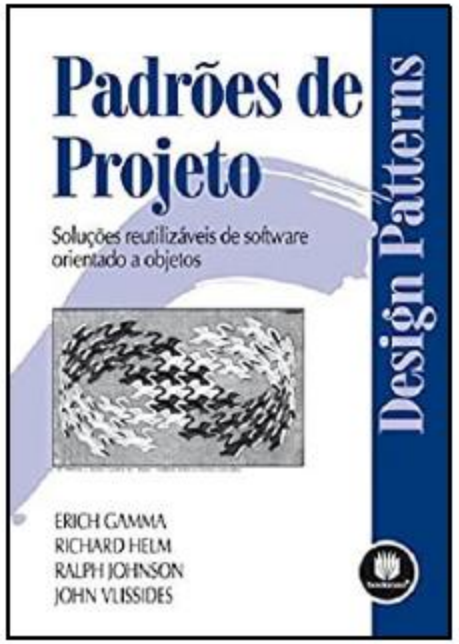
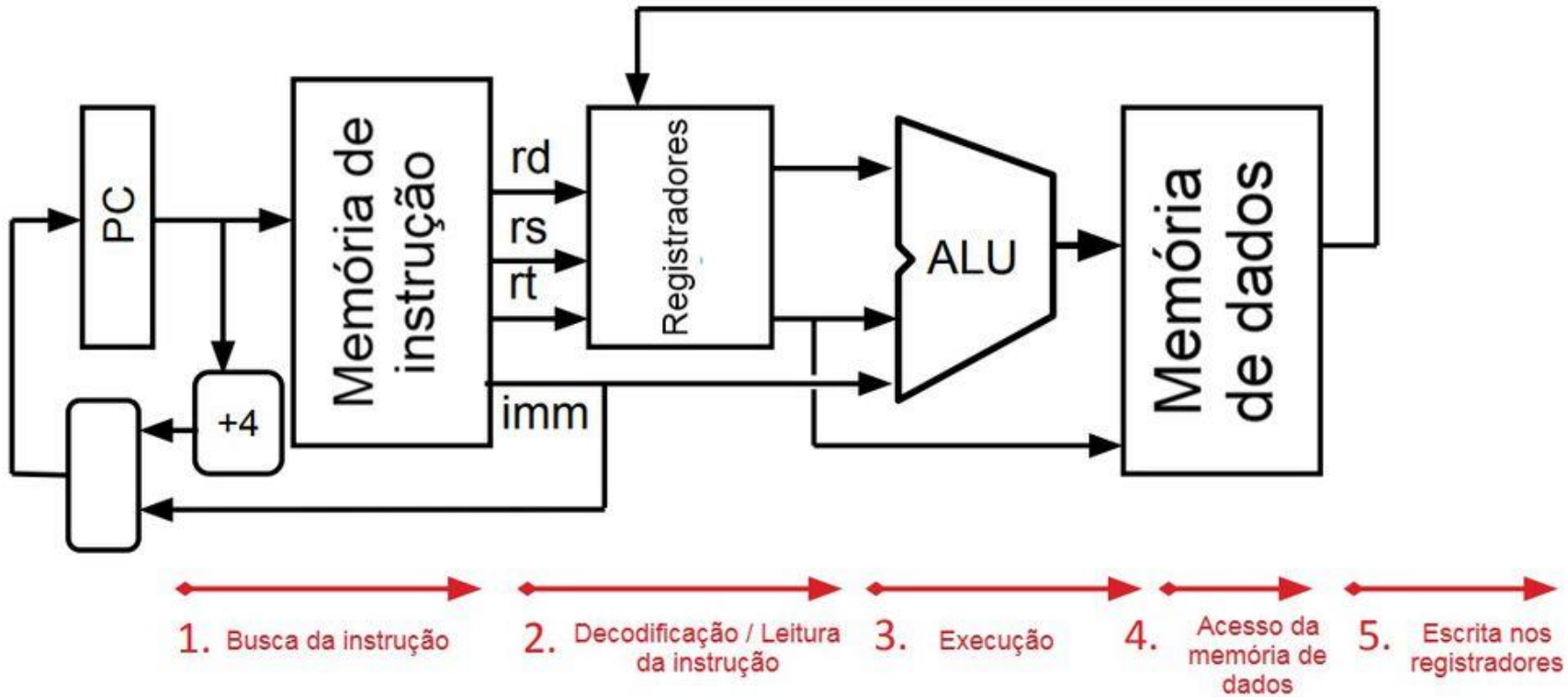
João Pedro Oliveira Batisteli

