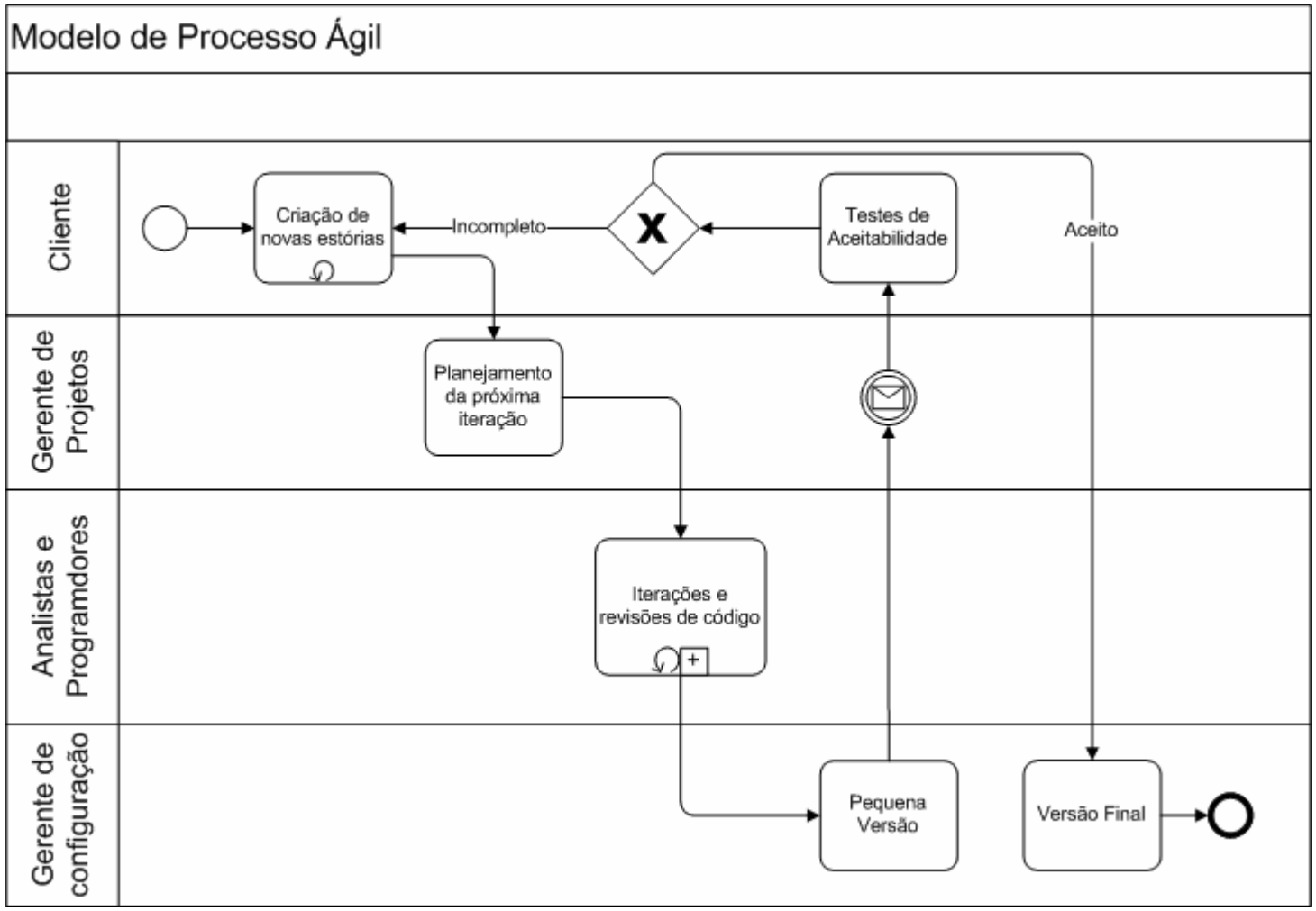


Figura 25: Modelagem do processo ágil (Fonte: Autor)

