3
Qualidade de Software

Este capítulo tem como objetivo esclarecer conceitos relacionados à qualidade de software; conceitos estes muito importantes para o entendimento do presente trabalho, cujo objetivo principal é o aumento da qualidade do software estudado a partir de melhorias nos processos internos de desenvolvimento.

3.1. Qualidade

A área de Qualidade na Engenharia de Software é similar a esta mesma área em outras engenharias. Porém, há algumas características que distinguem os produtos de software dos demais produtos industriais. Os processos de desenvolvimento de um produto da engenharia e de um software podem apresentar muitas semelhanças em relação ao seu ciclo: definição do produto que se deseja construir ou desenvolver, análise dos requisitos, definição da arquitetura macro e construção do projeto detalhado, fabricação ou desenvolvimento e entrega. Entretanto, a maneira como um software é produzido se difere, pois este é intangível e está mais sujeito ao erro humano.

Assim, se dois softwares forem produzidos utilizando-se o mesmo processo de desenvolvimento, provavelmente terão algumas características distintas. Por outro lado, é comum a produção de vários produtos industriais exatamente iguais, seguindo um mesmo processo de construção. Além disto, a indústria de software é relativamente recente quando comparada às outras existentes.

Com a grande utilização de software nas mais diversas áreas, a procura pela qualidade do sistema contratado tem sido uma preocupação crescente. A complexidade dos sistemas desenvolvidos também tem aumentado, bem como a concorrência entre as empresas fornecedoras. Assim, é muito comum haver reclamações em relação aos produtos de software gerados, apesar de estes serem imprescindíveis na execução de diversas tarefas. Ou seja, apesar de todas as reclamações existentes em relação aos softwares adquiridos, eles são bastante utilizados, possivelmente por falta de alternativa. Assim, a indústria de software

vem ocupando um importante espaço na economia mundial e é importante que tenha um foco no desenvolvimento de produtos de qualidade.

Neste contexto, qualidade de um artefato está relacionada ao grau de satisfação dos clientes em relação a um determinado produto. A Norma ISO9126 [12] define qualidade como "a totalidade de características de uma entidade que lhe confere a capacidade de satisfazer às necessidades explícitas e implícitas". Para Pressman [22], a qualidade de software é definida como "conformidade com requisitos funcionais e de desempenho explicitamente declarados, normas de desenvolvimento explicitamente documentadas e características implícitas, que são esperadas em todo software desenvolvido profissionalmente".

Em sua definição, ele relaciona qualidade com conformidade em relação aos requisitos funcionais e não-funcionais, presentes nos documentos de especificação do sistema. Porém, ele não menciona a possibilidade de as especificações estarem incorretas – o que não atenderia ao cliente. Portanto, para um software ser considerado de qualidade satisfatória, é importante que ele atenda a requisitos funcionais e não-funcionais, além de possuir uma análise de requisitos coerente com o desejo do cliente.

O autor também deixa de dizer que entre os requisitos implícitos estão aqueles que o usuário deseja e que se esqueceu de mencionar, possivelmente por achá-los óbvios considerando o domínio do problema. Mas isso é diferente de "desenvolvido profissionalmente", pois aquele aspecto está no domínio da solução.

A partir desta definição de Pressman, podem-se enfatizar três pontos importantes: requisitos funcionais e de desempenho, normas de desenvolvimento e requisitos implícitos. Os requisitos funcionais e de desempenho explicitamente declarados de um sistema são aqueles que representam as necessidades explícitas do usuário. Estão descritos no documento de requisitos que deve ser elaborado nas primeiras fases de desenvolvimento do sistema e expressam o que se espera do software, suas funções imprescindíveis e condições de utilização. Os requisitos de um software são a base para a avaliação da qualidade do sistema desenvolvido. Mesmo que o produto final não apresente erros do ponto de vista da codificação em si, se o software não tem uma boa relação com o efetivamente esperado pelo usuário, ele apresenta baixa qualidade.

As normas de desenvolvimento explicitamente documentadas auxiliam o processo de desenvolvimento de software, uma vez que reúnem, a partir de fontes confiáveis e de um conjunto de experiências anteriores, técnicas e métodos que têm por objetivo a garantia de produção, por construção, de software com qualidade.