ARQUITETURA DE SOFTWARE - INTRODUÇÃO

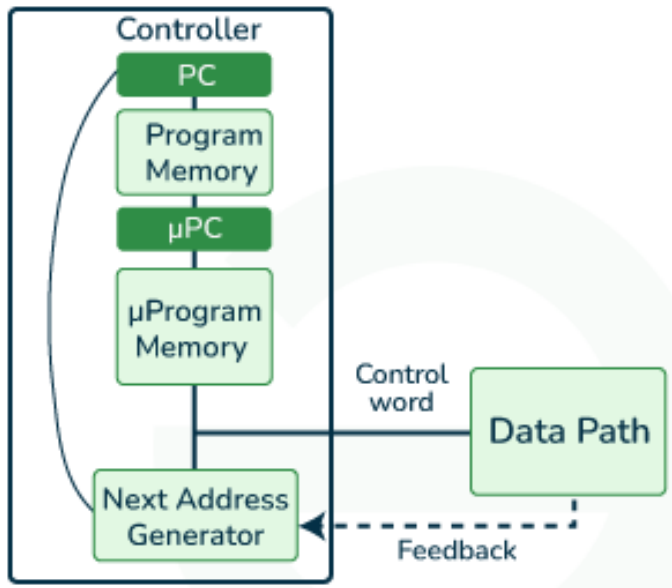
O QUE É ARQUITETURA NO CONTEXTO DA COMPUTAÇÃO?

“Architecture is about the important stuff. Whatever that is.”
– Ralph Johnson

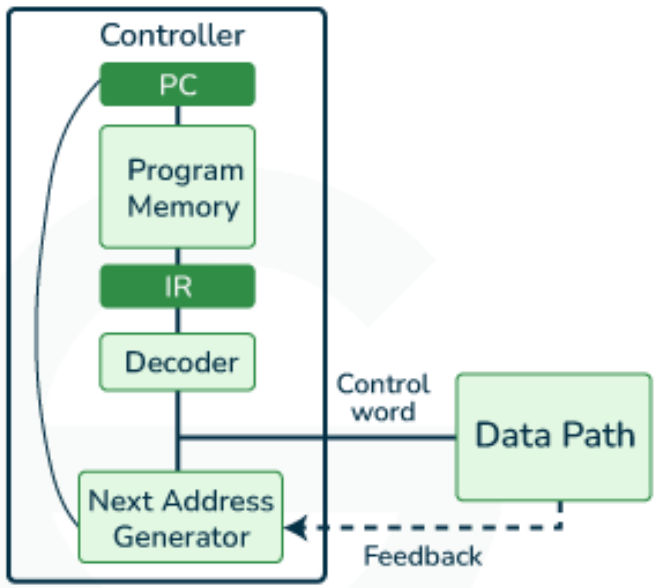




RS - Registrador do primeiro operando de origem
RT - Registrador do segundo operando de origem
RD - Registrador que recebe o resultado da operação
IMM - Endereço de memória



CISC
Complex Instructions Possible
1 Instruction = n μ-Instructions



RISC
Simple Instructions
No μ-programming
RISC PM = N x CISC PM

“Design e estrutura de um sistema computacional, abrangendo tanto o hardware quanto o software, e como seus componentes interagem entre si”

O QUE É ARQUITETURA DE SOFTWARE?

O QUE É ARQUITETURA DE SOFTWARE?

- **Existe mais de uma definição para arquitetura de software!**

- Vamos abordar algumas dessas definições e entender a diferença entre elas.

- O conceito de Arquitetura de Software surgiu nos anos 60 (com Dijkstra), mas se tornou popular nos anos 90 (Perry e Wolf).

ARQUITETURA DE SOFTWARE – DEFINIÇÃO 1

- Uma das definições mais comuns considera que arquitetura preocupa-se com projeto em mais alto nível.

- Ou seja, o foco deixa de ser a organização e interfaces de classes individuais e passa a ser em unidades de maior tamanho, sejam elas pacotes, componentes, módulos, subsistemas, camadas ou serviços, o nome não importa tanto neste primeiro momento.

ARQUITETURA DE SOFTWARE – DEFINIÇÃO 1

O foco não são unidades pequenas (classes, etc).

ARQUITETURA DE SOFTWARE – DEFINIÇÃO 1

Foco em unidades de maior tamanho

ARQUITETURA DE SOFTWARE – DEFINIÇÃO 1

- Além de possuírem um “maior tamanho”, os componentes arquiteturais devem ser relevantes para que um sistema atenda a seus objetivos.
- Ex: **suponha que você trabalhe em um sistema de informações**
  - Certamente, esse sistema inclui um **módulo de persistência**, que faz a interface com o banco de dados.

