O que estudaremos na aula de hoje?

Engenharia de Software: Ciclo de vida do software.

Introdução

Meu caro aluno e aluna, falaremos sobre a engenharia de software e seus diversos aspectos, mas Prof. Para que serve essa tal engenharia de software, quais seus fundamentos, quais suas vertentes, e o que eu irei fazer com mais esta disciplina? Calma, responderemos todas as suas indagações, meu querido e minha querida Padawan!

Pessoalmente, gosto muito da obra de Pressman e de sua abordagem nas definições e conceitos do que realmente é a engenharia de software, entretanto temos outras enciclopédias referência no assunto, faremos uma miscigenação das duas principais obras na área da Engenharia de Software, também, com base no que cai em suas questões de provas da área, acompanhadas de exercícios sanando assim todas as dúvidas dentro do assunto ministrado.

Conceitos básicos

Dos conceitos da Engenharia de Softwares, temos:

"Engenharia de software é o estabelecimento e o emprego de sólidos princípios de engenharia de modo a obter software de maneira econômica, que seja confiável e funcione de forma eficiente em máquinas reais".

"Engenharia de *software* é uma área da computação voltada à especificação, desenvolvimento, manutenção e criação de *software*, com a aplicação de tecnologias e práticas de gerência de projetos e outras disciplinas, visando a organização, produtividade e qualidade".

Pessoal, com o fundamento nos dois conceitos descritos acima, sobre o que vem a ser a Engenharia de Software, já conseguiremos desenhar um pouco do que é esta disciplina e seus fundamentos e como são empregados em nosso dia a dia. Atualmente, as tecnologias e práticas da Engenharia de Software, englobam as mais diversas linguagens de programação, banco de dados, ferramentas e entre elas as IDEs, plataformas, bibliotecas, padrões de projeto de software, processo de software e qualidade de software. Entretanto, a engenharia de software deve oferecer mecanismos para que possamos planejar e gerenciar o processo de desenvolvimento de um sistema computacional de qualidade e que atenda às necessidades de um requisitante de software.

As bases científicas para a Engenharia de Software se relacionam com o uso de modelos abstratos que permitem ao engenheiro especificar, projetar, implementar e manter sistemas de software, avaliando e garantindo suas qualidades. A área responsável que estuda e avalia os processos da engenharia de software, propondo a evolução dos processos, ferramentas e métodos de suporte a engenharia.

Do Histórico

O termo Engenharia de Software foi criado na década de 1960, porém só utilizado oficialmente por volta do ano 1968, na NATO Science Commite. Sua criação, emergiu na tentativa de contornar a crise do software e dar um tratamento de engenharia mais sistemático, controlado e qualidade mensurável ao desenvolvimento de sistemas e software complexos.

Um sistema de software complexo caracteriza-se por um conjunto de componentes abstratos de software, estruturas de dados e algoritmos encapsulados na forma de algoritmos, funções, módulos, objetos ou agentes interconectados, compondo a arquitetura do software, que deverão ser executados em sistemas computacionais.

No início, e ainda hoje em algumas empresas menores com a falta de profissionais experientes, desenvolver um sistema é uma jornada quase que infinita, ao oposto disso, a Engenharia veio para: criar modelos, padrões, processos, metodologias, documentações, uma linha de desenvolvimento. Pois como narrei no período inicial deste parágrafo, este processo era uma aventura e que na maioria das vezes a empresa a qual pagava pela construção de um determinado software sempre ficava no prejuízo e as razões eram as mais diversas, podemos até citar algumas destas: tempo prolongado no desenvolvimento, linguagem de programação não adequada, a não realização de levantamentos de requisitos funcionais e não funcionais, antes de entrar em produção o software já não atendia aos requisitos básicos ao qual foi construído, entre centenas de problemas que poderiam causar o seu insucesso.

A figura abaixo nos dá uma definição básica de fases da construção de um software.

Quando usamos das ferramentas certas ou adequadas, os métodos eficientes, o processo documentado e seguindo um fluxo, a qualidade se torna o resumo desta abstração.

Definição de Software

Amigos, começamos falando da Engenharia de Software, mas o que é realmente um software?

"É uma sequência de instruções a ser seguidas e/ou executadas, na manipulação, redirecionamento ou modificação de um dado (informação) ou acontecimento".

"*Software*, também é o nome dado ao comportamento exibido por essa sequência de instruções, quando executada em um computador ou máquina semelhante, além de um produto desenvolvido pela engenharia de *software*, e inclui não só o programa de computador propriamente dito, mas também manuais e especificações".

A respeito do software, para fins contábeis e financeiros, ele é considerado um bem de consumo!

