Engenharia de Software I- Modelos de Processo de Desenvolvimento de Software

PROFa LUCILENA DE LIMA

A Engenharia de Software é o elemento primordial no estudo da Ciência da Computação para capacitar as pessoas na utilização de teorias, técnicas e ferramentas para a produção e desenvolvimento de sistemas de softwares.

Fonte: Pressman 2011

Engenharia de Software

- Engenharia:
  - criação e construção de soluções para problemas práticos a serviço do homem (humano) com uso sistemático de conhecimentos científicos.

- Engenharia de Software:
  - é uma forma de engenharia que aplica conhecimentos da Ciência da Computação para criar soluções com melhor custo benefício para problemas de software.
  - emprego de sólidos conhecimentos científicos para a criação de um software, que tem como objetivo central o desenvolvimento de um produto de software economicamente viável e que funcione de forma eficiente em máquinas reais e, que atenda às necessidades reais do usuário.

Benefícios da Engenharia de Software

- Acompanhar os avanços tecnológicos;
- Produzir sistemas de software eficientes;
- Facilitar a vida do usuário/cliente;
- Aplicar sólidos princípios de engenharia (computacional) para a produção de software que apresente excelente custo-benefício ao usuário/cliente.

Processo (ciclo de vida) de Desenvolvimento de Software

- conjunto estruturado de atividades exigidas para desenvolver um sistema de software, sendo apontado como um elemento essencial para o sucesso no desenvolvimento de um software.
- conjunto de atividades e resultados associados que produzem um produto de software. As atividades desenvolvidas neste processo são ligadas por padrões de relacionamentos entre elas e, devem, ser seguidas de forma correta para que se obtenha o resultado desejado, ou seja, um software de qualidade.

Processo (ciclo de vida) de Desenvolvimento de Software

- É um conjunto estruturado de atividades exigidas para desenvolver um sistema de software, sendo apontado como um elemento essencial para o sucesso no desenvolvimento de um software.

- Existem vários modelos de processo de software (ou ciclo de vida de desenvolvimento de software) e cada um representa uma alternativa para colocar ordem em uma atividade que pode se tornar caótica, se feita sem um rigoroso processo de desenvolvimento.

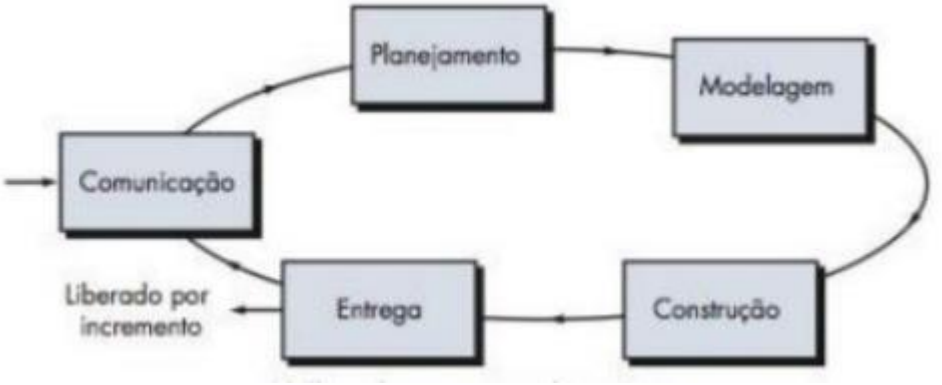
- Basicamente definem a ordem global das atividades envolvidas em um contexto de projeto de software e propõe uma estratégia de desenvolvimento que pode ser aplicada a um determinado projeto de software.

