UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

CÂMPUS PONTA GROSSA

DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO

VIII CURSO DE ESPECIALIZAÇÃO EM GESTÃO INDUSTRIAL: CONHECIMENTO E INOVAÇÃO

SUELYN FERNANDA DA SILVA

PROPOSTA DE METODOLOGIA PARA A FASE DE ANÁLISE DE REQUISITOS DE SOFTWARE BASEADO EM PRÁTICAS DE GESTÃO DO CONHECIMENTO PARA OBTENÇÃO DE QUALIDADE DE SOFTWARE.

MONOGRAFIA DE ESPECIALIZAÇÃO

PONTA GROSSA
2013

SUELYN FERNANDA DA SILVA

PROPOSTA DE METODOLOGIA PARA A FASE DE ANÁLISE DE REQUISITOS DE SOFTWARE BASEADO EM PRÁTICAS DE GESTÃO DO CONHECIMENTO PARA OBTENÇÃO DA QUALIDADE DE SOFTWARE.

Monografia apresentada como requisito parcial para a obtenção de título de especialista, do programa de pós-graduação da Universidade Tecnológica do Paraná, do curso de especialização em Gestão da produção – Conhecimento e inovação.

Orientadora: Profª Dra. Joseane Pontes

PONTA GROSSA
2013

Ministério da Educação
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CAMPUS PONTA GROSSA
Diretoria de Pesquisa e Pós-Graduação

TERMO DE APROVAÇÃO

Título da Monografia

PROPOSTA DE METODOLOGIA PARA A FASE DE ANÁLISE DE REQUISITOS DE SOFTWARE BASEADO EM PRÁTICAS DE GESTÃO DO CONHECIMENTO PARA OBTENÇÃO DE QUALIDADE DE SOFTWARE

por

Suelyn Fernanda da Silva

Esta monografia foi apresentada no dia 16 de março de 2013 como requisito parcial para a obtenção do título de ESPECIALISTA EM GESTÃO INDUSTRIAL: CONHECIMENTO E INOVAÇÃO. O candidato foi argüido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

Prof. Dr. Lourival Aparecido de Gois (UTFPR)

Prof. Dr. Guataçara dos Santos Junior (UTFPR)

Profª. Drª. Joseane Pontes (UTFPR)
Orientador

Visto do Coordenador:

Prof. Dr. Antonio Carlos de Francisco
Coordenador CEGI-CI
UTFPR – Câmpus Ponta Grossa

- O Termo de Aprovação assinado encontra-se na Coordenação do Curso -

Há quem diga que todas as noites são de sonhos. Mas há também quem garanta que nem todas, só as de verão. Mas no fundo isso não tem muita importância. O que interessa mesmo não são as noites em si, são os sonhos. Sonhos que o homem sonha sempre. Em todos os lugares, em todas as épocas do ano, dormindo ou acordado.

(Shakespeare)

AGRADECIMENTOS

Agradeço aos professores que tive em minha vida.

Minha Mãe Zilá.
Minhas Tias Zeli e Isabel.
Meu Namorado Fábio.
Minha Orientadora Joseane Pontes.

Todos os professores, que em um determinado momento me ensinaram algo, da forma mais clara e correta, com valores morais, intelectuais e sentimentais, e a união desses ensinamentos fizeram com que eu conseguisse finalizar mais esta etapa da minha vida. Obrigada.

SILVA, S. F. Proposta de metodologia para a fase de análise de requisitos de software baseado em práticas de gestão do conhecimento para obtenção da Qualidade de Software. Trabalho de Conclusão de Curso apresentado ao setor de Pós-Graduação da Universidade Tecnológica Federal do Paraná - Campus Ponta Grossa, 2013.

RESUMO

Resumo: O presente trabalho tem por finalidade propor uma metodologia para a fase de análise de sistemas, esta metodologia iria trabalhar diretamente na extração de requisitos do software. Nesta fase é necessário a comunicação do analista do sistema com o cliente, o que gera problemas pela dificuldade que existe em o cliente conseguir transmitir um conhecimento tácito, e o analista compreender o que foi transmitido e repassar para o papel de uma maneira consistente, podendo interferir futuramente na qualidade do software. A metodologia que será proposta utilizará o espiral do conhecimento, prática na Gestão do Conhecimento, como ferramenta para comunicação e a relacionaria com análise de requisitos, prática de Engenharia de Software, para identificar o que, e de que forma deve ser comunicado, estes conceitos seriam direcionados para chegar a um software de qualidade, para isso será apresentado o conceito de qualidade e qualidade de softwares através de um estudo teórico-bibliográfico dos conceitos e práticas de Gestão do conhecimento, engenharia de software e as metodologias existentes na área de Análise de sistemas e qualidade de software. Através do estudo pode-se perceber que o espiral do conhecimento é uma grande ferramenta para ser utilizada na extração do conhecimento e este transformado em requisitos do um software. O espiral modificado auxilia a criar uma nova metodologia para extração de requisitos. De uma maneira teórica, pois não houve aplicação mas sim sugestão para trabalho futuro, os requisitos obtidos serão mais completos e satisfatórios. Com isso alguns problemas que poderiam ocorrer por falha de comunicação no inicio do projeto e seriam repassadas para outras fases do projeto são eliminados desde o inicio, gerando assim softwares com maior qualidade.

Palavras-Chave: Espiral do Conhecimento; Analise de Requisitos; Qualidade de Software.

SILVA, S. F. Proposed methodology for the analysis phase of software requirements based on knowledge management practices to obtain the Software Quality. Completion of course work presented to industry Postgraduate Federal Technological University of Paraná - Campus Ponta Grossa, 2013.

ABSTRACT

Abstract: This paper aims to propose a methodology for the analysis phase of systems, this methodology would work directly in the extraction of software requirements. At this stage it is necessary to communicate the system analyst with the customer, who creates problems for the difficulty that exists on the client can convey tacit knowledge, and the analyst understand what was transmitted and to pass on the role in a consistent manner, which may interfere future in software quality. The proposed methodology will use the knowledge spiral, Knowledge Management in practice, as a tool to communicate and relate with requirements analysis, software engineering practice, to identify what and how they should be reported, these concepts would be targeted to reach a quality software, it will be presented to the concept of quality and quality of software through a bibliographic study theoretical concepts and practices of knowledge management, software engineering and methodologies in the area of systems analysis and software quality. Through the study can be seen that the spiral of knowledge is a great tool to be used in the extraction of knowledge and this turned into a software requirements. The modified spiral helps to create a new methodology for extracting requirements. From a theoretical way, because there was no application but suggestions for future work, the requirements will be obtained more complete and satisfactory. Thus some problems that could occur through miscommunication early in the project and would be passed on to other phases of the project are eliminated from the start, thus generating higher quality software.

Keywords: Spiral of Knowledge, Requirements Analysis, Software Quality.

LISTA DE FIGURAS

FIGURA 1 – Visibilidade dos requisitos implícitos e explícitos..... 27
FIGURA 2 – Processos da GC 31
FIGURA 3 – Passos da GC 31
FIGURA 4 – A espiral do conhecimento 32
FIGURA 5 – Metodologia para Análise de Requisitos 34
FIGURA 6 – Modelo de Planilha para Mapeamento de Competências..... 40
FIGURA 7 – Possíveis Relacionamentos entre habilidades..... 41
FIGURA 8 – Estimativa de Tempo para Aplicação da metodologia..... 44

LISTA DE QUADROS

QUADRO 1 – Normas ISO 17
QUADRO 2 – Principais organismos normativos..... 19
QUADRO 3 – Características existentes em um software de qualidade..... 20
QUADRO 4 – FASE 1 Socialização..... 39
QUADRO 5 – FASE 2 Externalização..... 41
QUADRO 6 – FASE 3 Combinação..... 72
QUADRO 7 – FASE 4 Internalização 43

SUMÁRIO

1 NTRODUÇÃO..... 11
1.1 JUSTIFICATIVA..... 11
1.2 PROBLEMATICA..... 14
1.3 BJETIVOS..... 14
1.3.1 Objetivo Geral..... 14
1.3.2 Objetivos Específicos..... 14
1.4 METODOLOGIA..... 15
2 FUNDAMENTAÇÃO TEÓRICA..... 15
2.1 QUALIDADE..... 17
2.3 ENGENHARIA DE SOFTWARES..... 21
2.3.1 Metodologias para Desenvolvimento de Softwares 22
2.3.2 Requisitos de *Software*..... 27
2.4 GESTÃO DO CONHECIMENTO..... 33
2.4.1 Conversão do Conhecimento..... 34
3 METODOLOGIA PROPOSTA..... 37
4 CONCLUSÃO..... 45
4.1 TRABALHOS FUTUROS..... 46
5 REFERÊNCIAS..... 47

1 INTRODUÇÃO

Os Sistemas de Informação são uma importante ferramenta utilizada pelas empresas em suas rotinas, o computador processa os dados obtendo resultados com velocidade, qualidade e segurança. Com isso, as tarefas a serem realizadas demandam menos tempo e a tomada de decisão baseada nestas informações é mais rápida, aumentando a competitividade da empresa e diminuindo custos.

Os Sistemas de Informação podem ser encontrados em diversos setores da empresa como: RH, financeiro, estoque, agendamentos, entre outros. Todo o setor que possui um software para execução de suas funções precisa ser alimentado por informações a serem processadas para gerar os resultados.

O controle de estoque é um exemplo que pode ser utilizado. Ao adquirir um produto é realizado uma entrada do produto no estoque, este valor é um dado a ser colocado no computador, o computador realiza o processamento e atualiza o estoque. Da mesma maneira acontece na saída onde é passado um valor de produtos a serem retirados e o computador diminui do estoque.

A função de controlar um estoque e outras funções passiveis de informatização se tornam complexas quando existe um grande numero de dados a serem processados, como produtos, fornecedores, funcionários, relatórios e clientes. Já para o computador é uma tarefa simples de executar.

Para LAUDON E LAUDON (2007) um sistema de informação pode ser definido como um conjunto de componentes inter-relacionados trabalhando juntos para coletar, recuperar, processar, armazenar e distribuir informações com a finalidade de facilitar o planejamento, o controle, a coordenação, a análise e o processo decisório em organizações.

Os Sistemas de Informação computacionais são utilizados para desenvolver as tarefas que o ser humano executa de maneira mais lenta, pois a sua capacidade de processamento é menor que a de um computador. Porém, os computadores não são inteligentes e precisam ser programados para realizar as funções desejadas.

