Estudo do ciclo de vida do *software* em uma empresa de desenvolvimento de sistemas[1]

Fabrício Luis Marinheiro Lourenço [2]
Márcio Alan Benine [3]

Resumo: Um estudo do ciclo de vida do *software* desenvolvido em uma empresa de desenvolvimento de sistemas é importante, pois construir um *software* com qualidade demanda a utilização e implantação de processos e técnicas de engenharia de *software*. Este processo é indispensável para que o produto seja entregue ao cliente dentro do prazo e orçamento planejado, alcançando a qualidade esperada. Considerando-se esse procedimento, realizou-se um estudo utilizando-se uma pesquisa bibliográfica, visando encontrar referencial teórico para dar sustentação à questão proposta e um estudo de caso na empresa *Nippon* Informática Ltda ME. Esta empresa localizada na cidade de Batatais/SP, atua com desenvolvimento de *software* para as áreas Contábil, Fiscal e Recursos Humanos. No estudo de caso buscou-se elaborar o mapeamento dos processos existentes e após análise dos mesmos, propor um cenário de solução adequado à realidade da empresa. Como resultado final deste estudo, foi apresentada uma proposta de implantação do modelo de ciclo de vida do *software*, proporcionando a construção do software com qualidade; fornecendo um modelo que atenda aos padrões da engenharia de *software* e que tenha aspectos de qualidade importantes. Concluiu-se que, cada vez mais, empresas de desenvolvimento de sistemas necessitam adotar processos de *software* adequados. A definição do ciclo de vida de um *software* é importante para se ter a visão completa do desenvolvimento do *software*. Com isto, foi possível definir etapas que abrangem desde a análise dos requisitos até a entrega do *software* para o cliente.

Palavras-chave: Engenharia de *software*. Processos de *software*. Requisitos. Ciclo de vida. Qualidade.

---

[1]Orientadora: Débora Camila Cordeiro da Trindade Silva. Especialista em Banco de Dados pela Universidade de Belo Horizonte (UNI-BH). Especialista em Gestão da Logística pelo Instituto de Educação Tecnológica (IETEC). Graduada em Tecnologia em Processamento de Dados pela Faculdade Brasileira de Informática (FABRAI). Docente do Centro Universitário Claretiano de Batatais (SP). E-mail: <deboradatrindade@hotmail.com>.

[2]Bacharel em Sistemas de Informação pelo Centro Universitário Claretiano de Batatais (SP). E-mail: <fabricio.marinheiro@hotmail.com>.

[3]Bacharel em Sistemas de Informação pelo Centro Universitário Claretiano de Batatais (SP).

1. INTRODUÇÃO

Este trabalho tem como tema um *"Estudo do ciclo de vida do software em uma empresa de desenvolvimento de sistemas"*, com o intuito de abordar assuntos relacionados à informática, mais precisamente, engenharia, modelos e processos de *software* utilizados em empresas do segmento.

Para alcançar o objetivo proposto, optou-se em realizar uma pesquisa bibliográfica, utilizando como ponto de partida, livros relacionados ao tema, visando encontrar referencial teórico para dar sustentação à questão proposta. Utilizou-se de um estudo prático na empresa *Nippon Informática Ltda ME*, em que foi feito o mapeamento dos processos existentes na empresa e, após análise dos mesmos, elaborado um cenário de solução.

2. ENGENHARIA DE SOFTWAREO termo engenharia de *software* foi criado nas décadas de 60 e 70 (séc. XX), objetivando contornar a crise do *software* e dar um tratamento de engenharia (mais sistemático e controlado) ao desenvolvimento de sistemas de *software* complexos (CARVALHO; CHIOSI, 2001).

Um sistema de *software* complexo se caracteriza por um conjunto de componentes abstratos de *software* (estruturas de dados e algoritmos) encapsulados na forma de procedimentos, funções, módulos, objetos ou agentes interconectados entre si, compondo a arquitetura do *software*, que deverão ser executados em sistemas computacionais.

Engenharia de *software* é uma área do conhecimento voltada para a especificação, desenvolvimento e manutenção de sistemas de *software* aplicando tecnologias e práticas de ciência da computação, gerência de projetos e outras disciplinas, almejando produtividade e qualidade. É uma área mais ampla por tratar de todos os aspectos de sistemas baseados em computadores, incluindo *hardware* e engenharia de processos além do *software*.

