INTRODUÇÃO À SISTEMAS DE INFORMAÇÃO

AULAS 16 E 17:
ENGENHARIA DE SOFTWARE

PROFª: LEONARA BRAZ
LEONARABRAZ@GMAIL.COM

A CRISE DO SOFTWARE

- O termo "crise do software" surgiu no fim da década de 1960 – início dos anos 1970
  - Alto custo de manutenção de sistemas
  - Alto custo de novos projetos que falhavam
    - Falhar = Não cumprimento de prazos;
    - Falhar = Orçamento estourado;
    - Falhar = Não satisfação dos requisitos;
    - Falhar = Produto de baixa qualidade;
    - Falhar = Produtos não gerenciáveis e difíceis de manter e evoluir

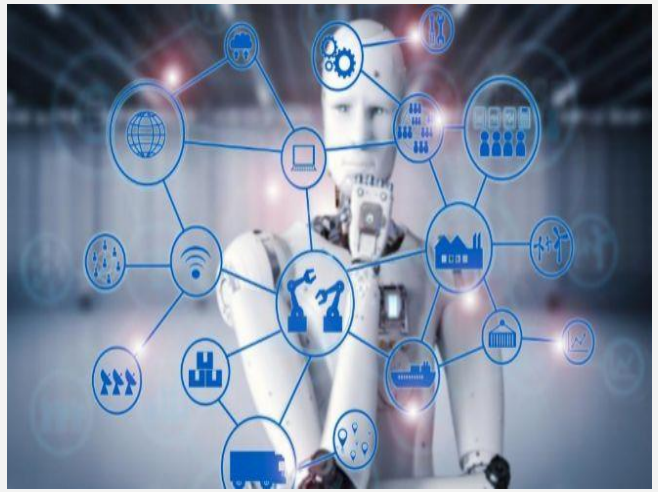
A CRISE DO SOFTWARE

"A maior causa da crise do software é que as máquinas tornaram-se várias ordens de magnitude mais potentes! Em termos diretos, enquanto não havia máquinas, programar não era um problema; quando tivemos computadores fracos, isso se tornou um problema pequeno e agora que temos computadores gigantescos, programar tornou-se um problema gigantesco."

Dijkstra (1971)

A CRISE DO SOFTWARE

- Partimos dos cartões perfurados de Jacquard (1804)
- À centenas de plataformas e dispositivos
- À inteligência artificial;



A crise do Software

OU

A crise dos desenvolvedores de Software

A CRISE DO (DESENVOLVEDOR DE) SOFTWARE

- Por onde você inicia o desenvolvimento de um novo projeto?
- Qual o passo seguinte?
- Qual o processo / método utilizado?

A CRISE DO (DESENVOLVEDOR DE) SOFTWARE

- Parece existir uma desorientação em relação sobre como planejar e conduzir o processo de desenvolvimento de software.
- Muitos desenvolvedores concordam que não utilizam um processo adequado e que deveriam investir em algum.
- E assim segue a indústria de software, década após década.
- Principais desculpas?
  - Tempo
  - Recursos financeiros

ENGENHARIA DE SOFTWARE

"A resposta a esses desafios, há alguns anos, vem sendo formulada no sentido de se estabelecer uma **execução disciplinada das várias fases** do desenvolvimento de um sistema computacional.

A **Engenharia de Software** surgiu tentando melhorar esta situação, propondo abordagens padronizadas para esse desenvolvimento."

OS PROBLEMAS PERSISTEM

- Mesmo após 40 anos de existência do termo "crise do software", ainda se vê:
  - Administradores de empresas e clientes reclamando sobre prazos não cumpridos;
  - Custos muito elevados;
  - Sistemas em uso exigindo muita manutenção;
  - Usuários reclamam de erros e falhas em sistemas;
  - Sentem-se inseguros em usá-los;
  - Reclamam das atualizações frequentes e dos preços;

E OS DESENVOLVEDORES?

- Sentem-se **pouco produtivos** em relação a seu potencial
- Sentem-se **pressionados** a cumprirem prazos e orçamentos apertados
- Sentem-se **inseguros** com as mudanças de tecnologia (qualificação x mercado)

A CRISE DO SOFTWARE CONTINUA

"Muitas vezes chamamos essa condição de 'crise do software', mas, francamente, um mal que vem sendo carregado a tanto tempo deveria ser chamado de 'normal'". (BOOCH, 1994)

Enquanto os desenvolvedores continuarem a utilizar processos artesanais...