Por último, as características implícitas de um software refletem os seus requisitos implícitos. Estes podem não ser citados pelo usuário como um atributo que o software deve possuir, porém, espera-se que o software possua, como, por exemplo, facilidade de uso e boa manutenibilidade [22].

Segundo Staa [29], "A qualidade de um artefato é um conjunto de propriedades a serem satisfeitas em determinado grau, de modo que satisfaçam as necessidades explícitas e implícitas de seus usuários e clientes".

O problema está no implícito: não necessariamente é conhecido pelos desenvolvedores. Estas são em geral as causas das alterações de requisitos no decorrer do desenvolvimento. O foco desta definição é o cliente e o usuário. Especificações, normas, proficiência e outros documentos são acessórios para se atingir o objetivo.

O processo de desenvolvimento é um processo de aprendizado. Durante o desenvolvimento, aprendem-se coisas novas a respeito do domínio da aplicação, e das necessidades e expectativas do usuário [29]. Também se aprendem coisas novas com relação à tecnologia utilizada (ex. arquitetura, projeto, plataforma). Mesmo nas engenharias tradicionais ocorrem muitas mudanças durante a fase de projeto. Por outro lado, quando se parte para a construção, a maior parte das causas para mudanças já está entendida e absorvida nos planos de construção.

Software não possui construção. Mesmo a programação é um período em que se tomam decisões e fazem-se balanços entre o desejável, o possível e o conhecido. Ou seja, mesmo programação é um período de micro-planejamento.

Além disto, o usuário pode aprender coisas a respeito do domínio do problema e de sua representação computacional no decorrer do desenvolvimento, ou ao final dele, o que leva a mudanças nos requisitos. Isso não se deve a uma suposta volubilidade do usuário, mas um desconhecimento ou incapacidade de imaginar o potencial de uma solução a partir de uma conjectura quando da redação da especificação.

3.2.Garantia de Qualidade

A garantia de qualidade é "o conjunto de atividades sistemáticas a serem realizadas visando assegurar a adequação ao uso do artefato como um todo" [1].

Visto sob esta ótica, a garantia da qualidade de software compreende uma série de ações que têm por objetivo medir o grau de qualidade encontrado no produto avaliado e assegurar que atenda às especificações explícitas e implícitas.

Ela depende de alguns fatores principais, como o processo de desenvolvimento utilizado, as pessoas participantes do projeto e a elaboração de atividades específicas voltadas para a garantia da qualidade.

A fim de alcançar uma boa qualidade do produto, os processos de desenvolvimento devem estar claramente definidos e documentados e precisam ser rigorosamente seguidos pelas pessoas que participarão do seu desenvolvimento [7]. Desta maneira, a gerência de uma organização deve possuir um alto comprometimento com as técnicas de garantia de qualidade e exigir que todos os membros da organização também o tenham.

A garantia de qualidade está intimamente ligada ao conceito de software correto por desenvolvimento. Isto é, devem-se desenvolver artefatos que compõem o software com qualidade já antes do primeiro controle da qualidade ou uso em etapas subseqüentes do desenvolvimento ou em produção. Assim, aplicam-se métodos e medidas técnicas sólidas, conduzem-se revisões técnicas bem definidas e se aplicam testes bem elaborados; tudo isso com vistas a evitar a introdução de defeitos durante o desenvolvimento e aumentar a probabilidade de o software ter uma qualidade satisfatória já antes do primeiro controle da qualidade [29].

A fim de assegurar a qualidade por construção de produtos de software, é importante que pessoas proficientes sejam empregadas no desenvolvimento do mesmo. Processos, métodos e ferramentas requerem conhecimento prévio para serem bem utilizados.

Os processos e métodos, para auxiliarem o desenvolvimento, devem ser efetivamente seguidos ao longo do ciclo de vida do software e não apenas mencionados nos planos de projeto [20]. As ferramentas, por sua vez, não são capazes de fazer o trabalho sozinhas, elas apenas auxiliam os desenvolvedores na

medida em que, se forem bem empregadas, reduzem o tempo que seria gasto com atividades mecânicas [7].

