UNIVERSIDADE DO SUL DE SANTA CATARINA

THIAGO THIESEN DE SOUZA

GESTÃO DE VERSÃO DE SOFTWARE:
UM ESTUDO DE CASO

FLORIANÓPOLIS

2015

THIAGO THIESEN DE SOUZA

GESTÃO DE VERSÃO DE SOFTWARE:
UM ESTUDO DE CASO

Trabalho de Conclusão de Curso de Especialização apresentado ao Curso de Especialização em Engenharia de Projetos de Software, da Universidade do Sul de Santa Catarina, como requisito parcial para obtenção do título de Pós-graduado.

Orientadora**:** Professora Maria Inés Castiñeira, Dra

FLORIANÓPOLIS

2015

THIAGO THIESEN DE SOUZA

GESTÃO DE VERSÃO DE SOFTWARE:
UM ESTUDO DE CASO

Trabalho de Conclusão de Curso de Especialização apresentado ao Curso de Especialização em Engenharia de Projetos de Software, da Universidade do Sul de Santa Catarina, como requisito parcial para obtenção do título de Pós-graduado.

Florianópolis, 11 de janeiro de 2015.

_____

Profa. e orientadora Maria Inés Castiñeira, Dra.

Universidade do Sul de Santa Catarina

_____

Profa. Vera R. N. Schuhmacher, Dra.

Coordenadora do Curso de Especialização em Engenharia de Projetos de Software

Universidade do Sul de Santa Catarina

Dedico este trabalho a minha mãe que sempre me ajudou, incentivou e apoiou em minhas escolhas.

À professora e orientadora Maria Inés Castiñera por auxiliar na orientação neste trabalho.

AGRADECIMENTOS

Agradeço à toda minha família por acreditar em meu potencial e pelo auxílio nos momentos difíceis.

A todos os professores que dividiram seu conhecimento conosco durante o curso, agradecendo especialmente a professora Maria Inés Castiñera, que foi uma pessoa de grande importância para a realização deste trabalho.

À namorada pela paciência nos momentos de ausência para o desenvolvimento deste trabalho.

“A ciência nunca resolve um problema sem criar pelo menos outros dez.” (George Bernard Shaw).

RESUMO

Este trabalho de conclusão de curso abordou a gestão de versão de *software*. Inicialmente foi realizado um levantamento bibliográfico sobre o processo de software, sendo descritos as quatro etapas fundamentais deste processo, a especificação, o desenvolvimento e a validação e evolução do *software*. Adicionalmente foi pesquisado sobre o gerenciamento e configuração de *software*, sendo apresentados os itens indispensáveis na realização do desenvolvimento e manutenção da vida útil do *software*. Posteriormente é exposto o funcionamento do controle de versão de arquivos, bem como os seus benefícios junto ao processo de desenvolvimento de *software*. Com base na teoria pesquisada, foi realizado um estudo de caso, na empresa de atuação profissional do autor deste trabalho. Neste estudo, foram apresentados os processos de desenvolvimento de *software* antes e depois da aplicação de práticas de controle de versão e configuração de *software*, realizadas com o apoio de uma ferramenta CASE. Os dois processos apresentados descrevem dois períodos de tempo, ou fases, no estágio de desenvolvimento da organização, e foram descritos com apoio de pessoas que os vivenciaram. A partir do estudo de caso é possível a identificação da utilidade de uma ferramenta CASE no apoio ao processo de versionamento de código.

Palavras-chave: Versionamento de Código Fonte. Gestão de Configuração de *Software*. Estudo de Caso. Processo de *Software*.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

CASE – *Computer Aided Software Engeneering*

SCM – *Software Configuration Management*

TFS – *Team Foundation Server*

UNISUL – Universidade do Sul de Santa Catarina

US – User Storie

SUMÁRIO

1 INTRODUÇÃO

O desenvolvimento com qualidade continua sendo muito importante para organizações desenvolvedoras de *software* e seus clientes. Para atingir essa qualidade a Engenharia de *Software* propõe a utilização de diversas práticas, tecnologias e ferramentas. Muitas dessas boas práticas estão incorporadas nos modelos de desenvolvimento de *software* e nos modelos de qualidade de *software* (PRESSMAN, 2006; SOMMERVILLE, 2003).

Além de modelos de qualidade, as organizações utilizam ferramentas e metodologias para dar suporte ao desenvolvimento de *software*. As ferramentas CASE (sigla do inglês para *Computer Aided Software Engeneering*), proporcionam a automação de atividades de processo e o fornecimento de informações sobre o *software* que está sendo desenvolvido (SOMMERVILLE, 2003).

Como complemento da qualidade na produtividade de *software*, as organizações adotam o gerenciamento de configuração de *sofware*, este é um conjunto de atividades de controle e rastreamento que começa quando um projeto de desenvolvimento inicia e só é finalizado quando o *software* é tirado de operação. A principal responsabilidade deste processo é o controle de mudanças, esse controle permite a melhora na qualidade do processo de desenvolvimento de *software*. (PRESSMAN, 2006). Uma das etapas desse processo é o controle de versão de *software*.

1.1 PROBLEMÁTICA

O desenvolvimento de *software* geralmente abrange diversas pessoas, que no decorrer do projeto podem querer utilizar e realizar alterações no mesmo arquivo simultaneamente, como exemplo, arquivos de código fonte do *software*. Com esta concorrência, surgiu a necessidade de as empresas organizarem os diversos arquivos de projeto.

Conforme Dias (2011), muitos problemas ne desenvolvimento de *software* são causados pela falta de controle de edição dos arquivos. No processo de software, muitas vezes, os arquivos compartilhados vão sendo editados e salvos, sempre sobrepondo a

informação contida no arquivo anterior, não havendo um versionamento destes. Caso isso acontecer e em algum momento seja necessário resgatar uma versão específica do arquivo, não existe como.

O desenvolvimento de *software* é um trabalho delicado, qualquer alteração feita de forma erronea pode levar ao não funcionamento de algo, este problema pode se agravar quando o mesmo código fonte é utilizado por mais de uma pessoa. Um simples retorno de versão poderia resolver o problema, porém quando não existe gestão em cima dos arquivos, este procedimento não é possível de ser realizado (DEVMEDIA, 2014).

Diante deste desafio, o presente estudo tem como assunto o processo de desenvolvimento de *software*, com foco no versionamento de arquivos.

1.2 OBJETIVOS

A seguir, serão apresentados os objetivos do estudo que estão subdivididos em geral e específicos.

1.2.1 Objetivo Geral

Descrever e analisar o processo de desenvolvimento de *software* de uma empresa estudo de caso com ênfase na gestão de versão do *software*, realizada com o apoio de uma ferramenta CASE.

1.2.2 Objetivos Específicos

Os objetivos específicos deste trabalho são:

- pesquisar na literatura sobre as boas práticas de desenvolvimento de *software*, com ênfase para a gestão de configuração e versão;
- descrever o processo de desenvolvimento de *software* da empresa estudo de caso antes de uma ferramenta CASE;
- modelar e documentar o processo de desenvolvimento de *software* com o uso da ferramenta CASE;
- analisar cada etapa do desenvolvimento de *software* com base no novo processo;
- descrever os pontos em que a ferramenta CASE está apoiando no desenvolvimento de *software* no processo atual e pontos de melhorias.

1.3 JUSTIFICATIVA

O versionamento de arquivos foi escolhido visando gerar conhecimento neste processo que vem sendo utilizado em muitas das empresas de desenvolvimento de *software*. O embasamento de estudo se deu através de diversos artigos estudados a fim de unir informações consistentes sobre o assunto.

Segundo Alexandre (2014), o versionamento de arquivos possibilita manter o histórico de evolução de um arquivo. Existem ferramentas no mercado que a partir do versionamento permitem detectar e mesclar alterações realizadas em um mesmo arquivo, bem como garantir a edição de um arquivo para uma única pessoa.

Para Alexandre (2014), com o uso de uma ferramenta de versionamento de arquivos, o desenvolvimento de um *software* em equipe também terá ganhos, o controle de versão garante que toda liberação de código fonte seja comentada, amenizando erros e esquecimentos.

O uso de ferramentas de versionamento está cada vez mais comum no mercado de tecnologia da informação, com isto, este trabalho aprofunda e reune informações sobre o assunto, possibilitando apoio aos profissionais da área que buscam informações afim de utilizar esse tipo de ferramenta.

1.4 MÉTODO

Este tópico apresenta aspectos acerca da metodologia utilizada no projeto.

1.4.1 Caracterização da pesquisa

Conforme Demo (1996), pesquisa é, de forma resumida, procurar respostas para indagações propostas. Ele insere a pesquisa como atividade cotidiana considerando-a como uma atitude, um "questionamento sistemático crítico e criativo, mais a intervenção competente na realidade, ou o diálogo crítico permanente com a realidade em sentido teórico e prático".

Para Silva e Menezes (2005), pesquisa é um conjunto de ações, propostas para encontrar a solução para um problema, que tem por base procedimentos racionais e sistemáticos. A pesquisa é realizada quando se tem um problema e não se tem informações para solucioná-lo.

1.4.2 Classificação

A pesquisa pode ser dividida, segundo a classificação apresentada a seguir:

Conforme a sua natureza: Pode ser básica ou aplicada. A básica envolve interesses universais, sem aplicação prática prevista. A aplicada envolve interesses locais, dirigida a problemas específicos. (SILVA e MENEZES, 2005).

Esta pesquisa é classificada como **aplicada**, visto que está voltada ao estudo de ferramentas para controle de versões que auxiliem no desenvolvimento de *software*.

Conforme a sua forma de abordagem, a classificação pode ser quantitativa ou qualitativa. A classificação quantitativa considera tudo o que pode ser mensurado em números. Envolve recursos estatísticos (percentagem, média, moda, mediana etc.). A

classificação qualitativa considera que há uma relação entre o mundo real e o sujeito, que não pode ser traduzida em números, é descritiva. (SILVA e MENEZES, 2005).

Esta pesquisa está classificada como **qualitativa**, visto que o versionamento de código visa a eliminar problemas no processo de desenvolvimento, fazendo com que seja possível o mapeamento do histórico das informações, gerando com isso, a fonte de informação para resolver possíveis problemas que necessitam ser investigados.