Enquanto erros e acertos não forem capitalizados...

A CRISE DO SOFTWARE CONTINUA

- Enquanto o desenvolvimento for como o artesanato da idade média.
  - **Exemplo:** Um par de sapatos único para cada cliente.
  - O artesão atendia o cliente, obtinha a matéria prima, cortava, costurava, conduzia a prova, alterava e entregava o produto.

A CRISE DO SOFTWARE CONTINUA

- Como tem acontecido com as outras indústrias, a (indústria) de software se desenvolverá mais rapidamente, e com mais qualidade, ao passo que processos industriais forem adotados.

OS MITOS DO SOFTWARE

OS MITOS DO SOFTWARE

- Deve-se ter muito cuidado para não acreditar em mitos que assombram a cultura do desenvolvimento de software.
  - Mitos administrativos
  - Mitos do cliente
  - Mitos do profissional

OS MITOS DO SOFTWARE: ADMINISTRATIVO

1. "A existência de um manual de procedimentos e padrões é suficiente para a equipe produzir com qualidade." (Seu futuro chefe, 2021)
   - ❖ O manual é usado (usável)?
   - ❖ É completo e atualizado? (...e as mudanças nas tecnologias e plataformas?)

✓ *Os processos precisam ser melhorados e refinados constantemente.*

OS MITOS DO SOFTWARE: ADMINISTRATIVO

2. "A empresa deve produzir com qualidade, pois tem ferramentas e computadores de última geração." (Seu futuro chefe, 2021)
   - Computadores e ferramentas boas são necessários
   - Computadores e ferramentas boas não são suficientes

✓*"Comprar uma ferramenta não lhe fará instantaneamente em um arquiteto"*

OS MITOS DO SOFTWARE: ADMINISTRATIVO

3. "Se o projeto estiver atrasado, sempre é possível adicionar mais programadores para cumprir o cronograma." (Seu futuro chefe, 2021)
   - Desenvolvimento de software é algo complexo.
   - O simples ato de adicionar pessoas ao time pode gerar mais atrasos.

✓ *Imagine construir um programa de 20 mil linhas de código com apenas um minuto de prazo.*

❖Bastaria contratar 20 mil programadores.

OS MITOS DO SOFTWARE: ADMINISTRATIVO

4. "Um bom gerente pode gerenciar qualquer projeto" (Seu futuro chefe, 2021)
   - Desenvolvimento de software é algo complexo
   - Sem boa *comunicação* com a equipe, nada ele poderá fazer
   - Sem uma equipe tecnicamente capacitada para o projeto, nada ele poderá fazer
   - Sem um processo gerenciável dificilmente conseguirá cumprir os prazos e metas

OS MITOS DO SOFTWARE: CLIENTES

1. "Uma declaração geral de objetivos é suficiente para iniciar a fase de programação. Os detalhes podem ser adicionados depois." (Seu futuro cliente, 2021)
   - Esperar que a especificação esteja 100% completa e correta é utópico.
     - No entanto não se deve conformar-se.
   - Poucos detalhes significa retrabalho.

✓ *Técnicas mais sofisticadas de análise de requisitos e uma equipe bem treinada poderão ajudar a construir especificações melhores em menos tempo.*

OS MITOS DO SOFTWARE: CLIENTES

2. "Os requisitos mudam com frequência, mas sempre é possível acomodá-los, pois o software é flexível. Código é fácil de mudar!" (Seu futuro cliente, 2021)
   - Escrever código sem criar faltas (erros) é difícil, especialmente em empresas sem processos maduros.
   - O software para ser flexível de fato precisa ser projetado para isso.
     - Identificar requisitos permanentes x mutáveis (transitórios)

✓ *"Software não é um edifício, mas é difícil alterá-lo. Manutenção implica esforço e custo (tempo e recursos)".*

OS MITOS DO SOFTWARE: CLIENTES

3. "Eu sei do que preciso." (Seu futuro cliente, 2021)
   - Desenvolvedores geralmente discordam: "o cliente nunca sabe o que quer, nem o que precisa".
   - Analistas devem entender que os clientes raramente sabem o que precisam
   - Analistas devem entender que os clientes muitas vezes tem dificuldade de lembrar de suas próprias necessidades