O grupo responsável pela garantia da qualidade deve ser formado por uma equipe externa ao projeto – para que haja a imparcialidade necessária. Os principais fatores que influenciam a garantia da qualidade do software são o tempo disponível para o desenvolvimento do software, bem como seu orçamento e sua complexidade, somado à experiência prévia da organização em áreas correlatas [21].

Porém, a garantia de qualidade, além de atuar sobre o processo de desenvolvimento, requer a revisão e a auditoria da qualidade dos artefatos produzidos e dos processos de desenvolvimento para verificar se estão de acordo com os padrões e requisitos estabelecidos. É neste contexto que surge o conceito de controle de qualidade.

3.3.Controle de Qualidade

O controle de qualidade envolve uma série de atividades que têm como objetivo avaliar o produto gerado e a partir desta avaliação, determinar se o mesmo está pronto para ser comercializado. O controle de qualidade tem por objetivo a detecção e a promoção da correção de defeitos, enquanto que a garantia de qualidade tem como objetivo a prevenção dos mesmos.

O controle de qualidade deve ser aplicado ao longo do processo de desenvolvimento do software e pode ser de responsabilidade dos próprios desenvolvedores, ou, idealmente, de um departamento independente da organização. Ele se baseia no plano de garantia de qualidade e inclui atividades de verificação, validação e aprovação. Em se tratando do método ágil objeto de estudo deste trabalho, o *Scrum*, os próprios times costumam conter especialistas em controle de qualidade. Isto é, a idéia de possuir um departamento independente é deixada de lado.

Para que este controle possa ser montado, os requisitos funcionais e não funcionais do software devem estar claramente estabelecidos. Também é necessário saber qual o grau de qualidade esperado de determinadas características presentes nos artefatos. Portanto, um importante fator do controle de qualidade é que os artefatos produzidos tenham especificações definidas e

mensuráveis, com as quais seja possível fazer a comparação do resultado de cada processo [22].

As atividades de verificação avaliam os artefatos individualmente, verificando se os padrões estão sendo corretamente seguidos, a fim de diminuir o número de possíveis erros remanescentes e facilitar o entendimento do código por parte da equipe que irá mantê-lo.

Elas também devem assegurar que o artefato reflita exatamente aquilo que o redator queria, ou seja, a pessoa que for ler o documento deve ter o entendimento desejado pelo redator. Por último, a verificação deve avaliar se os artefatos apresentam o nível de abstração adequado, ou seja, deve ter objetivos bem estabelecidos e a sua implementação não deve ir além dos objetivos descritos, como também não deve deixar de implementá-los.

A fase de validação é onde se verifica se os artefatos foram produzidos conforme suas especificações. Também é neste momento que se estuda o conjunto de artefatos que possuem alguma ligação, a fim de se observar se suas interfaces estão corretamente implementadas, para garantir que os artefatos envolvidos tenham algum sentido em conjunto.

Por último, são realizadas as atividades de aprovação [29]. Após as fases anteriores, tem-se a certeza de que os artefatos foram produzidos da maneira que se esperava, ou seja, atendendo aos requisitos impostos. Porém, se houver alguma falha na especificação desses requisitos, a verificação e validação não teriam como descobrir tal problema.

É neste contexto que surge a fase de aprovação. Nela, é verificado se o artefato, da maneira como foi especificado e implementado, irá atender às necessidades do cliente.

Umas das principais atividades desenvolvidas no controle de qualidade são os testes, as inspeções e as revisões. Sabe-se que não é possível testar um programa completamente, de modo a assegurar que não possua mais defeitos; deste modo, os testes são utilizados para diminuir as chances de um software apresentar falhas em ambientes de produção.

Porém, em determinadas situações, algumas características do código são muito difíceis ou caras para serem testadas. Assim, torna-se necessária a utilização de outras técnicas de controle de qualidade, como as revisões e inspeções.

As revisões de artefatos são atividades que visam encontrar problemas em todos os tipos de documentos, por meio de uma leitura crítica e de uma profunda avaliação da documentação gerada [22]. Ao final de uma revisão, deve-se ter um documento contendo uma lista dos defeitos, dúvidas e dificuldades encontradas, bem como sugestões de melhorias. Há várias técnicas de revisões, como: revisão pelo próprio autor, revisão por pares, revisões progressivas, *walk-through*. As revisões, apesar de em alguns casos ser uma alternativa mais barata aos testes, elas dependem de alguns fatores para o seu sucesso: habilidade dos revisores e rigor adotado pelo revisor.