Conforme seus objetivos, para Gil (2002), as pesquisas podem ser classificadas em três grandes grupos:

a) pesquisa exploratória: visa a proporcionar maior familiaridade com o problema. Assume, em geral, as formas de pesquisas bibliográficas e estudos de casos;
b) pesquisa descritiva: assume, em geral, forma de levantamento. Utiliza técnicas de coleta de dados, questionário e observação sistemática;
c) pesquisa explicativa: visa a identificar os fatores que determinam ou contribuem para a ocorrência dos fenômenos. Aprofunda o conhecimento da realidade, pois explica a razão, o "porquê" das coisas.

Esta pesquisa é classificada como pesquisa **exploratória**, pois faz referência a artigos, a estudos e a livros referentes ao assunto abordado, sendo que também será realizado um estudo de caso sobre o assunto.

1.4.3 Planejamento

Para Silva e Menezes (2005), o planejamento de uma pesquisa se dá em três etapas:

a) fase decisória: referente à escolha do tema, à definição e à delimitação do problema de pesquisa;
b) fase construtiva: referente à construção de um plano de pesquisa e à execução da pesquisa propriamente dita;
c) fase redacional: referente à analise de dados e a informações obtidas na fase construtiva. É a organização das idéias visando à elaboração de um relatório final.

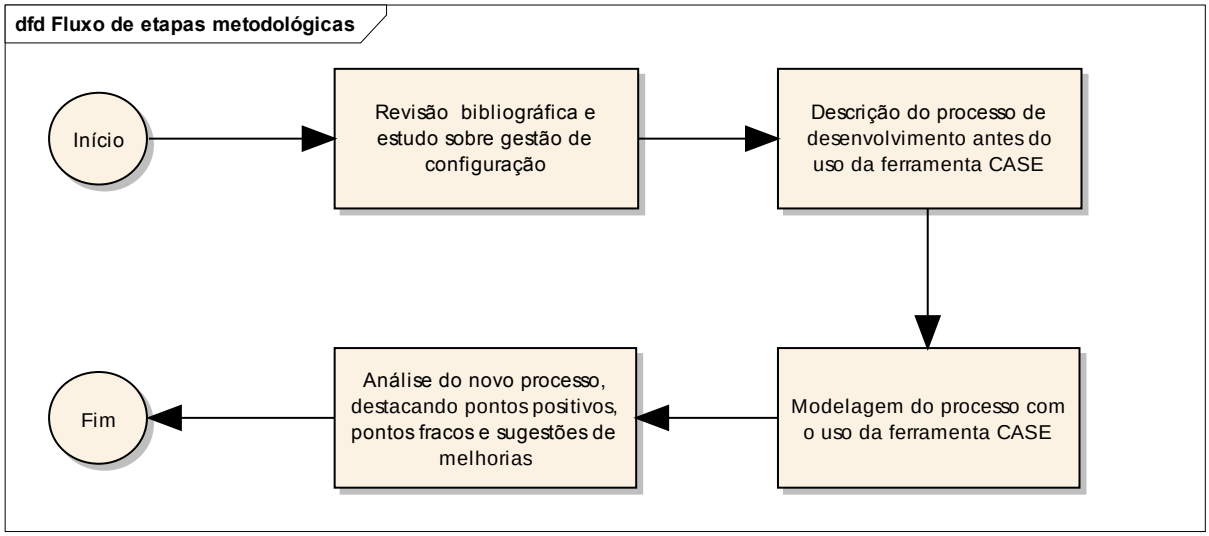
1.4.4 Etapas metodológicas

As etapas deste trabalho são as seguintes:

a) revisão bibliográfica com o estudo da teoria sobre gestão de configuração e sistemas de versionamento de arquivos;
b) descrição do processo de desenvolvimento antes do uso da ferramenta CASE;
c) modelagem do processo com o uso da ferramenta CASE;
d) análise desse processo destacando pontos positivos, pontos fracos e sugestões de melhorias.

Estas etapas são apresentadas na forma de um fluxograma na figura a seguir:

Figura 1 – Fluxograma com as etapas metodológicas



Fonte: Autor, 2015.

1.4.5 Proposta

Este trabalho apresenta a proposta de analisar o processo de desenvolvimento de *software* de uma empresa estudo de caso, com foco na gestão de configuração e versionamento de arquivos.

1.4.6 Delimitações

O produto deste trabalho tem a finalidade de descrever sobre o versionamento de arquivos, voltado ao versionamento de código fonte, oferecendo informações a pessoas que buscam conhecimento nesta área.

Como delimitações deste trabalho, podem ser destacadas as seguintes:

- será descrita apenas a ferramentas de versionamento de arquivos *Team Foundation Server* (TFS), seleção realizada pela empresa estudo de caso;
- o foco deste trabalho não é o desenvolvimento e a criação de novas tecnologias;
- o foco desta análise é no processo de desenvolvimento de *software*, a etapa da análise não será abordada.

1.5 ESTRUTURA DA MONOGRAFIA

Este trabalho está organizado como descrito a seguir.

Inicialmente, foi descrito o tema, objetivo geral e objetivos específicos seguidos da justificativa e do método, apresentando a classificação do tipo de pesquisa realizada, as atividades que compõem a proposta da solução e as delimitações do projeto.

Nos capítulos 2, 3 e 4, é apresentada a revisão da literatura, abordando a conceituação dos assuntos relacionados ao domínio do problema, baseados em uma revisão

bibliográfica sobre o que é e como funciona o controle de versão, bem como a sua utilização junto ao desenvolvimento de *software*.

No capítulo 5, é apresentado o desenvolvimento da proposta. Primeiramente é apresentada a empresa estudo de caso para então ser descrito o antigo e o novo processo de desenvolvimento de *software*, voltado ao versionamento de arquivos, onde é destacada a utilização de uma ferramenta CASE que apóia o processo.

Para finalizar, o capítulo 6 descreve as conclusões e futuros desdobramentos desta monografia.

2 SOFTWARE E O SEU PROCESSO

Com base em autores de referência na área, neste capítulo é apresentado o *software* e o seu processo, juntamente com ferramentas CASE.

2.1 SOFTWARE

Pressman (2006) diz que *software* pode ser descrito como um conjunto de instruções que quando executadas produzem a função e o desempenho desejado, compostas por estruturas de dados que possibilitam que os programas manipulem adequadamente a informação, de acordo com os documentos que descrevem a operação e o uso dos programas.

De acordo com Sommerville (2003), o termo *software* pode ser tratado como a junção do programa com toda a documentação associada e os dados de configuração necessários para que seja operado normalmente.

Conforme Pressman (2006) o *software* é um elemento de sistema lógico desenvolvido por pessoas, ele não tem desgaste e sim deteriorização, podendo ser aplicado a qualquer situação em que um conjunto previamente especificado e definido.

2.2 PROCESSO DE SOFTWARE

O processo de *software* é um conjunto de atividades e resultados associados que resultam em um produto de *software*. Os diversos processos de *software* existem, muitos são baseados em modelos, porém todos possuem quatro etapas fundamentais e comuns a todos os processos de *software*, são elas: especificação, desenvolvimento, validação e evolução de *software* (SOMMERVILE, 2003).

De acordo com Sommervile (2003), as etapas do processo de *software* podem variar de acordo com cada organização, pois podem utilizar diferentes processos para

produção do mesmo produto. Pressman (2006) diz que os projetos reais raramente seguem o fluxo sequencial que os modelos propõem.

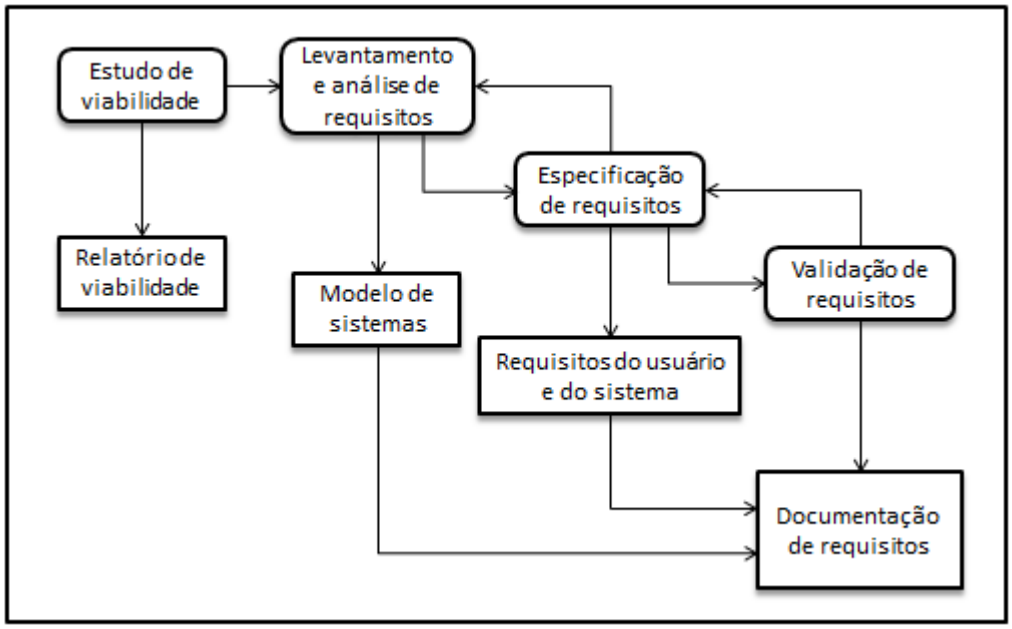
A seguir serão descritos os quatro processos sugeridos por Sommerville (2003).

2.2.1 Especificação de software

Conforme Sommerville (2003), nesta etapa são estabelecidas as funções requeridas pelo sistema e as restrições sobre a operação e o desenvolvimento do mesmo. Uma falha nesta etapa fará com que problemas apareçam nas próximas etapas do projeto.

Frequentemente chamado de engenharia de requisitos, neste processo é realizada a especificação do sistema, resultando na documentação dos requisitos, conforme demonstrado na imagem abaixo (SOMMERVILLE, 2003). Pressman (2006) diz que após documentados, os requisitos devem ser validados com o cliente.

