Estudo dos Requisitos para o Desenvolvimento de um Mecanismo de Apoio as Atividades da Garantia da Qualidade de Software

Gabriela Medeiros Aguiar[1]

[1] – Centro Universitário La Salle (UNILASALLE)
Av Victor Barreto, 2288 - Centro – Canoas – RS – Brazil

gabriela.aguiar@hp.com

Abstract. *The quality became in last decades one of the main requirements of competition in all the economy areas. In this context there are maturity models as the CMM. The CMM model is organized in levels, composed by key areas. One of the key areas of level 2 is the Software Quality Assurance key practice area. The objective of this article is to present a specification of the main requirements for the implementation of a mechanism that aims to assist the activities played for the role of the SQA. This requirements set will contribute in the area of the software quality, with reduction of papers and the agility where the consultations could be carried through.*

Resumo. *A qualidade tornou-se, nas últimas décadas um dos principais requisitos de competição em todas as áreas da economia. Neste contexto tem-se modelos de maturidade como o CMM. Uma das áreas-chave do nível 2 é a Garantia da Qualidade de Software (Software Quality Assurance key practice area). O objetivo deste artigo é apresentar uma especificação dos requisitos esseciais para a implementação de um mecanismo que vise auxiliar as atividades desempenhadas pelo papel do SQA.Esta coleção de requisitos terá grande contribuição na área da qualidade de software, com redução de pápeis e a agilidade no qual as consultas poderão ser realizadas.*

Palavras-Chave: Qualidade de software, CMM, SQA, Garantia da qualidade

1. Introdução

A qualidade tornou-se, nas últimas décadas um dos principais requisitos de competição em todas as áreas da economia. Esta cada vez mais difícil sobreviver no mercado de desenvolvimento de software, pois a exigência pela qualidade e por um bom preço tornaram-se necessidades evidentes.

Ao longo da evolução dos conceitos de software, constatou-se a necessidade de não enfocar os aspectos de qualidade apenas no produto final, mas também, na qualidade do processo de desenvolvimento de software. É por este motivo que a cada dia cresce o número de empresas investindo na melhoria de seus processos, com a preocupação de melhorar a qualidade de seu produto final.

Neste contexto, surgem modelos de qualidade densos, como por exemplo o CMM (do inglês: Capability Maturity Model), usados pelas organizações como guias para definir e melhorar seus processos de desenvolvimento de software através de níveis de maturidade. É neste modelo que encontramos uma função denominada Garantia da Qualidade de Software (no inglês: Software Quality Assurance - SQA) tendo como objetivo avaliar se as evidências encontradas durante as revisões/auditorias estão aderêntes aos processos estabelecidos na empresa.

O objetivo deste artigo é apresentar uma especificação dos requisitos essenciais para a implementação de um mecanismo que vise auxiliar as atividades desempenhadas pelo papel do SQA em uma organização, atendendo os critérios estabelecidos pelo modelo CMM nível 2.

O artigo está organizado em 7 seções. A seção 1 apresenta uma introdução do assunto que será abordado. As seções 2 e 2.1 apresentam o cenário das organizações visando a qualidade de software e a diferença entre qualidade de produto e qualidade de processo. Na seção 3 discorre-se sobre o modelo de maturidade de processos CMM. Na seção 4 aborda-se sobre a função de SQA no contexto do modelo CMM. Na seção 5 argumenta-se sobre o apoio automatizado que será implementado para auxiliar as atividades do SQA. A seção 6 apresenta as considerasções finais. As referências bibliográficas se encontram na seção 7.

2. Qualidade de Software

A produção de software com qualidade assegurada e elevada produtividade dentro do prazo e orçamento previamente estabelecido e sem necessitar de mais recursos dos que os alocados, tem sido um grande desafio para diversas indústrias de desenvolvimento de software de todo o mundo, durante várias décadas [Christensen 2001], [Fiorini 99].

As indústrias de software estão despreparadas para atender às exigências dos mercados, simplesmente porque não investiram na melhoria dos processos internos. Devido a isso, tornou-se comum e freqüente, no dia-a-dia das organizações, surgirem vendedores com soluções milagrosas sob forma de ferramentas, métodos ou outra tecnologia modista. Entretanto, sabe-se que tais soluções não existem. Cada vez mais as organizações desconhecem completamente um processo de desenvolvimento de software, fazendo com que seus produtos não sejam entregues dentro do prazo estabelecido e com o custo diferente do negociado.

A principal causa da falta de qualidade em software, é devido à falta de um processo de desenvolvimento pré-estabelecido e efetivo. Segundo Christensen, [Christensen 2001], 25% dos projetos de software não conseguem se realizar e 40% deles estão acima do orçamento estabelecido. Por este motivo, a maioria das organizações estão investindo em melhoria de processos de desenvolvimento de software, buscando, assim, um aumento da qualidade de seus produtos.

Conhecer o processo é primordial para entender como os produtos e serviços são planejados, produzidos e, enfim, entregues ao cliente. Quando não temos esta visibilidade do projeto como um todo, só conseguimos enxergar as entradas e as saídas, tornando maior a chance de encontrar algum problema, porque o processo é uma caixa preta.

