"A IMPLEMENTAÇÃO DA ENGENHARIA DE REQUISITOS COMO FATOR CRÍTICO DE SUCESSO PARA DESENVOLVIMENTO DE SOFTWARE".

MARCELO NOGUEIRA
Universidade Paulista
Rua Dr. Bacelar 1212 – 4º - CEP 04026-002 – São Paulo – SP.
marcelo@noginfo.com.br

Resumo

As empresas de desenvolvimento de sistemas têm seus recursos materiais e humanos tanto quanto sobrecarregados, diante da demanda a ela submetida. Porém a não implementação da engenharia de requisitos como fator crítico de sucesso no desenvolvimento de software, ou por falta de cultura ou até por desconhecimento da sua existência, levam projetos de suma relevância ao insucesso e aumentando ainda mais a quantidade de casos dos sistemas inacabados, conseqüentemente desperdiçando tempo, recursos financeiros e não atendendo as necessidades dos clientes e nem do desenvolvedor.

Palavras-chave: Engenharia de Requisitos, Engenharia de Software.

Abstract

The systems development companies have their material and human resources as much as overloaded, in front of the demand to her submitted. However the not implementation of the requirements engineering as critical factor of success in the software development, or for lack of culture or until for ignorance of your existence, carry projects of vanishes relevance to the failure and even increasing the quantity of cases of the unfinished systems, consequently wasting time, financial resources and not attending the needs to clients and neither of the developer.

Key words: Requirements engineering, Software Engineering.

1. OBJETIVO

Este trabalho tem por objetivo principal apresentar os métodos da engenharia de software, no âmbito da engenharia de requisitos, como processo fundamental e fator crítico de sucesso no desenvolvimento de software através de aplicação de um modelo estruturado para resolução de problemas / questões apresentadas pelos clientes / usuários.

Este artigo pretende contribuir com outros estudos de implementação da Engenharia de Requisitos para desenvolvedores de software.

2. INTRODUÇÃO

Num ambiente competitivo e de mudança cada vez mais complexo, a gestão adequada da Informação assume uma importância decisiva no processo de tomada de decisão nas organizações.

Tratando-se de um tema simultaneamente abrangente e especializado, a adoção da Engenharia de Requisitos como linha base da Gestão da Informação, possibilitará, não só desenvolver e consolidar

os conhecimentos no desenvolvimento de software, bem como prepara-los para encarar com confiança os novos desafios no mundo dos negócios, e também reforçar as competências profissionais, mantendo-se atualizado em relação ao potencial dos sistemas de informação e das novas tecnologias numa perspectiva empresarial e competitiva globalmente.

A partir do conhecimento adquirido da Engenharia de Requisitos de Software, o desenvolvedor será elemento multiplicador de soluções, contribuindo e agregando valor aos sistemas novos e para os já existentes, com aplicação de metodologias e tecnologias adequadas, capazes de gerir com sucesso as informações relevantes aos negócios aplicáveis, trazendo ás organizações, vantagens competitivas.

No estudo da Engenharia de Software, o autor Roger S. Pressman [PRESSMAN02], demonstra preocupação com a "Crise do Software" que atualmente ele intitula como "Aflição Crônica", chegando a determinar números expressivos sobre a não finalização de projetos de sistemas começados e não terminados.

Num mundo cada vez mais de recursos financeiros escassos, como é possível aceitar tal desperdício de tempo e dinheiro. O mesmo autor também aponta para o possível problema causador de tal absurdo: "A falta de adoção de métodos, ferramentas e procedimentos no desenvolvimento de software e a difícil relação de entendimento entre o usuário com o desenvolvedor".

Existem várias técnicas de levantamento de requisitos e modelagem de software e o desenvolvedor que não às implementam tem dificuldades de realizar um projeto de sistemas livre de manutenções e re-trabalhos, condenando diretamente a qualidade do produto.

A implementação da Engenharia de Requisitos, pelo desenvolvedor de software, direciona como realizar uma especificação de requisitos que venha estruturar a fase de análise do desenvolvimento do software, proporcionando método sistemático de levantamento dos requisitos acompanhando todo o processo, permitindo que o software venha representar a realidade da empresa modelada para geração de um sistema customizado, atendendo assim as necessidades desta empresa, obtendo qualidade no software, bem como criar a real possibilidade de extrair de um sistema, informações relevantes que venham não só para contribuir com a decisão, mas para ser um fator de excelência empresarial, permitindo novos negócios, permanência e sobrevivência num mercado atuante.