Engenharia de software é uma área da computação voltada ao desenvolvimento de sistemas através da especificação, desenvolvimento e manutenção do software (FALBO, 1998). A especificação de software é definida por SCHWARTZ(1975), PRESSMAN(1997) e SOMMERVILLE(1995) como Análise de

Sistemas fase onde ocorre o levantamento das necessidades do software a ser implementado. A Análise de sistemas tem como objetivo produzir uma especificação de requisitos, que convencionalmente é um documento onde está descrito o que o usuário deseja que o software realize.

A comunicação entre o analista de sistema, profissional que realiza a análise dos requisitos, e o cliente é fundamental na fase de análise de requisitos. Algumas técnicas são utilizadas nesta fase, os analistas seguem algumas rotinas para que a comunicação venha a ser mais proveitosa.

Porem mesmo utilizando técnicas existentes como: entrevistas, etnografia, questionários, cenários, existem algumas dificuldades nesta comunicação das quais podemos citar:

- Os usuários não tem uma ideia precisa do sistema por eles requerido,
- Têm dificuldades para descreverem seu conhecimento sobre o domínio do problema
- Usuários e analistas têm diferentes pontos de vista do problema (por terem diferentes formações).

Para resolver os problemas originados pela dificuldade de comunicação a criação de uma metodologia focada na a análise de requisitos, iria auxiliar na correção dos problemas, obtendo melhores resultados na construção do software.

A área que a ser utilizada para auxiliar na resolução deste problema é a de gestão do conhecimento (GC) pois segundo a definição de SALAZAR (2002) pode ser entendida como um processo sistemático de detectar, selecionar, organizar, filtrar, compartilhar, adquirir e utilizar a informação dentro da empresa.

Através da definição de SALAZAR (2002), pode-se perceber que o processo sistemático da GC pode ser uma possível solução para a dificuldade de comunicação que interfere diretamente na analise de requisitos. Esta solução criaria uma metodologia para Analise de requisitos, baseada em gestão do conhecimento.

1.1 JUSTIFICATIVA

Para aumentar o seu diferencial competitivo as empresas necessitam de uma ferramenta tecnológica para auxilia-las em seus processos, pois os softwares aumentam a produtividade global da empresa com uma gestão eficiente da

informação (NEVES, 2006).

As empresas adquirem softwares para auxiliar em seus sistemas internos, interligando todas as fases do seu processo através de um programa de computador que atue de maneira continua e sem erros ao longo do processo, prevendo possíveis erros e mostrando resultados para a empresa de maneira clara e objetiva.

Mas a realidade dos softwares existentes é muito diferente disso, os sistemas normalmente são incompletos e para executar todas as funcionalidades dos processos da empresa são necessários vários softwares. Com isso se perde a noção do continuo, e os resultados apresentados não são exatamente como os desejados.

A maioria das empresas perdem recursos, dentre eles recursos financeiros com softwares que não tem em seu escopo todos os requisitos necessários para executar a função com qualidade, velocidade e confiabilidade. E a ideia de ferramenta de auxilio acaba se desfazendo pela dificuldade em obter um software nos moldes da empresa. Segundo ROCHA (2001), muitos desses problemas são derivados da falta de atenção para a tarefa de definir e acompanhar a evolução dos requisitos durante o processo de construção de software.

Conforme PRESSMAN (1995) uma compreensão completa dos requisitos de software é fundamental para um bem-sucedido desenvolvimento de software. Não importa quão bem projetado ou quão bem codificado seja, um programa mal analisado e especificado desapontará o usuário e poderá trazer problemas ao desenvolvedor. Conforme BURNETT (1998), requisitos, simplesmente podem ser definidos como "algo que um cliente necessita". Para LOPES (1999) requisitos de um sistema definem os serviços que o sistema deve oferecer e as restrições aplicáveis à sua operação.

Segundo HATANO (2012) o analista para desempenhar bem suas funções deve ter como características:

- Capacidade de compreender conceitos abstratos, reorganizá-los em divisões lógicas e sintetizar "soluções" baseadas em cada divisão.
- Capacidade de absorver fatos pertinentes de fontes conflitantes ou confusas.
- Capacidade de entender os ambientes do usuário.
- Capacidade de aplicar elementos do sistema de hardware e/ou software

aos elementos do usuário/cliente.

- Capacidade de se comunicar bem nas formas escrita e verbal.

A Análise de requisitos é um processo cuidadoso de interação com pessoas aliado à necessidade de avaliação da organização, do domínio da aplicação e dos processos de negócio em que o sistema operará. Pode parecer apenas um processo de transferência de conhecimento entre pessoas, e seria, se o cliente soubesse exatamente o que está precisando, fato que raramente se encontra no mundo real.

Outro fator a se considerar segundo STÁBILI et all (2005), é que o profissional da informática não tem como característica principal o perfil comunicativo, e o cliente, assim como qualquer pessoa, tem dificuldades em conseguir colocar por escrito o que realmente deseja.

Segundo STANDISH GROUP (1994) os fatores que contribuem para o fracasso de um projeto são:

1º Requisitos incompletos (13,1%);

2º Falta de envolvimento por parte do usuário (12,4%);

3º Falta de recursos (10,6%);

4º Expectativas não realistas (9,9%);

5º Falta de apoio dos executivos (9,3%);

6º Modificações nos requisitos e nas especificações (8,7%);

7º Falta de planejamento (8,1%);

8º O sistema não era mais necessário (7,5%).

A criação de uma metodologia para a fase que ocasiona mais fracassos no desenvolvimento do software seria de grande auxilio para profissionais da informática e clientes pois a qualidade do produto a ser construído dependerá diretamente da qualidade dos requisitos identificados. Como as ferramentas existentes não resolvem este problema é necessário estudar outras áreas do conhecimento para serem combinadas gerando uma metodologia mais eficiente.

1.2 PROBLEMÁTICA

Como envolver a fase de requisitos de software baseado em prática de Gestão do Conhecimento para a obtenção da Qualidade do Software?

1.3 OBJETIVOS

1.3.1 Objetivo Geral

Propor uma metodologia para a fase de análise de requisitos de software baseado em práticas de gestão do conhecimento para obtenção da qualidade de software.

1.3.2 Objetivos Específicos

- Analisar os requisitos exigidos para considerar como um software sendo de qualidade;
- Identificar as principais metodologias que visam obter qualidade no software;
- Apresentar práticas de gestão do conhecimento que podem auxiliar na fase de análise de requisitos;

1.4 METODOLOGIA

Para GIL (1999, p.42), a pesquisa tem um caráter pragmático, é um "processo formal e sistemático de desenvolvimento do método científico. O objetivo fundamental da pesquisa é descobrir respostas para problemas mediante o emprego de procedimentos científicos".

Segundo GIL (1999) a pesquisa é um conjunto de ações, propostas para encontrar a solução para um problema, que têm por base procedimentos racionais e sistemáticos. A pesquisa é realizada quando se tem um problema e não se têm informações para solucioná-lo.

Esta pesquisa pretende ser realizada através de revisão bibliográfica e análise exploratória em periódicos, livros e trabalhos científicos sobre os seguintes temas: Engenharia de software, Gestão do conhecimento, Qualidade de software.

A partir desta pesquisa pode-se identificar o problema na fase de análise de requisitos, e apresentado uma possível solução através da Gestão do Conhecimento.

1.5 ESTRUTURA DO TRABALHO

O capítulo 1 apresentou a contextualização do tema, sua importância, a problemática envolvida e seus objetivos gerais e específicos e a metodologia que será utilizada.

O capítulo 2 apresentará a fundamentação teórica através dos temas: Qualidade e as Normas ISO, Qualidade de Software e suas Normas e Organismos Normativos, a Engenharia de Software, suas metodologias de desenvolvimento de software e também requisitos de software e Gestão do Conhecimento com suas Conversões.

O capítulo 3 apresentará a proposta da metodologia para a solução da problemática.

Por último será apresentado a conclusão, propostas de trabalhos futuros e referencial bibliográfico.

2 FUNDAMENTAÇÃO TEÓRICA

Para explorar o tema sobre Qualidade de software, faz-se necessário uma explicação mais clara sobre qualidade de uma maneira geral e qualidade na área de *softwares*. Ao relacionar o que é um *software* de qualidade e como fazer um *software* de qualidade estar-se-a entrando no universo de Engenharia de softwares que irá apresentar o "como fazer" de maneira eficiente, e juntamente quais as dificuldades encontradas para seguir esta maneira eficiente. É neste ponto que será necessário conhecer as metodologias mais utilizadas para a produção do *softwares* e suas ferramentas.

Pode-se identificar as falhas que ocorre entre o "como fazer" e o "o que fazer". Na teoria o processo de Análise de Requistos é simples. O cliente informa ao analista sobre as funções que deseja que seu software realize, o analista por sua vez prepara e passa as informações recebidas do cliente para a próxima fase.

A teoria parece simples, porém com informações na área de Gestão do conhecimento entenderemos o quão complicado é transferir conhecimento, principalmente se as duas pessoas que trocarão este conhecimento não estiverem preparadas para este processo.

A seguir será mostrado de uma forma mais clara conceitos sobre qualidade, engenharia de software, gestão do conhecimento, e qualidade de software, que servirão de base para o entendimento sobre o desafio proposto pelo trabalho.

2.1 QUALIDADE

Qualidade é um conceito subjetivo que está relacionado diretamente às percepções de cada indivíduo. Autores como CROSBY (1979), FEIGENBAUM (1991), JURAN (1991), GIANESI e CORRÊA (1996), GARVIN (2002), conceituaram qualidade de diferentes formas como pode ser visto a seguir.

CROSBY (1979) definiu qualidade como conformação aos requisitos, ou seja, requisitos estabelecidos para produtos e serviços, que devem ser cumpridos da melhor forma possível. Conforme FEIGENBAUM (1991), a palavra qualidade não tem o significado popular de melhor, no sentido absoluto. Ela significa melhor para certos requisitos do cliente. Estes requisitos são o uso real e estão relacionados ao

preço de venda do produto, principalmente. Portanto, a qualidade seria a maximização do valor do produto, tratando valor como a relação entre expectativa do cliente e preço de venda. Para JURAN (1991), qualidade é adequação ao uso. Assim, para que um produto tenha qualidade, ele deve atender às necessidades dos clientes.