Processo (ciclo de vida) de Desenvolvimento de Software

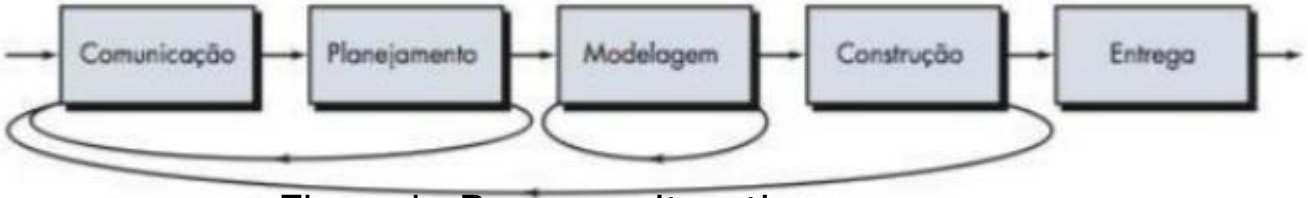
Segundo Pressman (2011), uma metodologia genérica de engenharia de processo de software estabelece cinco atividades: comunicação, planejamento, modelagem, construção e entrega. Além disso, um conjunto de atividades de apoio são aplicadas ao longo do processo, como o acompanhamento e o controle do projeto, a administração de riscos, a garantia da qualidade, o gerenciamento das configurações, as revisões técnicas, entre outras.

Fluxo de Processo – Definido por Pressman (2011)

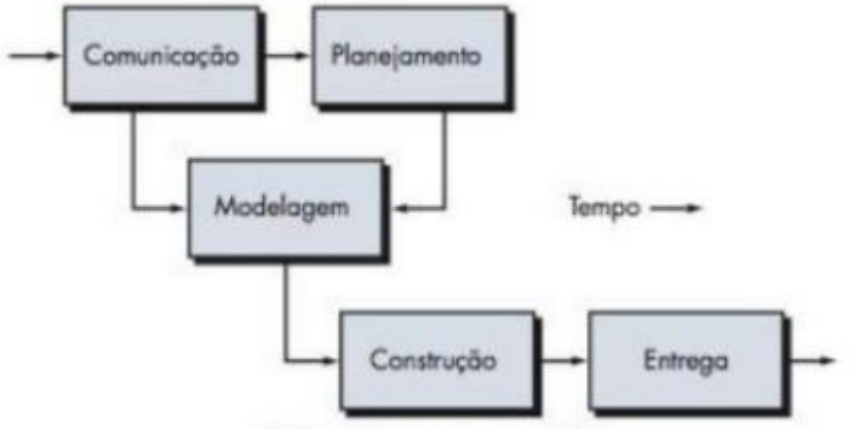
Fluxo de Processo Linear



Fluxo de Processo evolucionário



Fluxo de Processo iterativo



Fluxo de Processo paralelo

Processo (ciclo de vida) de Desenvolvimento de Software

- Qual é a finalidade de se ter um ciclo de vida de projeto?
  - 1. Definir as atividades a serem executadas em um projeto.
  - 2. Introduzir a coerência entre muitos projetos na mesma organização
  - 3. Fornecer pontos de checagem para controle de gerência e pontos de checagem para avaliar o estado do projeto (decidir continuar ou não).

Modelos de Processos para o Desenvolvimento de Software

Existe, no mercado, diversos modelos de processo de desenvolvimento de software, tais como:

- Modelo Sequencial Linear - também chamado Modelo Cascata.
- Modelo de Prototipação.
- Modelo RAD (*Rapid Application Development*).
- Modelos Evolutivos de Processo de Software.
- Modelo Incremental.
- Modelo Espiral.
- Modelo de Montagem de Componentes.
- Modelo de Desenvolvimento Concorrente.
- Modelo de Métodos Formais.
- Técnicas de 4ª Geração.

Modelos de Processos para o Desenvolvimento de Software

McCONNEL (1996, p. 154), apresenta os seguintes questionamentos para auxiliar na escolha de modelo:

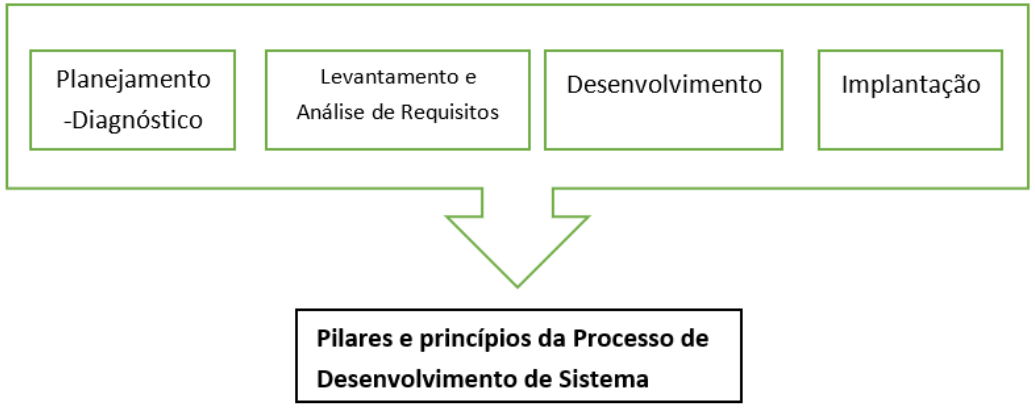
- Qual o nível de compreensão do usuário e desenvolvedores em relação aos requisitos no início do projeto? É provável que a compreensão mude significativamente durante o desenvolvimento do projeto?
- Qual o nível de compreensão dos desenvolvedores em relação a arquitetura do sistema? É provável que mudanças na arquitetura do sistema ocorram no meio do caminho?
- Qual nível de confiança é necessário?
- O quanto é necessário planejar e projetar durante o projeto prevendo mudanças em versões futuras?
- Qual o nível de riscos implícitos no projeto?
- Pode ser limitado em um cronograma?
- É necessário habilidade para realizar correções no meio do projeto?
- É necessário mostrar ao cliente o progresso durante o projeto?
- É necessário demonstrar ao usuário aspetos gerenciais durante o projeto?

Modelos de Processos para o Desenvolvimento de Software

Não existe uma regra ou um ciclo de vida padrão para desenvolvimento de sistema. Pode-se usar um ciclo pré-definido por um determinado autor, pode-se usar o ciclo de um autor e moldá-lo conforme seu trabalho ou ainda criar seu próprio ciclo.

Pilares e/ou princípios para Modelos de processo de desenvolvimento de software

1. Planejamento-Diagnóstico (planejamento – relatórios de reuniões – definição de escopo do sistema e do projeto de desenvolvimento)
2. Levantamento e Análise de requisitos (descoberto de requisitos funcionais e não funcionais – regras de negócios – documentação de requisitos – validação e verificação)
3. Desenvolvimento (codificação – testes - manutenção)
4. Implantação (treinar usuários – manter o software)