3. RELEVÂNCIA

Atualmente com a visão global permitindo a participação nas exportações de software para outros países, cada vez mais a qualidade no processo de desenvolvimento e do produto de software ganha maior observação e adoção das melhores práticas e soluções tecnológicas que atendam os requisitos estabelecidos.

Considerado por Brooks [BROOKS87] como problema essencial:

"A parte mais difícil do desenvolvimento de software é decidir precisamente o que será desenvolvido. Nenhuma outra parte do trabalho é tão difícil quanto estabelecer (definir) os detalhes técnicos necessários incluindo todas as interfaces para pessoas, máquinas e para outros sistemas de software. Nenhuma outra parte do trabalho é tão possível de ocasionar erros no sistema como essa. Nenhuma outra parte é tão difícil de ser posteriormente consertada".

Para compreender melhor a engenharia de requisitos, veremos alguns conceitos e objetivos a seguir da engenharia de software.

4. ENGENHARIA DE SOFTWARE

Segundo Rezende [REZENDE99], Engenharia é a arte das construções, embasada no conhecimento científico e empírico, adequada ao atendimento das necessidades humanas.

Engenharia de Software é a metodologia de desenvolvimento e manutenção de sistemas modulares, com as seguintes características [REZENDE99]:

- Adequação aos requisitos funcionais do negócio do cliente e seus respectivos procedimentos pertinentes;
- Efetivação de padrões de qualidade e produtividade em suas atividades e produtos;
- Fundamentação na tecnologia da informação disponível, viável e oportuna;
- Planejamento e gestão de atividades, recursos, custos e datas.

Segundo Pressman [PRESSMAN02], Engenharia de Software é:

- "O estabelecimento e uso de sólidos princípios de engenharia para que se possa obter economicamente um software que seja confiável e que funcione eficientemente em máquinas reais";
- "Descendente da engenharia de sistemas e de hardware. Abrange um conjunto de 3 elementos fundamentais (métodos, ferramentas e procedimentos), que possibilita, ao gerente, o controle do processo de desenvolvimento do software e oferece ao profissional uma base para a construção de software de alta qualidade".

Segundo Martin [MARTIN91], Engenharia de Software é:

- "É o estudo dos princípios e sua aplicação no desenvolvimento e manutenção de sistemas de software";
- "Tanto a engenharia de software como as técnicas estruturadas são coleções de metodologias de software e ferramentas";

Como conclusão, pode-se relatar que engenharia de software é um conjunto de práticas para desenvolvimento de soluções de software, ou seja, roteiro que pode utilizar diversas técnicas.

A seqüência de passos preestabelecidos permite optar e variar de técnicas e ferramentas na suas diversas fases [REZENDE99].

5. OBJETIVOS DA ENGENHARIA DE SOFTWARE

De um modo geral, considera-se que os objetivos primários da Engenharia de Software são o aprimoramento da qualidade dos produtos de software e o aumento da produtividade dos engenheiros de software, além do atendimento aos requisitos de eficácia e eficiência, ou seja, efetividade [MAFFEO92].

A Engenharia de Software visa sistematizar a produção, a manutenção, a evolução e a recuperação de produtos intensivos de software, de modo que ocorra dentro de prazos e custos estimados, com progresso controlado e utilizando princípios, métodos, tecnologia e processos em contínuo aprimoramento. Os produtos desenvolvidos e mantidos, seguindo um processo efetivo e segundo preceitos da Engenharia de Software asseguram, por construção, qualidade satisfatória, apoiando adequadamente os seus usuários na realização de suas tarefas, operam satisfatória e economicamente em ambientes reais e podem evoluir continuamente, adaptando-se a um mundo em constante evolução [FIORINI98].

Associando a esses objetivos, o termo engenharia pretende indicar que o desenvolvimento de software deve submeter-se a leis similares às que governam a manufatura de produtos industriais em engenharias tradicionais, pois ambos são metodológicos [MAFFEO92].

Com base nos objetivos da Engenharia de Software, fica evidente a necessidade da adoção de um modelo sistêmico para padronizar e gerenciar os processos de desenvolvimento de software.

6. CRISE DO SOFTWARE

Para generalizar o termo, ocorre quando o software não satisfaz seus envolvidos, sejam clientes e/ou usuários, desenvolvedores ou empresa [REZENDE99].

