Capítulo 1

CICLO DE VIDA DO SOFTWARE

1. Introdução

Qualificar um produto é muito bom para que tenhamos certeza de que há seriedade e preocupação com a satisfação em tê-lo, mas, qualificar o processo de produção é mais importante para obter um produto melhor. Ambas as qualificações (da produção e do produto) são largamente utilizados na produção de muitos produtos inclusive no desenvolvimento de softwares.

Hoje, temos normas da ISO 9003 que certificam o processo de produção de software bem como o software pronto. Tais normas exigem cada vez mais qualidade no gerenciamento do projeto e tais exigências são convertidas em benefícios para os usuários e desenvolvedores. Todo desenvolvimento de um software é caracterizado por fases que quando colocados em seqüência obtêm-se um Ciclo de Vida do Sistema e é este ciclo de vida que deve ter qualidade.

Conforme observações feitas durante os últimos 20 anos, em centenas de organizações que variam de tamanho de algumas até milhares de pessoas, com uma faixa de 1 até 1000 projetos simultaneamente a caminho.

Como pode ser esperado, as menores firmas costumam ser relativamente informais: os projetos são iniciados como resultado de uma discussão verbal entre o usuário e o gerente do projeto, e o projeto prossegue da análise de sistemas até o projeto e implementação sem muita algazarra. Em grandes organizações, no entanto, as coisas são feitas em uma base muito mais formal.

As várias comunicações entre os usuários gerência e equipe do projeto costumam ser documentados por escrito, e todos entendem que o projeto passará por diversas fases antes que seja terminado. Ainda assim existe grandes diferenças entre o modo com que dois gerentes de projetos na mesma organização conduzem seus respectivos projetos. Normalmente fica a cargo do gerente de projeto determinar de que fases e atividades o seu projeto consistirá e como essa fases serão conduzidas.

1.1. Definição de um ciclo de vida

Recentemente, no entanto, o método assumido para o desenvolvimento de sistemas começou a mudar. Mais e mais grandes e pequenas organizações estão adotando um único e uniforme ciclo de vida do projeto, ou simplesmente "o modo com que fazemos as coisas por aqui". Normalmente contido em um livro tão exuberante quanto o manual de padrões que se encontra (fechado) a mesa de cada programador, o ciclo de vida documentado do projeto fornece uma forma comum para que todos na organização passem a entender como pode ser desenvolvido um sistema de computador.

O método pode ser caseiro, ou alternativamente, a organização pode decidir comprar um pacote de gerenciamento de projeto e depois moldá-lo às necessidades da companhia. Deve ser aparente que, além de fornecer emprego para as pessoas que criam manuais de ciclo de vida de projetos, a metodologia de projeto é desejável. Qual, então, é a finalidade de se ter um ciclo de vida de projeto? Há três objetivos principais:

1. Definir as atividades a serem executadas em um projeto.
2. Introduzir a coerência entre muitos projetos na mesma organização
3. Fornece pontos de checagem para controle de gerência e pontos de checagem para a decisão "ir / não ir".

O primeiro objetivo é particularmente importante numa grande organização em que novas pessoas estão constantemente, entretanto nas fileiras da gerência de projeto. O gerente de projeto inexperiente pode não examinar ou subestimar a significância de importantes fases do projeto se seguir apenas sua própria intuição.

Na verdade, pode acontecer que os programadores e analistas de sistemas júnior não entendam onde e como os seus esforços se encaixam no projeto de um modo geral, a menos que tenham recebido uma descrição apropriada de todas as fases do projeto.

O segundo objetivo é importante em uma grande organização. Para os níveis mais altos de gerência, pode ser extremamente desconcertante tentar supervisionar 100 projetos diferentes, cada um deles sendo executado de uma forma diferente.

O terceiro objetivo de um ciclo de vida padrão do projeto refere-se à necessidade de gerência em controlar um projeto. Em projetos triviais, o único ponto de checagem é provavelmente o fim do projeto: ele foi terminado dentro do prazo e com o orçamento previsto? Mas, para grandes projetos, a gerência deve ter uma série de pontos de checagem intermediários, gerando oportunidade para determinar se o projeto está atrasado e se precisam ser obtidos recursos adicionais. Além disso, um usuário inteligente também pedirá pontos de checagem em diversos estágios no projeto, de modo que possa determinar se quer continuar prorrogando o prazo!

Apesar de tudo isto o ciclo de vida não irá fazer o projeto, mais ele ajudará a organizar as atividades, tornando mais provável que você atinja os problemas exatos na hora exata.

1.2. Escolha do Ciclo de Vida

Não existe uma regra ou um ciclo de vida padrão para desenvolvimento de sistema. Você pode usar um ciclo pré-definido por um determinado autor, pode usar o ciclo de um autor e moldá-lo conforme seu trabalho ou ainda criar seu próprio ciclo.
Porém, a escolha deve ser feita analisando os fatores:

- Características do fornecedor (equipe de desenvolvimento)
- Habilidade - Analisar os conhecimentos técnicos da equipe.
- Visão - A equipe deve ter conhecimento do domínio do problema a ser informatizado.
- Recursos - Equipamentos e material humano
- Tempo - Tempo disponível para o desenvolvimento
- Características do cliente (usuário)
- Visão - Conhecimento de suas necessidades
- Tempo - Aceitável para a implantação
- Recurso financeiro - Ter disponibilidade financeira para o investimento em software e hardware
- Insegurança - Confiabilidade nos resultados a serem gerados pelo sistema

Mesmo adotando um modelo de desenvolvimento, se os itens acima não forem bem avaliados, a qualidade do produto final poderá não ser satisfatório. Tanto modelagem estrutural quanto na modelagem orientada existem muitos ciclos. A seguir serão apresentados os mais usuais.

