Universidade Estadual de Maringá
Centro de Tecnologia
Departamento de Engenharia de Produção
Curso de Engenharia de Produção

USO DE FERRAMENTAS DE QUALIDADE NO PROCESSO DE DESENVOLVIMENTO DE SOFTWARE

Cristyan Leonardo Paintner

TCC-EP-32-2009

Maringá - Paraná
Brasil

Universidade Estadual de Maringá
Centro de Tecnologia
Departamento de Engenharia de Produção
Curso de Engenharia de Produção

USO DE FERRAMENTAS DE QUALIDADE NO PROCESSO DE DESENVOLVIMENTO DE SOFTWARE

Cristyan Leonardo Paintner

TCC-EP-32-2009

Trabalho de Conclusão de Curso da Engenharia de Produção, do Centro de Tecnologia, da Universidade Estadual de Maringá.
Orientador(a): Prof.(ª): Gislaine Camila Lapasini Leal

Maringá - Paraná
2009

Cristyan Leonardo Paintner

USO DE FERRAMENTAS DE QUALIDADE NO PROCESSO DE DESENVOLVIMENTO DE SOFTWARE

Este exemplar corresponde à redação final do Trabalho de Conclusão de Curso aprovado como requisito parcial para obtenção do grau de Bacharel em Engenharia de Produção da Universidade Estadual de Maringá, pela comissão formada pelos professores:

_____

Orientador: Gislaine Camila Lapasini Leal
Departamento de Engenharia de Produção, CTC

_____

Prof: Edwin Vlademir Cardoza Galdamez
Departamento de Engenharia de Produção, CTC

Maringá, Setembro de 2009

Agradecimentos

A Deus, por me amar primeiro.

A Professora Gislaine Camila Lapasini Leal, pelas orientações, ensinamentos e principalmente pela sua SUPER paciência, para que eu concretiza-se esse trabalho.

Aos professores do departamento de Engenharia de Produção, pelos conhecimentos transmitidos e pela amizade nesses anos de convivência. E também aos professores do departamento de Informática que contribuíram na minha formação.

A aqueles com quem aprendi a ouvir sertanejo e hoje os chamo de companheiros Kleber Basso, Luiz Alexandre Mareze, Kelen Basso, Rafael Barreiros, Gabriel Barreiros, Vagner Rodrigo, Marcelo Detomasi. Em especial quero agradecer ao Kleber e ao Luiz Alexandre pelos cinco anos que passamos juntos na República Mata-Virgem.

A Humberto Schiavon Filho e William Fabris Araujo em nome de todos os amigos de perto que dividiram comigo esses últimos cinco anos da minha vida, mais que amigos vocês são meus irmãos.

A Marllon Singh Zorzenoni e Guilherme Minikovisk em nome de todos os amigos que estão longe, pois mesmo não os vendo sempre tenho a segurança que posso contar sempre.

A toda turma de engenharia de produção de 2005.

Enfim, a todos que de forma direta ou indiretamente estiveram envolvidos construção do Crys Paintner que existe hoje.

Dedico

A minha mãe Myriam Jaqueline Batista Faleiro por todo o esforço incondicional para que eu pudesse chegar até aqui.
A Minha Avó Benedita Raymunda de Jesus que com certeza é a maior “culpada” de eu estar onde estou. Com um porto seguro igual a este só Deus sabe onde posso chegar.

Ofereço

A Família Paintner a qual me orgulho muito de fazer parte, aos Faleiros que também chamo de família. E a minha Bisavó Palmira Poiatti de Jesus, pelas orações e conselhos.

“[...] Se eu soube-se antes o que sei agora...
Erraria tudo exatamente igual [...]”

Agagê

RESUMO

Para a nova fase onde o software emprega robustez é fundamental que o seu processo de desenvolvimento deixe de ser um trabalho artesanal para se tornar algo próximo a uma manufatura de bens tangíveis. Neste contexto surgem então diversas metodologias de desenvolvimento para garantir a qualidade de software. Este trabalho mostra que a qualidade de software é algo que deve ser levado em consideração em todo momento do ciclo de vida do aplicativo. Uma das características que podem elevar o nível de qualidade do software é o desenvolvimento iterativo e incremental. Além disso, são abordadas as falhas do processo de desenvolvimento em cascata e a apresentação e descrição das características do *Rational Unified Process* (RUP).

Além da metodologia iterativa e incremental o uso de ferramentas clássicas de qualidade vem sendo empregado no desenvolvimento de softwares, tais ferramentas sofrem algumas customizações e são amplamente utilizadas dentro de diversas metodologias.

Este trabalho integrará as ferramentas QFD e ciclo PDCA numa adaptação da metodologia de desenvolvimento da Rational conhecida por RUP.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

QFD	*Quality Function Deployment*
SQFD	*Software Quality Function Deployment*
RUP	*Rational Unified Process*
PMI	*Project Management Institute*
PDCA	*Plan, Do, Check, Action*
FMEA	*Failure Mode and Effect Analysis*
PMBOK	*Project Management Body of Knowledge*
TQC	*Total Quality Control*

1 INTRODUÇÃO

O software é definido como sendo, um conjunto de instruções que quando executadas produzem a função e o desempenho desejados, ou também como "estrutura de dados que possibilitam que programas manipulem adequadamente a informação". (Pressmam, 2005 p. 12).

Segundo Pressman (2005), durante as três primeiras décadas da era do computador, os esforços eram concentrados em produzir um hardware que diminuísse o custo de processamento e armazenagem dos dados. Atualmente o software ultrapassou o hardware como a chave para o sucesso de muitos sistemas baseados em computador.

Para essa nova fase onde o software ganha corpo e robustez foi fundamental que o seu processo de desenvolvimento fosse redesenhado deixando de ser um trabalho artesanal para se tornar algo próximo a uma manufatura de bens tangíveis. Neste contexto surgiu então a gerência de projeto de software para as mais diversas metodologias de desenvolvimento.

A gerência de projetos consiste na aplicação de conhecimento, habilidades, ferramentas e técnicas às atividades do projeto a fim de atingir os requisitos do mesmo. A gerência eficaz de projetos é conseguida através do uso de processos, tais como: iniciar, planejar, executar, controlar e encerrar. Segundo Beck (2000), existem quatro variáveis principais durante o desenvolvimento de um projeto: custo, tempo, qualidade e escopo. E estas precisam ser controladas com eficiência a fim de se obter os resultados esperados. Muito comum um projeto de software passar por problemas durante a sua realização e ser encerrado antes da sua conclusão, mais comum ainda é o projeto estourar o orçamento do projeto.

O Gerenciamento de projeto, no entanto não é visto como uma etapa clássica do processo de desenvolvimento de software, uma vez que ele acompanha todas as etapas tradicionais: Concepção analise, projeto, desenvolvimento, testes e manutenção.

O PMI - *Project Management Institute*, caracteriza projeto como um "empreendimento temporário, planejado, executado e controlado com o objetivo de criar um produto ou serviço único" (PMBOK, 2004). De acordo com Pressman (2005), para que um projeto de software

seja bem sucedido, é fundamental que parâmetros como, escopo do software, os riscos envolvidos, os recursos necessários às tarefas a serem realizadas, os indicadores a serem acompanhados, os esforços e custos aplicados e a sistemática a ser seguida, sejam corretamente analisados. É atribuído ao gerenciamento de projeto a função de análise destes e outros parâmetros ainda antes da etapa de desenvolvimento, ou seja, antes do trabalho técnico, e tal analise percorre em paralelo ao desenvolvimento à medida que o software vai se concretizando.

Entre os diversos modelos de processo encontrasse o RUP – *Rational Unified Process,* "processo capaz de fornecer uma abordagem disciplinada para assumir tarefas e responsabilidades dentro de uma organização de desenvolvimento" (Kruchtem, 2003 p.15). Seu foco é assegurar a produção de software de alta qualidade que satisfaça a necessidades de seus usuários finais dentro do prazo e orçamento previsíveis. Segundo Herden (2007) o RUP pode ser considerado um processo estabelecido e que se adapta de acordo com o domínio da aplicação e que captura muitas das praticas modernas de desenvolvimento de software como desenvolvimento iterativo, gerencia de requisitos e controle continuo da qualidade do software.

Com a crescente demanda por projetos de software, a indústria de software vem buscando novas metodologias e ferramentas para o desenvolvimento de sistemas de alta qualidade e a custos reduzidos. Nesse contexto, ferramentas e metodologias voltadas para produção de bens e serviços tangíveis foram adaptadas para a indústria de software, como exemplo o caso do QFD - *Quality Function Deployment,* no português, Desdobramento da Função Qualidade, que sofreu algumas adaptações e deu origem a SQFD – *Software Quality Function Deployment*, metodologia bastante utilizada no processo de análise de requisitos.

Outras ferramentas e metodologias como o ciclo PDCA, PMBOK, FMEA e Certificações também estão sendo incorporadas por empresas de desenvolvimento de software a fim de tornar o processo de desenvolvimento mais eficaz e a entrega de um produto de qualidade ao usuário final.

Este trabalho tem o intuito de identificar ferramentas que podem ser aplicadas nas diversas etapas as quais o software é submetido durante seu processo de construção de modo a agregar valor ao projeto de software e consequentemente agregando valor ao produto final.

1.1 Objetivos

1.1.1 Objetivo geral

Este trabalho tem como objetivo incorporar ao desenvolvimento de software o uso de ferramentas e metodologias de qualidade e desenvolvimento de produto.

1.1.2 Objetivos específicos

- Estudar o processo de desenvolvimento de software;
- Estudar as ferramentas e metodologias de qualidade;
- Incorporar tais ferramentas ao processo de desenvolvimento.

1.2 Estrutura do Trabalho

Este trabalho encontra-se estruturado em cinco capítulos. O capítulo 1 apresenta uma breve introdução sobre processos de desenvolvimento de software, metodologias, processos e como o uso de ferramentas pode vir a agregar valor ao produto final. O capitulo 2 é abordado o conceito de processo e metodologia e é realizada uma breve comparação entre alguns processos já estabelecidos de desenvolvimento, tal comparação serve como embasamento para a justificativa da escolha da metodologia que será utilizada para o desenvolvimento da pesquisa. Ainda no capitulo 2, o tópico 2.2 traz a importância da qualidade atualmente nos processos de produção de bens e serviços, e por fim é apresentada uma abordagem sobre as ferramentas tradicionais de qualidade e suas adaptações ao processo de desenvolvimento de software.

O capítulo 3 apresenta a abordagem proposta, apresentando e descrevendo as etapas e atividades realizadas durante o processo de desenvolvimento de software e já mostrando em quais momentos do processo as ferramentas de qualidade podem ser incorporadas.

O capítulos 4 abordará a conclusão sobre os possíveis ganhos com o uso da abordagem proposta do capítulo 3.

2 REVISÃO DA LITERATURA

2.1 Processo de desenvolvimento de software

O software é caracterizado como o combustível dos negócios modernos, sendo atribuído a ele as tarefas de criar, acessar, e visualizar informações de formas anteriormente inconcebíveis (Kruchten, 2003).

O desenvolvimento de software consiste da atividade de criar ou modificar softwares existentes. Para que tal desenvolvimento ocorra de forma eficaz e os resultados finais possam ser atingidos é fundamental que este desenvolvimento seja moldurado em processos de desenvolvimento de software e exista uma eficiente gerencia durante todo o projeto. De acordo com Vasco (2006), o sucesso de um projeto de desenvolvimento de software começa no devido planejamento e na escolha de uma metodologia compatível com as características do mesmo.

Segundo Marreco (2006 apud Humphrey 1987, p.15), o "processo de desenvolvimento de software é definido como um conjunto de atividades, métodos, práticas e tecnologias que as pessoas utilizam para desenvolver e manter software e produtos relacionados". É possível encontrar diversos níveis de maturidade dos processos de desenvolvimento de software, desde processos maduros, onde o processo é todo documentado, padronizado e apoiado por ferramentas e metodologias, como também processos ad-hoc ou relativamente novos, que estão constantemente passando por aperfeiçoamentos.

A escolha do modelo que vai servir de estrutura para o processo é um dos fatores que possui influencia direta nas atividades de desenvolvimento, pois o conjunto de características de cada modelo são únicos. De acordo com o modelo escolhido atividades e responsabilidades são estabelecidas e se torna possível identificar os chamados elementos básicos de desenvolvimento. Esses elementos ficam mais ou menos evidentes dentro do desenvolvimento dependendo do processo instanciado. Alguns destes são artefatos, papéis, fluxos e atividades.

Por artefato entende-se um pedaço de informação que é produzida ou consumida por um processo, estes são produtos tangíveis do projeto, que servem como entrada para determinadas atividades e também são os produtos de outras. Como exemplos de artefatos têm-se modelos de documentos, código fonte, arquivos executáveis e outros (Kruchtenk, 2003). A atividade é considerada uma unidade de trabalho executada por um papel, geralmente com um objetivo claro de criar ou atualizar um artefato e o elemento papel pode ser um desenvolvedor ou um grupo de indivíduos responsáveis por determinada tarefa dentro do contexto de uma organização de desenvolvimento de software.

Na busca por melhorias em termos de produtividade e qualidade, uma parcela de especialistas tentam formalizar as etapas da tarefa de escrever um software, enquanto outra parcela aplicam técnicas de gerenciamento de projeto na escrita de software. Da junção dessas abordagens e outras existentes surgiram inúmeros modelos de processo de desenvolvimento de software.