✓ *Analistas devem tomar cuidado para não confundirem as necessidades do cliente (análise) com as soluções possíveis (projeto)*

OS MITOS DO SOFTWARE: PROFISSIONAIS

1. "Assim que o programa for colocado em operação, nosso trabalho terminou." (VOCÊ, 2021)

✓ *Alguns estudos apontam que mais da metade do esforço aplicado com um sistema de software ocorre após a sua implantação.*

OS MITOS DO SOFTWARE: PROFISSIONAIS

2. "Enquanto o programa não estiver funcionando, não será possível avaliar sua qualidade" (VOCÊ, 2021)
   - O programa é apenas um dos artefatos produzidos (sim, certamente o mais importante).

✓ *A qualidade dos requisitos, modelos, casos de uso, protótipos, fazem parte do processo de desenvolvimento e influenciam diretamente no produto final.*

OS MITOS DO SOFTWARE: PROFISSIONAIS

3. "Se eu esquecer de algo, posso consertar depois." (VOCÊ, 2021)
   - Quanto mais complexo fica o sistema, mais custosa fica a manutenção.
   - ***Nota mental: Conserte agora!***

OS MITOS DO SOFTWARE: PROFISSIONAIS

4. "A única entrega importante em um projeto de software é o software funcionando." (VOCÊ, 2021)
   - Sim, certamente a mais importante.
   - Mas se o usuário não conseguir utilizá-lo?
   - E se o usuário não cadastrar corretamente as informações?
   - E se os dados não foram importados corretamente?

✓ *É necessário realizar testes de operação; Treinar os usuários; Definir processos operacionais; Talvez a elaboração de manuais também seja importante*

ATIVIDADE

1. Qual processo de desenvolvimento você utiliza em seu ambiente de trabalho/estudo?
2. Como você considera sua produtividade?
3. Como você analisa a qualidade dos softwares/projetos produzidos pela sua equipe?
4. Você sente pressão por prazos?
5. Quais dos mitos do software você já vivenciou?

ENGENHARIA DE SOFTWARE

ENGENHARIA DE SOFTWARE
DEFINIÇÃO

"O processo de **estudar, criar** e **otimizar** os processos de trabalho para os desenvolvedores de software."

(WAZLAWICK, 2013)

ENGENHARIA DE SOFTWARE
DEFINIÇÃO

"Engenharia de software é a **aplicação** de **abordagens sistemáticas**, **disciplinadas** e **quantificáveis** ao desenvolvimento, operação e manutenção de software, além do estudo dessas abordagens"

IEEE Computer Society (2004)

DÚVIDA

Engenheiro de Software

=

Desenvolvedor

O ENGENHEIRO DE SOFTWARE

- "O engenheiro de software não desenvolve nem especifica software. Ele viabiliza e acompanha o processo de produção, fornecendo e avaliando as ferramentas e técnicas que julgar mais adequadas a cada projeto ou empresa." (Wazlawick, 2013)
- O engenheiro de software tem um *metapapel* em relação ao processo de desenvolvimento.
- O engenheiro de software não coloca a mão na massa, assim como o engenheiro civil não vai à obra assentar tijolos ou concretar uma laje.

O DESENVOLVEDOR DE SOFTWARE

- "Os desenvolvedores, de acordo com seus papéis, têm a responsabilidade de descobrir os requisitos e transformá-los em um produto executável." (Wazlawick, 2013)
- Desenvolvedor é todo aquele que é executor do processo de construção de software
  - Gerente de Projeto
  - Analista
  - Arquiteto / designer
  - Programador

O DESENVOLVEDOR DE SOFTWARE
GERENTE DE PROJETO

- Cuida de um projeto específico garantindo o cumprimento dos prazos e orçamento.
- Segue as práticas definidas no processo de engenharia.
- É responsável por verificar a aplicação do processo pelos desenvolvedores

O DESENVOLVEDOR DE SOFTWARE
ANALISTA

- É um desenvolvedor responsável por compreender o problema relacionado ao sistema.
- Realiza o levantamento de requisitos e sua modelagem.
- O analista deve descobrir o que o cliente precisa.
  - Controle de estoque; Controle de vendas; Cadastro dos clientes

O DESENVOLVEDOR DE SOFTWARE
ARQUITETO / DESIGN

- Toma como base as especificações do analista e propõe a melhor tecnologia para produzir um sistema executável para elas.
- Deve apresentar uma solução para as necessidades levantadas pelo analista.
  - O arquiteto deve pensar em escalabilidade
  - O arquiteto deve pensar em segurança
  - O arquiteto deve pensar em balanceamento de carga