2. Modelo em Cascata (*Waterfall*)

Este abordagem baseia-se no modelo cascata ou método linear de desenvolvimento (fig.1). Podem ser utilizados conceitos de Engenharia de Software, a qual prevê atividade de verificação (estamos fazendo o produto de forma correta?), validação (estamos fazendo o produto certo?) e de controle de qualidade.

O ciclo é representado pelas seguintes fases :

1. Levantamento: definição preliminar do escopo do sistema, restrições e conceitos alternativos;
2. Análise: especificação funcional do sistema (Projeto Lógico);
3. Projeto: especificação completa da arquitetura de hardware e software, estruturas de controle, estruturas de dados do sistema, interfaces;
4. Codificação: codificação e teste individual dos programas;
5. Teste : teste dos componentes integrados do sistema;
6. Implantação : implantação de maneira gradativa, a fim de evitar insatisfação e possibilitando a correção do sistema. Implantação piloto / paralela e definitiva;
7. Operação e Manutenção : utilização do sistema e modificações decorrentes de erros,mudança de necessidades, etc.

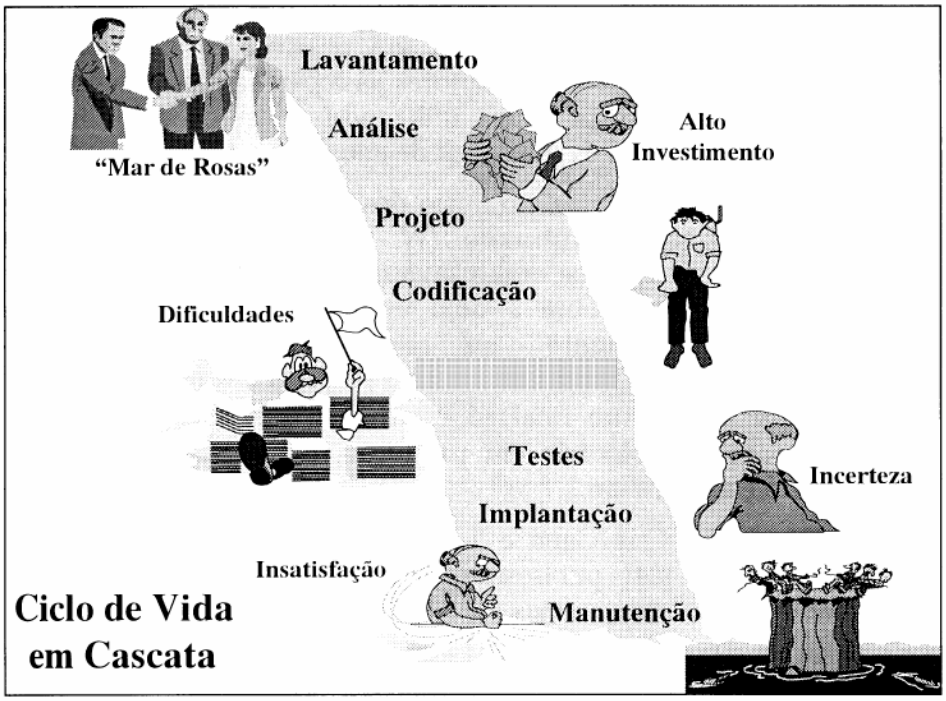

Figura 1 – Modelo em Cascata

O modelo cascata é apropriado para sistemas transacionais onde as rotinas e procedimentos a serem automatizados são altamente estruturados.

A principal desvantagem desta abordagem é o alto custo de correção das especificações quando nas fases de Teste e Implantação.

Nesse ciclo, nenhum tipo de modelo é criado, não são utilizadas técnicas de estruturação e quase não existe oportunidade para o usuário realizar alguma alteração em pontos dos requisitos congelados. As atividades são realizadas em seqüência e não existem retornos entre as atividades e toda a documentação é produzida após o término do projeto. Assim, fica evidente que os projetos realizados com este ciclo de vida se

caracterizam pela alta incidência de manutenção, pois estão sujeitos a poucas alterações durante o desenvolvimento.

3. Modelo da Análise Estruturada

O conceito de programação estruturada foi introduzido em 1962 através de artigos escritos por E.W.Dijkstra e C.Bohm / G.Jacopini, os dois últimos afirmavam que era possível escrever qualquer programa utilizando os três construtores básicos: sequência, repetição e decisão, eles afirmavam que utilizando estes construtores, a programação tornar-se-ia mais fácil de entender e manter.

A partir destas idéias, no início dos anos 70, foram surgindo os conceitos do projeto estruturado (W.Stevens. - G.Myers - L.Constantine), no qual se organizavam as funções de um programa de forma hierárquica, sobre a qual estão presentes dois conceitos fundamentais: acoplamento - comunicação entre os módulos do sistema; e coesão - relações internas dos módulos.

O produto final só estaria em um nível aceitável de qualidade para ser colocado em produção, quando possuísse baixo acoplamento e alta coesão. mais tarde, Chris Gane e T.Sarson, Tom Demarco e Edward Yourdon publicaram livros descrevendo um método estruturado de analisar sistemas. Este ciclo de vida é caracterizado pelo uso das técnicas estruturadas, incluindo as revisões estruturadas.

Muitas das atividades são realizadas em paralelo, produzindo documentação nos vários estágios do desenvolvimento. revisões periódicas são realizadas para se detectar o mais cedo possível problemas que podem influenciar no produto final.