Um caso exemplar é o desempenho dos sistemas operacionais da *Microsoft* que não são 100%, mas se for levado em conta a complexidade do *software* com o número de linhas de código (5-10 mil para o *Windows 98*

e 12 milhões para o *Windows XP*) pode-se dizer que a qualidade de certa forma foi garantida. Atualmente, estas tecnologias e práticas englobam linguagens de programação, bases de dados, ferramentas, plataformas, bibliotecas, *standards* e processos.

Os fundamentos científicos para a engenharia de *software* envolvem o uso de modelos abstratos e precisos que permitem ao engenheiro especificar, projetar, implementar e manter sistemas de software, avaliando e garantido suas qualidades. Além disto, a engenharia de *software* deve oferecer mecanismos para se planejar e gerenciar o processo de desenvolvimento.

A engenharia de *software* se concentra nos aspectos práticos da produção de um sistema de *software*, enquanto a ciência da computação estuda os fundamentos teóricos dos aspectos computacionais. A engenharia de sistemas é uma área mais ampla por tratar de todos os aspectos de sistemas baseados em computadores, incluindo *hardware* e engenharia de processos além do *software*.

3. PROCESSOS

Segundo Paula Filho (2003), um processo é uma receita seguida por um projeto; o projeto concretiza uma abstração, que é o processo. Não se deve confundir processo com o produto ou a execução de um processo por meio de um projeto. Por exemplo: uma receita de risoto de camarão (processo), o risoto do camarão (produto), a confecção de um risoto por determinado *chef* (projeto). Um processo é definido quando tem uma documentação especificando o produto (o que é feito), o procedimento (quando), os agentes (por quem), os insumos (o que é utilizado) e os resultados (o que é produzido).

Em engenharia de *software*, segundo Paula Filho (2003), processos são atividades como desenvolvimento, manutenção, aquisição e contratação de *software*. Processo de *software* é um conjunto coerente de práticas que objetiva o desenvolvimento ou evolução de sistemas de *software*. Estas

práticas englobam as atividades de especificação, projeto, implementação e testes e caracterizam-se pela interação de ferramentas, pessoas e métodos.

3.1 Modelos de processos de software

Um modelo de processo de desenvolvimento de *software*, ou simplesmente modelo de processo, pode ser visto como uma representação, ou abstração dos objetos e atividades envolvidas no processo de *software*. Além disso, oferece uma forma mais abrangente e fácil de representar o gerenciamento de processo de *software*, além do progresso do projeto.

De acordo com Pfleeger (2004, p. 36) pode-se "[...] considerar um conjunto de tarefas ordenadas como um processo, ou seja, uma série de etapas que envolvem atividades, restrições e recursos para alcançar o objetivo almejado".

O *modelo Cascata* (do inglês *waterfall*) com fases distintas de especificação, projeto e desenvolvimento, segundo Pressman (1995), se inicia no nível de sistema e avança ao longo da análise, projeto, codificação, teste e manutenção. É considerado o modelo mais antigo e o mais usado no contexto da engenharia de *software*.

Pressman (1995) enfatiza que no *modelo cascata*, também conhecido como modelo clássico, o paradigma do ciclo de vida requer uma abordagem sistemática, sequencial ao desenvolvimento do *software*, que se inicia ao nível do sistema e avança ao longo da análise, projeto, codificação, teste e manutenção.

De acordo com Pressman (1995) o ciclo de vida clássico é o paradigma mais antigo e o mais amplamente usado na engenharia de *software*.

3.1.2 Prototipação

A prototipação, segundo Pressman (1995) é um processo que capacita o desenvolvedor a criar um modelo de *software* que será implementado. Trata-se de um modelo eficiente da engenharia de *software,* tendo como principal fator a concordância entre cliente e desenvolvedor para que o protótipo seja construído para servir como um mecanismo a fim de definir os requisitos.

O modelo de prototipação ou evolutivo, segundo Carvalho e Chiosi (2001) pode ser de dois tipos: desenvolvimento exploratório, em que o objetivo do processo é trabalhar junto com o usuário para descobrir seus requisitos, de maneira incremental até alcançar o produto final; e o protótipo descartável que objetiva entender os requisitos do usuário para obter uma melhor definição dos requisitos do sistema.

