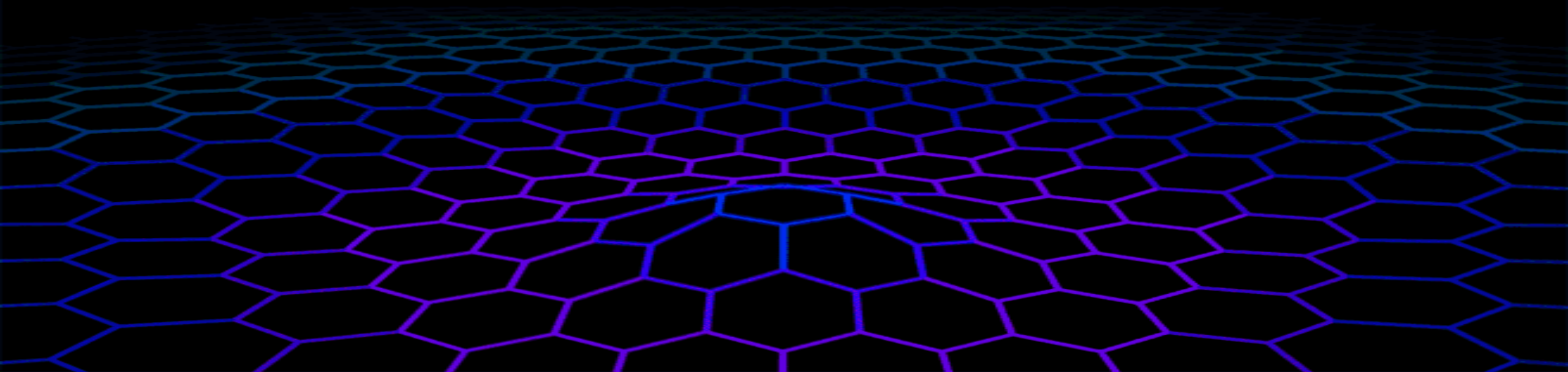
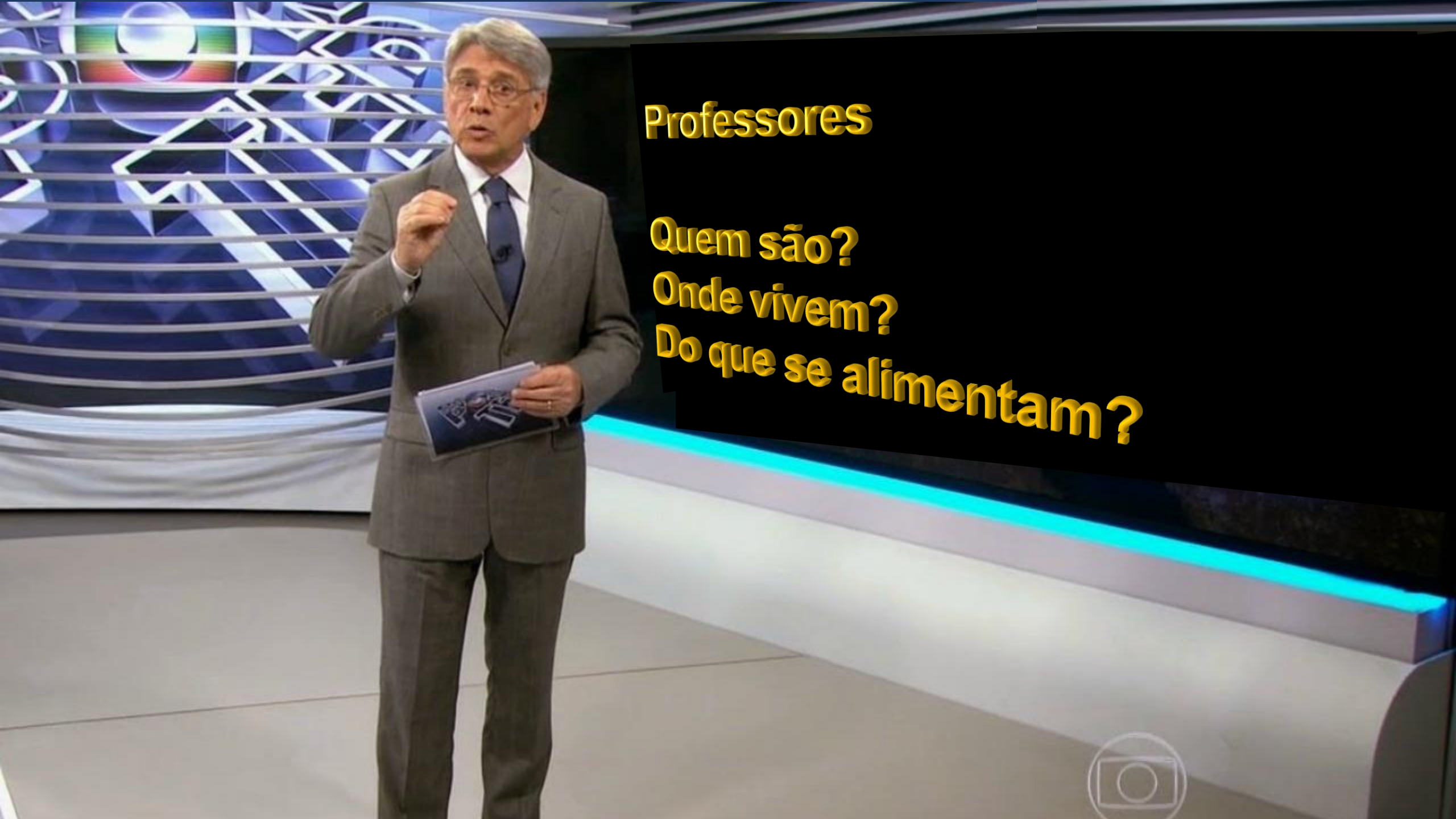
Faculdades senac

Engenharia de Software 1




BIENVENIDOS


Professores
Quem são?
Onde vivem?
Do que se alimentam?

Quem é Gladimir?