GIANESI e CORRÊA (1996) afirmam que a qualidade significa atender às reais necessidades dos clientes, sejam elas explícitas ou implícitas, dentro do prazo que o cliente deseja e a um justo valor.

Segundo CAMPOS (2004), um produto ou serviço de qualidade é aquele que atende perfeitamente às necessidades do cliente, de maneira confiável e segura, dentro de um tempo determinado.

Os autores concordam que qualidade está diretamente ligada ao cliente e aos requisitos do cliente para o produto, os requisitos são importantes pois através dele podemos comparar com as características que eles possuem, e assim definir se este é de qualidade ou não. Isto significa que quanto mais longe o produto estiver das especificações, menor será a sua qualidade. Para auxiliar na comparação das características os padrões foram criados e permitem que consumidores encontrem compatibilidade entre produtos, processos e serviços. Com isso surgiu a necessidade de normas para regular as atividades ou seus resultados, a fim de obter um ótimo grau de ordenação em um dado contexto.

A International Electrotechnical Commission (IEC), de 1906, tem sido considerada a pioneira na coordenação do processo para o estabelecimento de medidas e critérios unificados, respeitados entre indústrias de diferentes países.

Em 1946, foi criada a International Organization for Standardization, a conhecida ISO, organização não governamental de órgãos de normalização de todo o mundo. A ISO oferece soluções que vêm ao encontro das necessidades das organizações, da sociedade e seus padrões já foram adotados por cerca de 630 mil organizações em 152 países, trazendo melhorias nas áreas de desenvolvimento, produção e suprimentos das organizações. O Quadro 1 mostra as áreas da indústria que possuem normas ISO e suas datas de criação.

Indústria	**1991-95**	**1996-2000**	**2001-06**
Qualidade	ISO 9000:1994(ISO)	ISO 9000:2000 (ISO)	
Ambiente		ISO 14000:1999 (ISO)	ISO 14001:2004 (ISO)

Segurança		OHSAS18001:1999 (UK)	SMS (Singapura,2002), Cool Program (Singapura, 2003)
Segurança Alimentar	HACCP (Dinamarca, 1994)	BRC (UK, 1998), SQF20000 (Suíça)	ISSO 22000:2005
Tecnologias de Informação (IT)	BS7799-1 (UK,1995)	CMM (US, 2000). BS7799-1/2(UK,1998/1999), ISO 17799-1(ISO,2000)	CMMI (US,2002)
Automóvel QS	9000 (Big 3, 94 e 95)	QS 9000 (Big 3, 1998), ISO/TS 16949 (ISO/IAOB 1999)	ISO/TS 16949 (ISO/IAOB, 2002)

Quadro 1 – Normas ISO

Fonte: Autoria Própria

Das área existentes na ISO será apresentada a norma regulamentadora que representa a atual padronização mundial para a qualidade de produtos de software chamada ISO/IEC 9126 e foi publicada em 1991. Além da padronização ISO/IEC 9126 outras também são utilizadas, o Quadro 2 mostra o organismo normativo e a descrição das Normas relacionadas com Softwares.

ORGANISMO NORMATIVO	**DESCRIÇÃO**
ISO 9126	Características da qualidade de produtos de software
NBR 13596	Versão brasileira da ISO 9126
ISO 14598	Guias para a avaliação de produtos de software, baseados na utilização prática da norma ISO 9126
ISO 12119	Características de qualidade de pacotes de software (software de prateleira, vendido com um produto embalado)
IEEE P1061	Standard for Software Quality Metrics Methodology (produto de software)
ISO 12207	Software Life Cycle Process. Norma para a qualidade do processo de desenvolvimento de software.
NBR ISO 9001	Sistemas de qualidade - Modelo para garantia de qualidade em Projeto, Desenvolvimento, Instalação e Assistência Técnica (processo)
NBR ISO 9000-3	Gestão de qualidade e garantia de qualidade. Aplicação da norma ISO 9000

	para o processo de desenvolvimento de software.
NBR ISO 10011	Auditoria de Sistemas de Qualidade (processo)
CMM	Capability Maturity Model. Modelo da SEI (Instituto de Engenharia de Software do Departamento de Defesa dos EEUU) para avaliação da qualidade do processo de desenvolvimento de software. Não é uma norma ISO, mas é muito bem aceita no mercado.
SPICE ISO 15504	Projeto da ISO/IEC para avaliação de processo de desenvolvimento de software. Ainda não é uma norma oficial ISO, mas o processo está em andamento.

Quadro 2 – Principais organismos normativos

Fonte: Autoria Própria

O Quadro 2 mostra que a área de Engenharia de software possui regulamentações para avaliação de todo o processo de desenvolvimento desde a qualidade do processo, até a qualidade final, os requisitos necessários e a auditoria.

Segundo ROCHA (1994), no contexto de desenvolvimento de software, qualidade pode ser entendida como um conjunto de características a serem satisfeitas, de modo que o produto de software atenda às necessidades explícitas e implícitas de seus usuários. Entretanto, não se obtém qualidade do produto de forma espontânea. Ela tem de ser construída. Assim, a qualidade do produto depende fortemente da qualidade de seu processo de desenvolvimento.

Para analisar a qualidade de um software precisa-se definir quais critérios que servem para julga-lo. A norma 9126 se foca na qualidade do produto de software, propondo Atributos de Qualidade, distribuídos em seis características principais, com cada uma delas divididas em sub-características, conforme podemos o Quadro a seguir.

CARACTERÍSTICA	SUBCARACTERÍSTICA	PERGUNTA CHAVE PARA SUBCARACTERISTICA
Funcionalidade (satisfaz as necessidades?)	Adequação	Propõe-se a fazer o que é apropriado?
	Acerácea	Faz o que foi proposto de forma correta?
	Interoperabilidade	Interage com os sistemas especificados?
	Conformidade	Está de acordo com as normas, leis, etc?
	Segurança de acesso	Evita acesso não autorizado dos dados
Confiabilidade (É imune a falhas?)	Maturidade	Com que frequência apresenta falhas?
	Tolerância a falhas	Ocorrendo falhas, como ele reage?
	Recuperabilidade	É capaz de recuperar dados em caso de falhas?
Usabilidade (é fácil de usar?)	Intelegibilidade	É fácil de entender o conceito e a aplicação?
	Apreensibilidade	É fácil aprender a usar?

	Operacionalidade	É fácil de operar e controlar?
Eficiência (é rápido e "enxuto"?)	Tempo	Qual é o tempo de resposta, a velocidade de execução?
	Recursos	Quanto recurso usa? Durante quanto tempo?
Manutenibilidade (é fácil de modificar?)	Analisabilidade	É fácil de encontrar uma falha quando ocorre?
	Modificabilidade	É fácil de modificar e adaptar?
	Estabilidade	Há grande risco quando faz alterações?
	Testabilidade	É fácil testar quando se faz alterações?
Portabilidade (é fácil de usar em outro ambiente?)	Adaptabilidade	É fácil adaptar a outros ambientes?
	Capac. Para ser instalado	É fácil instalar em outros ambientes?
	Conformidade	Está de acordo com padrões de portabilidade?
	Capac. Para substituir	É fácil usar para substituir outro?

Quadro 3 – características de um software de qualidade

Fonte: Autoria Própria

Para poder se adequar as normas é necessário utilizar metodologias e ferramentas de apoio para o desenvolvimento do software, estas metodologias e ferramentas serão utilizadas para garantir que os objetivos e características do software sejam obedecidos. Todo o processo de produção de software se encontra na Engenharia de softwares que o próximo item irá abranger.

2.3 ENGENHARIA DE SOFTWARE

O termo Engenharia pretende indicar que o desenvolvimento de software deve submeter-se a leis similares as que governam a manufatura de produtos industriais em engenharias tradicionais, pois ambos são metodológicos (MAFFEO, 1992). A Engenharia Civil é um exemplo de engenharia tradicional metodológica que possui os seguintes passos: contexto, arquitetura, projeto técnico, construção e validação. Similarmente, num projeto de Construção de Software as etapas são: Engenharia de Sistema, Analise de requisitos, projeto técnico, construção e validação (MARTINS, 2007).

Segundo REZENDE (2005) nos dois casos (elaboração de produtos industriais e elaboração de sistemas de software), o processo exige que seja criticamente considerada a análise e especificação de requisitos bem como a seleção criteriosa da metodologia de "produção". Distinções relevantes entre os dois

tipos de “produtos” circunscreverão (limitarão) a apropriação do termo “engenharia” para a área de software.

Para REZENDE (2005) Engenharia de Software é a metodologia de desenvolvimento e manutenção de sistemas modulares. Esta metodologia possui as seguintes características: processo (roteiro) dinâmico, integrado e inteligente de soluções tecnológicas; adequação dos requisitos funcionais dos negócios do cliente e seus respectivos procedimentos pertinentes. Os procedimentos são: efetivação de padrões de qualidade, produtividade e efetividade em suas atividades e produtos, fundamentação na tecnologia da informação disponível, viável, oportuna e personalizada. Planejamento e Gestão de atividades, recursos, custos e datas.

Para SOMMERVILE (2003) A engenharia de software (ES) envolve questões técnicas e não técnicas, tais como a especificação do conhecimento, técnicas do projeto e implementação, conhecimento dos fatores humanos pelo engenheiro de software e ainda a gestão do projeto.

Para MAFFEO (1992) A ES é: “a área interdisciplinar que engloba vertentes tecnológica e gerencial visando abordar de modo sistemático (modular) os processos de construção, implantação e manutenção de produtos de software com qualidade assegurada por construção segundo cronogramas e custos previamente definidos.

A seguir será apresentado as metodologias existentes para a Engenharia de softwares e as metodologias mais utilizadas.

2.3.1 Metodologias para Desenvolvimento de *Softwares*

Existem diversas metodologias que são utilizadas, elas auxiliam o desenvolvimento do softwares através de um conjunto de atividades.

Segundo SOMMERVILE (2003) existem atividades comuns em todas elas:

- Especificação: definição das funcionalidades e demais características do produto;
- Projeto e Implementação: o software é produzido de acordo com as especificações;
- Validação: atividade de revisão e testes visando assegurar que os requisitos sejam cumpridos;
- Evolução: atividade de manutenção do software.

Existem 2 tipos de metodologias para desenvolvimento de softwares, as metodologias tradicionais, e as ágeis.