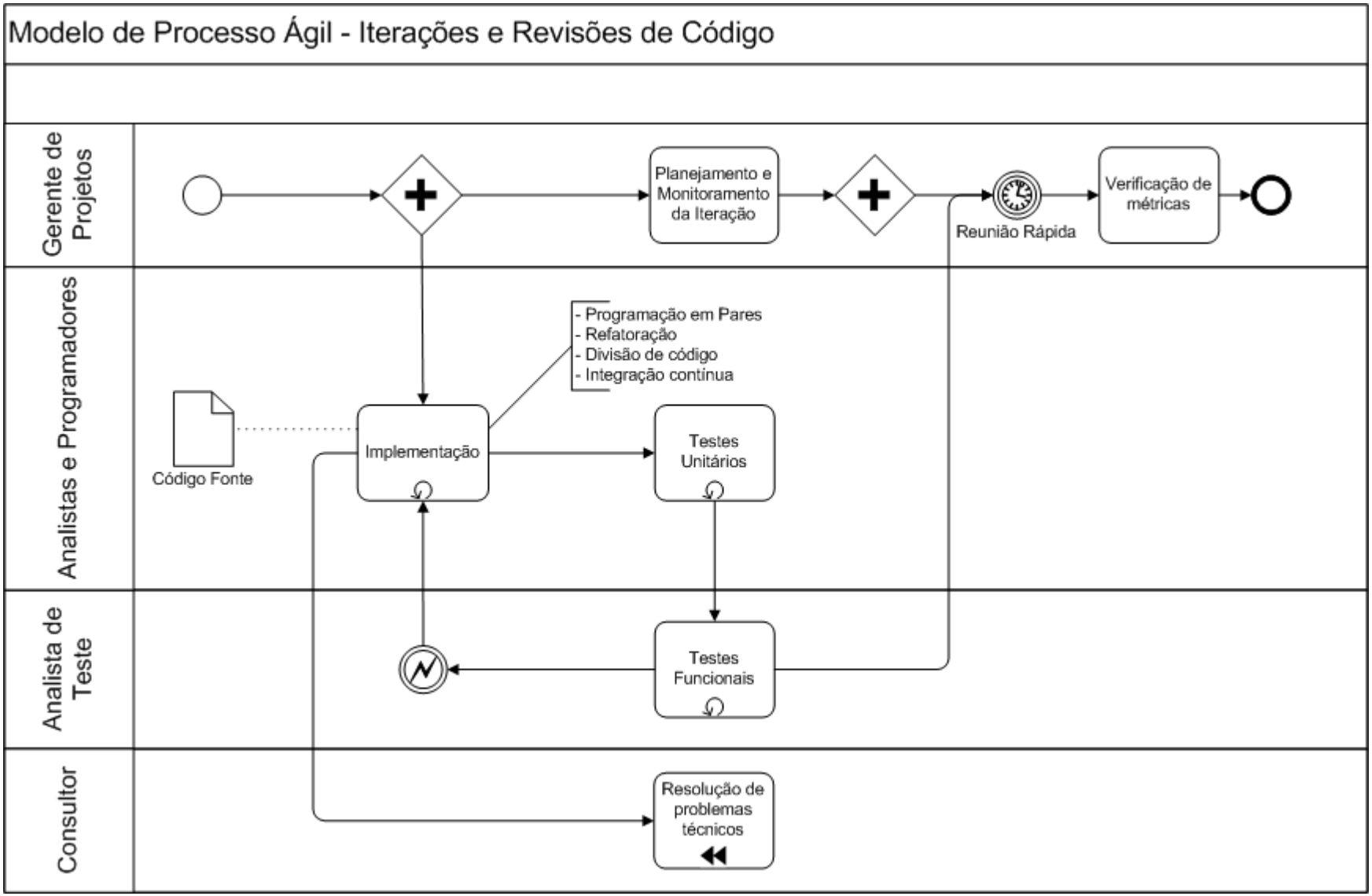


Figura 26: Modelagem do processo ágil - Iteração (Fonte: Autor)

4.3. Implementação do Workflow

Através de um estudo de similaridade com os requisitos propostos, foi necessária uma implementação própria de um sistema de workflow para que pudesse ser atendido todo o requerimento do trabalho.

Diversas ferramentas foram analisadas, umas direcionadas para abordagem *bottom-up* e outras para o *top-down*. As ferramentas *bottom-up* têm como característica a utilização de processos e sistemas já existentes, geralmente utilizadas para fins de automação de processos. Já as *top-down* realizam uma verdadeira reengenharia de processos antes mesmo de mencionar qualquer automação ou a criação do workflow, sendo a de maior destaque para clientes que não possuem processos claros ou bem definidos.

Para que este sistema seja construído, fez-se um levantamento amplo de ferramentas e frameworks baseado em critérios básicos, seguidos da eliminação das dificuldades encontradas na instalação, programação ou execução de exemplos simples de processos de negócio.

Alguns critérios podem ser listados abaixo para a escolha da ferramenta ou framework:

- Ter como linguagem básica, ser desenvolvida ou permitir integração com linguagens conhecidas ou familiares pelo desenvolvedor do workflow. Isso pode facilitar a adoção do mesmo, sem ter a linguagem de programação como barreira impeditiva para muitos casos de uso;
- Obter ferramentas que dê suporte ao desenvolvimento deste workflow, bem como um ambiente de desenvolvimento integrado (IDE) (Gurley, 2003);
- Preferencialmente ter o código fonte aberto ou não haver dependências com código fechado, com a possibilidade de ser customizável pelo desenvolvedor;
- Não estar infringindo exigências comerciais ou estar legalizado perante a lei. Para isso, deve-se obter a licença para utilização do software ou estar utilizando software livre;

- Utilizar plataformas de sistemas operacionais amigáveis, além de poder ser executado pela plataforma do desenvolvedor e do cliente;
- Ser produtivo ou utilizar artifícios produtivos, como configurações a partir de interfaces gráficas, fácil codificação e manutenção, ou complementos para ajudar no desenvolvimento;
- Possuir documentação extensa ou comunidade de desenvolvimento ativa.

Após a instalação, programação e execução de diversas ferramentas e frameworks, muitas delas apresentaram dificuldades ou não estavam de acordo com os critérios estabelecidos para a sua utilização no trabalho. Outra questão que dificultou a utilização de algumas ferramentas foi a sua expressividade, pois elas não podiam representar um ou mais casos dos processos estudados.

Devido a estes problemas, optou-se por utilizar o jBpm. Além de estar de acordo com os critérios estabelecidos, o framework continha a expressividade necessária para a sua utilização no trabalho, adequando-se a todas as expectativas da pesquisa.

Inicialmente, a implementação do workflow foi realizada através da ferramenta de edição de fluxos do próprio jBpm, gerando a definição jPdl automaticamente. Após o desenho dos fluxos, foram acrescentadas as variáveis mínimas necessárias para o seu bom funcionamento, assim como formulários necessários para a apresentação ou edição dessas variáveis. Nenhuma dessas tarefas ofereceu maiores dificuldades, uma vez que as variáveis de execução podem ser inseridas no fluxo através do próprio editor visual além de utilizar formulários que seguem padrões conhecidos.