A expressão "Crise do Software", que começou a ser utilizada na década de 60, tem historicamente aludido a um conjunto de problemas recorrentemente enfrentados no processo de desenvolvimento (Construção, implantação e manutenção) de software [MAFFEO92].

Esses problemas não se referem apenas a programas que não funcionam.Na verdade, a chamada "Crise do Software" abrange todos os problemas relacionados a [REZENDE99]:

- Como sistemas computacionais são construídos;
- Como sistemas computacionais são implantados, referindo-se aqui ao processo de substituir sistemas antigos, desativando sistemas correntemente em operação, ou ao processo de instalar um sistema inteiramente novo;
- Como é provida a manutenção da quantidade crescente de software construído, associado a sistemas computacionais cada vez mais complexos;
- Como fazer face à crescente demanda para construção de software, visando satisfazer ao conjunto enormemente variado de anseios por informatização, atualmente detectado na sociedade moderna;
- Como administrar as questões comportamentais, envolvendo os clientes e/ou usuários e a política, cultura e filosofia empresarial.

Apesar da enorme variedade de problemas que caracterizam a crise do software, engenheiros de software e gerentes de projetos para desenvolvimento de sistemas computacionais tendem a concentrar suas preocupações no seguinte aspecto: "A enorme imprecisão das estimativas de cronogramas e de custos de desenvolvimento" (Tabelas 1, 2, 3 e 4) [LEE02].

A não implementação da engenharia de requisitos permite erros no levantamento dos requisitos do sistema, e somente percebidos na implantação do sistema, gerando assim uma fase de manutenção e correção do software interminável, excedendo no custo desta fase (Tabela 1) e alongando o prazo de término substancialmente (Tabela 2).

Tabela 1- Excedentes de Custo [LEE02].

Excedentes de Custo	
% Excedente de Custo	% de Respostas
<20%	15,5%
21% - 50%	31,5%
51% - 100%	29,6%
101% - 200%	10,2%
201% - 400%	8,8%
>400%	4,4%

Tabela 2- Excedente de Prazo [LEE02].

Excedente de Prazo	
% Excedente de Prazo	% de Respostas
<20%	13,9%
21% - 50%	18,3%
51% - 100%	20,0%
101% - 200%	35,5%
201% - 400%	11,2%
>400%	1,1%

O custo do desenvolvimento do software varia de acordo com cada fase de seu processo (Tabela 3).

No entanto com a implementação da engenhara de requisitos, os erros de levantamento de requisitos identificados na fase de análise do projeto de software, reduz o custo para correção (Tabela 4).

Tabela 3- Custos em projeto de software por fase de desenvolvimento [LEE02].

Custos em projeto de software por Fase de Desenvolvimento	
Etapa de Trabalho	**%**
Análise de Requisitos	3
Desenho	8
Programação	7
Testes	15
Manutenção	67

Tabela 4- Custos para correção de erros de software [LEE02].

Custos para Correção de Erros de Software				
Fase de desenvolvimento do software	***% de Desvios*** ($)	***Erros Introduzidos*** (%)	***Erros encontrados*** (%)	***Custo Relativo para*** **Correção**
Análise de Requisitos	5	55	18	1,0
Desenho	25	30	10	1,0 - 1,5
Teste do Código e da Unidade	10			
Teste de Integração	50	10	50	1,0 - 5,0
Validação e Documentação	10			
Manutenção Operacional		5	22	10 – 100

"Muitos desses erros poderiam ser evitados se as organizações dispusessem de um processo de engenharia de requisitos definido, controlado, medido e aprimorado. No entanto, percebe-se que para muitos profissionais de informática esses conceitos não são muito claros, o que certamente dificulta a ação dos gerentes no sentido de aprimorar os seus processos de desenvolvimento" [BLASCHEK03].

7. ANTICRISE DO SOFTWARE

Segundo Rezende [REZENDE99], pode-se resumir que a anticrise é a união e trabalho conjunto e harmonioso de três elementos: Empresa (Alta Administração), Cliente e/ou usuário e a unidade de informática (Desenvolvedores de soluções).

A Unidade de informática é um dos principais agentes de mudança nas organizações, preocupando-se com o negócio empresarial, auxiliando efetivamente os gestores nos processos de tomada de decisão, tanto operacionais, como gerenciais e estratégicas.

