GESTÃO DE QUALIDADE NO PROCESSO DE DESENVOLVIMENTO DE SOFTWARE

BIANCA PINTO DE AMORIM, DANIELE RIBEIRO SANTOS, DIEGO ANTUNES MUNIZ, JANAINA PEDRINA DE MORAES COSTA, LEANDRO ALAOR ANTUNES, ELIANE CRISTINA AMARAL, ELINEY SABINO, NARUMI ABE

Resumo

A engenharia de software tem por finalidade desenvolver softwares sistemáticos e econômicos, resultando produtos confiáveis e eficientes, a estimação é realizada com base numa abstração em forma de requisitos. Por definição, ela não pode ser exata ao considerar as características apresentadas acima, pois a estimativa é, na verdade, baseada numa ideia pré-concebida do software. Mesmo medidas do produto final de software são subjetivas. Não existem métricas objetivas de tamanho, qualidade, eficiência, robustez, usabilidade e tantos outros aspectos. Não é tarefa fácil conciliar ambas as perspectivas, sendo um desafio alinhar os objetivos estratégicos de uma organização com os aspectos técnicos e abstratos de um produto de software.

Do ponto de vista de quem está gerenciando o projeto, prazo e cronograma são fundamentais para que decisões sejam tomadas e estratégias de negócio estabelecidas, portanto boas estimativas são necessárias. Isso pode ser um problema do ponto de vista da equipe de desenvolvimento, por exemplo, quando prazos arbitrários são definidos ou tempo e recursos disponíveis são tecnicamente insuficientes.

Palavras chaves: Engenharia de Software, Gestão de Projetos, Software.

Abstract

Software engineering aims to develop systematic and economical software, resulting in reliable and efficient products, estimation is performed based on an abstraction in the form of requirements. By definition, it cannot be accurate in considering the characteristics presented above, since the estimate is actually based on a preconceived idea of the software. Even measures of the final software product are subjective. There are no objective metrics of size, quality, efficiency, robustness, usability and many other aspects. It is not an easy task to reconcile both perspectives, and it is a challenge to align an organization's strategic objectives with the technical and abstract aspects of a software product.

From the point of view of who is managing the project, timing and timing are crucial for decisions to be taken and established business strategies, so good estimates are needed. This can be a problem from the development team's point of view, for example, when arbitrary deadlines are set or time and available resources are technically insufficient.

Keywords: Software Engineering, Project Management, Software.

Introdução

Um aplicativo de software é desenvolvido através de um processo. Não é algo que se fabrica a partir de matéria prima, nem é montado a partir de partes menores.

Segundo Pressman (2006, p. 4), o software apresenta esta característica especial em relação a outros tipos de produtos, ou seja: ele não é fabricado no sentido clássico, mas desenvolvido, passando por um processo de engenharia.

A Engenharia de Software fornece abordagens sólidas para aumentar as chances de que os objetivos de negócio sejam atingidos em termos de prazo, qualidade e funcionalidades

> ...as organizações enfrentam hoje o desafio de desempenhar suas atividades de forma produtiva, com qualidade e cumprindo o planejamento estratégico. Portanto, o uso de uma abordagem adequada no desenvolvimento de um software para elicitação de requisitos, estimação, desenvolvimento e controle é fundamental para as organizações. Campos (2009, p. 2),

Segundo Pressman (1995) a engenharia de software surgiu em um cenário aonde o os softwares se encontravam em crise, com isso a engenharia foi desenvolvida para solucionar os problemas que acompanhavam a sociedade de desenvolvedores ao longo dos anos. A partir disto, surgiu a necessidade de tornar o desenvolvimento de software em um processo organizado, planejado e padronizado para que todas as necessidades fossem atendidas e os gastos com informatização de processos de informações se tornassem compensadores.

Entretanto para Pressman (2002), a engenharia de software tem por finalidade desenvolver softwares sistemáticos e econômicos, resultando produtos confiáveis e eficientes.