O DESENVOLVEDOR DE SOFTWARE
PROGRAMADOR

- Constrói a solução física a partir das especificações do designer.
- O programador é responsável por gerar o produto final: *O programa*.
- O programador deve:
  - Conhecer profundamente a linguagem de programação e seu ambiente.
  - As bibliotecas que serão utilizadas.
  - Conhecer algo sobre testes e depuração de software

O DESENVOLVEDOR DE SOFTWARE

- Nem sempre esses papéis serão bem observados nas organizações. Mas eles estarão lá!
- Ainda que uma pessoa execute múltiplos papéis, os papéis são distintos e está acima das pessoas.

TIPOS DE SOFTWARE

TIPOS DE SOFTWARE

- Não existe um único processo para desenvolvimento de software.
- Um bom processo é aquele que é adequado ao tipo de software que se pretende desenvolver, considerando suas particularidades, características.
- Podemos agrupar sistemas com características comuns. Do ponto de vista da engenharia de software eles são classificados como:
  - Softwares de tempo real
  - Softwares Comerciais
  - Softwares Científicos
  - Softwares...

TIPOS DE SOFTWARE
SOFTWARES BÁSICOS

- Componentes do nosso sistema operacional
- Drivers
- Compiladores

TIPOS DE SOFTWARE
SOFTWARES DE TEMPO REAL

- Sistemas que monitoram, analisam e controlam eventos do mundo real.
- Exemplo:
  - Sistema de monitoramento de tráfego
  - Sistema de segurança
  - Sistema de esteiras em fábricas

TIPOS DE SOFTWARE
SOFTWARES COMERCIAIS

- Sistemas aplicados nas empresas.
- Exemplo:
  - Controle de vendas;
  - Controle de Estoques
  - Gerenciamento de clientes e relacionamentos etc.
  - Sistemas que acessam bancos de dados.
- São conhecidos como *sistemas de informação.*

TIPOS DE SOFTWARE
SOFTWARES CIENTÍFICOS

- Sistemas que utilizam processamento pesado de números.
- Exemplo:
  - Softwares de cálculo de estruturas
  - Softwares de modelagem
  - Ferramentas CAD

TIPOS DE SOFTWARE
SOFTWARES EMBARCADOS

- Sistemas de software presentes em celulares, eletrodomésticos, automóveis...
- Normalmente esses softwares lidam com limitações de espaço, tempo de processamento, memória, energia etc.

TIPOS DE SOFTWARE
SOFTWARES PESSOAIS

- Sistemas de uso pessoal no cotidiano
- Exemplo:
  - Processadores de texto
  - Processadores de Planilhas

TIPOS DE SOFTWARE
JOGOS

- O campo dos jogos exigem características e competências das mais diversas
  - Orçamentos astronômicos
- Existem jogos que exigem:
  - Processamento complexo
  - Processamento gráfico altíssimo
  - Necessidade de reação em tempo real
  - Alta demanda por conexão com a internet rápida

TIPOS DE SOFTWARE
INTELIGÊNCIA ARTIFICIAL

- Os sistemas especialistas, redes neurais e sistemas capazes de alguma forma de aprendizado.
- Podem ser sistemas independentes ou embutidos em outros.

PROJETO DE DESENVOLVIMENTO DE SOFTWARE

DEFINIÇÃO

- **Software:** Criação intelectual compreendendo os *programas, procedimentos, regras e qualquer documentação* correlata à operação de um sistema de processamento de dados.
- **Produto de Software:** Conjunto completo de *programas de computador, procedimentos e documentação* correlata, assim como dados designados para entrega a um usuário.
- **Item de Software:** Qualquer parte identificável de um produto de software

PRINCÍPIOS

- Recebemos como herança e missão de propagar boas práticas no desenvolvimento de novos projetos.
- Lições aprendidas ao longo do tempo sobre *COMO* desenvolver software.
- Não se trata de simples regras, mas de filosofia / princípios

DECOMPOSIÇÃO

- *Como tratar a complexidade inerente a sistemas de software ?*
  - A decomposição funcional é uma maneira de conceber o software como um conjunto de funções de alto nível (requisitos) que são decompostas em partes cada vez mais simples até chegar a comandos individuais de uma linguagem de programação.