As inspeções são práticas similares às revisões, porém mais formais e podem ser mais caras. São realizadas por uma equipe, seguindo um plano definido e obedecendo a regras bem definidas. Ao final de uma inspeção, também se deve ter uma lista com defeitos, dúvidas e melhorias.

Ambas as atividades são complementares aos testes e podem ser aplicadas desde as primeiras fases do projeto, o que diminui o custo relacionado ao conserto dos problemas encontrados.

3.4.Modelo GQM

A utilização de métricas é muito importante na medição da qualidade do software desenvolvido, ou seja, seu conceito está intimamente ligado ao conceito de qualidade de software. Para a obtenção de *feedback* dos usuários dos softwares desenvolvidos e avaliação do processo utilizado pela equipe técnica, é necessária a utilização de um mecanismo de medição. Desta forma, as métricas auxiliam a construção de uma memória de trabalho, ajudando na resposta a perguntas relacionadas ao desenvolvimento do software.

Este trabalho utiliza técnicas da estratégia *Goal-Question-Metric* (GQM) [4], [3] para definir seus objetivos e as maneiras para verificar se estes foram atingidos. Criado por Basili, este método define um modelo de medição de software em três níveis: meta, pergunta e métrica.

A meta é o que se deseja atingir ao aplicar o modelo, devendo ter algumas características, como objeto de estudo, propósito, foco, pessoas envolvidas e outros fatores de contexto. Por exemplo, deseja-se melhorar a usabilidade do

software, diminuir a quantidade de *bugs* encontrados e melhorar a performance do sistema.

A pergunta é representada por uma série de questões que visam verificar o resultado de um trabalho. Por exemplo: a usabilidade do software melhorou para os clientes? O número de *bugs* reportados na iteração diminuiu? A performance do sistema melhorou na última iteração?

Por último, a métrica deve estar associada à pergunta a ser respondida, para que a mesma possa ser respondida de uma maneira mensurável. As métricas podem ser objetivas – quando não dependem da interpretação de uma pessoa – e subjetivas - quando são resultados da avaliação de pessoas.

Exemplos de métricas podem ser: avaliação dos usuários em relação à usabilidade do software, a quantidade de *bugs* encontrados na última iteração comparando com a quantidade média de *bugs* por iteração e o tempo de resposta do sistema para uma determinada tarefa, em relação a este tempo algumas iterações atrás.

Neste trabalho, os seguintes objetivos foram estabelecidos:

a) Diminuir o número de erros reportados em um mês;

b) Melhorar o processo de atualização do site;

c) Diminuir os problemas relacionados ao retrabalho inútil;

d) Aumentar a popularidade do site;

e) Aumentar a popularidade das mídias sociais e seus conteúdos relacionados;

A partir destes objetivos, métricas GQM foram criadas para mensurar os avanços obtidos e seus resultados podem ser encontrados no Capítulo 8.

3.5.Resumo

A procura por fornecimento de software em geral vem crescendo atualmente. As empresas estão informatizando cada vez mais os seus processos, o que tem contribuído para o aumento da complexidade do software desenvolvido. Com esse crescimento de demanda e de software utilizado, a qualidade dos produtos desenvolvidos tem se tornado algo muito importante para os clientes de software.

Deste modo, preocupar-se com a criação de software correto por construção é muito importante, uma vez que diminui a probabilidade de ocorrências de defeitos durante a utilização do software. Porém, somente a preocupação com a correta construção do software não é suficiente para garantir uma qualidade satisfatória do mesmo. É necessário, também, que atividades de controle de qualidade (como testes, revisões e inspeções) sejam executadas. A partir destas atividades de controle, métricas são coletadas, tornando possível a verificação da qualidade do software desenvolvido, a comparação com a qualidade desejável e por fim, meios para alcançá-la.

Estes conceitos são muito importantes para este trabalho, uma vez que ele propõe aumentar a qualidade do software estudado, melhorando seu processo interno de desenvolvimento.