8. QUALIDADE DE SOFTWARE

Atingir um alto nível de qualidade de produto ou serviço é o objetivo da maioria das organizações. Atualmente não é mais aceitável entregar produtos com baixa qualidade e reparar os problemas e as deficiências depois que os produtos foram entregues ao cliente. [SOMMERVILLE03]

Segundo Machado [MACHADO01], para muitos engenheiros de software, a qualidade do processo de software é tão importante quanto à qualidade do produto. Assim na década de 90 houve uma grande preocupação com a modelagem e melhorias no processo de software. Abordagens importantes como as normas ISO 9000 e a ISO / IEC 12207, o modelo CMM (Capability Maturity Model) e o SPICE (Software Process Improvement and Capability dEtermination) sugerem que melhorando o processo de software, podemos melhorar a qualidade dos produtos.

A qualidade é conseqüência dos processos, das pessoas e da tecnologia. A relação entre e qualidade do produto e cada um desses fatores é complexa. Por isso, é muito mais difícil controlar o grau de qualidade do produto do que controlar os requisitos.[PÁDUA03]

Prevê-se que na primeira década dos anos 2000, após ajustarem seus processos para a produção de software de qualidade dentro de prazos e orçamentos confiáveis, as organizações serão pressionadas por seus concorrentes a reduzir substancialmente os prazos para a entrega de produtos.Organizações que sejam capazes de integrar, harmonizar e acelerar seus processos de desenvolvimento e manutenção de software terão a primazia do mercado [MACHADO01].

A globalização da economia vem influenciando as empresas produtoras e prestadoras de serviços de software a alcançar o patamar de qualidade e produtividade internacional para enfrentarem a competitividade cada vez maior. A norma internacional NBR ISO/IEC 12207 – Tecnologia da Informação – Processos de Ciclo de Vida de Software [ISO12207: 97] é usada como referência em muitos países, inclusive no Brasil, para alcançar esse diferencial competitivo [MACHADO01].

A norma ISO/IEC 12207 trata os requisitos do software como fator fundamental no processo de ciclo de vida do software, gerando documentação para a fase de desenvolvimento do software, possibilitando que o produto seja verificado e validado posteriormente, atendendo assim as necessidades do adquirente.

Diante deste fato, podemos afirmar que por falta de utilização de métodos e modelos de gerenciamento da qualidade de software com base em requisitos, produzimos softwares de qualidade contestável e participando efetivamente da "Crise do Software" [PRESSMAN02].

9. PRODUTO DE SOFTWARE

Quando entregamos a um cliente um pacote bem delimitado e identificado, podemos dizer que entregamos um produto [SPINOLA98].

A definição para produto de software segundo a norma IEEE-STD-610 [IEEE90] é:

"O conjunto completo, ou qualquer dos itens individuais do conjunto, de programas de computador, procedimentos, e documentação associada e dados designados para liberação para um cliente ou usuário final" [PAULK95].

10. PROCESSO DE SOFTWARE

O conceito de processo de software se baseia no conceito generalizado de processo, que pode ser definido como uma seqüência de estados de um sistema que se transforma [SPINOLA98].

O SEI (*Software Engineering Institute*), da *Carnegie Melon University* propõe o seguinte [SEI]:

“Um processo é uma seqüência de passos realizados para um dado propósito. Colocado de maneira mais simples, processo é aquilo que você faz. Processo é aquilo que as pessoas fazem, usando procedimentos, métodos, ferramentas, e equipamentos, para transformar matéria prima (entradas) em produto (saída) que tenha valor para o cliente”.[PAULK95]

“O processo de software pode ser definido como um conjunto de atividades, métodos, práticas, e transformações que as pessoas empregam para desenvolver e manter software e os produtos associados (ex. planos de projeto, documentos de projeto (design), código, casos de teste, e manual do usuário)”.[PAULK95]

11. REQUISITOS

Os problemas que os engenheiros de software têm para solucionar são, muitas vezes, imensamente complexos. Compreender a natureza dos problemas pode ser muito difícil, especialmente se o sistema for novo. Conseqüentemente, é difícil estabelecer com exatidão o que o sistema deve fazer. As descrições das funções e das restrições são os requisitos para o sistema [SOMMERVILLE03].

