Celulares e Gadgets

Pós-Graduação em Engenharia de Software

Gerência de Configuração, Controle de Versão e Mudanças

Prof. Jucimar Jr

2013-1

Mini Currículo

- Analista de Sistemas e Gerente de Projetos de Desenvolvimento de Software- FUCAPI (1995 – 2005)
- Subsecretário Municipal de Ciência e Tecnologia da Informação da Prefeitura de Manaus (2005 – 2007).
- Professor de graduação e pós-graduação da UEA
- Graduado em Ciência da Computação pela UFAM
- Mestre em Engenharia Elétrica pela UFPE
- Doutor em Engenharia Elétrica pela UFPE (final em jun/2013)
- Professor Homenageado pela Turma de Computação de 2009,2010,2011 e 2012
- Áreas de Interesse:
  - Desenvolvimento de linguagens de Programação e Compiladores
  - Desenvolvimento de Jogos
  - Engenharia de Software
  - Software Livre e Inclusão Digital

TOP3 Prêmio iBest 2006
Portal de Manaus
iBEST
IBest Regional - Amazonas
Votação Popular

Contato

- Email: jucimar.jr@gmail.com
- Gtalk: jucimar.jr@gmail.com
- Facebook: www.facebook.com/jucimar.jr

Quem é você?

- Nome completo
- O que espera da pós?
- O que espera do curso?
- Onde trabalha?
- Em que você trabalha?
- A quanto tempo desenvolve software?
- Em que linguagens você programa?
- Como é o processo de desenvolvimento onde você trabalha?
- Como é o controle de configuração, versão, mudanças onde você trabalha?
- Trabalha com algum software de controle de versão? Qual? Como?

Sumário

- Objetivos
- Regras do Jogo
- Ementa
- Aulas

Objetivos

Objetivo Geral

- Aprender conceitos de Gerencia de Configuração, Controle de Versão e Mudanças (CGS)
- Aprender a importância da GCS
- Aprender a criar um processo de GCS
- Aprender o básico do GIT e o GITHUB

Não

- Não é um curso de GIT
- Não é um curso de graduação de GCS
- Não é um curso sobre um processo específico

Regras do Jogo

Regras gerais

- Horário:
  - Quinta 18:30 -> 22:00
  - Sexta 18:30 -> 22:00
  - Sábado 8:30 -> 17:00
  - Sala-Laboratório A10

Ementa

- **Gerência de Configuração:** controle de versões, gerenciamento de espaços de trabalho, gerenciamento de build, atributos, branches e históricos de elementos versionados.
- **Controle de Mudanças:** métodos, técnicas e ferramentas para registrar e gerenciar solicitações de mudanças suportando o gerenciamento do projeto. Ferramentas para gerência e configuração.

- **BONUS: Ambiente**: definição de ambiente de desenvolvimento, teste e produção

Ementa

- O que é CGS?
- Revisão de ES
- Prática 1
- Versionamento, mudanças e ambiente
- Prática 2
- Introdução ao GIT
- Prática 3
- Versionamento x Branches e Merges
- Prática 4
- Prática 5

Notas

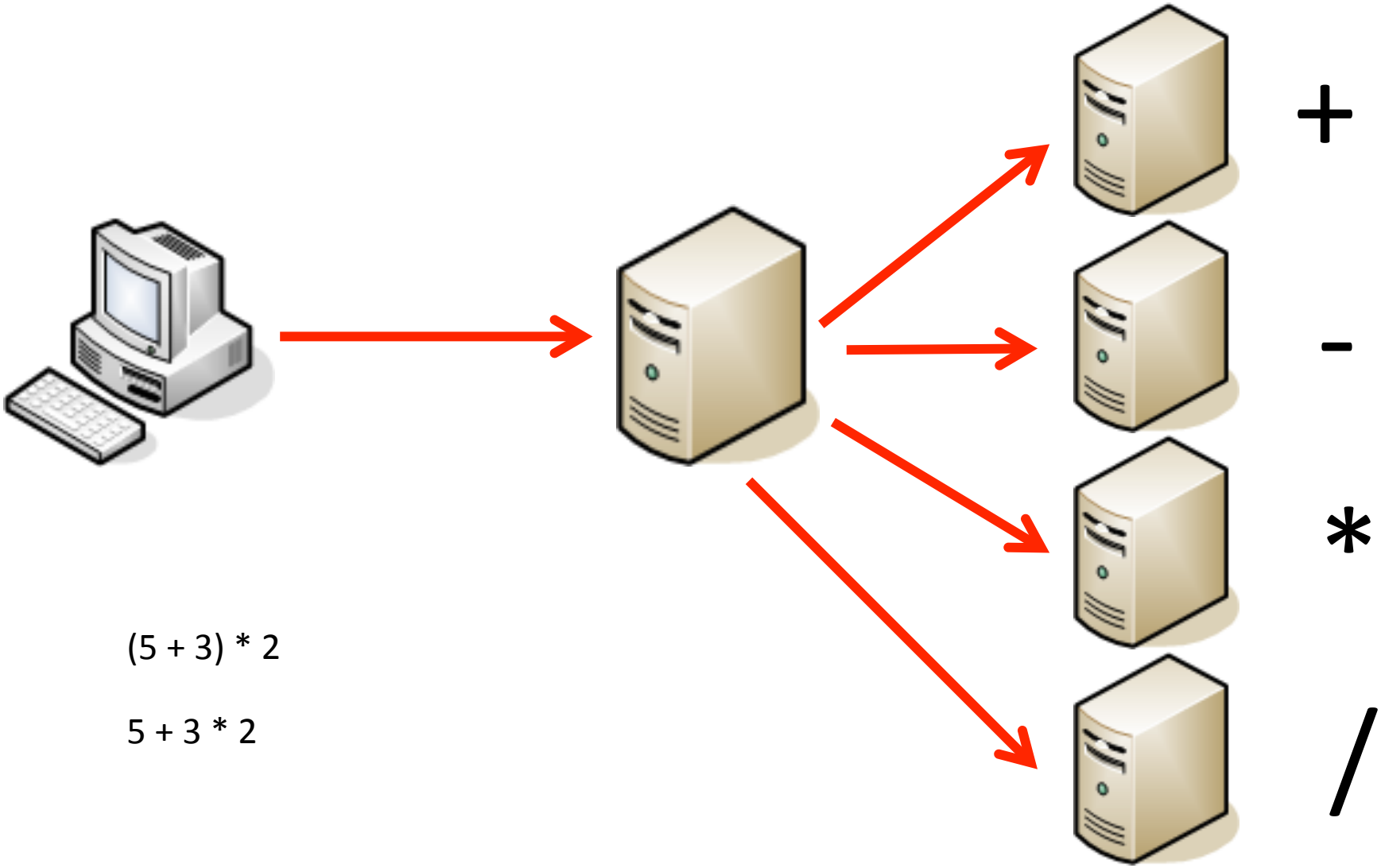
- 2 notas
  - Exercícios práticos e defesas
  - Projeto Final - 1 ou 2 semanas para o desenvolvimento
- Competição
  - Haverá sempre apenas um melhor trabalho. Todos os outros serão comparados a esse melhor trabalho
- Grupo de no máximo 5 pessoas

Projeto Final

- Ambiente de desenvolvimento
- Ambiente de teste
- Ambiente de produção
- Planejamento do versionamento do projeto
- Programa pronto
- Artefatos de GCS gerados durante o desenvolvimento

- **BONUS**: automação do processo

Projeto Final
Calculadora Distribuída



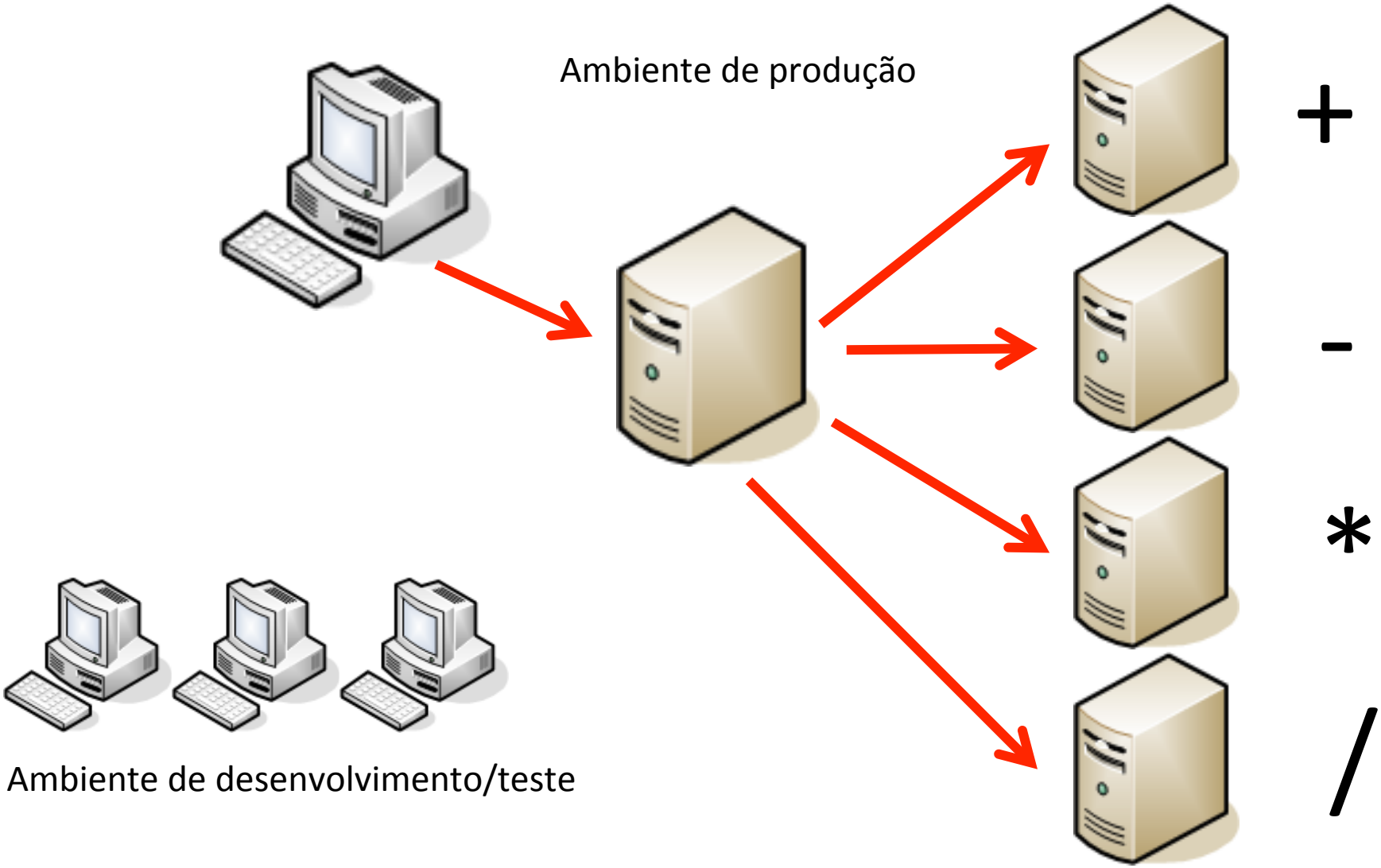
(5 + 3) * 2

5 + 3 * 2

Projeto Final
Calculadora Distribuída

http://www.ruby-lang.org/pt/

Calculadora Distribuída



Projeto Final Alternativo

- Descrever o processo de GCS da sua empresa
  - Ambiente de desenvolvimento
  - Ambiente de teste
  - Ambiente de produção
  - Planejamento do versionamento do projeto
  - Programa pronto
  - Artefatos de GCS gerados durante o desenvolvimento
- Descrever o processo de GCS que você, como desenvolvedor, montaria para sua empresa
- Comparar os dois processos

Perguntas

Conceitos e revisões

Engenharia

- **Engenharia** (do latim *ingeniu* = "faculdade inventiva, talento") é **a arte, a ciência e a técnica** de bem conjugar os conhecimentos especializados (científicos) de uma dada área do saber com a sua **viabilidade técnico-econômica**, para produzir novas utilidades e/ou transformar a natureza, em conformidade com idéias bem planejadas e em observância aos imperativos de preservação ambiental e de conservação ambiental, na escala que se fizer necessária.
- Fonte: **wikipedia**

Engenharia

- Processo+ Pessoas + Ferramentas
  - Faltou um desses -> Artesanato!
- **Modelo** = ciclo de vida. Abstrato
- **Metodologia** = “processo” informal. Filosófico
- **Processo** = formal (e certificado)
  - MPS-BR
  - CMMI