O modelo mais antigo, bem conhecido e considerado tradicional de desenvolvimento é o modelo em cascata, em que os passos de desenvolvimento são executados de forma estritamente sequencial pelos desenvolvedores. Depois que cada etapa é terminada, o processo segue para a próxima etapa e o produto da etapa anterior é tomado como base para o início da etapa posterior. Este modelo de processo implica em um enorme esforço nas etapas inicias, ou seja, durante o levantamento dos requisitos e no desenho da arquitetura. Pois tal modelo de processo não fornece meios para corrigir os erros nas etapas inicias, o que pode implicar em funcionalidades de software desnecessárias ou sem uso, e aumentando em muito a probabilidade de uma funcionalidade vital não ser abordada, por não ter sida compreendida pelo desenvolvedor. Neste tipo de processo a localização de uma falha, ou mesmo a alteração de algum requisito, pode significar impactantes alterações no projeto.

Na Figura 1 é possível observar a representação gráfica do modelo cascata, e as fases que o compõem.

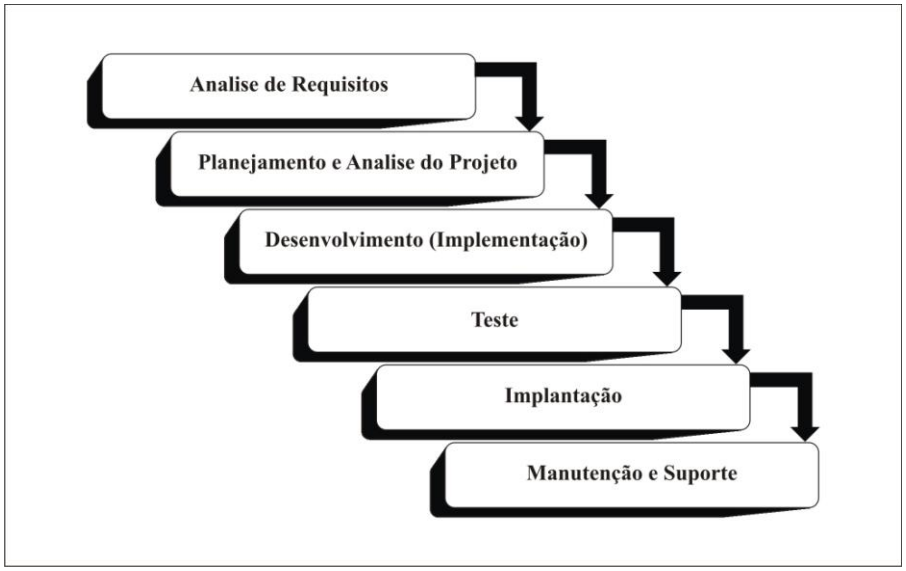


Figura 1 – Modelo cascata de desenvolvimento

Existem atualmente diversas metodologias de desenvolvimento, com abordagens diferentes de processo. Destacam-se neste contexto as metodologias iterativas e metodologias ágeis. Nas metodologias iterativas o desenvolvimento do projeto é feito de forma incremental e a cada iteração uma parte do sistema é desenvolvida, sendo o produto desta iteração superior à da iteração anterior (Vasco, 2006). Ao usar a forma incremental de desenvolvimento é possível que os próprios desenvolvedores aprendam sobre o mesmo, possibilitando a localização de futuros problemas em fases iniciais, extinguindo assim a maior falha do modelo cascata. São exemplos de processos iterativos o RUP – *Rational Unified Process*, o *OPEN* e o *Catalysis.*

As metodologias ágeis também são baseadas em iterações, buscando assim redução de riscos ao projeto. Cada iteração libera uma versão, no inglês *release*, com determinadas restrições de funcionalidades ao cliente. As metodologias ágeis se apóiam em aspectos humanos promovendo interação na equipe de desenvolvimento e o relacionamento de cooperação com o cliente neste modelo comunicação face-a-face é preferida à documentação compreensiva (Vasco, 2006). Exemplos de processo baseados nesta metodologia são *Extreme Programming (XP), Agile modeling, Crystal.*

O contraste de metodologias iterativas e ágeis se dá pelos períodos das iterações, embora as duas apresentem a estrutura de construção incremental, na metodologia ágil às iterações são menores, medidos em semanas ou quinzenas, enquanto iterações de processos iterativos podem ser mensuradas em meses.

Metodologias para o desenvolvimento de software independente de seus processos específicos buscam reduzir riscos e aumentar a qualidade do produto gerado. O RUP busca esse fim, através do envolvimento do cliente, iterações, testes contínuos e flexibilidade (Vasco, 2006).

2.1.1 RUP

O software é concebido para servir seus usuários, portanto para construir um sistema de sucesso deve-se saber quem são seus usuários e o que eles querem e precisam. O termo usuário representa alguém ou alguma coisa, como outro sistema que interage com o sistema que está sendo desenvolvido.

Um caso de uso é um "pedaço de funcionalidade do sistema que dá ao usuário um resultado de valor" (Martins, 1999). Casos de uso capturam requisitos funcionais e vários casos de uso juntos descrevem uma funcionalidade completa do sistema. Esta abordagem substitui a especificação funcional tradicional e ainda com um adicional, que seria uma especificação por usuário.

Os casos de uso direcionam o processo de desenvolvimento, já que, baseados nestes os desenvolvedores criam uma série de modelos de projeto e implementação. Os responsáveis pelos testes garantem que os componentes implementados cumpram corretamente os seus objetivos. Desta forma, os casos de uso não somente iniciam o processo de desenvolvimento, mas também o mantêm coeso.

Direcionar o processo a casos de uso significa executar uma seqüência de tarefas derivadas dos casos de uso. Eles são especificados, projetados e servem de base para a construção dos casos de teste (Martins, 1999). Estes são desenvolvidos juntamente com a arquitetura do sistema, de modo que, os casos de uso direcionam a arquitetura do sistema, que por sua vez influencia a seleção dos casos de uso. Portanto, afirma-se que ambos amadurecem no decorrer do ciclo de vida do sistema. A Figura 02 demonstra um exemplo de Casos de Uso na notação UML.

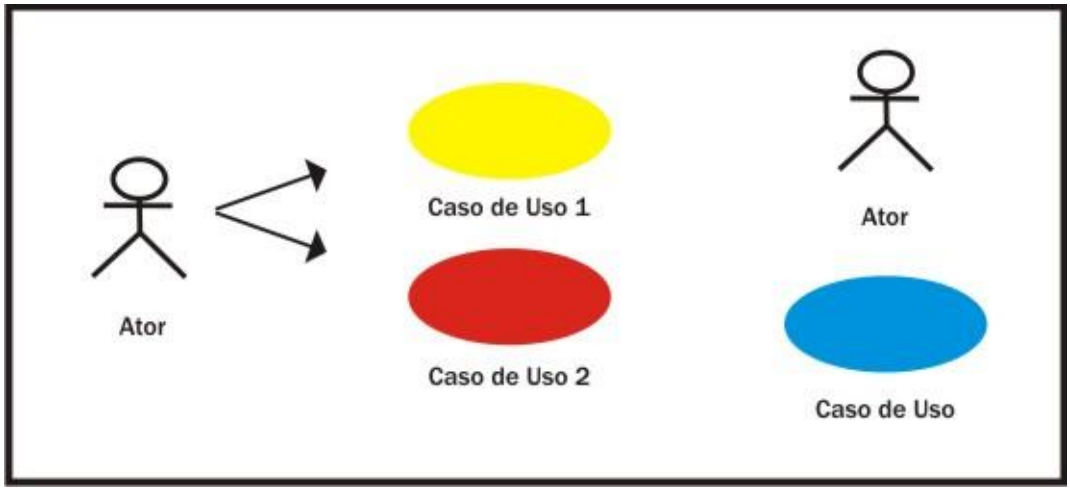


Figura 02 – Notação UML para casos de uso

O RUP é um processo iterativo e incremental que provê uma abordagem disciplinada para o desenvolvimento de software. É caracterizado por um conjunto de passos parcialmente ordenados com a intenção de construir um produto de software de qualidade, capaz de atender às necessidades e exigências do usuário final, de acordo com planejamento e orçamento previstos (Kruchten, 2003).

O desenvolvimento de software é uma tarefa que pode ser estendida por vários meses, possivelmente um ano ou mais, isso o torna de difícil mensuração e gerencia, e estes são fatores decisivos para seu sucesso. Portanto é mais prático dividir o trabalho em pedaços menores ou mini-projetos, e cada mini-projeto é uma iteração que resulta em um incremento (Ribeiro 2005).

Os desenvolvedores selecionam o que será contemplado em cada iteração baseado em dois fatores, os casos de uso que juntos estendam a usabilidade do produto em desenvolvimento e quais riscos mais significativos (Martins, 1999). Um incremento não é necessariamente a adição de código à iteração em andamento, especialmente nos primeiros ciclo de desenvolvimento, o incremento pode significar substituir um projeto superficial por um mais detalhado ou sofisticado. Em fases avançadas os incrementos são tipicamente aditivos.

A cada iteração identifica-se casos de uso relevantes, então se cria um projeto utilizando a arquitetura escolhida, implementa-se o projeto em componentes e após a implementação é

verificado se os componentes satisfazem os casos de uso. Se a iteração atinge seus objetivos, o desenvolvimento prossegue com o planejamento da próxima iteração, caso contrário, os desenvolvedores devem rever suas decisões e tentar uma nova abordagem.

Alguns do benefícios de adotar um processo iterativo são destacados na Quadro 01

Benefícios		Descrição
Redução dos riscos envolvendo custos a um único incremento		Se os desenvolvedores precisarem repetir a iteração, a organização perde somente o esforço mal direcionado de uma iteração, não o valor de um produto inteiro.
Redução do risco de lançar o projeto no mercado fora da data planejada		Identificando os riscos numa fase inicial o esforço despendido para gerenciá-los ocorre cedo, quando as pessoas estão sob menos pressão do que numa fase final de projeto.
Aceleração do tempo de desenvolvimento do projeto como um todo		Porque os desenvolvedores trabalham de maneira mais eficiente quando buscam resultados de escopo pequeno e claro
Reconhecimento de uma realidade freqüentemente ignorada		As necessidades dos usuários e os requisitos correspondentes não podem ser totalmente definidos no início do processo. Eles são tipicamente refinados em sucessivas iterações. Este modelo de operação facilita a adaptação a mudanças de requisitos

Quadro 01 – Quadro de Benefícios de um processo Iterativo

Fonte adaptada de Capra 2003

Segundo Andrade (2003), o RUP consegue incorporar o controle de qualidade e o gerenciamento de riscos contínuos e objetivos, pois a avaliação da qualidade é inserida no processo em todas as atividades. O Gerenciamento de riscos é inserido no processo de forma que os riscos que se opõem ao sucesso do projeto sejam identificados e atacados no início do processo de desenvolvimento. Além disso, O RUP proporciona uma arquitetura robusta minimizando o re-trabalho e aumentando a reutilização de componentes e a capacidade de manutenção do sistema.

Para atingir esta robustez, os desenvolvedores devem trabalhar na compreensão das funções chaves do sistema, isto é, dos casos de uso chaves. Estes devem ficar em torno de 5 a 10% de todos os casos de uso, mas são os mais significativos, aqueles que constituem o núcleo das funções do sistema (Martins, 1999). Em termos simplificados, os desenvolvedores criam um esboço da arquitetura iniciando com a parte que não é específica dos casos de uso, exemplo a plataforma. Depois se trabalha com um subconjunto dos casos de uso identificados como

funções chave do sistema em desenvolvimento. Cada caso de uso selecionado é especificado em detalhes e construído em termos de subsistemas, classes e componentes. À medida que os casos de uso são especificados e atingem maturidade, mais detalhes da arquitetura são descobertos, isto, por sua vez, leva ao surgimento de mais casos de uso. Este processo continua até que a arquitetura seja considerada estável.

Por Kruchten (*apud* OSTERWEIL 2001) os "processos de software são softwares também". O RUP se diferencia das metodologias tradicionais de desenvolvimento, pois ele é projetado, entregue e mantido como uma ferramenta de software.

O RUP é estruturado contendo cinco elementos principais, sendo eles papéis, atividades, artefatos, fluxos de trabalho e disciplinas. Papel é a definição do comportamento e das responsabilidades de um indivíduo ou grupo numa ação em equipe. Os papéis não são indivíduos e nem cargos ou funções. Um indivíduo pode ter vários papéis. Alguns exemplos de papéis são analista de sistema, assumindo o levantamento dos requisitos e a modelagem dos casos de uso e estabelecendo o escopo do sistema, projetista definindo responsabilidades, operações, atributos, relacionamentos de uma ou mais classes e como tais classes devem ser ajustadas para serem implementadas no ambiente e outros (Capra 2003).

Uma atividade é uma unidade de trabalho que um indivíduo executa quando está exercendo um determinado papel (Kruchten, 2003). Cada atividade dependendo de sua complexidade pode ser dividida em passos, algumas atividades são planejar uma iteração, encontrar casos de uso e atores, executar um teste de performance entre outras.

Segundo Kruchten (2003), uma disciplina é uma coleção de atividades relacionadas que fazem parte de um mesmo contexto. As disciplinas proporcionam um melhor entendimento do projeto sob o ponto de vista tradicional, tornando a compreensão de cada atividade mais fácil, porém dificulta mais o planejamento das atividades.