- 54 anos mas com corpinho de 53
- Mais de 34 anos de experiência profissional
- Mais de 25 anos lecionando em ensino superior
- Mestre em Educação e Tecnologia pelo IFSul
- Leciona em todos os cursos da Faculdade Senac
- Prefere Internet das Coisas do que as Coisas da Internet.
- Matou mais de 1.000 no Battlefield 2 (todos na faca)
- É um cara sério, mas faz cosplay de Stormtrooper

QUE A FORÇA ESTEJA COM VOCÊS!

gladimircc

BATTLE.NET

Origin

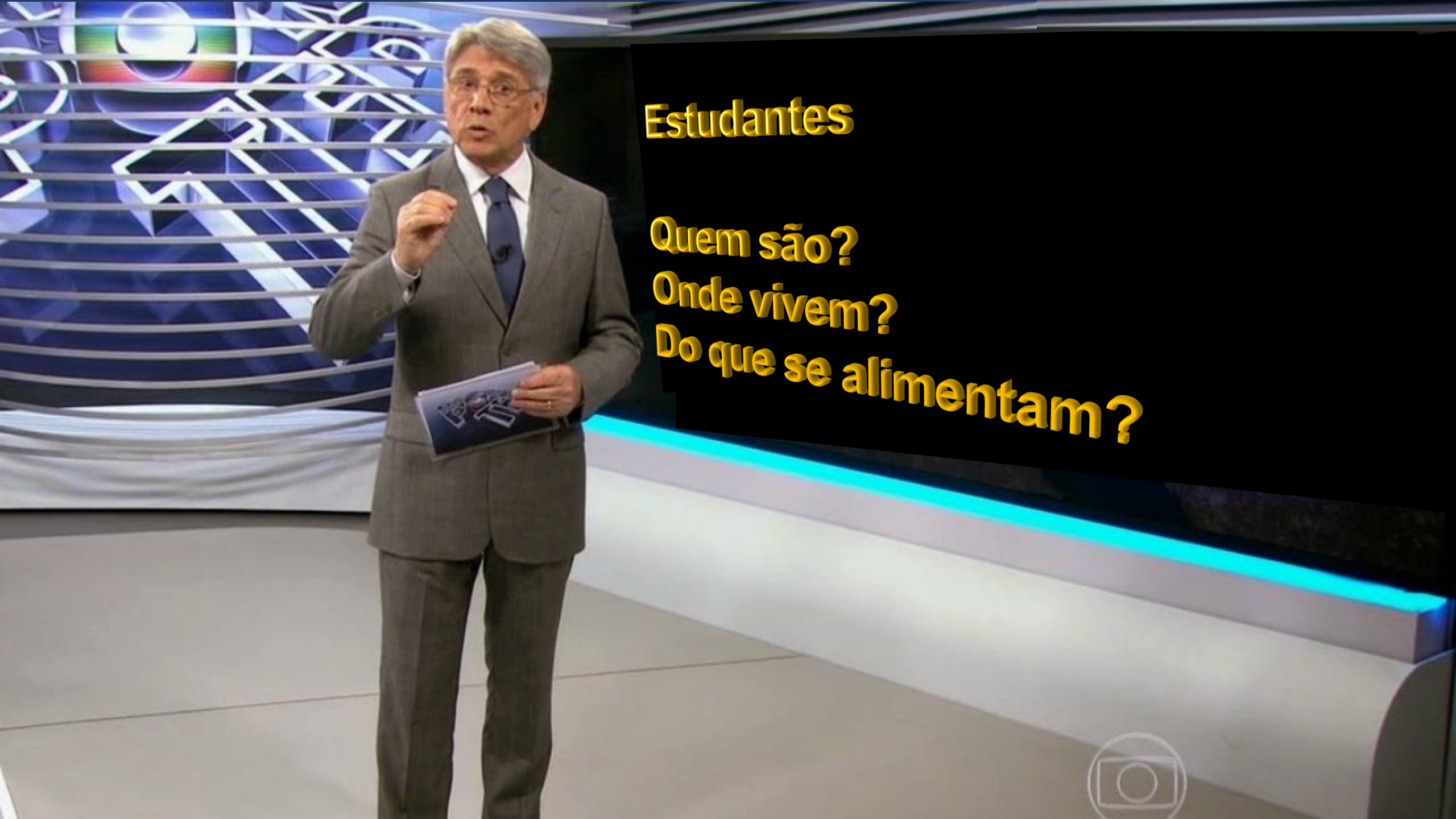
STEAM

XBOX LIVE

EPIC GAMES

UBISOFT

gladimir@gmail.com


Estudantes
Quem são?
Onde vivem?
Do que se alimentam?

CONVERGIR

Hoje – 10h30 (sala 102)

PLANO DE ENSINO

CARACTERIZAÇÃO DA UNIDADE CURRICULAR

Modelagem e especificação de necessidades a serem atendidas por um sistema de software na abordagem de análise orientada a objetos.

COMPETÊNCIA ESSENCIAL

Analisar problemas e oportunidades e propor soluções de software aderentes para elas.

ELEMENTOS DE COMPETÊNCIA

Compreender os papéis envolvidos no desenvolvimento de software.

Compreender as principais técnicas de levantamento de requisitos.

Propor abstrações coerentes para um software de acordo com o domínio de problema abordado.

Construir diagramas utilizando corretamente a especificação da UML.

BASES TECNOLÓGICAS - EMENTA

Introdução à Engenharia de Software.

Compreensão dos papéis envolvidos no desenvolvimento de software.

O papel do Analista de Sistemas.

Modelos de Processo de Software.

Identificação do problema: técnicas e artefatos relacionados.

Requisitos funcionais e não funcionais.

Definição de personas.

Histórias do usuário.

Casos de uso.

UML e artefatos em nível de análise.

BIBLIOGRAFIA BÁSICA

FERNANDES, João M. e MACHADO, Ricardo J. **Requisitos em projetos de software e de sistemas de informação**. Novatec. Edição 1. 2017.

GUEDES, Gilleanes T. A. **UML 2 - Uma Abordagem Prática**. Novatec. Edição 3. 2018.

DEBASTIANI, Carlos Alberto. **Definindo Escopo em Projetos de Software**. Novatec. Edição 1. 2015.

BIBLIOGRAFIA COMPLEMENTAR

BARNES, David J.; KÖLLING, Michael. *Programação orientada a objetos com Java*. São Paulo: Pearson Prentice Hall, 2004.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML: guia do usuário**. 2. ed. rev. atual. Rio de Janeiro: Elsevier, 2012.

LARMAN, Craig. **Utilizando UML e padrões: uma introdução à análise e ao projeto orientados a objetos e ao processo unificado**. 3. ed. Porto Alegre: Bookman, 2007.

PAULA FILHO, Wilson de Pádua. **Engenharia de software: fundamentos, métodos e padrões**. 3. ed. Rio de Janeiro: LTC, 2009.

RUMBAUGH, James et al. **Modelagem e projetos baseados em objetos**. Rio de Janeiro: Elsevier, 2004.

FINAL DO PLANO DE ENSINO

Introdução

Introdução

Engenharia de Software é uma área da computação voltada à **especificação**, **desenvolvimento** e **manutenção** de sistemas de software, com aplicação de tecnologias e práticas de gerência de projetos, visando **organização**, **produtividade** e **qualidade**.

Qual a importância da Engenharia de Software?

Pesquisas na área de desenvolvimento de software constataram que os principais problemas de um software vêm do seu desenvolvimento;

A Engenharia de Software busca resolver isso, melhorando a qualidade do desenvolvimento para termos menos erros no futuro.