Figura 2 – O processo de engenharia de requisitos



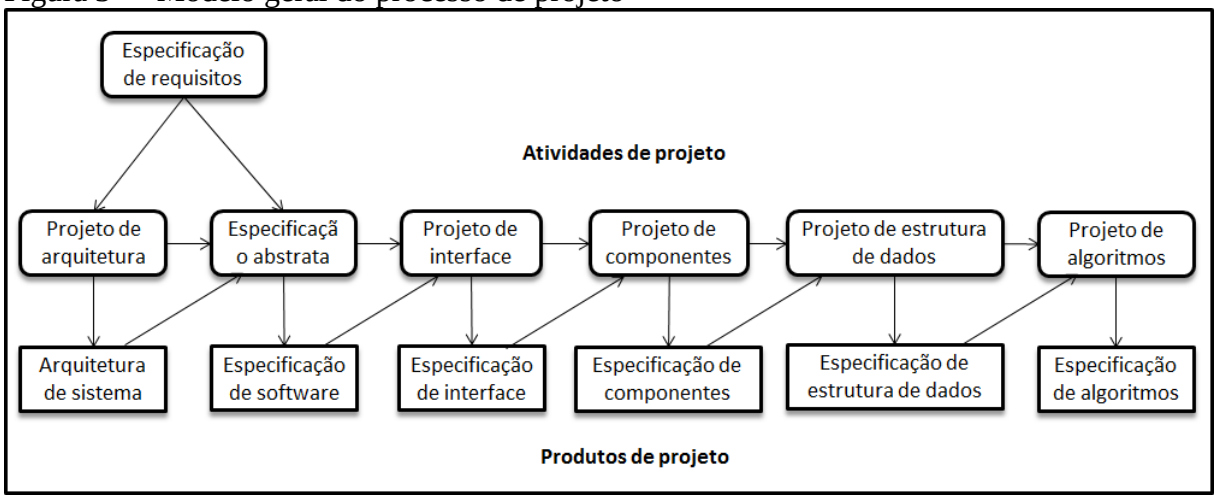
Fonte: adaptada de SOMMERVILLE, 2003.

2.2.2 Projeto e implementação de software

É nesta etapa que ocorre o processo de conversão da especificação de sistema em um sistema executável (SOMMERVILLE, 2003). Pressman (2006) afirma que o processo de feitoria do projeto traduz as exigências numa representação do *software*.

Conforme Sommerville (2003), o projeto é desenvolvido de forma iterativa, podendo envolver o desenvolvimento de vários modelos do sistema. Na figura a seguir é demonstrado um modelo desse processo, onde é sugerido o sequenciamento de forma intercalada dos estágios. Em todo o processo deve haver *feedback* de um estágio para outro.

Figura 3 – Modelo geral do processo de projeto



Fonte: adaptada de SOMMERVILLE, 2003.

Sommerville (2003) diz que esse é um modelo onde os processos práticos reais podem ser adaptados de diferentes maneiras, pois o modelo sugerido é genérico do processo de projeto.

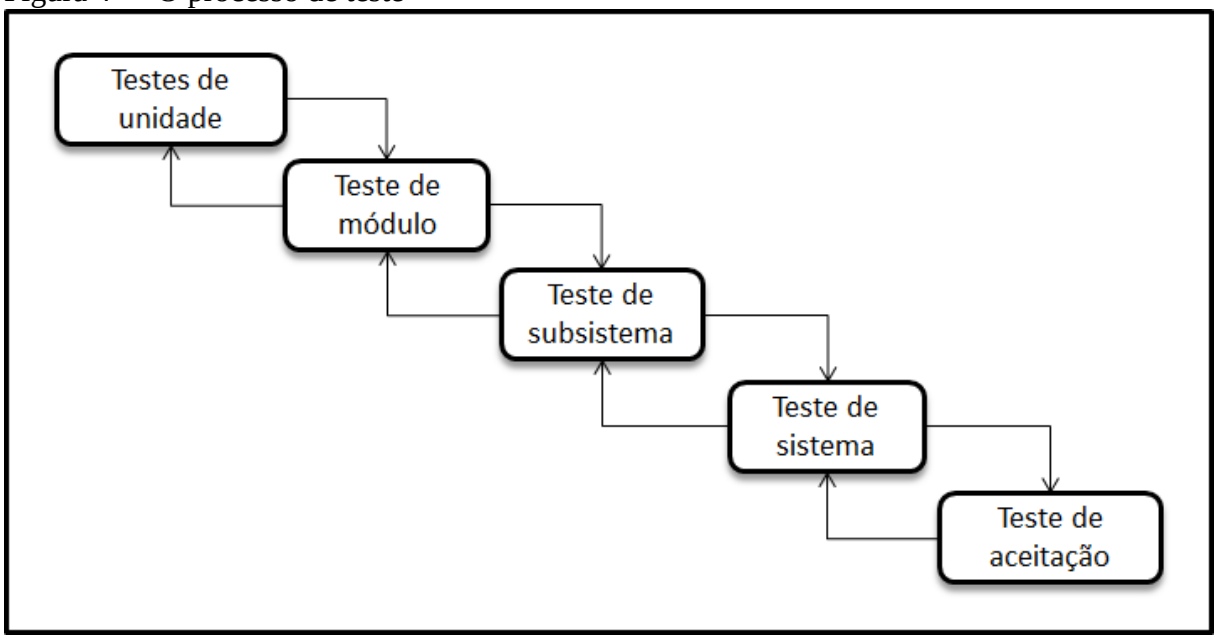
O projeto começa a ser modificado a medida que o sistema é implementado. Quando o estágio de implementação se completa, o projeto normalmente se torna tão modificado que o documento original do projeto é uma descrição incoerente e incompleta do sistema, isso acontece pois o controle formal das modificações, quando existe, é realizado de forma escassa (SOMMERVILLE, 2003).

2.2.3 Validação de software

Com base nas especificações de sistema, nesta etapa é realizada a verificação e a validação do *software*, a fim de garantir que o sistema está de acordo com suas especificações (SOMMERVILLE, 2003). Para Pressman (2006), a validação do *software* refere-se a garantia de que o *software* desenvolvido é rastreável às exigências do cliente.

Sommerville (2003) diz que o processo de teste deve ser realizado de forma incremental, em conjunto com a implementação do sistema. Conforme a imagem a seguir, esse processo é composto por cinco estágios, sendo que a medida que os defeitos são descobertos, o programa precisa ser reparado, podendo exigir a repetição de outros estágios no respectivo processo.

Figura 4 – O processo de teste



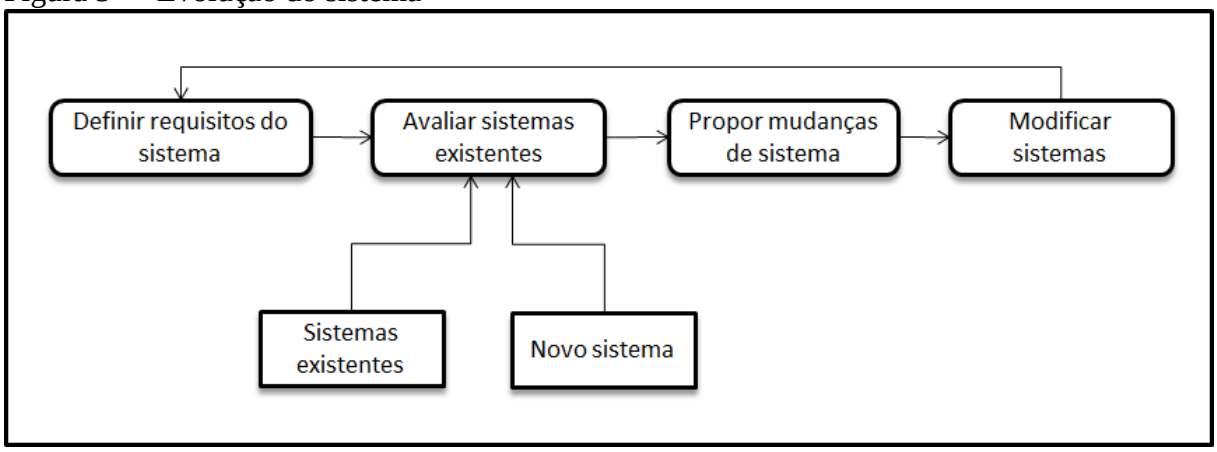
Fonte: adaptada de SOMMERVILLE, 2003.

Conforme Pressman (2003), nesta etapa do processo são realizados testes nos aspectos lógicos internos dos *softwares*, garantindo que todas as regras tenham sido testadas, e deve garantir também que os aspectos funcionais externos estejm produzindo resultados reais com os resultados exigidos.

2.2.4 Evolução de software

Sommerville (2003) diz que o *software* deve evoluir para atender às necessidades mutáveis do cliente. Este processo pode ser muito oneroso, porém as mudanças podem ser realizadas a qualquer momento durante a vida do sistema. A figura a seguir demonstra como é o processo de evolução do sistema.

Figura 5 – Evolução do sistema



Fonte: adaptada de SOMMERVILLE, 2003.

Pressman (2006) afirma que essas mudanças ocorrerão porque erros foram encontrados, para adaptações no *software* ou porque o cliente exige acrécimos funcionais ou de desempenho. Para ele, as alterações realizadas no *software* reaplicam todas as etapas precedentes do ciclo de vida a um programa existente.

2.3 FERRAMENTAS DE APOIO

Para Sommerville (2003), processos de *software* são processos complexos que dificilmente são automatizados e que dependem da ação humana. Este fato, contribuiu para que surgissem ferramentas de engenharia de *software* com auxílio de computador, comumente chamadas de ferramentas CASE (*Computer-Aided Software Engineering*),

As ferramentas CASE proporcionam a automação de algumas atividades de processo e o fornecimento de informações sobre o *software* que está sendo desenvolvido, gerando melhorias na qualidade e na produtividade de *software* (SOMMERVILLE, 2003).

As diversas ferramentas de apoio ao desenvolvimento de *software* são das mais diversas e compreendem desde um editor de texto para redigir as atas das reuniões, planilhas de cálculo para fazer orçamentos e cronogramas, editor de casos de uso, *softwares* de gestão de projetos, programas de automação de testes, compiladores, *frameworks* de desenvolvimento até, ferramentas com diversas funcionalidades integradas.

Abrangendo grande parte do processo de *software*, as ferramentas CASE são de grande importância no auxilílio dos profissionais que atuam em um processo altamente complexo. Uma das etapas nas quais essas ferramentas são de grande apoio é na gestão de configuração do *software*. Descrito no capítulo a seguir, o gerenciamento e configuração de *software* permite maior agilidade no processo de implementação e evolução do *software*.

2.4 CONSIDERAÇÕES FINAIS DO CAPÍTULO

O capítulo tratado, aponta informações referente ao termo *software*, sendo que ele pode ser tratado como a junção do programa com toda a documentação associada e os dados de configurações necessários para que ele seja operado normalmente.