Conforme Ribeiro (2005), um artefato é um trecho de informação que é produzido, modificado ou utilizado em um dado processo. Os artefatos são produtos de um projeto e são produzidos durante o desenvolvimento do projeto. Alguns artefatos são as entradas de

atividades e outros são produzidos como saída. Exemplos de artefatos são executáveis, código fonte, documentos e outros.

Segundo Ribeiro (2005) a atribuição de atividades, papéis e artefatos não formam um processo a não ser que exista uma seqüência estabelecida de desenvolvimento das atividades. Ainda, segundo o mesmo autor isto é fundamental para que possam ser produzidos artefatos de valor. A essa seqüência é atribuído o nome de fluxo de trabalho.

O RUP define o ciclo de desenvolvimento do projeto em quatro fases, cada uma com marcos de finalização bem definidos, a esses marcos é atribuído o nome de *milestones* (Kruchten, 2003). Os *milestones* não são atividades e não possuem duração, mais servem como indicadores de progresso do projeto, e como base para decisões para continuar, cancelar, ou mudar o rumo do projeto.

As fases dos RUP são, *Inception,* onde é determinado o escopo do desenvolvimento e levantado uma visão do produto final baseada nos principais casos de uso, que precisam ser elaborados com a precisão necessária para garantir estimativas de prazos, custos e esforços, assim a ênfase desta fase é o planejamento. É necessário levantar requisitos do sistema e preliminarmente analisá-los para decidir sobre a continuidade do desenvolvimento e a execução de um plano de projeto para o sistema a ser construído. As principais atividades se encontram no levantamento das necessidades e proposição do escopo do projeto, ou seja, como poderia ser a arquitetura de desenvolvimento para este software (Capra, 2003).

Em segunda estância é elaborado o planejamento das atividades e de recursos necessários para sua execução, neste momento é definida uma arquitetura sólida a fim de eliminar os elementos de projeto que oferecem maior risco. As atividades desta fase asseguram que os requisitos e a arquitetura estão suficientemente estáveis para prever com precisão os custos e prazos. As atividades recaem sobre a identificação e realização dos casos de uso mais críticos, e a projeção e validação da arquitetura do sistema, a esta fase é atribuído o nome de *Elaboration,* no português, elaboração.

Após o planejamento aparece a fase de construção, no inglês *Construction*. A esta fase é atribuída a atividade de implementação do software, ou seja, construção de código, testar o

sistema, gerar uma versão beta e planejar a transição. Para projetos de grande magnitude esta fase deve ser segmentada em outras várias subfases, ou iterações visando assim uma maior facilidade de gerencia (Ribeiro, 2005).

Na quarta e ultima fase do processo o produto e um breve treinamento são passados ao usuário, neste momento é executada uma avaliação do produto, um *beta-testing*, a fim de colher um *feedback* para a equipe de desenvolvimento e gerencia. Naturalmente nesta fase do processo surgem novas considerações que vão demandar a construção de novas versões para permitir ajustes do sistema, corrigir problemas ou concluir algumas características que foram postergadas. A esta fase é atribuído o nome de *Transition*, traduzindo ao português, transição. Este ciclo esta representado na Figura 04

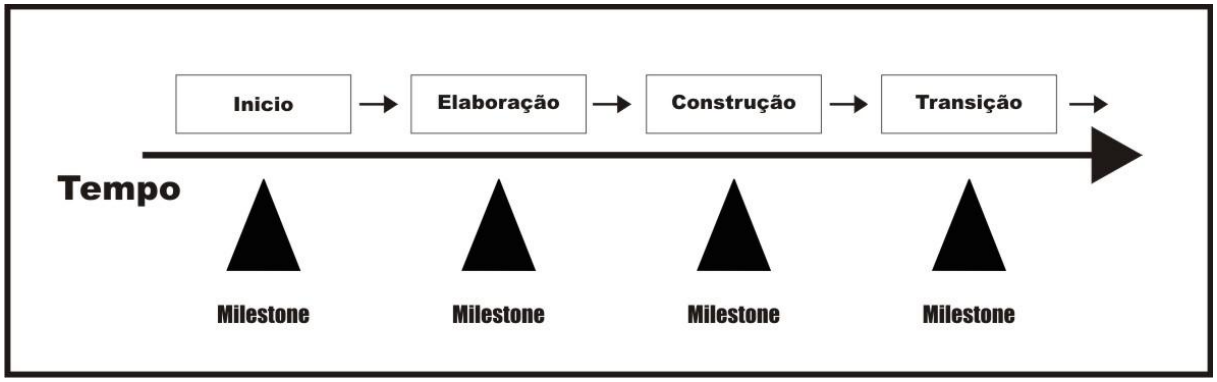


Figura 03 – O ciclo de vida do RUP

Cada uma dessas fases tem uma ou mais iterações, com foco em produzir os produtos e documentos necessários para alcançar os objetivos da fase (Kroll, 2001).

Como dito anteriormente, atividade é a unidade de trabalho realizado por um papel, usando artefatos de entrada e produzindo artefatos de saída. Na metodologia é possível encontrar trinta papéis, que definem as responsabilidades do individuo ou de uma equipe. Estes papeis são organizados em nove disciplinas, dependendo de suas características.

São disciplinas do RUP, a modelagem de negócio onde o objetivo é a compreensão da estrutura onde o software será aplicado e quais problemas a solução será capaz de atender. Em seguida a disciplina de analise de requisitos, o RUP aborda essa fase do processo com casos

de uso de negócio, traduzindo assim as necessidades de sistema e suas interfaces com o cliente.

A disciplina consequente a análise de requisitos é a fase de projeto, em que os casos de uso são especificados de forma clara e precisa já se pensando na arquitetura do sistema. Após esta especificação são implementados os objetos na forma de componentes e estes são testados individualmente a esta disciplina é atribuída o nome de implementação.

Após os testes individuais a solução é submetida a um teste integrado na quinta disciplina a fim de prover *feedback* à gerência de projeto sobre a qualidade do software, esta disciplina é denominada teste. *Deployment* é a "disciplina posterior e tem como objetivo a distribuição, instalação e teste em campo, provendo treinamento e possíveis migrações entre o sistema anterior e o atual" (Vasco, 2006).

Sequencialmente a disciplina de configuração e controle de mudanças com o objetivo de garantir a rastreabilidade de versões dentro do projeto, através de uma política de desenvolvimento e com suporte de ferramentas específicas. Esta disciplina se faz importante, pois, é possível mapear alterações efetuadas bem como recuperar códigos e versões antigas. E a ultima disciplina do processo RUP, denominada ambiente com o intuito de prover suporte à organização do projeto, em quesitos como ferramentas, métodos e processos.

A disciplina de gerência de projeto, que tem como objetivo o gerenciamento dos riscos do projeto a fim de prover formas para entrega do produto para o cliente, é considerada uma constante durante todo o processo. Isto fica mais claro se observado a Figura 05.

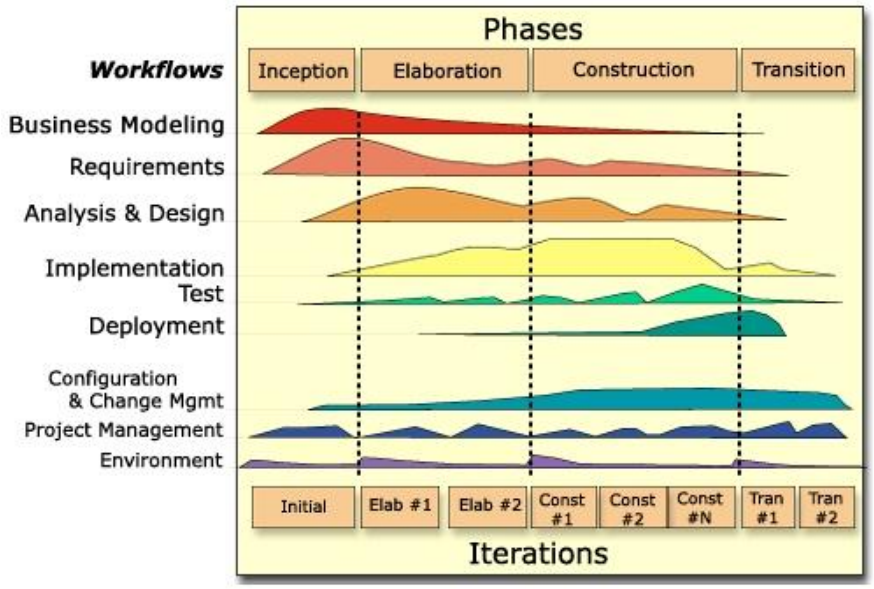


Figura 04 – Rational Ufinied Process

Fonte [era.nih.govdocsrup_fundamentals.htm]

Sendo assim afirma-se que o RUP estrutura o projeto em divisões bem definidas de atividades e define artefatos como produtos de entrada e de saída de processos. Essa estruturação permite a utilização do RUP em projetos grandes, e com distribuição geográfica, sendo adicionado um custo de gerência ao projeto.

Justificativas para se usar o RUP são evidentes, entre estas pode-se dizer que a *Rational Software,* hoje uma subdivisão da IBM, lança atualizações periodicamente deixando assim o RUP sempre atualizado, além de sua entrega ser realizada pela WEB facilitando muito o acesso dos desenvolvedores, ele é um processo genérico, ou seja, adaptável e configurável a diversas realidades atendendo assim necessidades específicas de distintos projetos.

2.2 Gerenciamento da Qualidade

A avaliação da qualidade surgiu a partir da indústria, onde a preocupação, a princípio era somente com o controle da qualidade dos produtos manufaturados. Hoje em dia, sabe-se, entretanto que a qualidade é essencial também para o setor de serviços, seja para empresas que trabalham exclusivamente na prestação de serviços, ou mesmo na inter-relação entre as áreas funcionais dentro de uma organização.

Segundo Juran (1991) “a qualidade consiste nas características do produto que vão de encontro das necessidades dos clientes e, dessa forma, proporcionam a satisfação em relação ao produto” e por Deming (1990) “qualidade é a perseguição às necessidades dos clientes e homogeneidade dos resultados do processo. A qualidade deve visar às necessidades do usuário, presentes e futuras”.

O uso de ferramentas e metodologias dão suporte à gestão de qualidade, pode-se dizer que são ferramentas de suporte, pois partes delas não foram desenvolvidas para tal finalidade, mas contribuem significativamente para a obtenção de qualidade nos produtos e serviços e também como fonte de informação para a gestão.

2.2.1 QFD

O controle de qualidade garante que as atividades de um programa ocorram conforme planejado. As atividades de controle da qualidade também poderão descobrir falhas no projeto e, assim, indicar mudanças que poderiam melhorar a qualidade. Para que exista esse eficiente controle da qualidade é fundamental que sejam utilizadas ferramentas de suporte a qualidade.

QFD é uma metodologia de planejamento estratégico, que está baseada na busca e posterior tradução dos desejos e necessidades dos clientes em características mensuráveis de produtos e serviços, assegurando que tais requisitos sejam incorporados pelas funções da empresa, promovendo a realização de produtos ou serviços superiores (Guimarães, 2003).

No desenvolvimento de produtos e serviços, o QFD surge como importante ferramenta apoiando a qualidade ainda na fase de requisitos. Este método foi concebido para operacionalizar o processo de planejamento da qualidade na forma de uma série de relações causa e efeito, operacionalizadas por meio de matrizes.

Algumas razões para usar o QFD são melhores projetos, melhor nível de satisfação de clientes, uma ferramenta eficaz para comunicação e coordenação interfuncional, redução do ciclo de desenvolvimento de produto e melhoria de produtos e serviços.

Segundo Favaretto (2007), o QFD fornece dados sólidos para ajudar as empresas a priorizar o uso de recursos escassos, além de melhorar o entendimento da problemática proposta criando um conhecimento de tendências e desejos não declarados pelos clientes. Outro ganho significativo é a redução ou eliminação na subjetividade da voz do cliente, possibilitando uma tradução fiel em características objetivas e mensuráveis de produtos, serviços e processos.

Em cada ciclo de QFD, relacionam-se as necessidades da qualidade com os requisitos da qualidade, identificando, na matriz de relações, a intensidade do relacionamento entre eles por meio de "símbolos de relações". Cada símbolo tem um peso numérico representando esta intensidade. A importância relativa é uma classificação de cada necessidade da qualidade. Essas necessidades são ponderadas segundo o grau de importância para o cliente, atribuindo a cada uma um valor numérico.

Com o uso desta ferramenta ocorre uma inversão do paradigma qualidade, deixando de ser qualidade inspecionada para qualidade desenhada (Favaretto, 2007). No Quadro 02 é apresentada uma comparação entre o uso do QFD com o método tradicional de desenvolvimento.

QFD	Desenvolvimento Tradicional
Desenvolvimento simultâneo	Seqüencial, desenvolvimento iterativo
Todas as funções participam desde o Início	As funções são envolvidas por fases
Autonomia do time para a tomada de Decisões	Aprovação pela hierarquia após cada fase
Tarefas partilhadas	Tarefas atribuídas especificamente a cada Função
Decisão por consenso	Decisões tomadas por funções
Reuniões de trabalho para desenvolver os resultados conjuntamente	Apresentação dos resultados obtidos

Quadro 02 – Quadro comparativo entre QFD e abordagem Tradicional

Fonte Costa, 2002

Para que a aplicação do QFD seja realizada de forma eficaz é importante que primeiro seja identificado qual a real necessidade dos clientes. A essa necessidade é atribuída o nome de requisito. A busca por tais requisitos precisa ser realizada de forma coesa pois servirá de base para toda a diagramação da metodologia QFD. É nesta fase que aparece o conceito "Voz do Cliente" que pode ser traduzido como a reprodução fiel do que os clientes dizem ou escrevem sobre suas necessidades (Guimarães, 2003).