Esse modelo é mais eficaz que o modelo cascata, porém apresenta alguns problemas, por exemplo, o processo não é visível, os sistemas são pobremente estruturados, quando um protótipo a ser descartado é construído o usuário faz pressão para algumas mudanças tentando igualá-lo ao produto final e, finalmente, o desenvolvedor faz algumas exceções e assume compromissos de implementações para garantir o funcionamento do produto com rapidez (CARVALHO; CHIOSI, 2001).

3.1.3 Espiral

Para Pressman (1995), o modelo espiral é a abordagem realística para o desenvolvimento de sistema e de *software* em grande escala, pois capacita o desenvolvedor e o cliente e entender e reagir aos riscos em cada etapa evolutiva. Esse modelo usa a prototipação como um mecanismo de redução dos riscos e possibilita que o desenvolvedor aplique a abordagem de prototipação em qualquer etapa da evolução do produto.

O modelo espiral, segundo Paula Filho (2003) desenvolve o produto em uma série de interações em que cada uma nova interação corresponde a uma volta na espiral. Esse procedimento permite construir produtos em

prazos curtos, com novas características e recursos que são agregados à medida que a experiência descobre sua necessidade. O principal problema do ciclo de vida em espiral é que ele requer gestão muito mais sofisticada para ser previsível e confiável.

Existem também os modelos: interativo e incremental. De acordo com Pfleeger (2004) no desenvolvimento incremental, o sistema é dividido em subsistemas por funcionalidades a cada versão. O desenvolvimento nessa abordagem chega lentamente à funcionalidade total, por meio de novos modelos. O desenvolvimento iterativo entrega um sistema completo desde o começo e então muda a funcionalidade de cada subsistema a cada novo modelo versão, o qual se aprimora com novos recursos sobre a versão a anterior.

4. PROJETO DE SOFTWARE Segundo Carvalho e Chiosi (2001) o projeto de *software* envolve a representação das funções do sistema em uma forma que possa ser transformada em um ou mais programas executáveis. Para se conduzir bem um projeto de software, deve-se compreender o escopo do trabalho a ser realizado, os riscos a correr, os recursos exigidos e disponíveis, as tarefas a serem executadas, o custo e a programação a ser seguida. Essa compreensão é proporcionada pela gerência de projetos de *software*.

De acordo com Vieira (2008, p. 3), apesar da divergência que existe entre vários autores a respeito da definição de projeto de *software*, pode-se conceituá-lo como um processo que:

> A partir dos requisitos do *software* e do domínio do problema, decompõe o sistema em componentes e determina seus relacionamentos, especificando suas interfaces, descrevendo suas funcionalidades e identificando oportunidades para o reuso. Ou seja, uma vez que o domínio do problema foi esclarecido, o sistema é projetado com o apoio de uma ou mais metodologias de forma a resolver o problema em questão.

O projeto do *software* é uma etapa crucial para o sucesso no desenvolvimento de qualquer sistema, pois com ele o projetista tem uma visão

ampla do que deve ser feito e aplica a estratégia que melhor atende às suas necessidades.

Ao desenvolverem um artigo sobre projetos de *software*, o autor Reeves concluiu que a única documentação do *software* que atualmente satisfaz o critério de engenharia de *software* é o código fonte. Neste ponto, Vieira (2008) discorda do autor, argumentando que o projeto e a arquitetura do *software* são os critérios mais importantes, pois estes critérios são obtidos com uma boa qualidade, o código fonte pode ser implementado por qualquer programador a partir da documentação.

Considerando-se que uma gerência de projetos de *software* é tão importante para o sucesso de um projeto, seria razoável presumir-se que todos os gerentes de projetos entendem como colocá-la em prática e que todos os profissionais entendem como trabalhar dentro dos limites estabelecidos por ela (PRESSMAN, 1995).

5. GERENCIAMENTO DE PROJETOS

Segundo Zanoni (2009), parte dos problemas relacionados aos projetos de *software* deve-se, principalmente, a problemas de administração ou gerenciamento do desenvolvimento de *software*. O autor utiliza-se do exemplo de Rutkowski, quando este autor diz que os consumidores estão demandando rapidez no mercado porque o tempo de um projeto afeta as operações nos negócios. Estes consumidores exigem execução sem falhas para concretizar oportunidades de negócios. Dessa maneira, uma gerência de projeto eficiente é a forma de fazer isso acontecer. Rutkowski destaca também que o gerente de projetos necessita de algumas habilidades para obter sucesso em suas atividades, entre elas destaca: a liderança, comunicação, resolução de conflitos, negociação, construção da equipe, habilidade de escutar e um bom gerenciamento de relacionamento.