ARQUITETURA DE SOFTWARE – DEFINIÇÃO 1

- Além de possuírem um “maior tamanho”, os componentes arquiteturais devem ser relevantes para que um sistema atenda a seus objetivos.
- Ex: **suponha que você trabalhe em um sistema de informações**
  - Certamente, esse sistema inclui um **módulo de persistência**, que faz a interface com o banco de dados.
  - Esse módulo é fundamental, pois eles têm como objetivo principal automatizar e persistir informações relativas a processos de negócio.

ARQUITETURA DE SOFTWARE – DEFINIÇÃO 1

- Contra exemplo: **sistema que usa técnicas de inteligência artificial para diagnosticar doenças**
  - O sistema também possui um módulo de persistência para armazenar dados das doenças que são diagnosticadas.
  - Algum problema com esse módulo?

ARQUITETURA DE SOFTWARE – DEFINIÇÃO 1

- Contra exemplo: **sistema que usa técnicas de inteligência artificial para diagnosticar doenças**
  - O sistema também possui um módulo de persistência para armazenar dados das doenças que são diagnosticadas.
  - Algum problema com esse módulo?
  - Esse módulo, além de simples, não é relevante para o objetivo principal do sistema. Logo, ele não faz parte da sua arquitetura.

ARQUITETURA DE SOFTWARE – DEFINIÇÃO 2

- "*Arquitetura de software inclui as decisões de projeto mais importantes em um sistema.*" – *Ralph Johnson*
- Essas decisões são tão importantes que, uma vez tomadas, dificilmente poderão ser revertidas no futuro.

ARQUITETURA DE SOFTWARE – DEFINIÇÃO 2

- Essa Segunda definição é mais ampla do que a primeira.
- Ela considera que **a arquitetura não é apenas um conjunto de módulos, mas um conjunto de decisões.**
- Dentre essas decisões, inclui-se a definição dos módulos principais de um sistema.
- Outras decisões também são contempladas, como a escolha da **linguagem de programação** e do **banco de dados** que serão usados no desenvolvimento.

ARQUITETURA DE SOFTWARE – DEFINIÇÃO 2

- Uma vez que um sistema é implementado com um determinado banco de dados, dificilmente consegue-se migrar para um outro banco de dados.

EXEMPLO - DEBATE TANENBAUM-TORVALDS

- Andrew Tenenbaum x Linus Trovalds.
- Arquitetura microkernel x arquitetura monolítica.

- *“ Não somos mais o kernel simples, pequeno e hiper-eficiente que imaginei há 15 anos. Em vez disso, nosso kernel está ficando grande e inchado. E sempre que adicionamos novas funcionalidades, o cenário piora.” - Linus Trovalds*

ARQUITETURA DE SOFTWARE – CONCLUSÃO

- “A arquitetura de software é a **espinha dorsal** de um sistema, composta por seus **componentes**, as **relações entre eles** e os **princípios** que orientam o seu **projeto** e sua **evolução**.”

(BOA) ARQUITETURA DE SOFTWARE – VANTAGENS

- Modularidade
- Flexibilidade
- Manutenção e testabilidade.

(MÁ) ARQUITETURA DE SOFTWARE – EXEMPLO

- Knight Capital Group em 2012 sofreu uma perda de US$ 440 milhões em apenas 45 minutos devido à implementação incorreta de software em seus servidores.

- O problema ocorreu porque um sinal de software crítico foi reutilizado, fazendo com que códigos antigos e não utilizados fossem executados durante as operações de trading.

(MÁ) ARQUITETURA DE SOFTWARE – EXEMPLO

QUESTÕES

- Como decompor um sistema em diversos subsistemas?
  - Quais classes devem estar em cada subsistema de forma a permitir mais reutilização de subsistemas?
  - Quais as dependências entre os subsistemas?
- Como distribuir cada um dos subsistemas em diferentes nós de processamento?
  - Apenas um subsistema por nó de processamento ou mais de um?
  - Qual o impacto no resto do sistema se agruparmos ou separamos subsistemas em diferentes nós?

QUESTÕES

- Como decompor um sistema em diversos subsistemas?
  - Quais classes devem estar em cada subsistema de forma a permitir mais reutilização de subsistemas?
  - Quais as dependências entre os subsistemas?
- Como distribuir cada um dos subsistemas em diferentes nós de processamento?
  - Apenas um subsistema por nó de processamento ou mais de um?
  - Qual o impacto no resto do sistema se agruparmos ou separamos subsistemas em diferentes nós?