Software

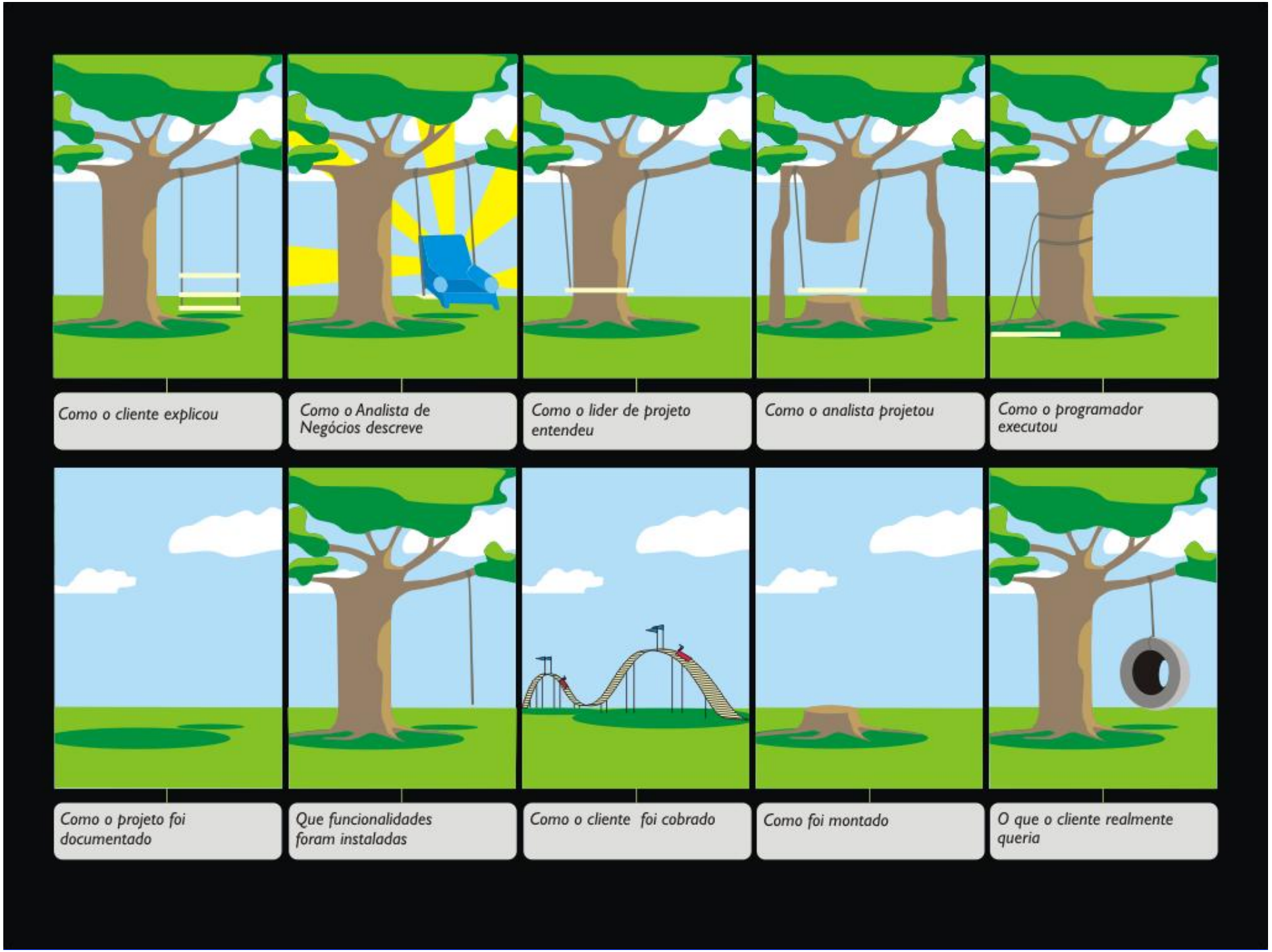
- Programas de computador e documentação associada. Os produtos de software podem ser desenvolvidos para um cliente específico ou para um mercado geral. **[Sommerville 2007]**

Desenvolvimento de Software

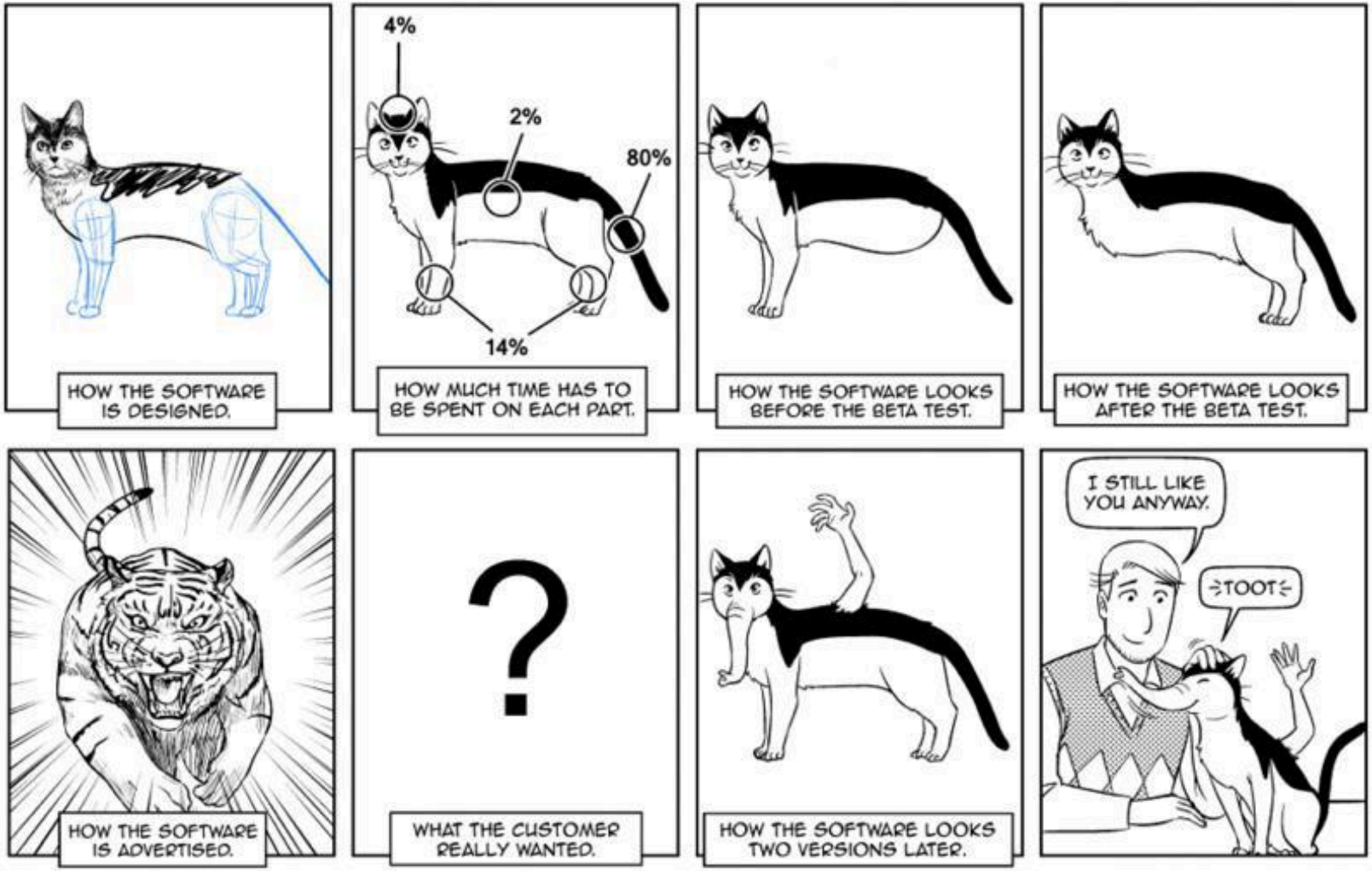
- Ato de elaborar e implementar um sistema computacional, isto é, transformar a necessidade de um utilizador ou de um mercado em um produto de software
- **[Wikipedia]**

Engenharia de Software

- Disciplina de engenharia relacionada com todos os aspectos da produção de software, desde os estágios iniciais de especificação do sistema até sua manutenção, depois deste entrar em operação
- **[Sommerville 2007]**


Como o cliente explicou
Como o Analista de Negócios descreve
Como o lider de projeto entendeu
Como o analista projetou
Como o programador executou
Como o projeto foi documentado
Que funcionalidades foram instaladas
Como o cliente foi cobrado
Como foi montado
O que o cliente realmente queria

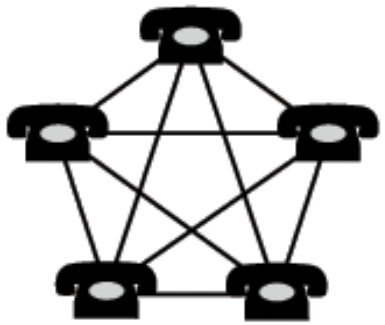
Richard's guide to software development

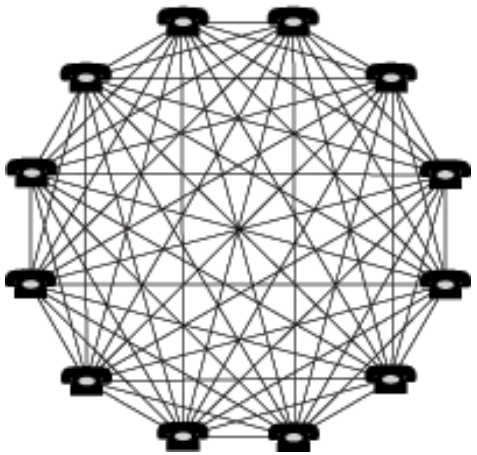


Sandra and Woo by Oliver Knörzer (writer) and Powree (artist) – www.sandraandwoo.com

Mito de Software

- **Aumentar o tamanho da equipe resolve o problema de um projeto atrasado.**





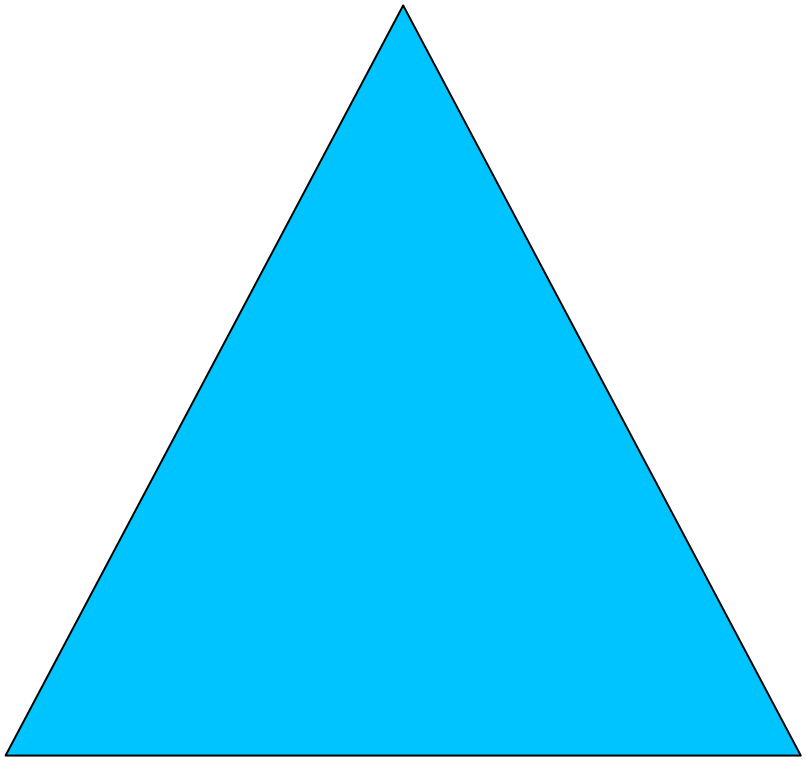
Lei de Metcalfe

O valor de um sistema de comunicação cresce na razão do quadrado do número de usuários do sistema

$$\frac{n(n-1)}{2}$$

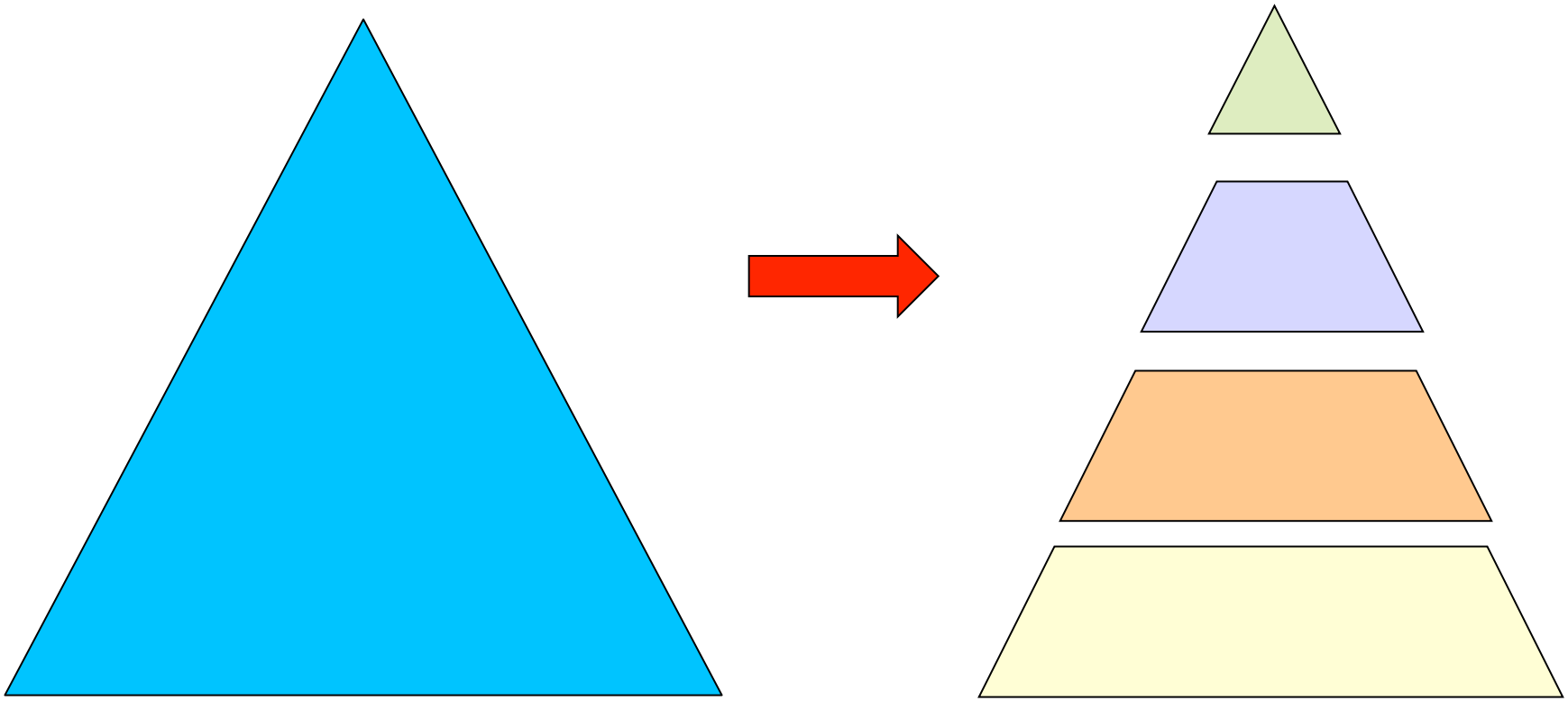
A equipe é uma rede onde cada membro é um host. A informação precisa chegar a cada host. Quanto maior a equipe, mais complicada é a comunicação. **Lembrando ainda que os humanos não são tão rápidos quanto máquinas**