Diversos autores, tais como Fenton [Fenton 2000], Abran [Abran 98], Harter [Harter 2000], Scout [Scout 2001], desenvolveram trabalhos sobre a constante busca da melhoria dos processos utilizados para a construção de software, obtendo-se, desta maneira, uma significativa melhoria na qualidade de software nas organizações.

Avaliar constantemente o processo de desenvolvimento de software garante que os problemas, ao serem identificados, sejam rapidamente revertidos. O custo de encontrar um problema ou até mesmo um defeito no software na fase inicial de desenvolvimento, ou ainda durante sua investigação, é muito menor que identificar o mesmo problema somente na fase de qualificação [Kaner 99]. Por isto, a Engenharia de Software pesquisou diversos modelos de definição, avaliação e processos de melhoria dos mesmos.

Para que se possa conhecer melhor os produtos produzidos, assim como o processo que está sendo utilizado para o seu desenvolvimento, é necessário organizar as informações coletadas, para que, posteriormente, possam ser medidas e avaliadas.

Infelizmente o cenário mais encontrado é o do desenvolvedor heróico, que não usa nenhum processo documentado, no qual, muitas vezes, somente ele conhece e entende o método utilizado. O resultado deste cenário, geralmente, são programas que não funcionam como deveriam, além de apresentarem custos excessivos, atraso do projeto, enfim, o caos. Para reduzir o efeito negativo destes problemas, a organização precisa adaptar o processo utilizado a sua realidade.

Durante a última década, foi desenvolvido um grande número de modelos com abordagem de melhoria contínua em processos de software. Um dos mais conhecidos é o CMM. Sua primeira versão foi desenvolvida em 1987 pelo SEI (do inglês: Software Engeering Institute). Segundo Christensen, [Christensen 2001], outros modelos foram inspirados na sua estrutura, tais como: Bootstrap, Trillium, PSP/TSP, ISO 9001 e TickIT e SPICE (do inglês:Software Process Improvement and Capability dEtermination).

Os benefícios do processo de melhoria de software são vários, dentre eles estão: o aumento da qualidade do produto e da produtividade; satisfação do cliente e habilidade para não ultrapassar o orçamento. De acordo com diretrizes apontadas pelo SEI, os custos para a implantação da melhoria de processos em uma empresa deve ficar entre 3% e 5% do custo de desenvolvimento do software.

2.1 Qualidade de Processo e Produto

O produto "software"está passando a ser um componente comum dentre diversos outros, desde carros, elevadores, telefones e televisão. O mercado exige qualidade e preço acessível. Portanto, o software como produto deve ter qualidade básica[1] exigido, e o seu desenvolvimento deve arcar com o melhor custo possível [Rocha 2001].

A sociedade, a cada dia, pressiona mais o setor de software para que a "qualidade" seja uma característica preponderante [Rocha 2001]. Por este motivo, cada vez mais o nome Engenharia de Software vem sendo citado nas organizações, para, realmente, fazer engenharia de produto (software). Para que isso ocorra é necessário: planejar, acompanhar, executar e controlar [Fiorini 99]. A Engenharia de Software tem como objetivo prover a melhoria da qualidade de seu produto. Pois é neste contexto que se inserem as duas visões de processo e produto. Não cabe a dúvida com relação à importância da existência de ambas as visões, pois as duas abordagens são necessárias e complementares para o sucesso do projeto [Tsukumo 2001].

A qualidade de software é determinada pela qualidade dos processos que são utilizados para o desenvolvimento do produto final. Sabe-se que a qualidade do produto é reflexo da qualidade e gerenciamento do processo que está sendo utilizado em seu desenvolvimento [Fiorini 99]. Por este motivo, a cada dia cresce o número de organizações debatendo sobre o valor agregado da melhoria de processos de desenvolvimento de software, em inglês conhecido como SPI (do inglês: Software Process Improvement) [Conradi 2002].

A qualidade de um produto de software é o resultado das atividades realizadas durante o processo de desenvolvimento do mesmo. Para avaliar a qualidade de um produto de software é necessário verificar, através de técnicas e até mesmo atividades operacionais, até que ponto os requisitos estão sendo atendidos. Estes estão explicitando, de um modo geral, a expressão das necessidades do usuário, e têm o objetivo de definir as características do software [Tsukumo 2001].

As duas visões de qualidade de processo e produto são imprescindíveis durante o desenvolvimento de software. Por serem complementares, ambas asseguram uma melhor qualidade

1 Qualidade Básica: funcionalidade, confiabilidade, facilidade de uso, economia e segurança de uso.

do produto final, pois proporcionam uma grande visibilidade de "como" e quais "caminhos" deverão ser percorridos para alcançar um processo mais maduro e efetivo na busca da qualidade.

