SIN222 – Fund. dos Sist. de Informação

Desenvolvimento de Sistemas

Rodrigo Smarzaro
smarzaro@ufv.br

Universidade Federal de Viçosa
Campus Rio Paranaíba

2025-1

UFV
Curso de Sistemas de Informação

Outline

Introdução

Participantes no desenvolvimento de sistemas

Equipe de Desenvolvimento

- Determinam os objetivos para os sistemas de informação
- Entregam o sistema que atende aos requisitos

Gerente de Projetos

- Responsável por coordenar todas as pessoas e recursos necessários para completar o projeto no tempo previsto
- Geralmente um Profissional de SI da empresa, mas pode ser um consultor externo

Projeto

Coleção planejada de atividades que cumprem uma determinada meta

Participantes no desenvolvimento de sistemas

Stakeholders ou os Tomadores de Decisão

Pessoas que terão benefício com o projeto

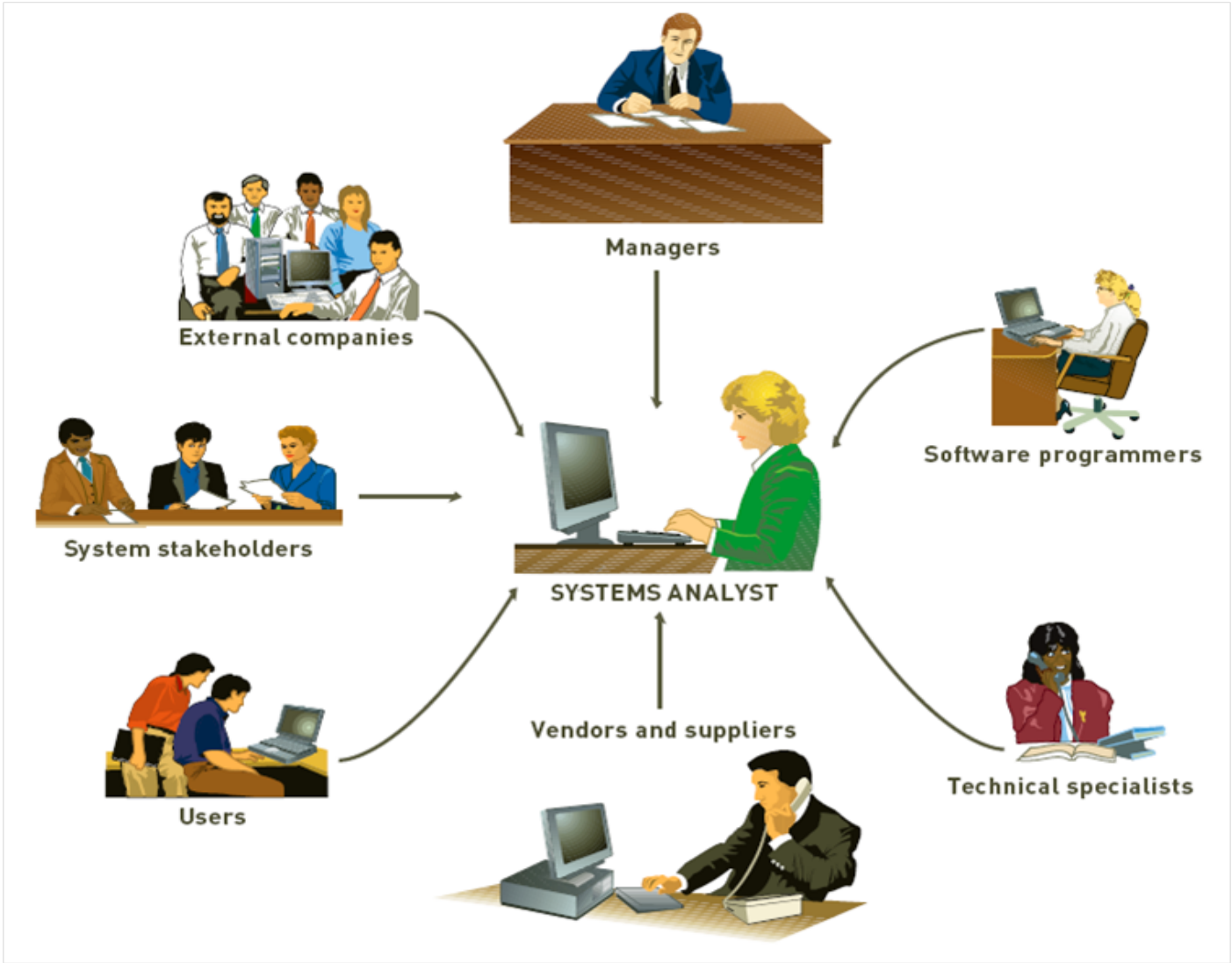
Usuários

Pessoas que irão interagir com o sistema regularmente

Especialistas no desenvolvimento dos sistemas

- Engenheiros de Software / Analistas de Sistemas
- Programadores

Equipe de Desenvolvimento



Iniciando o Desenvolvimento do Sistema

- Iniciativas de desenvolvimento de sistemas
  - Aparecem em todos os níveis de uma organização
  - Podem ser planejadas ou não planejadas
- Razões para iniciar projetos de desenvolvimento de sistemas
  - Fusão ou aquisição de novas empresas, mudanças na legislação, novas tecnologias (RFID, *Cloud Computing*, ...), entre outros
  - Falta de suporte para algum sistema ou software antigo

Planejamento dos sistemas de informação e alinhamento dos objetivos

Planejamento dos sistemas de informação

Tradução dos objetivos estratégicos e organizacionais em iniciativas de desenvolvimento de sistemas

Alinhamento dos objetivos organizacionais e de SI

- Crítico para o sucesso do desenvolvimento de sistemas
- Difícil de medir. Como saber se um novo sistema pode ajudar nos negócios?
- Métrica geralmente utilizada: Estimativa de `ROI` (*Return of Investment*) ou Retorno do Investimento.

Planejamento dos sistemas de informação e alinhamento dos objetivos

Desenvolvendo uma vantagem competitiva

Pensar competitivamente requer, geralmente, análise criativa e crítica

- Análise Criativa - Investigar novas abordagens para os problemas existentes.
- Análise Crítica - questionamento cuidadoso e isento sobre a melhor forma de como organizar os elementos envolvidos no sistema.
  - Nossos métodos atuais são eficientes e efetivos?

Objetivos para o desenvolvimento de sistemas

Objetivo geral do desenvolvimento de sistemas é alcançar as metas da empresa, não metas técnicas

Sistemas de missão crítica

- Fazem um papel principal nas operações da organização e no cumprimento de objetivos
- Sistemas que, no caso de falhas, colocam em sério risco o cumprimento das metas da empresa

Fatores Críticos de Sucesso (CSF - *Critical Success Factors*)

- Fatores que são essenciais ao sucesso de uma área funcional de uma organização
- Esses fatores podem ser traduzidos como objetivos específicos de um sistema

Objetivos para o desenvolvimento de sistemas (cont.)

Objetivos de Desempenho

- Qualidade e utilidade das saídas geradas
- Precisão da saída
- Velocidade em que a saída é produzida
- Escalabilidade do sistema resultante
- Risco do sistema