Evolução

Técnicas da Engenharia de Software vem sendo criadas desde a década de 60;

Muitas começaram a ter destaque apenas após a década de 90;

Focando no problema

O software é o combustível utilizado pelos negócios modernos.

Construir e manter softwares de qualidade é e se tornará cada vez mais difícil.

Qual será a razão dessa dificuldade?

Quais softwares vocês consideram muito bons?

Falhas

Geralmente os projetos de desenvolvimento de software falham devido às seguintes causas:

Gerência “por demanda” dos requisitos;

Comunicação ambígua e imprecisa;

Arquitetura fracamente definida;

Complexidade subestimada;

Testes insuficientes;

Entre outros.

Metodologias de desenvolvimento

Através de uma metodologia de desenvolvimento de software podemos tratas essas causas;

Os sintomas serão eliminados e será mais fácil desenvolver e manter um software de qualidade de forma previsível e que possa ser repetida.

Desenvolvimento de software

Vocês foram contratados pela Faculdade Senac Pelotas a fim de desenvolver um software para ser utilizado pelos alunos e professores.

Seu objetivo é que os alunos possam acompanhar sua frequência e seus conceitos pelo sistema.

Pergunta: quais as funcionalidades mais importantes deste software?

Voltando para as metodologias

O que é uma metodologia?

Segundo o dicionário Aurélio:

Metodologia é o estudo dos métodos;

Caminho pelo qual se atinge um objetivo.

Modo de proceder, maneira de agir.

Por que utilizar uma metodologia?

Pode ser um marco para iniciar as melhorias;

Traz benefícios para todo o grupo, compartilhando as experiências;

Estabelece uma linguagem comum;

É um caminho para definir metas de melhoria contínua;

Traz facilidade na manutenção de sistemas;

Reduz dependência de pessoas chaves;

Facilita o processo de testes.

O que muda?

O processo de implantação de uma metodologia tende a aumentar o trabalho e a burocracia, tornando o trabalho mais lento;

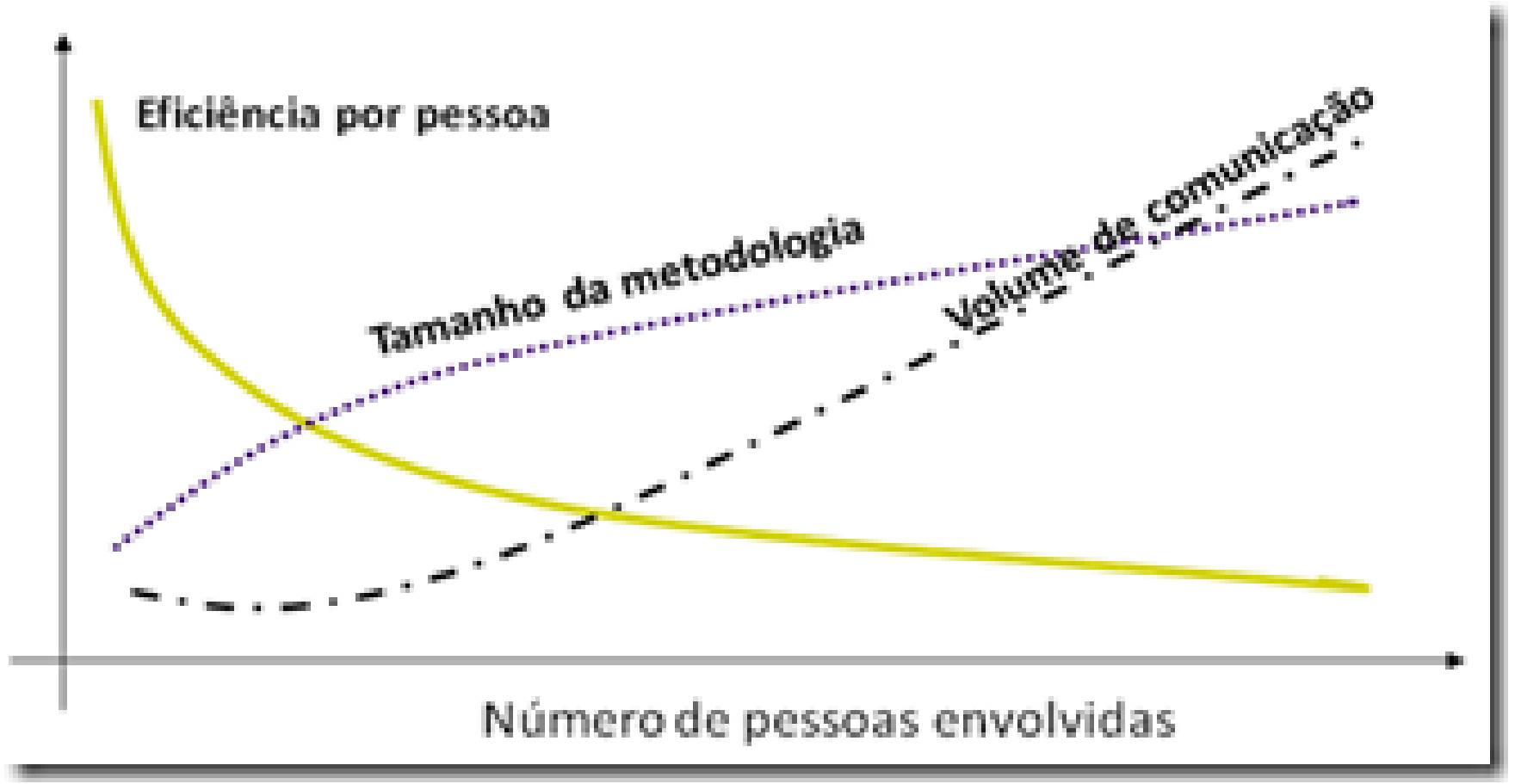
Dificuldade do aprendizado (além do treinamento custar tempo e dinheiro);

A manutenção da documentação pode ser tediosa;

Porém, a longo prazo os benefícios aparecem e não são poucos; É importante que se escolha a metodologia certa para cada situação.

Os três princípios de uma metodologia

Quanto mais pessoas envolvidas em um projeto, maior a metodologia necessária



Um pequeno aumento na metodologia, acresce muito o custo do projeto

Exemplificando

Se toda a turma se juntasse para criar uma aplicação única?

Qual tamanho de metodologia que seria necessário?

Princípios básicos do desenvolvimento

O que é um princípio?

Segundo o dicionário:

> “Um princípio é uma regra que se funda num juízo de valor e que constitui um modelo para a ação”

Essa regra é quem diz “faça” ou “não faça”;

Ao longo dos anos foram criados alguns princípios básicos para o desenvolvimento de software, os quais nos trazem vários benefícios para a criação de software.