> Gerência de projeto é a aplicação de conhecimentos, habilidades, ferramentas e técnicas no sentido de concluir atividades que atendam ou

> excedam às necessidades e expectativas dos *stakeholders* deste projeto. Gerência de projeto surgiu como um processo de gerenciamento para lidar com a complexidade do trabalho em grupo baseado em conhecimento e pelas demandas de novos métodos de gerenciamento. (ZANONI, 2009, p. 8).

Segundo Meredith e Mantel (*apud* ZANONI, 2009) as três forças básicas que vêm impulsionando a aplicação de gerência de projetos são: o crescimento exponencial do conhecimento humano; a demanda crescente por produtos e serviços mais complexos e padronizados; e a evolução da competição global pela produção de produtos e serviços.

Há diversos modelos de gerência de projetos desenvolvidos por estudiosos como Boehm (1989), Cantor (1998), Duncan (1996), Schwalbe (2000). O modelo de gerência de projetos orientado a objetos, desenvolvido por Cantor (1998) faz uso da *Unified Modeling Language* (UML) e do *Unified Process* (UP) nas fases do projeto, para que cada participante possa manter o processo de desenvolvimento e uma comunicação padrão no decorrer do projeto.

No modelo de gerência de projetos processual, desenvolvido por Schwalbe (*apud* ZANONI, 2009) é destacada a forma com que os processos são divididos, facilitando a comunicação entre as fases do projeto. O modelo incorpora os conceitos de gerência de projeto do *Project Management Institute* (PMI) em um modelo de gerência de projeto de *software*.

Conforme o modelo, os gerentes devem abordar conhecimento em nove áreas de conhecimento gerencial: integração, escopo, tempo, custo, qualidade, recursos humanos, comunicação, risco e aquisição do projeto.

Para Schwalbe (*apud* ZANONI, 2009), a gerência de projeto consiste basicamente em processos de inicialização, planejamento, execução, controle do projeto e encerramento do projeto.

6. QUALIDADE EM SOFTWARE

Um bom *software*, na opinião de Pfleeger (2004) é quando fica evi-

denciado a preocupação dos fabricantes em assegurar a qualidade dos produtos que produzem e, também quando os engenheiros buscam métodos para assegurar a qualidade e utilização dos mesmos. Dessa forma, bons engenheiros utilizam sempre uma estratégia para a produção de *software* de qualidade.

Qualidade é um substantivo que pode ter muitos significados. Isso acontece pela forte ligação com as percepções das pessoas, que tem pensamentos e gostos diferentes. Sob esse enfoque, a qualidade de *software* estaria diretamente ligada às percepções do ser humano, uma vez que se encontra ligada diretamente as opiniões das pessoas, que neste caso, são representadas pelos clientes, usuários e envolvidos com o projeto de *software*.

A qualidade de *software* possui as seguintes características:

- Qualidade de *software* está fortemente relacionada à conformidade com os requisitos;

- Ela caracteriza o grau de satisfação do cliente;

- Não é responsabilidade de apenas uma área da empresa, e sim de todos;

- Deve estar presente desde o planejamento do *software*.

Atualmente, qualidade de *software* vem ganhando um grande foco nas empresas de Tecnologia da Informação (TI), pois se percebe que a qualidade de *software* não é um gasto e sim um investimento. E com a evolução constante da tecnologia, os clientes estão cada vez mais exigentes.

A qualidade de *software* é uma área de conhecimento da engenharia de *software* que objetiva garantir a qualidade do *software* através da definição e normatização de processos de desenvolvimento. Apesar dos modelos aplicados na garantia da qualidade de *software* atuar principalmente no processo, o principal objetivo é garantir um produto final que satisfaça às expectativas do cliente, dentro daquilo que foi acordado inicialmente.

Segundo a norma ISO 9000 (versão 2000), a qualidade é o grau em que um conjunto de características inerentes a um produto, processo ou sistema cumpre os requisitos inicialmente estipulados para estes.