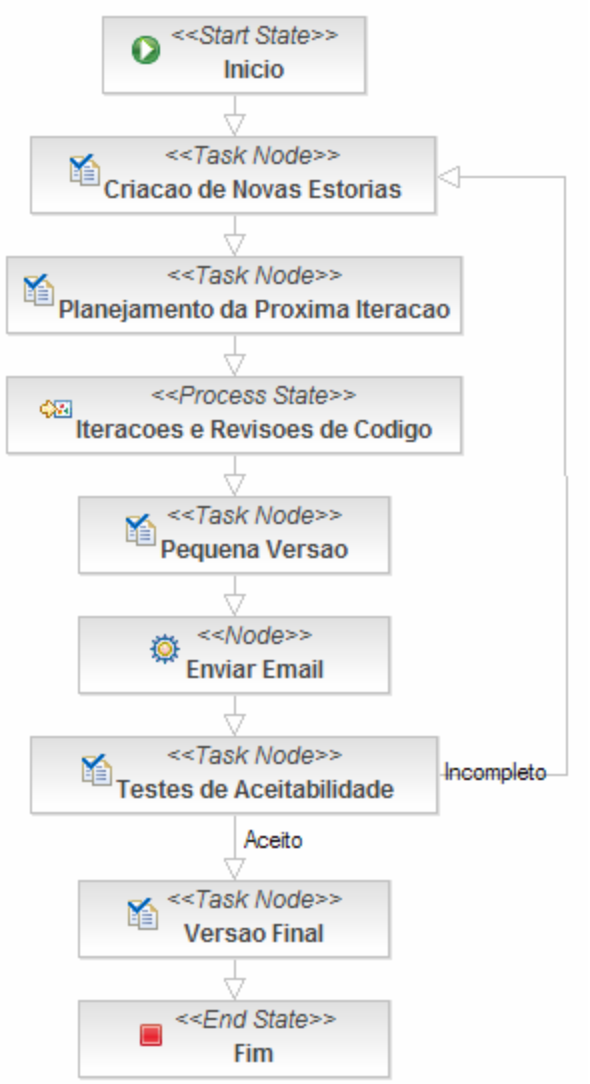


Figura 27: Definição do Modelo Ágil no editor visual do jBpm (Fonte: Autor)

Através do editor, foi possível implantar o processo facilmente, já que não foi necessário montar todo o arquivo xml manualmente. Na figura seguinte, é demonstrado o documento jPdl gerado através da ferramenta visual do jBpm.

```
<?xml version="1.0" encoding="UTF-8"?>
<process-definition
  xmlns=""  name="Modelo Agil">
   <swimlane name="Cliente"></swimlane>
   <swimlane name="Gerente de Projetos"></swimlane>
   <swimlane name="Analistas e Programadores"></swimlane>
   <swimlane name="Gerente de Configuracao"></swimlane>
   <start-state name="Inicio">
      <transition name="" to="Criacao de Novas Estorias"></transition>
   </start-state>
   <end-state name="Fim"></end-state>
   <task-node name="Criacao de Novas Estorias">
      <task name="Criar" swimlane="Cliente"></task>
      <transition name=" " to="Planejamento da Proxima Iteracao">
      </transition>
   </task-node>
   <task-node name="Pequena Versao">
      <task name="Implantar" swimlane="Gerente de Configuracao"></task>
      <transition name="" to="Enviar Email"></transition>
   </task-node>
   <task-node name="Versao Final">
      <task name="Implantar" swimlane="Gerente de Configuracao"></task>
      <transition name="" to="Fim"></transition>
   </task-node>
   <task-node name="Planejamento da Proxima Iteracao">
      <task name="Planejar" swimlane="Gerente de Projetos">
      </task>
      <transition name=" "
            to="Iteracoes e Revisoes de Codigo"></transition>
   </task-node>
   <process-state name="Iteracoes e Revisoes de Codigo">
      <transition name="" to="Pequena Versao"></transition>
   </process-state>
   <task-node name="Testes de Aceitabilidade">
      <task name="Testar" swimlane="Cliente"></task>
      <transition name="Aceito" to="Versao Final"></transition>
      <transition name="Incompleto"
            to="Criacao de Novas Estorias"></transition>
   </task-node>
   <node name="Enviar Email">
      <transition name="" to="Testes de Aceitabilidade"></transition>
   </node>
</process-definition>
```

Figura 28: Definição do Modelo Ágil em jPdl (Fonte: Autor)

Assim que as definições dos processos foram realizadas, foi necessária a execução dos mesmos para demonstrar sua aplicabilidade. Todos os processos foram desenvolvidos através de um sistema de testes e monitoramento do próprio jBpm, que possui diversas funcionalidades e utilitários para o gerenciamento dos processos implantados. Além disso, o aplicativo fornece

comandos úteis para simular e disparar eventos necessários para que ocorra a execução do fluxo corretamente.

Para simular os papéis e regras de permissão dos processos, foram criados alguns usuários com seus respectivos papéis nos fluxos de trabalho. Para cada modelo de processo de software estudado, foram atribuídos os papéis necessários para sua execução. Logo em seguida, alguns papéis em comum foram reutilizados em outros modelos, não influenciando na execução de processos distintos.

Figura 29: Tela principal do sistema e usuários disponíveis (Fonte: Autor)

Em algumas tarefas, foram criados formulários específicos com a finalidade de agregar as informações necessárias para a continuidade do fluxo.