Os Requisitos (*Requirements*) podem ser definidos como as necessidades básicas do cliente, geralmente explicitadas como condição de negócio no contrato com o fornecedor. São características, tais como especificações técnicas, prazo de entrega, garantia, que o cliente “requer” do produto [MCT02].

“Uma condição ou capacidade necessitada por um usuário, para resolver um problema ou alcançar um objetivo” [IEEE90].

Segundo Tonsig, Os requisitos compõem o conjunto de necessidades estabelecido pelo cliente / usuário do sistema que definem a estrutura e comportamento do software que será desenvolvido [TONSIG03].

Segundo Pádua, o fluxo de requisitos reúne as atividades que visam a obter o enunciado completo, claro e preciso dos requisitos de um produto de software. Esses requisitos devem ser levantados pela equipe do projeto, em conjunto com representantes do cliente, usuários chaves e outros especialistas da área de aplicação [PADUA03].

Os requisitos de sistema de software são, freqüentemente, classificados com funcionais ou não funcionais ou como requisitos de domínio [SOMMERVILLE03]:

- Requisitos Funcionais: São declarações de funções que o sistema deve fornecer, como o sistema deve reagir a entradas específicas e como deve se comportar em determinadas situações. Em alguns casos, os requisitos funcionais podem também explicitamente declarar o que o sistema não deve fazer. Exemplo: O software deve permitir ampla consulta sobre os dados dos pedidos do cliente, inclusive via Internet [TONSIG03].
- Requisitos Não Funcionais: São restrições sobre os serviços ou as funções oferecidas pelo sistema. Entre eles destacam-se restrições de tempo, restrições sobre o processo de desenvolvimento, padrões, entre outros. Exemplo: O acesso aos recursos do software deve ser restrito a pessoas autorizadas [TONSIG03].
- Requisitos de domínio: São requisitos que se originam do domínio de aplicação do sistema e que refletem características desse domínio. Podem ser requisitos funcionais ou não funcionais. Exemplo: Deve haver uma interface-padrão com o usuário para todos os bancos de dados, que terá como base o padrão X [SOMMERVILLE03].

12.ENGENHARIA DE REQUISITOS

"O processo de descobrir, analisar, documentar, e verificar as funções e restrições do sistema, é chamado de engenharia de requisitos" [SOMMERVILLE03].

Engenharia de requisitos, uma subárea da engenharia de software, tem por objetivo tratar o processo de definição dos requisitos de software. Para isso estabelece um processo pelo qual o que deve ser feito é elicitado, modelado e analisado. Esse processo deve lidar com diferentes pontos de vista e usar uma combinação de métodos, ferramentas e pessoal. O produto desse processo é um modelo, do qual um documento chamado 'requisitos' é produzido. Esse processo é perene e acontece em um contexto previamente definido e que chamamos de 'Universo de informações' [LEITE01].

"O conjunto de técnicas empregadas para levantar, detalhar, documentar e validar os requisitos de um produto forma a engenharia de requisitos" [PADUA03].

A engenharia de requisitos fornece um mecanismo adequado para entender o que o cliente deseja, analisar as necessidades, avaliar a exeqüibilidade, negociar uma solução razoável, especificar a solução de maneira não-ambígua, validar a especificação e administrar os requisitos à medida que eles são transformados num sistema em operação. O processo da engenharia de requisitos pode ser descrito em cinco passos distintos [PRESSMAN02]:

- Elicitação de requisitos
- Análise e negociação de requisitos
- Especificação de requisitos
- Modelagem do sistema
- Validação de requisitos
- Gestão de requisitos

12.1 ELICITAÇÃO DE REQUISITOS

Certamente parece muito simples – pergunte ao cliente, aos usuários e a outros quais são os objetivos do sistema ou do produto, o que precisa ser conseguido, como o sistema ou o produto se encaixa nas necessidades do negócio e, finalmente, como o sistema ou o produto vai ser usado no dia-dia. Mas não é simples – é muito difícil.
"Na industria de software, as necessidades dos clientes são muito difíceis de serem acessados ou descobertos. Geralmente deve-se deixar margens nos sistemas para futuras mudanças. Os requisitos geralmente são vagos" [FREITAS00].