Na engenharia de software, para Maldonado (2001), é uma estratégica para o sucesso dos projetos de software. No entanto, ainda há dificuldades em produzir softwares que atendam as expectativas dos usuários e que sejam entregues dentro do prazo. Conseguir qualidade nos processos de software não é uma tarefa fácil, o desconhecimento de ferramentas e a força aos processos internos dentro de uma organização, são razões que dificultam atingir os objetivos de qualidade.

ABORDAGEM TEÓRICA SOBRE MÉTRICAS DE SOFTWARE

A falta de padrões de desenvolvimento de software tornou necessária a criação da engenharia de software e consequentemente de seus processos de desenvolvimento e medição de tamanho, custo e tempo de construção. Para tal diversas metodologias foram desenvolvidas ao longo dos anos, cada uma buscando uma abordagem diferente para a resolução do problema de medição de software. Revista Pensar Tecnologia, v.4, n.2, jul. 2015 Para Sommervile (2003) a engenharia de software é uma disciplina que se ocupa de todos os aspectos da produção de software, desde os estágios iniciais de especificação do sistema até a manutenção do sistema, depois que ele entrou em operação.

> ... assinala que um processo de software é: Um conjunto de atividades, ligadas por padrões de relacionamento entre ela, pelas quais se as atividades operarem corretamente e de acordo com os padrões requeridos, o resultado desejado é produzido. Jalote (2005, 566)

Segundo Pressman (2010), resultado desejado é um software de alta qualidade e baixo custo. Obviamente, um processo que não aumenta a produção (não suporta projetos de software grandes) ou não pode produzir software com boa qualidade não é um processo adequado. As métricas de software proporcionam uma maneira quantitativa de avaliar a qualidade dos atributos internos do produto, possibilitando avaliar a qualidade antes que ele seja criado. As métricas proporcionam as informações necessárias para criar requisitos eficazes e modelos de projeto, código sólido e testes completos. As métricas de software permitem avaliar o projeto a ser desenvolvido de uma forma mais completa, por isso tal procedimento é tão importante no cenário atual do desenvolvimento.

> ... medidas quantitativas que permitem que você tenha discernimento sobre a eficácia do processo de software e os projetos que são executados usando o processo como estrutura. Pressman (2010, 583)

Conforme Pressman (2010), as medições foram criadas principalmente para garantir que indicativos pudessem ser obtidos e assim houvesse uma otimização dos custos de produção já que na década de 90, bilhões de dólares eram gastos em softwares que não atendiam as empresas da época. As métricas de software podem ser divididas em dois tipos, as medidas diretas e as medidas indiretas. As medidas diretas incluem fatores como o custo de produção, o trabalho aplicado para a produção do produto de software, tamanho do software em linhas de código e outras mais, este tipo de medida pode

ser estimado antes mesmo da produção do produto de software enquanto as medidas indiretas baseiam em fatores como funcionalidades, qualidade e complexidade e estas características não podem ser medidas de uma forma simples por isso são consideradas medidas indiretas. Quando um software é desenvolvido, é certo que um produto perfeito é impossível de ser obtido a partir do desenvolvimento primário e por isso cada vez mais se faz necessária medição de diversos fatores a fim de se alcançar um software com alto nível de qualidade.

Segundo (PRESSMAN, 2010) a medição de software torna possível que gerentes e profissionais entendam melhor o processo de engenharia de software e o produto (software) que ele produz. Usando medidas diretas e indiretas, as métricas de produtividade e qualidade podem ser definidas as Métricas orientadas a tamanho As métricas orientadas a tamanho levam em consideração a quantidade de linhas de código escritas no decorrer do desenvolvimento do software e podem ser representadas até mesmo e assim ao armazenar dados de um projeto desenvolvido pode se ter base para aplicar o mesmo em futuros projetos.

A atividade de estimação consiste em tentar antecipar o tamanho ou esforço de desenvolvimento de um produto abstrato. Existe uma série de desafios que fazem com que o software seja difícil de estimar e, consequentemente, o projeto difícil de planejar.