Gerenciamento Simples



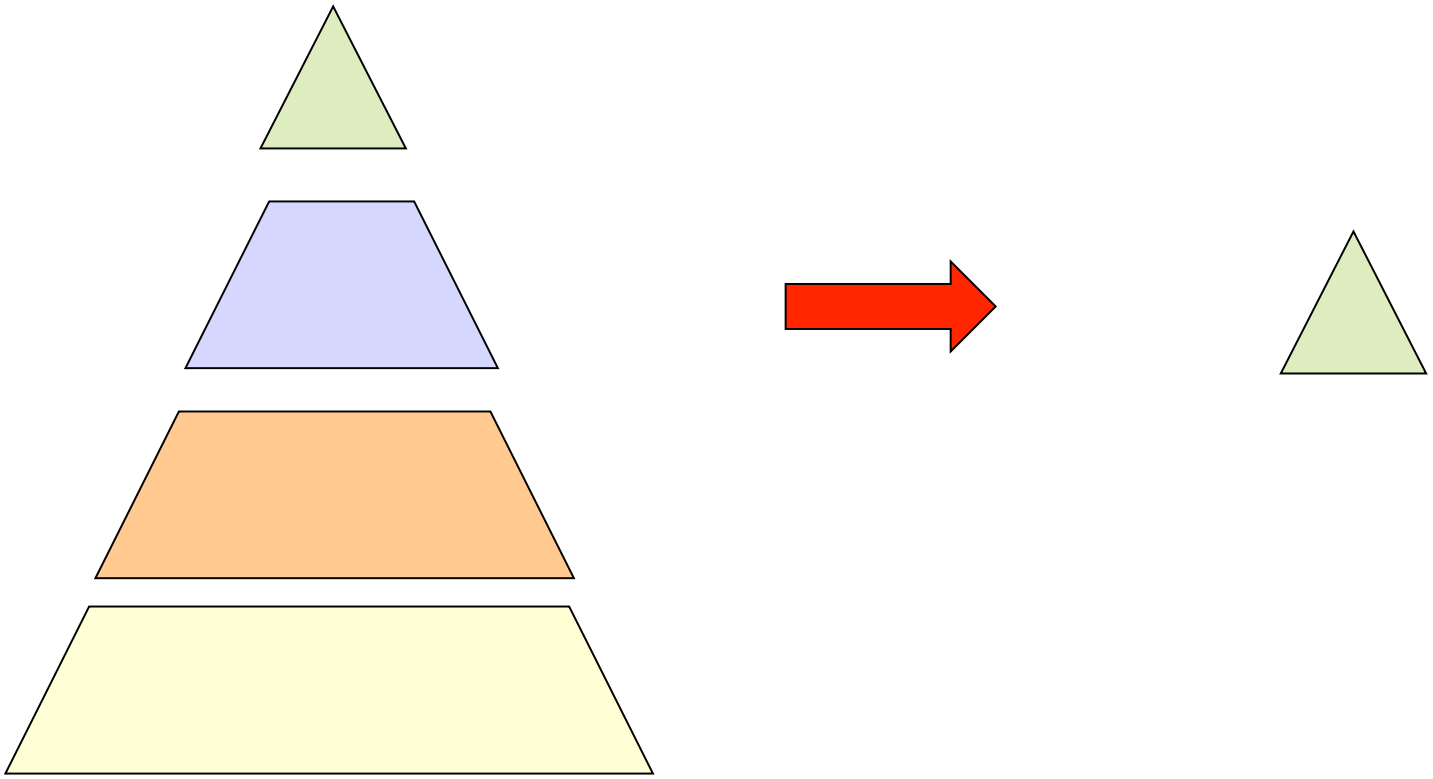
O problema é grande, intimidador e obscuro

Gerenciamento Simples



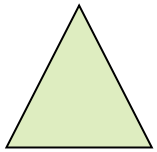
Quebrar o **TODO** em pedaços menores para melhor gerenciar

Gerenciamento Simples



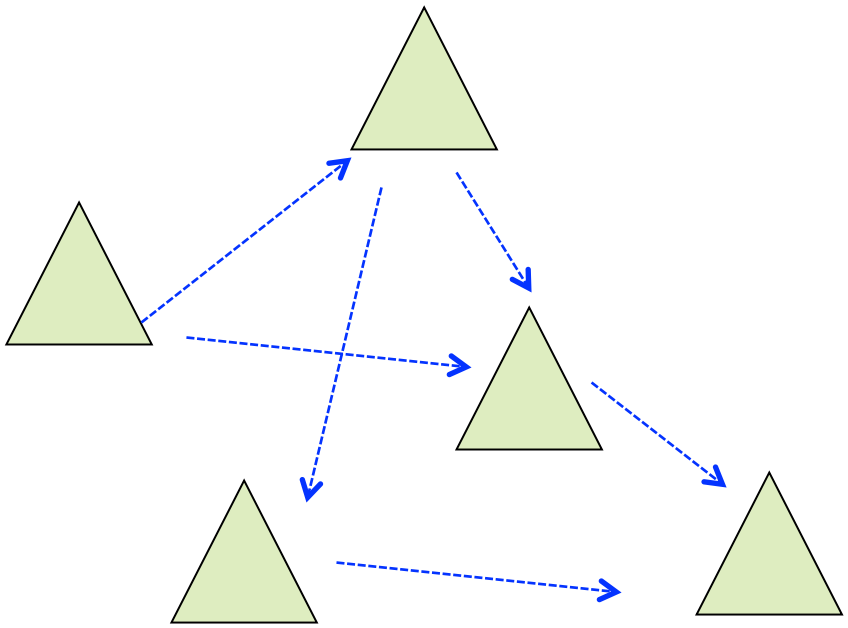
Selecione o um pedaço (módulo) prioritário

Gerenciamento Simples
Como escolher o módulo prioritário?



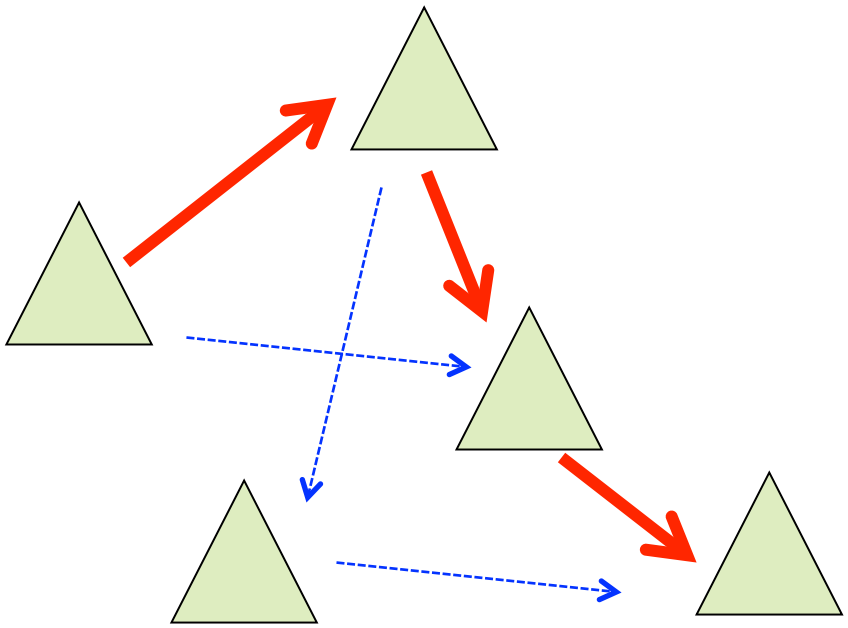
1. Maior risco técnico
2. Selecionado pelo cliente
3. Selecionado pelo gerente para agregar mais valor
4. Caminho crítico
5. Selecionado pela equipe

Gerenciamento Simples
Caminho crítico



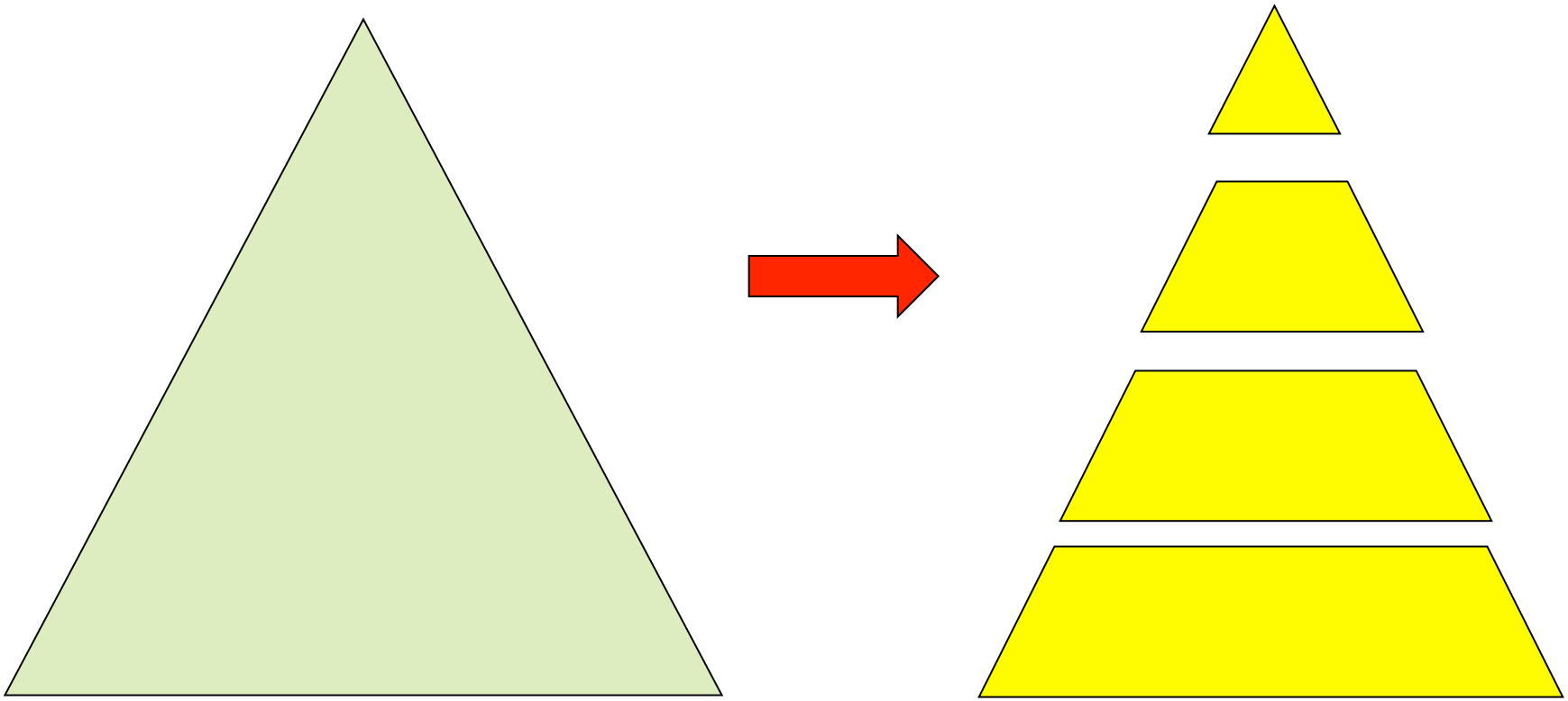
Ordem de fazer as tarefas que se não for seguida implicará em atraso para o projeto

Gerenciamento Simples
Caminho crítico



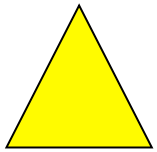
Ordem de fazer as tarefas que se não for seguida implicará em atraso para o projeto

Gerenciamento Simples



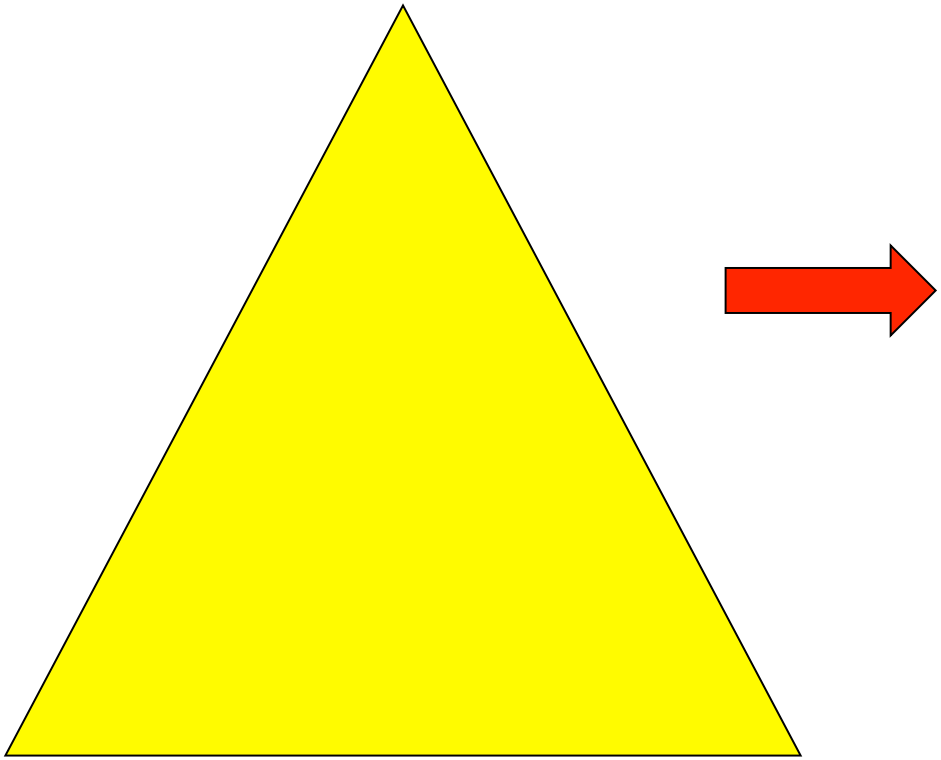
Quebrar um módulo e divide em funcionalidades/tarefas para melhor gerenciar

Gerenciamento Simples
Como escolher a funcionalidade prioritária?