PADRÕES ARQUITETURAIS

- Propõem uma organização de mais alto nível para sistemas de software.
- Incluindo seus principais módulos e as relações entre eles.
- Essas relações definem, por exemplo, que um módulo A pode (ou não pode) usar os serviços de um módulo B.

PADRÕES ARQUITETURAIS

- Arquitetura em camadas
- Model-View-Controller (MVC)
- Microsserviços
- Orientada a Mensagens
- Publish/Subscribe

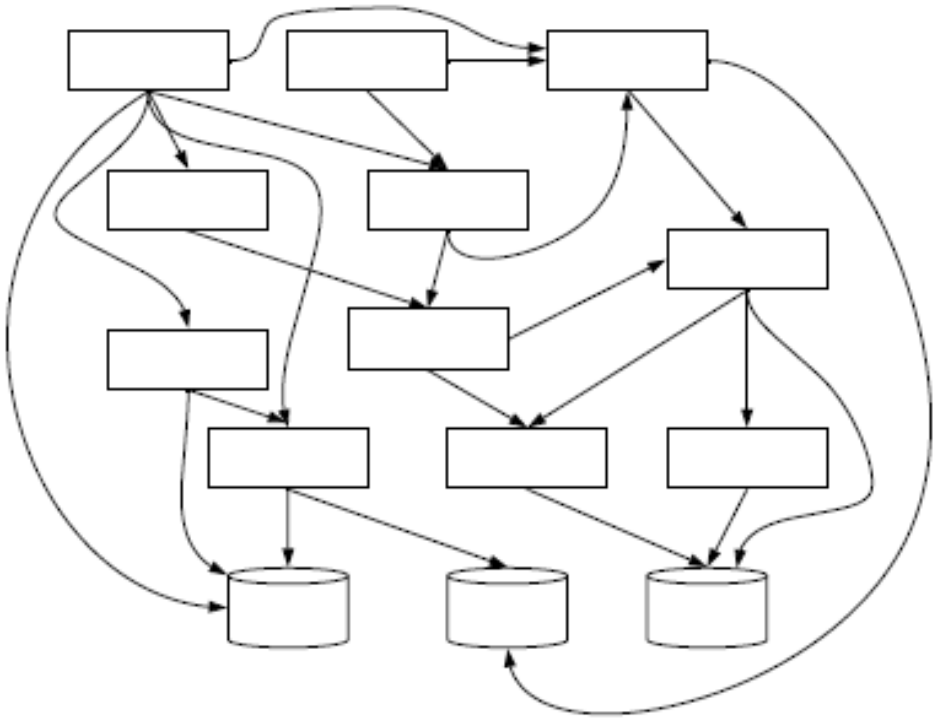
ANTI-PADRÕES ARQUITETURAIS

- Organização de sistemas que **não é recomendada**.
- O mais conhecido anti-padrão é chamado de ***big ball of mud*** (ou grande bola de lama).
- Proposto por Brian Foote e Joseph Yoder, esse padrão descreve sistemas nos quais qualquer módulo comunica-se com praticamente qualquer outro módulo do.

ANTI-PADRÕES ARQUITETURAIS

- Organização de sistemas que **não é recomendada**.
- O mais conhecido anti-padrão é chamado de ***big ball of mud*** (ou grande bola de lama).
- Proposto por Brian Foote e Joseph Yoder, esse padrão descreve sistemas nos quais qualquer módulo comunica-se com praticamente qualquer outro módulo do.

BIG BALL OF MUD

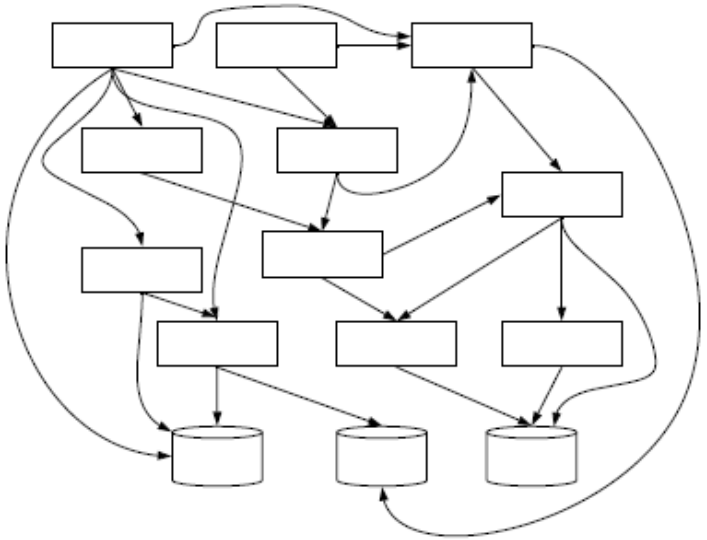


BIG BALL OF MUD

- **Não possui uma arquitetura definida!**

- É uma explosão no número de dependências, que dá origem a um espaguete de código.