Conforme visto, as etapas do processo de *software* podem variar de acordo com cada organização, da mesma forma podemos dizer que ele é evolutivo, para que se possa atender às necessidades mutáveis do cliente.

Diante da complexidade dos processos de desenvolvimento do *software*, foram criadas diversas ferramentas que auxiliam e contribuem para o andamento do desenvolvimento do mesmo. Essas ferramentas são comumente chamadas de ferramentas CASE.

Visto isso, a seguir é descrito o processo de gerenciamento e configuração de s*oftware*.

3 GERENCIAMENTO DE CONFIGURAÇÃO DE SOFTWARE

Durante a vida útil de um *software*, as mudanças fazem parte do seu caminho, pois ao longo do tempo surgem as necessidades de que os requisitos sejam alterados. (SOMMERVILLE, 2003).

De acordo com Pressman (2011) as mudanças são inevitáveis quando o *software* é construído, dessa forma Sommerville (2003) reforça que o monitoramento de projeto é uma atividade contínua. O gerente deve manter o acompanhamento do andamento do projeto e comparar os progressos e custos reais com os que foram planejados.

A medida que se desenvolve os sistemas, são criadas muitas versões em uso e em desenvolvimento ao mesmo tempo. Para Sommerville (2003) é necessario manter o controle das mudanças que foram implementadas e de como essas mudanças foram incluídas no *software.*

Segundo Pressman (2006), as mudanças aumentam o nível de confusão entre os colaboradores de um projeto, necessitando de um controle de modo que se melhore a qualidade se reduza o erro.

A gestão da configuração do *software* é uma atividade chamada por Pressman (2011) de “guarda-chuva”, pois é aplicada em toda a gestão de qualidade.

Os tópicos a seguir são constituídos de uma descrição detalhada do gerenciamento de configuração de *software*, assim como os principios que norteiam as etapas de construção e finalização do mesmo.

3.1 GERENCIAMENTO DE CONFIGURAÇÃO DE SOFTWARE

Conforme Pressman (2006), o gerenciamento de configuração de *sofware* é um conjunto de atividades de controle e rastreamento que começa quando um projeto de desenvolvimento inicia e só é finalizado quando o *software* é tirado de operação.

Babich (apud PRESSMAN, 2011) afirma que:

> A arte de coordenar desenvolvimento de *software* para minimizar... a confusão é chamada de gestão de configuração. A gestão de configuração é a arte de identificar, organizar e controlar modificações no *software* que está sendo criado por uma

> equipe de programação. O objetivo é maximixar a produtividade minimizando os erros.

A garantia da qualidade de um *software* é um elemento muito importante no gerenciamento de configuração de *software* (*Software Configuration Management* – SCM), sendo que a principal responsabilidade deste processo é o controle de mudanças (PRESSMAN, 2006).

Para Pressman (2011) as mudanças podem ocorrer a qualquer instante, as atividades SCM são desenvolvidas para (1) identificar a alteração, (2) controlar a alteração, (3) assegurar que a alteração esteja sendo implementada corretamente e (4) relatar as alterações a outros interressado.

Conforme Sommerville (2003), um plano de gerenciamento de configuração descreve os padrões e os procedimentos que devem ser utilizados para o gerenciamento de configuração.

De acordo com Pressman (2011), conforme vão avançando os trabalhos de engenharia de *software,* vão se criando hierarquias de itens de configuração de *software*. Esses itens são informações com nome e podem ser pequenos como um simples diagrama UML, ou grande como um documento de projeto completo.

Na ótica de Pressman (2011) a gestão de configuração de *software* é um conjunto de atividades que são desenvolvidas para gerenciar todas as alterações durante todo o ciclo de vida do *software*. Deste modo, nas seções seguintes serão descritas as pincipais tarefas e itens que compõe o gerenciamente de configuração de *software*.

3.1.1 Baselines

Para Pressman (2006) uma *linha básica* (ou *baseline*) é um conceito de gerenciamento de configuração de *software* que nos ajuda a controlar as mudanças, sem impedir seriamente as mudanças justificáveis.

Segundo Sommerville (2003) os sistemas controlados são, algumas vezes, chamados de *baselines*, quando são o ponto de partida para o desenvolvimento controlado.

Baseline pode ser definida como um marco de referência no desenvolvimento de um *software*, sendo caracterizada pela entrega e aprovação de um ou mais itens de configuração (PRESSMAN, 2006).

3.1.2 Itens de configuração de software

Itens de configuração de *software* são informações criadas como parte do processo de engenharia de *software* (PRESSMAN, 2011).

De acordo Pressman (2006), os itens que compreendem todas as informações produzidas como parte do processo de engenharia de *software* são chamados coletivamente de *configuração de software*.

O *Software Configuration Itens* – SCI ou também chamado de itens de configuração de *software*, cresce rapidamente a medida que avança o processo de engenharia (Pressman,2006).

Pressman (2011) define:

> Um item de configuração de *software* como a informação que é criada como parte do processo de engenharia de *software*. Ao extremo, um SCI poderia ser considerado uma única seção de uma grande especificação ou um caso de teste num grande conjunto de testes

Com o desenvolvimento de um sistema de *software* de grande porte, muitos documentos estão sujeitos a frequentes mudanças. Esses documentos podem ser de interrese de alguém, ou mesmo pessoas que queiram manter um histórico do projeto. Dessa forma, Sommerville (2003) diz que durante o processo de planejamento de gerenciamento de configuração, decide-se exatamente quais itens, (ou classes de itens) devam ser controlados.

Os documentos, conjuntos inteiros de casos de teste ou componentes de programas com um nome, são SCIs que tornam-se alvos das técnicas de gerenciamento da configuração, formando assim um conjunto de linhas básicas descritos abaixo, conforme descrição de Pressman (2006).

1- Especificação do Sistema
2- Plano de Projeto do *Software*
3-
a. Especificação dos Requisitos de *Software*
b. Prototipo executável ou “em papel”
4- Manual Preliminar do Usuário
5- Especificação de Projeto
a. Descrição do Projeto de Dados
b. Descrição do Projeto arquitetural
c. Descrições do Projeto modular
d. Descrições do Projeto de interfaces
e. Descrições do objeto (se forem usadas técnicas orientadas a objetos)
6- Listagem do código-fonte
7-
a. Plano e Procedimentos de Testes
b. Casos de teste e resultados registrados
8- Manuais Operacionais e de instalações

9- Programa executável
a. Módulos – código executável
b. Módulos interligados
10- Descrição do banco de dados
a. Esquema e estrutura de arquivo
b. Conteúdo inicial
11- Manual feito de acordo com o usuário
12- Documentos de manutenção
a. Relatórios de problemas de *software*
b. Solicitações de manutenção
c. Pedidos de mudança de engenharia
13 – Padrões e procedimentos para engenharia de *software*.

3.2 O PROCESSO DE GERENCIAMENTO DE CONFIGURAÇÃO DE SOFTWARE

O Gerenciamento de configuração de *software*, também conhecido como SCM (*Software Configuration Management*) é uma base importante para garantir uma boa qualidade do *software*, ele tem como responsabilidade o controle das mudanças, a identificação de itens de configuração e o versionamento do *software* (PRESSMAN, 2006).

3.3 IDENTIFICAÇÃO DE OBJETOS NA CONFIGURAÇÃO DE SOFTWARE

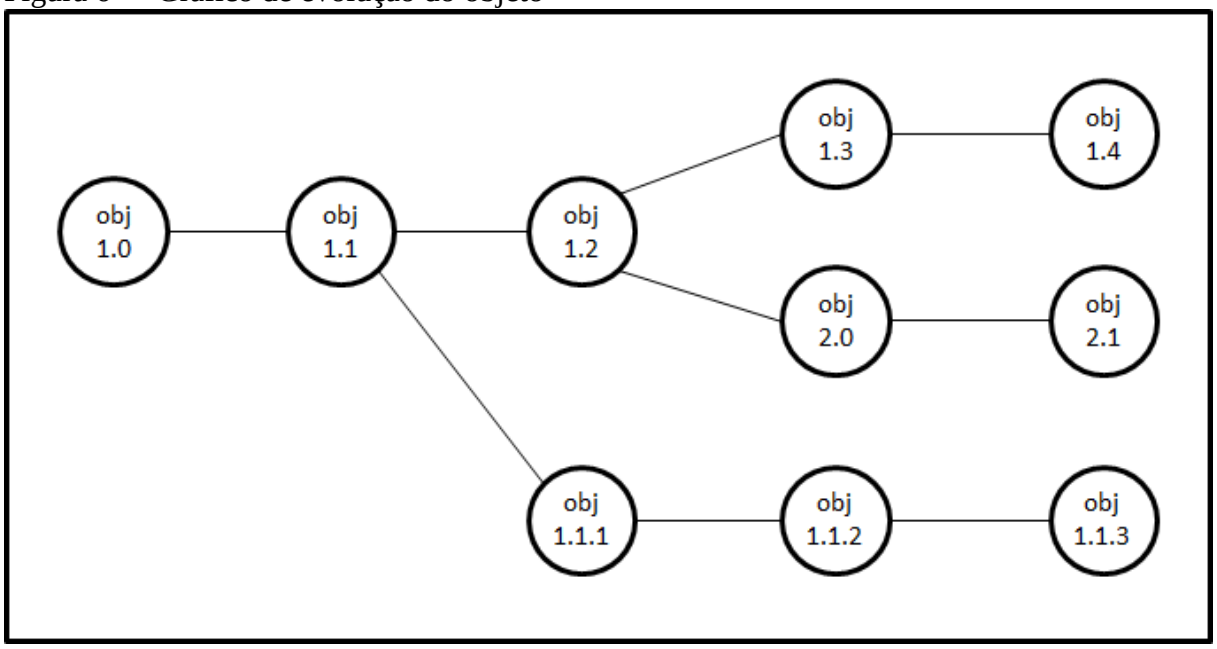
Para administrar, assim como controlar os itens de configuração de *software*, cada item deve ser nomeado separadamente para depois ser organizado, usando uma abordagem orientada a objeto. São identificados a partir de dois tipos de objetos: básicos e compostos. (PRESSMAN, 2006)

Segundo Pressman (2006) um objeto básico é uma "unidade de texto" que foi criada por um engenheiro de *software* durante a análise, o projeto, a codificação ou o teste. Para Pressman (2006), um objeto composto é uma coleção de objetos básicos e outros objetos compostos. Como exemplo de objeto composto, podemos citar a especificação de projeto, quem contém diversos objetos básicos, como o projeto de dados, projeto arquitetural, projeto modular e projeto de interfaces.