A seguir, vários problemas que nos ajudam a compreender por que a elicitação de requisitos é difícil [PRESSMAN02]:

- Problemas de escopo: O limite do sistema é mal definido. Detalhes técnicos desnecessários que podem confundir, em vez de esclarecer, os objetivos globais do sistema.
- Problemas de entendimento: Os clientes não estão completamente certos do que é necessário, têm pouca compreensão das capacidades e limitações de seu ambiente computacional e têm dificuldade de comunicar as necessidades ao engenheiro de sistemas, omitem informações que acreditam ser "óbvia", especificam requisitos que são ambíguos ou impossíveis de testar.
- Problemas de volatilidade: Os requisitos mudam ao longo do tempo.

Para ajudar a contornar esses problemas, os engenheiros de sistemas devem abordar a atividade de coleta de requisitos de um modo organizado.

Sommerville [SOMMERVILLE03] sugere um conjunto de diretrizes detalhadas para a elicitação de requisitos, que são resumidas nos seguintes passos:

- Avalie a viabilidade técnica e de negócios do sistema proposto.
- Identifique as pessoas que vão ajudar a especificar os requisitos e compreenda seu viés organizacional.
- Defina o ambiente técnico no qual o sistema ou produto vai ser colocado.
- Identifique "restrições de domínio" que limitam a funcionalidade ou o desempenho do sistema ou do produto a ser construído.
- Defina um ou mais métodos de elicitação de requisitos (entrevistas, reuniões etc.).
- Solicite a participação de muitas pessoas de modo que os requisitos sejam definidos de diferentes pontos de vista; certifique-se de identificar a razão de cada requisito que é registrado.
- Identifique requisitos ambíguos como candidatos para prototipagem.
- Crie cenários de uso para ajudar clientes / usuários a melhor identificar requisitos-chave.

Os produtos de trabalho produzidos como conseqüência das atividades de elicitação de requisitos vão variar de acordo com o tamanho do sistema ou do produto a ser construído. Para a maioria dos sistemas, os produtos de trabalho incluem:
- Uma declaração da necessidade e da viabilidade.
- Uma declaração delimitada de escopo do sistema ou do produto.
- Uma lista de usuários e interessados que participaram da atividade de elicitação de requisitos.
- Uma descrição do ambiente técnico do sistema.
- Uma lista de requisitos e as restrições do domínio que se aplicam a cada um.
- Um conjunto de cenários de utilização que fornece entendimento do uso do sistema ou do produto em diferentes condições de operação.
- Quaisquer protótipos desenvolvidos para melhor definir os requisitos.

Cada um desses produtos de trabalho é revisado por todo o pessoal que participou da elicitação de requisitos.

12.2 ANÁLISE E NEGOCIAÇÃO DE REQUISITOS

Uma vez reunidos os requisitos, os produtos de trabalho mencionados anteriormente formam a base para a análise de requisitos. A análise categoriza os requisitos e os organiza em subconjuntos relacionados; explora cada um em relação aos demais; examina-os quanto à consistência, omissões e ambigüidade; e ordena-os com base nas necessidades dos clientes / usuário [PRESSMAN02].

À medida que à medida que a análise de requisitos começa, as seguintes perguntas são formuladas e respondidas:
- Cada requisito está consistente com o objetivo global do sistema/produto?
- Todos os requisitos foram especificados no nível de abstração adequado?
- O requisito é realmente necessário, ou pode ser essencial para o objetivo do sistema?
- Cada requisito é limitado e não-ambíguo?
- Cada requisito tem atribuição? Isto é, uma fonte está associada a cada requisito?
- Algum requisito conflita com outros requisitos?
- Cada requisito é realizável no ambiente técnico que vai alojar o sistema ou o produto?
- Cada requisito pode ser testado, quando estiver implementado?

Não é incomum que os clientes / usuários peçam mais do que pode ser conseguido, considerando os recursos limitados do negócio.

O engenheiro de sistemas precisa reconciliar esses conflitos por intermédio de um processo de negociação. Os riscos associados a cada requisito são identificados e analisados. Estimativas grosseiras do esforço de desenvolvimento são feitas e usadas para avaliar o impacto de cada

requisito no custo do projeto e no prazo de entrega. Usando uma abordagem iterativa, requisitos são eliminados, combinados ou modificados de modo que cada parte alcance algum grau de satisfação.