Neste ciclo de vida, o envolvimento do usuário é bastante significativo. sua participação na maioria das revisões traz novas sugestões e correções dos aspectos não compatíveis com suas necessidades

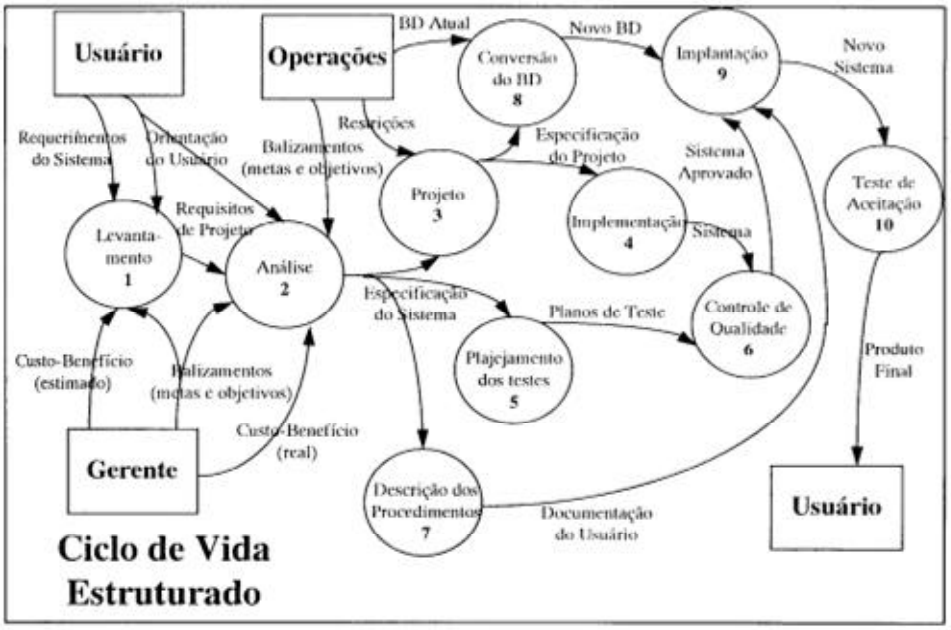

Fig 2. Ciclo de Vida Estruturado

4. Modelo da Engenharia de Software

Nos anos 70 - com a utilização da análise estruturada, o desenvolvimento de sistemas ganhou um impulso muito grande. A necessidade de novos sistemas cresceu rapidamente e as manutenções começaram a ter um papel proeminente no ciclo de vida dos sistemas e levou a um aumento natural do custo de desenvolvimento e manutenção.

Surgiu uma preocupação maior com a produtividade dos analistas e programadores, com a qualidade dos produtos e com os aspectos de segurança de programas e foi criado o ciclo de vida da engenharia de software, o qual veio para preencher certas lacunas deixadas pelo ciclo de vida da análise estruturada.

Na engenharia de software se busca uma maior disciplina em termos de desenvolvimento de sistemas e é caracterizada pela forte orientação por processos, pela determinação bem acentuada de cada fase, enfatiza a reutilizarão de código de programa, provê revisões e pontos de checagem bem determinados e define métricas bem fundamentadas para o gerente realizar o controle da produtividade, a qualidade e o custo do produto final.

Algumas métricas da engenharia de software foram apresentadas por Bany Boehm e por Amdt Von Staa. aa engenharia de software é fundamentada em sete fases: viabilidade, análise, projeto, implementá-lo teste do sistema, teste do usuário e produção. Quando algum problema ocorre em uma das fases, retoma-se a fase imediatamente anterior para se rever os passos que levaram ao desenvolvimento daquela onde ocorreu o problema.

No decorrer de 20 anos de uso de técnicas para o desenvolvimento de sistemas, no qual a idéia central era analisar com base nos processos atuais apresentados no ambiente do usuário e nos propósitos para se chegar ao sistema final, começou-se a notar que os processos dentro de uma empresa, corporação, repartição, etc. eram fortemente influenciados pelo meio ambiente externo aos locais de utilização destes processos.

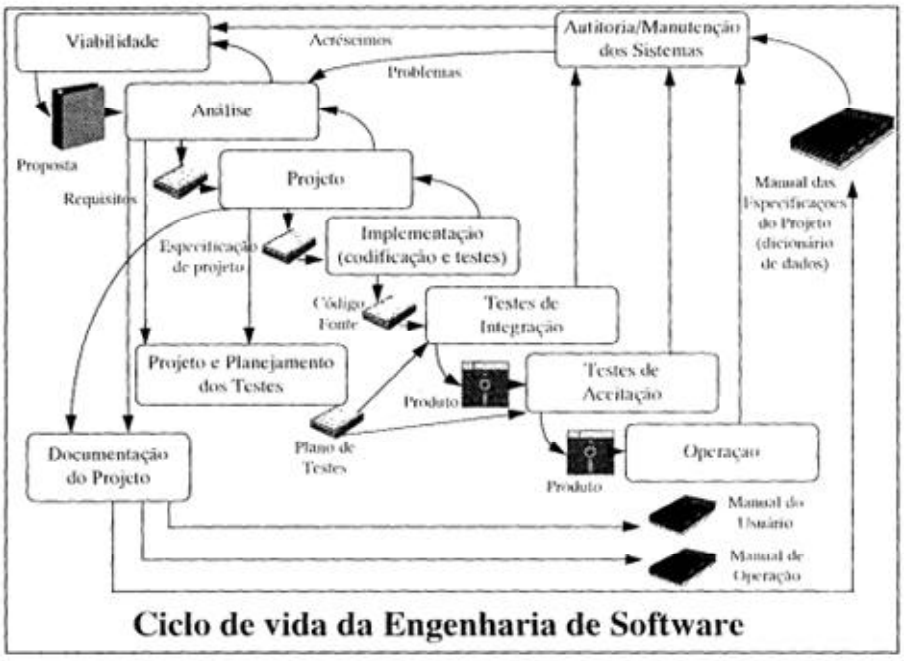

Fig 3. Ciclo de vida da Engenharia de Software

5. Modelo da Engenharia da Informação

Concluiu-se que os dados envolvidos em cada processo eram extremamente estáveis, se comparados com os processos e esta estabilidade era devido ao fato de que os dados só sofrem algum tipo de mudança no momento em que o negócio também muda - evolui.