3. Capability Maturity Model (CMM)

O modelo de Maturidade de Capacitação para software (CMM) foi desenvolvido pelo SEI, sediado na Universidade Carnegie Mellon (CMU), em conjunto com o Departamento de Defesa dos Estados Unidos (no inglês: Departament of Defense - DoD). O DoD patrocionou a criação do SEI, em meados de 1984, depois de observar que a situação dos seus contratos de desenvolvimento de software tornou-se insustentável [Paulk 2002]. O principal objetivo era proporcionar condições para a evolução de melhores práticas de Engenharia de Software, tendo como objetivo alcançar o mesmo nível de respeitabilidade e controle, encontrado em outros setores da atividade industrial e nos projetos de desenvolvimento de software dos fornecedores do DoD.

Segundo Weber, em [Weber 2001], o CMM foi baseado em conceitos da Qualidade Total - estabelecido por Crosby, mostrando que a implementação de sistemas de qualidade segue um amadurecimento gradativo em patamares nos quais o autor enumerou como: incerteza, despertar, esclarecimento, sabedoria e certeza.

O CMM propõe um caminho gradual que leva as organizações a se aprimorarem continuamente em busca da solução de problemas inerentes ao desenvolvimento de software. O modelo descreve os principais elementos de um processo de software efetivo e prevê um caminho de melhoria evolutiva revertendo um processo *ad hoc* e imaturo para um de teor maduro e altamente disciplinado [Fiorini 99], [Marczak 2003].

As organizações de software evoluem através dos seguintes estágios: quando elas definem, implementam, medem, controlam e melhoram seus processos de software. Seguindo esta idéia, para que este caminho de melhoria contínua possa ser alcançado, o CMM define 5 níveis de maturidade: Inicial, Repetitivo, Definido, Gerenciado e Otimizado. Na figura 1 é possível identificar cada nível e seu principal objetivo.

Cada um destes níveis está dividido em KPA (do inglês: Key Process Area). Ao todo são 18 áreas-chave. Cada uma contém os quesitos a serem cumpridos na implantação do modelo (ao todo são 316 práticas que especificam a função a ser desempenhada) [Fiorini 99].

Foram criados alguns modelos específicos baseados nesta estrutura para atender diferentes áreas e aspectos de uma organização, tais como a área de Recursos Humanos (People CMM) e de sistemas gerais (SE-CMM), entre outras. Existem, também, modelos para conduzir avaliações (SCE) e planejar melhoria de processos (IDEAL). O CMM abrange práticas de planejamento, engenharia e gerência para o desenvolvimento e manutenção de software, capacitando uma organização de software a produzir produtos de qualidade assegurada. [Marczak 2003], [Fiorini 99].

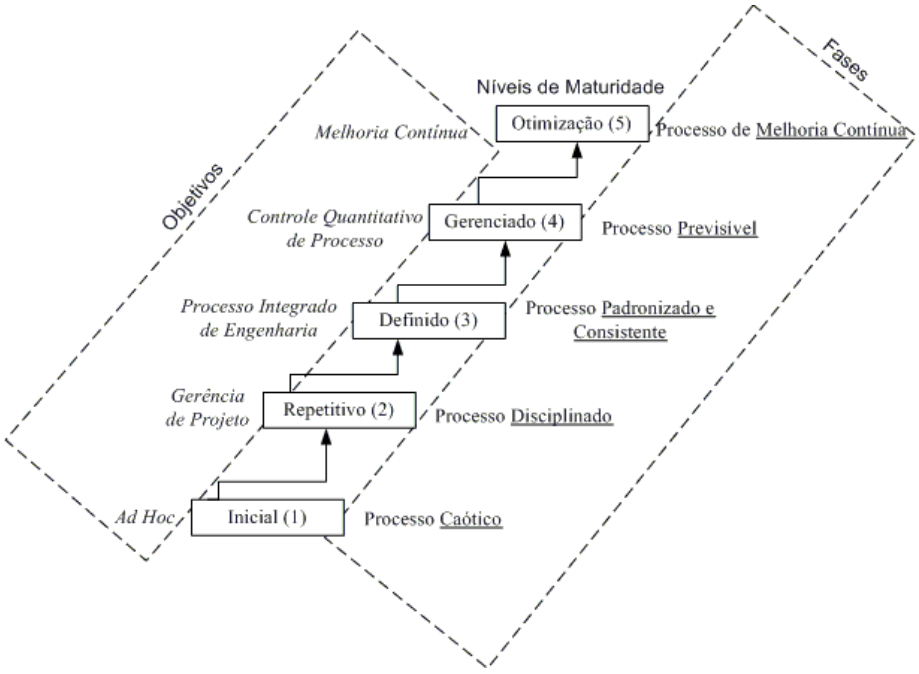

Figura 1. Os cinco níveis de maturidade do CMM [Fiorini 99]

O primeiro degrau a ser alcançado é o nível 2, pois no nível 1 o processo de desenvolvimento de software é uma caixa preta, como podemos observar na figura 2, visto que apenas as entradas e o produto final podem ser vistos com clareza. Neste nível, os maiores problemas são de ordem gerencial e não técnica.