Pressman (2006) diz que o esquema de identificação de objetos de *software* deve reconhecer que os mesmos são evoluidos ao longo do projeto, essas evoluções são frequentes

e podem ser mapeadas através de um gráfico da evolução, conforme imagem a seguir. O gráfico da evolução contém o histórico de mudanças do objeto.

Figura 6 – Gráfico de evolução do objeto



Fonte: adaptada de PRESSMAN, 2006.

3.4 CONTROLE DE VERSÃO

De acordo com Sommerville (2003) o gerenciamento de versões e releases consiste no processo de identificar e companhar o desenvolvimento de diferentes versões e releases de um sistema.

Pressman (2006) diz que o controle de versão combina procedimentos e ferramentas para gerenciar diferentes versões de objetos de configuração.

No capítulo 4 será descrito detalhadamente o que é, como funciona e quais os benefícios do controle de versão.

3.5 CONTROLE DE MUDANÇAS

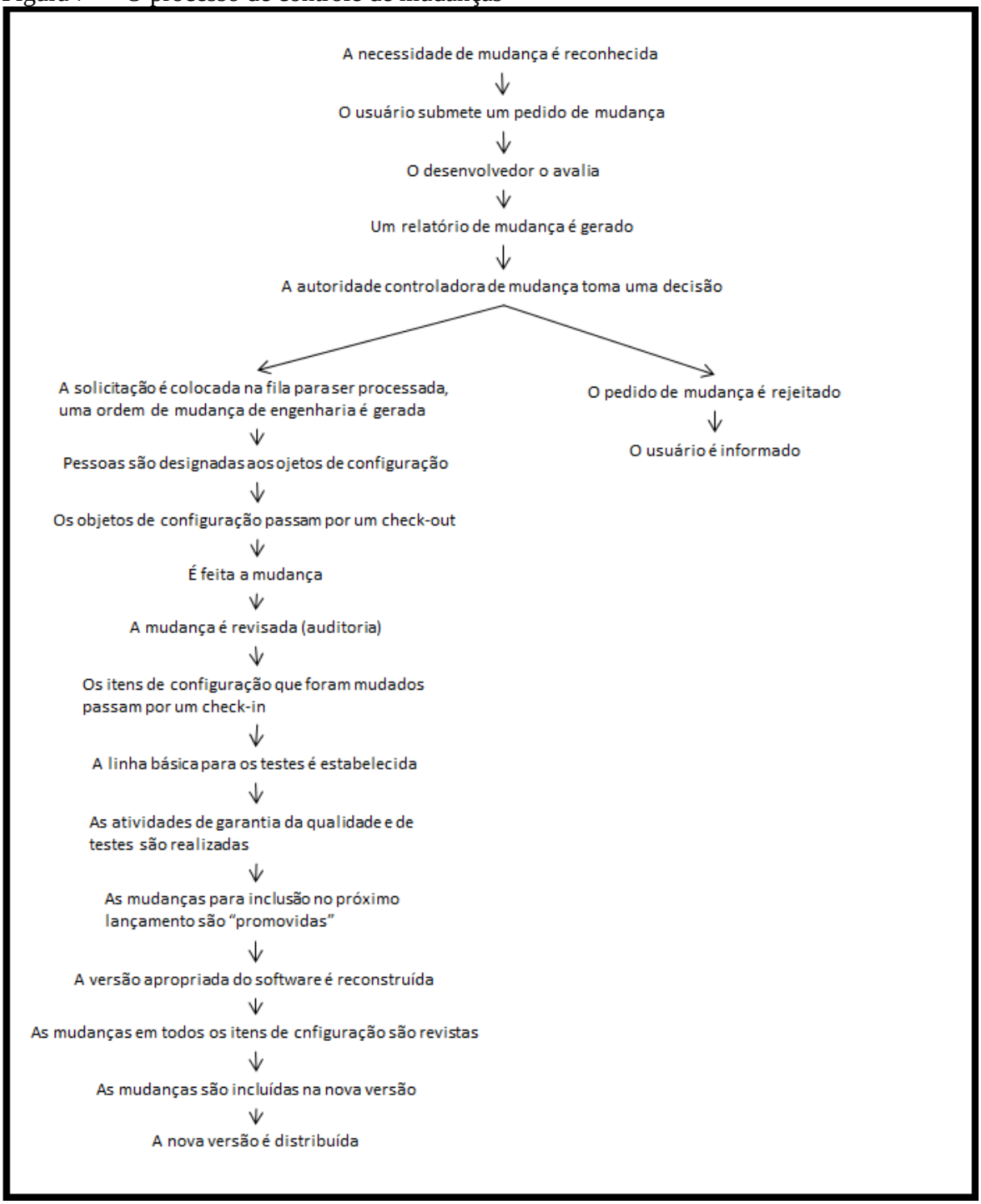
Para Sommerville (2003) as mudanças são um ‘fato da vida’ para os sistemas de *software* de grande porte. Com o tempo ocorrem mudanças nas organizações necessitando de novos requisitos dos sistemas.

De acordo com Sommerville (2003) o processo de gerenciamento de mudanças deve entrar em ação quado o *software*, ou a documentação associada, estiver sob o controle da equipe de gerenciamento.

Segundo Pressman (2006), a combinação de procedimentos humanos e ferramentas automatizadas proporcionam o controle de mudanças.

Para o processo de mudança, contamos com alguns estagios, desde a necessidade de mudança até a liberação da versão com a respectiva mudança. Todo este processo está descrito na imagem a seguir, coforme Pressman (2006).

Figura 7 – O processo de controle de mudanças



Fonte: adaptada de PRESSMAN, 2006.

3.6 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Na aboradagem do capítulo, foram apresentados itens que são indispensaveis na realização do desenvenvolvimento e manutenção da vida útil do *software*.

Verificou-se que é habitual ocorrer mudanças durante toda a vida do *software*, diante desse fato, ter um plano de gerencimento de configuração torna-se essencial para um bom gerenciamento de configuração de *software*, controlando as mudanças e garantindo a sua qualidade.

Diante deste tema de mudanças e qualidade, é descrito a seguir o controle de versão durante o desenvolvimento e vida útil do *software*.

4 CONTROLE DE VERSÃO NO DESENVOLVIMENTO DE SOFTWARE

Á medida que o número de pessoas de uma equipe de projeto de *software* aumenta, a produtividade global pode sofrer as consequências disso (PRESSMAN, 2006). Este capítulo versa sobre o funcionamento do controle de versão de arquivos, desenvolvimento de *software* e quais os benefícios do controle de versão no desenvolvimento de *software*.

4.1 O QUE É CONTROLE DE VERSÃO

Segundo Oliveira (2012), controle de versão de arquivos é o controle das alterações realizadas no arquivo, afim de que se tenha um histórico de tudo que foi alterado duante sua existência.

Muito utilizado no desenvolvimento de *softwares*, o versionamento de código geralmente é feito com a utilização de ferramentas que permitem o controle de versão, de forma que diferentes usuários possam realizar alterações no código fonte de forma concorrente, liberando todas as alterações em um repositório de dados único (DEVMEDIA, 2014).

Para Molinari (2007), um dos principais mecanismos para permitir o controle sobre as modificações é fazer com que o usuário preencha uma solicitação oficial de alteração. Esta deverá ser avaliada por uma comissão de controle de alteração que será responsável por aprovar e autorizar as alterações. Um histórico da evolução deverá ser mantido junto com um registro das razões para sua alteração.

4.2 DESENVOLVIMENTO DE SOFTWARE

Falbo (2005) diz que a utilização de computadores nas mais diversas áreas do conhecimento humano tem gerado uma crescente demanda por soluções computadorizadas. Afim de atender a demanda por soluções computadorizadas, o desenvolvimento de *software* é uma atividade de crescente importância na sociedade contemporânea.

Fator (2015) diz que o desenvolvimento de *software* é o ato de elaborar e implementar um sistema computacional, isto é, transformar a necessidade de um utilizador ou de um mercado em um produto de *software*. Também é entendido como a aplicação dos processos da engenharia de *software* combinados com a pesquisa das necessidades do produto para desenvolver *software*.

4.3 O CONTROLE DE VERSÃO NO DESENVOLVIMENTO DE SOFTWARE

Para Oliveira (2012), o versionamento de arquivos pode evitar que arquivos ou documentos sejam alterados por duas pessoas em paralelo e um sobrescreva a alteração do outro, permitindo também que seja resgatada uma versão específica de um arquivo, evitando que se percam arquivos funcionais que foram alterados e apresentaram problemas.

Conforme Dias (2011), o controle de versão permite que ao longo do desenvolvimento de *software* seja registranda cada alteração realizada no código fonte, sendo possível descobrir quem fez, o que fez e quando fez, permitindo que seja feito o resgate de uma versão específica.

Molinari (2007) diz que a adoção de ferramentas é imprescindível para o sucesso do controle de versão. Existem no mercado ferramentas gerenciam o controle de versão, que facilitam a gestão dos arquivos. Essas ferramentas, por exemplo, permitem operações como a comparação de qualquer arquivo versionado para cada versão, ou seja, é possível comparar o código fonte de uma classe da versão 1.0 com o código fonte da mesma classe já na versão 2.0. Pode-se recuperar um objeto de uma versão anterior ou bloquear um objeto que esta sendo usado por um dos integrantes da equipe, de forma que os demais não tenham acesso a

ele. Normalmente as ferramentas de apoio ao controle de versão no processo de desenvolvimento de *software* permitem recursos descritos abaixo (MOLINARI, 2007):

- *Get Latest Version*: copia a última versão do arquivo presente no banco de dados da ferramenta para a máquina local;
- *Check-out*: copia a última versão do arquivo presente no banco de dados da ferramenta para a máquina local e bloqueia o mesmo para que ninguém mais possa alterá-lo;
- *Check-in*: grava as alterações feitas no arquivo no banco de dados, liberando para que outra pessoa possa alterá-lo;
- *Undo check-out*: desfaz o comando *check-out*, liberando os arquivos para edição de outras pessoas;
- *Label*: permite a criação de rótulos para versões (linhas de base);
- *History*: permite a visulização do histórico de alterações no arquivo, onde a partir dele é possível comparar versões do mesmo arquivo.