1. Maior risco técnico
2. Selecionado pelo cliente
3. Selecionado pelo gerente para agregar mais valor
4. Caminho crítico
5. Selecionado pela equipe

Gerenciamento Simples



WHAT? (O QUE ?)
WHERE? (ONDE ?)
WHEN? (QUANDO ?)
WHY? (PORQUE?)
WHO? (QUEM ?)

HOW? (COMO?)

Para cada tarefa, os dados deverão estar esclarecidos

Gerenciamento Simples

WHAT? (O QUE ?)

Descrição sucinta do que fazer

WHERE? (ONDE ?)

Onde será realizada a tarefa?

WHEN? (QUANDO ?)

Início e Fim Programado da Tarefa. Qtde de horas necessárias

WHY? (PORQUE?)

Qual a importância de fazer esta tarefa? Indicará a prioridade da mesma

WHO? (QUEM ?)

Quem fará essa tarefa? Qual o perfil?

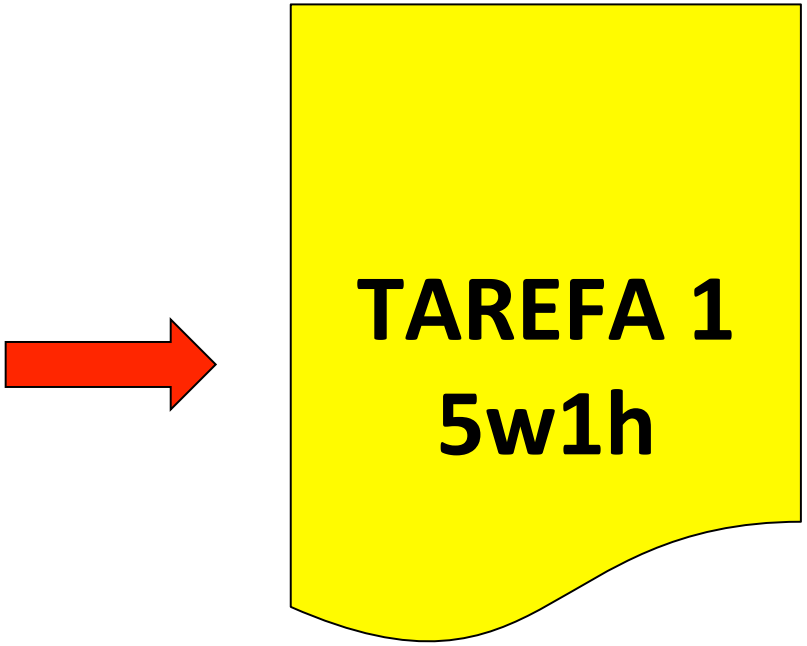
HOW? (COMO?)

Detalhamento de como a tarefas será realizada. Só detalhar quando a mesma for realizada

Gerenciamento Simples

WHAT? (O QUE ?)
WHERE? (ONDE ?)
WHEN? (QUANDO ?)
WHY? (PORQUE?)
WHO? (QUEM ?)

HOW? (COMO?)

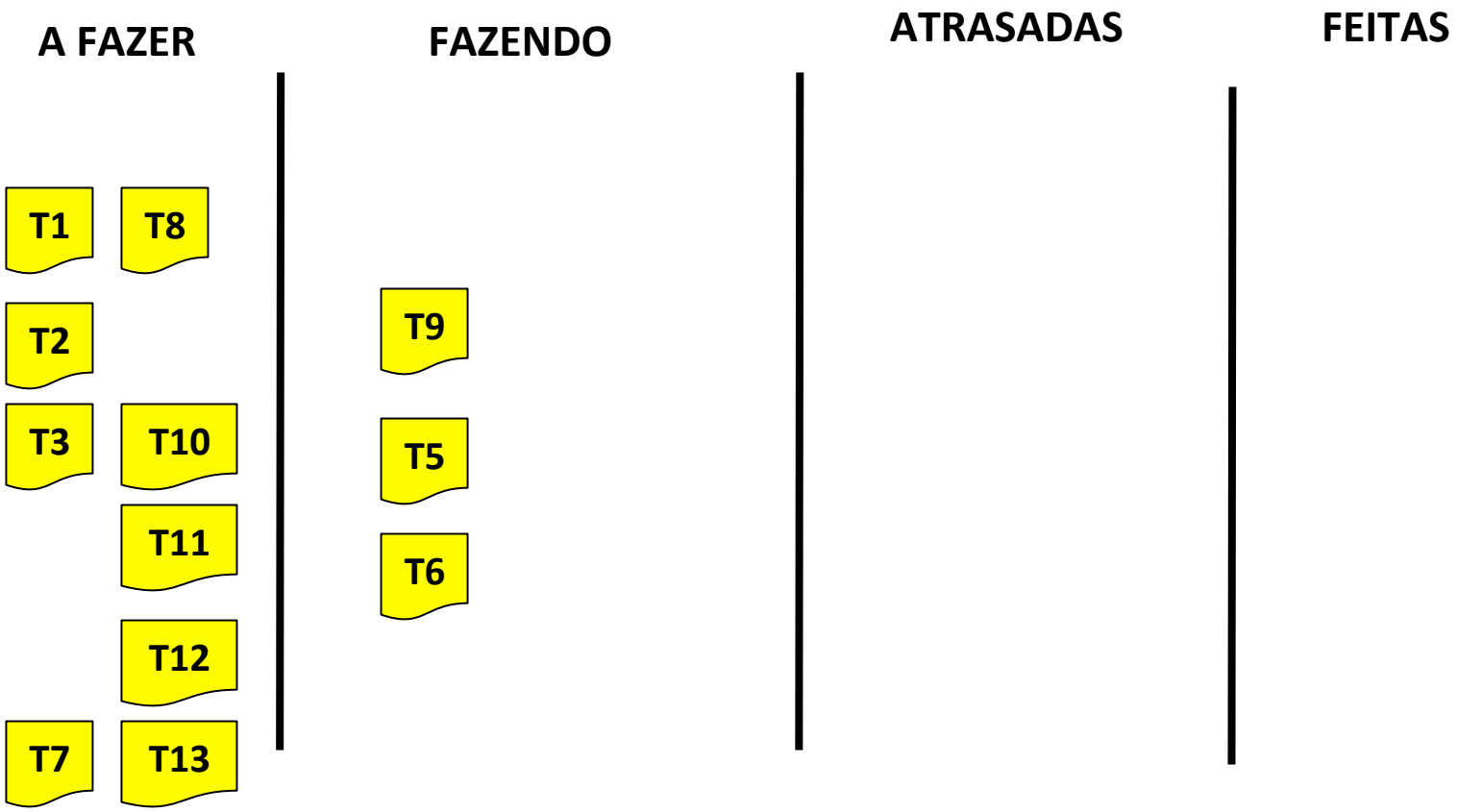


Preencha um cartão (post-it) para cada tarefa

Gerenciamento Simples

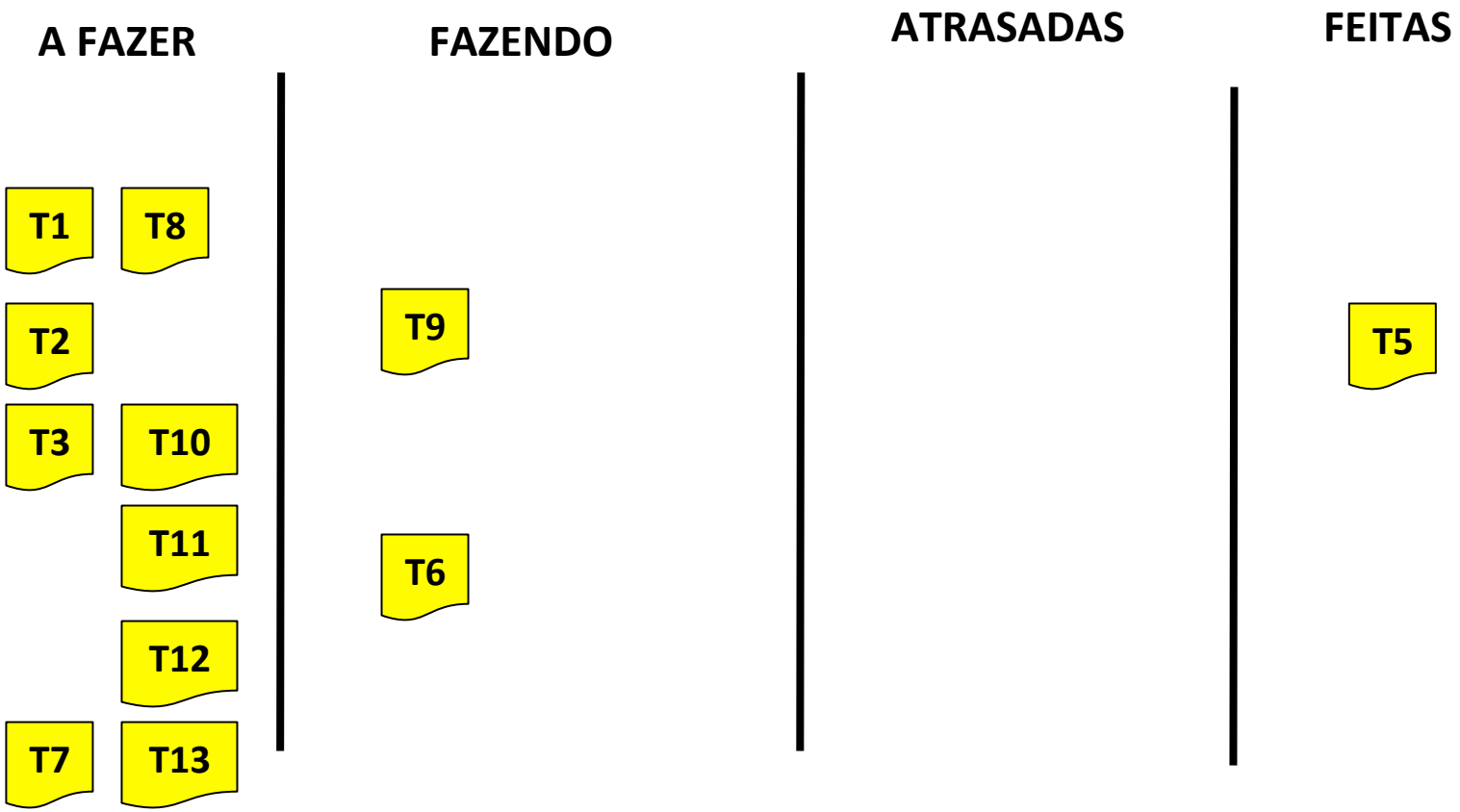
Crie um quadro e coloque todas as tarefas na coluna **A FAZER**

Gerenciamento Simples



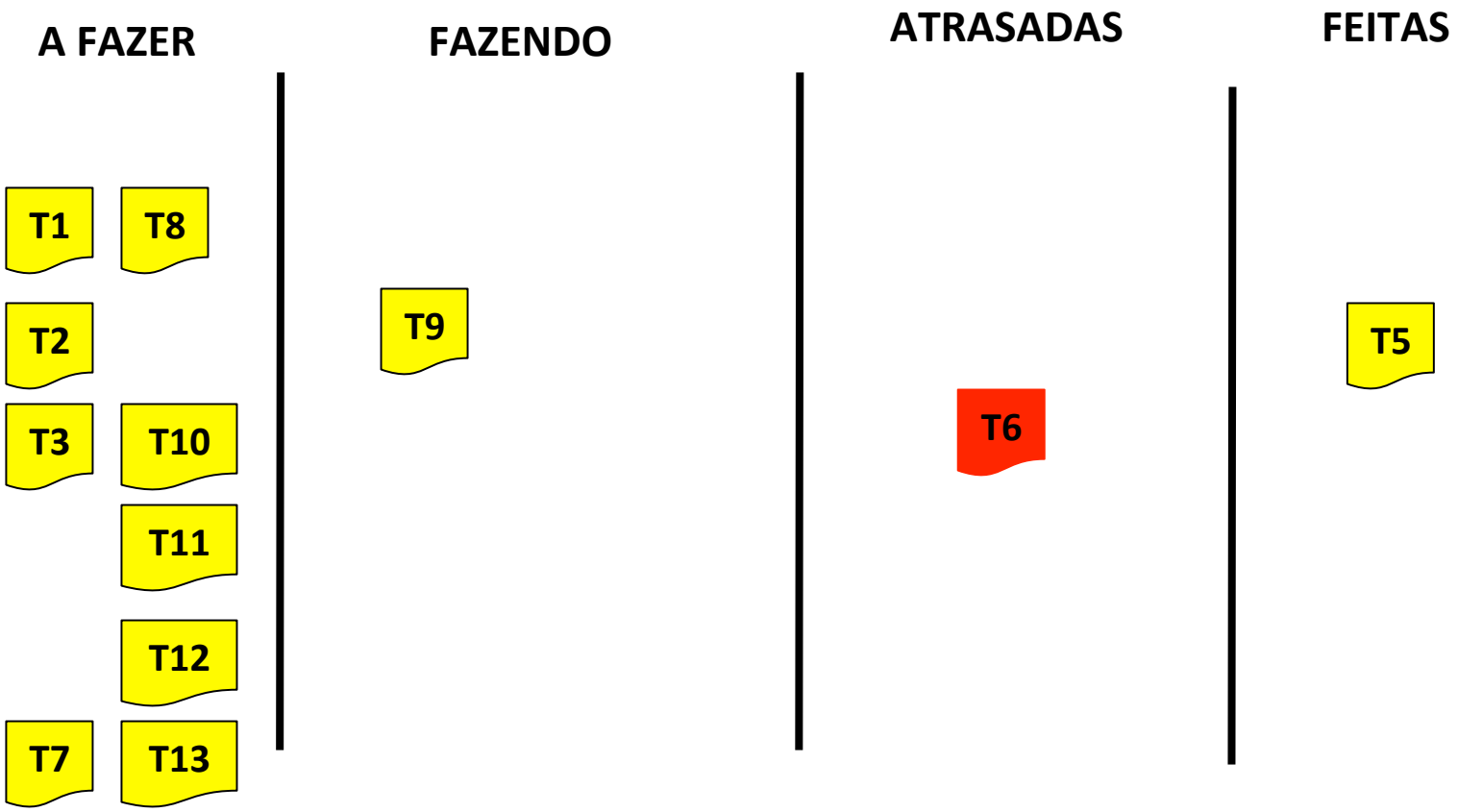
Selecione as prioritária , distribua entre a equipe (WHO?)
e coloque-as na coluna **FAZENDO**