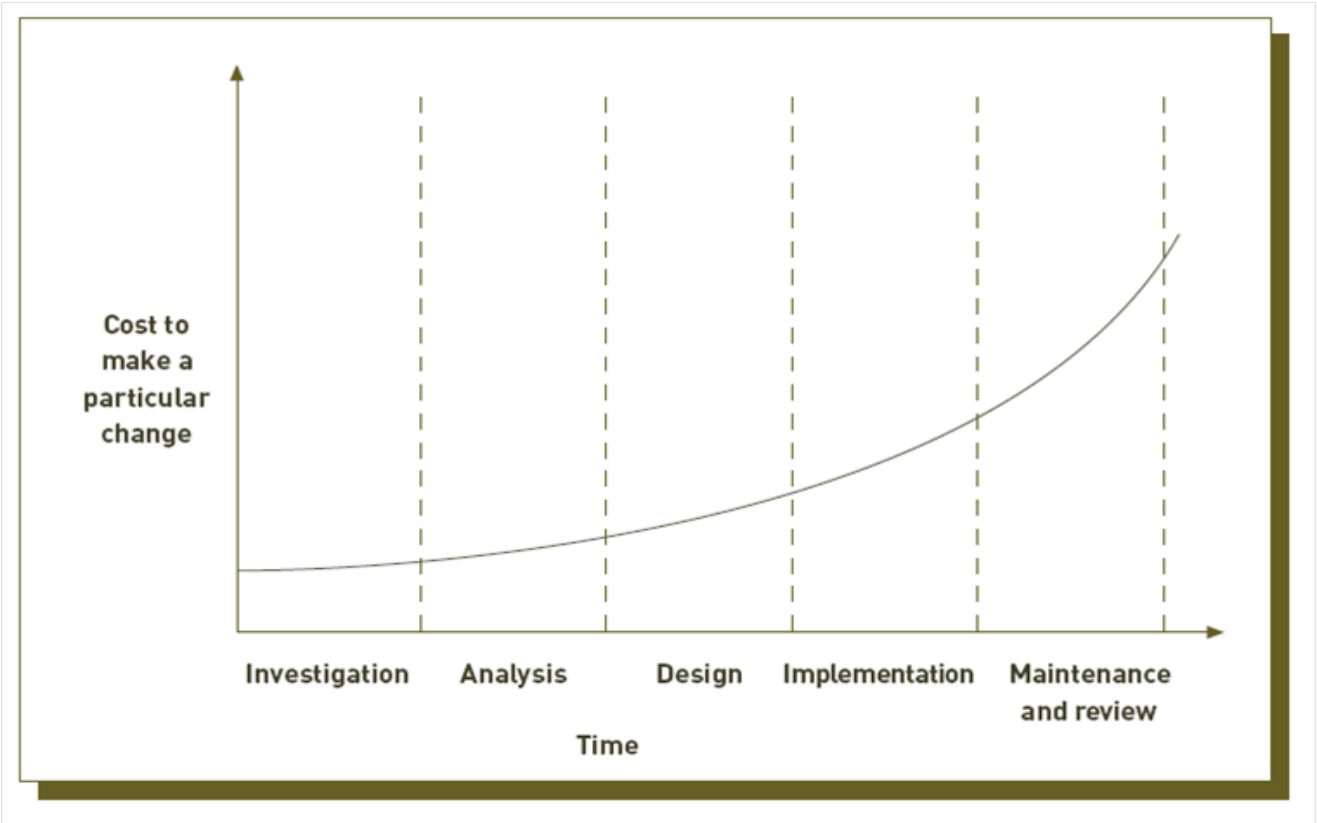
Objetivos de Custo

- Custo do desenvolvimento
- Custo de ter uma aplicação que tem um único uso
- Investimentos em hardware e equipamentos relacionados
- Outros custos operacionais (pessoal, suprimentos, etc. . .)

Ciclos de Vida do Desenvolvimento de Sistemas

Ciclos de Vida do desenvolvimento de sistemas

- Quanto mais tarde um erro for descoberto no ciclo de vida de desenvolvimento, mais caro é para corrigir
  - Fases anteriores devem ser retrabalhadas
  - Muitas pessoas são afetadas



Ciclos de Vida do Desenvolvimento de Sistemas

Ciclos de vidas mais comuns no desenvolvimento

- Tradicional
- Prototipação
- *Rapid Application Development* (RAD)
- Desenvolvimento do Usuário final
- *Open Source*
- Métodos Ágeis

Adotar um método ou *framework* serve para aumentar as chances de sucesso no desenvolvimento do sistema