O controle de versão possibilita que vários desenvolvedores trabalhem em paralelo, da forma que a alteração de um não seja sobrescrita pela alteração de outro que está editando no mesmo momento (DIAS, 2011).

Quando se tem o código versionado, é possível fazer com que haja mais de uma linha de desenvolvimento do mesmo *software* sem haver conflito na utilização dos arquivos, podendo haver um código fonte de produção e outro de desenvolvimento, por exemplo (DIAS, 2011).

4.4 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Neste capítulo, foi exposto o funcionamento do controle de versão de arquivos, bem como os benefícios do versionamento de arquivos no desenvolvimento de *software*. Essas informações são grande importância para o entendimento de como funciona o controle de versão de código fonte no desenvolvimento de *software.*

5 ESTUDO DE CASO

Neste capítulo, é apresentada a empresa estudo de caso, é descrito o seu processo de desenvolvimento de *software*, dando ênfase ao controle de configuração. Inicialmente é descrito o processo de desenvolvimento antes de serem aplicadas as práticas de controle de versão e configuração. A seguir é descrito o processo que é atualmente executado, com o apoio de uma ferramenta CASE. Com base no processo de desenvolvimento atual, após a apresentação dos dois processos são destacadas as diferenças entre ambos. O processo atual foi analisado com base nas referências estudadas, para então ser avaliado.

5.1 A EMPRESA ESTUDO DE CASO

Com o objetivo de preservação da imagem da empresa estudada, não serão divulgadas informações que possibilitem sua identificação. A empresa é uma entidade dedicada ao desenvolvimento de *software*, situada no estado de Santa Catarina, com foco no desenvolvimento de *software* de comércio eletrônico.

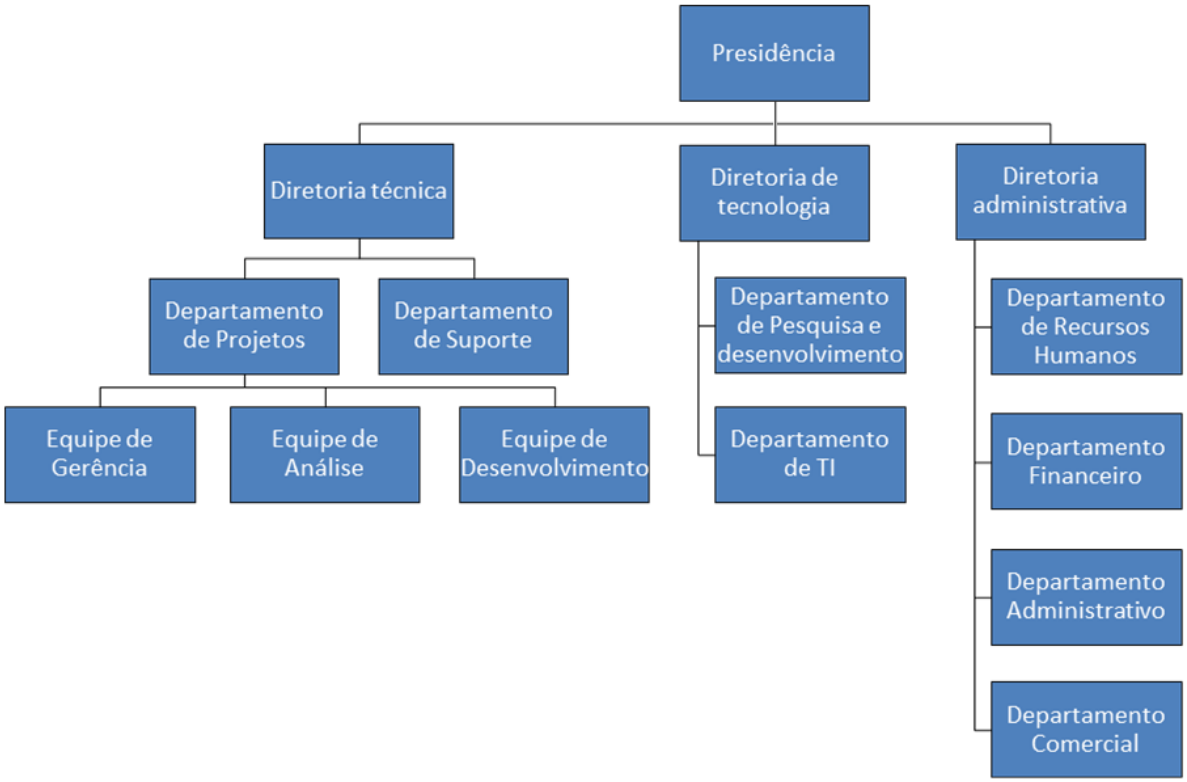
Atualmente, a empresa conta com aproximadamente 100 colaboradores.

5.1.1 Organograma

A estrutura organizacional da empresa estudada respeita a sua divisão interna, onde existe um presidente e três diretores que fazem a gestão dos diretores existentes.

O organograma exibido abaixo foi desenvolvido pelo autor, visto que a empresa não divulgou seu organograma.

Figura 8 - Organograma da empresa estudo de caso



Fonte: Autor, 2015.

5.2 PROCESSO DE DESENVOLVIMENTO DE SOFTWARE

O processo de desenvolvimento de *software* é executado por aproximadamente 60 colaboradores que atuam diretamente nesta área. A seguir será apresentado o processo de desenvolvimento de *software* antes da aplicação das práticas de controle de versão e configuração e o processo atual, com uma comparação entre esses dois processos.

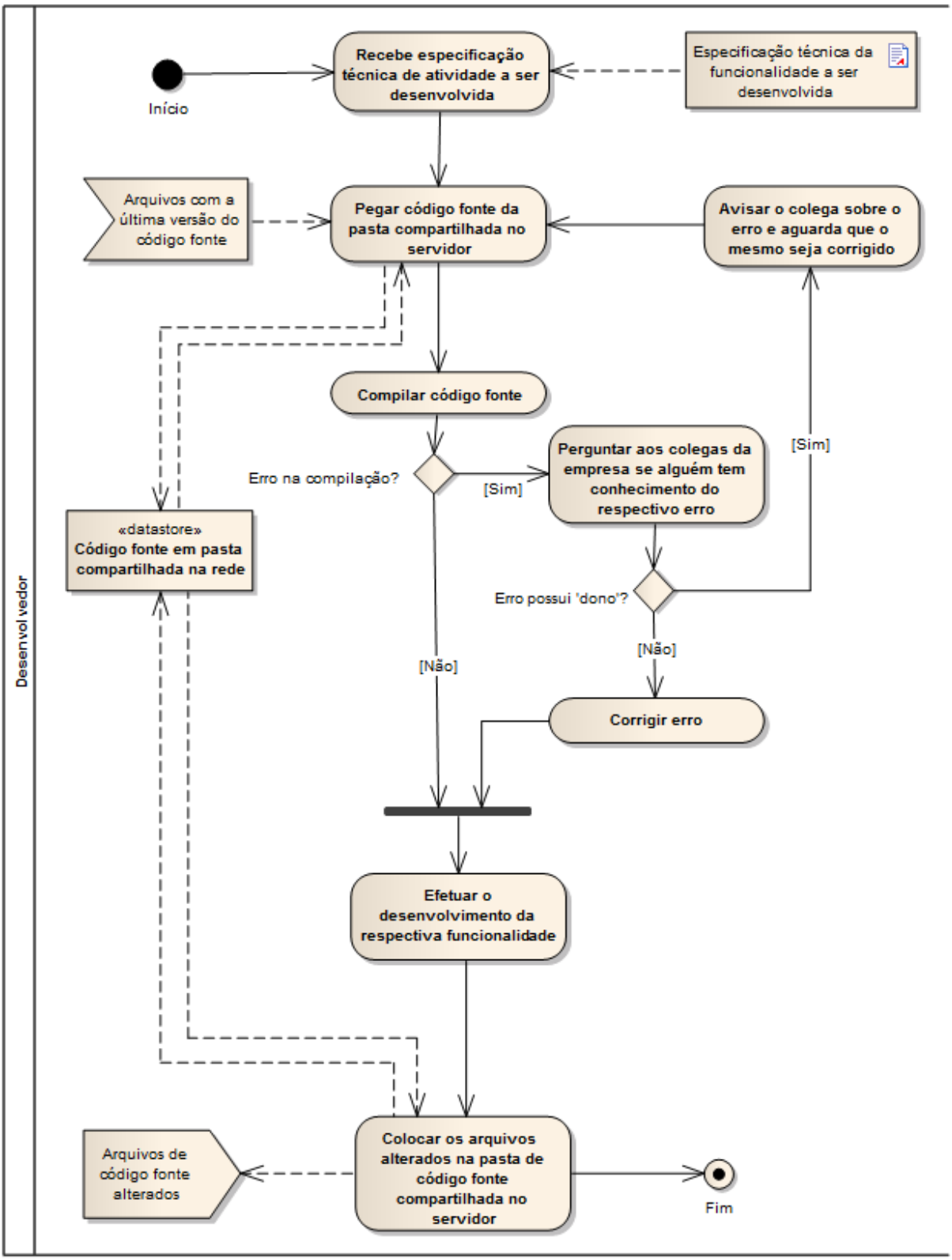
5.2.1 Processo de desenvolvimento anterior

Nesta sessão é apresentado o processo antigo de desenvolvimento de *software* realizado na empresa estudo de caso.

O levantamento do processo antigo foi realizado através de entrevistas informais realizadas com diversos colaboradores da empresa, dentre eles, o diretor de tecnologia, o diretor técnico e diversos integrantes da equipe de desenvolvimento. Todos os entrevistados vivenciaram o processo de desenvolvimento antigo. Cabe destacar que o autor desta monografia trabalha profissionalmente na empresa há sete anos, quando esse processo ainda era realizado, e foi um dos artífices nas mudanças que serão apresentadas no processo atual.

O processo de desenvolvimento anterior é representado na figura na continuidade.

Figura 9 - Processo de desenvolvimento anterior



Fonte: Autor, 2015.

A figura é bem representativa de como as tarefas eram realizadas e quais atividades eram (ou não) desenvolvidas. Nela pode-se observar que não havia um controle formal de acesso aos códigos fontes por parte dos desenvolvedores, bem como a não existência de vínculo da especificação técnica com o desenvolvimento da atividade.