A metodologia tradicional surgiu em um contexto de desenvolvimento de software diferente do atual, a principal metodologia tradicional é a clássica.

A principal característica da metodologia clássica é a inflexibilidade, ela esta dividida em fases, e após cada fase não se preveem mudanças das especificações. REIS (2001).

As metodologias ágeis são as mais utilizadas para o desenvolvimento de softwares pois elas trabalham com mudanças frequentes de requisitos. Ao invés de prever o futuro esta metodologia trabalha com as mudanças.

Segundo BECK (1999) e SCHWABER E BEEDLE (2002) as metodologias ágeis mais conhecidas são Metodologia Scrum e a metodologia XP.

- Metodologia Extreme Programming

A Extreme Programming (XP) é uma metodologia ágil para equipes pequenas e médias que desenvolvem software baseado em requisitos vagos e que se modificam rapidamente BECK (1999). Dentre as principais diferenças da XP em relação às outras metodologias estão:

- Feedback constante;
- Abordagem incremental;
- A comunicação entre as pessoas é encorajada.

A finalidade do princípio de comunicação é manter o melhor relacionamento possível entre clientes e desenvolvedores, preferindo conversas pessoais a outros meios de comunicação. A comunicação entre os desenvolvedores e o gerente do projeto também é encorajada.

A simplicidade visa permitir a criação de código simples que não deve possuir Funções desnecessárias. Por código simples entende-se implementar o software com o menor número possível de classes e métodos. Outra ideia importante da simplicidade é procurar implementar apenas requisitos atuais, evitando-se adicionar funcionalidades que podem ser importantes no futuro. A aposta da XP é que é melhor fazer algo simples hoje e pagar um pouco mais amanhã para fazer

modificações necessárias do que implementar algo complicado hoje que talvez não venha a ser usado, sempre considerando que requisitos são mutáveis.

A prática do *feedback* constante significa que o programador terá informações constantes do código e do cliente. A informação do código é dada pelos testes constantes, que indicam os erros tanto individuais quanto do *software* integrado. Em relação ao cliente, o *feedback* constante significa que ele terá frequentemente uma parte do software totalmente funcional para avaliar. O cliente então constantemente sugere novas características e informações aos desenvolvedores. Eventuais erros e não conformidades são rapidamente identificados e corrigidos nas próximas versões.

Desta forma, a tendência é que o produto final esteja de acordo com as expectativas reais do cliente. É necessário coragem para implantar os três valores anteriores. Por exemplo, não são todas as pessoas que possuem facilidade de comunicação e têm bom relacionamento. A coragem também dá suporte à simplicidade, pois assim que a oportunidade de simplificar o software é percebida, a equipe pode experimentar. Além disso, é preciso coragem para obter *feedback* constante do cliente.

Segundo BECK (1999), a XP baseia-se nas 12 práticas a seguir:

- Planejamento: consiste em decidir o que é necessário ser feito e o que pode ser adiado no projeto. A XP baseia-se em requisitos atuais para desenvolvimento de *software*, não em requisitos futuros. Além disso, a XP procura evitar os problemas de relacionamento entre a área de negócios (clientes) e a área de desenvolvimento. As duas áreas devem cooperar para o sucesso do projeto, e cada uma deve focar em partes específicas do projeto. Desta forma, enquanto a área de negócios deve decidir sobre o escopo, a composição das versões e as datas de entrega, os desenvolvedores devem decidir sobre as estimativas de prazo, o processo de desenvolvimento e o cronograma detalhado para que o software seja entregue nas datas especificadas.
- Entregas frequentes: visa à construção de um *software* simples, e conforme os requisitos surgem, há a atualização do *software*. Cada versão entregue deve ter o menor tamanho possível, contendo os requisitos de maior valor para o negócio. Idealmente devem ser entregues versões a cada mês, ou no máximo a cada dois meses, aumentando a possibilidade de *feedback* rápido do cliente. Isto evita surpresas caso o software seja entregue após muito tempo, melhora as avaliações e

o *feedback* do cliente, aumentando a probabilidade do software final estar de acordo com os requisitos do cliente.

- Metáfora: são as descrições de um *software* sem a utilização de termos técnicos, com o intuito de guiar o desenvolvimento do software.

- Projeto simples: o programa desenvolvido pelo método XP deve ser o mais simples possível e satisfazer os requisitos atuais, sem a preocupação de requisitos futuros. Eventuais requisitos futuros devem ser adicionados assim que eles realmente existirem. Esta forma de raciocínio se opõe ao “implemente para hoje e projete para amanhã”.

- Testes: a XP focaliza a validação do projeto durante todo o processo de desenvolvimento. Os programadores desenvolvem o *software* criando primeiramente os testes.

- Refatoração: focaliza o aperfeiçoamento do projeto do *software* e está presente em todo o desenvolvimento. A refatoração deve ser feita apenas quando é necessário, ou seja, quando um desenvolvedor da dupla, ou os dois, percebe que é possível simplificar o módulo atual sem perder nenhuma funcionalidade.

- Programação em pares: a implementação do código é feita em dupla, ou seja, dois desenvolvedores trabalham em um único computador. O desenvolvedor que está com o controle do teclado e do mouse implementa o código, enquanto o outro observa continuamente o trabalho que está sendo feito, procurando identificar erros sintáticos e semânticos e pensando estrategicamente em como melhorar o código que está sendo implementado. Esses papéis podem e devem ser alterados continuamente. Uma grande vantagem da programação em dupla é a possibilidade dos desenvolvedores estarem continuamente aprendendo um com o outro.

- Propriedade coletiva: o código do projeto pertence a todos os membros da equipe. Isto significa que qualquer pessoa que percebe que pode adicionar valor a um código, mesmo que ele próprio não o tenha desenvolvido, pode fazê-lo, desde que faça a bateria de testes necessária. Isto é possível porque na XP todos são responsáveis pelo *software* inteiro. Uma grande vantagem desta prática é que, caso um membro da equipe deixe o projeto antes do fim, a equipe consegue continuar o projeto com poucas dificuldades, pois todos conhecem todas as partes do *software*, mesmo que não seja de forma detalhada.

- Integração contínua: interagir e construir o sistema de *software* várias vezes por dia, mantendo os programadores em sintonia, além de possibilitar processos

rápidos. Integrar apenas um conjunto de modificações de cada vez é uma prática que funciona bem porque fica óbvio quem deve fazer as correções quando os testes falham: a última equipe que integrou código novo ao software. Esta prática é facilitada com o uso de apenas uma máquina de integração, que deve ter livre acesso a todos os membros da equipe.

- 40 horas de trabalho semanal: a XP assume que não se deve fazer horas extras constantemente. Caso seja necessário trabalhar mais de 40 horas pela segunda semana consecutiva, existe um problema sério no projeto que deve ser resolvido não com aumento de horas trabalhadas, mas com melhor planejamento, por exemplo. Esta prática procura ratificar o foco nas pessoas e não em processos e planejamentos. Caso seja necessário, os planos devem ser alterados, ao invés de sobrecarregar as pessoas.

- Cliente presente: é fundamental a participação do cliente durante todo o desenvolvimento do projeto. O cliente deve estar sempre disponível para sanar todas as dúvidas de requisitos, evitando atrasos e até mesmo construções erradas. Uma ideia interessante é manter o cliente como parte integrante da equipe de desenvolvimento.

- Código padrão: padronização na arquitetura do código, para que este possa ser compartilhado entre todos os programadores.

A XP é ideal para projeto em que os cliente não sabem exatamente o que querem e podem mudar muito de opinião no desenvolvimento do sistema, KOSCIANSKI E MARTINS (2007).

- Metodologia Scrum

A metodologia Scrum propõe uma forma de trabalho flexível que se adapte a ambientes muito dinâmicos. Segundo KOSCIANSKI E MARTINS (2007) visa tratar mudanças frequentes de requisitos do *software* e outras situações como troca de equipe, adaptação de cronogramas e orçamentos, troca de ferramentas de desenvolvimento ou linguagem de programação.

Na metodologia Scrum existem reuniões diárias com duração de aproximadamente 15 minutos.

O ciclo de vida da Scrum é baseado em três fases principais, Pré-planejamento, Desenvolvimento e pós-planejamento.

No pré planejamento os requisitos são descritos em um documento chamado

backlog. Posteriormente são ordenados por prioridade e, para cada um, fazem-se estimativas de esforço para o desenvolvimento. Esta fase é concluída com a proposta de uma arquitetura de desenvolvimento.

No desenvolvimento o *software* é desenvolvido em ciclos, em que novas funcionalidades são adicionadas. Cada um destes ciclos passa pela fase de Análise, projeto, implementação e testes. O tempo dos ciclos devem durar entre uma semana a um mês.

Na fase de pós planejamento é feito a integração dos softwares, os testes finais e a documentação para o usuário. A equipe se reúne para analisar o projeto e demonstrar o software atual para o cliente.

Com base no que foi apresentado sobre a metodologia Scrum e XP pode-se perceber que as duas trabalham com a mudança frequente de requisitos, estas metodologias foram criadas para que se possa alterar rapidamente todas as fases do projeto quando os requisitos forem modificados.

A utilização de metodologias que controlam essas mudanças mostram o quão é importante se obter requisitos de maneira mais eficiente, estes requisitos são determinantes para se verificar a qualidade do software.

Na fase de analise de requisitos é necessário saber exatamente o que construir. A partir destes requisitos podemos compara-los com o projeto final e avaliar se os requisitos definidos foram atendidos. Devido a sua importância no processo de produção o capitulo seguinte abordará os requisitos de software.

2.3.2 Requisitos de *Software*

Segundo BRAUDE (2002) a análise de requisitos é o processo de entender e colocar no papel, uma declaração do que uma aplicação destina-se a fazer depois de construída.

Para KOSCIANSKI, MARTINS (2007) os requisitos de software são as descrições sobre seu comportamento, funções e especificações das operações que deve realizar e especificações sobre suas propriedades ou atributos.

Os atributos compreendem as funcionalidades presentes no software quando estiver pronto para ser executado (KOSCIANSKI e MARTINS, 2007).

Os requisitos de software podem ser definidos como “o que” o cliente deseja que o software realize. A qualidade dos requisitos depende das informações que o usuário precisa passar para o analista e da experiência do analista para conseguir requisitos.