"Um software normalmente é composto por diversas funções, bibliotecas e módulos que gera um programa executável ao final do processo de desenvolvimento e este, quando executado, recebe algum tipo de entrada de dados, no caso (*input*), processa as informações segundo uma série de algoritmos ou sequências de instruções lógicas e libera uma saída, esta, o (*output*) como resultado deste processamento. Um software bem desenvolvido é normalmente criado pela área da engenharia de software e inclui não apenas o programa de computador em si, mas também manuais, especificações e configurações".

Bom pessoal, agora que sabemos na prática o que é um software, vamos partir para suas aplicações.

Campos de Aplicação

Os campos de aplicações são os mais diversos, pois a Engenharia de Software surgiu para colocar ordem na casa, rs. Antigamente, se desenvolvia um projeto de qualquer maneira, as datas de entrega da ferramenta quase sempre eram desobedecidas, construía-se algo de qualquer jeito, o que, para o cliente no final, não era o resultado desejado, e isto, não ocorria somente para software de sistemas, mas para diversos modelos de negócios, segue a relação das principais áreas em que a Engenharia de Software é aplicada:

- **Software de sistema:** estes são, conjuntos de programas desenvolvidos para atender a outros programas ou necessidades fins. A exemplo temos: compiladores, editores e utilitários para gerenciamento de arquivos que processam estruturas de informações complexas, Outras aplicações de sistema, por exemplo, componentes de sistema operacional, drivers, softwares de redes de computadores, processadores de telecomunicações que processam dados amplamente indeterminados. Em ambos os casos, a área de software de sistemas é caracterizada por uma enorme interação com o hardware dos computadores, o uso intenso por múltiplos usuários, as operações concorrentes que requerem escalas de ordens altíssimas, compartilhamento de recursos e gestão de processo sofisticados; estruturas de dados complexas e múltiplas interfaces externas.

- **Software de aplicação:** são programas específicos que solucionam uma necessidade específica de negócio. Aplicados nesta área, processam dados comerciais ou técnicos de uma forma que facilite as operações comerciais ou tomadas de decisões administrativas/técnicas. Além das aplicações convencionais de processamento de dados, o software de aplicação é usado para controlar funções de negócio em tempo real, a exemplo, processamento de transações em pontos de venda, controle de processos de fabricação em tempo assíncrono.

- **Software científico / engenharia:** de uso e característico por algoritmos **number crunching**, para o processamento numérico pesado. As aplicações neste modelo vão da astronomia à vulcanologia, da análise de tensões na indústria automotiva à dinâmica orbital de ônibus espaciais, também da biologia molecular à fabricação automatizada. Entretanto, aplicações modernas dentro da área de engenharia/científica estão se afastando dos algoritmos numéricos

convencionais. Projeto com o auxílio de computador, simulação de sistemas e outras aplicações interativas começaram a ter características de sistemas em tempo real e até mesmo de software de sistemas.

- **Software embutido:** residente em produtos específicos ou sistema é, utilizado para implementar e controlar características e funções para o usuário final e para o próprio sistema. Executa funções limitadas e específicas, a exemplo: controle do painel de um forno micro-ondas ou fornece funções significativas e capacidade de controle, funções digitais de automóveis, tal como controle do nível de combustível, painéis de controle e sistemas de freios.

- **Software para linha de produtos:** projetado para prover a capacidade específica de utilização por muitos clientes diferentes. Pode focalizar um mercado limitado e particularizado: produtos para controle de estoques ou direcionar-se para os mercados de consumo de massa, processamento de texto, planilhas eletrônicas, computação gráfica, multimídia, entretenimento, gerenciamento de bancos de dados e aplicações financeiras pessoais e comerciais.

- **Aplicações para a Web:** chamadas de WebApps, essa categoria de software centralizada em redes, alcança uma vasta gama de aplicações. Em sua forma mais simples, as WebApps podem ser pouco mais que um conjunto de arquivos de hipertexto interconectados, apresentando informações por meio de texto e gráficos limitados. Entretanto, com o aparecimento da Web 2.0, elas têm evoluído para sofisticados ambientes computacionais que não apenas fornecem recursos especializados, mas funções computacionais e conteúdo para o usuário final, como a integração de bancos de dados corporativos e aplicações comerciais.

- **Software de inteligência artificial:** faz uso de algoritmos não numéricos para solucionar problemas complexos que não são passíveis de computação ou de análise direta. Aplicações nessa área incluem: robótica, sistemas especialistas, reconhecimento de padrões de imagem e voz, redes neurais artificiais, prova de teoremas e jogos. Milhões de engenheiros de software em todo o mundo trabalham arduamente em projetos de software nas mais diversas categorias. Em alguns casos, novos sistemas estão sendo construídos, porém em outros casos as aplicações já existentes e estão sendo corrigidas, adaptadas e aperfeiçoadas. Não é incomum para um jovem engenheiro de software trabalhar em um programa mais velho que ele.

