OS PROCESSOS PLANEJE-E-DOCUMENTE

ENGENHARIA DE SISTEMAS DE INFORMAÇÃO

Daniel Cordeiro

8 de agosto de 2017

Escola de Artes, Ciências e Humanidades | EACH | USP

http://edisciplinas.usp.br

(24 INSCRITOS / 33 MATRICULADOS)

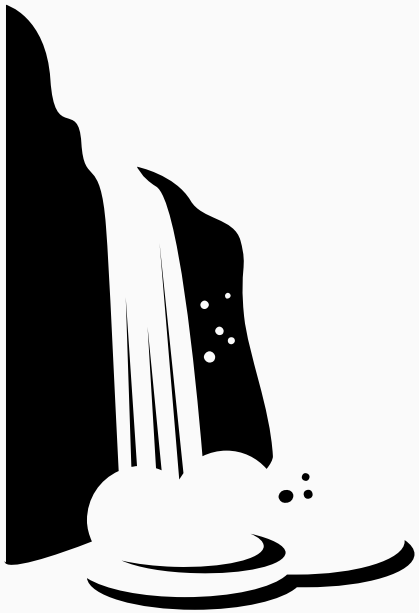
COMO EVITAR AS DESONRAS DO DESENVOLVIMENTO DE SOFTWARE?

- Será que não podemos construir software de forma que possamos predizer o cronograma, custo e qualidade da mesma forma que engenheiros constroem pontes?
- Se for possível, que tipo de processo de desenvolvimento tornaria a construção de software uma atividade mais previsível?

- A ideia é trazer a disciplina usada pelos engenheiros para a construção de software
  - Termo criado 20 anos após o primeiro computador
  - Estuda métodos de desenvolvimento tão previsíveis em qualidade, custo e tempo como a engenharia civil
- “Planeje-e-Documente”
  - Antes de escrever qualquer linha de código, o gerente do projeto estabelece um plano
  - Escreve uma descrição detalhada de todas as fases do plano
  - Progresso do projeto é medido sempre em comparação ao plano original
  - Mudanças no projeto devem ser repassadas à documentação e, possivelmente, também ao plano original

1º processo de desenvolvimento: cascata (1970)

Processo de desenvolvimento com um “ciclo de vida” em Cascata (*Waterfall*) composto de 5 fases:



Created by Robert Bjurshagen from the Noun Project

1. Análise de requisitos & especificação
2. Projeto de arquitetura
3. Implementação & Integração
4. Verificação
5. Operação & Manutenção

Ideia: completar cada fase antes de iniciar a próxima

- Por quê? Quanto antes encontrarmos um bug, mais barato será
- Uso extensivo de documentos/fases para novos desenvolvedores

QUÃO BOM É O MODELO EM CASCATA?

E os usuários exclamaram com uma risada e uma provocação: "isso é exatamente o que nós pedimos, mas não o que nós queríamos. — Anônimo"

- Normalmente quando um consumidor vê o produto pronto, ele quer mudar o produto

QUÃO BOM É O MODELO EM CASCATA?