Por um lado, isso se deve a características intrínsecas ao software. Brooks (1987) afirma que, por definição, o software é complexo e irredutível, ou seja, não é possível simplificá-lo sem que haja perda de informação. Ele apresenta quatro características importantes que contribuem para isso, tornando o software essencialmente difícil de construir:

- **Complexidade**: entidades de software são extremamente complexas para seu tamanho e não existem duas partes iguais no nível de algoritmo. Isso faz parte da natureza do software, não ocorrendo por acaso. A escalabilidade de um software, por exemplo, não consiste simplesmente em fazer suas partes maiores ou menores, diferentemente de construções físicas realizadas pelo homem.
- **Conformidade**: apesar de existirem áreas onde pessoas lidam com alta complexidade, o software possui algumas complicações adicionais. Por exemplo, um físico lida com a complexidade do átomo, mas este possui uma interface “bem definida” pela natureza, ou seja, os átomos possuem uma conformidade. O software tem diversas interfaces diferentes definidas por diversas pessoas e geralmente há pouca ou nenhuma conformidade nestas definições.
- **Mutabilidade**: o software parece estar sempre pressionado a mudar. Produtos fabricados como pontes, carros e máquinas também estão, mas geralmente eles são substituídos por novos modelos, não sendo modificados com a mesma facilidade de um software. Eles são produtos

tangíveis, com escopo bem definido e o custo de modificações maiores é impeditivo. Um software, por ser abstrato e com baixos custos diretos para se mexer, muitas vezes sem aparentes consequências imediatas, sofre constantes mudanças. Além disso, a maioria das mudanças tem como objetivo fazer com que o software vá além de seus limites iniciais.

- **Invisibilidade ou intangibilidade**: apesar de existirem interfaces e linguagens amigáveis, o software não pode ser visualizado através de imagens. Em geral, desenvolvedor e usuário não conseguem enxergar o software como um todo. Isso ocorre porque os usuários geralmente não compreendem as questões tecnológicas e os desenvolvedores não entendem todas as regras do negócio, além do que um software pode ser desenvolvido por diversos indivíduos especialistas em diferentes áreas. Portanto, como cliente e desenvolvedor não conseguem ter em mente o software por completo, as chances de erros aumentam.

A estimação é realizada com base numa abstração em forma de requisitos. Por definição, ela não pode ser exata ao considerar as características apresentadas acima, pois a estimativa é, na verdade, baseada numa ideia pré-concebida do software.

Mesmo medidas do produto final de software são subjetivas. Não existem métricas objetivas de tamanho, qualidade, eficiência, robustez, usabilidade e tantos outros aspectos. Mesmo o número de linhas de código ou caracteres do código fonte não possuem uma relação direta com as funcionalidades e características do software, embora muito esforço tenha sido empreendido em conseguir uma aproximação razoável. Isso implica em que qualquer comparação entre softwares, processos de desenvolvimento, técnicas estimação e de engenharia de requisitos somente podem ser realizadas através de critérios específicos e objetivos. Portanto, não é possível afirmar categoricamente que uma técnica, método ou modelo seja superior a qualquer outro.

Por outro lado, o fator humano acrescenta ainda mais dificuldades no desenvolvimento e na estimação de um software. Enquanto desenvolvedores e arquitetos pensam em nível técnico e funcional, os gerentes observam o cronograma e os custos. Não é tarefa fácil conciliar ambas as perspectivas, sendo um desafio alinhar os objetivos estratégicos de uma organização com os aspectos técnicos e abstratos de um produto de software.

Do ponto de vista de quem está gerenciando o projeto, prazo e cronograma são fundamentais para que decisões sejam tomadas e estratégias de negócio estabelecidas, portanto boas estimativas são necessárias. Isso pode ser um problema do ponto de vista da equipe de desenvolvimento, por exemplo, quando prazos arbitrários são definidos ou tempo e recursos disponíveis são tecnicamente insuficientes.