- **Computação mundial aberta:** o acelerado crescimento das redes sem fio pode, em breve, conduzir a uma verdadeira computação distribuída e ampliada, compartilhada e incorporada nos ambientes domésticos e comerciais. O desafio para os engenheiros de software será o de desenvolver sistemas e software aplicativo que permitam que dispositivos móveis, computadores pessoais e sistemas corporativos se comuniquem através de extensas redes. O que hoje já acontece em grande escala!

- **Netsoursing (recursos via Internet):** a Internet rapidamente tornou-se um grande e avassalador mecanismo computacional, como um provedor de conteúdo. O desafio para os engenheiros de

software, consiste em arquitetar aplicações simples, isto é, planejamento financeiro, pessoal e sofisticados que forneçam benefícios aos mercados mundiais de usuários finais visados.

- **Software aberto:** tendência crescente que resulta na distribuição de código-fonte para aplicações de sistemas, por exemplo: sistemas operacionais, bancos de dados e ambientes de desenvolvimento, de forma que muitas pessoas possam contribuir para seu crescimento. O desafio para os engenheiros de software, consiste em construir um código fonte auto descritivo, mais importante ainda é, desenvolver técnicas que permitam que tanto os clientes quanto os desenvolvedores saibam quais alterações foram feitas e como se manifestam dentro do software.

Disciplinas da Engenharia de Software

Dentro do processo da Engenharia de Software, iremos destacar algumas disciplinas, lembrando que, não detalharemos esta parte, pois possuímos livros eletrônicos que descrevem cada uma delas em detalhes:

Requisitos de software: Na sistematização e engenharia de software, **análise de requisitos** engloba todas as tarefas que lidam com a investigação, definição e escopo de novos sistemas ou alterações. Análise de requisitos é parte importante do processo de desenvolvimento de softwares, na qual o engenheiro de requisitos e o analista de negócio, juntamente com engenheiros de sistema ou desenvolvedores de software, identificam as necessidades ou requisitos de um cliente. Uma vez que os requisitos do sistema tenham sido identificados, os projetistas de sistemas estarão preparados para projetar a solução.

A análise de requisitos é uma das primeiras atividades no desenvolvimento de software. O produto do seu trabalho é a especificação dos requisitos, esta, define o escopo do software em duas dimensões: **Requisito funcional e Requisito não-funcional**. É nesta fase que o analista faz as primeiras reuniões com os clientes e/ou usuários do software para conhecer as funcionalidades do sistema que será desenvolvido. Também, ocorre a maior parte dos erros, pois a falta de experiência dos clientes ou usuários faz com que nem sempre tenham claro em sua mente quais funcionalidades o software terá.

Outro conceito é: *Análise de Requisitos é o processo que envolve o estudo das necessidades do usuário para encontrar-se uma definição correta ou completa do sistema ou requisito de software*.

Projeto de software: parte responsável da engenharia de software que se encarrega de fazer todo o planejamento anterior ao desenvolvimento, incluindo a definição da arquitetura do software, e transformar tudo em um documento ou conjunto de documentos capazes de ser interpretados diretamente pelo programador. Para atingir este objetivo, o projetista deve mapear as estruturas e funcionalidades identificadas na análise de requerimentos dentro do contexto e das restrições da arquitetura, de forma a tornar possível a construção do software. Ao longo do tempo e nos diversos processos de software existentes, várias ferramentas foram idealizadas para facilitar e atingir este objetivo.

Construção de software: Um processo de desenvolvimento de software é um conjunto de atividades, parcialmente ordenadas, com a finalidade de obter um produto de software. Estudado dentro da área de Engenharia de Software, considerado um dos principais mecanismos para se obter software de qualidade e cumprir corretamente os contratos de desenvolvimento, sendo uma das respostas técnicas adequadas para resolver a crise do software. Este é um dos pilares da entrega de produto em tempo hábil.

Teste de software: O teste do software é a investigação do funcionamento do mesmo, a fim de fornecer informações sobre sua qualidade em relação ao contexto em que ele deve operar, se relaciona com o conceito de verificação e validação. Estas que foram levantadas através de requisitos funcionais e não funcionais. Isto inclui o processo de utilizar o produto para encontrar seus defeitos. O teste é um processo realizado pelo testador de software, que permeia outros processos da engenharia, e que envolve ações que vão do levantamento de requisitos até a execução do teste propriamente dito.