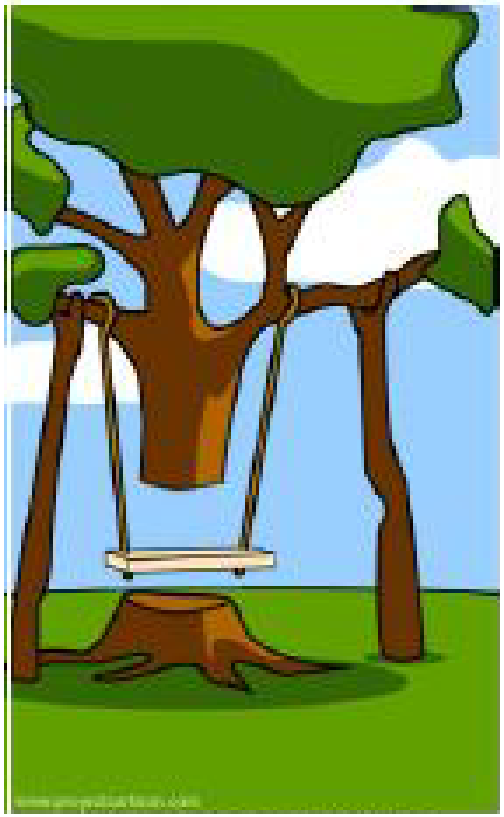
Engenharia de Software

Como o cliente explicou

Engenharia de Software

Como o líder de projeto entendeu

Engenharia de Software



Como o analista planejou

Engenharia de Software

Como o programador codificou

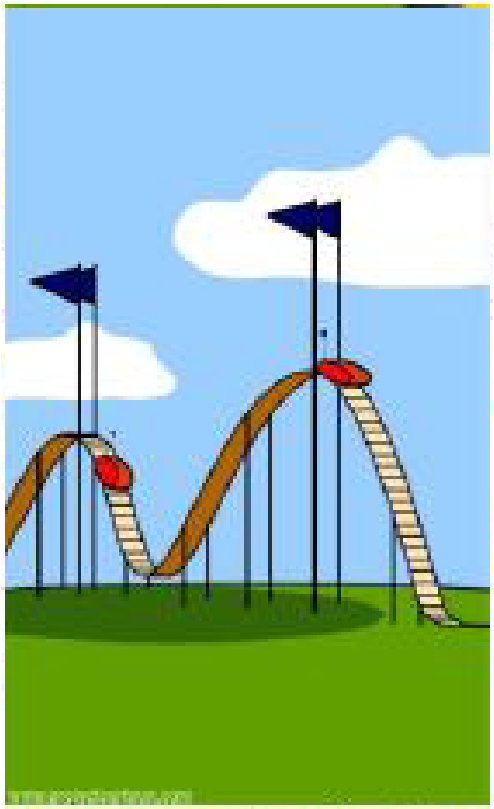
Engenharia de Software

O que os testadores receberam

Engenharia de Software

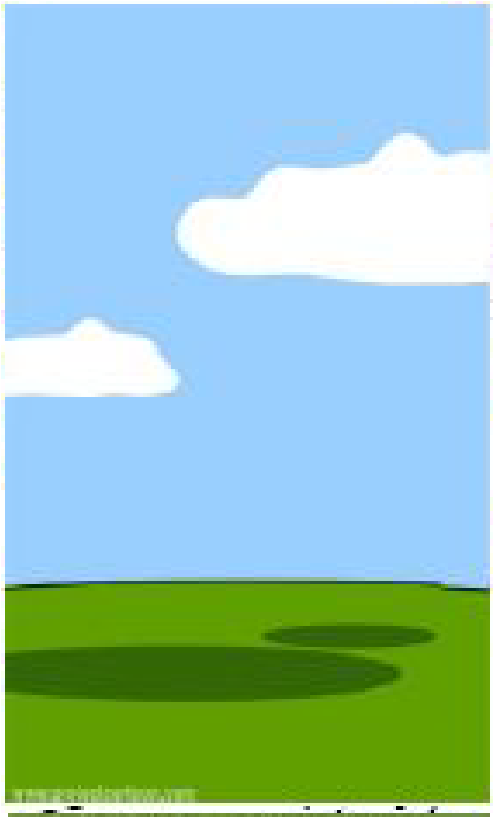
Como o pessoal das vendas descreveu

Engenharia de Software



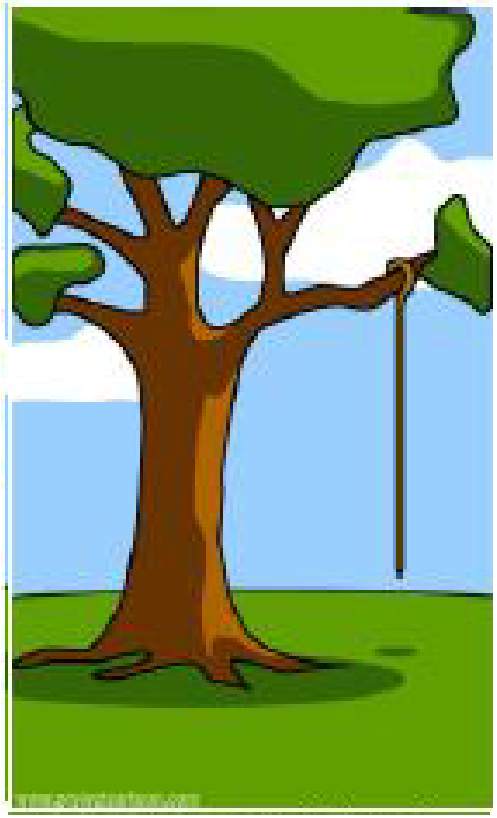
O que o cliente pagou

Engenharia de Software



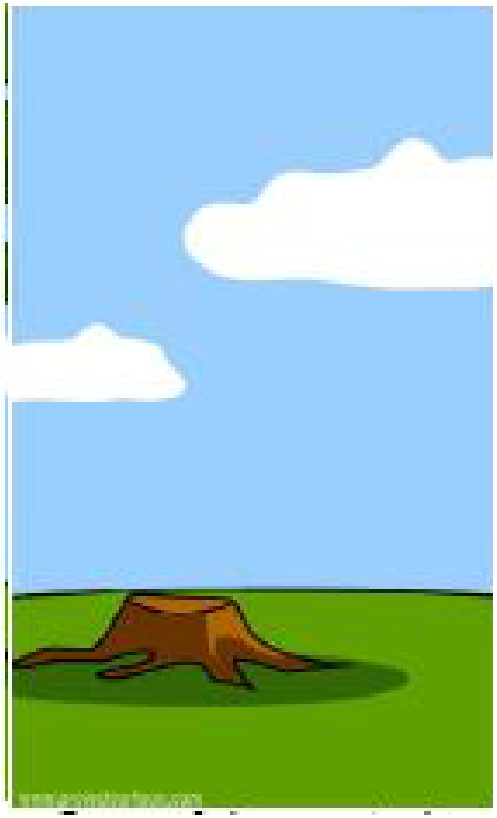
A documentação do projeto

Engenharia de Software



O que foi instalado no cliente

Engenharia de Software



Como foi feita a manutenção

Engenharia de Software

Quando o projeto foi entregue

Engenharia de Software

O que o cliente precisava

O ciclo de vida de desenvolvimento Tradicional

Investigação do sistema

- Identifica o problema e oportunidades em relação aos objetivos do negócio
- *Qual é o problema? Vale a pena (ou seja, $$) resolvê-lo?*

Análise do sistema

- Estuda sistemas e processos existentes para identificar pontos fortes, fracos e oportunidades de melhorias
- *O que o sistema de informação deve fazer para resolver o problema?*

O ciclo de vida de desenvolvimento tradicional (cont.)

Projeto do Sistema

- Detalha as saídas, entradas e interface com o usuário do sistema...
- Deve especificar: hardware, software, bancos de dados, telecomunicações, pessoas, componentes, procedimentos

Implantação do sistema

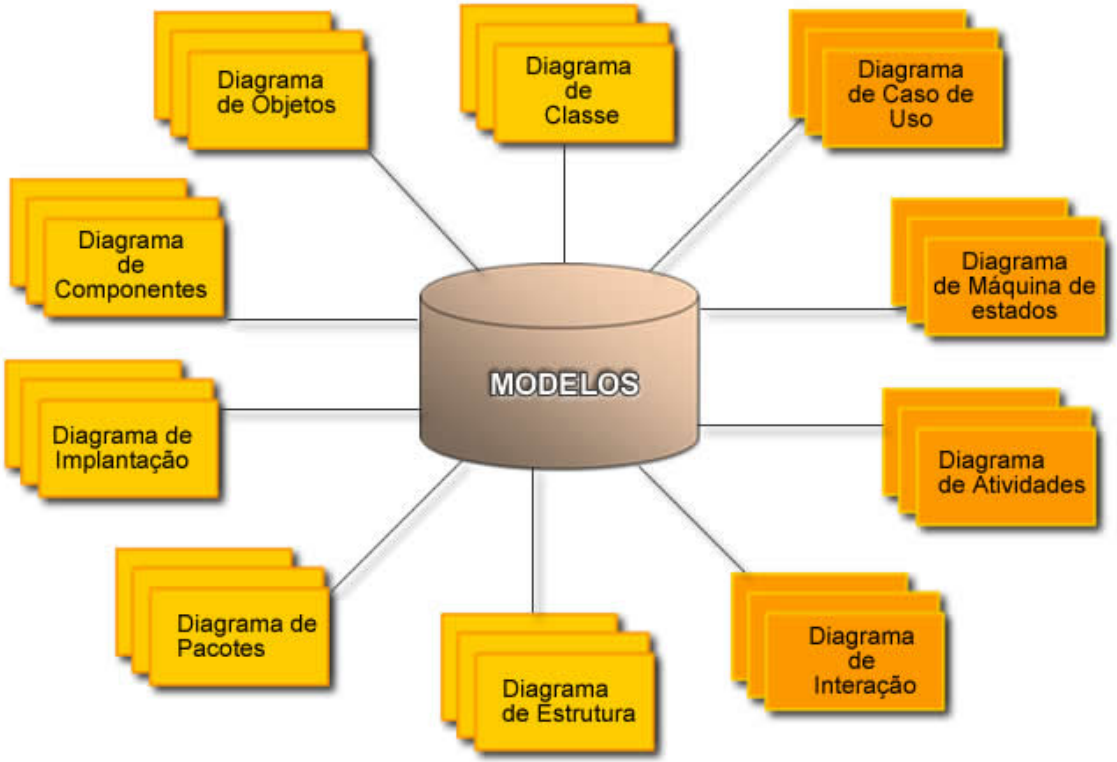
- Cria ou adquire os vários componentes detalhados no projeto do sistema, organiza-os e coloca o novo (ou modificado) sistema em funcionamento

Manutenção e revisão do sistema

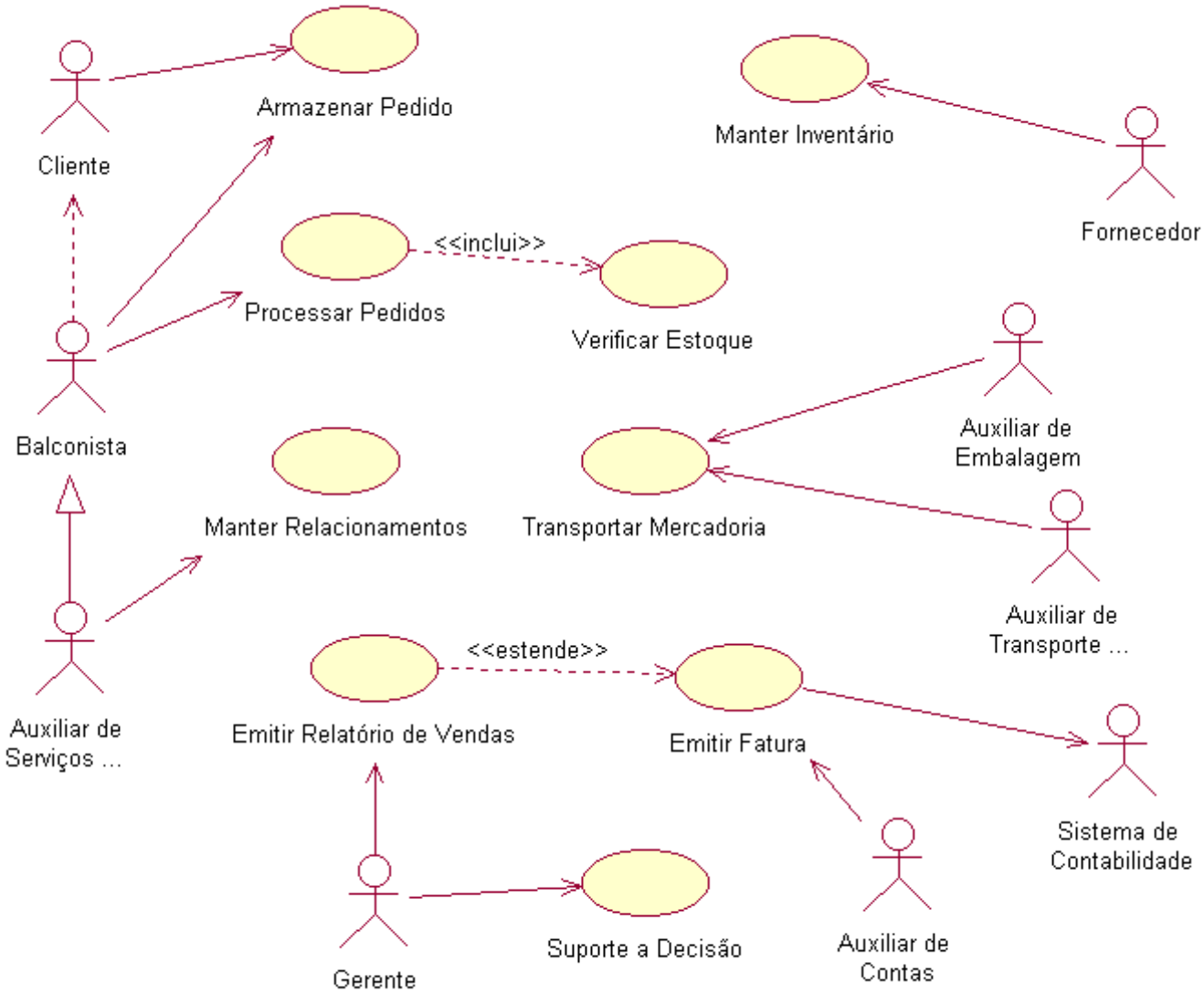
- Assegura que o sistema está funcionando como planejado
- Modifica o sistema para que continue atendendo as necessidades da empresa

O ciclo de vida de desenvolvimento tradicional (cont.)