Portanto, tanto a natureza do software quanto os fatores humanos devem ser considerados na estimação. Da mesma forma, quando há alterações no contexto do projeto de software, tal como mudança nos requisitos, no domínio ou na equipe, as estimativas são afetadas e devem ser ajustadas.

Apesar de se mostrar um modelo muito interessante para a medição de software as métricas orientadas a tamanho não são aceitas universalmente e um dos principais motivos se deve ao fato de haver diferença entre a quantidade de linhas de código necessárias e diferentes tipos de linguagens de programação (PRESSMAN, 2010) por isso outros modelos ganharam espaço no mercado utilizando de outras abordagens.

Métricas orientadas a função. As métricas orientadas a função para medir a quantidade de funções de um software são estimadas através de dados históricos, os custos e o esforço para o projeto codificação, teste de novos produtos de software. Pontos de função são derivados, por meio de uma relação empírica baseada em medidas calculáveis (diretas) do domínio de informações do software e avaliações qualitativas de complexidade do software (PRESSMAN, 2010).

Segundo Weber (2001) qualidade de software é um conjunto de propriedades que um determinado produto deve alcançar para atender as necessidades do cliente.

Conforme Maldonado (2001), para se desenvolver um software é preciso atender as necessidades do cliente para o uso especificado, identificar as características de qualidade necessária para satisfazer o cliente as características de um software são:

- **Funcionalidade:** Trata-se das ações que o sistema pode exercer.
- **Confiabilidade:** É definida como a possibilidade do sistema funcionar sem ocorrências de falhas em um período ou ambiente especificado.
- **Usabilidade:** É definido no campo da interação Humano-computador, uma característica fácil de usar e de aprender.
- **Eficiência:** Trata-se do nível do desempenho do software e a quantidade de recursos utilizados.
- **Manutenibilidade:** Processo de modificação de um produto de software
- Portabilidade: Trata- se da capacidade do software ser transferido de um ambiente para outro
- **Efetividade :** Trata-se da Capacidade do software em possibilitar aos usuários atingir metas especificadas.
- **Produtividade:** Trata-se da quantidade adequada de recursos em que o usuário pode utilizar

- **Segurança:** Refere-se ao conjunto de necessidades de segurança que o software deve atender para não prejudicar pessoas, negócios ou propriedades especificas.
- **Satisfação:** Trata- se da capacidade em satisfazer o cliente.

É fundamental que um software seja confiável, eficaz e siga os padrões exigidos.

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 Linguagem Acadêmica, Batatais, v. 1, n. 2, p. 183-201, jul./dez. 2011 187 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 protó- tipo 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).

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 188 Linguagem Acadêmica, Batatais, v. 1, n. 2, p. 183-201, jul./dez. 2011 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.

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

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).

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, 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.

Na opinião de Pfleeger (2004), 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. 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.

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

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 Bibliográficas

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.

BOEHM, Barry W. Software Engineering Economics. Prentice Hall. 1981

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

IFPUG. Manual de Praticas de Contagem de Pontos de Função. Versão 4.3, 2010.

JALOTE, P. An Integrated Approach to Software Engineering. 3.ed. New York: Springer, 2005, 566p.

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

PRESSMAN, R S. Engenharia de Software. 5ª edição. Rio de Janeiro: McGraw-Hill, 2002.

MALDONADO, Qualidade de Software , Teoria e prática. São Paulo: Pearson, 2001.

PRESSMAN, Roger S. Engenharia de software: uma abordagem profissional. 7. ed. NY: AMGH ,2011.

SOMMERVILLE, I. Engenharia de Software. 6ª ed. São Paulo: Addison Wesley, 2003, 592p.

VASQUEZ, Carlos Eduardo; SIMÕES, Guilherme Siqueira; ALBERT, Renato Machado; Análise de ponto de função: Medição, Estimativas e Gerenciamento de Projetos de Software. 13. ed. /SP: Érica, 2013.