ABSTRAÇÃO

- Descrever um elemento em uma linguagem de nível mais alto do que o necessário para sua construção.
  - Ou seja: simplificar! Muitas vezes deixando escapar alguns detalhes propositalmente.
- É um auxílio importante para que todos os interessados no desenvolvimento possam entender estruturas grandes e complexas.

GENERALIZAÇÃO

- Agrupar conceitos com atributos comuns.
- Permite a reutilização de definições em itens de software.
- A Orientação a Objetos é fruto da ideia de generalização:
  - Professores e Alunos têm em comum o fato de serem Pessoas, eles possuem atributos comuns.

PADRONIZAÇÃO

- A padronização auxilia na elaboração de produtos com qualidade mais previsível.
  - Padrões permitem capitalizar experiências de outros projetos.
  - Erros já experimentados e que já possuem solução conhecida.

GERENCIAMENTO DE REQUISITOS

- É possível identificar todos os requisitos de um sistema desde o início do projeto?
  - Requisitos mudam com muita frequência.
  - É necessário gerenciar sua mudança / evolução.
  - Faz parte do processo de desenvolvimento e evolução do software.

GERENCIAMENTO DE MUDANÇAS

- É imprescindível manter o controle da evolução e alterações em produtos de software.
  - Há situações nas quais é necessário voltar atrás.
  - Há situações nas quais é necessário desenvolver duas versões de um mesmo componente paralelamente.
  - Um bom sistema de gerenciamento de configuração e mudança permite efetividade nessa tarefa.

DESENVOLVIMENTO INTERATIVO

- É possível desenvolver software em um único ciclo com início, meio e fim?
  - Atualmente se trabalha com o entendimento de processo de desenvolvimento interativo:
    - Vários ciclos de desenvolvimento são realizados.
    - Cada ciclo visa atender um conjunto de objetivos (decomposição).
    - Cada ciclo contribui para a geração e amadurecimento do produto final.

GERENCIAMENTO DE RISCOS

- Sempre existem riscos em um projeto.
  - Eles precisam ser previstos.
  - É necessário prevenir-se.
  - É preciso definir plano de ação.
- Exemplos:
  - Equipe com alta rotatividade.
  - Equipe inexperiente.
  - Pressão pela redução de prazos.
  - Exceder limite orçamentário.

LEVANTAMENTO DE REQUISITOS

TÉCNICAS PARA LEVANTAMENTO DE REQUISITOS

- O início para toda a atividade de desenvolvimento de software é o levantamento de requisitos.
- Sommerville (2003) propõe um processo genérico de levantamento e análise que contém as seguintes atividades:
  - **Coleta de requisitos:** É o processo de interagir com os stakeholders do sistema para descobrir seus requisitos. A compreensão do domínio se desenvolve mais durante essa atividade;
  - **Resolução de conflitos:** Quando múltiplos stakeholders estão envolvidos, os requisitos apresentarão conflitos. Essa atividade tem por objetivo solucionar esses conflitos;
  - **Definição das prioridades:** Em qualquer conjunto de requisitos, alguns serão mais importantes do que outros. Esse estágio envolve interação com os stakeholders para a definição dos requisitos mais importantes;
  - **Verificação de requisitos:** Os requisitos são verificados para descobrir se estão completos e consistentes e se estão em concordância com o que os stakeholders desejam do sistema.

TÉCNICAS PARA LEVANTAMENTO DE REQUISITOS

- Entre as dificuldades encontradas na fase de levantamento de requisitos estão:
  - O usuário principal do sistema não sabe o que quer que o sistema faça – ou sabe e não consegue transmitir para o analista;
  - Requisitos identificados, mas que não são realistas e não identificam os requisitos similares informados por pessoas diferentes.
  - Um stakeholder errado afetará em perda de tempo e dinheiro para ambas as partes envolvidas no desenvolvimento do sistema.

TÉCNICAS PARA LEVANTAMENTO DE REQUISITOS

- As técnicas de levantamento de requisitos têm por objetivo superar as dificuldades relativas a esta fase.
  - Exemplos:
    - Entrevistas
    - Workshops
    - BrainStorming
    - Questionários
    - Grupo focal

TÉCNICAS PARA LEVANTAMENTO DE REQUISITOS
ENTREVISTA

- A entrevista é uma das técnicas tradicionais mais simples de utilizar e que produz bons resultados na fase inicial de obtenção de dados.
- Convém que o entrevistador dê espaço ao entrevistado para esclarecer as suas necessidades. É uma discussão do projeto desejado com diferentes grupos de pessoas.