A figura a seguir mostra como os requisitos de software podem ser vistos pelos analistas e pelos usuários.

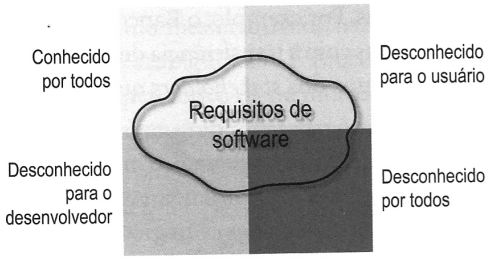

Figura 1: Visibilidade dos requisitos implícitos e explícitos.
Fonte: Koscianski e Martins (2007) pág. 176

De acordo com a figura 1 pode-se analisar que existem 4 quadrantes que representam os requisitos de um software o que representa os requisitos conhecidos por todos, os que somente o usuário conhece e deverá passar ao analista, os que somente o analista percebe e é passado para o cliente e o que ambos desconhecem.

Segundo KOSCIANSKI e MARTINS (2007) realizar corretamente o levantamento e administração dos requisitos é vital para a qualidade do software.

As principais dificuldades de falhas em projetos são relativas aos requisitos. Essas falhas se devem a dificuldade em entender o que o usuário quer, descrições incompletas e mudanças não controladas nos requisitos. (STANDISH GROUP, 1995)

Durante o desenvolvimento de um software os requisitos sofrem influência de diferentes pessoas ou grupos de pessoas como gerentes, operadores, analistas, programadores, leis, entre outros.(KOSCIANSKI e MARTINS, 2007)

Segundo MORAES (2011), as dificuldades encontradas na fase de levantamento de requisitos são: o usuário principal do sistema não sabe o que quer que o sistema faça ou sabe e não consegue transmitir para o analista; requisitos identificados, mas que não são realistas e não identificam os requisitos similares informados por pessoas diferentes.

A automatização de algumas tarefas podem diminuir a carga de trabalho das pessoas e, ao mesmo tempo, garantir uniformidade: quando é uma ferramenta que executa a tarefa, há menos chances de o resultado ser diferente porque diversas pessoas a utilizaram.

Os métodos e ferramentas de engenharia de software servem, entre outras coisas, ao propósito de garantir, ou pelo menos facilitar, a obtenção do objetivo de ter qualidade nos programas.

Para realizar o levantamento de requisitos de maneira mais adequada o analista pode aplicar técnicas para tratar alguns problemas que podem ser encontrados como:

- O *Stakeholder* mais influente, como um gerente, poderá impor requisitos que não sejam ideais ao software, pois ele pode não ser a pessoa que realiza aquela função;
- Não existir cooperação e a especificação gerada não é fiel;
- Falta de tempo para a especificação de requisitos;
- O desenvolvedor não está preparado para a fase de análise;
- Alguns *stakeholders* podem simplesmente não querer que o sistema seja desenvolvido.

Como o analista precisará do maior número de requisitos possíveis deverá ter a habilidade de comunicação para abstrair todos os requisitos existentes. Segundo KOSCIANSKI E MARTINS (2007) um desenvolvedor pode ter a sorte de encontrar um cliente com habilidades pedagógicas extraordinárias, que são motivados a dividir seu conhecimento e dão verdadeiras aulas sobre suas profissões. Por outro lado existem os que não gostam de descrever sua função, ou não conseguem organizá-la para explicitar.

Não existe um processo ideal de levantamento de requisitos que seja adaptável a todas as empresas KOSCIANSKI E MARTINS (2007).

Segundo PRESSMAN (1997) e SOMMERVILLE (1995) o processo de extração de requistos está dividido em 4 fases.

- Entendimento do domínio: os desenvolvedores devem entender o domínio da aplicação (documentos, livros, pessoas).
- Extração e análise de requisitos: acontece a descoberta, revelação e entendimento dos requisitos, através de interação com o(s) usuário(s); e desenvolvedores envolvendo:

  – Classificação e organização dos requisitos;

– Determinação de suas prioridades;
– Resolução de inconsistência;
– Conflitos e a descoberta de omissões.

- Especificação dos requisitos: armazenamento dos requisitos em uma ou mais formas, incluindo linguagem formal ou semiformal, representações simbólicas ou gráficas.
- Validação dos requisitos: verificação dos requisitos, visando determinar se estão completos com as necessidades do usuário.

Existem diversas técnicas de identificação de requisitos, e que são adequadas a diferentes situações, segundo KOSCIANSKI E MARTINS (2007) as principais são:

Entrevistas e Questionários

Entrevistas e Questionários é talvez a técnica mais simples de utilizar. Ainda que seja bastante eficaz numa fase inicial de obtenção de dados e esclarecimento de algumas dúvidas, está condicionada a alguns fatores:

- Influência do entrevistador nas respostas do cliente: convém que o entrevistador dê margem ao entrevistado para expor as suas ideias sem as enviesar logo à partida.
- Relação pessoal entre os intervenientes na entrevista.
- Predisposição do entrevistado: caso, por exemplo, o papel do entrevistado venha a ser afetado pela introdução de um sistema na organização, este pode propositadamente dificultar o acesso à informação.
- Capacidade de seguir um "plano" para a entrevista: na ausência destes planos é natural que haja tendência para que os intervenientes se dispersem um pouco, levando a que a entrevista demore mais tempo do que seria suposto. Caso a entrevista se torne demasiado longa, as pessoas podem cair na tentação de "querer despachar" sendo os últimos pontos da entrevista abordados de forma superficial (ou podem nem chegar a ser abordados).

Workshops de requisitos

O Workshop de Requisitos consiste numa técnica usada através de uma reunião estruturada, da qual devem fazer parte um grupo de analistas e um grupo representando o cliente, para então obter um conjunto de requisitos bem definidos. Ao contrário das reuniões, promove-se a interação entre todos os elementos presentes no workshop fomentando momentos de descontração como forma de

dinamizar o trabalho em equipe, existindo um facilitador neutro cujo papel é conduzir o workshop e promover a discussão entre os vários intervenientes (ainda que não tenha realmente poder de decisão). As tomadas de decisão devem seguir processos bem definidos e devem resultar de um processo de negociação, mediado pelo facilitador. Uma técnica que é também útil em workshops é o uso de brainstorming (tempestade de ideias) como forma de gerar um elevado número de ideias numa pequena quantidade de tempo. Como resultado dos workshops deve ser produzida documentação que reflita os requisitos e decisões tomadas sobre o sistema a implementar

Cenários (Série de Eventos Hipotéticos)

Uma forma de levar as pessoas a imaginarem o comportamento de um sistema é o uso de cenários. Através de exemplos práticos descritivos do comportamento de um sistema, os seus usuários podem comentar acerca do seu comportamento e da interação que esperam ter com ele. Trata-se de uma abordagem informal, prática e aplicável a qualquer tipo de sistema. De um modo geral, os cenários devem incluir os seguintes elementos:

- estado do sistema no início do cenário;
- sequência de eventos esperada (na ausência de erros) no cenário;
- listagem de erros que podem ocorrer no decorrer dos eventos do cenário e de como estes erros serão tratados;
- outras atividades que podem ser executadas ao mesmo tempo em que as deste cenário;
- estado do sistema depois de o cenário terminar.

Prototipagem

O uso de prototipagem é feito em diversas fases do processo de engenharia de requisitos (por exemplo na identificação, análise e validação). Trata-se de uma versão inicial do sistema, baseada em requisitos ainda pouco definidos, mas que pode ajudar a encontrar desde cedo falhas que através da comunicação verbal não são tão facilmente identificáveis. Neste tipo de abordagem apenas são desenvolvidas algumas funcionalidades sendo normalmente desenvolvidas primeiro aquelas que são mais fáceis de compreender por parte do utilizador e que lhe podem trazer maior valor acrescentado (principalmente na prototipagem evolutiva, isto é, aquela que mais tarde é evoluída para a fase de desenvolvimento). O uso de protótipos deve ser considerado apenas mediante uma análise custo-benefício, já

que os custos de desenvolvimento de um protótipo podem facilmente crescer, sendo particularmente úteis em situações em que a interface com os usuários é, para eles, um aspecto crítico.

Estudo etnográfico

Os Estudos Etnográficos são uma análise de componente social das tarefas desempenhadas numa dada organização. Quando um dado conjunto de tarefas se torna rotineiro para uma pessoa, é de se esperar que esta sinta dificuldade em articular todos os passos que segue ou todas as pessoas com as quais interage para levá-las a cabo. Através de uma observação direta das atividades realizadas durante um período de trabalho de um funcionário é possível encontrar requisitos que não seriam observáveis usando técnicas convencionais. Esta observação pode ser acompanhada de registros áudio/vídeo, porém não convém usá-los em demasia visto que o tempo necessário para processá-los pode ser demasiado. Nesta técnica assume-se que o representante do cliente observado desempenha as suas funções "corretamente", pelo que convém ter algum cuidado na escolha do mesmo.

As técnicas podem ser escolhidas de acordo com a empresa, pois o analista encontra pessoas diferentes para obter os requisitos, de acordo com cada pessoa muda a maneira de se comunicar e as técnicas a utilizar.

Nestas circunstâncias os analistas precisam conversar com diversas pessoas ligadas ao processo, cada uma com uma informação que futuramente poderá se transformar em requisito e utilizando uma técnica para cada um. Controlar este número de informações se torna cada vez mais difícil de acordo com o numero de pessoas e técnicas envolvidas. E considerando a dificuldade de se abstrair o conhecimento através da comunicação que por si já é difícil se torna necessário o uso de uma metodologia especifica para a analise de requisitos.

Atualmente não existe nenhuma metodologia que auxilie nas etapas da comunicação com o cliente em Engenharia de Software. A criação de um processo envolvendo etapas para a comunicação e entendimento auxiliaria o analista a conseguir informação e organizar estas informações transformando-as em requisitos. Este processo não existe em Engenharia de software porém em gestão do Conhecimento ele é utilizado. A seguir será explanado o conceito de gestão do conhecimento e suas práticas.

2.4 GESTÃO DO CONHECIMENTO

DAVENPORT E PRUZAK (1999) definem o conhecimento como uma mistura fluida de experiência que contempla valores, informação contextualizada e insight experimentado, a qual proporciona uma estrutura para avaliação e incorporação de novas experiências e informações. Ele tem origem e é aplicado na mente dos conhecedores. Nas organizações, ele costuma estar embutido não só em documentos ou repositórios, mas também em rotinas, processos, praticas e normas organizacionais. O conhecimento deriva da informação da mesma forma que a informação deriva dos dados.