Manutenção de software: Em engenharia de software, manutenção de software é o processo de melhoria e otimização de um software já desenvolvido, lembrando que, um produto mal desenvolvido com certeza necessitará de manutenções constantes e reparos de defeitos. A manutenção do software é uma das fases do processo de desenvolvimento de software, ocorre após a entrada do software em produção. A manutenção pode ser preventiva ou mesmo corretiva. Também ocorre quando aponta para falhas no Processo de Desenvolvimento de Software - PDS que devem ser identificadas e tratadas, visando a melhoria contínua do processo. As anomalias são inseridas pelo processo em si ou pelo não entendimento dos requisitos do produto.

Gerência de configuração de software: A Gerência de configuração de software ou gestão de configuração de software é, a área da engenharia de software responsável por fornecer o apoio para o desenvolvimento deste. Suas atribuições básicas são o controle de versão, o controle de mudança e a auditoria das configurações.

Gerência de engenharia de software: A Engenharia de software é, a área da computação voltada à especificação, desenvolvimento, manutenção e criação de software, com a aplicação de tecnologias e práticas de gerência de projetos e outras disciplinas, visando organização, produtividade e qualidade. Atualmente, essas tecnologias e práticas englobam linguagens de programação, banco de dados, ferramentas, plataformas, bibliotecas, padrões de projeto de software, processo de software e qualidade de software. Além disso, a engenharia de software deve oferecer mecanismos para se planejar e gerenciar o processo de desenvolvimento de um sistema computacional de qualidade e que atenda às necessidades de um requisitante de software.

Processos de Engenharia de Software: Um processo de Engenharia de Software é composto por um conjunto de passos de processo parcialmente ordenados, relacionados com artefatos, pessoas, recursos, estruturas organizacionais e restrições, tendo como objetivo produzir e manter os produtos de software finais requeridos.

Ferramentas e Métodos de Engenharia de Software: Outro ponto crucial é o uso de ferramentas CASE do inglês Computer-Aided Software Engineering. Essa classificação abrange toda ferramenta baseada em computadores que auxiliam as atividades da engenharia de software, desde a análise de requisitos, modelagem e até fase de testes. Os ambientes de desenvolvimento integrado (IDEs) têm maior destaque e suportam, entre outras:

- ✓ Editor.
- ✓ Debug.
- ✓ Geração de código.
- ✓ Modelagem.
- ✓ Deploy.
- ✓ Testes não automatizados.
- ✓ Testes automatizados.
- ✓ Refatoração (Refactoring).
- ✓ Gestão de Riscos nos projetos de Software.
- ✓ Uso da Prototipagem na Eng. de Requisitos.

Qualidade de software: O significado de qualidade possui várias definições na literatura. O glossário do IEEE define qualidade como Grau de conformidade de um sistema, componente ou processo com os respectivos requisitos ou alternativamente, como Grau de conformidade de um sistema, componente ou processo com as necessidades e expectativas de clientes ou usuários.

Ambas as definições refletem os aspectos importantes da qualidade, diversos autores apresentam outras definições que, geralmente giram em torno dos temas de conformidade com os requisitos e atendimento das expectativas. Naturalmente, pode haver diferenças entre as aplicações dessas definições se os requisitos explícitos não refletirem corretamente as necessidades reais.

Ciclo de Vida / Modelo de Processo

Conhecido como Ciclo de Vida ou Modelo de Processo é a estrutura contendo processos, atividades e tarefas envolvidas no desenvolvimento do software, operação e manutenção de um produto de software, abrangendo a vida do sistema, desde a definição de seus requisitos até o término de seu uso final.

O modelo de ciclo de vida é, a primeira escolha a ser feita no processo de software. A partir desta definição, definir-se desde a maneira mais adequada de obter as necessidades do cliente, até quando e como o cliente receberá sua primeira versão operacional do sistema. O Processo de software é o conjunto de atividades que constituem o desenvolvimento de um sistema computacional. Estas atividades são agrupadas em fases, a exemplo: definição de requisitos, análise, projeto, desenvolvimento, teste e implantação. Em cada fase são definidas, além das suas atividades, as funções e responsabilidades de cada membro da equipe e como produto resultante, os artefatos.

O que diferencia um processo de software para outro é, a ordem em que as fases vão ocorrer e isto veremos mais a frente, o tempo e a ênfase dado a cada fase, as atividades presentes e os produtos entregues. Com o crescimento do mercado de software, houve uma tendência a repetirem-se os passos e as práticas que deram certo. A etapa seguinte foi a formalização em modelos de ciclo de vida. Em outras palavras, os modelos de ciclo de vida são o esqueleto, ou as estruturas pré-definidas nas quais encaixamos as fases do processo. De acordo com a norma NBR ISO/IEC 12207:1998, o ciclo de vida é sem dúvidas a estrutura contendo os processos, atividades e tarefas envolvidas no desenvolvimento, as operações e manutenções de um produto de software, abrange a vida do sistema, desde a definição de seus requisitos até o término de seu uso.