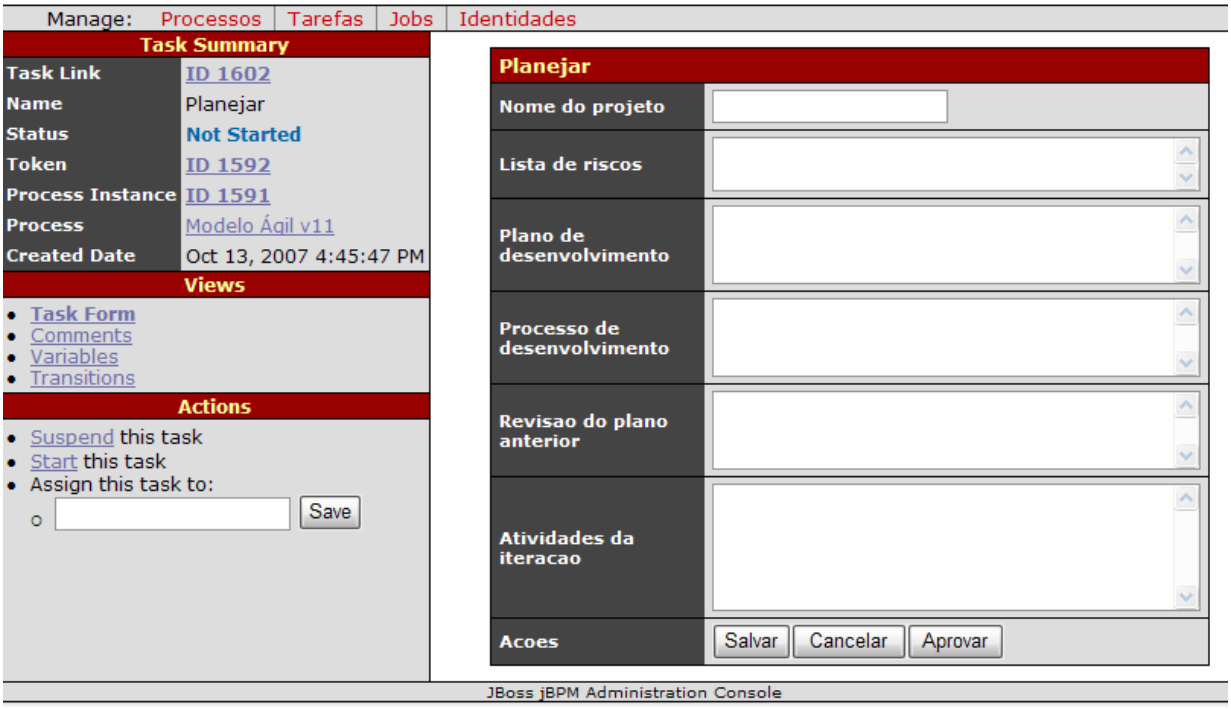


Figura 30: Exemplo de formulário (Fonte: Autor)

Com o sistema de monitoramento do jBpm, o acompanhamento dos processos foi bastante facilitado. Com uma visão gráfica do local onde se encontra o fluxo do processo, é possível definir quantos processos se encontram na fila de trabalho de cada tarefa. Com isso, medidas corretivas podem ser rapidamente realizadas para agilizar as tarefas que estão “engarrafando” o processo, aumentando sua produtividade.

Além disso, sugere-se medidas podem ser feitas para agilizar essas tarefas demoradas, como atenção redobrada para esta tarefa, a investigação do foco de improdutividade, ou até mesmo a redefinição do processo, podendo dividir a responsabilidade de uma tarefa entre os demais participantes do fluxo.

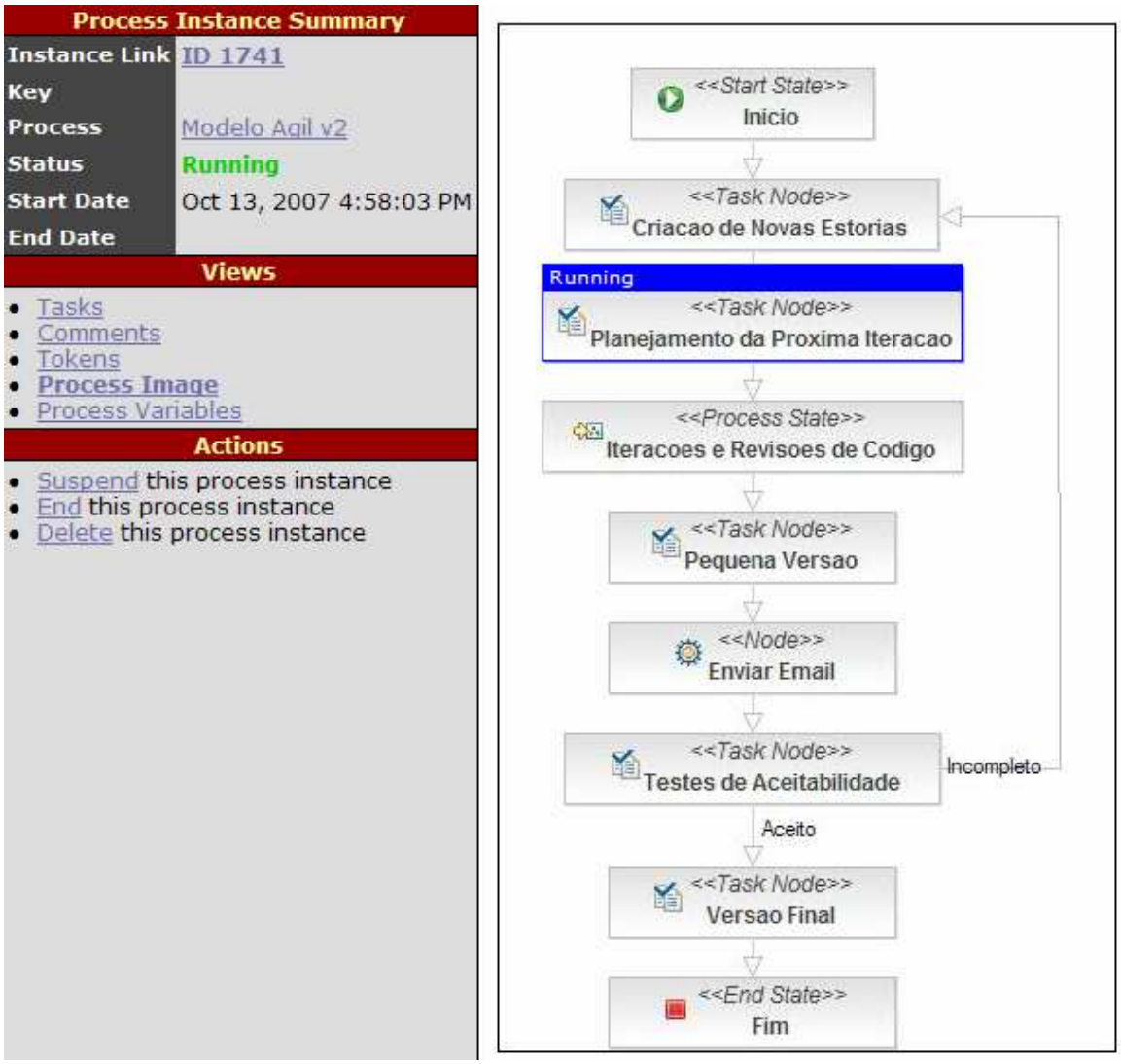


Figura 31: Monitoramento de tarefas (Fonte: Autor)

Com a elaboração da automação dos processos analisados, foi possível ter uma visão transparente da aplicabilidade dos mesmos no sistema.