- **Vantagem:**
  - O analista terá facilidade em descobrir que informação o usuário está mais interessado e usar um estilo adequado ao entrevistar
- **Desvantagem:**
  - Podem ocorrer desvios de curso, no decorrer da entrevista
  - Consumir mais tempo e recursos com sua realização;

TÉCNICAS PARA LEVANTAMENTO DE REQUISITOS
WORKSHOP

- Trata-se de uma técnica de elicitação em grupo usada em uma reunião estruturada.
- Devem fazer parte do grupo uma equipe de analistas e uma seleção dos stakeholders que melhor representam a organização e o contexto em que o sistema será usado
  - Obtendo assim um conjunto de requisitos bem definidos.

- **Vantagem:**
  - Trabalho em equipe tornando o levantamento de requisitos mais eficaz;
  - Baixo custo e resposta relativamente rápida;
  - Tempo de obtenção de informações é reduzido.
- **Desvantagem:**
  - Não abre caminho para ideias externas além da equipe de analistas

TÉCNICAS PARA LEVANTAMENTO DE REQUISITOS
BRAINSTORMING

- É utilizado normalmente em workshops.
- Após os workshops serão produzidas documentações que refletem os requisitos e decisões tomadas sobre o sistema a ser desenvolvido.
- Seu objetivo é uma apresentação do problema/necessidade a um grupo específico, requerendo assim soluções.

- **Vantagem:**
  - Várias pessoas pensam melhor do que uma
  - Rompe a inibição de ideias;
- **Desvantagem:**
  - Disponibilidade de todos pode inviabilizar o levantamento de dados.

TÉCNICAS PARA LEVANTAMENTO DE REQUISITOS
QUESTIONÁRIO

- Diferente da entrevista, essa técnica é interessante quando temos uma quantidade grande de pessoas para extrair as mesma informações.
- As questões são dirigidas por escrito aos participantes com o objetivo de ter conhecimento sobre opiniões das mesmas questões.
- São auto-aplicáveis pois o próprio informante responde.

- **Vantagem:**
  - Atinge um grande número de pessoas; Menores custos;
  - Permite que os participantes respondam no momento em que acharem conveniente;
  - Questões padronizadas garantem uniformidade.
- **Desvantagem:**
  - Os resultados são bastante críticos em relação ao objetivo, pois as perguntas podem ter significados diferentes a cada participante questionado.

TÉCNICAS PARA LEVANTAMENTO DE REQUISITOS
GRUPO FOCAL

- É um grupo de discussão informal e de tamanho reduzido (até 12 pessoas), com o propósito de obter informação qualitativa em profundidade.
- As pessoas são convidadas para participar da discussão sobre determinado assunto.

- **Vantagem:**
  - Baixo custo, resposta rápida e Flexibilidade;
  - Obtêm informações qualitativas a curto prazo;
  - Eficiente para esclarecer questões complexas no desenvolvimento de projetos;
- **Desvantagem:**
  - Exige facilitador/moderador com experiência para conduzir o grupo
  - Depende da seleção criteriosa dos participantes

CICLO DE VIDA DO SOFTWARE

CICLO DE VIDA DO SOFTWARE

O ciclo de vida é a estrutura contendo *processos, atividades e tarefas* envolvidas no *desenvolvimento, 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.*

CICLO DE VIDA DO SOFTWARE

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

A partir desta escolha definir-se-á desde a maneira mais adequada de obter as necessidades do cliente, até quando e como o cliente receberá sua primeira versão operacional do sistema.

CICLO DE VIDA DO SOFTWARE

- Processo de software é o conjunto de atividades que constituem o desenvolvimento de um sistema computacional.
  - Estas atividades são agrupadas em fases, como: *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.

CICLO DE VIDA DO SOFTWARE

- O que diferencia um processo de software do outro é a ordem em que as fases vão ocorrer, o tempo e a ênfase dados 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.

CICLO DE VIDA DO SOFTWARE

"Estrutura contendo processos, atividades e tarefas envolvidas no desenvolvimento, 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"

NBR ISO/IEC 12207:1998

CICLO DE VIDA DO SOFTWARE

- 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).