Resumindo: O modelo de ciclo de vida é a primeira escolha a ser feita no processo de software.

Não existe um modelo ideal. O perfil e complexidade do negócio do cliente, o tempo disponível, o custo, a equipe, o ambiente operacional são fatores que influenciam diretamente na escolha do ciclo de vida de software a ser adotado. Da mesma forma, também é difícil uma empresa adotar um único ciclo de vida. Na maior parte dos casos, vê-se a presença de mais de um ciclo de vida no processo. Os ciclos de vida se comportam de maneira sequencial, fases seguem determinada ordem e/ou incremental divisão de escopo e/ou iterativa retroalimentação de fases e/ou evolutiva software é aprimorado.

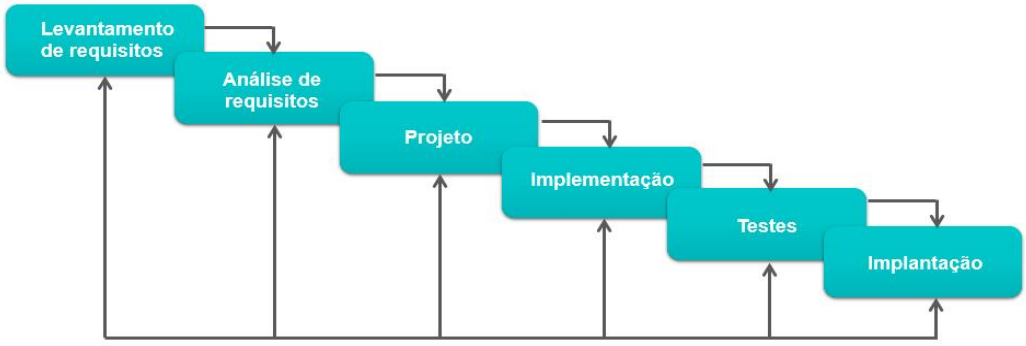
Apresentaremos alguns dos modelos mais utilizados neste processo:

1. **Modelo Cascata.**
2. **Modelo Incremental.**
3. **Modelo Evolutivo.**
4. **Modelo Prototipagem.**
5. **Modelo Espiral.**

6. **Modelo RAD.**
7. **Modelo V.**

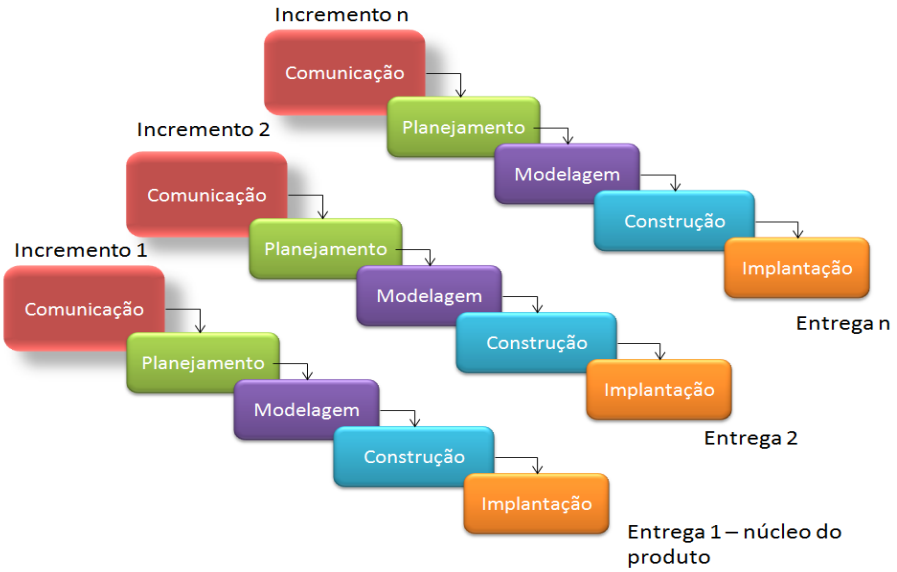
Modelo Cascata - O modelo cascata, algumas vezes chamado ciclo de vida clássico, sugere uma abordagem sequencial e sistemática para o desenvolvimento de software, começando com o levantamento das necessidades por parte do cliente, avançando pelas fases de planejamento, modelagem, construção, emprego e culminando no suporte contínuo do software concluído.
O modelo cascata é o paradigma mais antigo da engenharia de software. Entretanto, ao longo das últimas três décadas, as críticas a este modelo de processo fez com que até mesmo seus mais ardentes defensores questionassem sua eficácia. Entre os problemas encontrados quando se aplica o modelo cascata.
Formalizado por Royce em 1970, é o modelo mais antigo. Suas atividades fundamentais são:

- Análise e definição de requisitos;
- Projeto;
- Implementação;
- Teste;
- Integração.