Gerenciamento Simples



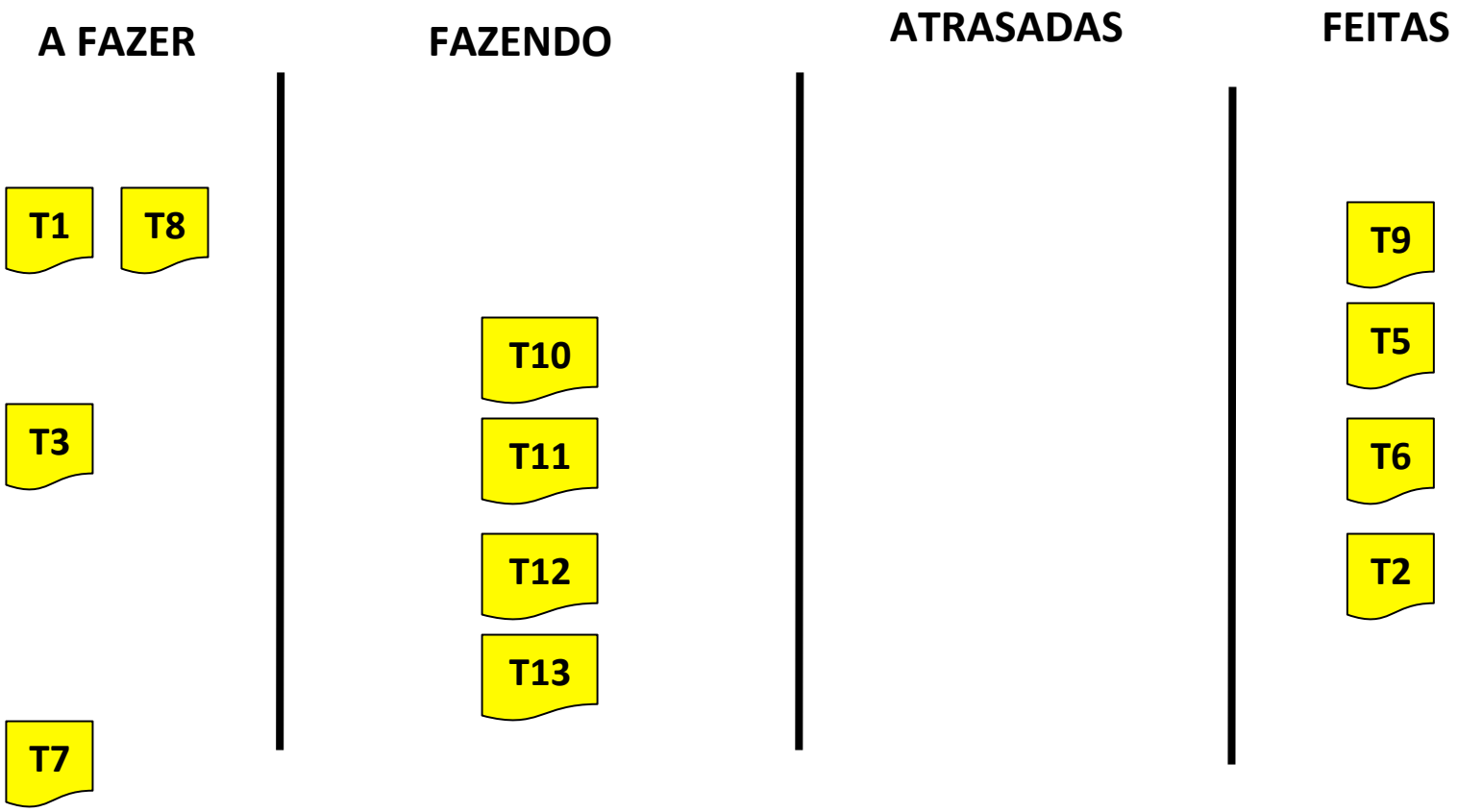
Quando a tarefa estiver 100% completa vai para a coluna de **FEITAS.**
Alocar o pessoal em nova tarefa

Gerenciamento Simples



Quando a tarefa atrasada vai para a coluna de **ATRASADAS.**
Resolver primeiro a atrasada antes de pegar uma próxima tarefa

Gerenciamento Simples

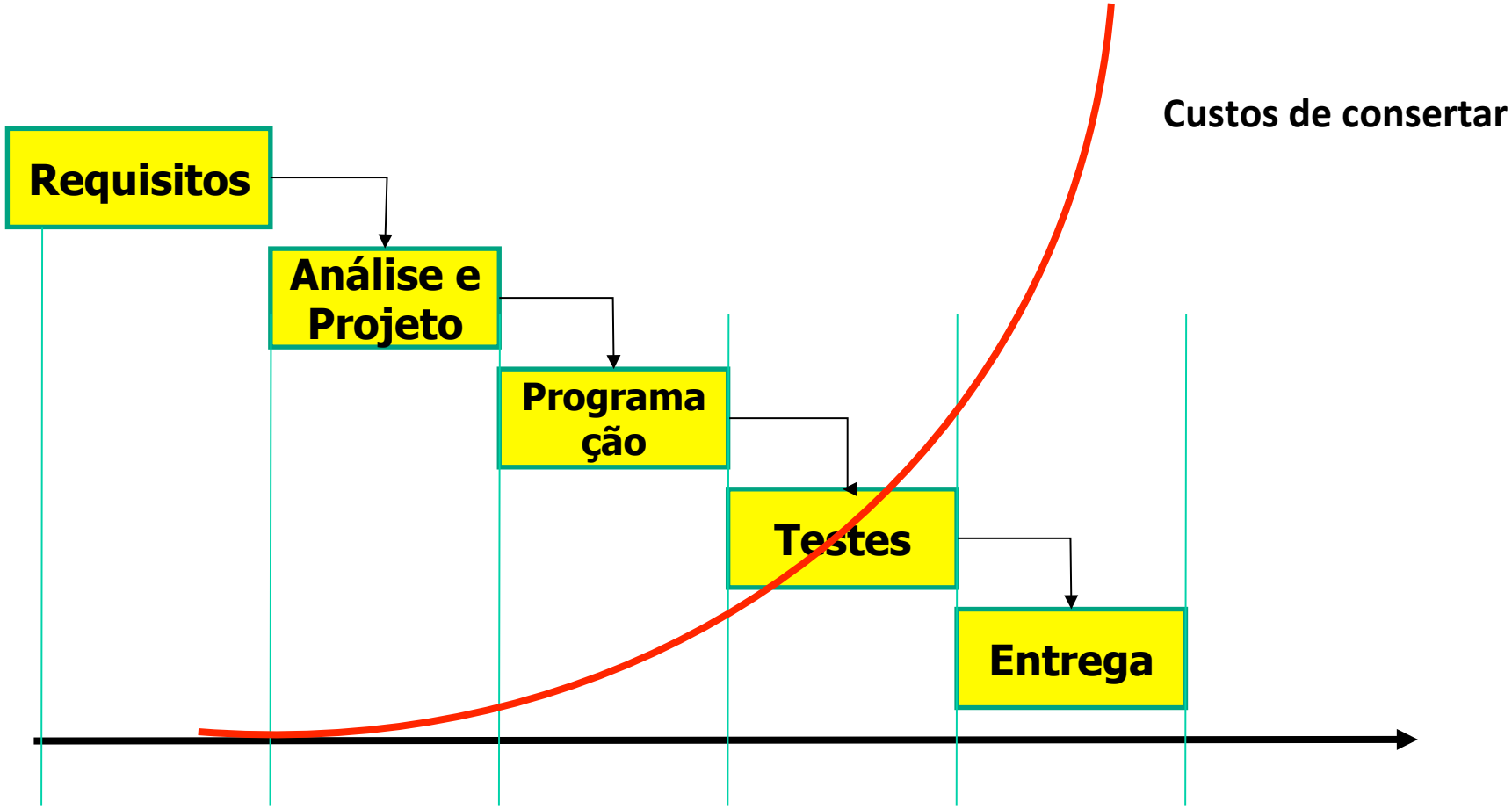


Continuando ...

Gerenciamento Simples

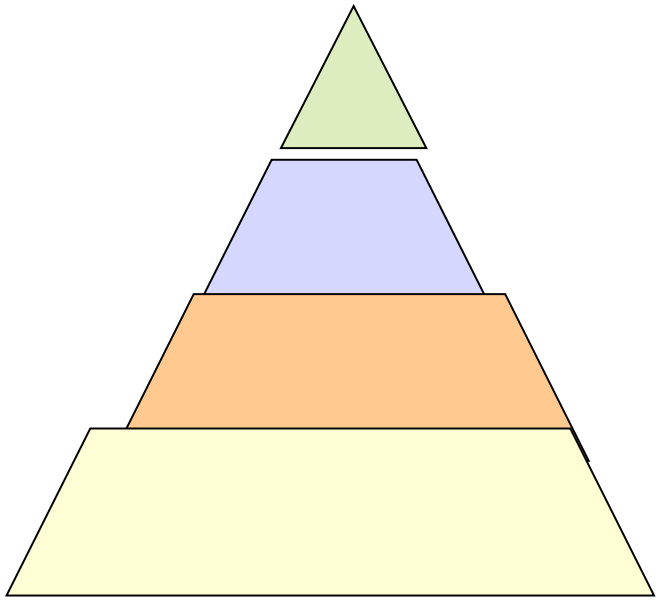
Quando todas as tarefas estiverem feitas, o módulo estará pronto!

Cascata



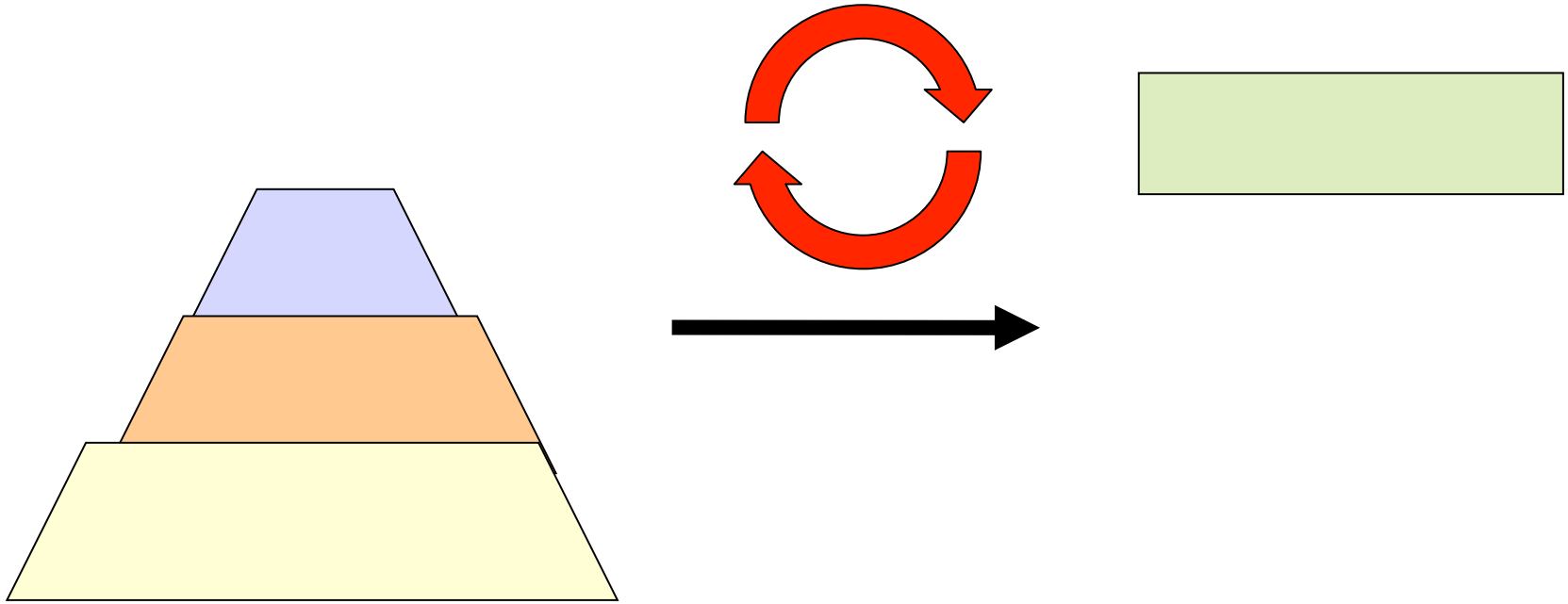
Quanto mais avançado estiver o desenvolvimento, mas caro ele é para consertar!