O primeiro passo é definir claramente qual o mercado que se quer atender e quais os clientes objetivo para o produto ou serviço a ser desenvolvido. Baseado nesta definição deverá ser adotado um critério para seleção de alguns clientes como amostra para a busca da voz do cliente. Com a amostra de clientes já escolhida é a hora de definir o escopo do projeto. Um escopo definido claramente pode evitar problemas tais como criar falsas expectativas, mascaramento de resultados, falsos indicadores entre outros.

Com a definição do escopo, é chegado o momento de se construir uma lista detalhada com os dados dos clientes que serão contemplados na pesquisa de busca de requisitos. Esta lista deve conter além dos dados genéricos de cada cliente, a data e o método de abordagem adotado.

Independentemente da escolha da abordagem, é importante que as questões sejam construídas de forma a extrair as respostas mais significativas, tornando assim a metodologia QFD mais eficaz. Entre varias formas de abordagens pode-se citar grupos criativos, entrevista de clientes, grupo interativos, mala direta e outros (Guimarães, 2003).

Após a aplicação da abordagem escolhida é o momento de analisar os dados recolhidos a fim de se extrair os desejos e necessidades dos clientes. Identificados quais são os requisitos os mesmos devem ser agrupados por categorias e a cada grupo de requisito agrupado deve ser atribuído um cabeçalho.

Nesse próximo momento é novamente utilizado o conceito “voz do cliente”, pois novamente é executada uma sessão de pesquisa aos clientes a fim de se obter uma análise qualitativa dos dados obtidos na primeira sessão de pesquisa, em outras palavras, qual a importância daquele requisito ser satisfeito e quanto de valor ele agrega (Guimarães, 2003).

Como esta pesquisa é mais objetiva e amigável para responder do que a anterior, pois solicita que o cliente preencha somente um grau de importância a alguns graus de satisfação representados por números inteiros, a pesquisa escrita pode ser utilizada a mais vontade respeitando os devidos critérios amostrais.

Antes que se evolua para a segunda etapa da metodologia QFD que seria a implementação da casa da qualidade é recomendado por Guimarães (2003) apresentar os resultados da primeira fase e confirmar os compromissos de recursos para a etapa seguinte. Esta fase de apresentação é uma oportunidade de revisar fatores como equipe, progresso do projeto, custo entre outros. A Figura 07 sumariza os estágios da metodologia QFD.

Figura 05 – Modelo de Busca de Requisitos

Nesta segunda etapa da metodologia em posse da lista de requisitos e seus níveis de importância dar-se-a inicio a formatação sistêmica das informações obtidas em uma matriz denominada de Casa da qualidade.

A Figura 08 mostra a estrutura padrão da casa da qualidade e o preenchimento sistêmico de cada um de seus compartimentos.

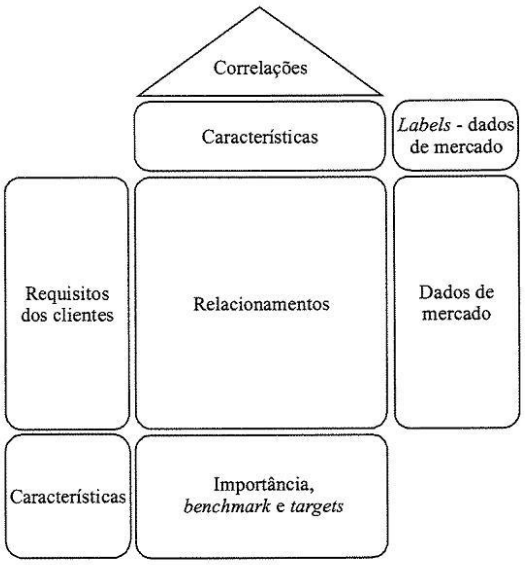


Figura 06 – Estrutura da Casa da Qualidade

Fonte Guimarães, (2003)

A Construção da casa da qualidade é o marco de um processo que denomina-se como tradução da voz do cliente, pois nesta etapa da metodologia a diagramação permite de forma visual identificar quais fatores devem ser priorizados e assim ser construída a estratégia de melhor uso dos recursos disponíveis.

O lado esquerdo da casa da qualidade é utilizado para listar os requisitos dos clientes que foram identificados durante o processo de identificação de requisitos. Cada requisito ocupa uma linha e caso existam muitos requisitos, é aconselhável que estes sejam agrupados por categorias. Na frente de cada requisito é necessário dizer qual a sua orientação, ou em outras palavras, se este requisito é melhor atendido com valores grandes ou valores pequenos, essa indicação pode ser feita através de setas ascendentes ou descendentes. É possível também que para o requisito ser melhor atendido ele precise ter um valor fixo, nestes casos em vez da indicação da orientação é utilizado um símbolo padronizado, como por exemplo um alvo (Miguel, 2001).

As colunas na parte superior da casa da qualidade são utilizadas para demonstrar quais características técnicas satisfazem os requisitos dos clientes. O Objetivo é apontar pelo menos uma característica técnica que atenda cada requisitos do cliente. Nas colunas superiores é

mantido o formato de orientação utilizado nos requisitos, ou seja, deve-se informar se a busca é pela ascendência ou descendência da característica.

Tendo requisitos e características técnicas dispostos na casa da qualidade é preciso representar quais os relacionamentos e correlações que surgirão entre cada requisito com as qualidades técnicas propostas. Esta representação é feita através de símbolos previamente definidos pela equipe que esta utilizando a metodologia QFD. Alguns símbolos podem representar valores numéricos implícitos, direção ou tendências, é possível também evidenciar com cores tais relações. Alguns dos símbolos mais utilizados são mostrados na Figura 09.

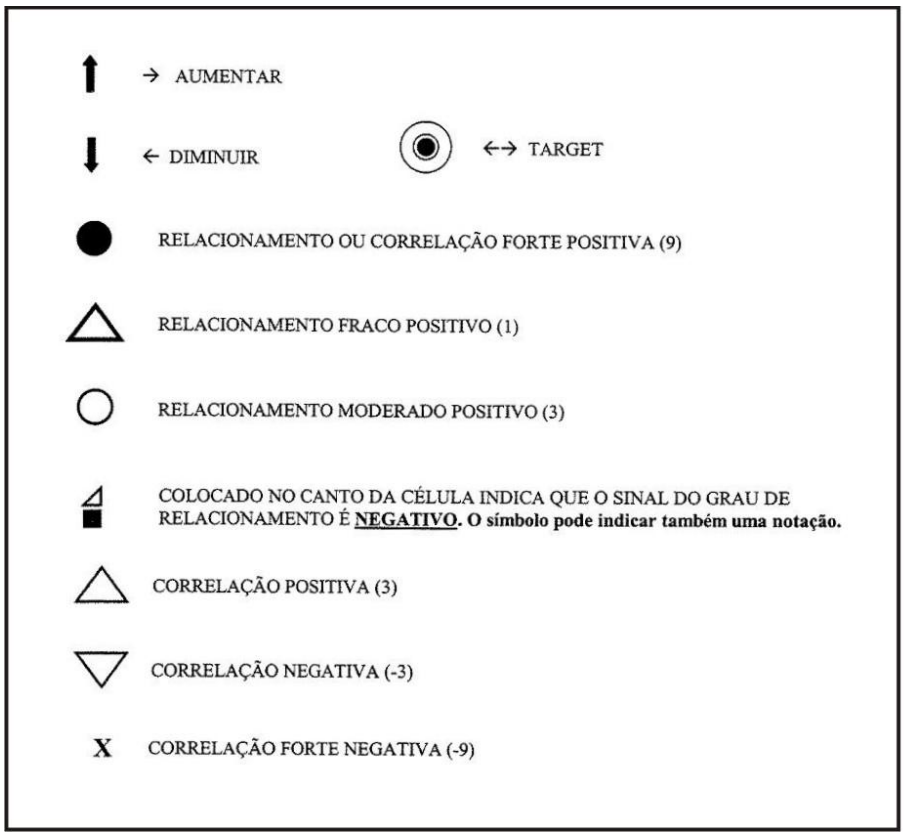


Figura 07 – Simbologia QFD

Fonte Guimarães, (2003)

A parte central da casa da qualidade identifica os relacionamentos entre os requisitos e as caracterizas propostas. Esses relacionamentos indicam a existência, intensidade e orientação das relações entre requisito e característica. Comumente haverá outros requisitos que não

aqueles que originaram tal características também influenciados, e essa influencia pode ser positiva ou negativa.

A característica deve ter sempre uma relação positiva com o requisito que a originou, independente de sua intensidade, caso isso não ocorra fica caracterizado um erro de percepção da equipe que maneja o QFD (Miguel 2001). A Figura 10 mostra os relacionamentos entre características e requisitos.

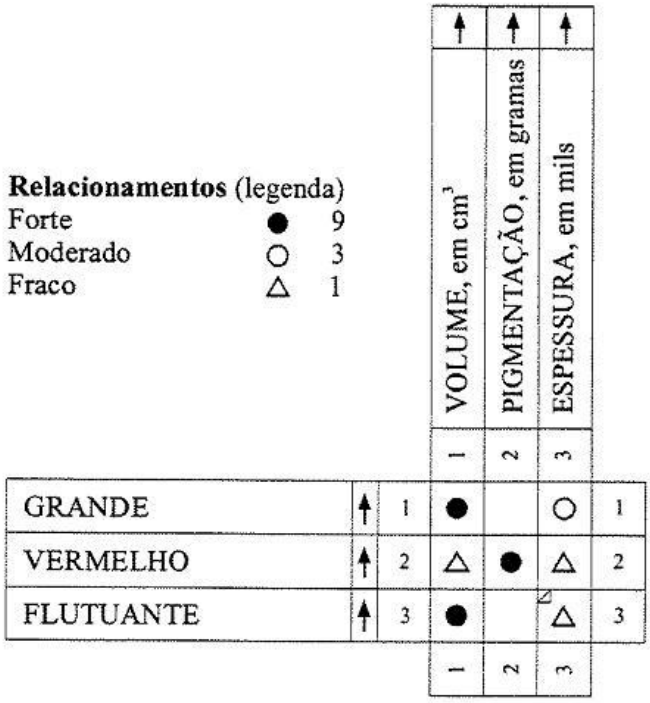


Figura 08 – Requisitos, Características e Relacionamentos

Fonte Guimarães, (2003)

Por sua vez o teto da casa da qualidade é utilizado para evidenciar relacionamento entre duas características. Do mesmo modo em que as relações entre características e requisitos, podem ser positivas ou negativas, as relações entre duas características também pode apresentar esse comportamento. Para evidenciar tais relações é comumente utilizado os mesmos símbolos utilizados na relação requisito e característica técnica. É através da identificação dessas relações que são aplicados os conceitos de trade-offs, e de breakthough´s (Guimarães, 2003).

Na parte da direita da casa da qualidade é apresentada a coluna que carrega o grau de importância de cada requisito, ou seja, sua faixa de representação, esse valor é crucial quando identificados trade-offs, ou mesmo caso exista a necessidade de uma priorização de requisitos (Guimarães, 2003). Caso o QFD seja para aperfeiçoamento de um produto já existente é possível adicionar ao lado da coluna importância, a coluna satisfação. Esse valor pode ser um forte indicador, pois se esse valor se apresenta em níveis baixos significa que é necessário atenção especial a esse requisito, pois o mesmo tem gerado insatisfação e descontentamento com o produto atual (Miguel 2001).

Ainda na lateral direita da casa da qualidade podem ser associadas colunas caracterizadas como percepções de mercado, como exemplo satisfação do concorrente que seria como o seu concorrente atende a cada requisito, plano de qualidade que é o fator teórico planejado para atingir superar o concorrente, ponto de venda que é um atributo de importância para que o marketing possa saber em qual requisito dar mais ênfase entre outros.

Usualmente todos os valores da coluna a direita são pré-definidos em um intervalo, como por exemplo, entre um e cinco.

Por fim a parte inferior da casa da qualidade traz os valores denominados dados técnicos. A Primeira linha é responsável pela importância técnica, que é o produto dos valores correspondentes ao nível de importância de cada requisito pelos valores correspondentes ao grau dos relacionamentos. Tal valor serve para evidenciar qual fator deve receber mais atenção dos projetistas, Abaixo desta linha comumente é normalizados os valores de importância a 100%, tornando assim ainda mais evidente a importância de cada característica em relação a outras.

Ainda, com dados técnicos pode-se evidenciar na casa da qualidade em sua parte inferior o valor do atributo apresentado pela empresa líder de mercado, caso o produto em questão seja produzido por outra empresa, além do valor atualmente encontrado no próprio produto e por fim qual o valor objetivo a ser atingido.

A Figura 11 ilustra a casa da qualidade após todos os passos de sua construção.