12.3 ESPECIFICAÇÃO DE REQUISITOS

No contexto de sistemas baseados em computador, o termo especificação significa coisas diferentes para pessoas diferentes. Uma especificação pode ser um documento escrito, um modelo gráfico, um modelo matemático formal, uma coleção de cenários de uso, um protótipo ou qualquer combinação desses elementos. Para sistemas grandes, um documento escrito, combinando descrições em linguagem natural e modelos gráficos podem ser a melhor abordagem. Cenários de uso podem, entretanto, ser tudo que é necessário para produtos ou sistemas pequenos que residem em ambientes técnicos bem-entendidos [PRESSMAN02].

12.4 MODELAGEM DO SISTEMA

Cada sistema baseado em computador pode ser modelado como uma transformação de informação usando um gabarito entrada – processamento – saída. Essa visão pode ser estendida para incluir duas características adicionais dos sistemas – processamento e manutenção de interface com o usuário e autoteste [PRESSMAN02].

12.5 VALIDAÇÃO DE REQUISITOS

Os produtos de trabalho produzidos como conseqüência da engenharia de requisitos são avaliados quanto à qualidade durante o passo de validação. A validação de requisitos examina a especificação para garantir que todos os requisitos do sistema tenham sido declarados de modo não-ambíguo; que as inconsistências, omissões e erros tenham sido detectados e corrigidos e que os produtos de trabalho estejam de acordo com as normas estabelecidas para o processo, projeto e produto [PRESSMAN02].

12.6 GESTÃO DE REQUISITOS

Gestão de requisitos é um conjunto de atividades que ajuda a equipe de projeto identificar, controlar e rastrear requisitos e modificações de requisitos em qualquer época, à medida que o projeto prossegue [PRESSMAN02].
“Geralmente, requisitos corretos no início do projeto são mudados posteriormente” [FREITAS00].

13. ERGONOMIA COGNITIVA

A Ciência Cognitiva, segundo Peters [PETERS01], é o estudo de como o conhecimento é adquirido, representado na memória e utilizado na resolução de problemas. A abordagem da ciência cognitiva à programação de computadores enfoca como os programadores representam o conhecimento, como as estruturas dos programas são aprendidas e como o conhecimento é aplicado no desenvolvimento de software.

A ergonomia cognitiva é o estudo das habilidades de resolução de problemas, análise de informações e procedurais relativas à influência dos fatores humanos. As questões dos fatores humanos relativos a estruturas de linguagem de programação, programação de computadores e interação homem-computador são respondidas de forma cognitiva em termos de esforço mental exigido.

A limitação da memória de curto prazo é o grave obstáculo ao desenvolvimento de programas de computador em grande escala. Isso significa que não é possível compreender tantos elementos ao

mesmo tempo a ponto de controlarem as várias conexões existentes entre os componentes de um sistema de software. O agrupamento é visto como uma forma de combater esse problema. O agrupamento é o processo de reunir itens com atributos semelhantes ou relacionados para formar um item único.

O agrupamento fornece a base para os modelos de compreensão de programas úteis na manutenção de software, ou seja, a compreensão do software é auxiliada por representações internas de um sistema com planos e esquemas. O processo de entendimento coincide documentos do sistema, como requisitos, aos planos de programação utilizando regras do discurso para selecionar planos. O resultado de cada coincidência de um documento externo com um plano é a representação mental atualizada, que é armazenada como um novo plano.

Concluindo, podemos detectar que a dificuldade de compreender e especificar requisitos em diferentes níveis de abstração vai defrontar com a nossa capacidade cognitiva de adquirir o conhecimento para criar soluções de software. O estudo da ergonomia cognitiva e sua aplicação juntamente com a implementação da engenharia de requisitos, serão fatores críticos de sucesso no desenvolvimento de software.

14. CONCLUSÃO

É importante que os desenvolvedores de software reconheçam que não é possível desenvolver sistemas com qualidade, cumprir prazos e custos e atender às expectativas dos usuários sem ter um processo de desenvolvimento de requisitos definido, compreendido e utilizado por toda a equipe.
Quanto à engenharia de requisitos podemos ainda acrescentar:

- Adequação aos princípios e objetivos da Engenharia de Software.
- O nível de complexidade da sua implementação pode ser dimensionada de acordo com o porte do sistema, viabilizando para as pequenas organizações desenvolvedoras de software.
- Possibilita então ser fator crítico para o sucesso no desenvolvimento de software.
- Pode ser utilizada alinhada as normas ISO de qualidade de software 12207, 9000-3.
- Com a sua implementação é possível contribuir para o fim da "Crise do Software".
- Alinhada a ciência cognitiva, permite transformar problemas em agrupamentos menores, facilitando a compreensão e criação de solução.
- Com os processos de desenvolvimento de software controlados, documentados e gerenciados o desenvolvedor poderá assumir projetos de alta complexidade, aliados a técnica e criatividade, pois terá mais chance de sucesso.
- Melhor capacitado e provedor de metodologias que levam ao desenvolvimento de software com qualidade, o desenvolvedor poderá criar soluções que atendam as necessidades e os requisitos da empresa, contribuindo para criação de vantagens competitivas, sustentando as bases estratégicas das organizações.