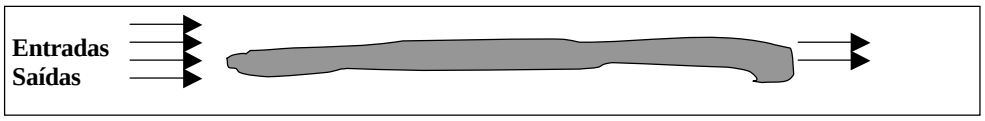

Figura 2. Caracterização: Processo imprevisível e quase sem controle [Fiorini 99]

No nível 2, observamos que o foco é gerencial tendo uma preocupação exclusiva com o planejamento de projeto de software. O modelo propõe no nível 2, um processo repetitivo, fazendo com que o mesmo passe a ter um referencial básico de controle. Neste, são estabelecidas políticas, visando gerenciar projetos de desenvolvimento de software, adequando procedimentos para implementá-los. Nesta fase, os gerentes de projetos possuem uma visibilidade do todo, conseguindo acompanhar custos, cronogramas e até mesmo funcionalidades do software a ser desenvolvido.

No nível 3 a organização possui um processo de desenvolvimento de software definido e satisfatoriamente entendido e visível, ou seja, a caixa preta passa a ser aberta. As organizações neste nível são padronizadas e consistentes porque ambas as atividades de gerência e engenharia de software estão estáveis e retíveis. Os custos, cronogramas e funcionalidades estão sendo controlados e a qualidade de software é acompanhada.

No nível 4 a gerência tem bases objetivas para a tomada de decisões, pois o processo é medido e gerenciado [Weber 2001] . Os gerentes são capazes de antecipar os resultados, cada vez mais precisos, tendo-se em vista que a variabilidade do processo se torna menor.

No nível 5, o foco é na melhoria contínua do processo, no qual mudanças na tecnologia e no próprio processo são gerenciadas de modo a não causarem impacto na qualidade do produto final. É

neste mesmo nível que se tenta de maneira sistemática e controlada, identificar, avaliar e desenvolver novas e melhores maneiras de construir software, objetivando garantir a qualidade assegurada dos produtos. É nesta fase que encontramos a organização como um todo engajada em busca da melhoria contínua de seus processos.

4. Função de SQA

Conforme o modelo de qualidade de software que é adotado por uma organização, o conceito de SQA (do inglês: Software Quality Assurance) ganha diferentes finalidades e objetivos. As abordagens de qualidade de processo e qualidade de produto mencionandas anteriormente, definem de formas diferente a atividade de SQA. A visão na qual está centrado o produto, visa garantir a qualidade do software que está sendo desenvolvido. De outro lado, a visão de qualidade do processo busca verificar a conformidade dos produtos e artefatos de um projeto se confrontado aos processos e ao procedimento que estão sendo utilizados [Marczak 2003].

A perspectiva de visão voltada para o produto não será adotada neste artigo, por não condizer com o proposto pelo modelo CMM para a função de SQA. Conforme o CMM, o(s) SQA(s) tem como propósito prover à Gerência Sênior uma visibilidade apropriada sobre o processo que está sendo aplicado no projeto de software e pelos produtos que estão sendo desenvolvidos. Para que isso ocorra, o SQA tem como objetivo revisar e auditar os processos de software e verificar se as atividades estão de acordo com os procedimentos e padrões definidos. Mediante estes resultados, deve-se informar aos gerentes envolvidos a "fotografia" do momento atual. Segundo Fiorini, em [Fiorini 99], para atingir tais objetivos, o SQA deve executar as seguintes atividades expressas no diagrama de caso de uso da Figura 3.

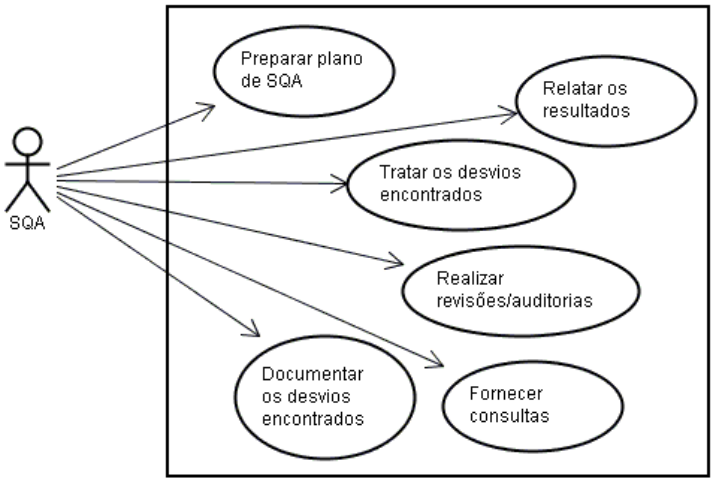

Figura 3: Diagrama de caso de uso que define as atividades do SQA

- ***Atividade 1:*** *Preparar Plano de SQA*

Quando a fase de planejamento de projeto inicia-se é gerado um documento geralmente conhecido como plano de projeto, no qual estarão previamente definidas as atividades a serem executadas e os profissionais que farão parte da equipe de projeto. Nesta fase o SQA deve planejar quais as atividades irá realizar no projeto e documentá-las em um Plano de SQA. Este deve preocupar-se em abranger os seguintes itens: processos e procedimentos que serão utilizados nas auditorias/revisões; métricas que serão coletadas; cronograma previsto para a realização das auditorias, assim como a periodicidade e a quem serão reportados os desvios encontrados.Todas as

atividades desempenhadas pelo SQA devem estar de acordo com o plano de garantia da qualidade (Plano de SQA).