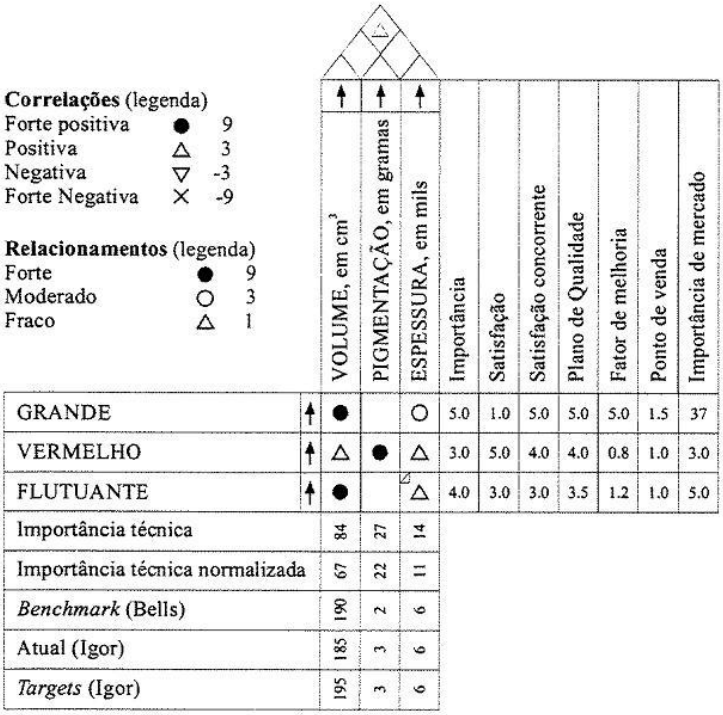


Figura 09 – Casa da Qualidade

Fonte adaptada Guimarães, (2003)

2.2.2. CICLO PDCA

Segundo Ishikawa (1985) “Se você não tem item de controle, você não gerencia”, para auxiliar o responsável a possuir um nível de gerencia mais eficiente é empregado o uso do método do ciclo PDCA.

O ciclo PDCA, ilustrado na Figura 12, nasceu no escopo da tecnologia TQC - *Total Quality Control* como uma ferramenta que melhor representava o ciclo de gerenciamento de uma atividade. Este conceito evoluiu ao longo dos anos figurando a idéia de que um objetivo a ser atingido precisa ter planejamento e controle das atividades a ele relacionado.

Figura 10 – Ciclo PDCA

O PDCA é aplicado para se atingir resultados dentro de um sistema de gestão e pode ser utilizado em qualquer empresa de forma a garantir o sucesso nos negócios, independente da área de atuação da empresa.

A primeira fase do ciclo PDCA é a fase do planejamento no inglês *PLAN*, e consiste basicamente em localizar problemas e estabelecer metas e estabelecer planos de ação. Esta é considerada a fase de estabelecimento da diretriz de controle.

A fase procedente e a fase de execução, no inglês *DO*, tradução do verbo fazer. Neste momento as atividades são executadas exatamente como planejadas na fase anterior, é imprescindível o investimento de treinamento no trabalho. Esta também é a fase responsável por executar uma coleta de dados para uma posterior verificação das metas estabelecidas.

Após a execução das tarefas é iniciada a fase de verificação, a terceira fase do ciclo PDCA no inglês *CHECK*, onde o objetivo é averiguar se os objetivos alcançados são condizentes com as metas estabelecidas na fase de planejamento.

E encerrando o ciclo a fase de atuação corretiva, no inglês *ACTION*, os pontos que divergiram de suas metas anteriormente estabelecidas, são analisados a fim de se encontrar uma solução para que os mesmos não voltem a ocorrer.

Essas fases e seus fluxos são mostrados na Figura 13.

FASE	FLUXO	ETAPA	OBJETIVO
P	1	Identificação do Problema	Definir claramente o problema/processo e reconhecer sua importância.
	2	Observação	Investigar as características específicas do problema/processo com uma visão ampla e sob vários pontos de vista.
	3	Análise	Descobrir a causa fundamental.
	4	Plano de ação	Conceber um plano para bloquear a causa fundamental.
D	5	Execução	Bloquear a causa fundamental.
C	6	Verificação	Verificar se o bloqueio foi efetivo.
A	7	Padronização	Prevenir contra o reaparecimento do problema.
	8	Conclusão	Recapitular todo o método de solução do problema para trabalhos futuros.

Figura 11: Fluxo e fases do PDCA

A utilização do Ciclo PDCA promove o aprendizado contínuo dos processos e também repercute positivamente na tomada de decisão da parte do gestor, pois favorece a obtenção de informações oportunas e confiáveis durante a execução do projeto.

Desta forma, o PDCA pode ser utilizado na realização de toda e qualquer atividade da organização. Sendo ideal que todos da organização utilizem esta ferramenta de gestão no dia-a-dia de suas atividades, pois elimina a cultura que incentiva a se realizar o trabalho sem antes planejar, desprezando o autocontrole, o uso de dados gerados pelas medições por indicadores e a atitude preventiva, para que os problemas dos processos nunca ocorram.

O ciclo PDCA é um ciclo de desenvolvimento que tem foco na melhoria contínua e esta melhoria contínua aperfeiçoa a execução dos processos, possibilita a redução de custos e o aumento da produtividade.

2.3 Qualidade de Software

Segundo Campos (2001) a qualidade não basta existir, ela deve ser reconhecida pelo cliente. Para facilitar esse reconhecimento existem certificações e selos de qualidade, tratando de produtos de software é possível citar as normas ISO/IEC 9126, ISO/IEC 14598 e ISO/IEC 12119 que descrevem requisitos de qualidade e mostram diretrizes para um processo avaliativo.

A qualidade de software é uma área da engenharia de software que tem por objetivo atingir a qualidade de software através de processos estabelecidos de desenvolvimento. Apesar dos modelos aplicados na garantia da qualidade de software atuarem principalmente no processo, o principal objetivo é garantir um produto final que satisfaça às expectativas do cliente, dentro daquilo que foi acordado inicialmente.

Segundo Arouck (2001) no desenvolvimento de software, a qualidade do produto está diretamente relacionada à qualidade do processo de desenvolvimento, desta forma, é comum que a busca por um software de maior qualidade passe necessariamente por uma melhoria no processo de desenvolvimento.

Esta melhora na qualidade de software e desenvolvimento eficiente são assuntos chaves desde 1990, nesta época foram criadas diferentes ferramentas para o auxilio do processo de desenvolvimento, e neste contexto o QFD foi adaptado em SQFD, permitindo assim o seu uso na indústria de software como importante ferramenta de levantamento de requisitos de sistemas.

O uso do SQFD se assemelha ao uso de seu precursor o QFD, onde os requisitos funcionais são levantados junto ao cliente e depois convertidos em afirmações técnicas e mensuráveis. Em seguida e montada uma matriz de correlação entre os requisitos levantados e as especificações técnicas para que seja identifica a intensidade de cada relacionamento, e através dos dados extraídos desta matriz e feita uma priorização dos requisitos levantados.

Com a aplicação desta ferramenta é possível identificar benefícios como ação preventiva da qualidade, a consciência e política de atender as perspectivas do usuário evitando assim trabalho com alterações após a entrega do produto, decisões melhores justificadas e menores ciclos de desenvolvimento.

Sendo assim pode-se dizer que o SQFD para garantir a qualidade do software, tem como alvo a fase de análise de requisitos. A aplicação desta ferramenta é um assunto relativamente novo e ainda não se encontra difundido entra as organizações de desenvolvimento, pois seu processo se encontra ainda com pouca maturidade.

Atuando diretamente no processo de desenvolvimento pode-se inserir o ciclo PDCA que como dito anteriormente é um método de melhora contínua da qualidade que consiste em uma seqüência lógica de quatro etapas que se repetem. Na indústria de software este se enquadra perfeitamente em diversos modelos de desenvolvimento, a exemplo disto as Figuras 14 e 15.

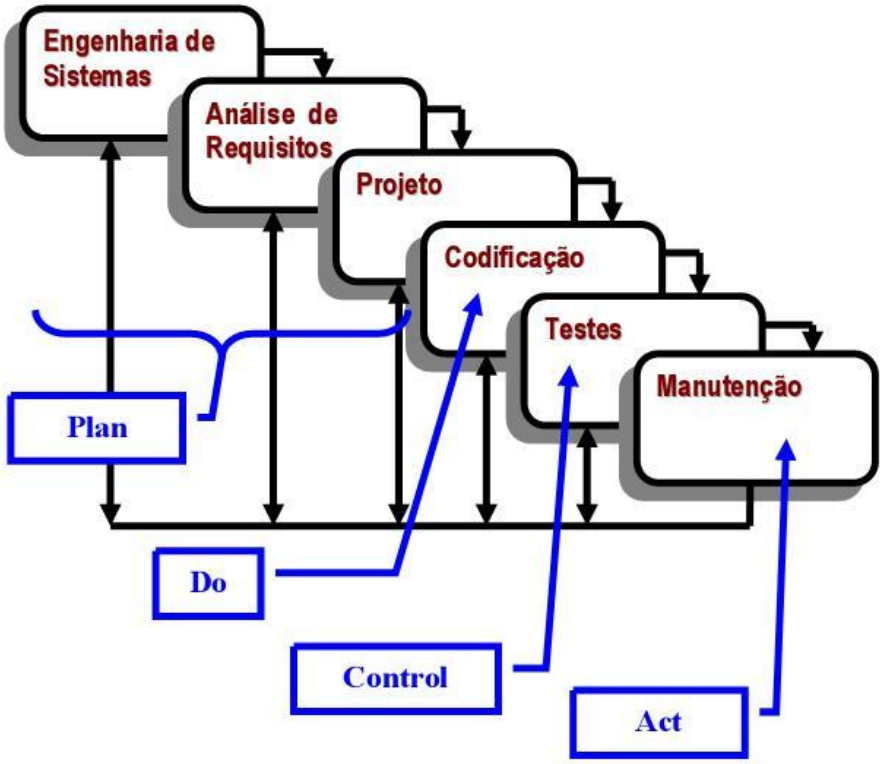


Figura 12 – Ciclo PDCA embutido no modelo cascata

Fonte http://www.er.les.inf.puc-rio.br/pes/resumos.htm

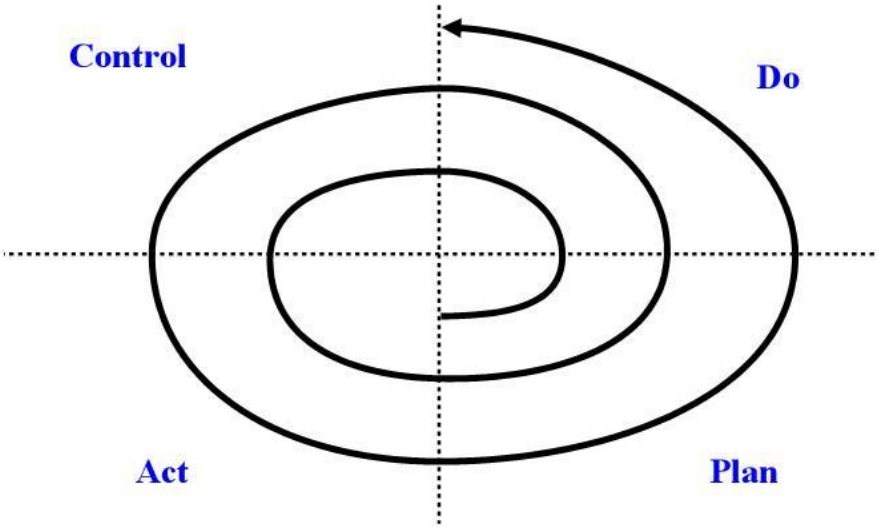


Figura 13 – Ciclo PDCA embutido no modelo espiral

Fonte http://www.er.les.inf.puc-rio.br/pes/resumos.htm

Basicamente pode-se empregar o ciclo PDCA durante qualquer fase do desenvolvimento sempre que surgir a suspeita de que algum procedimento não está atendendo as conformidades planejadas. A Figura 16 mostra de forma sistêmica o uso do ciclo PDCA.

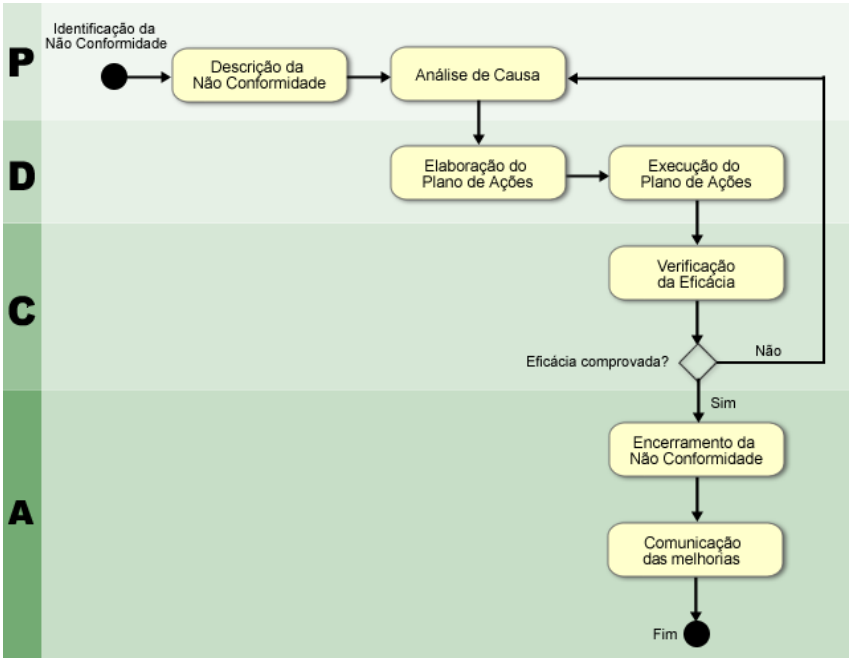


Figura 14 – Forma sistêmica do uso do ciclo PDCA

3 ABORDAGEM PROPOSTA

A preocupação com a qualidade de software está se tornando cada vez maior em função do grande volume de software produzido e a exigência dos usuários que almejam antes de tudo softwares confiáveis e eficientes. Cada vez mais organizações e empresas se tornam dependentes de softwares em seus negócios. Grandes organizações tem seus modelos de negocio baseados nas informações geradas por sistemas de informação, e em grande maioria o seio do sistema de informação destas organizações é o componente software que processa, organiza e compara dados de modo que o usuário possa extrair a informação que deseja em tempo hábil.