- O ciclo de vida tradicional permite muito controle do processo de desenvolvimento
- Cliente demora muito para ver o sistema
- Produz muita documentação. Exemplo: Diagramas UML

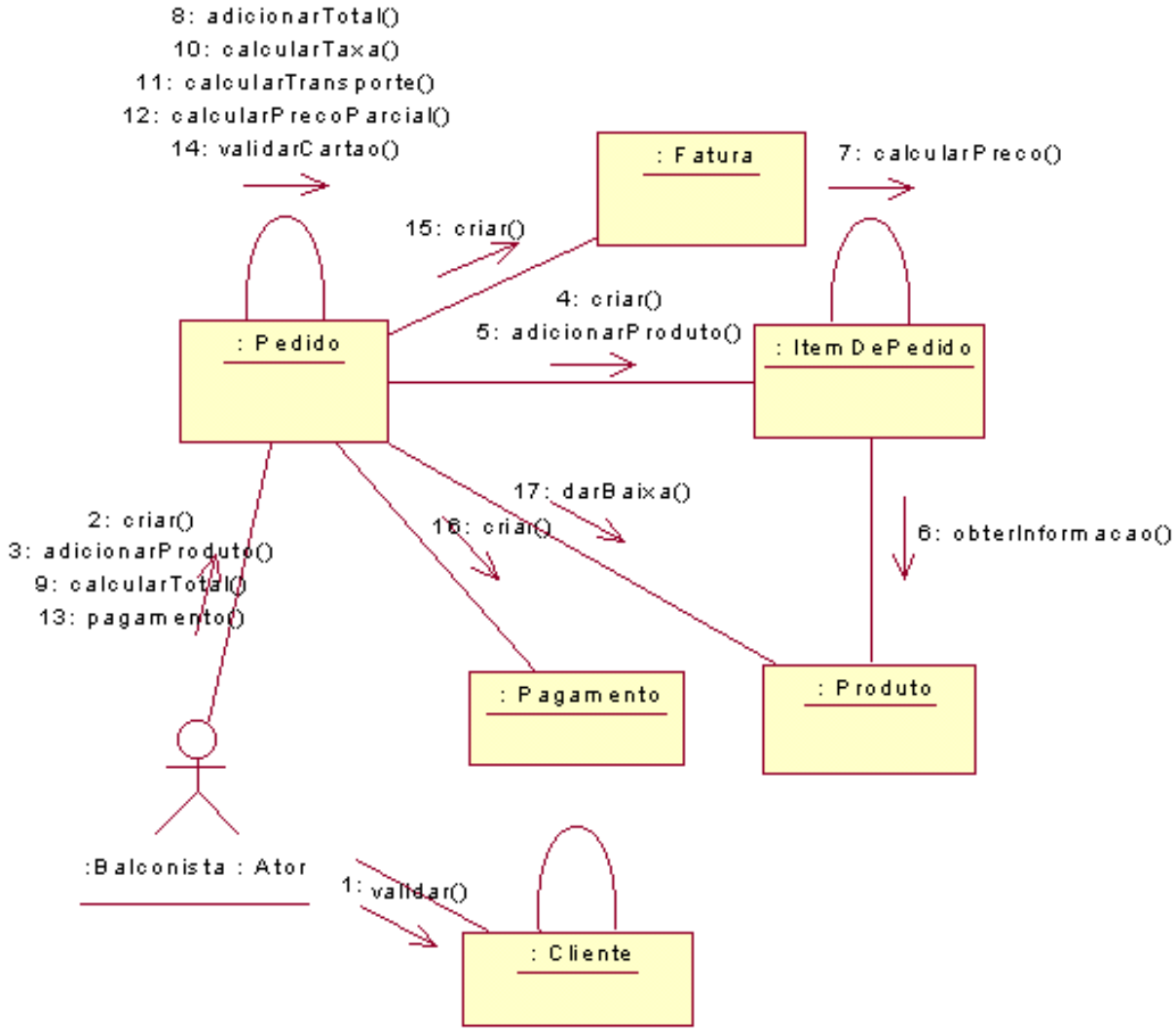


Exemplo UML: Diagrama de Casos de Uso



Exemplo UML: Diagrama de Colaboração

"Processar Pedido"



Prototipação

- Abordagem interativa
- Protótipo operacional
  - Protótipo que funcione
  - Acessa dados reais, edita dados de entrada, realiza as computações necessárias e mostra resultados reais
- Protótipo não-operacional
  - Um modelo que inclui especificações de entrada e saída
  - Pode até ser baseado em papel

Outras abordagens de desenvolvimento

Rapid Application Development (RAD)

- Utiliza ferramentas, técnicas e metodologias projetada para acelerar o desenvolvimento da aplicação
- Ferramentas geradoras de código a partir da modelagem
- Reuniões frequentes entre usuários, desenvolvedores e *stakeholders* para análise e proposta de soluções

Outras abordagens para o desenvolvimento

- Desenvolvimento Ágil (*Agile development*)
- *Extreme Programming* (XP)
- *Rational Unified Process* (RUP)

Desenvolvimento pelo usuário final

- Desenvolvimento no qual os gerentes de negócio e usuários assumem o esforço principal
- Utiliza ferramentas próprias que simplificam o trabalho de desenvolvimento (*plug and play*, *No Code*, *Low Code*)
- Exemplos[1]:
  - Webflow - Criação de sites
  - Bubble - Criação de aplicativos
  - Shopify - Criação de sites de *e-commerce*

[1] Uma lista de ferramentas No Code pode ser consultada em `https://userguiding.com/pt-br/blog/ferramentas-no-code`

Terceirização e computação sob-demanda

- Reduz os custos (Comparação de valor pago para desenvolvedores[2]

País	**Junior/Mid (USD/h)**	**Senior (USD/h)**
Estados Unidos	50 – 100	100 – 200
Brasil	5 – 16; 17 – 50 (*offshore*)	40 – 70
Índia	2.5 – 12	20 – 50
China	25 – 28	60 – 80

- Mais facilidade para obter tecnologia no estado da arte
- Pode diminuir problemas de pessoal e/ou de infraestrutura que a empresa possa ter
- Aumenta a flexibilidade tecnológica

[2] Médias entre 2024-2025

Fatores que Afetam o Sucesso do Desenvolvimento

Fatores que afetam o desenvolvimento de sistemas

Um desenvolvimento de sucesso. . .

. . . entrega um sistema que atende as necessidades do usuário e da organização no tempo e orçamento corretos

Crítico para o sucesso dos projetos de desenvolvimento

- Envolver os usuários e *stakeholders* no processo
- Ter apoio da gerência de alto nível

Fatores que afetam o desenvolvimento de sistemas

Possíveis resistências ao desenvolvimento de sistemas

- Medo de perder o emprego, poder ou influência na empresa
- Novo sistema vai gerar mais trabalho ao invés de poupar
- Relutância em trabalhar com os *nerds* do computador
- *Pessoal do computador* não sabe *"o modo como as coisas são feitas por aqui"*
- Resistência para aprender novos procedimentos

Qualidade e Padrões de Desenvolvimento

Qualidade do planejamento do projeto

- Quanto maior o projeto, maior as chances de que um planejamento ruim causará problemas significativos
- Ex. Copa do mundo?? Olimpíadas???

Capability Maturity Model (CMM)

- ou Modelo de Maturidade de Processos
- Uma maneira de "medir" a experiência organizacional no desenvolvimento de software
- Possui 5 níveis em ordem crescente de maturidade: inicial, repetível, definido, gerenciado, otimizado

CMM - Capability Maturity Model

Nível 1 - Inicial

- O processo de desenvolvimento é imprevisível e mal controlado
- O desenvolvimento depende muito das pessoas envolvidas
- Pouca ou nenhuma documentação formal é gerada
- Projetos geralmente atrasam e/ou ultrapassam o orçamento
- Ex. Em uma empresa Nível 1 cada equipe trabalha do seu jeito, sem padrões definidos

CMM - Capability Maturity Model

Nível 2 - Repetível

- O processo de desenvolvimento é baseado em práticas gerenciais básicas
- Basicamente são feitos controles de cronograma e orçamento.
- A empresa é capaz de repetir sucessos passados em projetos com características semelhantes
- Ex. Nesta empresa os projetos são entregues no prazo e dentro do orçamento com regularidade, mas ainda dependem muito da experiência dos gerentes.