- Neste contexto, neste artigo apresentaremos alguns modelos de ciclo de vida, quais sejam:
  - Cascata
  - Modelo em V
  - Incremental
  - Evolutivo

CICLO DE VIDA
MODELO EM CASCATA

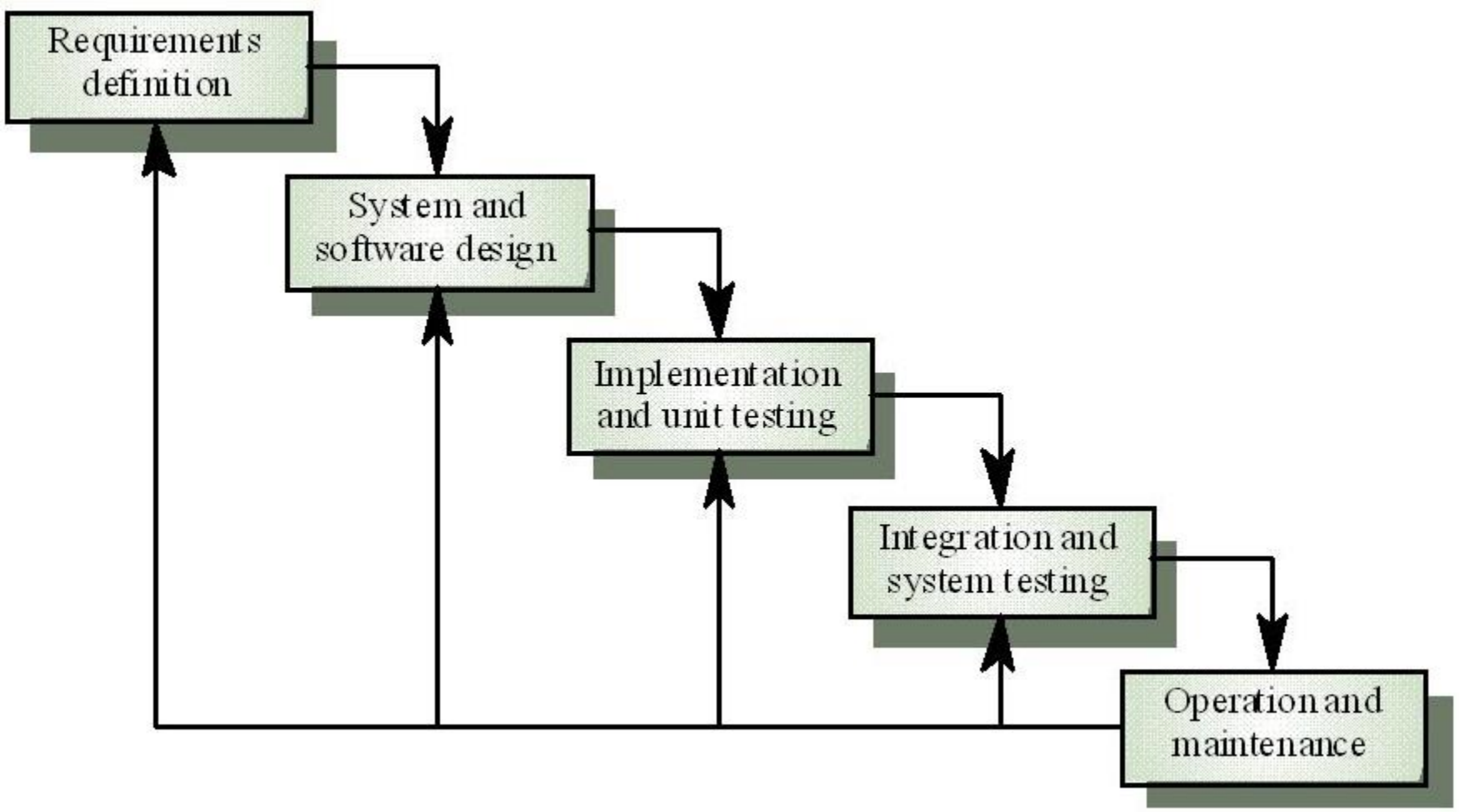
- Formalizado por Royce em 1970, é o modelo mais antigo. Suas atividades fundamentais são:
  - Análise e definição de requisitos;
  - Projeto;
  - Implementação;
  - Teste;
  - Integração.