David Hooker

David Hooker propôs 7 princípios centrais da prática e da engenharia de software como um todo.

Esses princípios se aplicam a quem deseja construir um software da melhor forma possível, escrevendo um código melhor possível.

Alan M. Davis

Alan M. Davis, escreveu o livro **201 Princípios do Desenvolvimento de Software**

(201 Principles of Development);

Esse livro é importante tanto para desenvolvedores como para analistas de sistemas e gerentes de projetos, pois abrange de forma ampla os princípios de várias naturezas.

Metodologias de desenvolvimento de software

Qual objetivo?

As metodologias de desenvolvimento de software servem para não tornar a tarefa, complexa por natureza, um verdadeiro caos;

Dependendo do projeto, os métodos tradicionais podem deixar os desenvolvedores “amarrados” a requisitos desatualizados, que não correspondem às reais necessidades do cliente.

Quantas metodologias existem?

Muitas!

Existem metodologias boas criadas a muito tempo;

Existem metodologias que não são mais usadas;

Existem metodologias novas para desenvolvimento;

Qual a ideia chave?

Muitas metodologias foram criadas ao longo dos anos e seguem sendo usadas até hoje.

A ideia é ver as metodologias mais populares para entender as diferenças entre elas e, principalmente, o que elas têm em comum.

No fim do semestre veremos também o que são metodologias ágeis.

Quais metodologias vamos estudar?

Codifica-Corrige

Modelo em Cascata (ciclo de vida clássico)

Prototipação

Modelo Espiral

Modelo por Incremento

RUP (Rational Unified Process)

Codifica-Corrige

Codifica-Corrige

Ela não é bem uma metodologia, mas é muito usada por desenvolvedores até hoje;

Nessa metodologia o desenvolvimento possui ciclos rápidos de codificação seguidos por correção;

Funcionamento

Inicialmente, o cliente pode fornecer uma especificação do que ele precisa.

Essa especificação pode ser obtida através de algumas anotações, e-mail, ou de qualquer outra fonte não muito consistente.

Desvantagens

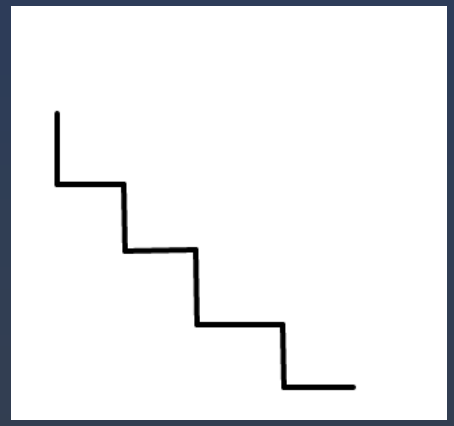
A qualidade do produto é baixa;

O sistema frequentemente se transforma num código bagunçado, com falta de adaptabilidade e reuso;

Os sistemas são difíceis de serem mantidos e aprimorados.

Os sistemas frequentemente tornam-se complicados e com baixa escalabilidade.

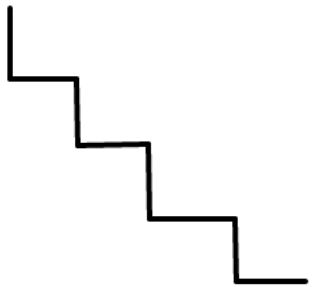
Modelo em Cascata



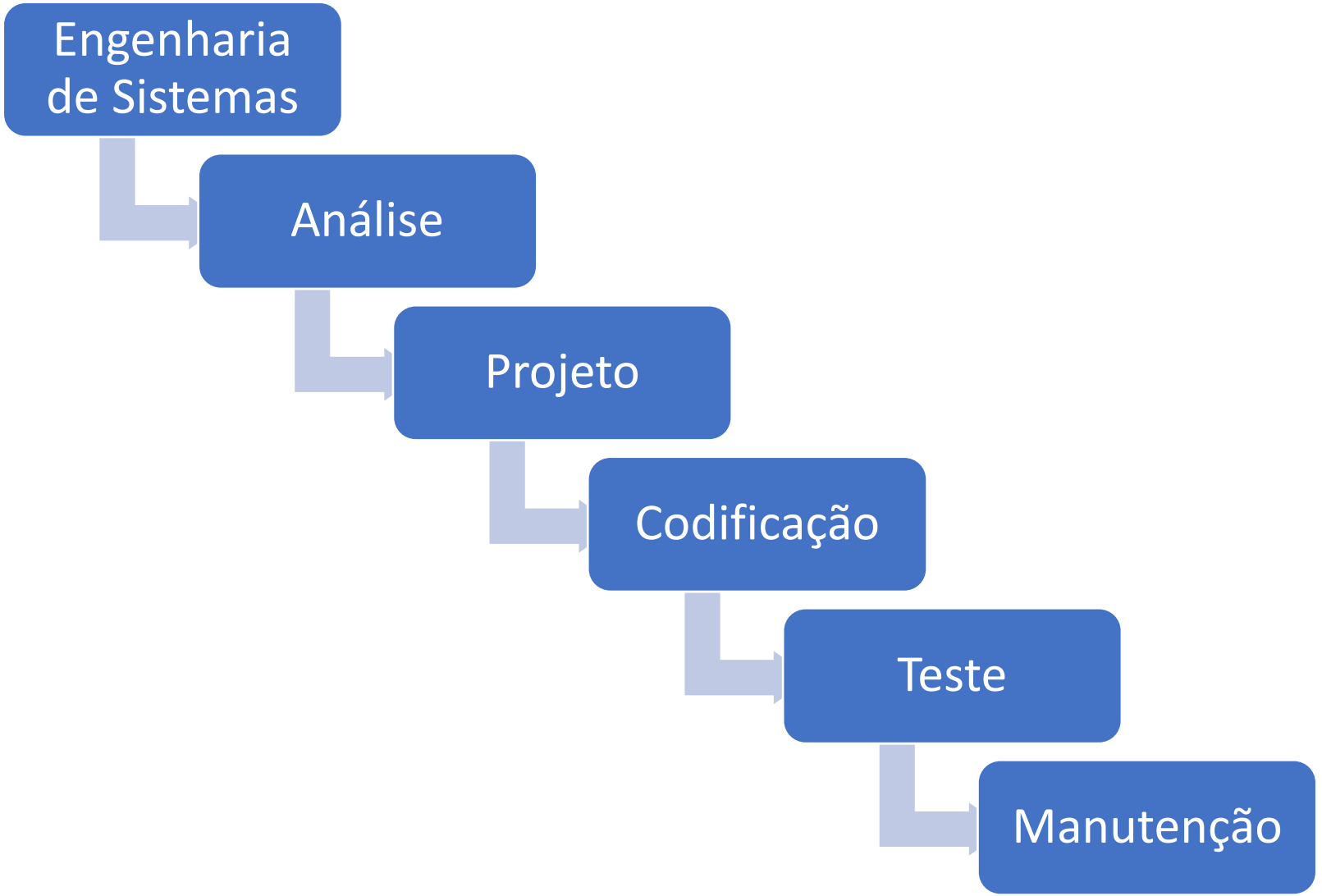
Modelo em Cascata

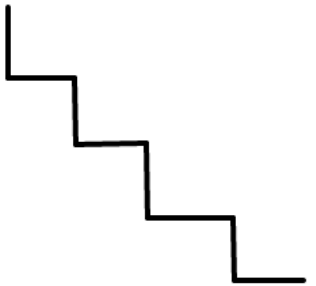
A metodologia de desenvolvimento em cascata foi desenvolvida pela marinha norte-americana nos anos 60 para permitir o desenvolvimento de softwares militares complexos.