MIRANDA (2004), define o conhecimento como um conjunto de saber na informação que se torna justificada, verdadeira e confiável, assumindo o caráter cumulativo e compõe-se de duas vertentes: a tácita, própria do individuo e, portanto, subjetiva e a explicita, externa ao indivíduo, também denominada conhecimento objetivo.

MARIN-GARCIA E ZARATE-MARTINEZ (2007) descrevem a Gestão do Conhecimento (GC) como uma arte em que a informação e o capital intelectual são transformados em valores permanentes para a organização bem como para seus parceiros e clientes. Eles também definem como um processo em que o uso da tecnologia da informação proporciona uma combinação sinérgica entre dados e informação, e a capacidade criativa e inovação dos seres humanos pelo outro, formando um grupo complexo de habilidades dinâmicas em constante mudança.

Segundo SALAZAR (2002), a GC pode ser entendida como um processo sistemático de detectar, selecionar, organizar, filtrar, compartilhar, adquirir e utilizar a informação dentro da empresa. Já SETZER(1999), diz que “o conhecimento é uma abstração interior, pessoal de algo que foi experimentado, vivenciado por alguém”. Neste sentido o que se descreveria seria a informação através do conhecimento que foi explicitado.

Para NONAKA E TAKEUCHI (1997), o “conhecimento humano é criado e expandido através da interação social entre conhecimento tácito e o conhecimento explícito”.

Para YOUSSEF (2003) uma vez assumido que a maior parte do

conhecimento é tácito, torna-se mais difícil compartilha-lo com outras pessoas ou organizações. Diz também que o fato de a maior parte do conhecimento ser tácita não invalida a necessidade de torna-la explicita.

Ao analisar os conceitos anteriores pode-se perceber que o conhecimento da organização pode ser encontrado de diversas maneiras, porém o conhecimento mais rico é o que se encontra na mente das pessoas, o conhecimento tácito, e é este conhecimento tácito o mais importante e o mais difícil de ser explicitado.

Segundo OECD (2000) existem limites naturais que impedem a explicitação de conhecimento tácito. Por este motivo observa ser importante codificar o conhecimento para poder compartilha-lo com um maior numero de pessoas, enquanto o conhecimento não codificado permanece com o seu detentor e aprendiz.

A seguir será apresentado maneiras de se realizar esta conversão de conhecimentos.

2.4.1 Conversão do Conhecimento

DAVENPORT E PRUZAK (1999) questionam que a GC é a forma de codificar o conhecimento existente e disponível na organização a fim de torna-lo acessível aquele que precisa dele, por meio de funções e qualificações para desempenhar o trabalho de aprender, distribuir e usar o conhecimento.

DEPRES E CHAUVEL(1999) estabelecem 6 processos para o processo da GC como mostra a Figura 2.

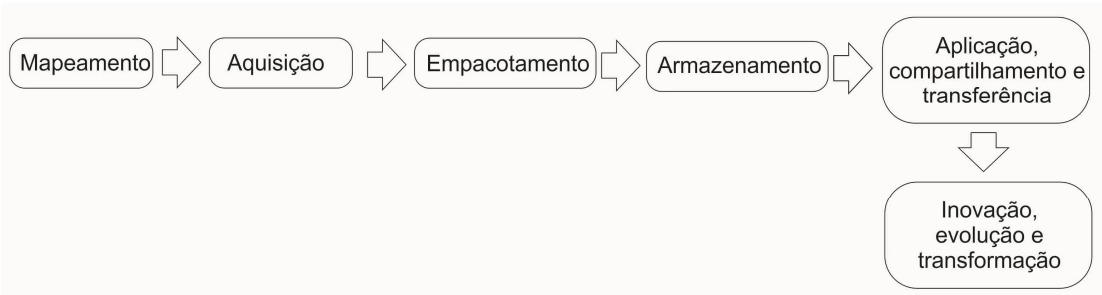

Figura 2 - Processos da GC

Fonte: Autoria própria

Já PROBST, RAUB E ROMBAERT(2002), delinearam os seguintes passos para a GC conforme apresenta a figura 3.

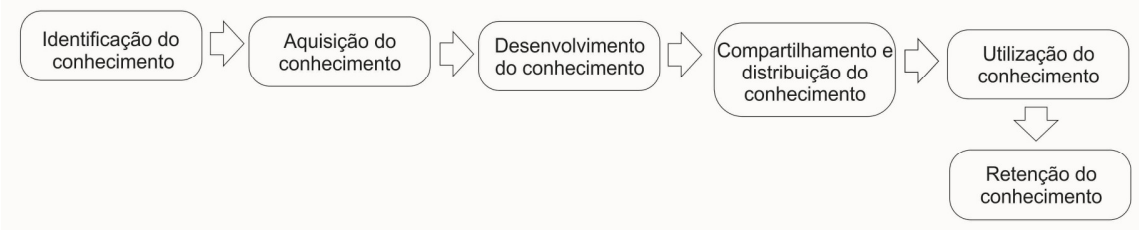

Figura 3 - Passos da GC

Fonte: Autoria própria

Segundo NONAKA E TAKEUCHI(1997) o processo de criação de conhecimento ocorre quando a espiral do conhecimento (figura 4) movimenta-se entre as duas dimensões provocando a interação entre os conhecimentos (tácito e explícito) e entre os níveis de conhecimento (o conhecimento é sempre gerado pelos indivíduos sendo ampliado intraorganizacionalmente e interorganizacionalmente). Tal processo, conforme NONAKA E TAKEUCHI(1997) é operacionalizado pela "conversão do conhecimento" em quatro modos:

Figura 4: A espiral do conhecimento

Fonte: Nonaka e Takeuchi (1997, p. 80)

- Socialização: é o processo no qual as experiências baseadas em modelos mentais ou habilidades pessoais são compartilhadas para criação de novos conhecimentos tácitos. Pode ser representado através de técnicas de observação, imitação e práticas de relacionamento entre aprendizes e mestres. No meio organizacional, a socialização ocorre por meio de atividades, treinamentos, interações com clientes, sessões informais, *"brainstorms"* entre outros.

- Externalização: representa o processo de transformação do conhecimento tácito

em explícito sendo expresso na forma de metáforas, analogias, conceitos, hipóteses ou diálogos. O conhecimento decorrente da externalização e torna-se facilmente transmissível e articulo, geralmente construído a partir de palavras ou números. A externalização é a chave para a criação do conhecimento pois cria modelos novos e explícitos a partir do conhecimento tácito.

- Combinação: é o processo de conversão do conhecimento explícito em explícito. Ocorre por meio da troca e combinação de conjuntos diferentes de conhecimento explícito. A troca e a combinação entre os indivíduos ocorrem sobretudo por meio de documentos, reuniões, conversas ao telefone ou redes de comunicação computadorizadas.

- Internalização: processo que ocorre a conversão do conhecimento explícito em tácito, caracterizando-se pelo “aprender fazendo”. A verbalização e a diagramação sob a forma de documentos, manuais ou histórias orais relatando as experiências dos indivíduos são práticas de extrema relevância para a internalização. Após este processo o novo conhecimento deve ser socializado com outros colaboradores (na forma de conhecimento tácito) iniciando assim uma nova espiral da criação do conhecimento.

O tópico anterior mostrou a dificuldade existente na comunicação entre o analista e o cliente, e a necessidade de uma metodologia para guiar este processo para que se obtenha requisitos que levem a produzir software de maior qualidade. A partir disso, o próximo capítulo descreve uma metodologia que uniria estes dois conceitos, Análise de requisitos e Gestão do Conhecimento para resolução do problema.

3 METODOLOGIA PROPOSTA

A maioria das mudanças nos projetos ocorrem devido a alterações dos requisitos do software pelo usuário. Muitas vezes o usuário não sabe exatamente o que quer do sistema, ou não sabe explicitar o que deseja, KOSCIANSKI E MARTINS (2007). Este trabalho objetiva apresentar a proposta de uma nova metodologia, específica para a análise de requisitos, que será baseada em gestão do conhecimento, mais precisamente no espiral do conhecimento.

O objetivo desta metodologia é ser mais eficiente na análise dos requisitos, diminuindo assim as mudanças que podem ocorrer durante o projeto do software. A metodologia baseada em GC está dividida em 5 etapas, como mostra Figura 5.

Figura 5: Metodologia para Análise de Requisitos

Fonte: Autoria própria

A metodologia proposta está dividida em 4 FASES: Socialização, Externalização, Combinação e Internalização.

Fase 1: A primeira FASE da metodologia proposta é a Socialização, esta fase está dividida em 2 etapas: Análise de Viabilidade do Sistema e Mapeamento do Conhecimento conforme o quadro a seguir.

FASE 1 - SOCIALIZAÇÃO	
1ª ETAPA Análise de viabilidade do sistema.	- Deve ser desempenhada pelo Analista; - Ocorrerá na empresa; - Gerar um relatório de viabilidade;
2ª ETAPA Mapeamento do Conhecimento	- Deve ser desempenhada pelo analista; - Ocorrerá na Empresa - Gerar uma planilha de mapeamento do conhecimento;

Quadro 4 – FASE 1 Socialização

Fonte: Autoria Própria

A 1ª ETAPA deverá resultar em um relatório de viabilidade, para isso o analista irá as dependências da empresa conversar com o profissional que deseja o sistema. As seguintes perguntas devem ser respondidas nesta primeira etapa:

O sistema ira contribuir significativamente para os objetivos da organização?

Dadas as restrições tecnológicas, organizacionais (econômicas, políticas, ambientais, recursos disponíveis) e temporais associadas ao projeto, o sistema pode ser implementado?

Caso haja necessidade de integração entre diferentes sistemas, será que esta é possível?

Se o novo sistema não fosse implementado, quais seriam as alternativas para a organização?

Quais são os problemas que os sistemas atuais apresentam e como um sistema novo irá resolver estas falhas?

De que forma o sistema irá contribuir diretamente para os objetivos da organização?

É possível a integração com os outros sistemas da organização (de um ponto de vista tecnológico)? Com que facilidade que se consegue partilhar informação

entre estes sistemas?