Pfleeger (2004) enfatiza que para medir a qualidade de um produto

com a de outro é necessário medir a qualidade do *software*. Primeiro, deve-se identificar os aspectos particulares do sistema que contribuem para a qualidade do produto como um todo. Normalmente, os usuários avaliam as características externas, tais como a qualidade e o tipo de falhas. Muitos engenheiros de *software* consideram que a qualidade do processo de desenvolvimento e manutenção é tão importante quanto à qualidade do produto.

7. GARANTIA DA QUALIDADE

De acordo com Paula Filho (2003), para se obter a garantia da qualidade, alguns procedimentos devem ser obedecidos, por exemplo, o procedimento de controle, gabaritos e padrões, além das ferramentas. Os procedimentos de controle são executados de maneira uniforme, diferenciando-se, apenas por ciclo de vida. Esse procedimento possibilita a aprovação de uma interação de um projeto para a fase seguinte.

Na revisão gerencial de fechamento da interação, o gerente do projeto determina conclusão ou não da interação, após ouvir a opinião da equipe, caso não seja aprovada a interação, o processo é novamente executado. Somente em casos positivos de interação é que o gerente faz o balanço e passa à fase seguinte. Quanto às revisões técnicas e inspeções, estas são o principal meio de garantia da qualidade quanto aos aspectos técnicos. "As revisões técnicas são aplicáveis nos *scripts* das interações, o planejamento delas pode ser alterado no Plano de Qualidade" (PAULA FILHO, 2003, p. 50).

As auditorias de qualidade, normalmente são realizadas por um grupo independente de Garantia de Qualidade que verifica a conformidade das atividades realizadas com os padrões determinados pelo processo. O grupo verifica, apenas, os relatórios desses procedimentos, portanto, não faz as revisões técnicas ou outros tipos de serviços.

As aprovações do cliente geralmente são necessárias para tomada de decisões para continuar ou não o projeto (fim da concepção e elaboração)

ou aceitação do produto (fim da construção e transição). Esses pontos de aceitação pelo cliente marcam, por definição, os finais das fases.

Os gabaritos têm a forma de modelos de documentos (*templates*) do *Microsoft Word* e do *Microsoft Excel*. No entanto, alguns gabaritos têm partes preenchidas, facilitando o trabalho. Essas partes precisam apenas ser alteradas, caso haja alteração no processo. Para muitos artefatos, são fornecidos exemplos. No caso das ferramentas, segundo Paula Filho (2003), por razões de custo e simplicidade de uso, optou-se por utilizar um processador de texto e uma ferramenta de planilha para a maioria dos artefatos.

Para uma utilização profissional do processo, é recomendável utilizar ferramentas mais sofisticadas para alguns artefatos, por exemplo, o Cadastro dos Requisitos é mais bem implementado usando-se uma ferramenta de bancos de dados (principalmente se for especializada para gestão de requisitos). Para a Memória de Planejamento, é usada uma planilha *Microsoft Excel*, mas uma ferramenta de gestão de projetos (por exemplo, o *Microsoft Project* ou equivalente) pode ser útil para estimativas mais precisas de cronograma. Algumas ferramentas específicas são necessárias, mesmo para projetos mais simples. Recomenda-se o uso de uma ferramenta de gestão de configurações.

7.1 Testes

De acordo com Paula Filho (2003), os testes são indicadores de qualidade do produto, considerados meios de detecção e correção de erros. Quanto maior o número de defeitos detectados em um *software*, provavelmente maior também o número de defeitos não-detectados. A constatação de um número elevado de defeitos em uma bateria de testes indica uma provável necessidade de redesenho dos itens testados.

Na leitura do autor há alguns tipos de baterias de testes que são:

- *Testes de aceitação* – objetiva validar o produto e verificar se ele atende aos requisitos especificados.

- *Testes de integração* – objetiva verificar as interfaces entre as partes de uma arquitetura de produto.

- *Testes de unidade* – objetiva verificar um elemento que possa ser tratado como uma unidade de implementação.

8. ESTUDO DE CASO

A empresa *Nippon Informática* é uma empresa de desenvolvimento de *software* com atuação nas áreas Contábil, Fiscal e Recursos Humanos, além de grande especialização e conhecimento em desenvolvimento de *software* para tratamento destas questões. É Empresa com forte relacionamento em empresas de contabilidade, nas quais encontrou abertura para os clientes dessas empresas, aumentando o portfólio de atuação de seu *software* em área comercial e industrial.