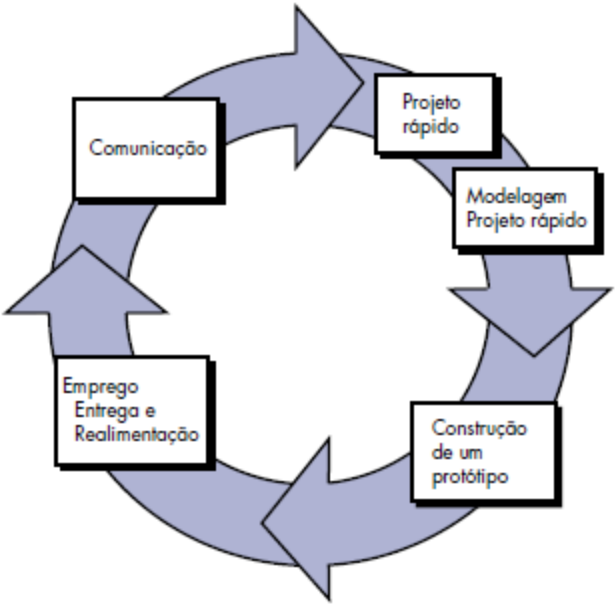
Modelo Incremental – Neste modelo, de Mills em 1980, os requisitos do cliente são obtidos e de acordo com as funcionalidades, são agrupados em módulos. Após este agrupamento, a equipe junto ao cliente, define a prioridade em que cada módulo que será desenvolvido, a escolha deve ser baseada na importância daquela funcionalidade ao negócio do cliente.

Cada módulo passará por todas as fases cascata do projeto e, será entregue ao cliente um software operacional. Assim, o cliente receberá parte do produto final em menos tempo. O modelo incremental combina elementos dos fluxos de processos lineares e paralelos, o modelo incremental aplica sequências lineares de forma escalonada à medida que o tempo vai avançando.



Modelo Evolutivo ou Evolucionário - Neste modelo, os requisitos são adquiridos em paralelo à evolução do sistema. O modelo evolutivo inicia-se do princípio de que o cliente desconhece grande parte dos requisitos funcionais, Desta forma, a análise é feita em cima dos requisitos conseguidos até então, a primeira versão é entregue ao cliente. Logo em seguida. O cliente usa o software no seu ambiente operacional e, como feedback ao pessoal da área de requisitos, esclarece o que não ficou bem entendido, passando assim mais informações sobre o que precisa e sobre o que deseja.

A partir deste feedback é, realizada nova análise, projeto e desenvolvimento são realizados e uma segunda versão do software é entregue ao cliente que novamente retorna com mais feedbacks. Assim, o software vai evoluindo e o processo torna-se agradável a todos e bem mais completo, até atender todas as necessidades do cliente dentro do escopo estabelecido. Tem-se a versão final.



Modelo Prototipagem - Prototipagem é a construção de um exemplar do que foi entendido dos requisitos capturados do cliente. Ou seja, praticamente iremos desenvolver algo para que o cliente possa usar como se

fosse inicialmente o sistema. Pode ser considerado um ciclo de vida ou pode ser usado como ferramenta em outros ciclos de vida.

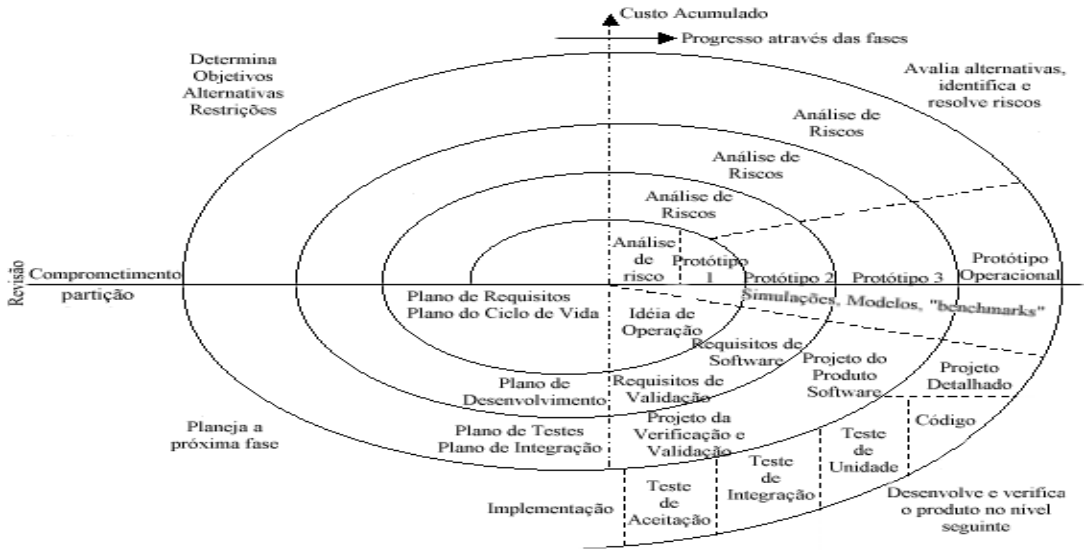
Um protótipo em engenharia de software pode ser o desenho de uma tela, um software contendo algumas funcionalidades do sistema. São considerados operacionais, podem ser utilizados pelo cliente no ambiente real, ou seja, em produção, ou não operacionais, não estão aptos para ser utilizados em produção. Os protótipos podem ser descartados ou reaproveitados para evoluírem até a versão final que no caso é o que acontece com uma recorrência muito grande.