Os ganhos percebidos na implementação frente aos processos modelados se resumiram à automatização dos processos, verificando sua consistência no modelo e formação de fluxos concorrentes e paralelos. Observa-se com isso, uma melhoria na visibilidade do andamento das tarefas por parte do gerente de projetos, podendo ser útil no gerenciamento da equipe e avaliar se o processo está produtivo e eficiente.

4.4. Análise e Apresentação dos resultados

Após a definição dos critérios de avaliação, modelagem dos processos e implementação, fez-se análise das diferenças das metodologias na aplicabilidade em sistemas de gestão de processos.

4.4.1. Análise Individual dos modelos

4.4.1.1. Modelo tradicional

Os modelos tradicionais de desenvolvimento se mostraram os mais fáceis de implementar devido a sua simplicidade de processo. Sendo um tipo de modelo de poucas tarefas, os modelos apresentaram poucas transições, além de terem seu formato linear e seqüencial.

Outra questão que facilitou seu desenvolvimento foi o processo ser bem definido perante aos outros modelos analisados. Devido ao grau de definição, as atividades e suas responsabilidades foram facilmente mapeadas, porém com pouco nível de detalhamento.

Como a metodologia *cleanroom* necessita ser bem auditada, o sistema ajudaria na geração de *logs* para vistoria e monitoração do processo. Além disso, poderia agregar documentos que facilitassem a burocracia de cada tarefa do processo, afim de automatizar e agilizar as formalidades do projeto.

Critérios	Impacto	Comentários
Qualidade Agregada	Alto	• Geração de *logs* e relatórios durante as transições do fluxo.
Custos	Baixo	• Diminuição dos custos de auditoria.
Entrega	Médio	• Agilização da burocracia.
Tempo vs. Esforço por Tarefa	Baixo	• As tarefas são processadas e completadas de uma vez.
Artefatos	Alto	• Vinculação de documentos formais durante o processo e as tarefas.
Atividades	Baixo	• Atividades são realizadas apenas uma vez.
Papéis	Baixo	• Pouco nível de detalhamento não houve grandes resultados.
Fluxo de trabalho	Baixo	• Baseado em documentos formais. • Seqüencial e linear. • Fluxo bem definido.

Quadro 1: Avaliação do modelo tradicional (Fonte: Autor)

4.4.1.2. Modelo Iterativo e Incremental

O modelo iterativo e incremental se adaptou bem ao estudo proposto. Por ser adaptável, o workflow trouxe grandes melhorias no processo, podendo ser customizado por diversos tipos e tamanhos de produtos e projetos de software. Além disso, o sistema propôs uma maior dinamicidade nos ciclos de desenvolvimento, aumentando o controle e ganhos entre iterações de processo.

Com o desenho gráfico do processo, o projeto iterativo e incremental aumentou sua flexibilidade, podendo ser alterado a qualquer momento pelo engenheiro de processos ou pela gerência do projeto. Isso trás uma grande adaptabilidade no mercado de software, aumentando sua competitividade. Com

os cronogramas cada vez mais apertados, a equipe pode ser remanejada de acordo com as necessidades do projeto.

Como muitas metodologias iterativas e incrementais são voltadas também para o papel, o *workflow* também poderia ajudar na automatização da burocracia nesses projetos. Uma análise superficial poderia dizer que a melhoria na burocracia seria idêntica ao modelo analisado anteriormente (modelo tradicional), diminuindo apenas a quantidade de papel utilizado. Porém, este modelo pode ser muito mais beneficiado com isso, pois muitos artefatos podem ser utilizados nas atividades do *workflow*, não se restringindo a documentos formais. Alguns artefatos podem ser listados como: código fonte, modelagens e diagramas, documentos da análise dos requisitos do cliente, instruções aos colaboradores, arquivos executáveis, etc.

Critérios	Impacto	Comentários
Qualidade Agregada	Alto	• Geração de *logs* e relatórios durante as transições do fluxo. • Maior controle sobre o processo. • Extração de medições de iteração. • Automatização de normas e controles de qualidade.
Custos	Baixo	• Diminuição de custos para mudanças de escopo ou requisitos do cliente.
Entrega	Alto	• Agilização de iterações.
Tempo vs. Esforço por Tarefa	Médio	• Monitoramento das tarefas da equipe. • Iterações altamente gerenciadas. • Iterações longas, fazendo com que o esforço aplicado nas tarefas seja maior.
Artefatos	Médio	• Código fonte, modelos, diagramas, arquivos executáveis.

Atividades	Alto	• Atividades bem definidas para cada papel. • Padronização e verificação de atividades.
Papéis	Médio	• Controle de papéis e permissões.
Fluxo de trabalho	Médio	• Fluxo cíclico, agregando informações a cada iteração. • Casos de uso continuamente implementados, integrados e testados.

Quadro 2: Avaliação do modelo iterativo e incremental (Fonte: Autor)

4.4.1.3. Modelo de desenvolvimento ágil

Para projetos cujas especificações são passíveis de alterações, como no desenvolvimento ágil, o workflow se demonstrou bem apropriado para projetos desse tipo. Isso ocorre quando projetos ágeis possuem uma diretiva bem definida para suportar mudanças no contexto do desenvolvimento com as alterações de requisitos do cliente. Já para projetos que não possuem essa definição, o *workflow* acaba sendo um empecilho para o projeto já que nunca será seguido um fluxo definido.

Como no final de cada iteração de desenvolvimento ágil é de costume disponibilizar uma versão do sistema para o cliente, o *workflow* poderia controlar e automatizar esse processo. Já que o cliente tem participação no processo de desenvolvimento do produto, poderia existir ao final de cada versão uma tarefa para que o cliente diga seus comentários e opiniões sobre o sistema. Com isso, realimenta-se a próxima iteração do processo, aumentando assim, a qualidade a cada ciclo de desenvolvimento.

Critérios	Impacto	Comentários
Qualidade Agregada	Alto	• Geração de *logs* e relatórios durante as transições do fluxo. • Extração de medições de iteração.