- ***Atividade 2:*** *Fornecer consultas*

As atividades do SQA devem ser transparentes, fornecendo consultas a sua base de dados dando total visibilidade do andamento dos projetos. Ou seja, são as consultas que irão dizer se os procedimentos e padrões que estão sendo utilizados para o desenvolvimento dos produtos estão conforme os processos que foram estabelecidos pela organização.

- ***Atividade 3:*** *Relatar os resultados*

Baseado nos desvios encontrados nas auditorias/revisões realizadas, o SQA deve elaborar relatórios descrevendo todas as inconsistências e apresenta-los aos gerentes de projeto, a Gerência Sênior e ao cliente. Desta forma, ambos irão acordar como serão tratados os desvios encontrados.

- ***Atividade 4:*** *Realizar revisões/auditorias*

O SQA deve realizar revisões/auditorias periódicas, para verificar se as atividades do projeto estão sendo conduzidas de acordo com os procedimentos e padrões previamente estabelecidos pela organização. Ou seja, revisar se as atividades definidas para o projeto estão de acordo com os processos de software e se as mesmas estão sendo realizadas.

- ***Atividade 5:*** *Tratar os desvios encontrados*

Após a realização das auditorias/revisões o SQA deve apresentar os resultados a Gerência Sênior para este ter a total visibilidade de todo o processo e assim ambos tratarem os desvios encontrados. Desta maneira, é realizado um acompanhamento de todos os desvios, assim como a sua resolução.

- ***Atividade 6:*** *Documentar os desvios encontrados*

Sugere-se que todas as inconsistências, ou seja os possíveis pontos fracos do projeto, sejam documentados em forma de relatórios. Este documento deve conter a descrição dos desvios encontrados, as datas limite para estes serem corrigidos.

É importante ressaltar que as atividades de SQA, segundo o modelo CMM, podem ser desempenhadas por uma única pessoa ou por um grupo, desde que ambos sejam membros independentes do projeto que está sendo desenvolvido. Somente desta forma o SQA terá condições de fornecer uma visão "independente" sobre a aderência ou não aos padrões definidos. Para que essas condições sejam satisfeitas o SQA deve ser avaliado externamente de tempos em tempos.

Esta pessoa ou grupo de pessoas que desempenham o papel de SQA na organização, geralmente costumam fazer parte de um setor ou área específica, no qual normalmente é denominada de SEPG (do inglês: Software Engineering Process Group). O SEPG tem como objetivo definir e buscar a melhoria contínua dos processos de engenharia de software.

Para atuar em uma função típica de auditoria, o SQA deve estar atento para conceitos próprios desta função. O auditor não possui função punitiva, pois deve atuar como um assessor dos níveis de maturidade da organização, tendo como sua principal meta alertar sobre possíveis desvios e não-aderências dos produtos/artefatos de software com relação ao processo estabelecido.

Pode-se definir a função de SQA em coletar e avaliar as evidências para determinar o nível de aderência aos padrões e processos estabelecidos na organização. A função de SQA pode ser caracterizada como um processo de melhoria contínua e validação dos processos de software.

5. Apoio Automatizado

Um dos requisitos mais importantes para o sucesso de qualquer organização nos dias de hoje é a garantia da qualidade dos seus projetos. Devido a esta preocupação, cada vez mais organizações estão investindo em qualidade de software. Para que isso cresce diariamente o número de organizações que vêm adotando modelos de qualidade de software para garantir a qualidade de seus produtos.

Neste contexto, a função de SQA torna-se uma atividade quase indispensável. Tendo em vista este cenário, surge a necessidade de ferramentas de apoio visando garantir a qualidade constante dos processos de desenvolvimento de software.

A maioria dos modelos de qualidade de software propõe diversas atividades genéricas as quais devem ser realizadas de maneira freqüênte. Devido a este fato, é necessário que cada organização faça um mapeamento da sua realidade com o modelo escolhido. Por este motivo, surge um problema com relação à aquisição de ferramentas para auxiliar as atividades de SQA, considerando que a realidade de cada organização se difere por terem necessidades e culturas diferentes. Torna-se, portanto, praticamente inviável desenvolver um protótipo de ferramenta que auxilie de forma efetiva nas atividades do SQA contemplando qualquer organização. Então, as organizações estão procurando desenvolver soluções nas quais supram suas necessidades, ao ínves de procurá-las no mercado de software, baseando-se nesta limitação. Sendo assim, este artigo apresenta os requisitos elementares para a automação das principais atividades desempenhadas pelo papel do SQA, visando contribuir de forma significativa para área de Qualidade de Software. Com este apoio automatizado, será possível aumentar a produtividade do SQA, reduzindo, assim, o número de papéis e o tempo gasto para a consolidação dos resultados das revisões.