A primeira geração de softwares foi criada, em sua maioria, sem um processo definido de desenvolvimento. Estes desenvolvimentos eram realizados por uma pessoa ou pequeno grupo com idéias afins, que idealizavam um sistema, e o desenvolviam, baseados em sua própria experiência. Porem com o passar do tempo o software foi adquirindo robustez, o que implicou numa complexidade além da capacidade criativa de uma pessoa, ou de um pequeno grupo sem técnicas adequadas de organização ou projeto de software.

Essa necessidade de robustez que o software apresenta atualmente e também o uso cada vez mais gradual de sistemas de software em atividades de alto risco financeiro e humano fazem com que o seu desenvolvimento se torne uma atividade de grau de risco. Isto implica na necessidade de construir softwares confiáveis. Neste cenário, é apontada a primeira grande evidencia da necessidade de um processo eficiente de desenvolvimento.

Conforme Kruchtem (2003), um produto para obter qualidade, além de ausência de defeitos deve atender aos propósitos desejados, ou seja, deve fazer o que as pessoas querem que ele faça, pois se o produto final é livre de defeitos, mas não atende aquilo para o qual foi construído este é considerado desnecessário e sem qualidade. O compromisso de funcionar de acordo como foi especificado e de detectar problemas que eventualmente não tenham sido previstos em suas especificações é a segunda evidencia da necessidade de um processo eficiente de desenvolvimento para qualquer produto.

As metodologias foram criadas com o intuito de suprir tais deficiências, garantindo qualidade no desenvolvimento e manutenção de seus sistemas, bem como o desenvolvimento em menos tempo de aplicações mais complexas e custo. Além de beneficiar na qualidade de desenvolvimento e manutenção, a utilização de metodologias poupa, entre muitas outras coisas, re-trabalho e tempo à equipe responsável pelo seu desenvolvimento.

Considerando que softwares são construídos baseados nestas metodologias, a sua qualidade não pode ser avaliada de maneira isolada, é fundamental que a metodologia também apresente determinado grau de qualidade para que o objetivo final seja atingido. Um modelo pobre ou a ausência de uma metodologia pode ser apontada como uma provável causa de baixa qualidade de um software.

Como dito no capitulo dois deste trabalho o *Rational Unified Process* – RUP é um processo de engenharia de software criado pela Rational Software Corporation, que dita técnicas a serem obedecidas por cada membro de uma equipe de desenvolvimento vislumbrando o aumento da produtividade (Kruchtem, 2003).

Este trabalho pretende demonstrar como as ferramentas de qualidade podem apoiar o processo de construção de softwares, através de uma conceituada metodologia de desenvolvimento de sistemas *Rational Unifed Process* – RUP. Deste modo é proposto que seja customizada uma instancia do RUP e nesta instancia seja aplicadas as ferramentas de qualidade em diversas fases a fim de atingir melhores práticas de software.

A escolha desta metodologia se faz pelo fato do RUP ser totalmente customizável, se tornando assim compatível com diversas magnitudes de projetos. A adaptação do RUP se faz necessária pelo fato desta ser uma metodologia complexa voltada mais para grandes projetos, e como o escopo deste trabalho é o uso de ferramentas de qualidade afim de atingir melhores práticas de software, não evidencia a necessidade de ser instanciado todo o RUP.

Um dos principais fatores para obter o equilíbrio entre produzir software de qualidade e liberá-lo com rapidez é compreender os elementos essenciais do processo e seguir algumas

diretrizes para sua adaptação, a fim de satisfazer, da melhor maneira possível, as necessidades específicas do projeto.

Uma das principais características da maioria dos projetos pequenos é um nível menor de formalidade. Embora existam exceções, quanto maior for o número de pessoas no projeto e quanto maior e mais complexo for o produto, maior será a necessidade de um processo formal. Da mesma forma, um sistema de direcionamento de mísseis requer artefatos mais formais que a atualização de um sistema de inventário.

A metodologia de desenvolvimento será orientada à constante interação com os membros interessados, tanto na coleta de informações como na prestação de contas com relação ao andamento do projeto. Com relação ao ciclo de vida do desenvolvimento, sugerido pelo RUP será composto pelas fases de iniciação, elaboração, construção e transição e durante cada uma das fases, ocorrerá iteração com as disciplinas do RUP. Neste trabalho pela customização da metodologia serão abordadas somente as disciplinas de requisitos, analise, projeto e implementação. Cada uma das fases do ciclo de vida será evidenciada de forma mais ou menos intensa dependendo de qual disciplina o desenvolvimento se encontra. A Figura 17 mostra a cadencia do abordagem proposta.

Figura 15 – Fluxo das disciplinas propostas

Durante a disciplina de requisitos o objetivo é atingir o consenso com os clientes e outros envolvidos sobre o que o sistema deve fazer e oferecer aos desenvolvedores uma melhor compreensão dos requisitos do sistema. Nesta disciplina para atingir os objetivos propostos é importante antes de tudo, compreender a definição e o escopo do problema que se tenta resolver com o sistema. Artefatos gerados na disciplina anterior de modelagem de negocio aparecem como importantes elementos de informação nessa etapa. Por este trabalho não abordar a disciplina de modelagem de negócio, alguns desses artefatos serão concebidos e melhor detalhados dentro desta mesma disciplina.

Na customização proposta da disciplina de requisitos será feito o uso de dois papéis, sendo o papel de analista de sistema e o papel de especificador de requisitos. O papel analista de sistemas lidera e coordena a identificação de requisitos e a modelagem de casos de uso, delimitando o sistema e definindo sua funcionalidade e o papel de especificador de requisitos detalha a especificação de uma parte da funcionalidade do sistema, descrevendo o aspecto requisito de um ou de vários casos de uso e outros requisitos de software de apoio.

Como principais artefatos desta disciplina têm-se o documento de regra de negócios, abordando questões políticas ou condições que devem ser satisfeitas, uma descrição textual do sistema, digramas de visão do sistema, modelo de objetos de negócio, modelo de casos de uso e sua descrição e uma sintética descrição da arquitetura. Como atividade desta disciplina, importante salientar a atividade de formular o escopo do projeto, que envolve captura do contexto, bem como os requisitos e as restrições mais importantes, para assim desprender critérios de aceitação para o produto final.

Outras atividades desta disciplina são identificar as necessidades dos principais envolvidos, localizar atores e casos de uso que são desenvolvidas pelo papel de analista de sistema e as atividade de detalhar os casos de uso e requisitos, do papel de especificar de requisitos que acumulara a atividade de priorizar os casos de usos que usualmente fica a critério de um engenheiro de software. Os papéis desta disciplina são ilustrados na Figura 18.

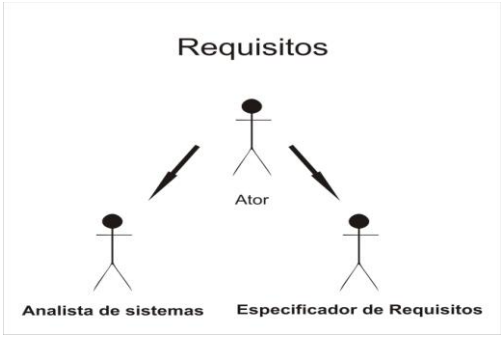


Figura 16 – Papéis da disciplina de requisitos

No estudo aqui proposto, a cada disciplina, será disparado o uso do ciclo PDCA como ferramenta de apoio ao desenvolvimento e melhoria continua de processo. Sendo assim a cada inicio de uma disciplina será concebido um novo artefato, que será um plano de ação. Este

será usado durante a terceira fase do ciclo de vida de desenvolvimento da disciplina a fim de confronto do obtido com o esperado para planejar as ações preventivas para a próxima iteração.

Para a elaboração do plano de ação é fundamental saber o que se espera fazer, ou seja, identificar e relacionar as atividades. E escolher uma técnica compatível com a complexidade do plano de ação para servir de estrutura para a sua construção. Nesta abordagem será utilizado uma simplificação da metodologia 5W 2H para a construção do plano de ação. Um modelo de plano de ação é disponibilizado em anexo a este trabalho.

E também na disciplina de requisitos, será utilizada a ferramenta QFD para captura dos requisitos e de suas respectivas prioridades diante do produto final. O uso desta ferramenta se da por a aplicação de um questionário a um grupo de usuários e através do retorno desse questionário é construída a casa da qualidade. O documento contendo a casa da qualidade servirá de forma gráfica e organizada como guia, para se fazer bom uso dos recursos disponíveis baseando-se nas prioridades identificadas pelos usuários. E este documento será mais um artefato que será incorporado ao processo de desenvolvimento proposto neste trabalho. A aplicação desta metodologia é realizada pelo papel de analista de sistema durante a atividade de identificação das solicitações dos principais envolvidos e a forma de condução é melhor detalhada no capitulo dois deste trabalho.

Baseado na magnitude do projeto que servirá de suporte para avaliação dos resultados esperados, e no fato da equipe de desenvolvimento ser composta de apenas um elemento executando todos os papéis, a customização da disciplina de requisitos se restringirá aos artefatos acima citados, produtos da aplicação do ciclo PDCA gerando um plano de ações e o documento contendo a casa da qualidade obtida da analise dos questionários QFD e alguns dos artefatos tradicionais desta disciplina.

Ao fim das atividades da disciplina é verificado junto ao plano de ação se o resultado obtido e o resultado planejado são coerentes, dando assim continuidade ao processo de desenvolvimento pela disciplina de análise. Caso isto não ocorra é necessário identificar a

causa e agir de forma a eliminá-la para próxima iteração. Na Figura 19 é apresentada uma ilustração geral da disciplina de requisitos.

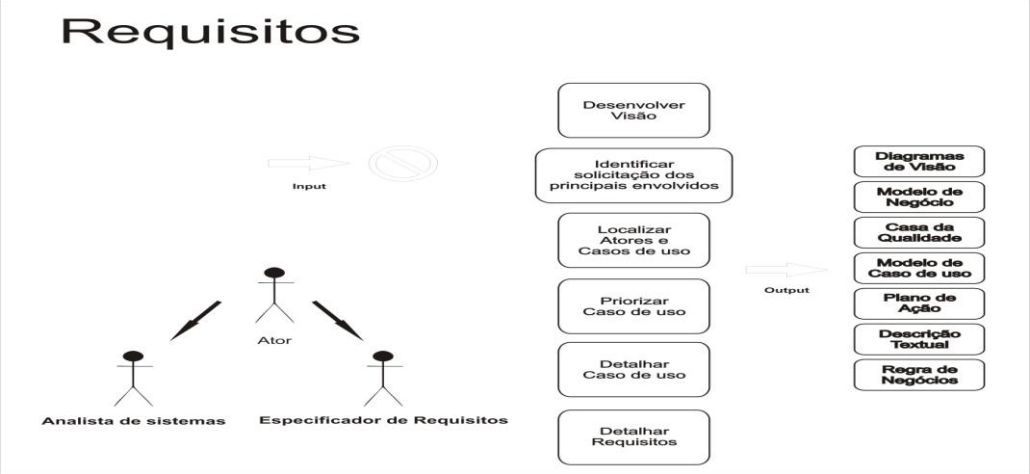


Figura 17 – Disciplina de requisitos

A disciplina análise obtém as informações de modelo de casos de uso e o Glossário dos artefatos gerados na fase requisitos. Falhas no modelo de casos de uso podem ser descobertas durante a atividade de análise e assim solicitações de mudança podem então serem geradas e aplicadas ao modelo de casos de uso.

As atividades de planejar e preparar um caso de negócio e avaliar alternativas para o gerenciamento de riscos, sintetizar uma possível arquitetura, para que seja possível estimar custo, programação e recursos são as principais atividades da disciplina de análise e serão desenvolvidas pelo papel de arquiteto de software. Este papel estabelece a estrutura geral de cada visão de arquitetura, a decomposição da visão, o agrupamento dos elementos e as interfaces entre esses principais agrupamentos. Portanto, comparado aos outros papéis, a visão do arquiteto de software é ampla, e não detalhada.

A arquitetura se desenvolve a partir de um exame dos requisitos mais significativos e de uma avaliação de risco. A estabilidade da arquitetura é avaliada através de um ou mais protótipos de arquitetura.

A Atividade de modelagem de dados fica a cargo do papel de designer de banco de dados. Este papel define tabelas, índices, visões, restrições, triggers, procedimentos armazenados e outras construções específicas de um banco de dados necessárias para armazenar, recuperar e excluir objetos persistentes. Essas informações são mantidas no artefato modelo de dados.