Iterativo e Incremental



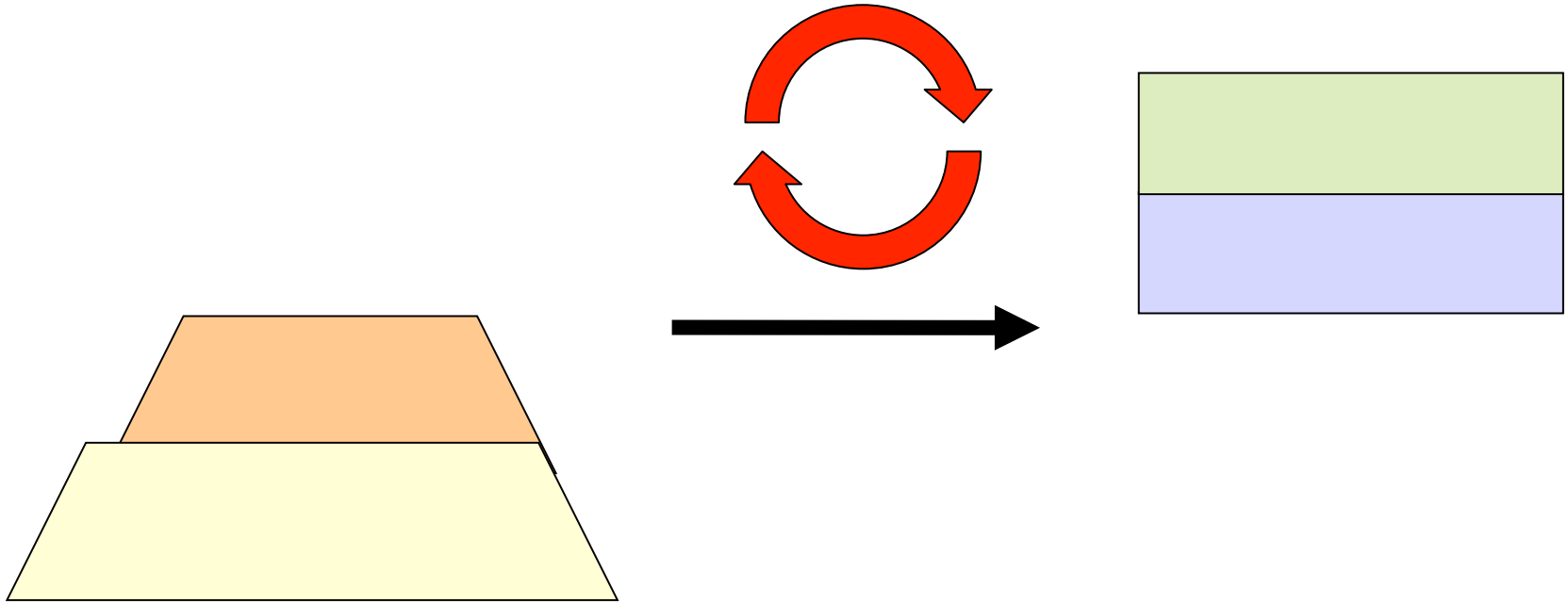
SOFTWARE 0% COMPLETO

Iterativo e Incremental



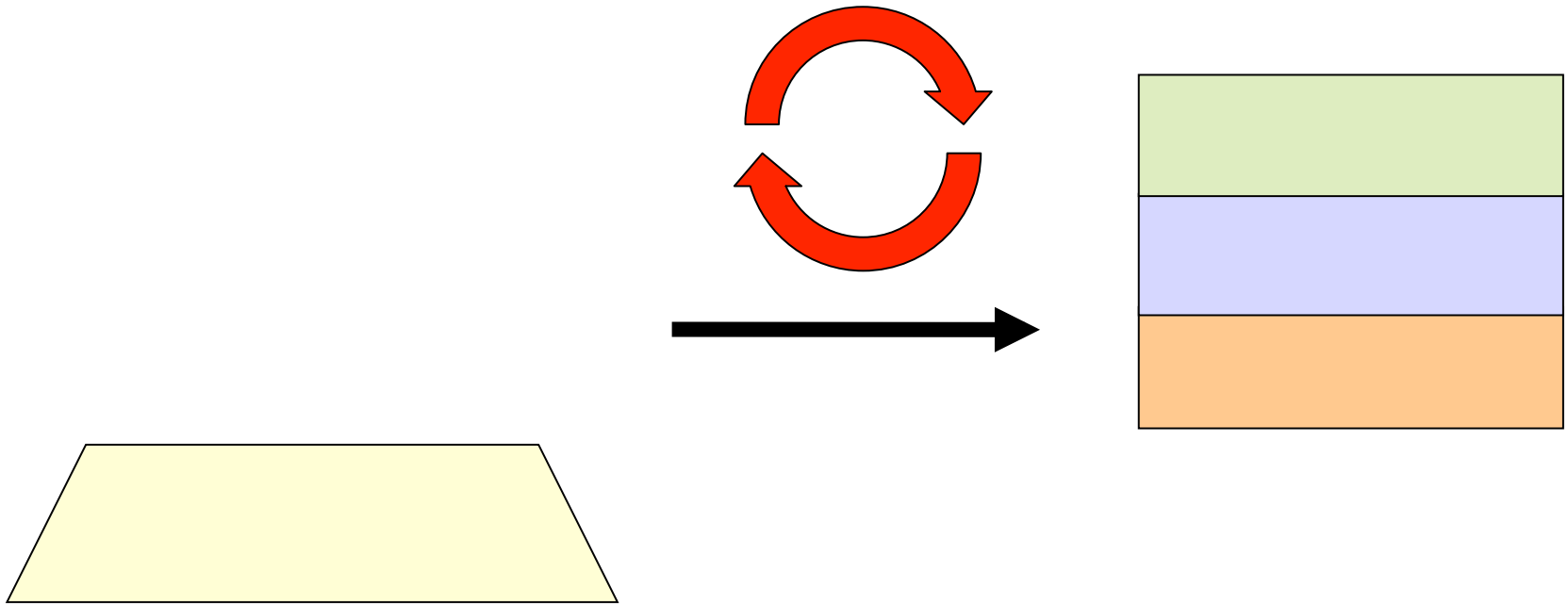
SOFTWARE 25% COMPLETO

Iterativo e Incremental



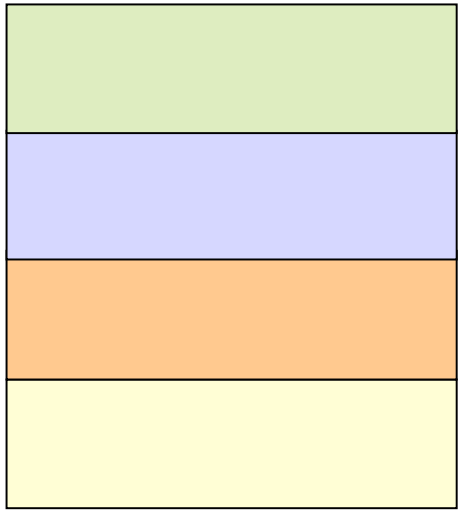
SOFTWARE 50% COMPLETO

Iterativo e Incremental



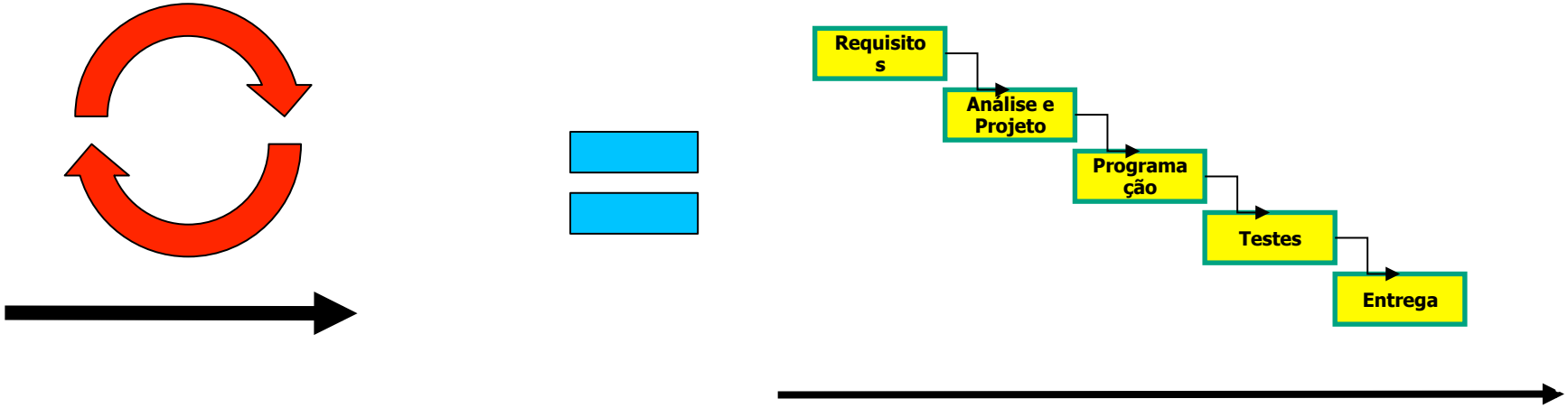
SOFTWARE 75% COMPLETO

Iterativo e Incremental



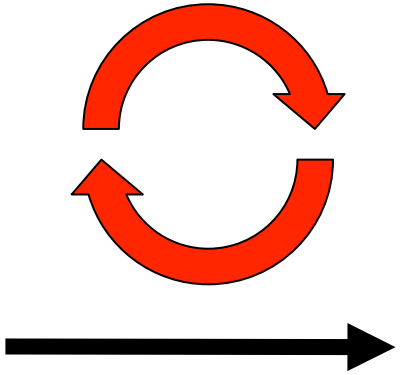
SOFTWARE 100% COMPLETO

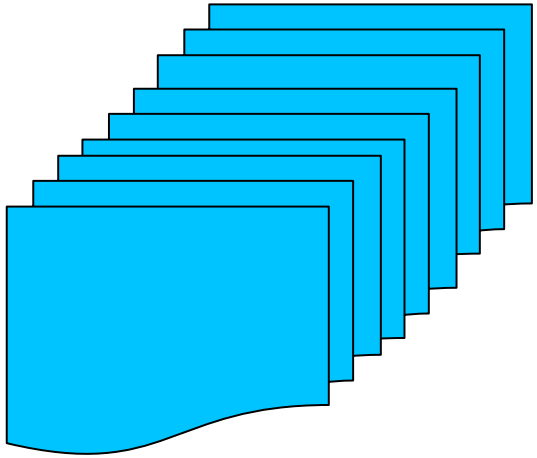
Iterativo e Incremental



Para cada "pedaço" do sistema, será repetido Requisitos, A&P, Codificação, Testes e Entrega

Iterativo e Incremental





Documentação
Gigante

Estouro de prazos/custos/qualidade/escopo
e valores altos de implantação do processo

Manifesto Ágil

"Estamos descobrindo maneiras melhores de desenvolver software fazendo-o nós mesmos e ajudando outros a fazê-lo. Através desse trabalho, passamos a valorizar:

- *Indivíduos e interações* ***mais que processos e ferramentas;***
- *Software em funcionamento* ***mais que documentação abrangente;***
- *Colaboração com o cliente* ***mais que negociação de contratos;***
- *Responder a mudanças* ***mais que seguir um plano.***

Ou seja, mesmo dando valor aos itens à direita, valorizamos mais os itens à esquerda."

Kent Beck*, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham,Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas*

http://agilemanifesto.org/

Perguntas

Conceito de Gerência de Configuração de Software

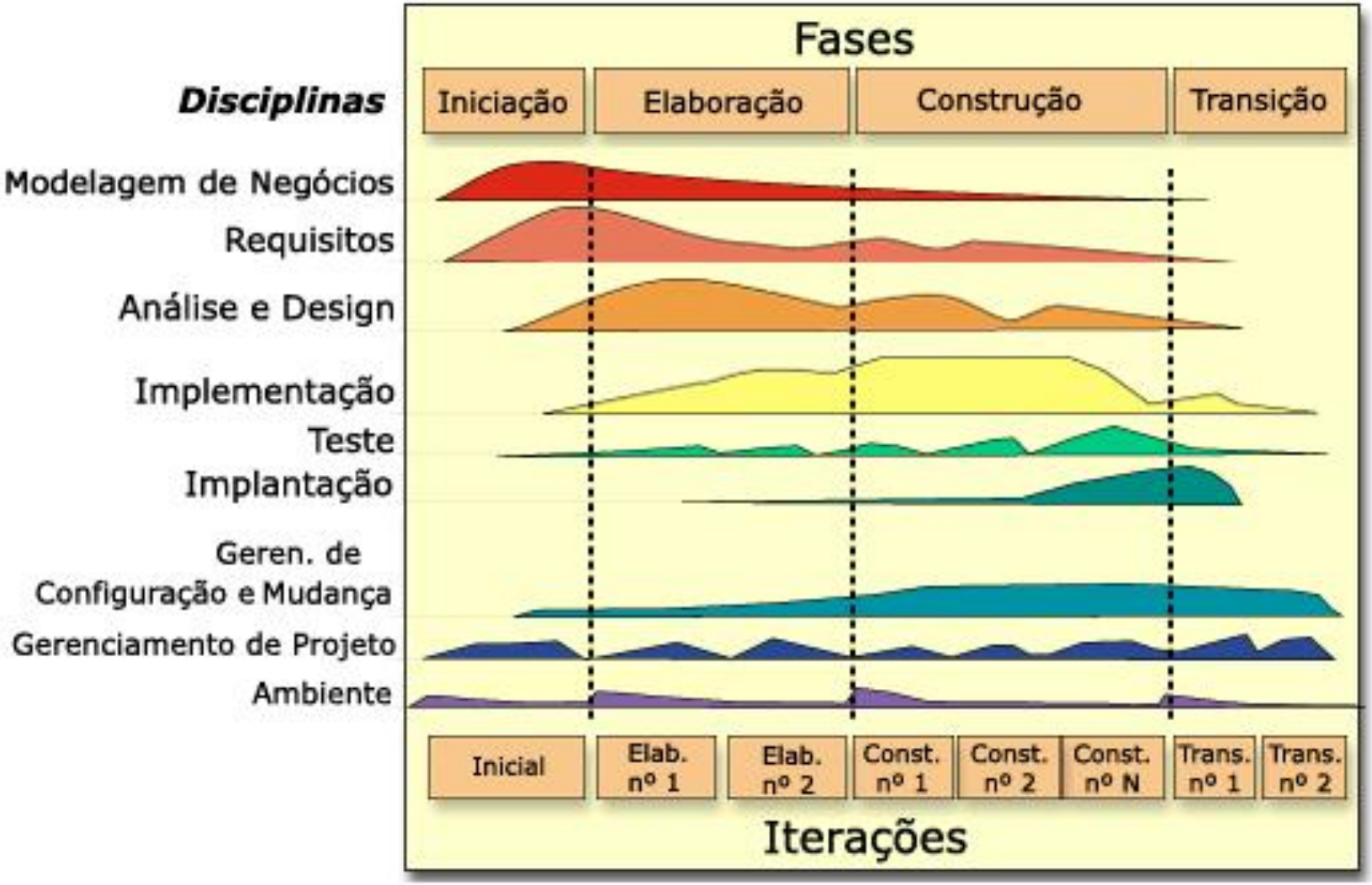
Gerencia de Configuração de Software (GCS)