Em 1981, Matt Flavin, James Martin e Clive Finkelstein introduziram o conceito de engenharia da informação. O principio fundamental baseava-se no fato de que o dado existe e é descrito, independentemente dos processos que podem utilizá-lo e como o centro desta metodologia é o dado, a idéia principal é levantar as estruturas de dados que vão dar origem aos bancos de dados, provendo um fácil acesso aos mesmos.

A engenharia da informação é um conjunto integrado de técnicas que organiza os dados de um determinado negócio e determina um acesso fácil, por parte do usuário final, a estes dados.

O suporte desta metodologia está baseado na técnica de modelagem de dados e seus relacionamentos, desenvolvida inicialmente por Peter Chen em 1976, chegando á modelagem da informação através de Flavin em 1981, e finalmente á modelagem semântica dos dados através de Shlaer e Mellor em 1988.

As fases são: planejamento estratégico das informações, análise da informação, modelagem de dados, formação dos procedimentos, análise do uso dos dados, análise da distribuição dos dados, projeto físico da base de dados, projeto físico da base de dados e especificação dos programas.

Os dados não possuem características táo voláteis que sua existência assuma caracterização temporal acentuada, pois nos refletem não um momento a ser modificado, mas sim uma realidade a ser automatizada. A modelagem de dados (ou de informações) está baseada no principio de que, comprovadamente, os dados são estáveis no decorrer da vida de uma empresa ou organização, não tendo volatilidade dependente de fatores pessoais, governamentais e temporais.

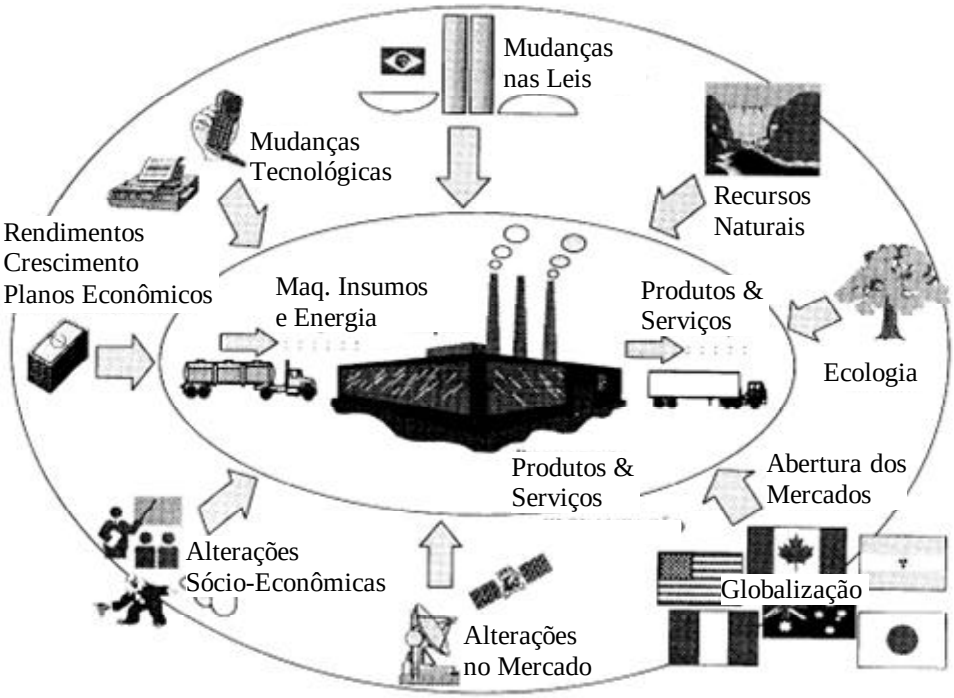

Fig. 4 Ciclo da Engenharia da Informação

Procedimentos possuem esta característica de volatilidade, pois sofrem constantes alterações, seja por fatores pessoais, quando existe troca de pessoas e métodos, por decisões governamentais e legislativas, por fatores de calamidade, e outros, externos às atividades normais da empresa. Para que uma empresa mude seus dados, toma-se necessário que a mesma mude sua atividade-fim, por exemplo: em uma indústria metalúrgica, seus dados somente irão mudar quando a mesma se transformar em uma empresa agrícola ou fechar e abrir como ferro-velho.

Conclui-se que os dados, são imutáveis durante o ciclo de vida de uma empresa, mudando somente os valores presentes nos dados, mas não o conceito do dado em si. Quando se adota uma metodologia baseada em modelo de dados, com trabalhos iniciais e extensos de modelagem, consegui-se obter um domínio do negócio da empresa, temos uma fotografia global da mesma.

Isto dá algum ganho no processo de desenvolvimento? se considerar que esses ganhos são uma relação direta entre tempo, qualidade e custo, toma-se óbvio que a análise de um sistema com técnicas que permitam conhecimento perfeito e compreensão ampla do negócio em um espaço de tempo reduzido, permitirá desenvolver e obter qualidade na aplicação, principalmente sobre a informação, e consequentemente um retorno positivo na relação de custo x beneficio para o projeto de desenvolvimento do si

6. Modelo Incremental

Modelo que divide o desenvolvimento do sistema em partes (módulos), cada uma das quais é desenvolvida seguindo as fases do modelo waterfall. Tem como características liberar porções de código mais cedo, porém requer cuidadoso planejamento.

O desenvolvimento incremental parece ser uma opção melhor para sistemas grandes, pois proporciona liberação por partes, exibindo resultados úteis já nos primeiros momentos do projeto. Na prática, funciona como se o sistema fosse dividido em subsistemas, e para cada subsistema houvesse a aplicação de um ciclo em cascata. Portanto, em vez de fazer análise de todo o sistema, é feita a análise de apenas parte do sistema; porém, esta análise precisa estar concluída para que seja iniciado o projeto.