No ciclo de vida de prototipagem o conhecimento aprofundado dos requisitos no primeiro momento não é tão exigido. Isso é bastante útil quando os requisitos não são totalmente conhecidos, são muitos complexos ou confusos. Desta forma, se o cliente não sabe expressar o que deseja, a prototipagem vem para ajudar. A maneira de evitar que se perca tempo e recursos com uma má interpretação.

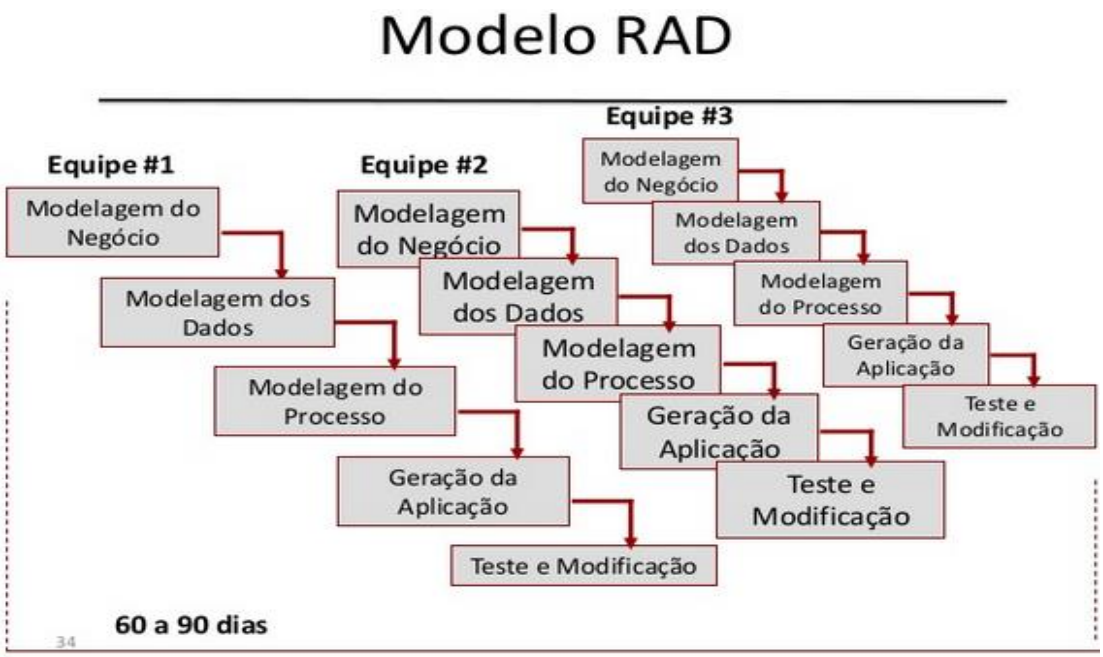
Desta forma, o cliente experimentará na prática como o sistema ou parte dele funcionará. A partir desse primeiro contato o cliente esclarece o que não foi bem interpretado, aprofunda-se alguns conceitos e até descobre um pouco mais sobre o que realmente precisa. Com este feedback, novos requisitos são colhidos e o projeto ganha maior dimensão em conhecimento.

Modelo Espiral - Originalmente proposto por Barry Boehm, o modelo espiral é o processo de software evolucionário que acopla a natureza iterativa da prototipação com os aspectos sistemáticos e controlados do modelo cascata. Fornece força para o rápido desenvolvimento de versões cada vez mais completas do software. O modelo espiral de desenvolvimento é um gerador de modelos de processos dirigidos a riscos e é utilizado para guiar a engenharia de sistemas intensivos de software, que ocorre de forma concorrente e tem múltiplos envolvidos. Possui duas características básicas que o distinguem. A primeira consiste em uma abordagem cíclica voltada para ampliar incrementalmente o grau de definição e a implementação de um sistema, enquanto diminui o grau de risco do mesmo. A segunda característica consiste em uma série de pontos âncoras de controle para assegurar o comprometimento de interessados quanto à busca de soluções de sistema que sejam mutuamente satisfatórias e praticáveis.