A ideia central é que somente após um estado ser finalizado pode-se iniciar outro.



Funcionamento



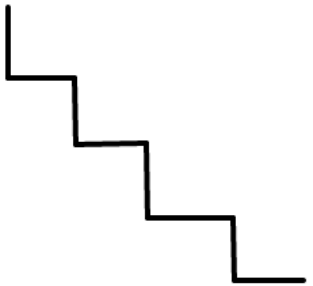


A metodologia

A Metodologia em Cascata é a metodologia mais antiga e mais amplamente usada no desenvolvimento de software;

Ela funciona bem quando os requisitos dos usuários são rígidos e podem ser conhecidos com antecedência;

Mesmo assim, alguns problemas podem surgir.



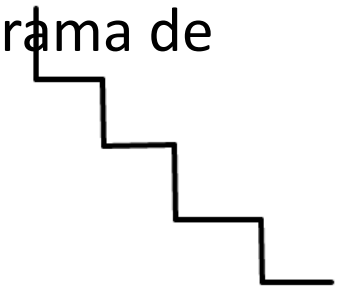
Desvantagens

Os projetos reais raramente seguem o fluxo sequencial que o modelo propõe;

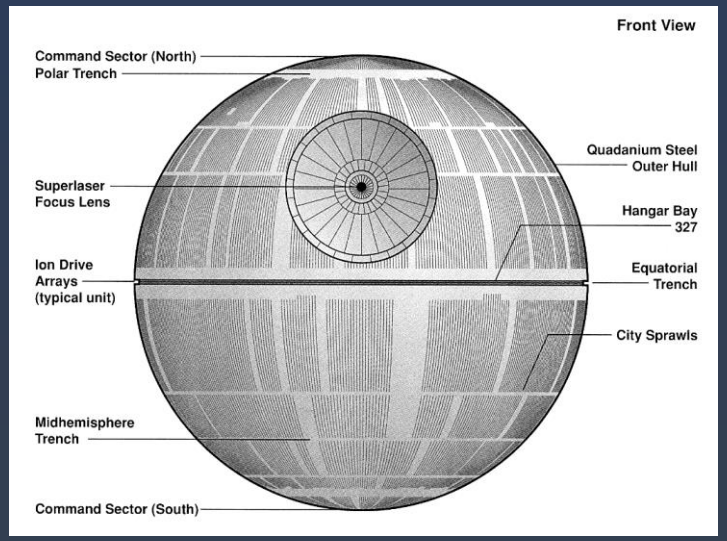
Muitas vezes é difícil para o cliente declarar todas as exigências explicitamente;

~~Os sistemas são difíceis de serem mantidos e aprimorados;~~

O cliente deve ter paciência, pois qualquer erro detectado após a revisão do programa de trabalho pode ser desastroso.



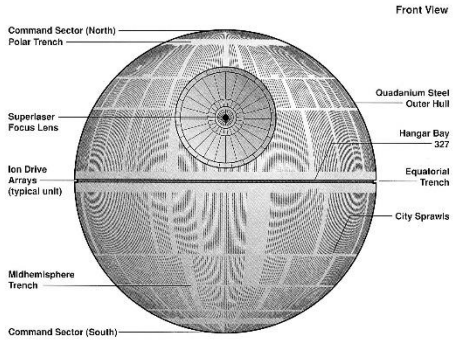
Prototipação



Prototipação

A prototipação capacita o desenvolvedor a criar um modelo de software que será implementado;

O objetivo é antecipar ao usuário final um modelo de sistema para que ele possa avaliar sua finalidade, identificar erros e omissões, quando em utilização, efetuando de imediato correções e ajustes;

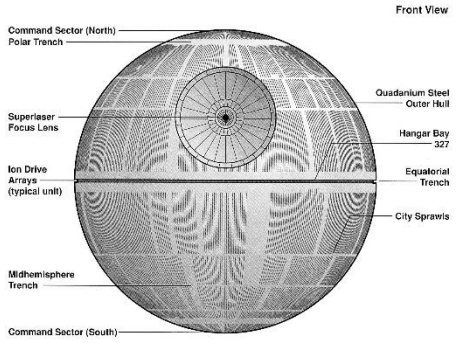


Tipos de protótipos

Um protótipo em papel ou no computador, que retrata a interação homem máquina;

Um protótipo que implemente funções específicas;

Um programa existente que executa parte ou toda a função desejada, mas que tem características que deverão ser melhoradas;



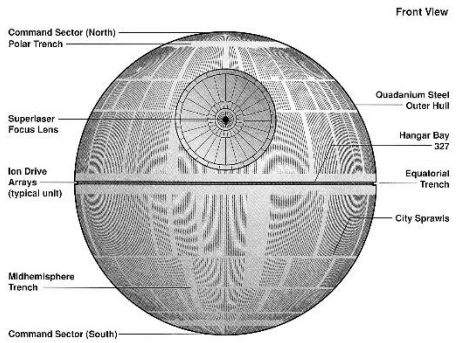
Funcionamento

Engenharia do Produto

Coleta e refinamento de requisitos

Projeto Rápido

Construção do Protótipo



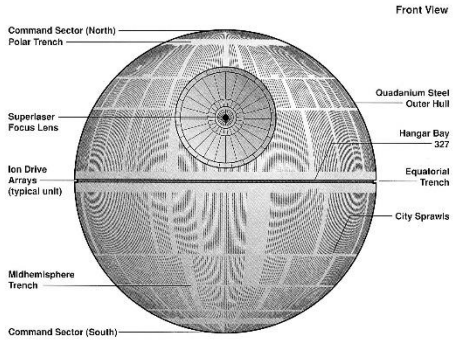
Avaliação do Protótipo pelo Cliente

Refinamento do Produto

Desvantagens

O cliente quer resultados, e, muitas vezes não saberá, ou não entenderá, que um protótipo pode estar longe do software ideal, que ele nem sequer imagina como é;