CMM - Capability Maturity Model

Nível 3 - Definido

- O processo de desenvolvimento é padronizado e documentado
- A organização possui definido o processo comum de desenvolvimento. Todas os projetos utilizam o padrão adotado.
- Treinamentos frequentes são oferecidos ás equipes
- Ex. Nesta empresa haverá uma manual para desenvolvimento de software e as equipes devem seguí-lo a risca, com pouca flexibilidade

CMM - Capability Maturity Model

Nível 4 - Gerenciado

- Utiliza métodos quantitativos para medir e controlar a execução dos projetos
- Métricas bem definidas para identificar a situação dos projetos. Exemplos de métricas:
  - Número de defeitos por mil linhas de código (KLOC)
  - Esforço em horas por fase
  - Taxa de retrabalho (tempo corrigindo erros de etapas anteriores)
  - lead time
- Nesta empresa é possível estimar com precisão quanto tempo e recursos serão necessários para desenvolver uma nova funcionalidade

Algumas Métricas Utilizadas no Nível 4 do CMM

Categoria	**Exemplos de Métricas**	**Finalidade**
Qualidade do Software	- Defeitos por KLOC - Taxa de defeitos em produção - Tempo médio de correção - Severidade dos defeitos	Avaliar a qualidade do produto e reduzir falhas em produção.
Desempenho de Processo	- Tempo por fase (análise, testes, etc.) - Retrabalho por fase - *Lead time* / *Throughput*	Medir a eficiência de cada etapa do desenvolvimento.
Produtividade	- KLOC por desenvolvedor/mês - Pontos de função por *sprint*	Estimar esforço, planejar recursos e prever entregas.
Estabilidade do Processo	- Desvio padrão da produtividade - Variação de defeitos entre versões - Estimativa vs. tempo real	Avaliar previsibilidade e controle estatístico do processo.
Satisfação e Feedback	- Nível de satisfação dos usuários - Taxa de aceitação na entrega - Participação dos *stakeholders*	Alinhar o desenvolvimento aos objetivos e necessidades do negócio.

CMM - Capability Maturity Model

Nível 5 - Otimizado

- A empresa possui histórico de desenvolvimentos com sucesso e o utiliza para melhoria contínua
- Utiliza um ciclo contínuo de aprendizado e inovação
- Nesta empresa se investe bastante em P&D, são revisadas as práticas constantemente e incorporados novas abordagens (DevOps, métodos ágeis, IA, ...) que possam trazer vantagens a manter a excelência

Níveis do CMM (Capability Maturity Model)

Nível	Nome	Características principais
1	Inicial	Processos imprevisíveis e caóticos. Sucesso depende de indivíduos e não de práticas estabelecidas.
2	Repetível	Gerenciamento básico de projetos. Sucessos anteriores podem ser repetidos com base na experiência.
3	Definido	Processos padronizados e documentados. Treinamentos e políticas organizacionais são aplicados.
4	Gerenciado	Processos são medidos e controlados quantitativamente. Decisões são baseadas em dados.
5	Otimizado	Foco em melhoria contínua, inovação e aprendizado. Processos são ajustados com base em métricas e feedback.

Ferramentas de Gerência de Projetos

Ferramentas de gerência de projetos - Conceitos

- **Cronograma do projeto**
  - Descrição detalhada sobre o que deve ser feito
- ***Milestones*** (Pontos de controle)
  - Datas críticas para completar partes importantes do projeto
- ***Deadline*** (data limite)
  - Data em que o projeto inteiro deve estar completo e operacional
- **Caminho Crítico**
  - Atividades que, se atrasarem, podem atrasar todo o projeto

Ferramentas de gerência de projetos (cont.)

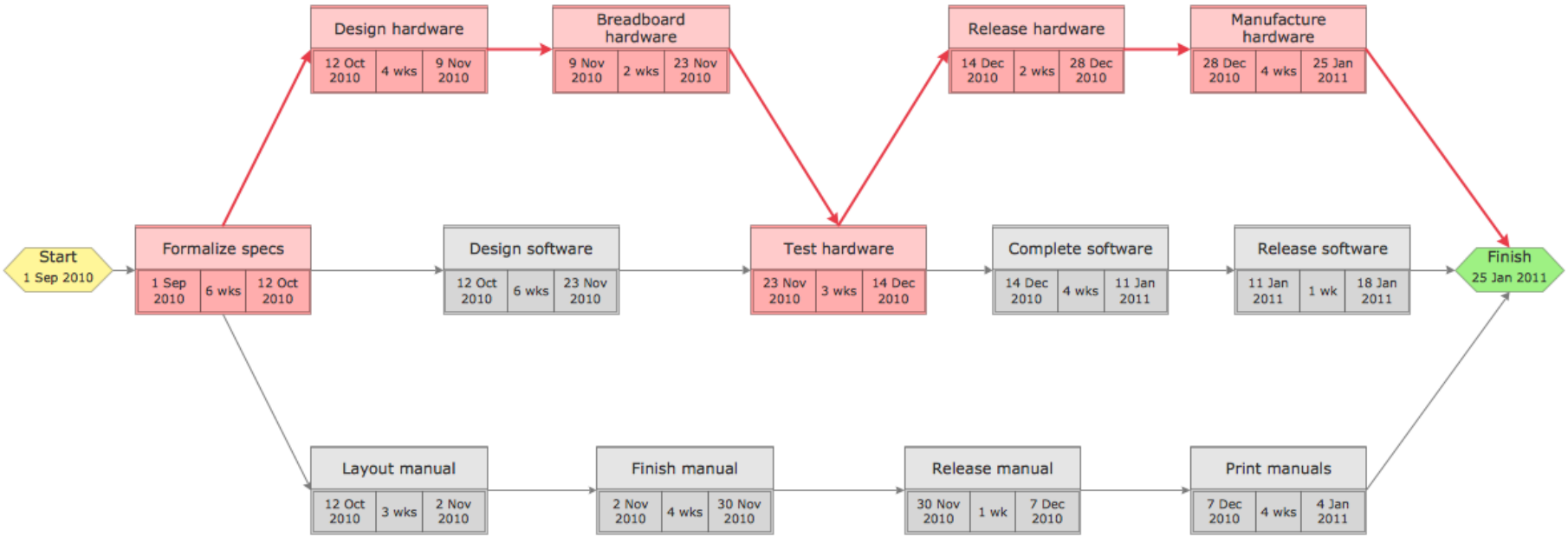
Program Evaluation and Review Technique (PERT)

- Cria 3 estimativas para cada atividade e usa uma média ponderada
  - Menor tempo possível (Otimista - O)
  - Tempo Médio (M)
  - Maior tempo possível (Pessimista - P)
  - Tempo estimado (TE): $(O + 4 \times M + P)/6$

Critical Path Method (CPM)

- Determinação do caminho crítico de uma sequência de atividades
- Se uma tarefa no caminho crítico sofre atraso, todo o projeto atrasa.
- Utiliza grafos para representar a dependência entre as atividades

Exemplo PERT/CPM



Exercitando CPM

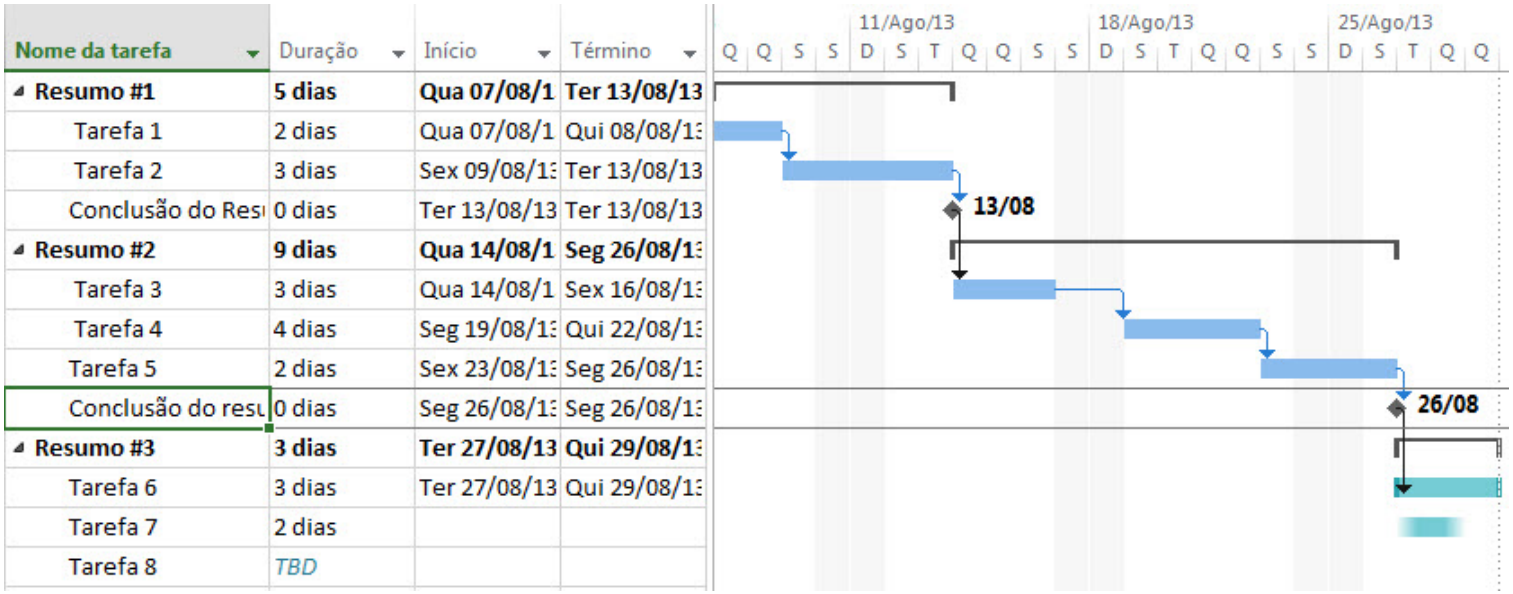
Atividade	Descrição	Precedente	Semanas
A	Escavação		2
B	Fundação	A	4
C	Paredes	B	10
D	Telhado	C	6
E	Encanamento Exterior	C	4
F	Encanamento Interior	E	5
G	Muros	D	7
H	Pintura Exterior	E,G	9
I	Instalação Elétrica	C	7
J	Divisórias	F, I	8
K	Piso	J	4
L	Pintura Interior	J	5
M	Acabamento Exterior	H	2
N	Acabamento Interior	K, L	6