15. REFERENCIAS BIBLIOGRÁFICAS

[BLASCHEK03] BLASCHEK, JOSÉ ROBERTO. *Gerência de Requisitos, o principal problema dos projetos de software,* Artigo Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2003.
[BROOKS87] BROOKS, F. *Essence and accidents to software engineering.* Los Alamos, California, IEEE Computer Society, v.4, n.3, 1987.
[FIORINI98] FIORINI, SOELI T., et al. *Engenharia de Software com CMM*, Rio de Janeiro, Ed. Brasport, 1998.
[FREITAS00] FREITAS, LUÍS RICARDO NAPOLITANO, *Projetos em Tecnologia da Informação – Como acertar através da análise dos erros.* Dissertação de Mestrado, USP-SP, 2000.

[IEEE90] IEEE STD. 610 12-1990, *IEEE Standard Glossary of Software Engineering Terminology*, IEEE, Piscataway, NJ, 1997.
[ISO12207: 97] NBR ISO/IEC 12207:1997, *Tecnologia de Informação – Processos de Ciclo de Vida de Software*, Rio de Janeiro, ABNT – Associação Brasileira de Normas Técnicas.
[LEE02] LEE, RICHARD C. e TEPFENHART, WILLIAM M., *UML e C++ - Guia de desenvolvimento orientado a objeto*, São Paulo, Ed. Makron Books, 2002.
[LEITE01] LEITE, JULIO CESAR SAMPAIO DO PRADO, in WEBER, KIVAL CHAVES, et al. *Qualidade e Produtividade em Software*, São Paulo, Ed. Makron Books, 2001.
[MACHADO01] MACHADO, CRISTINA ÂNGELA FILIPAK in WEBER, KIVAL CHAVES, et al. *Qualidade e Produtividade em Software*, São Paulo, Ed. Makron Books, 2001.
[MAFFEO92] MAFFEO, BRUNO, *Engenharia de Software e Especificação de Sistemas*, Rio de Janeiro, Ed. Campus, 1992.
[MARTIN91] MARTIN, JAMES*,Engenharia da Informação*, Rio de Janeiro, Ed. Campus, 1991.
[MCT02] MINISTÉRIO DA CIÊNCIA E TECNOLOGIA, Secretaria de Política de Informática, *Qualidade e Produtividade no Setor de Software Brasileiro*, Brasília, N.4, 2002.
[PADUA03] FILHO, WILSON DE PÁDUA PAULA, *Engenharia de Software*, Rio de Janeiro, Ed. LTC, 2003.
[PAULK95] PAULK, M.C. et al. *The Capatibility Maturity Model – Guidelines for improving the software process*, Addison Wesley, SEI series, 1995.
[PETERS01] PETERS, JAMES F. et al. *Engenharia de Software,*Rio de Janeiro, Ed. Campus,2001.
[PRESSMAN02] PRESSMAN, ROGER S., *Engenharia de Software*, Rio de Janeiro, Ed. McGraw-Hill, 2002.
[REZENDE99] REZENDE, DENIS ALCIDES, *Engenharia de Software e Sistemas de Informações*, Rio de Janeiro, Ed. Brasport, 1999.
[SEI] SEI, *Software Engineering Institute*, *Carnegie Melon University, http://www.sei.cmu.edu.*
[SOMMERVILLE03] SOMMERVILLE, IAN, *Engenharia de Software*, São Paulo, Ed. Pearson Education, 2003.
[SPINOLA98] SPINOLA, MAURO DE MESQUITA, *Diretrizes para o desenvolvimento de software de sistemas embutidos*, Tese de Doutorado, USP - São Paulo, 1998.
[TONSIG03] TONSIG, SÉRGIO LUIZ, *Engenharia de Software,* São Paulo, Ed. Futura, 2003.