- Consequentemente, a manutenção do sistema torna-se muito difícil e arriscada.

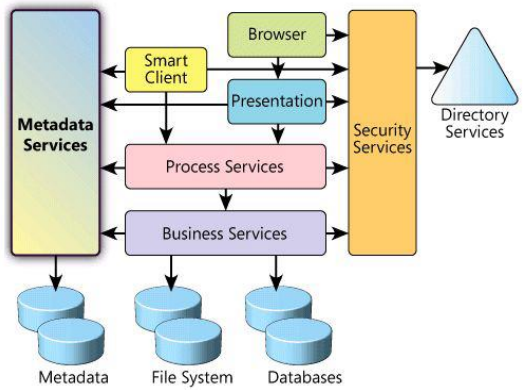


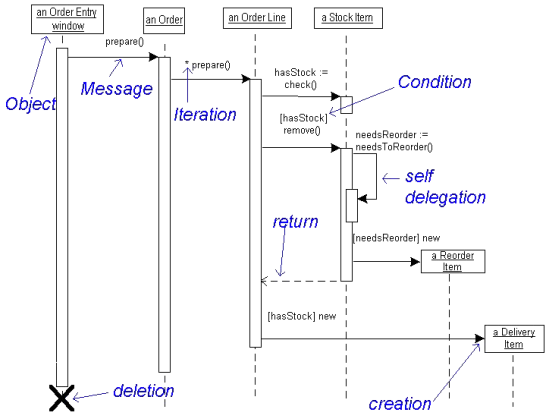
BIG BALL OF MUD

- **Artigo IEEE Software (2009)** – Santonu Sarkar e colegas (consultores da InfoSys) relatam experiência de modularização em um **grande sistema bancário**
- Sistema criado no final dos anos 90 → cresceu **10x**: de **2,5M** para **25M de linhas de código**
- Equipes com centenas de engenheiros
- Arquitetura caracterizada como "Big Ball of Mud"
  - Ex.: diretório *sources* com ~15 mil arquivos
- Problemas identificados:
  - Tempo de aprendizado de novos engenheiros: 3 → 7 meses em 5 anos
  - Correção de bugs frequentemente introduzia novos bugs Implementação de novas funcionalidades (mesmo simples) cada vez mais lenta
- **Casos como esse não são exceção – são mais comuns do que se imagina**

PROJETO X ARQUITETURA

- O projeto de software é composto de duas atividades
  - O projeto da arquitetura de software
    - Mais **alto nível** com granularidade **macro** do software, definindo apenas os components e a comunicação/iteração entre eles
    - **Satisfaz os requisitos de qualidade**
  - O projeto detalhado do software
    - Projeto mais **baixo nível**, com granularidade **micro** do software, definindo seus objetivos e a forma de colaboraçao entre eles
    - **Satisfaz os requisites funcionais.**





O PROCESSO DE ARQUITETURA DE SOFTWARE - STAKEHOLDERS

- Participantes (interessados)
  - Analista de requisitos - identifica os requisitos
  - Arquiteto de software – cria a arquitetura (pode ser um time com um arquiteto líder)
  - Projetista ou Desenvolvedor – implementa os componentes

- Outros: cliente/usuário, testador, gerente de projeto, programador, secretários, etc.

ARQUITETO DE SOFTWARE

- ser capaz de reconhecer estruturas comuns em sistemas já desenvolvidos.
- usar o conhecimento sobre arquiteturas existentes para tomar decisões de projeto em novos sistemas.
- ser capaz de realizar uma descrição formal da arquitetura de um sistema a fim de analisar as propriedades do sistema
- apresentar a arquitetura para outras pessoas

ARQUITETO DE SOFTWARE

Habilidades:

- compreender profundamente o domínio e as tecnologias pertinentes
- dominar técnicas de modelagem e metodologias de desenvolvimento
- entender as estratégias de negócios da instituição onde atua
- conhecer produtos, processos e estratégias de concorrentes

ARQUITETO DE SOFTWARE

Tarefas:

- Especificação da arquitetura do software e das bases para o sistema de acordo com os requisitos do cliente.
- Modelagem
- Análise de trade-offs e viabilidade
- Prototipação, simulação e realização de experimentos
- Análise de tendências tecnológicas