a) Qual a duração do projeto?

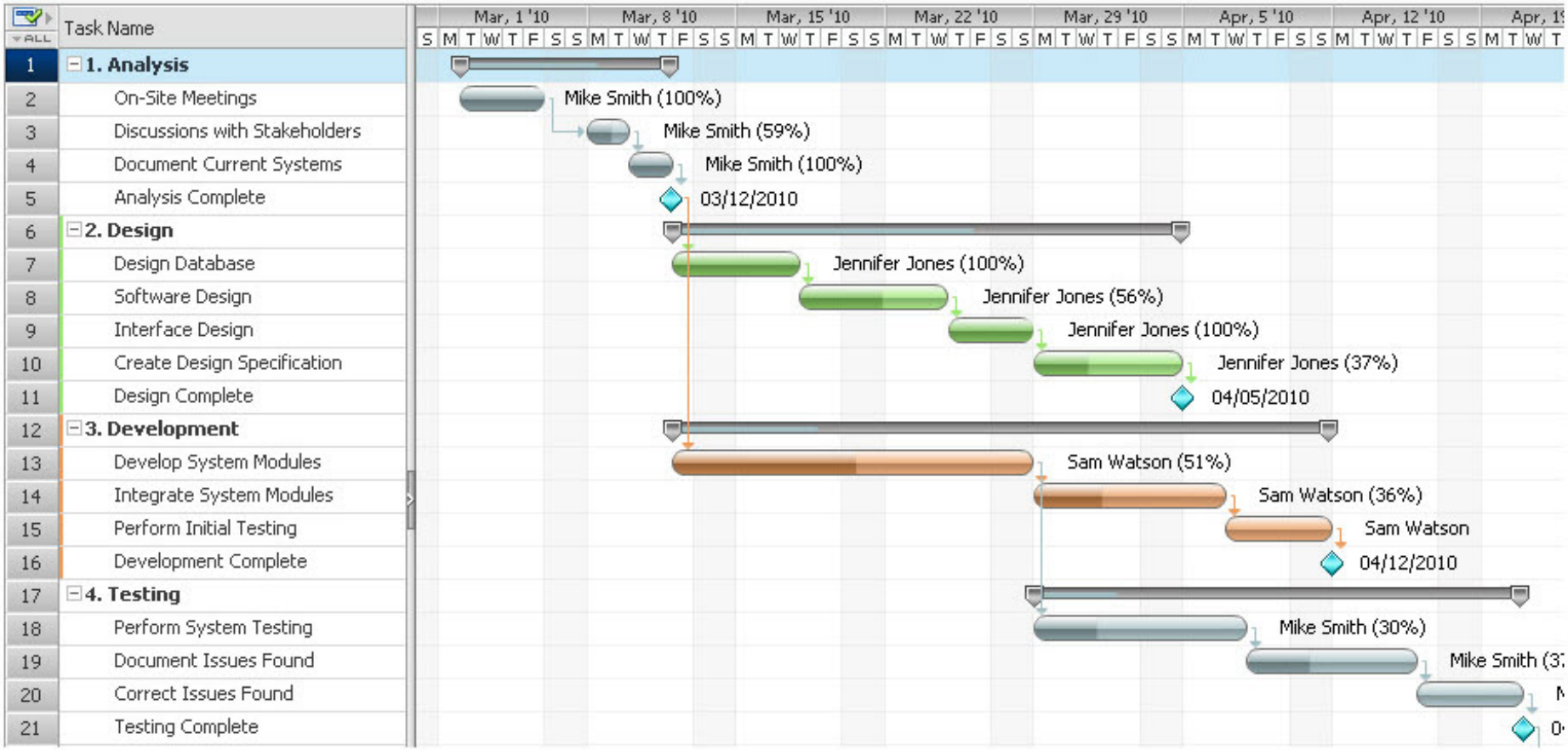
b) Que atividades não podem sofrer atrasos para não impactar o cronograma?

Exemplo Diagrama de Gantt

- Desenvolvido em 1917 por Henry Gantt
- Ilustra o avanço das várias etapas de um projeto
- O tempo necessário de cada tarefa é representado pela extensão horizontal em barras

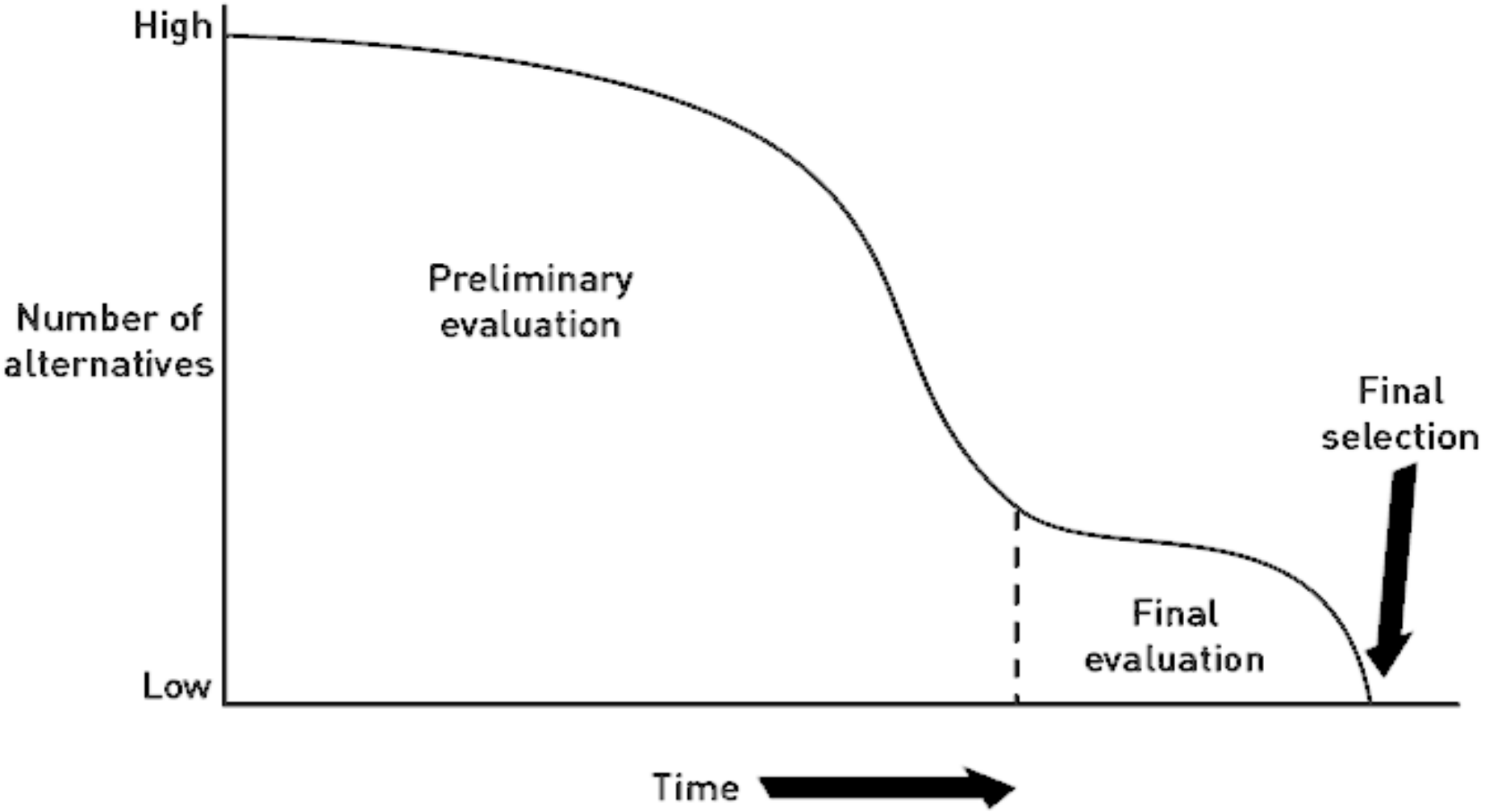


Exemplo Diagrama de Gantt



Análise de Alternativas de Software

Análise das Alternativas de Software



Técnicas de Avaliação de alternativas de software

- **Consenso do Grupo**
  - Um grupo é designado para tomar a decisão
- **Análise Custo/Benefício (ROI)**
- **Testes de *Benchmark***
  - Testes que comparam o desempenho de sistemas sob as mesmas condições
- **Avaliação por pontos**