CICLO DE VIDA
MODELO EM CASCATA

- O nome "cascata" foi atribuído em razão da sequência das fases, onde cada fase só começa quando a anterior termina
  - E da transmissão do resultado da fase anterior como entrada para a fase atual (o fim de cada fase resulta em um documento aprovado)

- Nesse modelo, portanto, é dada muita ênfase às fases de análise e projeto antes de partir para a programação, a fim de que o objetivo do software esteja bem definido e que sejam evitados retrabalhos

CICLO DE VIDA
MODELO EM CASCATA



MODELOS INCREMENTAIS

- Os modelos sequenciais pressupõem que o sistema é entregue completo, após a realização de todas as atividades do desenvolvimento.
  - Entretanto, os clientes não estão mais dispostos a esperar o tempo necessário para tal (sobretudo, quando se trata de grandes sistemas).

- No desenvolvimento incremental, o sistema é dividido em subsistemas ou módulos, tomando por base a funcionalidade.

- Os incrementos (ou versões) são definidos, começando com um pequeno subsistema funcional que, a cada ciclo, é acrescido de novas funcionalidades.
  - As funcionalidades providas anteriormente podem ser modificadas para melhor satisfazer às necessidades dos clientes / usuários.

MODELO INCREMENTAL

- O modelo incremental pode ser visto como uma filosofia básica que comporta diversas variações.

- O princípio fundamental é que, a cada ciclo ou iteração, uma versão operacional do sistema será produzida e entregue para uso ou avaliação detalhada do cliente.
  - Requisitos têm de ser minimamente levantados e há de se constatar que o sistema é modular, de modo que se possa planejar o desenvolvimento em incrementos.

MODELO INCREMENTAL

- O processo incremental propõe a aplicação do desenvolvimento cascata de forma iterativa, isto é, o sistema é desenvolvido por incrementos (subconjuntos da funcionalidade do sistema).

- **Características do modelo:**
  1. O desenvolvimento ocorre em várias iterações, cada uma delas resultando em extensão de funcionalidade e/ou maior conhecimento do sistema.
  2. Os maiores riscos devem ser tratados nas primeiras iterações.
  3. Cada iteração inclui integração e teste.
  4. Uma versão executável é produzida ao final de cada iteração, sendo testada e integrada com o resto do sistema.

MODELOS EVOLUTIVOS

- Sistemas de software evoluem ao longo do tempo.
  - Seus requisitos, muitas vezes, são difíceis de serem estabelecidos ou mudam com frequência ao longo do desenvolvimento.
  - Assim, é importante ter como opção modelos de ciclo de vida que lidem com incertezas e acomodem melhor as contínuas mudanças.

- Alguns modelos incrementais tomam por pressuposto que os requisitos são bem definidos.

- Modelos evolucionários ou evolutivos buscam preencher essa lacuna.

MODELOS EVOLUTIVOS

- Modelos incrementais têm por base a entrega de versões operacionais desde o primeiro ciclo.
  - Os modelos evolutivos não têm essa preocupação.

- À medida que o desenvolvimento avança e os requisitos vão ficando mais claros e estáveis, protótipos vão dando lugar a versões operacionais, até que o sistema completo seja construído.

- Quando o problema não é bem definido e ele não pode ser totalmente especificado no início do desenvolvimento, deve-se optar por um modelo evolutivo.

MODELO EVOLUTIVO

1. Versões parciais são desenvolvidas para atendimento aos requisitos conhecidos inicialmente.
2. Primeira versão utilizada para refinar os requisitos de uma segunda versão.
3. A partir do conhecimento sobre os requisitos, obtido com o uso, continua-se o desenvolvimento, evoluindo o produto.

- Modelo baseado em:
  - Versão inicial de implementação.
  - Entrega de implementações intermediárias.
  - Verificações constantes pelo usuário até entrega de uma versão final.

REFERÊNCIA

- Slides baseados nas aulas do professor Eliezio Soares – IFRN
  - http://docente.ifrn.edu.br/elieziosoares/disciplinas/projeto-de-software
- https://brunobrum.wordpress.com/2011/04/27/principais-tecnicas-de-levantamento-de-requisitos-de-sistemas/
- https://www.devmedia.com.br/tecnicas-para-levantamento-de-requisitos/9151
- https://www.devmedia.com.br/ciclos-de-vida-do-software-artigo-revista-engenharia-de-software-magazine-36/21099