- conjunto de atividades projetadas para **controlar as mudanças pela identificação dos produtos do trabalho que serão alterados**, estabelecendo um relacionamento entre eles, definindo o mecanismo para o **gerenciamento de diferentes versões destes produtos**, controlando as mudanças impostas, e auditando e relatando as mudanças realizadas.
- Fonte: **Roger Pressman**
- CGS = Controle de Configuração + Controle de Mudanças + Controle de versão + Controle de ambiente

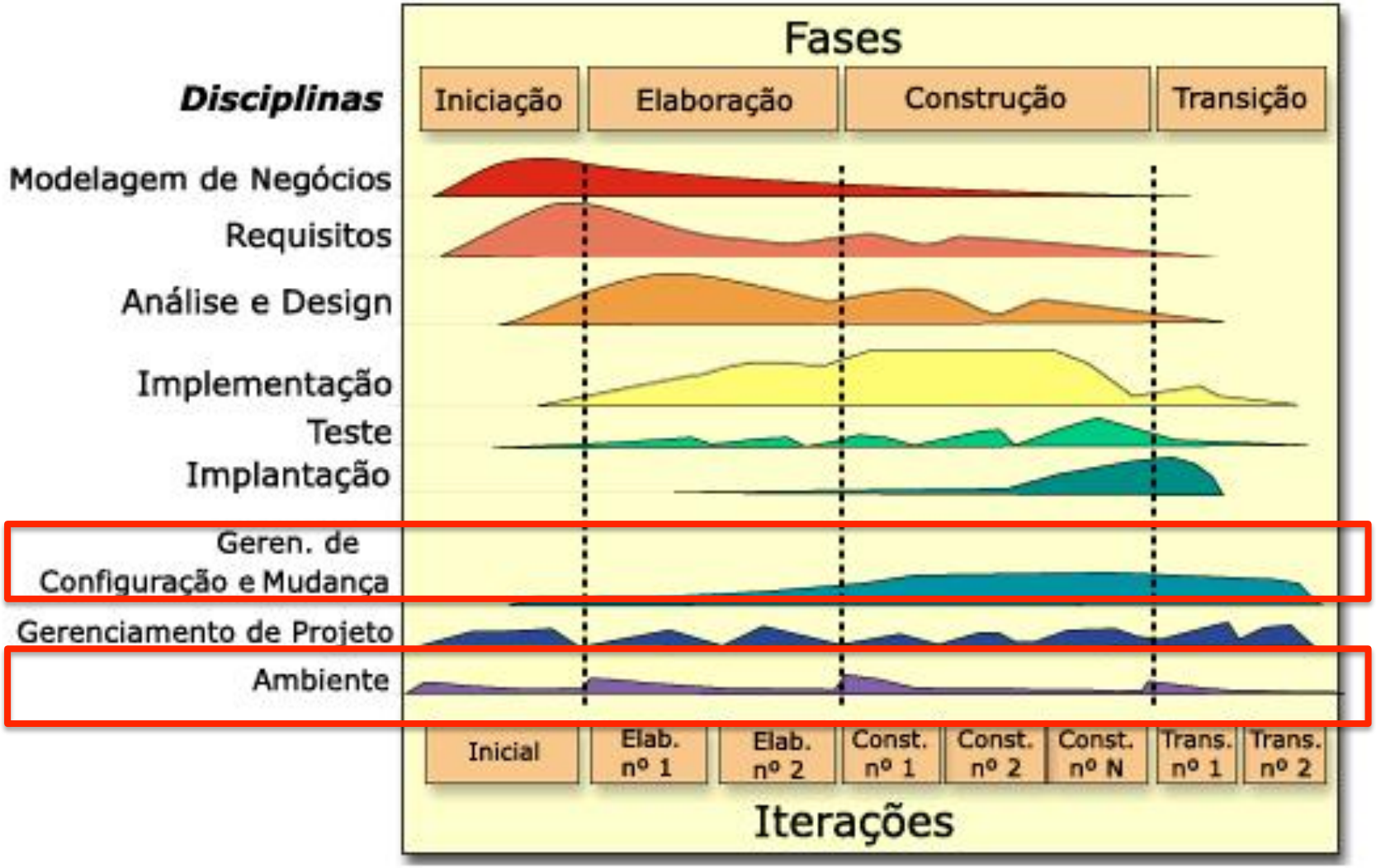
Gerencia de Configuração de Software (GCS)

- Controlar mudanças:
  - quando? o que? quem? por quê?
  - Ex: Bugzilla
- Controlar versões
  - Builds, marcos, realeases...
  - Ex: Git

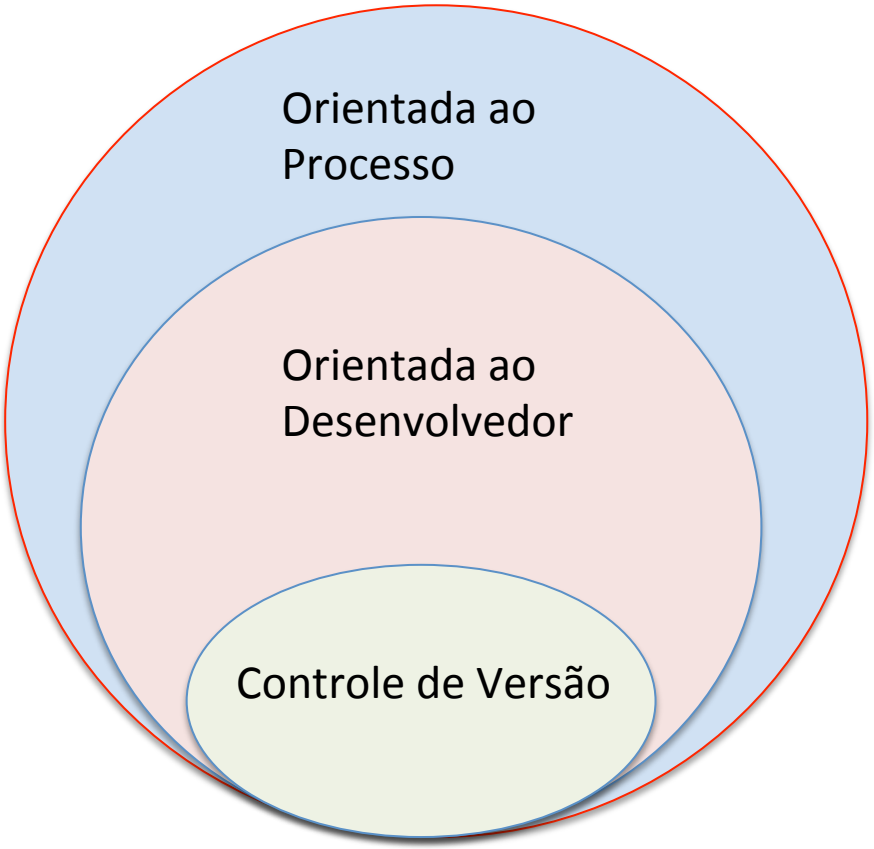
Rational Unified Process (RUP)



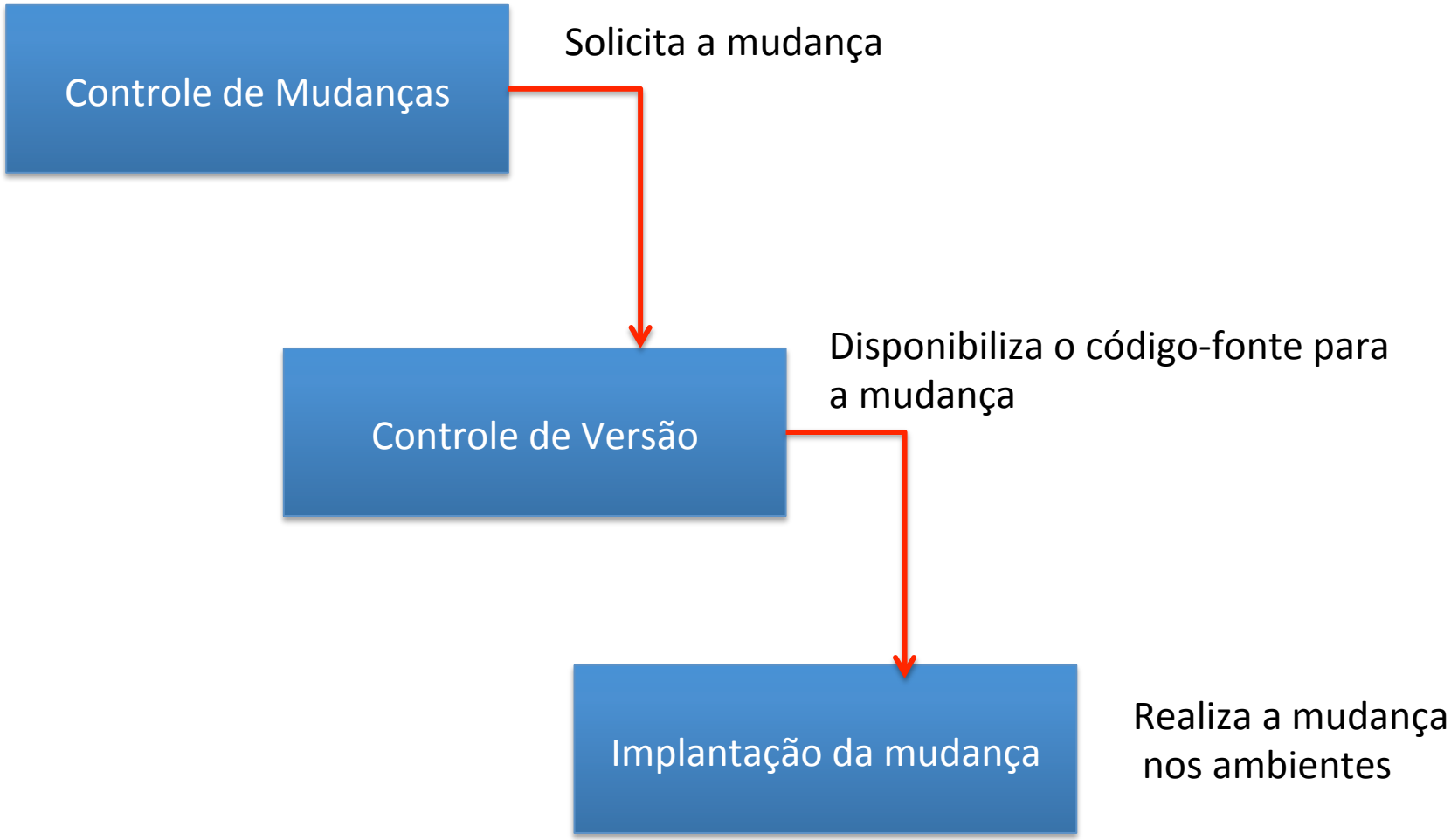
GCS no RUP


Fases
Disciplinas
Iniciação
Elaboração
Construção
Transição
Modelagem de Negócios
Requisitos
Análise e Design
Implementação
Teste
Implantação
Geren. de Configuração e Mudança
Gerenciamento de Projeto
Ambiente
Inicial
Elab. nº 1
Elab. nº 2
Const. nº 1
Const. nº 2
Const. nº N
Trans. nº 1
Trans. nº 2
Iterações

Ferramentas GCS

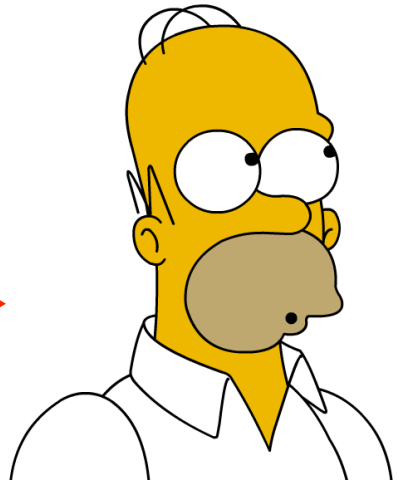


Ferramentas GCS



GCS no popular

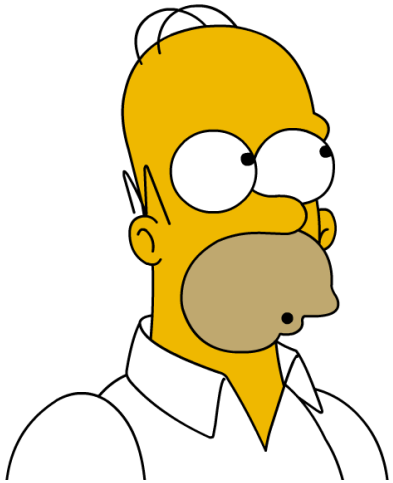
desenvolvedor



Usuário final

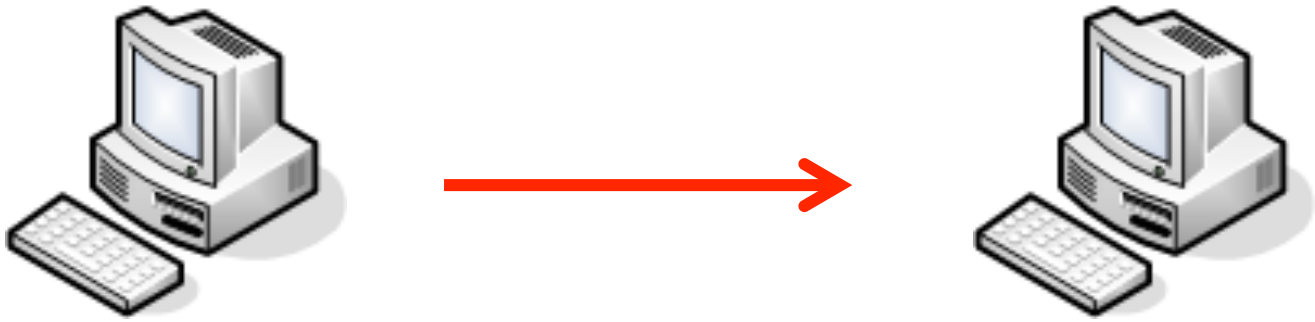
GCS no popular

desenvolvedor



Usuário final

GCS no popular



Ambiente de desenvolvimento

Ambiente de produção

- maquinas dos desenvolvedores
- servidores de desenvolvimento
- ...