Outra questão é que o custo de integração dos subsistemas pode ser alto, ou até mesmo inviável em alguns dos casos, onde subsistemas são desenvolvidos em paralelo por equipes diferentes. Este problema se agrava quando o grau de complexidade do sistema é elevado – o que tem sido muito comum nos dias de hoje.

O processo evolutivo compõe-se de duas fases, um período de exploração seguido de um período de desenvolvimento evolutivo. O período de exploração determina os requisitos que o sistema deverá atender, e o desenvolvimento evolutivo corresponde a uma série de etapas, cada qual destinada a produzir determinadas funcionalidades do sistema.

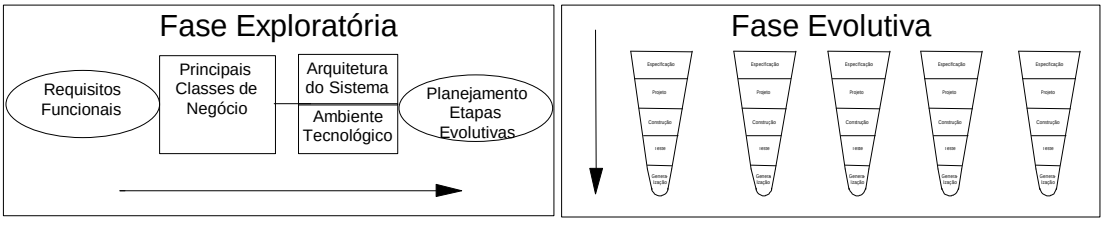

Fig 5. Ciclo de Vida Evolutivo

Durante a fase exploratória, o desenvolvedor dedica-se a compreender aquelas áreas mais desconhecidas para ele. Para tanto, esta fase inclui:

- Identificação dos requisitos funcionais
- Elaboração de um esboço do modelo do domínio da aplicação
- Elaboração de um esboço da arquitetura tecnológica
- Planejamento da fase evolutiva

Segue-se então a fase evolutiva, onde as diversas etapas planejadas darão origem ao sistema. As etapas podem ser executadas em seqüência ou em paralelo, dependendo da equipe disponível e respeitando as relações de precedência que algumas etapas possam apresentar.

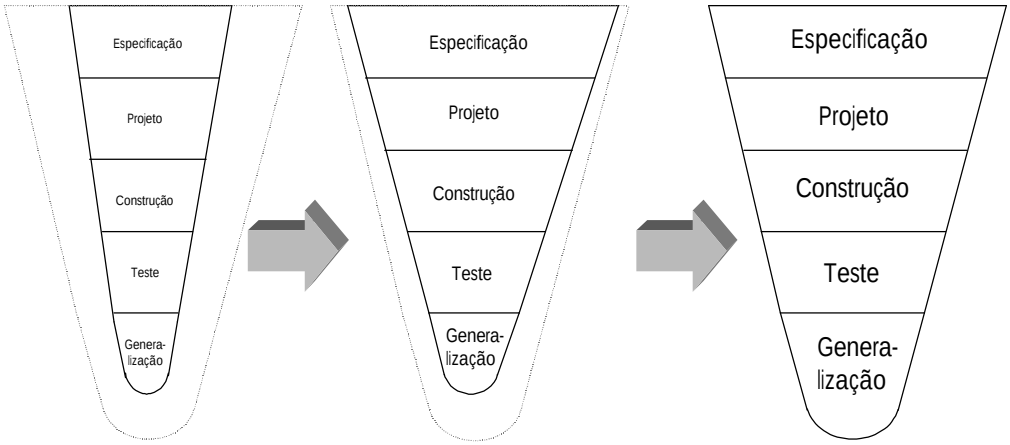

Fig.6 Fase Evolutiva

Os passos listados no ciclo evolutivo são:

- Especificação: identificação das classes e suas características – atributos, operações e associações.
- Projeto: definição da arquitetura das classes e da arquitetura de suas associações.
- Construção: codificar as classes, com todos os detalhes contemplados.
- Teste: verificação e validação do funcionamento das classes.
- Generalização: preparação para reutilização.

7. Modelo baseado em Prototipação

O modelo de prototipação se baseia na utilização de um protótipo do sistema real, para auxiliar na determinação de requisito. Um protótipo deve ser de baixo custo e de rápida obtenção, para que possa ser avaliado. Para isto, uma determinada parte do sistema é desenvolvida com o mínimo de investimento mas sem perder as características básicas, para ser analisada juntamente com o usuário.

A prototipação é um processo que habilita o desenvolvedor a criar um modelo do software que deve ser construído. O modelo pode tomar uma das 3 formas:

a) No papel ou num modelo baseado em PC em que a interação homem-máquina é desenhada para permitir ao usuário entender como tal interação irá ocorrer;
b) Um protótipo trabalhando, que implementa algum subconjunto da função requerida no software desejado ou
c) Um programa existente que executa parte ou toda a função desejada mas tem outras características que serão melhoradas no desenvolvimento do novo.

O protótipo pode servir como um "primeiro sistema". Alguns problemas surgem:

a) O cliente vê o que parece ser uma versão trabalhando do software, sem perceber que na pressa de oferecer algo trabalhando, não foram considerados aspectos de qualidade e de manutenção. Quando é informado de que o produto deve ser reconstruído, o cliente "implora" para não mude.
b) O desenvolvedor muitas vezes assume um compromisso de implementação para obter um protótipo trabalhando rapidamente. Um sistema operacional ou linguagem de programação inadequados podem ser usados simplesmente porque estão disponíveis e são conhecidos.