Atualmente a maior parte das auditorias realizadas pelo SQA são todas feitas manualmente, sem nenhum auxílio de automação. Para que os resultados sejam consolidados, em geral gasta-se um tempo considerável. A realização de auditorias com periodicidade pré-definida, figura como norma obrigatória no âmbito das organizações. Outro aspecto a ser destacado é a necessidade de tornar o procedimento ágil, através da automação, visto que, com o recurso manual, emprega-se tempo desnecessário e aumenta-se, substancialmente, o custo da mão-de-obra. Para solucionar este problema surgiu a necessidade de desenvolver um mecanismo de apoio as atividades do SQA.

Nesse contexto, foram realizadas entrevistas com o SQA de uma empresa na qual atualmente não está sendo utilizado nenhum tipo de auxílio de automação para esta atividade. Todo o processo de auditoria é realizado manualmente, desde a coleta de dados até a consolidação dos mesmos, sendo apenas utilizado como auxílio para esta atividade planilhas do MS Excell. Tendo em vista o grande tempo gasto para a realização destas atividades, foram levantados alguns requisitos que são de caráter essencial, para a implementação de um mecanismo de apoio às auditorias/revisões do SQA.

É importante salientar que os requisitos abaixo não são únicos, porém estes são elementares para a automação das principais atividades desempenhadas pelo SQA.

Os seguintes requisitos funcionais foram identificados como primórdiais para a automação de parte das atividades desempenhadas pelo SQA na organização:

Requisito 1 : Preparar Plano SQA (Cenário A)

Requisito 2 : Executar auditoria/revisão SQA (Cenário B)

Requisito 3 : Gerar relatório/gráfico (Cenário C)

Requisito 4 : Consultar as inconsistências de cada projeto (Cenário D)

Requisito 5 : Tratar os desvios encontrados (Cenário E)