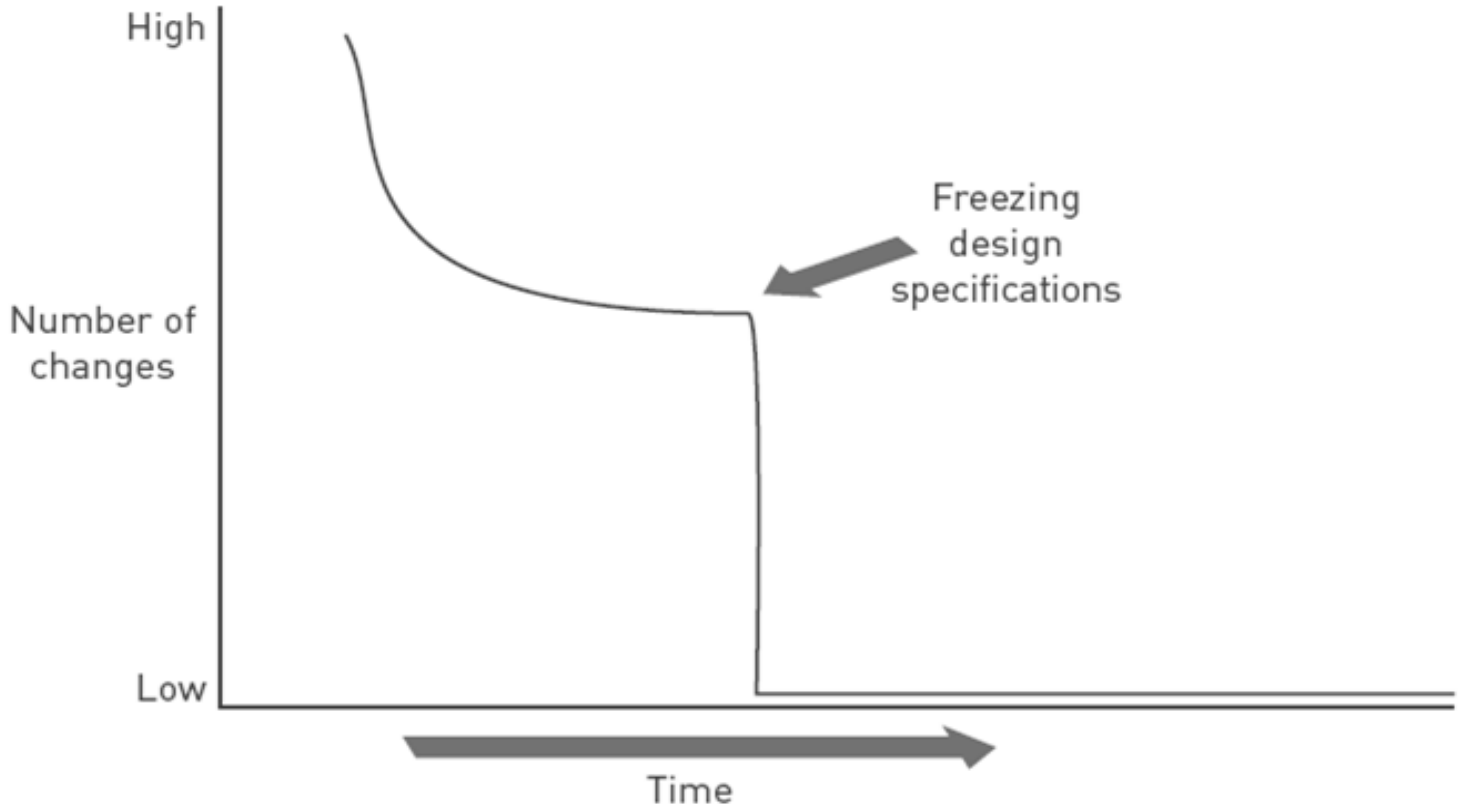
Avaliação por Pontos

- Avaliação por Pontos (cont.)
  - Em análises de custo/benefício é difícil monetizar os itens considerados
  - Pode-se optar por uma avaliação por "pontos" para uma comparação entre opções de software

		System A			System B		
Factor's importance		*Evaluation*		*Weighted evaluation*	*Evaluation*		*Weighted evaluation*
Hardware	35%	95	35%	33.25	75	35%	26.25
Software	40%	70	40%	28.00	95	40%	38.00
Vendor support	25%	85	25%	21.25	90	25%	22.50
Totals	100%			82.5			86.75

Congelando as especificações do projeto

- Usuários concordam (por escrito) que o projeto está aceitável
- Algumas organizações encorajam mudanças no projeto
  - Geralmente utilizam métodos ágeis de desenvolvimento



O Contrato

- Empresas fornecem contratos padrões que protegem elas mesmas
- Geralmente o contrato inclui o documento de **solicitação de proposta (RFP – *Request for Proposal*)**
- **RFP** - Documento detalhando tudo o que é esperado do fornecedor
  - Escopo do projeto;
  - Escopo do produto;
  - cronograma;
  - Condições para o projeto acontecer e restrições (tempo, custo, qualidade, . . .);
  - Responsabilidades;
  - Governança do projeto;
  - modelo do orçamento;
  - formas de pagamento; e
  - Termos e condições (confidencialidade, publicidade, etc)

Aquisição do software: fazer ou comprar?

Fazer

- Custo alto
- Software personalizado
- Qualidade pode variar dependendo da equipe de programação
- Pode levar anos para o desenvolvimento

Comprar

- Custo "baixo" (em relação ao desenvolvimento, geralmente)
- Pode não se adequar completamente as necessidades da empresa
- Alta qualidade (geralmente)
- Pode ser adquirido rapidamente
- Outras organizações podem obter a mesma "vantagem"

Preparação dos usuários

Preparação dos usuários

- Fase importante, mas muitas vezes ignorada na implantação de um novo software
- Preparar os gerentes, tomadores de decisão, empregados e outros usuário para o novo sistema
- Principal motivo em fazer as empresas atrasarem bastante a adoção de um novo sistema operacional.

Contratando e treinando pessoal de TI

- Sucesso de qualquer sistema depende de como é utilizado pelo departamento de TI de uma empresa
- Programas de treinamento específico para o pessoal de TI
  - Treinamento muito mais detalhado que o treinamento de usuários

Implantação de um novo sistema

- Processo de tornar um sistema totalmente operacional
- Possíveis Abordagens
  - Conversão direta
  - Por etapas
  - Projeto Piloto
  - Inicialização paralela

Conversão direta

O novo sistema substitui de uma só vez o sistema antigo

Implantação de um novo sistema

Por etapas

Apenas partes de uma nova aplicação ou somente alguns departamentos, agências ou fábricas são convertidos de cada vez. Uma conversão por etapas permite a ocorrência de um processo de implantação gradual dentro de uma organização

Projeto Piloto

Um departamento ou outro estabelecimento de trabalho funciona como local de teste. O novo sistema pode ser experimentado neste local até que os criadores do sistema sintam que ele pode ser implantado em toda a organização

Inicialização Paralela

Tanto o velho quanto o novo sistema estarão operando ao mesmo tempo até que a equipe de desenvolvimento do projeto e a administração do usuário final concordem em passar inteiramente para o novo sistema.

Aceitação do usuário

- Aceitação formal assinada pelo usuário que a implantação do sistema está completa
- Documento legal que remove ou reduz possíveis problemas para o fornecedor do sistema.