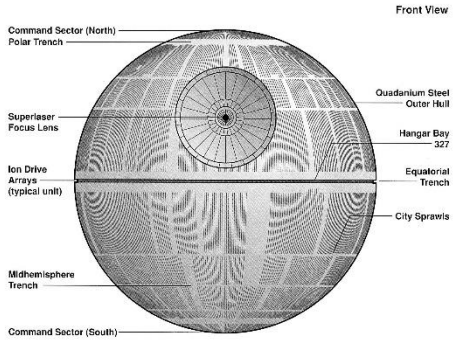
O desenvolvedor, na pressa de colocar um protótipo em funcionamento, é levado a usar uma linguagem de programação imprópria por simplesmente estar à disposição ou estar mais familiarizado. Essa atitude poderá levar a um algoritmo ineficiente;



Prototipoação

O protótipo é baseado no feedback dos usuários, devido a isso mudanças são feitas no protótipo;

Ainda que possam ocorrer problemas, o uso da prototipação é um paradigma eficiente, onde o “segredo” é o entendimento entre o desenvolvedor e o cliente.



Modelo Espiral

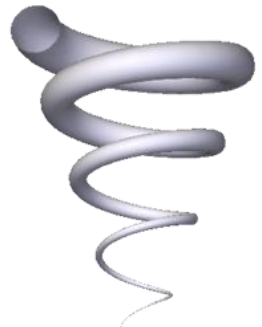
Espiral

Foi desenvolvido para abranger as melhores características do ciclo de vida clássico e da prototipação, ao mesmo tempo que adiciona um novo elemento, que é a análise de risco.

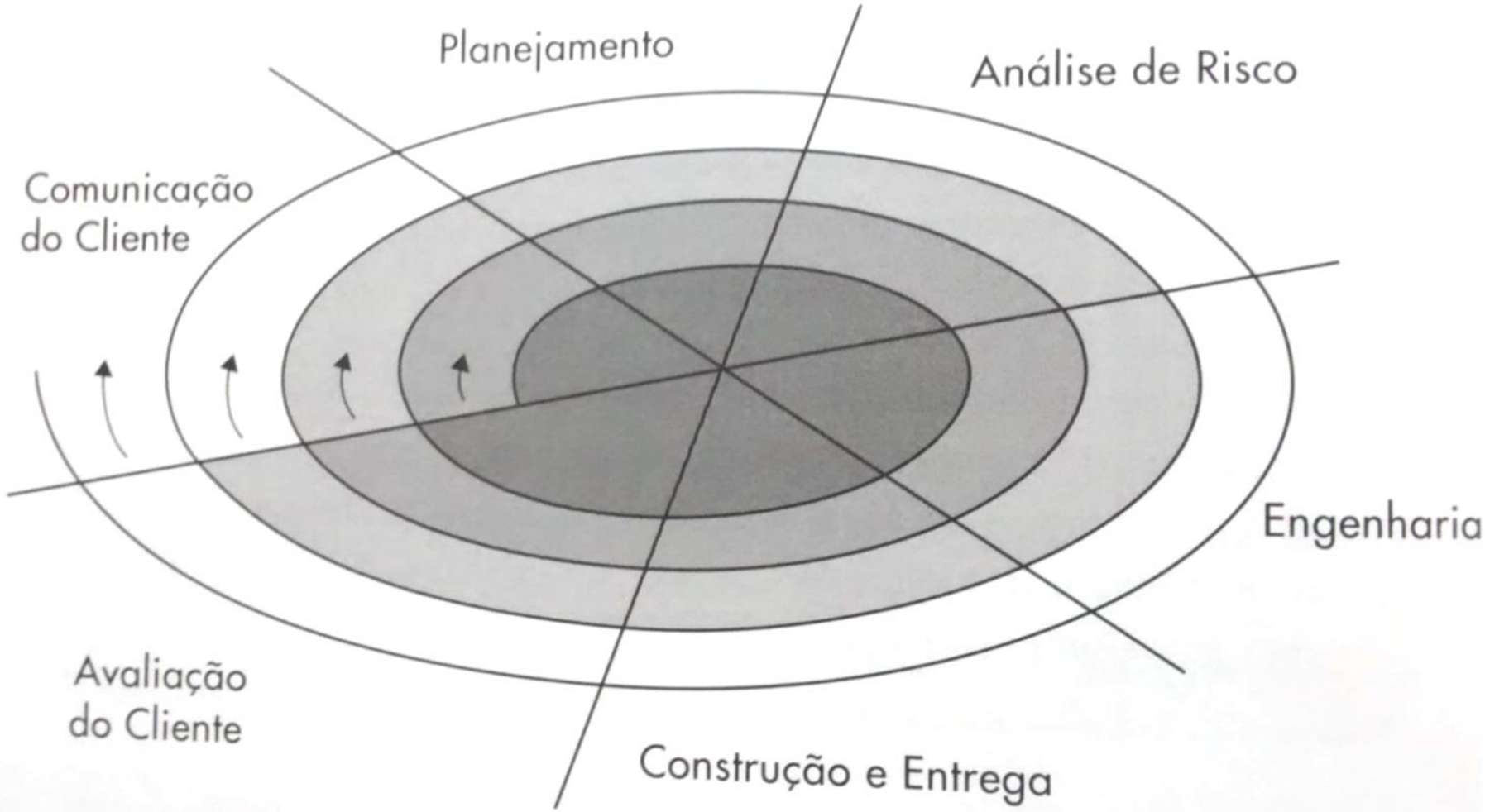
Este modelo de ciclo de vida se utiliza de protótipos por se adequar muito bem com esta filosofia de desenvolvimento;

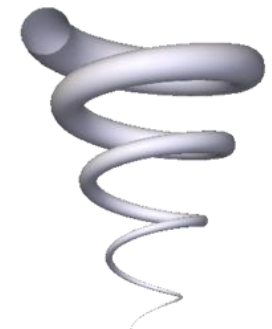
Cada passo através do ciclo inclui: planejamento, análise de risco, engenharia, construção e avaliação.

Os passos vão sendo repetidos até que um produto seja obtido.



Funcionamento





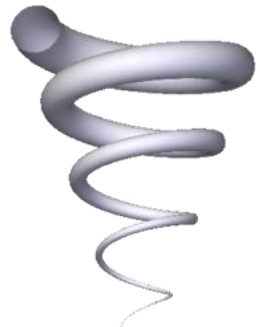
Alguns problemas

O problema a ser resolvido não está totalmente entendido;

A realidade pode mudar enquanto o sistema está sendo desenvolvido;

Pessoas que abandonam a equipe de desenvolvimento;

Ferramentas que não podem ser utilizadas ou falha em equipamentos.