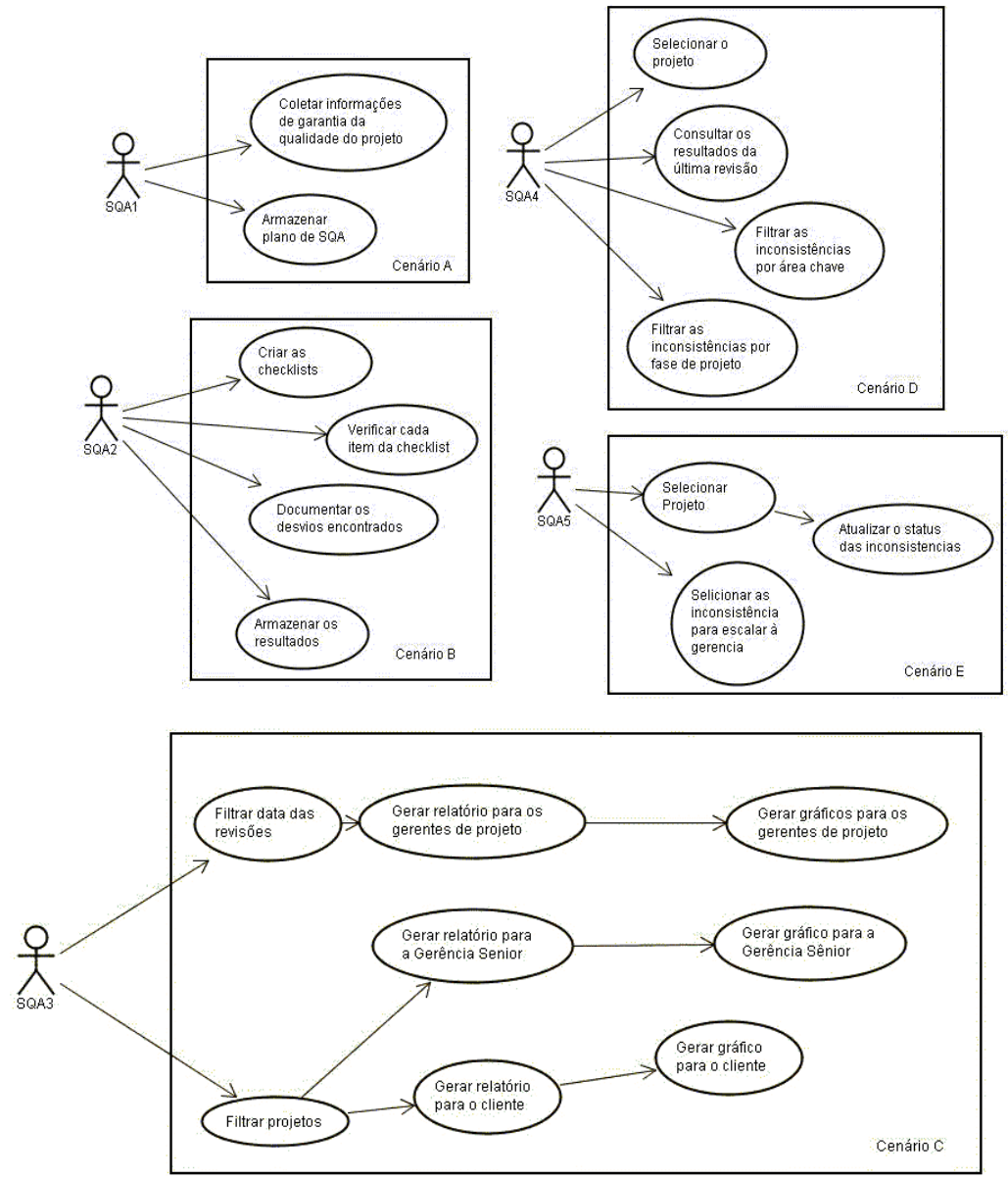

Figura 4. Diagrama de caso de uso de cada requisito com seus respectivos cenários

A Figura 4 representa em forma de caso de uso os requisitos que foram identificados como essencias, para a implementação de um mecanismo de apoio as atividades desempenhadas pelo SQA.

A atividade de fazer auditorias/revisões SQA é uma funcionalidade de extrema importância nas organizações, pois neste momento será identificado o número de inconsistências de cada projeto (Requisito 1). Para agilizá-la, surgiu a necessidade de outro requisito (Requisito 5), com a função de alertar as inconsistências existentes nos projetos, possibilitando uma maior agilidade ao SQA. Mediante os resultados, na maioria das vezes, é realizado uma reunião com o gerente do projeto para expor todos os pontos fracos que foram encontrados na auditoria. Geralmente estes resultados são apresentados aos gerentes de projeto na forma de relatório, por isso é identificado outro requisito (Requisito 3), para automação desta atividade.

Após a realização das auditorias e a exposição dos resultados aos gerentes de projeto, é acordado entre ele e o SQA uma data limite para regularização das inconsistências encontradas. Após esta data, é realizado uma nova revisão nos projetos para ver se ainda existe alguma inconsitência, caso seja encontrado alguma reincidência conforme os padrões previamente definidos pela organização, este desvio é encaminhado para a Gerência Sênior. Por este motivo, surge outro requisito (Requisito 5), com a funcuionalidade de escalar essas inconsistências para a Gerência Sênior.

O processo utilizado na organização deve ser sempre transparente para todos, inclusive para a Gerência Sênior. Por este motivo, após a consolidação dos resultados são gerados gráficos para analisar a evolução das inconsistências por projetos e na organização como um todo (Requisito 3).

Existe outro requisito (Requisito 4), muito importante que é a possibilidade de fazer consultas e o acompanhamento de cada projeto de forma mais rápida, aumentando assim a produtividade do trabalho do SQA. Com esse requisito implementado é possível aumentar o número de auditorias/revisões nos projetos, pois o tempo gasto para a consolidação dos dados tende a diminuir.

Este mecanismo, visa apoiar as atividades de auditoria realizada pelo SQA, possibilitando uma grande contribuição na área da qualidade de software, com redução de papéis e a agilidade em que poderão ser realizadas as consultas.

O principal objetivo deste mecanismo é verificar se os projetos que estão sendo desenvolvidos estão aderentes aos processos que foram estabelecidos na organização. Além disso, disponibilizar o relatório das auditorias realizadas para a Gerência Sênior e a todos os interessados que estão envolvidos no processo. Com este procedimento possibilita-se uma maior visibilidade das fases do ciclo de vida de cada projeto, sendo possível verificar de forma mais ágil se as atividades de desenvolvimento de software estão conforme o processo que está sendo utilizado.

6. Considerações Finais

Atualmente, aplicações de software fazem parte do nosso cotidiano. Embora muito já tenha sido feito nos últimos anos, resta, ainda, muito para ser executado na busca de melhorias no processo de desenvolvimento de software. Entretanto, é importante salientar que a qualidade de software está fortemente relacionada à qualidade do processo de software [Fiorini 99], [Rocha 2001]. Por este motivo, aumenta a preocupação das organizações com a modelagem e melhores práticas no processo de software.

Diversas abordagens importantes como as normas Bootstrap, Trillium, PSP/TSP, ISO 9001 e TickIT, SPICE e CMM, sugerem que, se o processo de software for melhorado, conseqüentemente podemos melhorar o produto [Rocha 2001], [Messnarz 99]. Para isso, é necessário documentar, definir, medir, analisar, comparar dados e alterar os processos.

Segundo Fiorini, em [Fiorini 99], o CMM é um modelo que demonstra quais etapas são necessárias para que uma empresa de software produza produtos de qualidade assegurada. O modelo

é composto por cinco níveis de maturidade. Cada um deles é composto por várias KPAs, com exceção do nível 1 que não possui nenhuma área-chave. No nível 2, observamos que o foco é gerencial, tendo uma preocupação exclusiva com o planejamento de projeto de software. Neste nível, encontramos uma KPA chamada SQA, que tem como propósito fornecer à gerência a total visibilidade da eficácia dos processos que estão sendo utilizados pelo projeto de desenvolvimento de software através de auditorias internas e/ou externas.