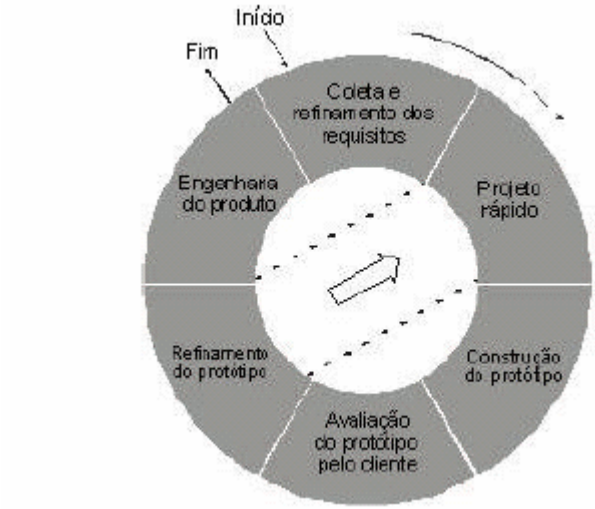

Fig. 7: Modelo Baseado em Prototipação

8. Modelo Interativo ou Espiral

O modelo espiral foi desenvolvido para abranger as melhores características tanto do modelo clássico como da prototipação, acrescentando ao mesmo tempo, um novo elemento - a análise dos riscos - que falta nos modelos anteriores. O modelo define quatro importantes atividades representadas pelos 4 quadrantes da figura 3:

Planejamento: determinação dos objetivos do sistema que será desenvolvido, restrições impostas à aplicação, tais como: desempenho, funcionalidade, capacidade de acomodar mudanças, meios alternativos de implementação;

Análise de risco: análise das alternativas e identificação/resolução dos riscos. Uma vez avaliados os riscos, pode-se construir protótipos para verificar se estes são realmente robustos para servir de base para a evolução futura do sistema;

Engenharia: desenvolvimento do produto do "próximo nível";

Avaliação do cliente: avaliação dos resultados de engenharia, pode-se efetuar a verificação e validação.

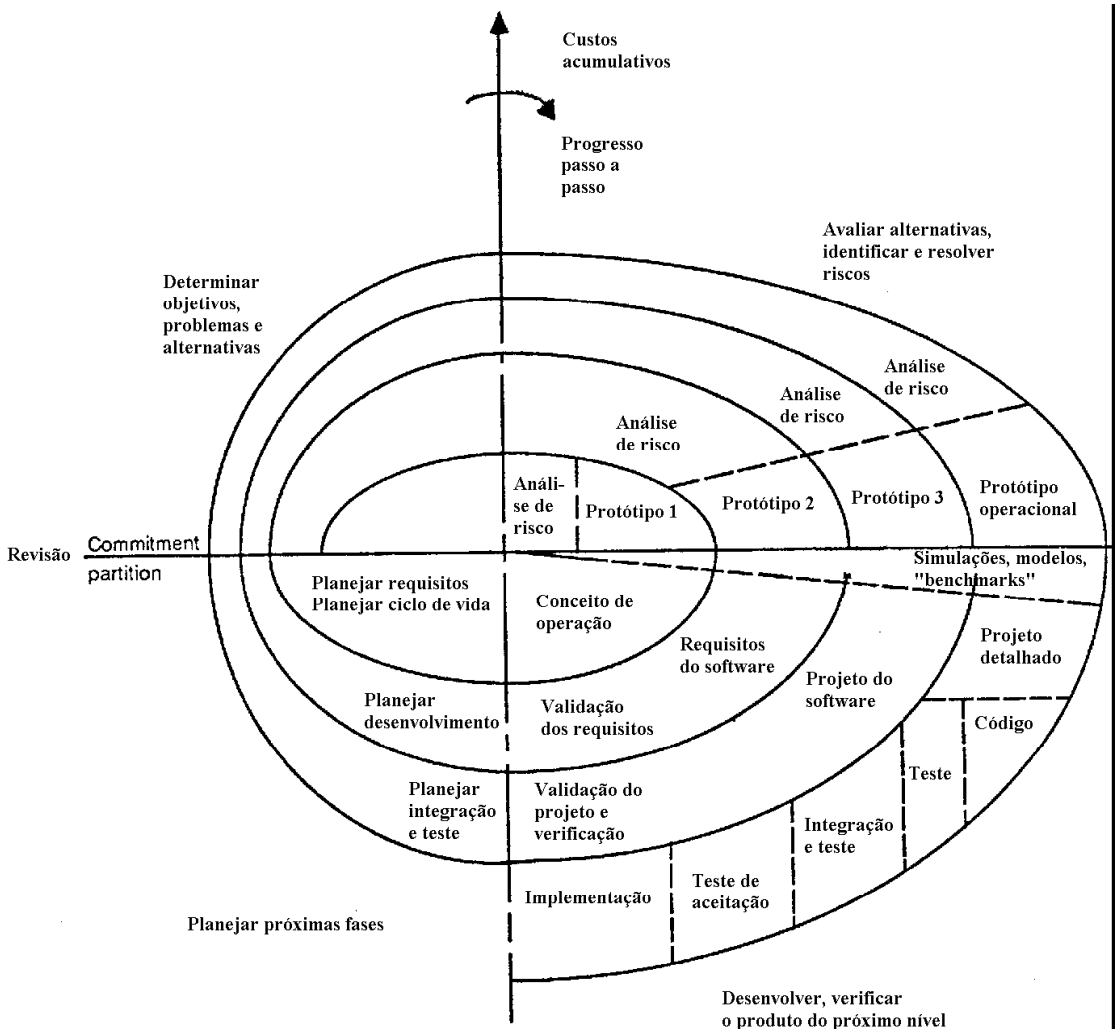

Fig. 8 – Modelo Espiral

Com cada iteração ao redor da espiral (começando no centro e trabalhando para fora), versões progressivamente mais completas do software são construídas.