Modelo V

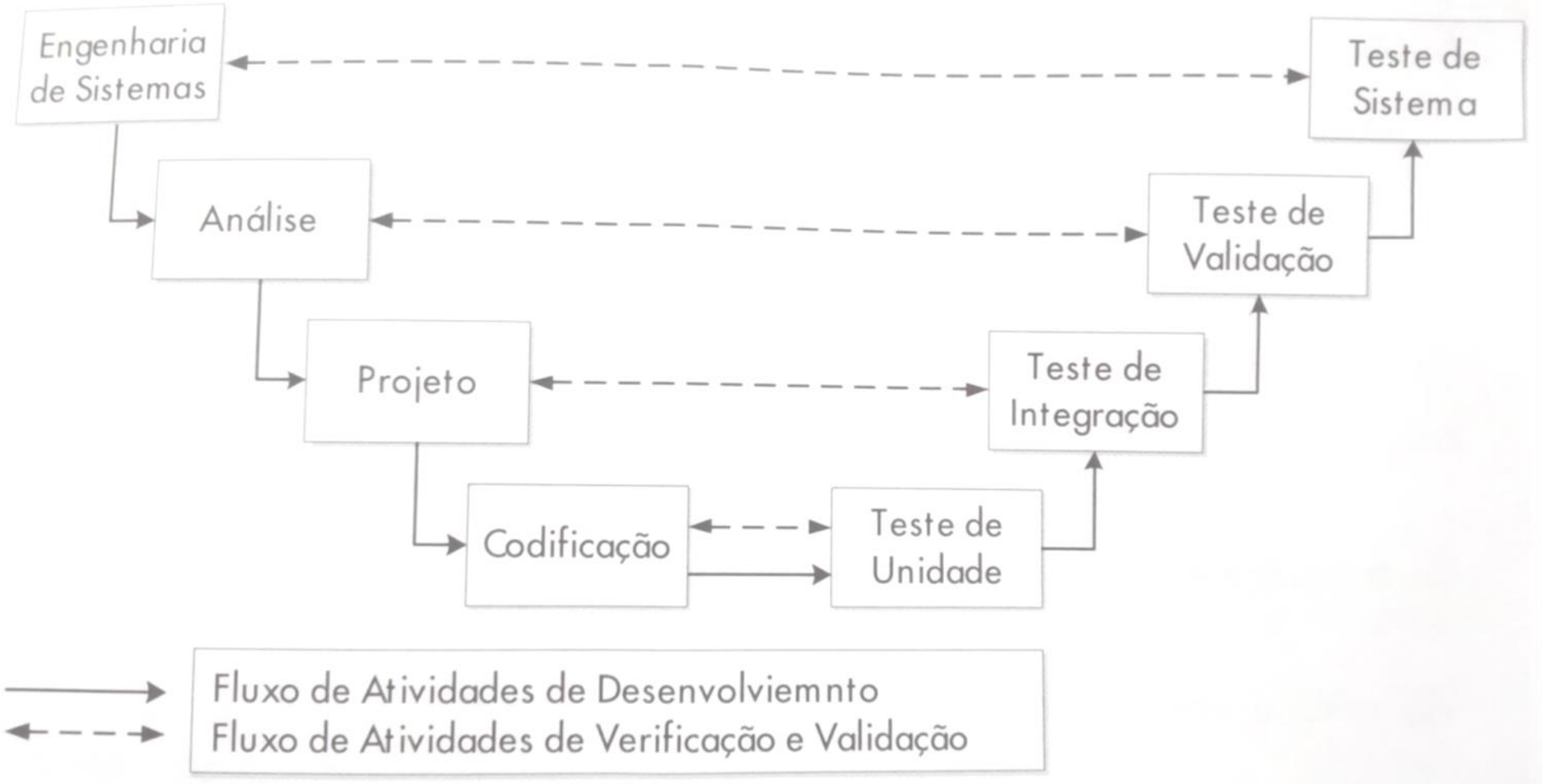
Modelo V

É um processo para desenvolvimento de sistemas, inspirado no modelo Cascata;

É usado como padrão em projetos de software na Alemanha;

O lado esquerdo dá ênfase à decomposição dos requisitos e o lado direito à integração do sistema;

Modelo V


Engenharia de Sistemas
Análise
Projeto
Codificação
Teste de Unidade
Teste de Integração
Teste de Validação
Teste de Sistema
Fluxo de Atividades de Desenvolviemnto
Fluxo de Atividades de Verificação e Validação

Principais características

Divisão entre atividades de desenvolvimento e de verificação e validação;

Clareza nos objetivos de verificação e validação;

Melhor planejamento de testes.