Manutenção do Sistema

- Mudanças nos processos de negócios
- Novos pedidos de recursos
- Bugs
- Problemas técnicos/hardware
- Aquisição ou fusão da empresa
- Mudanças na legislação

Tipos de manutenção

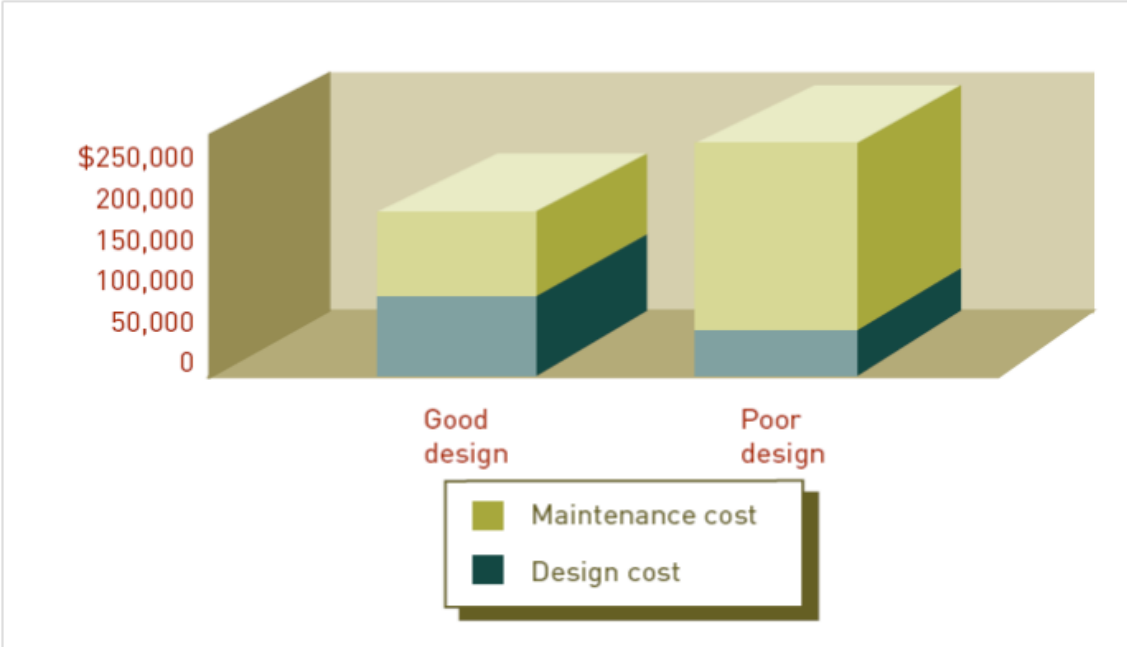
- **Atualização**
  - Geralmente envolve recompilar todo o código, para alguma correção muito simples
- ***Patch*** (correção)
  - Mudança menor para corrigir algum problema.
  - Não necessariamente exige recompilar todo código
- ***Release***
  - Conjunto de mudanças significativas
- **Versão**
  - Identificação numérica do software
  - É comum a utilização do padrão major, minor, patch

Exemplos de padrões de versões

Formato	Exemplo	Características
`MAJOR.MINOR.PATCH`	`2.3.1`	Versionamento Semântico
`YYYY.MM` ou `YYMM`	`2024.06`, `1809`	Baseado na data de lançamento
`X.Y.Z-labelN`	`1.0.0-beta.2`	Indica estágio: alpha, beta, rc
Número contínuo	`Build 453`	Contador incremental de builds
`X.Y.Z+build`	`2.0.1+build.42`	Build metadata
Produto.X.Y.Z	`Office.16.0.10396`	Versionamento modular ou por produto

Relação entre manutenção e projeto

- Programas são caros para desenvolver, mas podem ser ainda mais caros para manter
- Uma decisão a tomar é quando vale a pena investir em um novo software e abandonar o antigo.



Revisão do sistema

- Etapa final do desenvolvimento
- Analisa o sistema para verificar que o mesmo está operando conforme previsto
- **Revisão induzida por evento**
  - Um problema ou oportunidade dispara a necessidade de revisão do software
- **Revisão induzida pelo tempo**
  - Avaliações periódicas do sistema em busca de problemas e/ou oportunidades.
- Caso sejam detectados problemas, um novo ciclo de desenvolvimento pode ser iniciado.

Medindo o desempenho do sistema

- Monitorando o sistema
  - Número de erros encontrados
  - Quantidade de memória requerida
  - Quantidade de tempo de processamento
  - Etc...

- Geralmente feito com apoio de algum software específico
  - Software que monitora todos os componentes de um sistema de informação (*Profiler*)

Exemplos de *profilers*

Linguagem/Plataforma	Profiler	Descrição
Java	- VisualVM	- Interface gráfica, analisa uso de CPU, memória, threads e garbage collection.
	- JProfiler	- Ferramenta profissional com análise detalhada e suporte a profiling remoto.
	- YourKit	- Suporte a profiling de CPU, memória e integração com IDEs.
Python	- cProfile	- Embutido na linguagem, fornece estatísticas de chamadas de função.
	- Py-Spy	- Profiler leve e externo, sem necessidade de alterar o código.
	- line_profiler	- Mede o tempo de execução linha a linha de funções selecionadas.
C/C++	- gprof	- Profiler tradicional da GNU com suporte a análise de funções.
	- Valgrind (Callgrind)	- Instrumentação detalhada de CPU, com suporte a visualização via KCachegrind.
	- perf	- Ferramenta poderosa de análise de desempenho no Linux (amostragem).
.NET / C#	- dotTrace	- Da JetBrains, análise de CPU e memória com boa interface.
	- Visual Studio Profiler	- Integrado ao Visual Studio, suporta diversos tipos de análise.
JavaScript (Web)	- Chrome DevTools	- Ferramentas de performance no navegador Chrome.
	- Firefox Profiler	- Visualização detalhada de chamadas, layout, paint, etc.
Android	- Android Profiler	- Integrado ao Android Studio. Monitora CPU, memória, rede e energia.
iOS/macOS	- Instruments (Xcode)	- Análise profunda de desempenho, memória, energia e mais.
Outros / Multiplataforma	- perfetto	- Ferramenta open-source do Google para tracing de sistemas Android e Linux.
	- DTrace	- Instrumentação dinâmica de sistema, usada em macOS e BSD.
	- Intel VTune	- Análise avançada de desempenho e uso de hardware em C/C++ e Fortran.

"Morte" do sistema"

- Em algum momento o sistema se torna obsoleto ou inviável economicamente
- Medidas aa hora de "matar" um sistema:
  - Comunicado de intenção. Um aviso a todos os envolvidos com o software de que ele será descontinuado.
    Ex. A Microsoft emitiu comunicado em 2007 de que deixaria de dar suporte ao Windows XP em 2014.
  - Término de contratos. Todos os fornecedores de hardware, software ou serviços envolvidos com o sistema devem ser notificados para evitar multas;
  - Fazer *backups*. Antes de desativar o sistema e desinstalá-lo deve-se fazer cópias de backup de acordo com a política de dados da empresa
  - Apagar dados sensíveis. Certificar que todos os dados armazenados no hardware que podem comprometer a empresa ou clientes sejam devidamente apagados
  - Descarte do Hardware. Após fazer os backups e limpeza o hardware pode ser devidamente descartado

Apagando o HD

Por que o descarte seguro é importante?

- Dados deletados podem ser recuperados se não forem devidamente apagados.
- Formatação simples **não remove os dados** do disco.
- Vazamentos de informações sensíveis podem gerar sanções legais (ex: LGPD).

1. Eliminação segura de dados

HDs tradicionais (magnéticos):

- Sobrescrever com dados aleatórios:
- Ferramentas: **DBAN**, **Blancco**

SSDs:

- Utilizar **Secure Erase** via *firmware*/fabricante

2. Destruição física

HDs (magnéticos):

- Perfuração com furadeira (em vários pontos)
- Quebra dos pratos internos
- Desmagnetização (*degaussing*)

SSDs:

- Trituração dos chips NAND
- Perfuração nem sempre é suficiente

Para praticar...

Considere as atividades da tabela. Monte um diagrama PERT e indique qual a **duração esperada para todo o projeto**, o **caminho crítico**, e qual a **tarefa que possui a maior folga**.

Índice	Descrição da Atividade	Precedente Necessário	Duração (semanas)
A	Concepção / Design do Produto	(nenhum)	5
B	Pesquisa de Mercado	(nenhum)	1
C	Análise de Produção	A	2
D	Modelagem do Produto	A	3
E	Folheto de Vendas	A	2
F	Estimativa de Custos	C	3
G	Testes	D	4
H	Treinamento de Vendas	B, E	2
I	Precificação	H	1
J	Revisão do Produto	F, G, I	1

Referências

LAUDON, Kenneth C.; LAUDON, Jane P. **Essentials of Management Information Systems**. 12. ed.: Pearson, 2016.

STAIR, Ralph; REYNOLDS, George. **Fundamentals of Information Systems**. 8. ed.: Course Technology, 2015.

STAIR, Ralph; REYNOLDS, George. **Principles of Information Systems**. 13. ed.: Cengage Learning, 1 jan. 2017. 752 p.

STAIR, Ralph M.; REYNOLDS, George W. **Princípios de Sistemas de Informação**. Tradução da 9 edição americana. Cengage, 2013.