Durante o primeiro circuito ao redor da espiral, os objetivos, alternativas e restrições são definidas e riscos são identificados e analisados. Se a análise de risco indica que existem dúvidas nos requisitos, a prototipação pode ser usada no quadrante de engenharia para assistir tanto o desenvolvedor como o cliente.

Simulações e outros modelos podem ser usados depois para definir o problema e refinar os requisitos.

O cliente avalia o trabalho de engenharia (quadrante de avaliação do cliente) e faz sugestões para modificações. Com base nas entradas do cliente, ocorre a próxima fase do planejamento e análise de risco. A cada loop ao redor da espiral, o ponto culminante da análise de risco resulta na decisão "continuar ou não". Se os riscos forem muito altos, o projeto pode ser encerrado.

No entanto, em muitos casos, o fluxo ao redor da espiral continua, com cada caminho movendo os desenvolvedores para fora em direção a um modelo mais completo do sistema e, ultimamente, o modelo operacional. Todo circuito ao redor da espiral requer engenharia (o quadrante inferior) que pode ser executado usando a abordagem de prototipação ou o modelo clássico de ciclo de vida.

O modelo espiral para engenharia de software pode ser considerado como o modelo mais realista para o desenvolvimento de sistemas grandes. Ele usa uma abordagem

"evolucionária" para engenharia de software habilitando o desenvolvedor e o cliente a entender e a reagir aos riscos na evolução de cada nível.

Usa o mecanismo de prototipação como um mecanismo de redução, mas mais importante, habilita o desenvolvedor a aplicar a prototipação em qualquer estágio na evolução do produto. Mantém também a abordagem por etapas do modelo clássico, mas incorpora uma estrutura interativa que reflete de forma mais realista o mundo real. O modelo espiral demanda uma consideração direta dos riscos técnicos em todos os estágios do projeto, e se propriamente aplicado, reduziria os riscos antes que eles se tornem problemáticos.

9. NORMA ISO/IEC 12207

A Engenharia de software ganhou muita importância ultimamente, devido a grande demanda do uso de softwares, que são usados por muitos produtos. Só que o desenvolvimento e a manutenção do software é uma tarefa nova, e por não ter uma produção com regras bem definidas ou normalizadas como as demais engenharias, deixa muito a desejar seja por parte dos usuários que muitas vezes não vêem seus anseios totalmente ou parcialmente atingidos, além de terem gastos, ou por parte das entidades desenvolvedoras que também não conseguem alcançar uma produtividade e lucros satisfatórios.

Por isso a comunidade mundial envolvida com desenvolvimento de software vem criando normas para regular e orientar a produção do software.

Uma dessas normas é a **ISO/IEC 12207** sob o título de *Information technology - Software Life Cycle Process* (tecnologia da informação - Processos de Ciclo de Vida de Software), que foi criada para estabelecer uma estrutura comum de processos, para ser utilizada como referência em negócios relacionados a produtos de software, e também considera que o desenvolvimento e manutenção do software devem ser conduzidos de forma semelhante a engenharia.

Esta norma agrupa os processos de ciclo de vida do software em três classes, que representam a sua natureza. Cada Processo é definido pelas suas atividades, e cada atividade é adicionalmente definida pelas suas tarefas, sendo que cada atividade subordinada a um processo é um conjunto de tarefas intimamente ligadas.

Uma tarefa é um requisito, uma declaração própria, uma recomendação ou uma ação permissível, ou seja tarefa é o que o processo deve fazer. A norma também escreve o processo de adaptação que contém as atividades básicas para adaptar a norma a uma organização ou projeto específico.

A norma possui 74 atividades e 224 tarefas, conforme infográfico , que apresenta o desdobramento dos processos, atividades e tarefas da norma:

DESDOBRAMENTOS DOS PROCESSOS DA NORMA ISO/IEC 12207			
CLASSE	**PROCESSOS**	**ATIVIDADES**	**TAREFAS**
Fundamental	**5**	**35**	**136**
Apoio	**8**	**25**	**61**
Organizacional	**4**	**14**	**27**
Total	**17**	**74**	**224**

A norma possui uma guia *ISO/IEC/JTC1/SC7/WG7 N94 – Information Technology - Guide for ISO/IEC 12207,* que orienta as organizações em seus projetos de acordo com :

Tecnologias envolvidas – podendo ser utilizada por qualquer método ou técnica de engenharia de software, bem como para qualquer linguagem de programação.

Arquitetura do ciclo de vida - Estabelece uma arquitetura de alto nível do ciclo de vida de software, que tem como princípios : Modularidade e responsabilidade.

Aplicações em organizações : A norma forma um conjunto abrangente de processos para suprir vários tipos de organizações.

Aplicação em projetos : A norma foi escrita para ser usada em complexos projetos de software. Contudo ela foi planejada para ser adaptável a um projeto de software de qualquer tipo, tamanho e complexidade, ou quando o software for uma entidade isolada, uma parte embutida ou integral de um sistema total.

Implementação de princípios de gerência de qualidade : Implementa os princípios de gerência de qualidade, e os executa em três passos básicos : Integração da qualidade no Ciclo de vida, Processo de garantia da qualidade e Processo de melhoria em nível de organização e corporação, para gerenciamento da qualidade de seus próprios processos estabelecidos.

Os 17 processos da norma são agrupados em três classes gerais : **Fundamentais , Apoio e Organizacional**

9.1. PROCESSOS FUNDAMENTAIS

Atendem o início e a execução do desenvolvimento, operação ou manutenção de produtos de software.

1- Processo de Aquisição : Define as atividades do adquirente, organização que adquire um sistema ou produto de software, inclui também emissão de pedido de proposta, seleção de fornecedor e gerência do processo de aquisição.

2- Processo de Fornecimento : Define as atividades do fornecedor, organização que provê o produto de software ao adquirente, determina os procedimentos e recursos necessários para gerenciar e garantir o projeto, o desenvolvimento e a execução dos planos de projeto até a entrega do sistema, ou software para o adquirente.