		• Formalização da comunicação. • Formalização de atividades.
Custos	Baixo	• Nenhum
Entrega	Médio	• Agilidade na gestão do projeto. • Rápida alteração de escopo por parte do processo.
Tempo vs. Esforço por Tarefa	Alto	• Monitoramento das tarefas da equipe. • Iterações altamente gerenciadas. • Iterações curtas, fazendo com que o esforço aplicado nas tarefas seja menor.
Artefatos	Alto	• Código fonte, modelos, diagramas, arquivos executáveis. • E-mails e estórias do cliente. • A comunicação da equipe pode ser auditada.
Atividades	Baixo	• Atividades não tão bem definidas para cada papel, podem ser repassadas para outros participantes.
Papéis	Baixo	• O cliente participa do processo.
Fluxo de trabalho	Baixo	• Fluxo cíclico, agregando informações a cada iteração. • Integração contínua de código.

Quadro 3: Avaliação do modelo de desenvolvimento ágil (Fonte: Autor)

4.4.2. Análise comparativa entre os modelos

Observou-se, através dos quadros anteriores, que todas as dimensões de critérios foram avaliadas para os três modelos de processos de software considerando seus respectivos usos no sistema.

A qualidade proporcionada pelo sistema de *workflow* pode ser resumida através de diversas melhorias adquiridas. Pode-se citar como melhoria básica a padronização do processo, refletindo em aumento qualitativo das atividades desempenhadas e podendo ser utilizada para busca de certificações de qualidade. Além disso, outras melhorias podem ser citadas para organizações ou processos mais específicos.

Para a dimensão “custo”, apesar de ter grande peso na comparação entre os modelos, este critério avaliado se mostrou pouco afetado pela utilização do *workflow*. Entretanto, isto não deve ser pré-conceituado como pouco impactado, já que as otimizações nas outras dimensões trouxeram grandes benefícios para os processos, podendo influenciar indiretamente nos custos do processo.

Outro ponto que deve ser considerado na comparação foi à diminuição do tempo de entrega do processo. Isso pode ser observado devido à automatização de muitas tarefas e controles proporcionados pelo sistema de *workflow*, agilizando a comunicação e as atividades entre os participantes dos processos. Em conseqüência, isso pode ajudar em muitos projetos de software que sofrem com atrasos de cronogramas, garantindo a entrega do sistema ao cliente em tempo acordado.

O monitoramento trouxe um aumento significativo no controle dos processos. Através disso, consegue-se a um gerenciamento melhor nas atividades desempenhadas pelos colaboradores do processo, podendo diminuir o tempo e esforço por tarefas, delegando recursos para tarefas mais críticas.

Vale salientar ainda sobre os benefícios da alteração em tempo real do fluxo de trabalho. Com um editor gráfico de *workflow*, foi possível redefinir os papéis, atividades e o fluxo das tarefas em apenas alguns instantes. Pode-se dizer que isso é um dos grandes trunfos do sistema, já que uma reengenharia de processos pode ser aplicada livremente, não necessitando ter uma equipe de TI à disposição do gestor do processo para uma eventual reimplementação do sistema de controle dos processos.

4.4.2.1. Apresentação de um quadro comparativo

Verifica-se com o quadro a seguir, a comparação entre os resultados obtidos das análises anteriores, a respeito dos impactos e benefícios adquiridos.

Critérios	Tradicional	Iterativo	Ágil
Qualidade Agregada	Alto	Alto	Alto
Custos	Baixo	Baixo	Baixo
Entrega	Médio	Alto	Médio
Tempo e Esforço por Tarefa	Baixo	Médio	Alto
Artefatos	Alto	Médio	Alto
Atividades	Baixo	Alto	Baixo
Papéis	Baixo	Médio	Baixo
Fluxo de trabalho	Baixo	Médio	Baixo

Quadro 4: Impactos da utilização de *workflow n*os modelos (Fonte: Autor)

Outro quadro comparativo pode ser visto através da comparação entre os principais benefícios impactados nos modelos de software.

Tradicional	Iterativo	Ágil
• Geração de logs para auditorias. • Documentos formais durante todo o processo.	• Aplicação de normas de qualidade. • Medições e relatórios sobre o processo. • Monitoramento das tarefas. • Iterações mais ágeis. • Papéis altamente controláveis.	• Formalização da comunicação e das atividades. • Monitoramento das tarefas. • Diversos artefatos durante o processo. • Gestão de projetos ágeis.

Quadro 5: Principais benefícios dos modelos (Fonte: Autor)

5. Considerações Finais

5.1. Objetivos do trabalho

O objetivo deste trabalho foi realizar um estudo comparativo da utilização de sistemas de gestão de processos de negócios, bem como a aplicação de um *workflow*, com alguns modelos de desenvolvimento de software além de mostrar as barreiras e facilitadores para a conclusão da aplicação.

A pesquisa constituiu-se de uma revisão da literatura que abordou os conceitos necessários para o entendimento da comparação, abrangendo elementos conceituais de processos de negócios, reengenharia e gestão de processos, ciclo de desenvolvimento de software e modelos de processos para sua construção.

Posteriormente, procurou-se avaliar a modelagem de processos de software como se fossem processos de negócio além da construção de uma aplicação que atendesse estes modelos. Com isso, procurou-se avaliar o impacto de sua utilização através deste estudo, analisando e envolvendo os três modelos de processos de software mais bem consagrados pelo mercado de sistemas de informação.

Os resultados obtidos pelo estudo conceitual dos processos permitiram, dentro das limitações do método utilizado, estabelecer respostas em caráter comparativo e aos objetivos da pesquisa, agregando contribuições para trabalhos futuros nesta área de atuação.

5.2. Contribuições

Quanto ao objetivo principal de analisar e avaliar o impacto da automação de processos de software utilizando tecnologias de BPM e *Workflow*, verifica-se que os resultados obtidos pela comparação em relação aos critérios avaliados apontam para grandes benefícios no aumento da eficiência e eficácia desses processos.