A seguir é apresentado o processo atual de desenvolvimento de *software.*

5.2.2 Processo de desenvolvimento atual

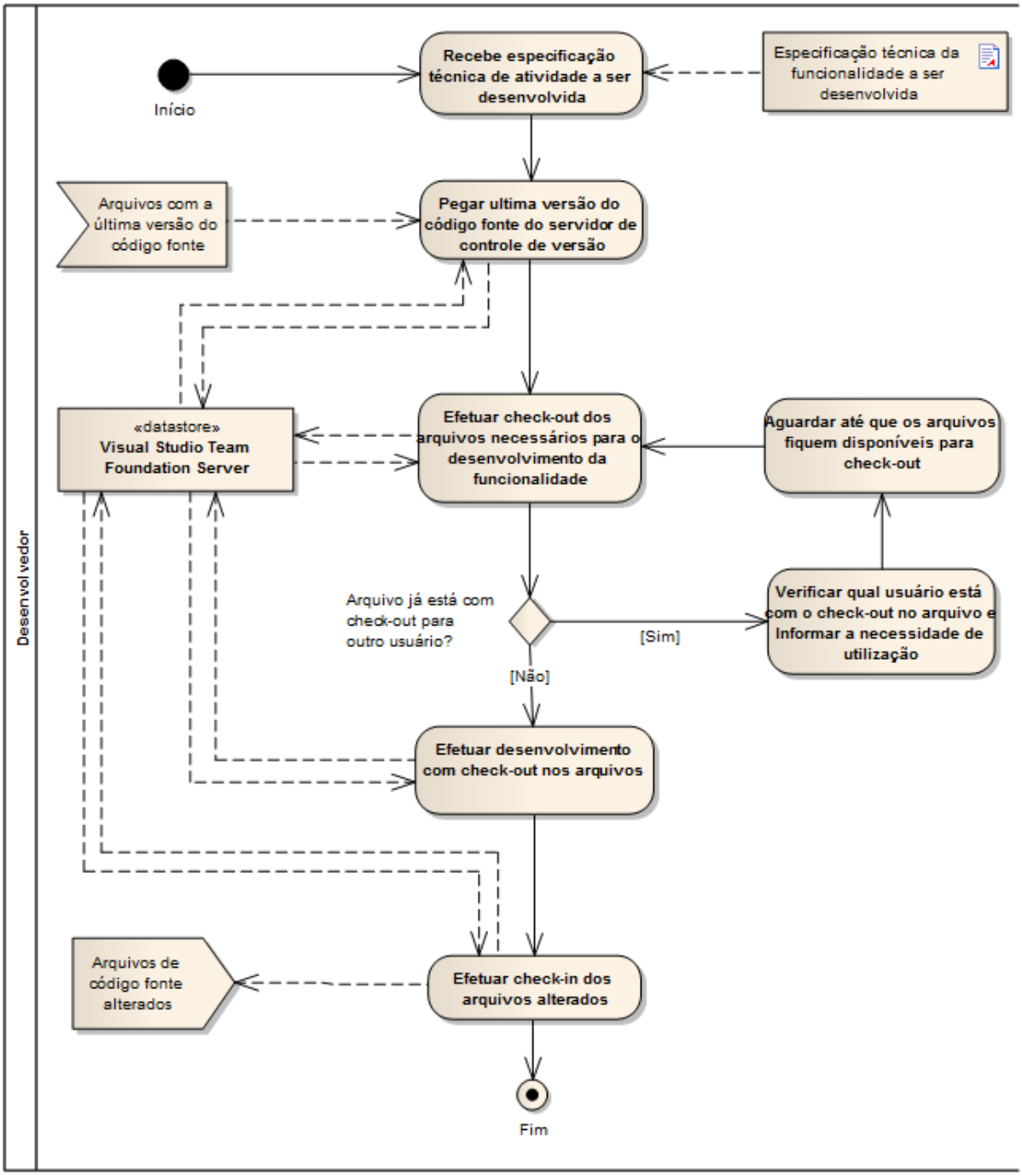
Com um mercado bastante competitivo, a empresa visa contornar os problemas do anterior ciclo de desenvolvimento, buscando realizar um processo com melhores indicadores de qualidade, diversas alterações foram propostas. Para alcançar o objetivo, a proposta mais significativa foi o uso de uma ferramenta CASE, o *Team Fundation Server* – TFS – da Microsoft, ferramenta esta, que integra totalmente com as tecnologias e linguagens já utilizadas pela empresa estudo de caso.

A decisão da incorporação de uma ferramenta para apoio na gestão de configuração foi tomada pelo diretor de tecnologia da empresa, que possuia conhecimento do assunto e da tecnologia a ser implantada. Em um primeiro momento, a respectiva ferramenta foi instalada em um ambiente para que o mesmo diretor efetuasse os testes, após a sua aceitação, o mesmo decidiu ativar a ferramenta em apenas uma equipe de desenvolvimento para analisar o desempenho da ferramenta no dia a dia. Posteriormente, esse novo processo foi implantado em todas as equipes de desenvolvimento da empresa estuo de caso.

A transição foi realizada rapidamente e bem aceita por toda a equipe de desenvolvimento. Todos se adaptaram em um curto período de tempo, visto que a ferramenta de desenvolvimento continuou a mesma, porém, complementada com o TFS, que definiu muito bem a gestão de condiguração.

A figura a seguir apresenta o processo atual de desenvolvimento de *software* da empresa estudo de caso.

Figura 10 – Processo de desenvolvimento atual



Fonte: Autor, 2015.

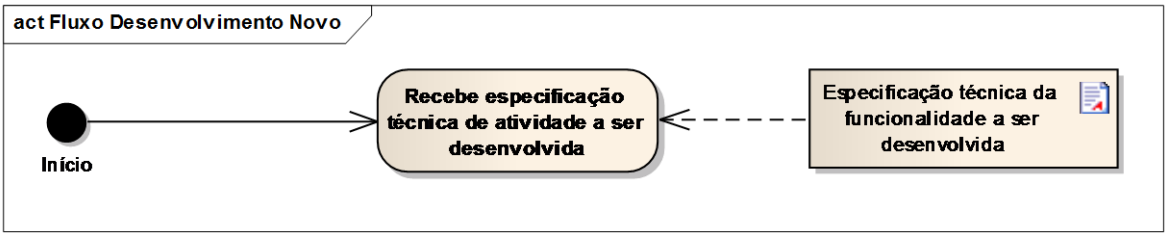
No processo de desenvolvimento de *software* atual, ilustrado pela figura anterior, é considerado apenas um autor, o desenvolvedor. Este é o ator principal no processo de desenvolvimento de *software*, ele é responsável por desenvolver/codificar as funcionalidades. Na empresa estudo de caso, todas as atividades a serem desenvolvidas possuem especificação

técnica, toda especificação técnica está associada a uma *User Storie* – US – cadastrada na ferramenta de apoio.

A seguir são descritas cada uma das atividades consideradas no processo de desenvolvimento de *software* atual, que foi ilustrado anteriormente.

A figura 11 define o início do desenvolvimento de uma atividade, que inicia a partir de uma especificação técnica.

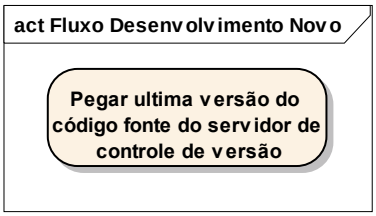
Figura 11 – Novo desenvolvimento



Fonte: Autor, 2015.

Com a documentação da demanda, está representada na figura 12, a ação em que o desenvolvedor deve solicitar no sistema de controle de versão, a última versão do código fonte, para então compilar a solução e começar os trabalhos.

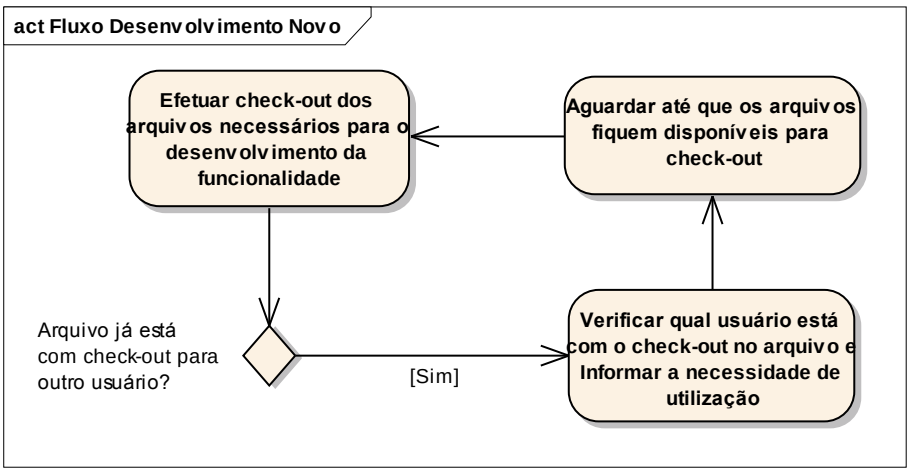
Figura 12 – Resgate da última versão do código fonte



Fonte: Autor, 2015.

Após ter a última versão do código fonte, o desenvolvedor deve efetuar *check-out* dos arquivos que serão utilizados durante o desenvolvimento da atividade. Esta ação pode acontecer durante todo o desenvolvimento da atividade, visto que o desenvolvedor pode efetuar *check-out* dos arquivos conforme sua necessidade. Ao efetuar *check-out* de arquivos, o desenvolvedor pode se deparar com uma mensagem alertando que o respectivo arquivo encontra-se com *check-out* de outro usuário, nesse caso, é necessário aguardar até que o arquivo seja liberado, conforme representado na figura 13. Conforme Dias (2011), esse controle permite exclusividade do arquivo, não sendo possível a edição do mesmo arquivo por mais de uma pessoa.

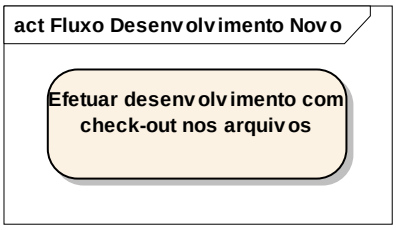
Figura 13 – *Check-out* dos arquivos necessários para edição



Fonte: Autor, 2015.

Com os arquivos reservados através da operação *check-out*, o desenvolvedor então pode realizar o desenvolvimento de sua atividade, conforme figura 14.