3- Processo de Desenvolvimento : Define as atividades do desenvolvedor, organização que define e desenvolve o produto, contém as atividades para análise de requisitos, projeto, codificação integração, testes, instalação e aceitação relativo ao software.

4- Processo de Operação : Define as atividades do operador, organização que provê serviço de operação de um sistema computacional no seu ambiente para seus usuários, cobre também o suporte operacional.

5- Processo de Manutenção : Define as atividades do mantenedor, organização que provê os serviços de manutenção do software, ou seja os serviços de manutenção para deixar o software atualizado. Seu objetivo é modificar um produto de software existente, preservando a sua integridade.

9.2. PROCESSOS DE APOIO

Auxiliam um outro processo e contribuem para o sucesso e qualidade do projeto de software, é empregado e executado, quando necessário por outro processo.

1. Processo de documentação : Define as atividades para registro das informações geradas por um processo ou atividade do ciclo de vida.

2. Processo de Gerência de Configuração : Define as atividades para a aplicação de procedimentos administrativos e técnicos, por toda a vida do software, identifica e define os itens de software em um sistema, bem como estabelece suas linhas básicas.

3. Processo de Garantia da Qualidade : Define as atividades para fornecer a garantia adequada para que o software esteja de acordo com seus requisitos especificados.

4. Processo de Verificação : Define as atividades para verificação se os produtos de software atendem completamente os requisitos impostos a ele.

5. Processo de Validação : Define se os requisitos do produto final atende o uso especifico proposto.

6. Processo de Revisão Conjunta : Define as atividades para avaliar a situação e produtos de uma atividade de um projeto, se apropriado, é executado tanto no nível de gerenciamento quanto no nível técnico.

7. Processo de Auditoria : Define as atividades para determinar adequação aos requisitos, planos e contrato, quando for apropriado.

8. **Processo de Resolução de Problema :** Define um processo para analisar e resolver os problemas, que são descobertos durante a execução do desenvolvimento, operação , manutenção ou outros processos.

9.3. PROCESSOS ORGANIZACIONAIS

São utilizados com o intuito de melhorar continuamente a estrutura e os processos do Ciclo de vida do Software.

1. Processo de Gerência : Define as atividades do responsável pelo gerenciamento do produto, do projeto e das respectivas tarefas, como aquisição, fornecimento, desenvolvimento, operação, manutenção ou processos de apoio.

2. Processo de Infra-estrutura : Define as atividades para estabelecer e manter a infra-estrutura necessária para qualquer processo, como hardware, software, ferramentas, para desenvolvimento e manutenção.

3. Processo de Melhoria : Define as atividades básicas para o estabelecimento, avaliação, medição, controle e melhoria do ciclo de vida de software.

4. **Processo de Treinamento :** Define as atividades para o treinamento do pessoal que vai utilizar o software.

Podemos concluir que para a área de tecnologia da informação, a **ISO/IEC 12207** é a norma mais significativa desde o surgimento da família de normas **ISO 9000,** pois é a primeira norma internacional que provê um conjunto completo de processos, atividades e tarefas que podem ser aplicados durante a aquisição de um sistema que contém software, de um produto de software independente ou de um serviço de software, e durante o fornecimento, desenvolvimento, operação e manutenção de produtos de software, provendo também processos que podem ser utilizados para definir, controlar e melhorar os processos de ciclo de vida de software.

9.4. PERGUNTAS RELATIVAS AO TEMA :

1. Cite o principal objetivo da norma internacional ISO/IEC 12207, e suas características?

R - O principal objetivo da norma ISO/IEC 12207, é o estabelecimento de uma estrutura comum de processos, para ser utilizada como referência em negócios

relacionados a produtos e serviços de software. Além disso, a norma considera que o desenvolvimento e a manutenção de software deveriam ser conduzidos da mesma forma disciplicada que a engenharia.

A estrutura descrita na norma utiliza-se de uma terminologia bem definida, composta de processos, atividades e tarefas a serem aplicados em operações que envolvam, de alguma forma, o software, seja através da aquisição, fornacimento, desenvolvimento, operação ou manutenção. Essa estrutura também permite estabelecer ligações claras com o ambiente de engenharia de sistemas, ou seja, aquele que inclui software, hardware, pessoal e práticas de negócios.

Características : A norma possui um guia que orienta as organizações na utilização e adaptação da norma em seus projetos de acordo com os seguintes tópicos :

Tecnologias envolvidas, arquitetura do ciclo de vida, aplicação em organizações, aplicação em projetos, e implementação de princípios de gerência de qualidade.

2. Explique de forma resumida as três classes em que são agrupados os 17 processos da norma ISO/IEC 12207 , e mostre para cada classe os seus respectivos processos.

R. Processos Fundamentais : Atendem o início e a execução do desenvolvimento, operação ou manutenção de produtos de software durante o ciclo de vida de software.

1. Processo de Aquisição
2. Processo de fornecimento
3. Processo de Desenvolvimento
4. Processo de Operação
5. Processo de Manutenção.

Processos de Apoio : Auxiliam um outro processo e contribuem para o sucesso e qualidade do projeto de software.

1. Processo de documentação
2. Processo de gerência de configuração
3. Processo de garantia da Qualidade
4. Processo de Verificação
5. Processo de Validação
6. Processo de Revisão Conjunta
7. Processo de Auditoria
8. Processo de resolução de Problemas.

Processos Organizacionais : São empregados por uma organização para estabelecer e implementar uma estrutura constituída de processos de ciclo de vida e pessoal associados, melhorando continuamente a estrutura e os processos.

1. Processo de gerência
2. Processo de Infra-estrutura
3. Processo de Melhoria
4. Processo de treinamento.