Verificou-se também que o redesenho de alguns modelos e processos de software podem ser facilmente melhorados e alterados com o sistema de

Workflow. De acordo com a demanda dos clientes, a empresa fornecedora de sistemas pode redefinir seus processos, afim de atender prontamente ao mercado.

De forma geral, o estudo trouxe algumas conclusões como:

- Redução de tempo para diversas atividades de desenvolvimento de software, reduzindo os riscos do cronograma do projeto e, conseqüentemente, a entrega do sistema para o cliente. Para que isso aconteça, podem-se eliminar algumas atividades desnecessárias, aumentar atividades em paralelo e monitorar as tarefas dos funcionários através de um sistema de *workflow*.
- Padronização e aumento de qualidade no processo de software tendo como resultado a diminuição de erros no sistema fornecido pela fábrica de software. Além disso, a padronização das atividades pode auxiliar na implantação de uma certificação de qualidade e retenção de conhecimento dentro da empresa.
- Controle sobre as atividades dos funcionários já que se pode monitorar de forma constante o processo. Através disso, um sistema de gestão de processo pode auxiliar na gestão de projetos realizando cobranças automáticas de tarefas aos colaboradores.

Finalmente, identificou-se através dos estudos da pesquisa a aplicação e avaliação dos conceitos descritos na literatura, relacionando à implementação de um sistema de informações voltado para a gestão de processos de negócio e a sua influência nos processos de desenvolvimento de software.

5.3. Limitações

Este trabalho possui natureza conceitual dos assuntos estudados, por isso possui diversas limitações.

Primeiramente, refere-se à revisão bibliográfica sobre o assunto já que nada foi encontrado sobre o estudo deste trabalho. Com isso, o resultado do trabalho foi uma unificação de literaturas sobre processos de software e processos de negócios.

Devido à falta de literatura sobre o assunto, os critérios de avaliação foram analisados a partir de casos que não seja especificamente a do tema deste trabalho, fazendo com que avaliação possa estar faltando critérios não analisados e que seja de importante interesse.

Outra limitação encontrada foi à impossibilidade de aplicar a implementação do trabalho em uma equipe de desenvolvimento de software real, fazendo com que o trabalho esteja limitado a conceitos acadêmicos. Para realizar o trabalho, o desenvolvimento do sistema foi implementado e executado pela mesma pessoa, no caso o autor, limitando drasticamente a influência interpessoal no trabalho.

5.4. Trabalhos Futuros

A seguir, alguns estudos futuros que poderão ser desenvolvidos com base neste trabalho:

- Utilização e replicação dos modelos apresentados ou outros modelos, afim de reavaliar o impacto do uso de sistemas de gerenciamento de processos e workflow no processo de software.
- Utilização da implementação dos casos apresentados em uma equipe de desenvolvimento de sistemas real, estudando fatores como a adequação ao sistema, a resistência a mudanças e a delegação de tarefas para as pessoas, considerando o processo de software constituído não somente por recursos de forma geral, mas por uma visão interpessoal e social da equipe.
- Estudo específico de uma implementação que atendesse a maioria dos casos analisados, podendo ser útil em diversas empresas de sistemas, sob a ótica da engenharia de software e a gestão de processos.
- Avaliação e análise da aplicabilidade da Gestão de Processos de Negócios para uma ou mais metodologias de gerenciamento de projetos, afim de abranger e contemplar a forma como deve ser construído um sistema de informação.

6. Referências Bibliográficas

AALST, W.M.P.; HOFSTEDE, A.H.M.; WESKE, M. Business Process Management: A Survey, Springer Verlag, 2003.

ABRAHAMSSON, P.; SALO, O.; RONKAINEN, J.; WARSTA J. Agile SoftwareDevelopment Methods: Review and Analysis, Espoo 2002, VTT Publications 478, 2002.

ABREU, Pedro Felipe; ABREU, Aline F. Sistemas de Informação Gerenciais: Uma abordagem orientada à gestão empresarial, UFSC, 2003.

BAEYENS, Tom. The State Of Workflow, 2004. Disponível em: <http://www.jboss.com/products/jbpm/stateofworkflow>

BEZERRA, Eduardo, Desenvolvimento incremental e iterativo, 2007. Disponível em: <http://www.mundooo.com.br/php/mooartigos.php>

BORTOLINI, Rafael. Padronizando Processos: BPMN, BPML, XPDL e BPEL, 2006. Disponível em: <http://www.cryo.com.br>

CAMPOS, Vicente F. Gerenciamento da Rotina. Rio de Janeiro: Qualitymark, 1997.

DAVENPORT, Thomas, Process Innovation: Reengineering work through information technology, Harvard Business School Press, Boston, 1993.

GARLAN, D.; SHAW, M. An Introduction to Software Architecture CMU/SEI, 1994.

DIETZEN, S. Standards for service-oriented architecture, Weblogic Pro. 2004. Disponível em: <http://dev2dev.bea.com/pub/a/2004/05/soa_dietzen.html>

ERIKSSON, H. E.; PENKER, M. UML Toolkit. New York : John Wiley, 1998.

EVANS, Eric. Domain-Driven Design: Tackling Complexity in the Heart of Software Addison-Wesley, 2003, 576 p.

FRUSCIONE, James. Workflow automatizado. São Paulo, Cenaden, 1996.

GEORGAKOPOULOS, D.; HORNICK, M. F.; SHETH, A. P. An overview of workflow management: From process modeling to workflow automation infrastructure. Distributed and Parallel Databases, 1995. Disponível em: <http://citeseer.ist.psu.edu/georgakopoulos95overview.html>

GURLEY, Willian. Pay attention to BPM, 2003. Disponível em: <http://www.news.com/2010-1071-994310.html>

HAMMER, Michael; CHAMPY, James. Reengineering the Corporation: A Manifesto for Business Revolution, Harper Business, 1993.

HAVEY, M. Essential Business Process Modeling. O'Reilly, 2005.