Hoje em dia, a maior parte das auditorias realizadas pelos SQA(s) são todas feitas manualmente, sem nenhum auxílio de automação. Para que os resultados sejam consolidados, em geral gasta-se um tempo considerável.

Neste contexto surgiu a necessidade de desenvolver um mecanismo de apoio às atividades do SQA, pois a maioria dos modelos de qualidade de software propõe diversos procedimentos genéricos, os quais devem ser realizados, porém é necessário que cada organização faça um mapeamento da sua realidade com o modelo escolhido. Entretanto, surge um problema com relação à aquisição de ferramentas para auxiliar nas atividades de SQA, considerando que a realidade de cada organização se difere por terem necessidades e culturas diferentes. Tendo em vista esta necessidade latente em grande parte das organizações, este artigo apresenta os requisitos elementares para a automação das principais atividades desempenhadas pelo papel do SQA, visando contribuir de forma significativa para a área de Qualidade de Software. Com este apoio automatizado, será possível aumentar a produtividade do SQA, reduzindo, assim, o número de papéis e o tempo gasto para a consolidação dos resultados das auditorias/revisões.

Este artigo é apenas o primeiro passo para o início do desenvolvimento desta solução. È importante ressaltar que será necessário construir um modelo priorizando realizar a automação destes requisitos funcionais que foram aqui mencionados. Após isso, será realizada uma experimentação de forma empírica para validar o protótipo que será construído. Para uma maior contribuição, a área de Qualidade de Software poderia auxiliar no andamento do processo disponibilizando um estudo mais detalhado visando definir os requisitos que possam ser utilizados de forma genérica em qualquer organização.

7. References

Christensen, Mark J.; Thayer, Richard H.(2001) "The Project Manager's Guide to Software Engineering's Best Practices", Los Alamitos, California:IEEE - Computer Society.

Fiorini, Soeli T; STAA, Arndt von; Baptista, Renan Martins.(1999) "Engenharia de Software com CMM"., Rio de Janeiro: Brasport.

Fenton, Norman E.; Neil, Martin, (2000) "Software Metrics: Roadmap"; Artigo ACM., Disponível em: http://portal.acm.org/citation.cfm?id=336588&coll=Portal&dl=ACM&CFID=11402421&CFTOKEN=77708879, May.

Abran, Alain; Merlo, Ettore, (1998) "Process Assurance Audits: Lessons Learned", Artigo ACM., Disponível em: http://portal.acm.org/citation.cfm?id=302219&coll=Portal&dl=ACM&CFID=11686374&CFTOKEN=5728596, Abril.

Harter, Donald E.; Slaughter, Sandra A.(2000) "Process Maturity and Software Quality: A field Study", Artigo, ACM., Disponível em: http://portal.acm.org/citation.cfm?id=359769&coll=Portal&dl=ACM&CFID=11686374&CFTOKEN=5728596., Dezembro.

Scout, Louise; Jeffery, Ross; Carvalho, Lucila; D'ambra, John; Rutherford, Philip.(2001) "Practical Software Process Improvement-The IMPACT Project", Artigo, IEEE., Disponível em: http://search2.computer.org/advanced/Proceeding_Result.jsp, Agosto.

Kaner,Cem; Falk, Jack; NGUYEN, Hung Quoc.(1999) "Testing Computer Software"., Estados Unidos: Wiley.

Rocha, Ana Regina Cavalcanti da; Maldonado, José Carlos; Weber, Kival Chaves.(2001) "Qualidade de Software Teoria e Prática"; São Paulo: Prentice Hall.

Tsukumo, Alfredo N.; Rêgo, Claudete M. ; Salviano, Clenio F. ; Azevedo, Glaucia F. ;Meneghetti, Luciano K. ; Costa, Márcia C. C. ; Carvalho, Mário B. ; Colombo, Regina M.T.(2001) "Qualidade de Software: Visões de Produto e Processo", Artigo, ATAQS, CTI. Disponível em: www.psphome.hpg.ig.com.br/downloads/ processos_produtos.pdf , Agosto.

Conradi, …Reidar; Fuggetta, Alfonso.(2002) "Improving Software Process Improvement", Artigo, IEEE., Julho/Agosto.

Paulk, Mark C.; Weber, Charles V.; Chrissis.(2002) "The Capability Maturity Model: Guidelines for Improving the Software Process"., United States of America: Carnegie Mellon University Software Engineering Institute.

Weber, Kival Chaves; Rocha, Ana Regina Cavalcanti da; Nascimento, Célia Joseli;.(2001) "Qualidade e Produtividade em Software"; São Paulo: Makron.

Marczak, Sabrina; Sá, Luciana; Ceccato, Ilmari; Audy, Jorge; Antunes, Dante;.(2003) "Uma proposta de Organização e funcionamento da função de Garantia da Qualidade de Software em um contexto de implantação do SW-CMM", Artigo, SBQS., Setembro.

Messnarz,Richard; Tully, Colin. Better (1999) "Software Practice For Bussiness Benefit Principles and Experience"., California: IEEE Computer Society Los Alamitos.