A empresa possui três funcionários, os quais atuam na área de desenvolvimento, suporte e manutenção do *software*. Avaliando-se o cenário atual, nota-se na empresa o acúmulo de funções dos funcionários, não havendo designação de tarefas. Com isto, foi possível observar que, as atividades e demandas se acumulam e não há priorização de atendimento aos clientes.

8.1 Mapeamento do processo de softwareA empresa adota poucos processos durante a construção do *software.* Para iniciar o entendimento das necessidades do cliente é feito um levantamento e análise das informações coletadas. Este levantamento é feito avaliando o negócio que o cliente possui e o que o sistema da *Nippon* oferece. O funcionamento das rotinas do *software* da *Nippon* como: cálculo de folha de pagamento, geração de guias para pagamento de impostos, é submetida à legislação fiscal e trabalhista. Desta forma, a *Nippon* ao levantar as necessidades do cliente, tenta ao máximo, adaptar o *software* e as rotinas que possui em outros clientes e que já está funcionando plenamente para evitar muitas alterações.

O *software* que o cliente irá utilizar é definido mediante a análise feita e indicação das devidas adaptações nas rotinas disponibilizadas. Em alguns clientes é necessário apenas alterar alguns *layouts* de telas e campos de armazenamento de informações.

Na maioria dos casos, os erros de funcionamento no *software* adquirido pelo cliente ficam praticamente nulos, pois apenas se utiliza alguma rotina que já está correta de um cliente para o outro. A empresa não utiliza até o momento, nenhuma ferramenta para registro das informações tratadas com os clientes e também não gera nenhum tipo de documentação. Na etapa de desenvolvimento do *software*, a empresa usava, no início de suas atividades de programação a linguagem *Quick Basic (versão 4.5)*, pois o sistema operacional era DOS. Posteriormente, com a necessidade dos clientes em utilizar o ambiente *Windows*, houve a necessidade de migração de todo o sistema para a linguagem *Visual Basic*.

8.2 Cenário de solução - fases e etapas propostas

O cenário de solução foi elaborado após um estudo do fluxo de processo que acontece no ciclo de vida de desenvolvimento do *software* da *Nippon*, avaliando os processos de engenharia de *software* e as metodologias de desenvolvimento.

Com este estudo, foi possível perceber os pontos falhos dos processos de desenvolvimento de *software* da empresa *Nippon*, uma vez que possibilitou também visualizar o que precisa ser melhorado para que o desenvolvimento seja realizado com sucesso e que o produto seja entregue ao cliente no prazo determinado, dentro do custo estimado e com melhor qualidade. No cenário estudado, foi detectada a ausência de um ciclo de vida de desenvolvimento do *software*. Para isto, propõe-se adotar o ciclo de vida composto das fases: Definição, Desenvolvimento, Manutenção e das etapas propostas na Figura 1 a seguir:

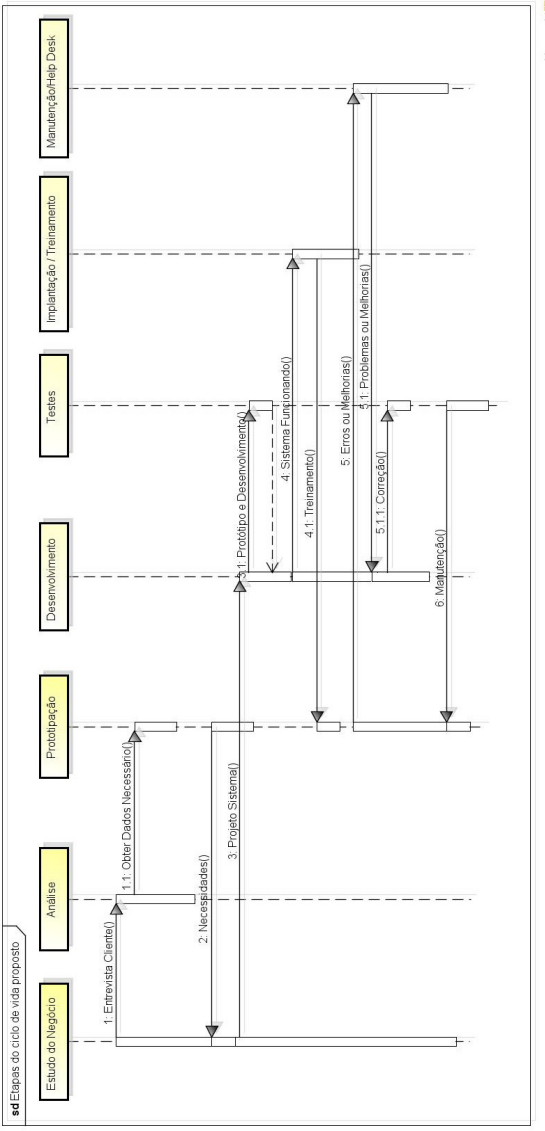

Figura 1: Etapas do Ciclo de Desenvolvimento - cenário de solução.
Fonte: Arquivo pessoal.

Na fase de "Definição", propomos as etapas de Levantamento de Requisitos e Análise de Requisitos.

No Levantamento de Requisitos sugerimos que seja realizado um estudo do negócio do cliente que será feita a proposta do *Software*, pois é de suma importância saber que atividades e objetivos a empresa possui no mercado. Após este estudo, temos base para que seja realizada uma entrevista com o cliente.

A entrevista deve ser realizada por meio de uma visita ao cliente. Nesta pode-se se aplicar um questionário para facilitar a compreensão das informações.

Na etapa de levantamento de requisitos, sugere-se também que a empresa *Nippon* apresente inicialmente uma proposta ao cliente e faça a documentação de todas as informações tratadas e de todos os dados que serão necessários para construir o sistema, com base na apresentação realizada e discussão gerada.

Posteriormente, caso o cliente aceite a proposta apresentada, deve ser realizada um detalhamento das necessidades do cliente.

O levantamento das necessidades do cliente é uma etapa primordial para o desenvolvimento do *software*, pois através desta é que se vai definir como vai ser a realização ou modificações necessárias ao *software*. Conforme a literatura mencionada anteriormente, esta etapa utiliza o levantamento de requisitos.

Em seguida a esta etapa, inicia-se o processo de adaptação das rotinas já existentes para uso do contratante do serviço. Neste processo, foi verificado que se a *Nippon* gerasse uma documentação de cada *software*, ficaria mais fácil e rápida a sua elaboração.

Após a entrevista, são coletadas as necessidades do cliente e realizada a Análise dos Requisitos que deve ser bastante criteriosa. Assim sendo, o analista poderá acatar as sugestões do cliente ou mesmo emitir opiniões sobre o desenvolvimento do novo *software*. Nesta etapa de análise, o analista deve proceder todo o entendimento dos requisitos do *software* procurando mapear todas as características funcionais do *software* que será construído ou alterado.

Como resultado desta etapa, propõe-se que a empresa *Nippon* gere um documento de Especificação dos requisitos do *software*. Este documento pode ser o Diagrama de Caso de Usos (Anexo um *template* deste documento).

Estudado e definido o escopo do projeto com o cliente, o fluxo do processo tem como próxima etapa, a Fase de Desenvolvimento.

Na Fase de Desenvolvimento teremos as etapas de Desenvolvimento e Testes.

Já com o escopo definido e definida linguagem de programação a ser adotada é iniciada a Codificação do Projeto. Nesta etapa, é importante que a empresa instrua os desenvolvedores quanto à padronização da codificação e a documentação do *software*.

Sugerimos também o uso de um sistema de Controle de Versões, no ambiente de desenvolvimento, já que foi verificado que a empresa não usa nenhum tipo de aplicativo nesse sentido para se controlar os produtos de trabalho que são gerados ao longo do processo de desenvolvimento do *software*. Uma sugestão seria o *Microsoft Visual Source Safe*, que é versão de sistema em nível de arquivo que permite que vários tipos de organizações possam trabalhar em várias versões do projeto ao mesmo tempo. Essa funcionalidade é bastante benéfica no ambiente de desenvolvimento, no qual é usado ao manter versões de código paralelas.

Após, codificado o sistema ou parte dele, passa-se para a etapa de Testes, onde são realizados vários tipos de testes com a intenção de encontrar erros. Estes são realizados com foco na prevenção e sintomas dos erros, fornecendo diagnósticos para que sejam corrigidos facilmente. Os erros também podem ocorrer depois dos testes, pois o sistema pode se comportar de forma diferente em outras máquinas, pois devem ser realizados testes em vários tipos de configurações.