Respondendo estas questões o Analista poderá verificar se é possível dar continuidade a Análise de requisitos passando para a 2ª etapa.

Não existe um padrão para realização desta fase, fica a critério do analista como vai documentar as questões respondidas.

A 2ª Etapa deverá resultar em uma Lista de stakeholders relevantes para o processo de Analise de requisitos, para isso o analista identificará através da primeira etapa quais setores da empresa participarão do processo de análise, e com o auxilio do Organograma apontará os profissionais que serão mais relevantes para esta etapa.

É nesta etapa que ocorre a maior comunicação entre o cliente e o funcionário, neste momento serão identificados as habilidades do funcionário, que estruturas ele precisa para desempenhar esta habilidade e que relacionamentos são necessários.

Estes 3 itens deverão ser organizadas em forma de planilha contendo o setor, o cargo e as habilidades e nas habilidades a estrutura e o relacionamento necessário. A figura 6 mostra um modelo de planilha.

Mapeamento de Competências		
Funcionário:		
Setor:		
Cargo:		
Habilidade	**Estrutura**	**Relacionamento**

Figura 6: Modelo de Planilha para Mapeamento de Competências

Fonte: Autoria própria

Fase 2: A segunda FASE da metodologia proposta é a Externalização. Esta fase está dividida em 3 etapas apresentadas no Quadro 5: elaboração de Documento de Requisitos provisório, validação da documentação e alterações do documento provisório.

FASE 2 - EXTERNALIZAÇÃO	
1ª ETAPA Elaboração de Documento de Requisitos provisório.	- Deve ser desempenhada pelo Analista; - Ocorrerá no local de trabalho do Analista; - Seleção de habilidades; - Gerar um documento de requisitos provisório.
2° ETAPA Validação da documentação	- Deve ser desempenhada pelo Analista; - Ocorrerá na empresa; - Feedback com o Stakeholder para validar tópicos do documento.
3ª ETAPA Alterações do documento provisório	- Deve ser desempenhada pelo Analista; - Ocorrerá no local de trabalho do Analista; - Gerar um documento de requisitos provisórios atualizado.

Quadro 5 – FASE 2 Externalização

Fonte: Autoria Própria

A 1ª Etapa devera resultar um documento de requisitos provisório. Este documento irá conter somente as informações relevantes ao processo de desenvolvimento de software na visão do analista. Estas informações foram selecionadas das planilhas com as habilidades dos funcionários.

No documento provisório as planilhas resultantes da primeira fase devem ser agrupadas por possíveis relacionamentos. O que dará noção do processo de execução das tarefas. A figura 7 apresenta um modelo deste documento.

Mapeamento de Competências	
Funcionário: Maria dos Santos	
Setor: Operacional	
Cargo: Enfermeira	
Habilidades	**Relacionamento**
Buscar paciente	Recepcionista

Mapeamento de Competências	
Funcionário: Ana da Silva	
Setor: At Cliente	
Cargo: Recepção	
Habilidades	**Relacionamento**
Informar Chegada	Enfermeira

Figura 7: possíveis relacionamentos

Fonte: Autoria própria

Na figura 7 pode-se perceber que existe uma relação entre as funções descritas, a habilidade que a enfermeira deve ter para buscar a paciente depende de a recepcionista informar sua chegada. Desta maneira o processo pode ser entendido de uma maneira mais clara pelo Analista.

A 2ª Etapa é realizada através da comunicação do Analista com o stakeholder para um feedback da primeira fase. O analista apresentará ao funcionário o novo documento com o que ele entendeu da função do funcionário no processo. Neste momento haverá uma validação das informações passadas pelo funcionário, é neste momento também que o analista esclarece dúvidas geradas na elaboração do documento provisório de requisitos.

Na 3ª Etapa o documento provisório de requisito que foi apresentado ao stakeholder para o feedback recebe as alterações gerando um novo documento atualizado.

Fase 3: A terceira Fase da metodologia proposta é a Combinação, esta fase está dividida em 4 etapas: elaboração de Documento de Requisitos provisório, validação da documentação de requisitos provisória, alteração da documentação incluindo requisitos do benchmarking como mostra o quadro a seguir.

FASE 3 - COMBINAÇÃO	
1ª ETAPA Benchmarking com outras Empresas do Ramo	- Deve ser desempenhada pelo Analista; - Ocorrerá em diversos locais; - Seleção de melhores práticas; - Gerar uma apresentação de processos ao gerente;
2° ETAPA Apresentação do benchmarking a gerência	- Deve ser desempenhada pelo Analista; - Ocorrerá na empresa; - Realizar uma apresentação das melhores práticas encontradas.
3ª ETAPA Validação da documentação de requisitos provisória	- Será realizado pelo analista; - Será desempenhada na empresa; - Apresentar documentação gerada pela 1ª fase; - Discutir processos realizados.
4ª ETAPA Alteração da documentação incluindo requisitos do benchmarking	- Será desempenhado pelo Analista - Alterar documentação de requisitos provisórias incluindo alterações informadas pela gerencia e melhores práticas.

Quadro 6 – FASE 3 Combinação

Fonte: Autoria Própria

Na 1ª Etapa o Analista deve pesquisar as melhores práticas no segmento da empresa, e propor algumas alterações nos processos da empresa. Estas propostas devem ser apresentadas a gerencia da empresa na 2ª Etapa. A forma como será apresentada é irrelevante.

Na 3ª etapa será validado a documentação de requisitos provisória, elaborada a partir do mapeamento do conhecimento dos funcionários, à gerencia com intuito de apresentar os processos de uma maneira mais sistemática para possível alteração no documento.

Na 4ª Etapa Será realizado a alteração do documento de requisitos provisório, conforme o desejado pela gerencia, e incluído melhores práticas de acordo com a aprovação da gerencia.

Fase 4: A quarta e ultima FASE da metodologia proposta é a Internalização, esta fase está dividida em 2 etapas: documentar os requisitos provisórios a partir de padrões de Análise de Requisitos, apresentar o Documento de requisitos aos integrantes do projeto de software, conforme mostra o Quadro 7.

FASE 4 - INTERNALIZAÇÃO	
1ª ETAPA Documentar os requisitos provisórios a partir de padrões de Análise de Requisitos.	- Deve ser desempenhada pela Equipe de desenvolvimento de software; - Ocorrerá no local de trabalho do analista; - Documento final de Requisitos;
2° ETAPA Apresentar o Documento de requisitos aos integrantes do projeto de software.	- Deve ser desempenhada pelo Analista; - Ocorrerá no local de trabalho do analista; - Realizar uma apresentação dos requisitos aos integrantes do projeto de software.

Quadro 7 – FASE 4 Internalização

Fonte: Autoria Própria

A 1ª etapa deve gerar o documento de requisitos final, este documento será apresentado de forma que seja entendido pela equipe de projetos. Na 2ª etapa este documento será apresentado a equipe de projetos que poderá dar andamento ao projeto do software.

A metodologia apresentada visa dar maior foco a análise dos requisitos através da gestão do conhecimento, as atividades desempenhadas pelas pessoas envolvidas no projeto ficam mais claras, podem ser aperfeiçoadas e aplicadas ao

projeto.

A maior dificuldade desta metodologia seria encontrar um profissional capacitado a realizar esta gestão, pois alem do conhecimento de analise de sistemas ela precisará de conhecimento da área de GC.

Esta metodologia limita as outras fases do projeto de software, pois bloqueia o projeto ate que todas as informações sejam colhidas, e só depois pode ser colocado em pratica e sofrer modificações. Neste trabalho a metodologia não foi aplicada devido a necessidade de delimitação do tema pelo curto tempo para termino.

Para que a metodologia fosse aplicada a uma empresa de pequeno porte, que em grau de dificuldade pode ser considerada a mais simples pois teria um numero reduzido de setores, funcionários e habilidades, necessitaria de um tempo de execução maior que o disponível. Preocupou-se então com a delimitação do tema.

O tempo de aplicação desta metodologia pode ser mensurado pela figura a seguir que apresenta uma tabela construída no Excel.

ESTIMATIVA DE TEMPO				
FASES	ETAPAS	VARIÁVEIS	VALOR	ACUMULO
FASE 1	ETAPA 1	PORTE DA EMPRESA PEQUENA:1 MÉDIA:2 GRANDE:3	1	3
	ETAPA 2	NUMERO DE FUNCIONÁRIOS?	20	40
FASE 2	ETAPA1			20
	ETAPA 2			3
	ETAPA 3			3
FASE 3	ETAPA 1			3
	ETAPA 2			1
	ETAPA 3			3
	ETAPA 4			3
FASE 4	ETAPA 1			40
	ETAPA 2			1
			TOTAL DIAS	120

Figura 8: Estimativa de Tempo para Aplicação da metodologia

Fonte: Autoria própria

Através da figura 8 pode-se perceber que o tempo de aplicação depende

diretamente do porte da empresa e do número de funcionários. A estimativa foi feita baseado no tempo médio de execução das fases da metodologia.

A metodologia depois de concluída beneficia as outras fases do projeto, pois apresenta de uma forma muito mais detalhada e completa as funcionalidades que o software deve conter, e após esta primeira apresentação as mudanças que podem ocorrer no projeto são reduzidas, pois não são identificados somente as funcionalidades desejadas, mas sim, identificado todo o processo dando mais confiabilidade as especificações encontradas pelo ao analista, que se tornou o detentor do conhecimento.

4 CONCLUSÕES

Atualmente o processo de análise de requisitos leva em conta as funcionalidades que o usuário deseja em seu sistema, mas muitas vezes estas funcionalidades não são as mais relevantes para o sistema.

Este trabalho objetivou analisar os requisitos exigidos para considerar um software sendo de qualidade, no Capítulo 2.1 isto foi atendido através de autores e Organismos normativos que foram citados. Outro objetivo era identificar as principais metodologias que visam obter qualidade no software, no Capitulo 2.3 foi apresentado as principais metodologias para desenvolvimento do software e as ferramentas utilizadas para a análise de requisitos.

A problemática do trabalho pretendia resolver a dificuldade de se obter requisitos de software, e pode ser proposta através da apresentação de práticas de gestão do conhecimento que podem auxiliar na fase de análise de requisitos.

A metodologia proposta baseada nos objetivos citados procura aumentar a eficiência na fase de Análise de requisitos de software, pois tratam especificamente desta fase. A metodologia utiliza a gestão do conhecimento e das competências na empresa, e as melhores práticas neste setor para chegar as funcionalidades que deverão estar contidas no software.