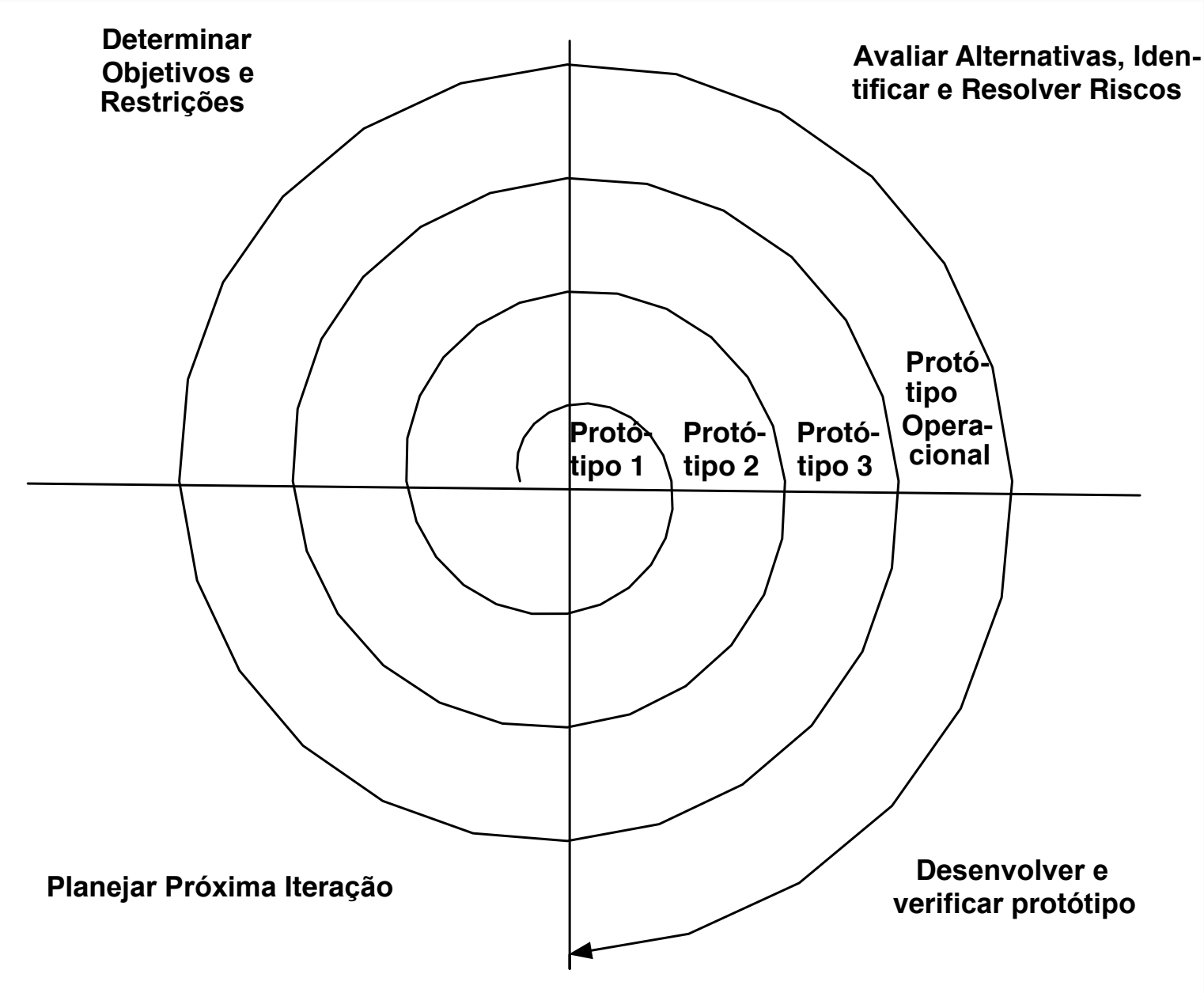
- “Se planeje para jogar fora uma [implementação]; você vai acabar tendo que fazer isso de qualquer jeito” – Fred Brooks, Jr. (vencedor do prêmio Turing em 1999)
- Normalmente, após terminar a primeira implementação, os desenvolvedores aprendem qual o jeito certo de resolver esse problema que deveria ter sido usado desde o início

Ciclo de vida Espiral (1986)

Ciclo de vida Espiral (*Spiral Lifecycle*)

- Combine o Planeje-e-Documente com protótipos
- Ao invés começar com o planejamento e documentação de todos os requisitos desde o início, planeje e documente os requisitos e desenvolva o protótipo do sistema, mas em várias iterações.