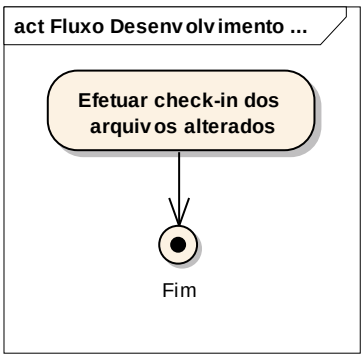
Figura 14 – Desenvolvimento da atividade



Fonte: Autor, 2015.

Por fim, conforme a figura 15, acontece a conclusão do desenvolvimento. Aqui o desenvolvedor precisa enviar as alterações realizadas para o repositório de código fonte. Esssa ação é realizada através da plataforma *Microsoft Visual Studio*. Através do uso da ferramenta, o desenvolvedor obrigatoriamente terá que associar a ação de *check-in* com um item de trabalho, assim é possível efetuar o rastreamento da atividade, desde o início até o fim do desenvolvimento.

Figura 15 – *Check-in* dos arquivos alterados



Fonte: Autor, 2015.

Ao possuir um novo *check-in*, o TFS inicia uma *build* automatizada, efetuadando a validação de integridade da nova versão do código fonte. Isso faz com que seja possível saber a situação do código fonte que está no repositório, que também está disponível para outros desenvolvedores.

O grande diferencial do processo de desenvolvimento antigo para o novo, é a utilização dessa ferramenta integradora que contempla toda a gestão de versionamento de código fonte e *build* do *software*. Tal ferramenta é descrita a seguir.

5.2.3 Tecnologias e ferramentas utilizadas

Para apoio ao processo de configuração atual, a empresa estudo de caso utiliza ferramentas caracterizadas como CASE, que proporcionam a automação de algumas atividades de processo e o fornecimento de informações sobre o *software* que está sendo desenvolvido, gerando melhorias na qualidade e na produtividade de *software* (SOMMERVILLE, 2003).

Como ferramenta integradora da gestão de configuração e versionamento de código, a empresa utiliza o *Team Fundation Server* – TFS. É uma ferramenta com proporciona diversos recursos, sendo que os dois considerados principais para o desenvolvimento de *software* são descritos a seguir (MICROSOFT, 2015a):

Controle de versão: conforme Microsoft (2015a), esse recurso permite o versionamento dos arquivos de forma centralizada. A partir desse, é possível o controle de

check-in e *check-out*, além de operações de *diff* ou *merge*. Essa operação reduz a dificuldade na classificação das versões e confusão na detecção de qual versão é a corrente, assim como na integração de versões quando há acesso concorrente no mesmo arquivo (DEVMEDIA 2014). Dias (2011) diz que o esse recurso permite que ao longo do desenvolvimento de *software* seja registranda cada alteração realizada no código fonte, sendo possível descobrir quem fez, o que fez e quando fez, permitindo que seja feito o resgate de uma versão específica.

Compilação (*build*): este recurso permite a configuração de *builds* que são executadas diretamente no servidor. Essas *builds* podem ser acionadas manualmente ou a partir de um gatilho previamente configurado. Esse gatilho pode ser o ato de envio de código fonte para o servidor, essa operação, por exemplo, pode acionar a execução de uma *build* de integração contínua, que garante a integridade do código fonte (MICROSOFT 2015b).

Os dois itens descritos acima mudaram completamente a forma com que a empresa armazenava os códigos fontes de seu *software*. Com a adesão do TFS, a empresa conseguiu também ter o controle de *builds*, sendo possível a automatização e padronização do processo de geração de *builds*.

Para o desenvolvimento, a empresa utiliza o Microsoft Visual Studio, que é uma ferramenta de desenvolvimento que está totalmente integrada ao *Team Foundation Server*. Conforme Microsoft (2015c), o Microsoft Visual Studio é um pacote de programas para desenvolvimento de *software* dedicado ao .NET *Framework* e às linguagens *Visual Basic* – VB, C, C++, C# e J#. Voltado para aplicações *Windows Forms* e *Web*.

5.3 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Comparando o processo de desenvolvimento de *software* anterior da empresa e o atual, percebe-se que a maior diferença está na automatização do versionamento e controle de código fonte através de uma ferramenta CASE. As atividades do processo antigo, de certa forma, se mantiveram no processo novo.

É muito importante a realização do versionamento e controle de código fonte de um *software*, pois, através dele, a empresa possui um histórico de todas as alterações

realizadas em cada arquivo, isso permite que a rastreabilidade seja realizada. A ferramenta proporcionou um repositório único de código fonte do *software*.

Através da ferramenta CASE adotada pela empresa, também foi possível a gestão do controle de acesso aos códigos fonte, esse controle faz com que haja muito menos retrabalho, além da automatização de *builds* que garantiu maior integridade ao código fonte contido no servidor.

6 CONCLUSÃO E TRABALHOS FUTUROS

Este capítulo apresenta os resultados obtidos no estudo de caso do processo de desenvolvimento de *software* em uma empresa de tecnologia. O estudo de caso visa descrever a identificação das melhorias realizadas no processo de desenvolvimento de *software* da empresa estudada.

6.1 ALCANÇANDO OS OBJETIVOS

Com base no objetivo principal deste trabalho, pode-se constatar a importância das ferramentas CASE na etapa de desenvolvimento de *software*.

Os passos para atingir este objetivo foram descritos nos objetivos específicos.

Primeiramente havia o objetivo de descrever sobre a utilização de ferramentas CASE no processo de desenvolvimento de software. A análise através de um estudo de caso foi escolhida pois é possível atingir esse objetivo. Foi analisado o processo de desenvolvimento de *software* praticado anteriormente na empresa estudo de caso sem o versionamento de código e o processo de desenvolvimento de *software* atualmente executado na empresa.

O primeiro passo para realização do estudo de caso foi estudo da literatura e o desenvolvimento de uma revisão bibliográfica do assunto a ser abordado. Assim, foram descritos o processo de *software*, o gerenciamento de configuração de *software* e o controle de versão no desenvolvimento de *software*, presentes nos capítulos 2, 3 e 4 deste trabalho.

O capítulo 5 apresenta o estudo de caso realizado na empresa em que o autor atualmente atua profissionalmente. A familiarização do autor com o processo de desenvolvimento de *software* anterior e o atual realizado na empresa estudo de caso deu maior agilidade no desenvolvimento da análise.

6.2 RESULTADOS E TRABALHOS FUTUROS

O resultado deste trabalho é um estudo geral sobre o processo de *software*, voltado ao gerenciamento de configuração de *software*. Com base nesse estudo, foi realizado um estudo de caso do processo de desenvolvimento de *software* anterior e o atual realizado na empresa estudo de caso, com foco no versionamento de código fonte.

Para a realização do estudo de caso, foram analisados os processos de desenvolvimento anterior e o atual, sendo que o autor contou com o apoio de colaboradores que vivenciaram o processo de desenvolvimento de *software* anterior e o atual, sendo possível a definição dos dois processos.

O estudo de caso proporcionou o aprendizado das boas práticas da gestão de configuração de *software* voltado ao versionamento de código fonte, bem como a análise da importância da ferramenta CASE adotada pela empresa. Nessa fase foi possível perceber que o versionamento de código fez com que não existisse mais perda de código desenvolvido.

O estudo de caso realizado neste trabalho foi analisado com base em um processo já existente, a partir disso, o autor compreende que o atual processo de desenvolvimento de *software* com versionamento de código fonte, atende os padrões da empresa, porém sugere como trabalho futuro, a análise e realização de um estudo de caso voltado ao versionamento de banco de dados.

REFERÊNCIAS

DEMO, Pedro. **Pesquisa e construção de conhecimento.** Rio de Janeiro: Tempo Brasileiro, 1996.

DIAS, André Felipe. **Conceitos Básicos de Controle de Versão de Software — Centralizado e Distribuído**. Artigo retirado do site da empresa Pronus Engenharia de Software, publicado 2011. Disponível em: <http://pronus.eng.br/artigos_tutoriais/gerencia_configuracao/conceitos_basicos_controle_versao_centralizado_e_distribuido.php> Acessado em 20 de janeiro 2015.

FALBO, Ricardo de Almeida. **Engenharia de software**. Publicado em 2005. Disponível em: <http://www.inf.ufes.br/~falbo/download/aulas/es-g/2005-1/NotasDeAula.pdf>. Acesso em 02 de fevereiro. 2015.

FATOR. **Desenvolvimento de Software**. Conteúdo retirado do site da empresa Fator Sistemas & Engenharia. Disponível em: <http://www.fatorsistemas.com.br/novo/desenvolvimento.php> Acessado em 20 de janeiro 2015.

GIL, Antonio Carlos. **Como elaborar projetos de pesquisa**. 4ª Ed. São Paulo: Atlas, 2002.

MICROSOFT. **Team Foundation Server**. Disponível em: <http://www.visualstudio.com/pt-br/products/tfs-overview-vs/>. Acesso em 02 de fevereiro. 2015a.

MICROSOFT. **Compilando arquivos no Visual Studio**. Disponível em: <https://msdn.microsoft.com/pt-br/library/vstudio/cyz1h6zd.aspx>. Acesso em 02 de fevereiro. 2015b.

MICROSOFT. **Edições do Visual Studio**. Disponível em: <http://www.microsoft.com/visualstudio >. Acessado em 03 de fevereiro 2015a.

MOLINARI, Leonardo. Gerência de Configuração – Técnicas e Práticas no desenvolvimento do Software. Florianópolis: Visual Books, 2007.

OLIVEIRA, Felipe Cesar. **Controle de versão e sua importância**. Artigo retirado de seu blog pessoal, publicado 2012. Disponível em: <http://felipecsr.wordpress.com/tag/controle-de-versao>. Acessado em 20 de janeiro 2015.

PRESSMAN, R. **Engenharia de Software**. 6ª ed. - São Paulo: McGraw-Hill, 2006.

PRESSMAN, R. **Engenharia de Software**. 7ª ed. - São Paulo: McGraw-Hill, 2011.

SILVA, E. L. da MENEZES, E.M. **Metodologia da Pesquisa e Elaboração de Dissertação**. 4ª ed. - Florianópolis 2005. Disponível em: < https://projetos.inf.ufsc.br/arquivos/Metodologia_de_pesquisa_e_elaboracao_de_teses_e_dissertacoes_4ed.pdf>. Acesso em 19 de janeiro 2015.

SOMMERVILLE, Ian. **Engenharia de Software**. 6ª ed. - São Paulo: Addison Wesley, 2003.

SENDIN, Rodrigo. **Visual Studio Team System e TFS.** 67ª ed. – São Paulo: .NET MAGAZINE, 2009.