HOLLINGSWORTH, D. The workflow reference model: 10 years on, 2004. Disponível em: <http://www.wfmc.org/standards/docs/tc003v11.pdf>

HURST, Jim. Applications and Systems Development Security, 2007. Disponível em: <http://www.giac.org/resources/whitepaper/application>

HURST, Jim. Comparing Software Development Life Cycles, 2007. Disponível em: <http://www.giac.org/resources/whitepaper/application>

IEEE 1074 IEEE Standard for Developing Software Life Cycle Process 1995.

IEEE Recommended Practice for Architectural Description, Esboço 3.0 de IEEE P1471, 1998. Disponível em: <http://www.pithecanthropus.com/~awg>

JACOBSON, Ivar; ERISSSON, MARIA & JACOBSON, Agneta. The Object Advantage: Business Process Reengineering with Object Technology, Addison Wesley, 1994.

JACOBSON, I. et al. The unified software development process. Reading : Addison-Wesley, 1999.

JOHANSSON, Henry J. et.al. Business Process Reengineering: BreakPoint Strategies for Market Dominance, John Wiley & Sons, 1993.

KAPPEL, G., RAUSCH-SCHOTT, S., and RETSCHITZEGGER, W. A framework for workflow management systems based on objects, rules and roles. ACM Computing Surveys, 2000. Disponível em: <ftp://ftp.ifs.uni-linz.ac.at/pub/publications/1998/1698.pdf>

LAUDON, K.C.; LAUDON, J.P. Management Information Systems: new approaches to organization & technology. New Jersey, USA: Prentice Hall, 1998.

MCLEOD, Graham, Advanced Extending UML for Enterprise and Business Process Modeling, Inspired, South Africa, 1998.

MITCHELL H. Levine. Analyzing the Deliverables Produced in the Software Development Life Cycle, 2000. Disponível em: <http://www.auditserve.com/articles/art_4.htm>

MORRIS, Daniel; BRANDON, Joel. Reengenharia de Processos: Como Inovar nas Empresa Através da Tecnologia da Informação, Rio de Janeiro, 1994.

PRESSMAN, Roger S. Engenharia de Software, 5. ed. Rio de Janeiro, MacGraw Hill, 2004.

REIS, Glauco. O que é uma Solução BPMS, 2006. Diponível em: <http://www.portalbpm.com.br>

ROYCE, W. W. Managing the development of large software systems. Proceedings of IEEE WESCON, 1970.

RUMMLER; BRACHE. Improving Performance: How to manage the white space on the organizational chart, Jossey-Bass, San Francisco, 1995.

SMITH, John. A Comparison of the IBM Rational Unified Process and eXtreme Programming. IBM, 2001.

SOMMERVILLE, I. Software engineering. 5th. ed. Addison-Wesley, 1995.

SOUZA, B. N. Integração de dispositivos móveis às ferramentas de workflow. Brasília, UnB, 2006.

TELES, Vinícius Manhães. Um Estudo de Caso da Adoção das Práticas e Valores do Extreme Programming. Rio de Janeiro, 2005.

USIRONO, C. H. Tecnologia Workflow: O impacto de sua utilização nos processos de negócio. São Paulo, USP, 2003.

VASCO, C.G.; Vithoft, M.H.; Estante, P.R. Comparação entre Metodologias RUP e XP. Curitiba, PUCPR, 2006.

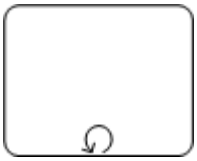
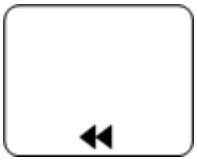
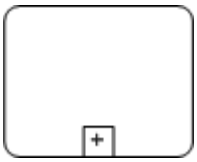
WFMC. Workflow management coalition XML process definition language, 2005. Disponível em: <http://www.wfmc.org>

WHITE, S. A. Introduction to BPMN, 2004. Disponível em: <http://www.bpmn.org>

APÊNDICE A

Notação BPMN: Elementos Básicos

Atividades

Elemento	Nome	Descrição
	Tarefa	Representa um trabalho ou atividade que um participante atua.
	Loop	Execução da mesma tarefa diversas vezes.
	Compensação	Logo após a finalização desta tarefa, o fluxo é desfeito e/ou retorna para o elemento que o solicitou.
	Sub-processo	Representação de um sub-processo no fluxo. Detalhes desse processo não estão visíveis no diagrama.

Eventos

Elemento	Nome	Descrição
	Simples	Indica o início, o meio e o fim de um processo (respectivamente).
	Mensagem	Uma mensagem chega para o participante e aciona/interrompe o fluxo.
	Temporal	Um temporizador para iniciar um modo de espera no processo.
	Exceção/Erro	Tratamento de uma exceção/erro específico.
	Múltiplo	Significa que múltiplos eventos podem ocorrer para acionar/interrromper o fluxo, bastando apenas um dos eventos especificados para que seja aplicado o fluxo.
	Compensação	Identifica que um segmento do fluxo deve ser desfeito ou retornado para sua origem.
	Regra de Negócio	Condição ou regra para que o fluxo seja seguido.

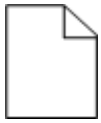
Gateways

Elemento	Nome	Descrição
	Decisão Exclusiva	Significa que o fluxo deve seguir apenas um caminho a partir de uma condição.
	Bifurcação/Junção Paralela	Significa que o fluxo possui paralelismo e é subdividido ou juntado em outros segmentos.

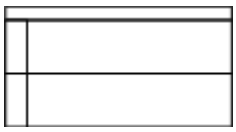
Conexões

Elemento	Nome	Descrição
	Transição sequencial	Mostra a ordem que as atividades são executadas.
	Transição de associação	Associa informações com objetos de fluxo, textos ou elementos gráficos.
	Transição de mensagem	Mostra o fluxo de mensagens entre dois participantes de processos diferentes.

Artefatos

Elemento	Nome	Descrição
	Objeto de Dados	Representa os dados que são requisitados ou produzidos pela atividade.
	Anotações	Fornece informações adicionais para o leitor do diagrama.
	Grupo	Pode ser utilizado para documentar ou analisar tendências, não afetando o fluxo.

Raias

Elemento	Nome	Descrição
	Raia	Representa um participante no processo.
	Piscina	Representa um conjunto de participantes no processo.