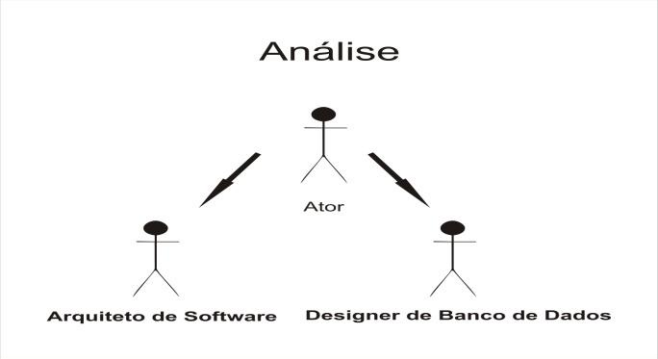


Figura 18 – Papéis da disciplina de análise

Como objetivos secundários desta fase têm-se, assegurar que a arquitetura, os requisitos e os planos sejam estáveis o suficiente e que os riscos sejam suficientemente diminuídos a fim de que se possa na disciplina de projeto determinar com segurança o custo e a programação para a conclusão do desenvolvimento, tratar todos os riscos significativos do ponto de vista da arquitetura do projeto e produzir se necessário um protótipo evolutivo, assim como um ou mais protótipos descartáveis para diminuir riscos específicos de design, requisitos e outros.

O artefato protótipo serve como apoio ao engenheiro de software e ao gerente de projeto como uma estratégia de redução de riscos. Esses protótipos podem ser puramente exploratórios e posteriormente descartados, como em casos que a arquitetura tenha sido construída como uma série de protótipos evolucionários.

E assim demonstrar que a arquitetura suportará os requisitos do sistema a um custo e tempo justos e tambem pré-estabelecer um ambiente de suporte, pois para atingir esses objetivos básicos é importante pré- configurar o ambiente de suporte para o projeto. Isso inclui criar um caso de desenvolvimento, criar templates e diretrizes, e configurar ferramentas. A disciplina de analise é ilustrada na Figura 21

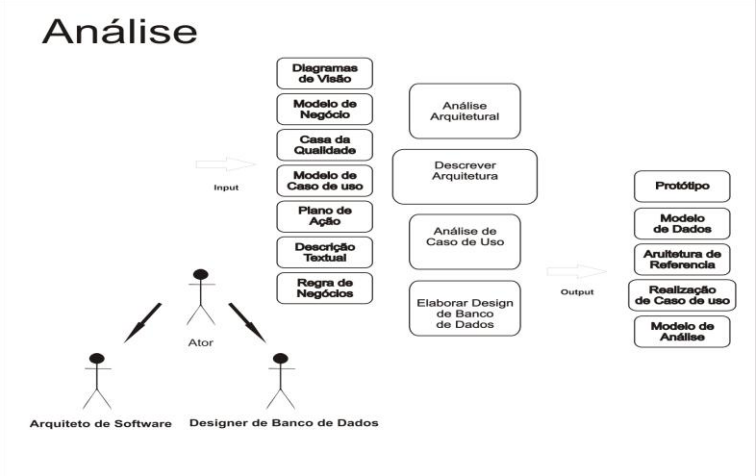


Figura 19 – Disciplina de análise

Para a fase de análise será proposto à mesma abordagem da fase anterior, ou seja, o uso de uma customização reduzindo assim o seu número de atividades e artefatos e o uso do ciclo PDCA como ferramenta de melhoria continua. Os artefatos selecionados para esta disciplina são documento de descrição da arquitetura, casos de uso de análise – *Realization,* o diagrama de classes global e por caso de uso, o modelo de dados e o protótipo.

Ao final das atividades desta disciplina novamente será confrontado o plano de ações obtido pela primeira etapa do ciclo PDCA com o resultado obtido, tendo assim um excelente modo de averiguar se o resultado obtido é o esperado para prosseguir a próxima disciplina ou se faz necessário, ajustes para a próxima iteração.

De posse dos artefatos esperados é o momento de projetar o sistema a ser concebido, a essa disciplina é dado o nome de projeto. Abordagens como a organização da equipe, o plano do projeto e as mudanças de custo/programação/lucros, e a preparação do ambiente para o projeto avaliando o projeto e a organização, selecionando ferramentas e decidindo quais partes do processo devem ser melhoradas.

As atividades desta fase se apresentam em torno de definir e validar a baseline da arquitetura de forma rápida e eficiente, submeter à visão a um processo de refinamento a fim de atingir uma visão mais sólida dos casos de uso críticos e criar planos de iteração detalhados e

baselines para a fase de implementação. Esta disciplina é conduzida pelo papel do gerente de projeto, esta pode ser um ou vários atores, dependendo da complexidade do projeto. Este papel aloca recursos, ajusta as prioridades, coordena interações com clientes e usuários e geralmente mantém a equipe do projeto concentrada na meta certa. A Figura 22 ilustra os papel de gerente de projeto.

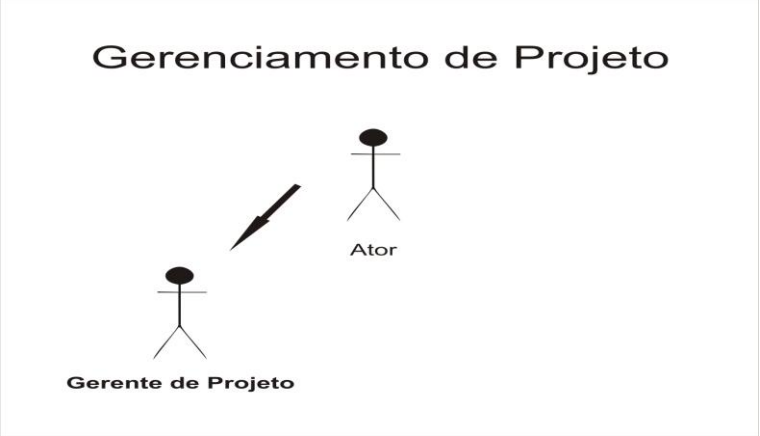


Figura 20 – Papéis da disciplina de Gerenciamento de Projeto

Ao final desta disciplina é esperado que se atinja uma baseline gerenciada para a arquitetura do sistema que permita o escalonamento da equipe do projeto na fase de construção. Examinam-se então novamente os objetivos e o escopo detalhado do sistema e a opção de arquitetura definida seguindo critérios como estabilidade de visão e requisitos integrados a arquitetura. E também a garantia se os planos de iteração para a fase de construção são garantidos por estimativas confiáveis e se a despesa real em oposição à despesa planejada com recursos é aceitável.

Considerando os fatores do projeto de estudo, a disciplina de projeto sofrerá uma simplificação, tornando assim o processo menos formal e menos complexo. Os artefatos dessa fase serão plano de desenvolvimento de software, plano de iteração e plano de resolução de problemas. A Figura 23 ilustra a disciplina de gerenciamento de projeto.

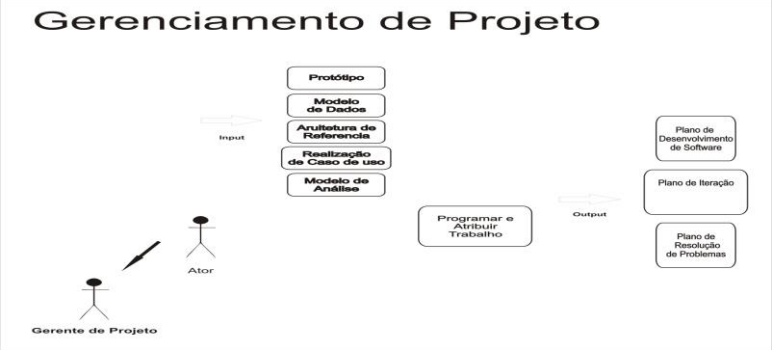


Figura 21 – Disciplina de gerenciamento de projeto

Com o ambiente de desenvolvimento ja preparado é visto então o momento da construção do sistema de fato, neste momento é importante esclarecer os requisitos restantes do sistema com base na arquitetura da baseline. Neste momento de certa forma o processo se assemelha a uma manufatura, onde a ênfase está no gerenciamento de recursos e controle de operações para otimizar custos, programações e qualidade. O Papel que atua nesta fase recebe o nome de implementador, e este é responsável por desenvolver e testar componentes de acordo com os padrões adotados para o projeto, para fins de integração com subsistemas maiores. Os papéis da disciplina de implementação são ilustrados na Figura 24

Figura 22 – Papéis da disciplina de implementação

Visa-se nessa disciplina atingir objetivos como minimizar os custos de desenvolvimento, otimizando recursos e evitando retalhamento e retrabalho desnecessários, atingir a qualidade adequada com rapidez e eficiência, atingir as versões úteis (alfa, beta e outros releases de teste) com rapidez e eficiência. Para isso busca-se dentro do desenvolvimento certo paralelismo entre o trabalho das equipes de desenvolvimento. Isso pode ser notado mesmo em projetos menores, normalmente há componentes ou módulos que podem ser desenvolvidos

independentes dos outros, permitindo o paralelismo natural entre as equipes. Com isso pode-se acelerar bastante as atividades de desenvolvimento, e em contra partida aumenta-se a complexidade do gerenciamento de recursos e da sincronização dos fluxos de trabalho. Uma arquitetura sofisticada será essencial para atingir um paralelismo significativo.

Os artefatos gerados por essa disciplina são subsistemas, código fonte, componentes, executáveis e outros desta natureza. A disciplina de implementação é abordada na Figura 25.

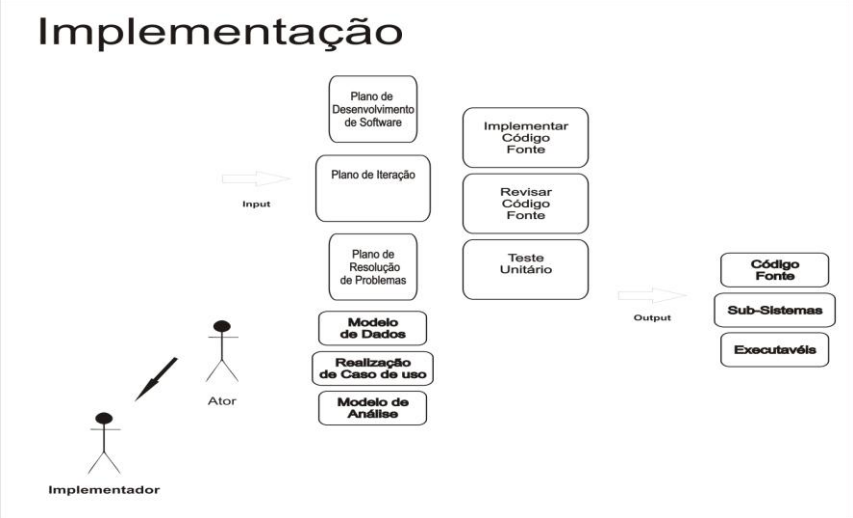


Figura 23 – Disciplina de Implementação

4 CONCLUSÃO

Atualmente a complexidade que software apresenta cresce rapidamente e a necessidade de uma metodologia é sua concepção é evidente. Apesar de muitas vezes a metodologia tornar-se algo muito burocrático e repetitivo, a mesma visa garantir o sucesso do projeto com a realização de várias atividades e artefatos relacionados com o produto do projeto. Tais atividades e artefatos são desenvolvidos de forma lógica e organizada baseado nas melhores praticas de desenvolvimento de software já conhecidas por projetos anteriores. Estes artefatos dão a cadencia ao andamento da produção do software em questão.

Porem apesar de ter uma metodologia bem definida, é importante evidenciar que o conceito de qualidade segundo alguns autores aborda o fator funcionalidade, ou seja, o se o software atende o propósito pelo qual foi gerado. Nesse ponto o uso do QFD pode trazer uma melhora significativa no entendimento e priorização das funcionalidades, fato esse que comprova sua usabilidade na indústria de softwares, como ferramenta de apoio a metodologia instanciada.

O uso do ciclo PDCA em cada fase do projeto evidencia a equipe de desenvolvimento que os esforços desprendidos estão sendo consumidos de forma correta. Isso é possível graças à comparação do plano de ações com os outputs de cada fase da metodologia instanciada. Caso seja identifica uma discrepância entre o obtido e esperado são tomadas ações de correção ainda na fase atual, para que o produto da fase não apresente nenhuma surpresa a etapa seguinte, isso serve como garantia de que cada produto consumido nas etapas de construção apresenta garantia e é possível projetar uma garantia esperada do produto final.

Baseado nos fatos acima apresentados, pode-se concluir que através da escolha da metodologia seguindo as especificações sugeridas, e com o uso das ferramentas de qualidade pode-se obter excelentes resultados no produto final em função das boas práticas. E que o uso de ferramentas clássicas de qualidade, mais voltadas para indústrias manufatureiras possuem robustez suficiente para serem incorporadas no processo de desenvolvimento de softwares, trazendo assim grandes benefícios ao produto final.

REFERÊNCIAS

ANDRADE, Rossana Maria de Castro. **Uma metodologia para aplicação de padrões de software integrada ao RUP**. 2003.

AROUCK, O. **Avaliação de sistemas de informação**, v. 13, n. 1, jan./jun., 2001.

BECK, Kent; FOWLER, Martin. **Planning Extreme Programming.** Boston: Addison-Wesley Professional, 2001.

CAPRA, Simão Pietro Mansur. **Adaptação do RUP para Projetos de Software E-Commerce.** Trabalho de Graduação – Universidade da Amazônia, 2003.