Modelo por Incremento

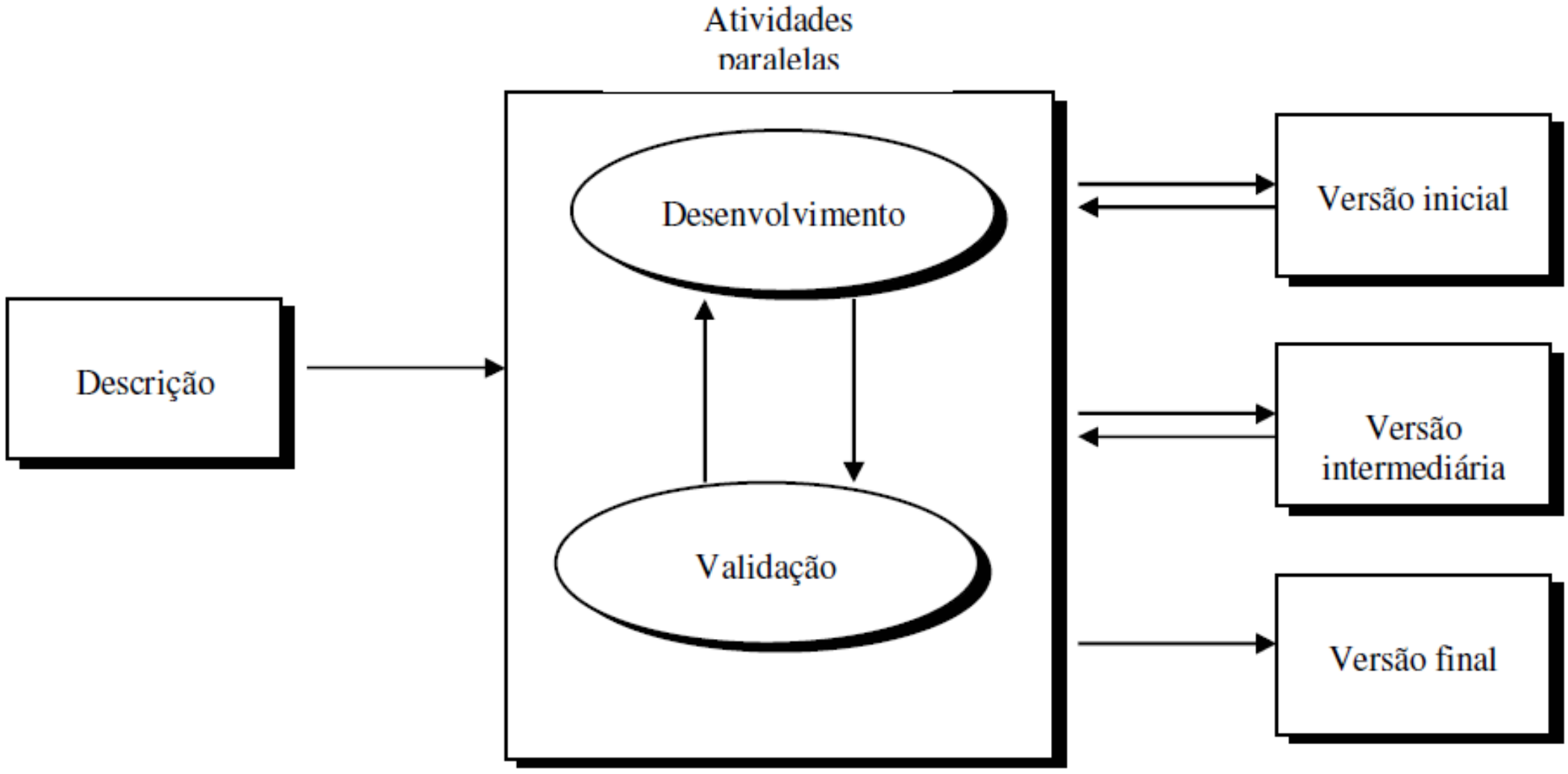
Modelo por incremento

Nos modelos por incremento, também chamado de modelo evolutivo, um único conjunto de componentes é desenvolvido simultaneamente;

Num primeiro tempo o núcleo do software é desenvolvido;

Depois os incrementos vão sendo agregados.

Funcionamento



Vantagens

Cada desenvolvimento é menos complexo;

As integrações são progressivas;

A cada interação, uma nova versão do produto pode ser distribuída;

Desvantagens

Problemas no núcleo do software, inviabilizando novos incrementos, e determinando o fim do ciclo de vida do software;

Incapacidade de integrar novos incrementos.

Desvantagens

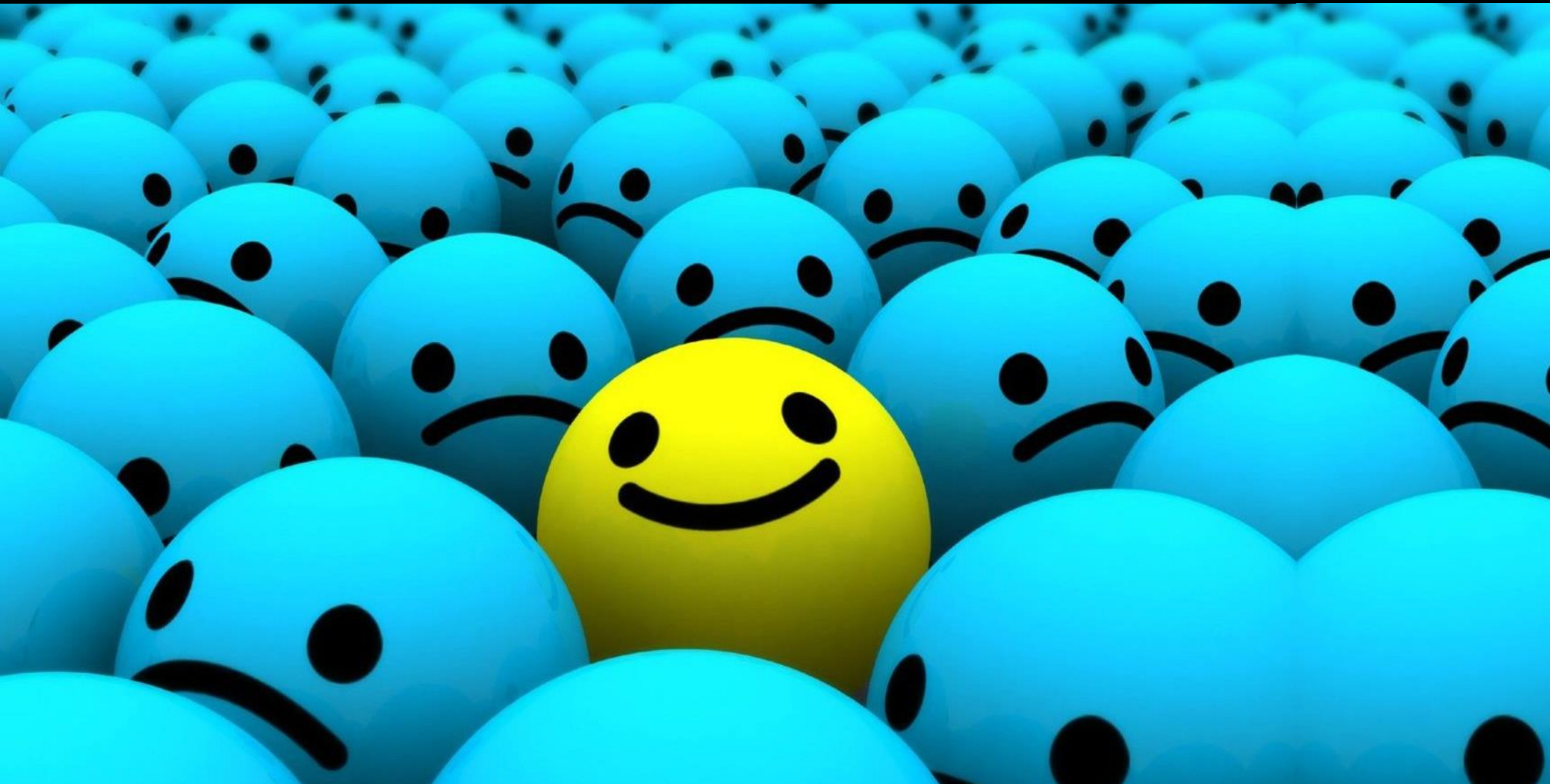
Nesse modelo de desenvolvimento incremental, temos mais uma metodologia:

- RAD (Rapid Application Development);

Fora isso, também temos algumas metodologias “pesadas” e interessantes como:

- UP (Unified Process)
  - RUP (Rational UP) criada pela IBM inspirada na UP;
  - Agile Unified Process também inspirada da UP.

Outras metodologias



Pesquise e responda:

Pesquise os 7 princípios básicos do desenvolvimento de software propostos por David Hooker;

Responda:

Esses 7 princípios básicos são suficientes para um bom desenvolvimento?