Após a realização de todos os testes, deve ser iniciada a fase de Manutenção, a qual sugerimos que tenha as etapas: Implantação, Treinamento e Manutenção do *Software*. Nessa etapa é realizada a instalação física do *software* e todo o treinamento para os usuários, ressaltando que neste momento, é importante que se tenha a equipe de suporte, analista e banco de

dados bem integrados. Todos os usuários, após estarem treinados e aptos a utilizarem o Sistema, têm-se, então, a realização final do *Software*.

Inicia-se a última etapa, Manutenção. Nesta etapa procura-se apoiar o cliente em tudo que o *software* necessitar, após sua entrega. Nesta, podem surgir correções de erros, customizações, desenvolvimento de novos requisitos, entre outros. Este se faz importante, pois é onde se mantém o compromisso estabelecido entre o cliente e a evolução do *software*.

A empresa também deve contar com um setor de *Help Desk* eficiente para esclarecer e solucionar possíveis dúvidas ou problemas que venham ocorrer no sistema.

9. CONSIDERAÇÕES FINAIS

Com este estudo realizado, concluímos que, cada vez mais empresas de desenvolvimento de sistemas necessitam adotar processos de *software* adequados. Isto é importante, pois construir um *software* com qualidade demanda a utilização e implantação de processos e técnicas de engenharia de *software*. Isto é indispensável para que o produto seja entregue ao cliente dentro do prazo e orçamento planejado, alcançando a qualidade esperada.

A definição do ciclo de vida de um *software* é importante para se ter a visão completa do desenvolvimento do *software*. Com isto, é possível definir etapas que abrangem desde a análise dos requisitos até a entrega do *software* para o cliente.

REFERÊNCIAS

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS - ABNT. *ISO* 9000: caminho para a qualidade total? **Revista de Administração.** São Paulo, USP, v. 29, n. 4, out./dez. 1994.

CARVALHO, Ariadne Maria Brito Rizzoni; CHIOSSI, Thelma Cecília dos Santos. **A introdução a engenharia de software.** Campinas: Unicamp, 2001.

PAULA FILHO, Wilson de Pádua. **Engenharia de software:** fundamentos, métodos e padrões. 2. ed. Rio de Janeiro: LTC, 2003.

PRESSMAN, Roger S. **Engenharia de software.** Tradução de José Carlos Barbosa dos Santos. 3. ed. São Paulo: Makron Books, 1995.

PFLEEGER, Shari Lawrence. **Engenharia de software:** teoria e prática. [Traduzido do original: Software engineering - theory an practice]. Tradução de Dino Franklin. 2. ed. São Paulo: Prentice Hall, 2004.

VIEIRA, Andreza. **Projetos de software.** 2008. Disponível em: <http://pt.shvoong.com/internet-and-technologies/software/1836632-projeto-software/>. Acesso em: 31 mar. 2010.

ZANONI, Roberto. **Gerência de projeto de software em ambiente fisicamente distribuído:** um estudo de caso. 2009. Disponível em: <http://www.inf.pucrs.br/~jaudy/Artigopdf>. Acesso em: 30 mar. 2010.

Title: Software lifecycle study in a software development company.
Authors: Fabrício Luis Marinheiro Lourenço; Márcio Alan Benine.

ABSTRACT: A study of the life cycle of software developed in an enterprise system development is important, because building a quality software requires the use and implementation of processes and techniques of software engineering. This process is essential if the product is delivered to the customer on time and within planned budget, achieving the expected quality. Given this procedure, we carried out a study using a literature search in order to find theoretical framework to give support to the proposal and issue a case study in Nippon Computer Ltda. This company located in the city of Batatais/SP, works with software development for areas Accounting, Tax and Human Resources. In the case study sought to establish the mapping of existing processes and after their analysis, propose a solution scenario reality of the company. As a final result of this study, we presented a proposal for implementation of the model of the software life cycle, allowing the construction of quality software, providing a model that meets the standards of software engineering and has important aspects of quality. It was concluded that, increasingly, companies need to adopt systems development processes appropriate software. The definition of the lifecycle of a software is important to have a complete view of software development. Therefore, it was possible to define steps ranging from requirements analysis to delivery of software to the client.
Keywords: Software engineering. Software processes. Requirements. Life cycle. Quality.