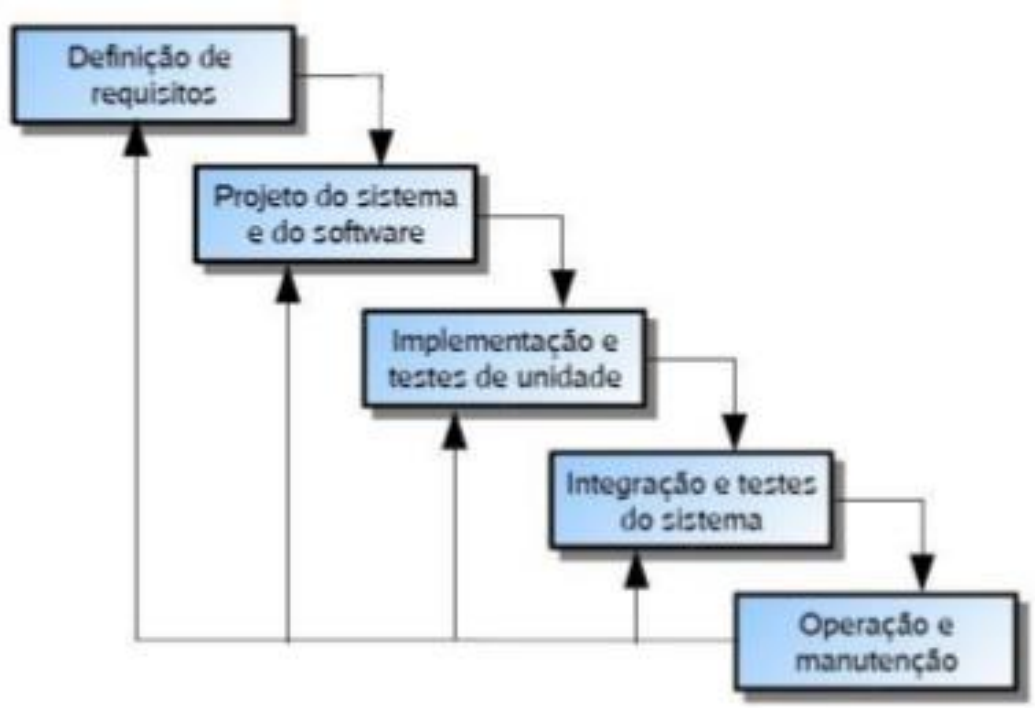


Modelo em cascata

O modelo cascata ou clássico também pode ser conhecido como “top-down”, tendo sido criado na década de 1970 por Royce, sendo o modelo mais aceito até a metade da década de 1980.

Oferece uma abordagem sequencial e sistemática, começando com o levantamento das necessidades do cliente, seguindo pelas fases de planejamento e modelagem, implementação, testes e implantação (manutenção e suporte).

Cada atividade é uma fase em separado. A passagem entre fases é formal.



Modelo em cascata

Vantagens

- O modelo cascata é utilizado principalmente quando os requisitos de um determinado problema são bem compreendidos. Se as especificações estiverem corretas (o que é muito difícil em projetos de médio e grande porte), um software pode ser desenvolvido de forma muito rápida.
- É útil quando precisa-se fazer adaptações ou aperfeiçoamentos em um sistema já existente.
- O modelo apresenta uma grande vantagem quando o escopo do trabalho é claramente definido.
- Pode-se utilizar quando um software necessita de uma nova funcionalidade e os requisitos estão bem definidos e são estáveis.
- O modelo cascata é o paradigma mais antigo da engenharia de software. Porém, mesmo sendo bastante antigo e ainda utilizado na indústria ainda recebe várias críticas.

Modelo em cascata

- **Desvantagens**
- Royce, criador do método, afirmou que, se a especificação não for bem definida, ele pode ser tornar *um risco e um convite para falhas* (TORRES, 2014).
- Como o modelo não possui flexibilidade, seguindo um sequencia engessada, é considerado um modelo frágil, principalmente em tempos atuais onde os projetos mudam muito no decorrer do desenvolvimento e Projetos grandes raramente seguem a sequencia proposta pelo modelo;
- Também é difícil estabelecer explicitamente todos os requisitos logo o início do projeto, pois no começo, sempre existe uma incerteza natural;
- O cliente deve ter muita paciência, pois uma versão executável do software só fica disponível numa etapa avançada do pode atender às necessidades do cliente, mas o código não reflete as regras de negócio, uma vez que os programadores “só programam”, não sendo possível identificar os termos de negócio olhando no código;
- A natureza linear do modelo leva a “estados de bloqueio”, nos quais alguns membros da equipe de projeto precisam esperar que outros membros completem as tarefas dependentes (PRESSMAN, 2006);

Fases - Modelo em Cascata

1- **Análise e Definição dos Requisitos**

Nesta fase, são estabelecidos os requisitos do produto, o que normalmente se baseia nas necessidades do cliente, sendo posteriormente, estabelecidos de uma forma adequada para que também sejam úteis para a próxima etapa. Esta fase compreende a documentação e o estudo da viabilidade.

2- Projeto do Sistema

O projeto de elaboração do sistema é composto por vários processos com o objetivo de estabelecer a estrutura de dados, a arquitetura do software, caracterização das interfaces e detalhes procedimentais. Nesta fase os requisitos são apresentados de uma maneira que possibilita a codificação do produto (sendo uma prévia fase de codificação).

Fases - Modelo em Cascata

3- Implementação

Fase na qual os programas (software) são criados, sendo recomendado adicionar um teste unitário de cada módulo que é desenvolvido nesta fase. Nesta situação, as unidades de código criadas são submetidas a testes individuais antes de progredir para a etapa de integração e teste global.

4- Teste do Sistema

A fase de teste do sistema inicia-se logo após o fim da etapa de codificação. Neste processo de teste é foca-se em dois pontos principais, que são as lógicas internas do software e as suas funcionalidades externas. Esta etapa é importante porque evidencia se os erros de comportamento do software foram solucionados e assegura que as entradas definidas resultem em saída eficientes e que estejam de acordo com os requisitos determinados.