Desta maneira os requisitos encontrados estarão altamente ligados aos processos da empresa, pois foram abstraídos de uma maneira mais confiável tornando o sistema altamente personalizado.

A personalização de um sistema leva a empresa a conseguir informações mais precisas, pois o sistema foi feito moldado nas atividades executadas em sua rotina se tornando um grande diferencial competitivo.

Isso se deve as 4 fases do espiral do conhecimento aplicadas a análise de requisitos, a forma de conseguir as informações é diferenciada sendo mais efetiva, pois ao passar pela socialização, onde a comunicação leva a externalização do conhecimento e em seguida é combinada com informações que fazem com que os requisitos possam ser utilizados de maneira mais eficiente na próxima fase do software, é um forte diferencial.

Se os requisitos do sistema não forem atendidos, em teoria a gestão do conhecimento e das competências da empresa está falho, precisando ser revisto e

definido antes de iniciar a próxima fase da ES. Nesta metodologia a gestão do conhecimento e a Analise de requisitos são altamente dependentes.

4.1 TRABALHOS FUTUROS

Este trabalho pode continuar sendo desenvolvido se junto a sua pesquisa for adicionado Engenharia reversa de sistemas, pois através da reengenharia pode-se analisar softwares já construídos conseguindo retornar no processo até a fase de analise de requisitos.

Através das especificações que foram utilizadas para sua produção pode-se definir se o sistema possui as funcionalidades desejadas. Estas funcionalidades interferem diretamente em sua qualidade.

Se o cliente observar que muitos itens não foram especificados na analise de requisitos, será necessária uma nova análise. A aplicação da metodologia pode fazer com que os requisitos sejam definidos de uma maneira diferente da primeira vez, a metodologia pode auxiliar na correção de problemas e aumentar a qualidade do software.

Outra linha pode ser seguida ao aplicar duas metodologias a um só cliente, e comparar seus resultado, sem que um analista tenha acesso aos requisitos do outro, assim poderá ser avaliado a eficiência da análise de requisitos no decorrer do projeto.

5 REFERÊNCIAS

BRAUDE, E. **Da programação a arquitetura: uma abordagem baseada em Java**. Bookman, 2002.

BATISTA, F. F. **O governo que aprende: gestão do conhecimento em organizações do executivo federal**. Brasília: IPEA, 2004 (Texto para Discussão nº 1022).

BATISTA, F. F.; QUANDT, C. O.; PACHECO, F. F. & TERRA, J. C. C. **Gestão do Conhecimento na Administração Pública**. Brasília: IPEA, 2005 (Texto para Discussão nº 1095).

BECK, K. **Programação Extrema Explicada.** Bookman, (1999).

BURNETT, Robert Carlisle. **Engenharia de requisitos: conceitos e fundamentos**, [S.l.], [1998]. Disponível em: <http://www.celepar.gov.br/batebyte/edicoes/1998/bb77/engenharia.htm>. Acesso em: 05/03/2013.

CAMPOS, V. F. TQC: **controle da qualidade total no estilo japonês**. 8ª ed. Nova Lima: INDG Tecnologia e Serviços, 2004

CORRÊA, H. L.; GIANESI, I. G. N. **Just in Time, MRP II e OPT: um enfoque estratégico**. Atlas S.A:São Paulo, 1996

CROSBY, P. B. **Qualidade é investimento**. Rio de Janeiro: José Olympio, 1992.

CROSBY, P.B. **Quality is free: the art of making quality certain**. New York: New American Library, 1979

DAVENPORT, T. H; PRUSAK, L. **Conhecimento empresarial: como as organizações gerenciam o seu capital intelectual**. Rio de Janeiro: Campus, 1998.

DESPRES, C. ; CHAUVEL, D. **How to map knowledge management.** Financial Times Mastering Information Management. Março 1999.

FEIGENBAUM, A.V. **Qualidade: a revolução da administração**. São Paulo, Marques Saraiva, 1991.

GARVIN, D.A. **Gerenciando a qualidade: a visão estratégica e competitiva**. Rio de Janeiro: Qualitymark, 2002

GIL, Antonio Carlos. **Métodos e técnicas de pesquisa social**. São Paulo: Atlas, 1999.

HATANO D. **As qualidades de um excelente analista de requisitos**. Disponível

em: <http://www.tiespecialistas.com.br/2012/03/as-qualidades-de-um-excelente-analista-de-requisitos/#.UTa7tKKHt9B>. Acessado em 06/03/2013.

JURAN, J. M.; GRYNA, F. M. **Controle de qualidade**. São Paulo, McGrawHill, 1991.

KOSCIANSKI, A. SOARES, M. S. **Qualidade de Software: Aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software**. Novatec, 2007. 2ª Ed.

LOPES, Sérgio Naddeo Dias. **Engenharia de requisitos: uma visão geral**, [São Paulo],[1999]. Disponível em: <http://www.ime.usp.br/~pnaddeo/EngenhariaDeRequisitos1.html>. Acesso em: 05/03/2013.

LAUDON, K. C.; LAUDON, J. P. **Sistemas de Informação Gerenciais.** 7 ed., São Paulo: Prentice-Hall, 2007.

MAFFEO, B. **Engenharia de Software e Especificação de Sistemas.** Campus, 1992.

MARTINS, J. C. C. **Técnicas para gerenciamento de projetos de software**. RJ brasport, 2007.

MARIN-GARCIA, J. & ZARATE-MARTINEZ, E. **A theoretical review of knowledge management and team working in the organizations**. ISSN 1750-9653, England, UK International Journal of Management Science and Engineering Management Vol. 2 (2007) No. 4, pp. 278-288

MIRANDA, R. C. R. **Gestão do Conhecimento estratégico: uma proposta de modelo integrado.** 268p. Tese (Doutorado em Ciência da informação). Universidade de Brasília: Departamento de Ciência da informação, 2004.

MORAES, J. B. D. **Técnicas para levantamento de Requisitos.** Revista Engenharia de Software 2. Disponível em: http://www.devmedia.com.br/artigo-engenharia-de-software-2-tecnicas-para-levantamento-de-requisitos/9151. Acessado em 01/08/2012.

NEVES J. M. **A implantação de tecnologias da informação como fator de competitividade nos sistemas produtivos e nos negócios**. XIII SIMPEP - Bauru, SP, Brasil, 6 a 8 de Novembro de 2006. Disponível em: http://www.simpep.feb.unesp.br/anais/anais_13/artigos/251.pdf Acessado em: 02/08/2012.

NONAKA, I; TAKEUCHI, H. **Criação de conhecimento na empresa**. Rio de Janeiro: Elsevier, 1997. 13º Ed

OECD. **Knowledge Management in the learning Society and Skills**. Rue Andre-pascal, 75775 Pariz Cedex 16, France. OECD Publication Service, 2000.

PRESSMAN, R. S. **, Software engineering: A practitioner's approach**. 4th. ed. McGraw-Hill, 1997.

PRESSMAN, Roger S. Engenharia de software. São Paulo: Makron Books, 1995.

PROBST, G; RAUB, S; ROMHART, K. (2002) **Gestão do conhecimento: os elementos construtivos do sucesso**. Porto Alegre: Bookman.

REIS C. **Caracterização de um modelo de processo para projetos de software livre**. Mestrado (Dissertação). ICMS, Universidade de São Paulo, 2001.

REZENDE, D.A. **Engenharia de software e Sistemas de informação**. 3ª Ed revisada e ampliada - Rio de Janeiro Brasport, 2005.

ROCHA, A. R. C. et al. **Uma Experiência na Definição do Processo de Desenvolvimento e Avaliação de Software segundo as Normas ISSO.** Relatório Técnico ES-302/94, COPPE/UFRJ, Junho 1994.

ROCHA, Ana Regina Cavalcanti et al. **Qualidade de software. Teoria e prática**. São Paulo: Prentice Hall, 2001.

SALAZAR, A. P. **A Gestão do Conhecimento nas Organizações**. Artigo Técnico, Disponível no site: www.kmol.online.pt/artigos, janeiro 2002. Acessado em: 28/07/2012.

SCHWARTZ, J. I. ,**Construction of software. In: Practical Strategies for Developing Large Systems**. Menlo Park: Addison-Wesley, 1st. ed., 1975.

SENGE, P.M. **A quinta disciplina**. São Paulo: Best Seller, 1990.

SETZER, V.W. **Dado, Informação, Conhecimento e Competência**. Disponível em: www.ime.usp.br/~vwsetzer. Acessado em: 09/07/2012.

STÁBILE, S. Et all. **A comunicação do profissional de informática na prestação de serviços: um estudo de caso em uma empresa fornecedora de sistemas integrados de gestão (ERP).** XXV ENEGEP. Porto Alegre, 2005

SVEIBY, Karl Erik. **A Nova Riqueza das Organizações: gerenciando e avaliando patrimônios de conhecimento**. Rio de Janeiro: Campus, 1998.

SOMERVILLE, I. **Engenharia de software**. 6° ed. Tradução Maurício de Andrade. São Paulo: Ed Addison-Wesley, 2003.

SOMMERVILLE, I. **Software engineering**. 5th. ed. Addison-Wesley, 1995.

FALBO, R. A. **Integração de Conhecimento em um Ambiente de Desenvolvimento de Software.** Tese de Doutorado, COPPE/UFRJ, Rio de Janeiro, Brasil, 1998.

STEFANOVITZ, J.P. **Criação de Conhecimento e Inovação na Indústria de Alta Tecnologia: estudo e análise de casos em uma empresa do setor de automação industrial**. 2006. Dissertação (Mestrado em Engenharia de Produção) – Programa de Pós-Graduação em Engenharia de Produção, Escola de Engenharia de São Carlos, USP, São Carlos.

The Standish Group. **CHAOS: Pesquisa sobre o desenvolvimento de softwares e o panorama da indústria de tecnologia de Informação da Atualidade**. Disponível em www.standishgroup.com/chaos.html.

WEINBERG G.M. **Software com qualidade**. Makroon, 1994.

YOUSSEF, A. Y. **Um modelo de Gestão do Conhecimento em administração para uma organização universitária Interinstitucional.** 2010. 420f. Tese de Doutorado (Engenharia e Gestão do Conhecimento). Centro Tecnológico, Universidade federal de Santa Cataria, Florianópolis.