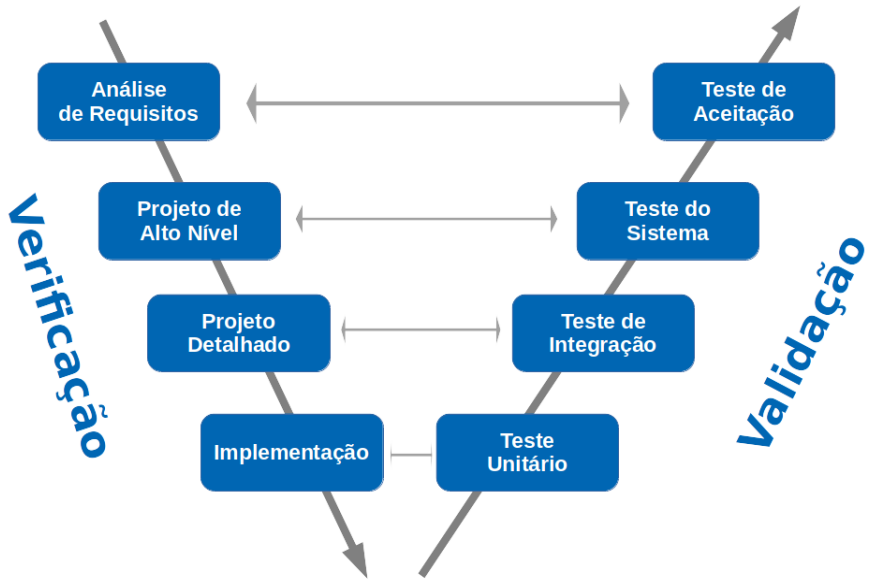
Fazendo o uso do modelo espiral o software será desenvolvido em uma série de versões evolucionárias ou evolutivas. Nas primeiras iterações, a versão pode consistir em um modelo ou em um protótipo.
Já nas iterações posteriores, são produzidas versões cada vez mais completas do sistema que passa pelo processo de engenharia.



Modelo RAD – (Rapid Application Development) modelo, formalizado por James Martin em 1991, como uma evolução da prototipagem rápida, destaca-se pelo desenvolvimento rápido da aplicação. O ciclo de vida é extremamente comprimido o que tem suas vantagens e desvantagens. É ideal para clientes que buscam lançar soluções pioneiras no mercado. O ciclo de vida é incremental e iterativo, também preferível que os requisitos tenham escopo restrito. A diferença principal do ciclo anterior é o forte paralelismo das atividades, requerendo assim, módulos bastante independentes. Aqui, os incrementos são desenvolvidos ao mesmo tempo, por equipes diferentes. Além do paralelismo, o baixo tempo de construção é devido a compressão agregada lá na fase de requisitos e na fase de implantação. Isso significa que na obtenção dos requisitos, costumam-se optar por metodologias mais dinâmicas e rápidas, como workshops ao invés de entrevistas.



Modelo V – é um modelo conceitual de Engenharia de Sistemas/Desenvolvimento de Produto visto como a melhoria ao problema de reatividade do modelo em cascata. Ele permite que durante a integração de um sistema em seus diversos níveis, os testes sejam feitos contra os próprios requisitos do componente/interface que está sendo testado, em contraste com modelos anteriores onde o componente era testado contra a especificação do componente. Nota-se a diferença entre requisito e especificação. O Modelo V, tornou-se um padrão da indústria de software depois de 1980 e após o surgimento da Engenharia de Sistemas, sendo assim o conceito padrão em todos os domínios da indústria. O mundo do software tinha feito poucos avanços em termos de maturidade, em achar na bibliografia corrente as referências que poderiam se aplicar ao sistema.



1. Processo de Desenvolvimento de Software

Um processo de desenvolvimento de software é um conjunto de atividades parcialmente ordenadas, com a finalidade de obter um produto de software. Também é o estudo da área da Engenharia de Software, conceituado como um dos principais mecanismos para se obter software de qualidade e cumprir corretamente os contratos de desenvolvimento, sendo uma das respostas técnicas adequadas para resolver a crise do software.

Outro conceito que podemos utilizar para compreender o processo de desenvolvimento de software foi o apresentado por 'Waslwick", é formado por um conjunto de processos parcialmente ordenados, relacionados a artefatos, pessoas, estruturas organizacionais e restrições, tendo como objetivo produzir e manter os produtos de software finais requeridos. Este é um conceito que considera o contexto ao qual o processo de desenvolvimento de software é aplicado, destacando as suas interdependências com outros fatores como restrições, pessoas, recursos, padrões.