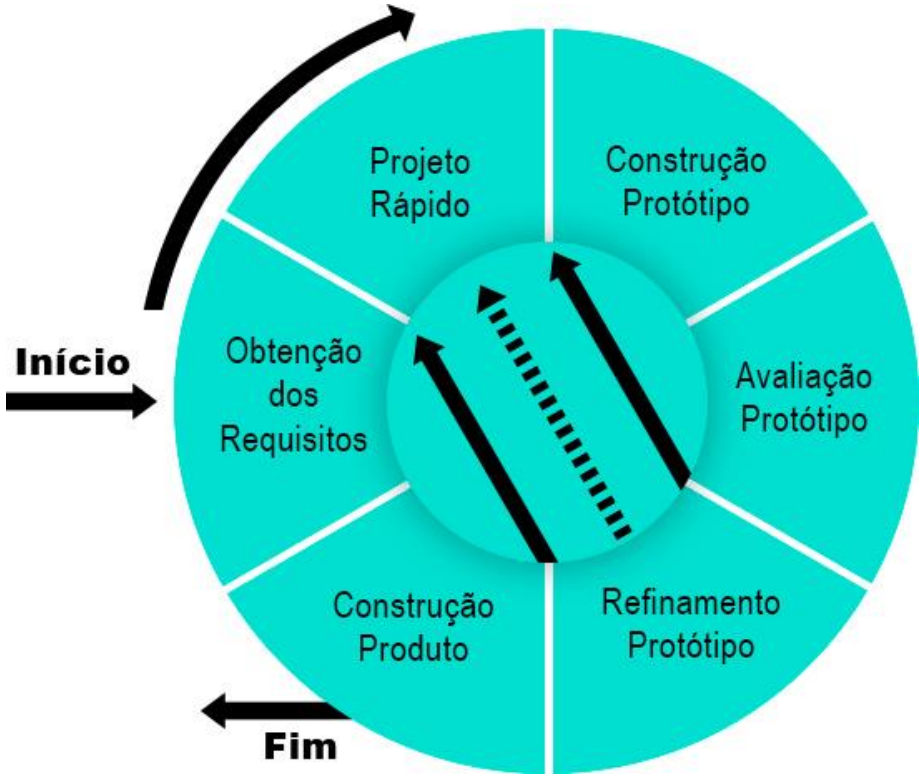
Fases - Modelo em Cascata

5- Manutenção

A fase da manutenção se baseia na correção de erros que não detectados durante os testes e em melhorias funcionais. Esta fase faz parte do ciclo de vida do produto de software e não pertence apenas ao seu desenvolvimento. As melhorias e alterações para correções do software podem ser classificadas como parte do processo de desenvolvimento.

Modelo de Prototipação

É uma abordagem baseada numa visão evolutiva do desenvolvimento de software. Neste modelo a abordagem envolve a produção de versões iniciais - protótipos (análogo a maquetes para a arquitetura) - de um sistema futuro com o qual pode-se realizar verificações e experimentações para se avaliar algumas de suas qualidades antes que o sistema venha realmente a ser construído.



Auxilia:

- Validação de requisitos para um melhor entendimento do projeto - reduzir as incertezas do projeto
- Prever problemas antes do tempo a fim de minimizar riscos - Há, assim, a chance do resultado final estar dentro das expectativas do cliente é muito maior.
- Melhor planejamento = melhores resultados.

Problemas com a prototipação:

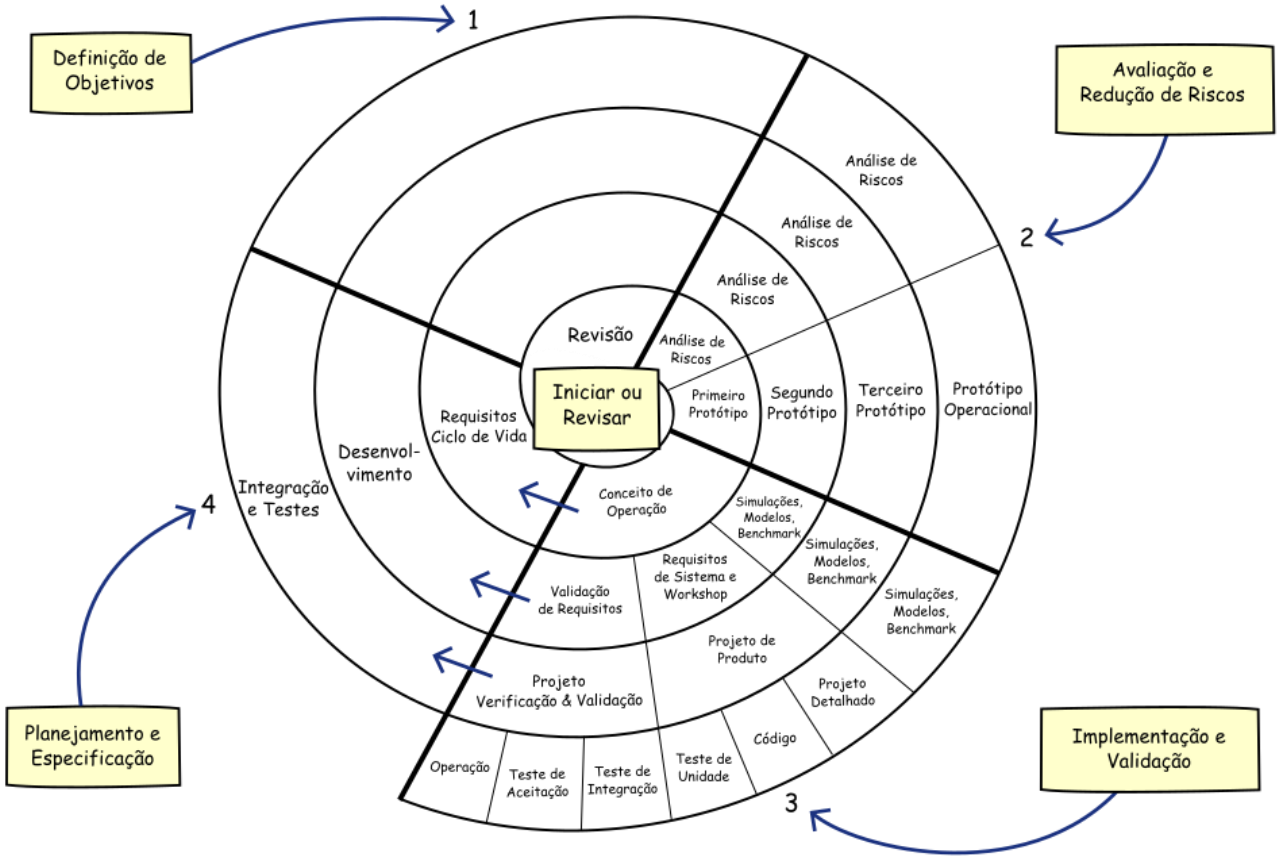
- cliente não sabe que o software que ele vê não considerou, durante o desenvolvimento, a qualidade global e a manutenibilidade a longo prazo.
- O cliente não entende que se trata de um protótipo e deseja(insiste) na utilização do protótipo como produto final.
- O software possui uma implementação comprometida (utilizando o que está disponível) com o objetivo de produzir rapidamente um protótipo.
- Depois de um tempo ele familiariza com essas escolhas, e esquece que elas não são apropriadas para o produto final.

Modelo de Prototipação

- Utilizado quando há incertezas de projeto.
- Projeto rápido ->geração de protótipos.
- É possível refazer o ciclo após feedback do cliente.
- Auxilia na identificação dos requisitos.
- Prototipação evolucionária ou descartável, sendo a evolucionária uma abordagem onde o protótipo inicial é produzido e refinado através de vários estágios até se tornar o produto sinal, já a descartável auxilia a levantar problemas com os requisitos e descartada na sequencia.

Modelo em Espiral

Criado por Barry Boehm em 1988, o *Modelo em Espiral* e possui esse nome por causa de sua representação, onde cada volta no espiral percorre todas as fases do processo de software. As voltas devem ser repetidas quantas vezes forem necessárias até que o software possa ser completamente desenvolvido e entregue.