CAMPOS, F. BRAGA, R; COELHO F. **Quality process to improve agricultural software products.** Germany; FEESMA, 2000.

COSTA, Paulo Eduardo C. **Aplicação do método QFD para facilitar o desenvolvimento de sites de comércio**
Eletrônico. Trabalho de Graduação – Universidade Federal Do Rio Grande Do Norte, 2002.

DEMING, W. E. Qualidade: **A Revolução da Administração**. Editora Marques Saraiva. Rio de Janeiro, 1990.

FAVARETTO, Rodrigo Guilger - **Modelo de aplicação de QFD no Desenvolvimento de Bebidas.** Dissertação (Mestrado) – Universidade de Campinas 2007.

GUIMARAES, Leovani Marcial - **QFD - Quality Function Deployment, Uma análise de aspectos culturais organizacionais como base para definição de implementação da metodologia.** Dissertação (Mestrado) – Universidade de Campinas 2003.

ISHIKAWA, Kaoru. **TQC Total Quality Control Estratégia e Administração da Qualidade**. Rio de Janeiro, 1985.

JURAN, J. M. e Gryna, F. M. **Controle da Qualiade Handbook**. Vol. 1. São Paulo. Makron Books do Brasil Editora Ltda, 1991.

KROLL, Pen. **The Spirit of the RUP**. The Rational Edge, IBM, 2001.

KRUCHTEN, Philippe. **Introdução ao RUP Rational Unified Process.** 2. ed. Rio de Janeiro: Ciência Moderna Ltda, 2003. 247 p.

MARTINS, Vidal. **O Processo Unificado de Desenvolvimento de Software**. Companhia de Informática do Paraná – CELEPAR: Paraná, 1999. Disponível em: *http://www.pr.gov.br/celepar/celepar/batebyte/edicoes/1999/bb89/software.htm.* Acessado em: 14/09/2009.

MARRECO, Daniel Catunda. **Um Processo Controlável de Desenvolvimento de Software Focado na Gestão da Qualidade em Pequenos Projetos.** Dissertação (Mestrado) – PUC, 2006.

MIGUEL, Paulo Augusto Cauchick. **Qualidade: Enfoques e Ferramentas.** São Paulo: Artliber, 2001. 263 p.

PRESSMAN, Roger S. **Engenharia de Software.** 3. ed. São Paulo: Pearson Education do Brasil, 2005. 1028 p.

PUENTE, J.; PINO, R.; PRIORE, P.; FOUENTE, D de L. **A decision support system for applying failure mode and effects analysis**. International Journal of Quality & Reliability Management, Bradford, v. 19, n. 2, p. 137-151, 2002.

RIBEIRO, Manoel Pinheiro. **Rational Unified Process, Uma abordagem gerencial** - Dissertação (Mestrado) – Instituto Militar de Engenharia, 2005.

VASCO, Carlos G., Marcelo Henrique Vithoft, Paulo Roberto C. Estante. **Comparação entre Metodologias RUP e XP -** Pontifícia Universidade Católica do Paraná (PUCPR) – Curitiba, PR – Brasil

ANEXO I

Nome: Edwin Cardoza
Data: 2009-10-14

Aplicação do QFD no Desenvolvimento de Software

Descrição do Sistema

O sistema é um gerenciador de pesquisas, para uso acadêmico. Com esta ferramenta o acadêmico poderá criar um perfil para o mesmo, e através desse perfil cadastrar os trabalhos que ele desenvolveu ou esta desenvolvendo.
O academico através do seu perfil pode pesquisar trabalhos que sejam semelhantes ao seu desenvolvidos por outros acadêmicos que estejam cadastrados no sistema e poderá ter acesso aos materias ou informações ultilizadas por eles.
Exemplo de campos de busca podem ser, String de busca, autor, ano, tipo…

Requisitos Funcionais.

Nesse contexto imagine-se sendo um dos usúarios do sistema e informe quais o requisitos funcionais, ou seja, quais as funções você julga mais importantes e que o sistema deva contemplar com o seu respectivo valor de importância.

Requisitos	**Valor de Importância (0 a 5)**
1. Organização / layout do sistema de busca de informações	5
2. Organização dos resultados da pesquisa	5
3. Atualização / manutenção da base de dados	5
4. Opções para classificar a pesquisa na base de dados (*quick research*)	5
5. Armazenagem das pesquisas realizadas	5
6. Informações dos responsáveis pela manutenção do sistema	3
7. Disponibilidade dos trabalhos para *downloads*	5
8. Apresentação de informações gerais (autores, título, ano de publicação, resumo e palavras-chave) do trabalho na lista de resultados	4
9. *Desing clean* do sistema	4

ANEXO II

Nome: Olivia Oiko
Data: 15/10/2009

Aplicação do QFD no Desenvolvimento de Software

Descrição do Sistema

O sistema é um gerenciador de pesquisas, para uso acadêmico. Com esta ferramenta o acadêmico poderá criar um perfil para o mesmo, e através desse perfil cadastrar os trabalhos que ele desenvolveu ou esta desenvolvendo.
O academico através do seu perfil pode pesquisar trabalhos que sejam semelhantes ao seu desenvolvidos por outros acadêmicos que estejam cadastrados no sistema e poderá ter acesso aos materias ou informações ultilizadas por eles.
Exemplo de campos de busca podem ser, String de busca, autor, ano, tipo…

Requisitos Funcionais.

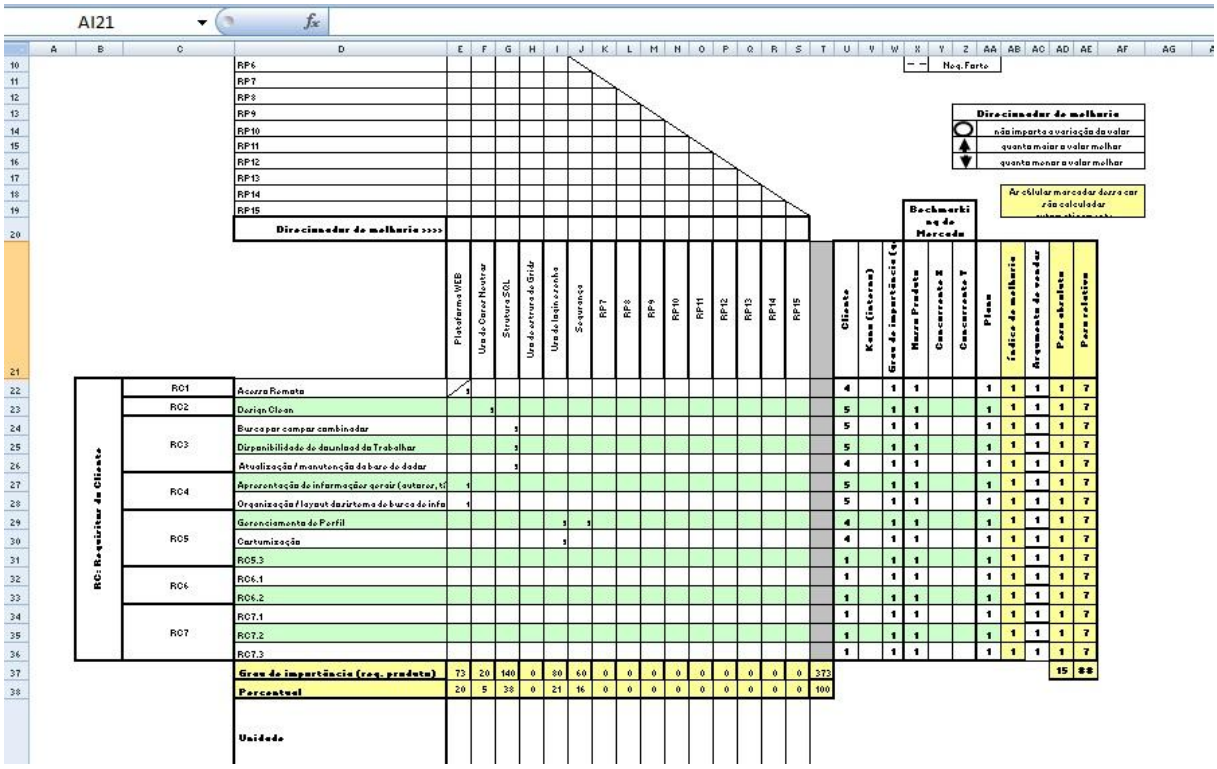
Nesse contexto imagine-se sendo um dos usúarios do sistema e informe quais o requisitos funcionais, ou seja, quais as funções você julga mais importantes e que o sistema deva contemplar com o seu respectivo valor de importância.

Requisitos **Valor de Importância (0 a 5 – mais importante)**

1. Acesso remoto (4)
2. Busca por: autor, palavra-chave, ano, strings, AND, OR (5)
3. Escolher o que eu quero que seja visível para outros perfis (2)
4. Exportar as referências selecionadas para um documento de texto, no formato de citação escolhido (5)
5. Permitir criar anotações sobre uma determinada referência e anexar outros arquivos (como um resumo, resenha etc) (5)
6. Classificar por importância (4)
7. Separar por categorias que eu criar (5)
8. Gerar estatísticas sobre os trabalhos (3)
9. Resgatar os trabalhos de acordo com as categorias ou importância que eu atribuir (4)

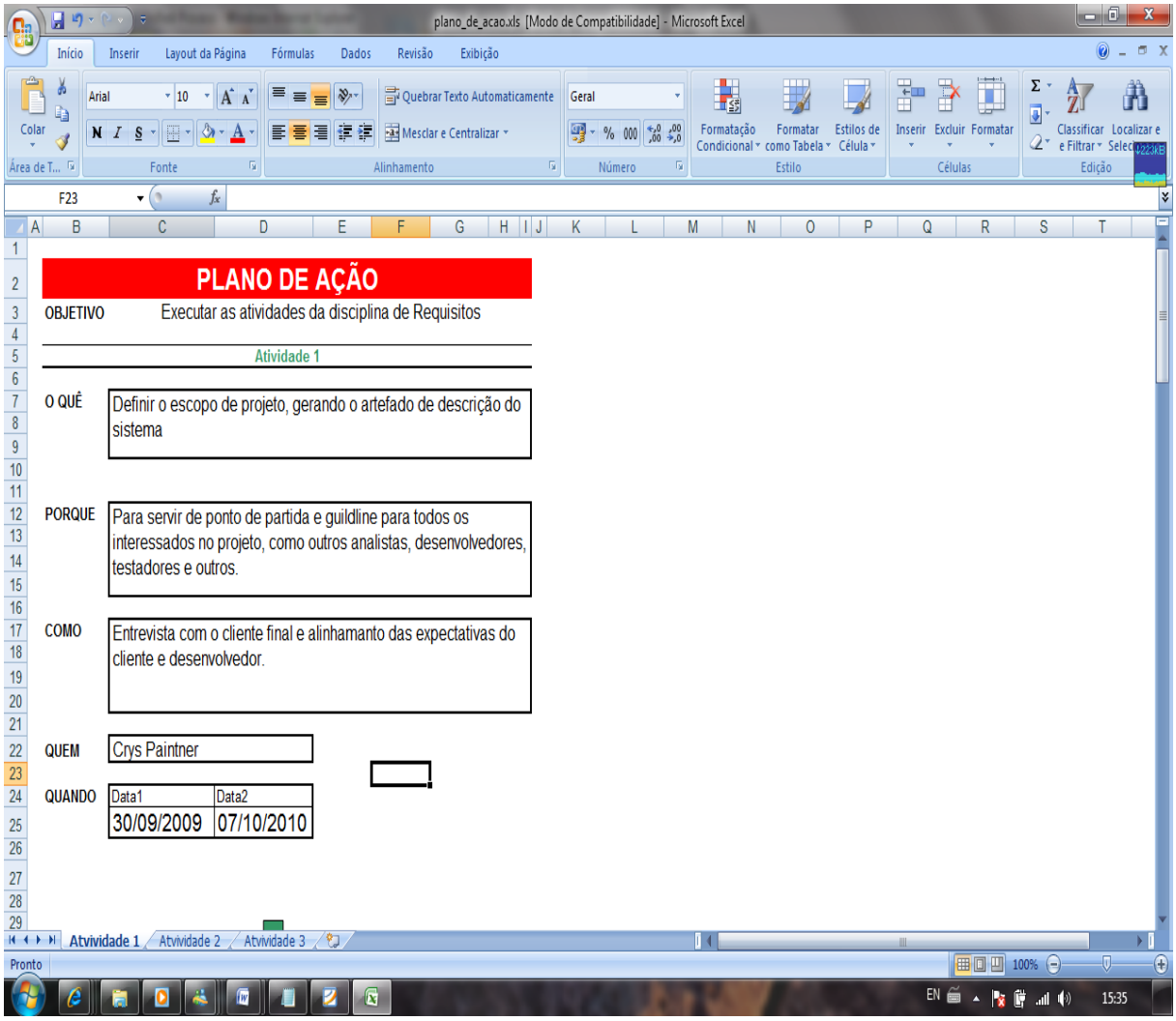
ANEXO III

Layout da planilha utilizada para a construção da Casa da Qualidade.



ANEXO IV

Exemplo de modelo de plano de ação adotado.



ANEXO V

FLUXOGRAMA DE VALIDAÇÃO

TITULO: Validar artefato

Fluxo:	Notas:
	Verificar confiabilidade do artefato Verificar integridade do artefato