CICLO DE VIDA ESPIRAL



ESPIRAL: PRÓS & CONTRAS

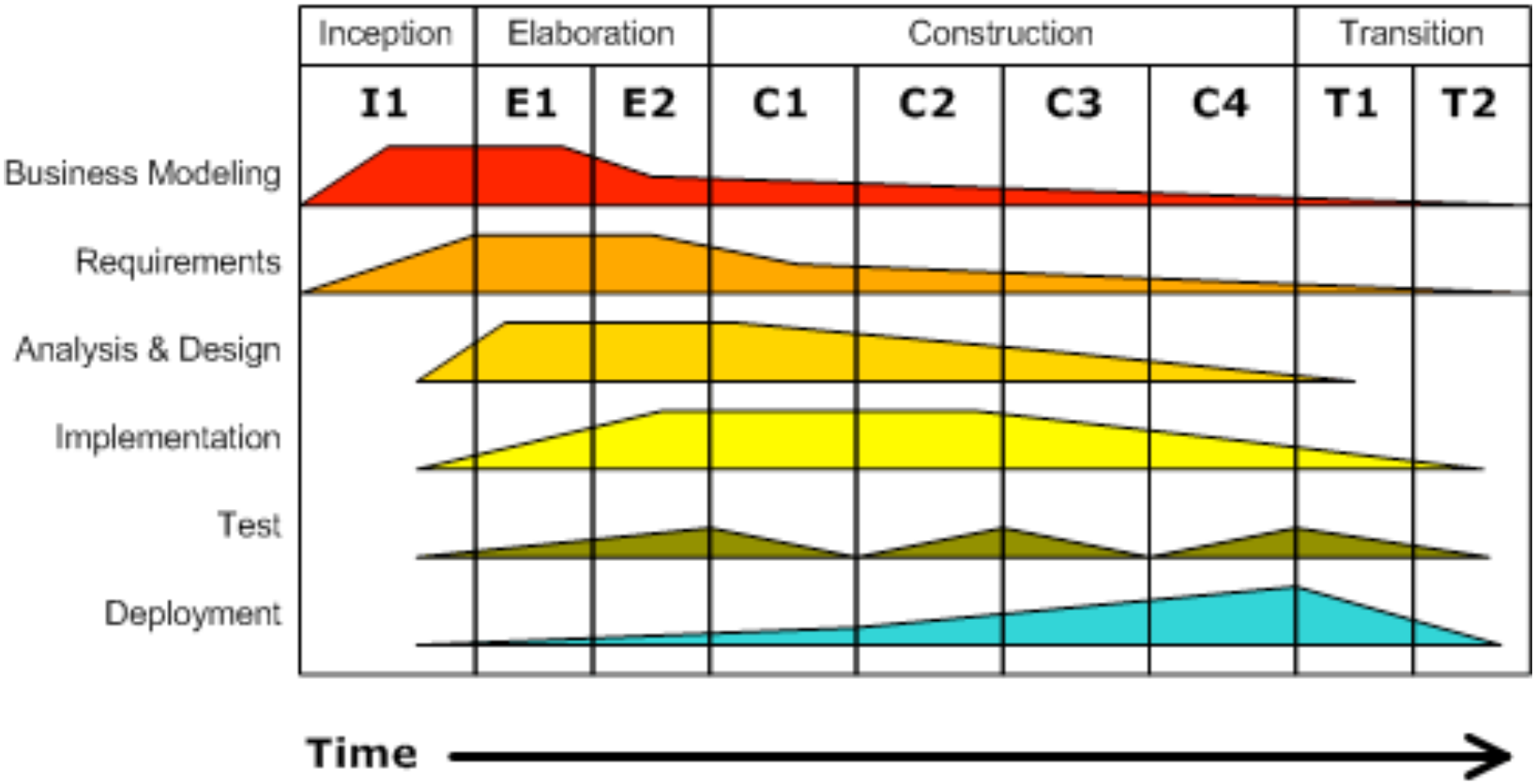
Prós

- Realizar iterações envolve o cliente no processo antes do produto ser terminado
  - reduz as chances de mal entendidos
- O gerenciamento de risco é parte do ciclo de vida
- Monitorar o projeto é mais fácil
- Prazos e custos ficam mais realísticos ao longo do tempo

Contras

- Iterações de 6 a 24 meses
  - dá tempo do cliente mudar de ideia
- **Muita** documentação por iteração
- Várias regras a serem seguidas, difícil seguir todas em todos os aspectos do projeto
- Custo do processo alto
- Torna difícil manter o orçamento e prazo dentro do combinado

Rational Unified Process (RUP, 2003)



FASES DO RUP

RPU possui 4 fases (que podem ser iteradas)

1. **Iniciação**: define o negócio do software, define cronograma e avaliação inicial de riscos
2. **Elaboração:** casos de uso, arquitetura do software, protótipo
3. **Construção:** codifica e testa o produto, 1ª versão
4. **Transição:** desloca o produto do desenvolvimento para o ambiente real de produção, inclui a aceitação do cliente

DISCIPLINAS DE ENGENHARIA DO RUP

Há 6 disciplinas (ou fluxos de trabalho) que as pessoas que trabalham no projeto devem cobrir?

1. Modelagem de negócios
2. Requisitos
3. Análise e Projeto
4. Implementação
5. Teste
6. Implantação (*deployment*)

RUP: prós & contras

Prós

- Práticas de negócio amarradas ao processo de desenvolvimento
- Várias ferramentas desenvolvidas pela Rational (agora IBM)
- Ferramentas ajudam na melhoria gradual do projeto

Contras

- As ferramentas custam caro (não são de código aberto)
- Muitas opções para ajustar RUP a cada empresa, impede o uso só das ferramentas
- Funciona bem apenas para projetos grandes ou de tamanho médio
- O tamanho de cada iteração é decisão do gerente

PLANEJE-E-DOCUMENTE: GERENTE DE PROJETO

P-e-D depende de **Gerentes de Projeto**:

- Escrevem contratos para ganhar os projetos
- Recrutam o time de desenvolvimento
- Avalia o desempenho dos desenvolvedores de software, o que define os salários
- Estima custos, mantém o cronograma, avalia riscos e os supera
- Documenta o plano de planejamento do projeto
- Recebe crédito pelo sucesso ou é o culpado se o projeto atrasa ou estoura o orçamento

P-E-D TAMANHO DO PROJETO

> *“Adicionar mais pessoas para ajudar em um projeto que está atrasado só faz o projeto atrasar mais.” — Fred Brooks Jr.*, The Mythical Man-Month

- Um novo desenvolvedor demora algum tempo para aprender sobre o projeto
- O tempo gasto com comunicação cresce com o tamanho, deixando menos tempo para o trabalho de desenvolvimento
- Grupos de 4 a 9 pessoas, mas organizadas de forma hierárquica para compor o projeto

PERGUNTA:

Qual afirmação abaixo é FALSA sobre o ciclo de vida dos processos em Cascata, Espiral e Rational Unified Process?