Modelo em Espiral

É um processo evolucionário, ou seja, adequado para softwares que precisam passar por inúmeras evoluções na medida que o desenvolvimento acontece.

Diferente do *Modelo Incremental*, que entrega partes prontas uma de cada vez, o *Modelo Espiral* é mais iterativo e tenta fazer sucessivos refinamentos, além de ser um modelo no qual é possível realizar o desenvolvimento de protótipos e o gerenciamento de riscos.

A aplicação inicia-se pelo centro da espiral e prossegue no sentido horário. Os riscos são considerados à medida que cada evolução é realizada. A primeira atividade se dá com o desenvolvimento de uma especificação de produto, as próximas iterações podem ser usadas para desenvolver um protótipo e, assim sucessivamente se evolui para versões cada vez mais sofisticadas do software.

Modelo em Espiral

SOMMERVILLE (2011), apresenta a espiral dividida em quatro setores, a saber:

1. **Definição de objetivos:** onde são definidos os objetivos para essa fase do projeto, identificando as restrições e preparando um plano de gerenciamento detalhado que inclui todos os possíveis riscos do projeto;

2. **Avaliação e redução de riscos:** para cada risco identificado é feita uma "*Análise de Risco"* detalhada com o objetivo de identificar estratégias para reduzi-lo ou evitá-lo, pode ser , por exemplo, uma dificuldade na especificação clara de um requisito (risco de requisito inadequado/mal estruturado/ambíguo etc) o que pode ser solucionado com o desenvolvimento de um protótipo.

3. **Implementação e validação:** O software é desenvolvido em uma série de versões evolucionárias, partindo de um protótipo nas primeiras iterações, e nas posteriores são produzidas versões cada vez mais completa do sistema.

4. **Planejamento e Especificação:** o projeto todo é analisado para verificar o que foi realizado e planejar quais serão os próximos passos para iniciar novas voltas do espiral ou concluir o sistema.

Técnica de 4ª Geração

A Técnica de Quarta Geração (4GT) concentra-se na capacidade de se especificar o software a uma máquina em um nível que esteja próximo à linguagem natural.

Caracteriza-se pelo suporte automatizado à especificação de requisitos.

Composta por um conjunto de ferramentas de software que possibilitam que o sistema seja especificado em uma linguagem de alto nível e o código fonte seja gerado automaticamente a partir dessas especificações

Técnica de 4ª Geração

Atualmente , o ambiente de desenvolvimento de software que sustenta o paradigma 4GT inclui as seguintes ferramentas:

- Linguagens não procedimentais para consulta de banco de dados
- Geração de relatórios
- Manipulação de dados Interação e definição de telas
- Geração de códigos
- Capacidade gráfica de alto nível
- Capacidade de planilhas eletrônicas

Técnica de 4ª Geração

Vantagem:

- Redução dramática no tempo de desenvolvimento do software (aumento de produtividade).

Desvantagens:

- As 4GL atuais não são mais fáceis de usar do que as linguagens de programação.
- O código fonte produzido é ineficiente.
- A manutenibilidade de sistemas usando técnicas 4GL ainda é questionável.

Fonte:

Audy, J. L.N.; Andrade, G. Fundamentos de Sistemas de Informação – 2ª ed. Bookman – 2007 Pressman, R. S. Engenharia de Software: uma abordagem profissional – 7ª ed. Bookman – 2011

Sommerville, I. Engenharia de Software. Addison Wesley, 2003

Sommerville, I. Engenharia de Software – 9ª ed. – Person Prentice Hall – 2011

Gonçalves, R. B. Sistemas de Infomração - Versão impressa desta obra: 2017 – Sagah – 2017

IEEE, Software/Computer/Transactions on Software Engineering

Tonsig, S. L. Engenharia de software – Análise e Projeto de Sistemas 2ª ed revisada e ampliada – Ciência moderna – 2008.

OTAN. Organização do Tratado do Atlantico norte. *Science Committee Conferences on Software Engineering*

Machado, F. N. R. Análise e Gestão de requisitos de Software: onde nascem os sistemas – 3 ed. – Érica – 2016

Silva, N. P. Análise de Sistemas de Informação – conceitos, modelagem e aplicações 1ª ed. – Érica 2014