- Todos dependem de um planejamento meticuloso e medem o progresso do projeto de acordo com o plano inicial
- Todos dependem de uma documentação completa e minuciosa
- Todos dependem de um gerente que será responsável pelo projeto
- Todos usam iterações e protótipos

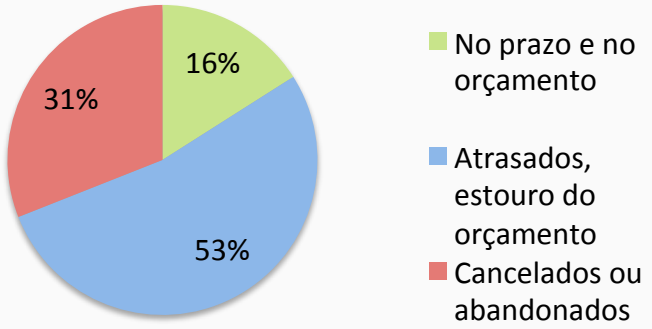
HÁ OUTROS PROCESSOS ALTERNATIVOS?

- Quão bem os processos Planeje-e-Documente atingem os objetivos de custo, cronograma e qualidade?
- P-e-D requerem extensa documentação e planejamento que dependem de um gerente com muita experiência
  - Podemos construir software de forma eficaz sem um planejamento cuidadoso e sem documentação?
  - Como evitar que todos “só saiam programando”?

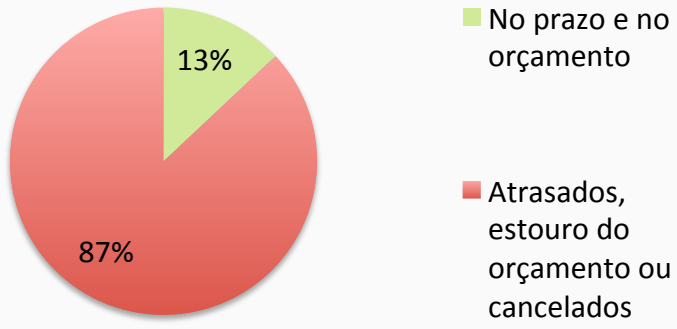
QUÃO BEM OS PROCESSOS PLANEJE-E-DOCUMENTE FUNCIONAM?

- IEEE Spectrum “Software Wall of Shame”
- 31 projetos (4 citados anteriormente + 27 outros) perderam US$ 17 bilhões

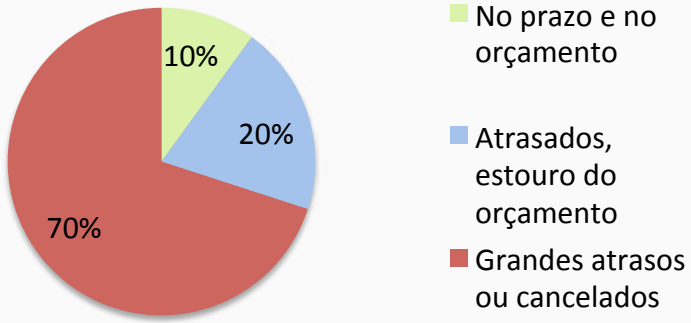
a) Projetos de Software (Johnson 1995)



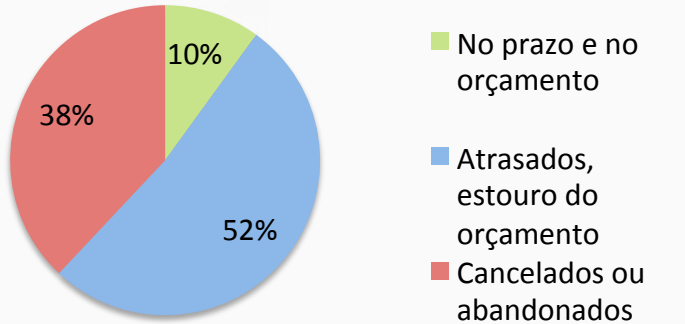
b) Projetos de Software (Taylor 2000)



c) Projetos de Software (Jones 2004)



d) Projetos de Software (Johnson 2013)



LEI DE PERES

“Se um problema não tem solução, ele pode não ser um problema, mas sim um fato – não a ser resolvido, mas sim para lidarmos com ele ao longo do tempo.”

— Shimon Peres (vencedor do Prêmio Nobel da Paz em 1994)