- maquinas dos usuários
- servidores de produção
- ...

Problemas de GCS

"Usuários do Windows 7 estão enfrentando problemas sérios com uma atualização disponibilizada na noite de terça-feira, 9, pela Microsoft. A instalação deste update faz com que o computador não possa ser iniciado novamente.

O problema, originado quando a pessoa instala a atualização KB2823324, só está atingindo quem utiliza a versão de 32 bits do sistema operacional."

http://olhardigital.uol.com.br/produtos/digital_news/noticias/usuarios-do-windows-7-tem-pc-inutilizado-por-atualizacao-do-sistema

Conceitos

Gerentes na GCS

Gerentes na GCS

Gerente de Projeto

- Preocupado com o custo/ tempo/escopo do projeto
- "O que for colocado em produção irá atender às necessidades/desejos dos *stakeholders*?"

Gerente de CGS

- Preocupado com as baselines
- Preocupado com as releases
- Preocupado com as mudanças nos ambientes de desenvolvimento, teste e produção
- "O que for colocado em produção irá rodar sem causar danos no ambiente de produção?"
- "Tudo colocado e tirado do repositório está rastreável?"
- "Tudo que foi colocado e tirado dos ambientes de produção e desenvolvimento está rastreável?"

Configuração

- É o estado em que um sistema se encontra em um determinado momento. Este sistema pode ser composto de todo tipo de elementos, como peças de hardware, artefatos eletrônicos ou não (i.e. documentos em papel), etc. A Configuração de Software trata apenas dos elementos que se encontram em formato eletrônico e fazem parte dessa configuração. Isso inclui todos os arquivos fontes, todos os documentos eletrônicos, as ferramentas de software utilizadas para construir ou mesmo ler estes arquivos, o sistema operacional utilizado, as bibliotecas de software, etc.
- **Wikipedia**

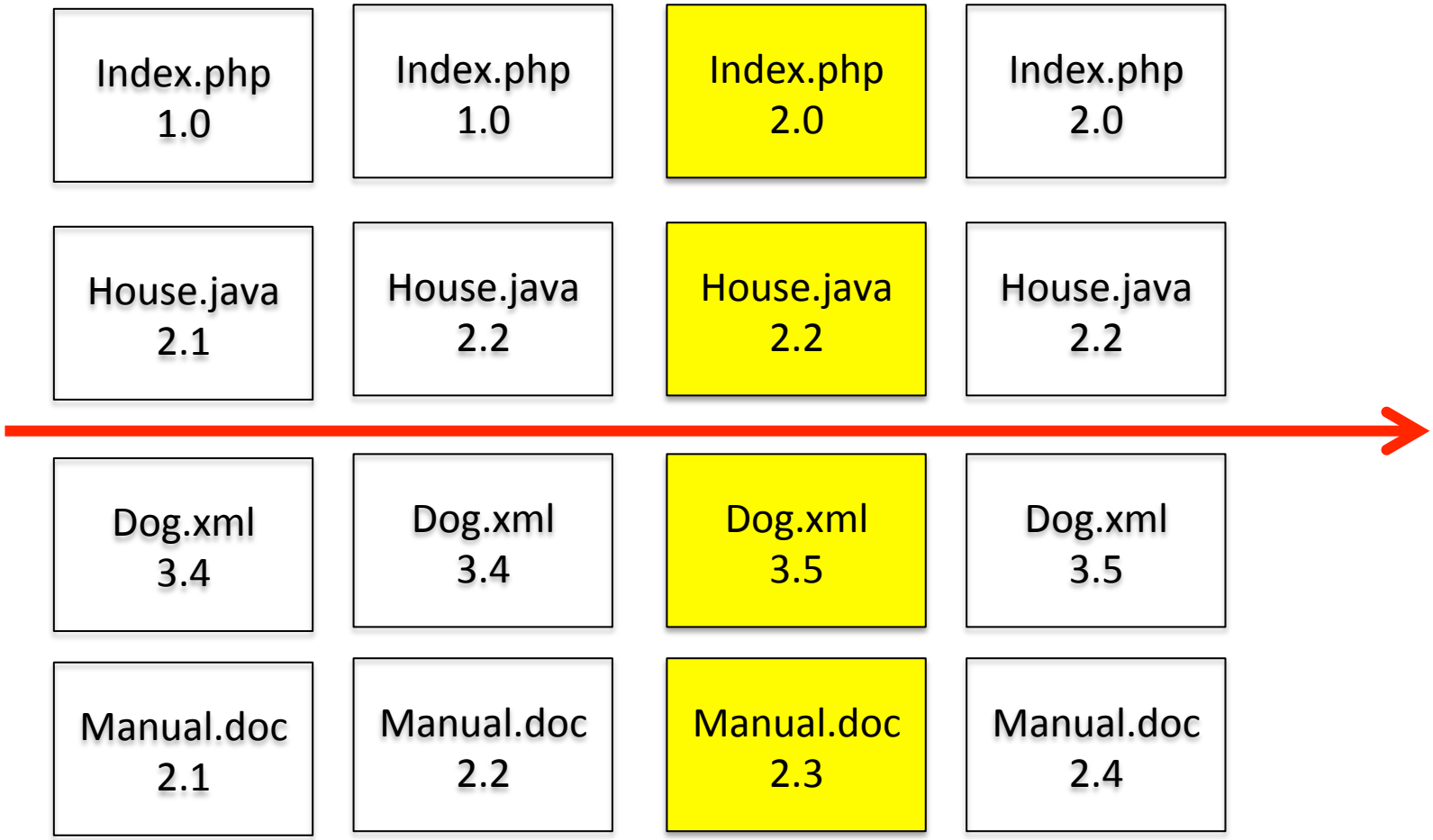
Item de configuração

- É um elemento unitário que será gerenciado: um arquivo de código fonte, um documento de texto, um projeto de uma placa eletrônica, uma planta feita em papel, um CD-ROM de instalação de um sistema operacional, etc. A configuração de um sistema é basicamente a lista de todos os itens de configuração necessários para reproduzir um determinado estado de um sistema. Em geral números de versão são associados aos itens de configuração de forma a podermos identificar também a evolução destes itens.
- Artefatos gerados durante do desenvolvimento

Baseline

- É um conceito de gerenciamento de configuração de software que nos ajuda a controlar as mudanças, sem impedir seriamente as mudanças justificáveis. Segundo PRESSMAN no contexto de engenharia de software, definimos uma linha-base como **um marco de referência no desenvolvimento de um software**, que é caracterizado pela entrega de um ou mais itens de configuração (SCI) e pela aprovação desses SCIs, obtida por meio de uma revisão técnica formal.
- Quando a *baseline* coincide com a entrega formal para o cliente, ela é chamada de ***release***

Baseline


Index.php
1.0
Index.php
1.0
Index.php
2.0
Index.php
2.0
House.java
2.1
House.java
2.2
House.java
2.2
House.java
2.2
Dog.xml
3.4
Dog.xml
3.4
Dog.xml
3.5
Dog.xml
3.5
Manual.doc
2.1
Manual.doc
2.2
Manual.doc
2.3
Manual.doc
2.4

Repositório

- É o local de armazenamento dos itens de configuração que estão sob controle da Gerência de Configuração de Software.
- Cada item de configuração possui um número de versão. Quando o item é armazenado pela 1ª vez, sua versão é 1
- Cada vez que um item de modificado e recolocado no repositório, sua versão é incrementada em uma unidade
- A ação de colocar no repositório é chamada ***check-in***. A ação de tirar do repositório é chamada ***check-out***

Branch

- Quebra da linha do desenvolvimento em mais de um caminho. Isso pode ser chamado de ramificação (ramo), braços ou em inglês *branches*. Isso é muito útil quando se conquista uma versão estável dos documentos (ou software) ou quando se quer fazer uma tentativa "não convencional" fora do ramo principal.
- **Wikipedia**

Conflito

- É a situação em que dois ou mais usuários modificam o mesmo documento no mesmo intervalo de tempo entre o envio de uma nova versão ao sistema. **A chance de conflitos aumentam quando aumentam o número de usuários utilizando o mesmo conjunto de documentos.** O envio de alterações muito grandes podem também aumentar a chance de conflito.
- **Mesclagem** (ou *merge* em inglês) consiste na aglutinação (ou fusão) automática de versões através da comparação entre elas, quando há um conflito de edições simples
- Quando dois ou mais desenvolvedores modificam uma mesma região num documento é necessário resolver o conflito "**manualmente**".

Conflitos

P3	P4	P5

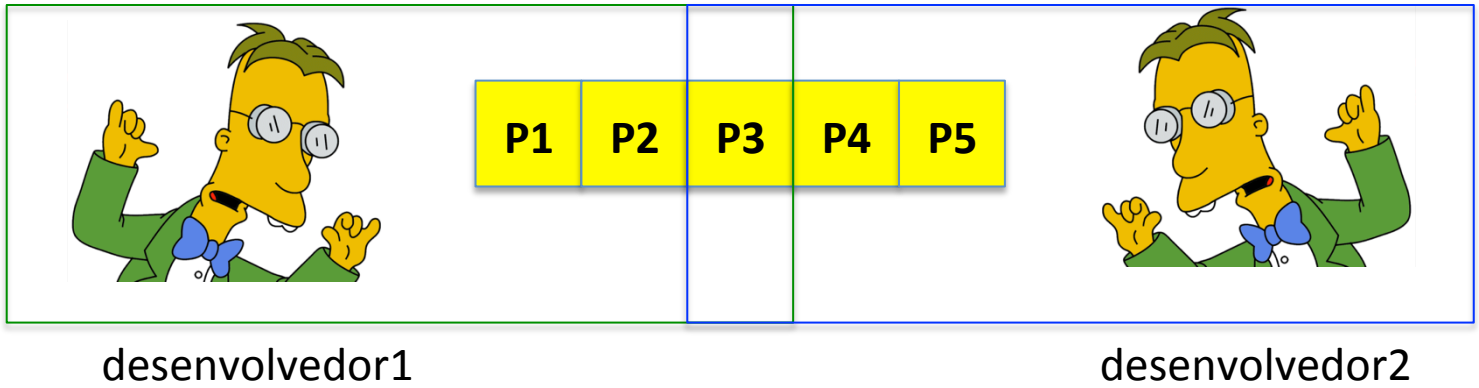
desenvolvedor2

Conflitos

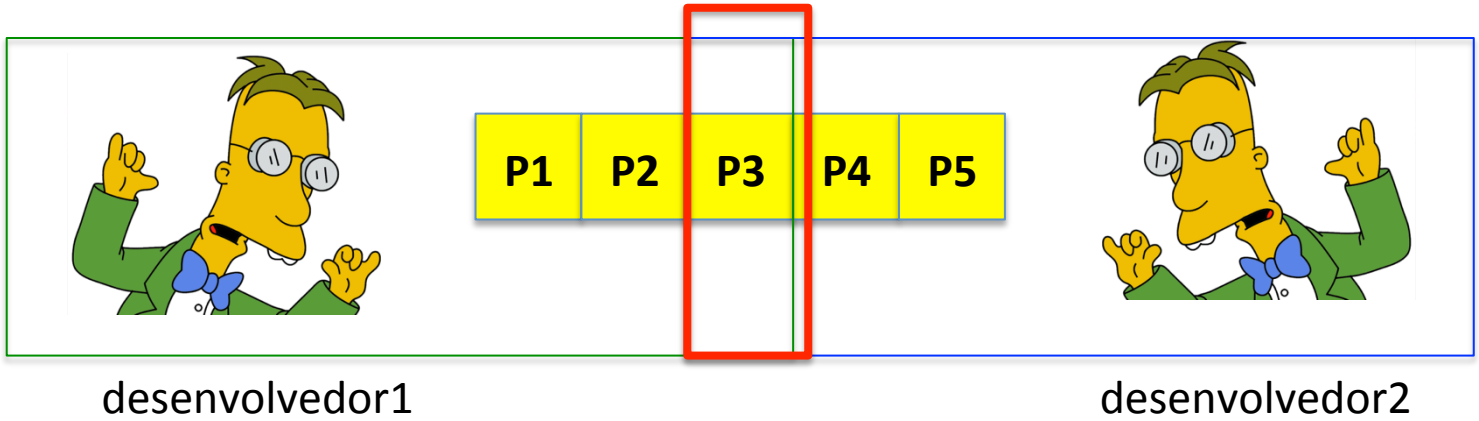
desenvolvedor1

P1	P2	P3

Conflitos



Conflitos



Atividade prática 1

Atividade prática 1

- Grupo de 3 a 5 pessoas
- 30 min para resolver tudo. No final apresentar para turma sua solução e sua divisão dos exercícios
- Fazer em Java
- Um **main()** único para todos os exercícios
- A versão entregue não pode destruir o que já está rodando. Planeje suas entregas para que o impacto no ambiente de produção seja **mínimo**.
- Definir o **ambiente de desenvolvimento** (Sistema Operacional, Compilador, IDE, ??)
- Definir o **ambiente de produção** (Sistema Operacional, ??)
  - Dica: como faço para instalar o programa em produção?

Atividade prática 1

- Vamos simular um dia de trabalho! A cada **6 min**, cada equipe tem de entregar uma versão **compilável** da lista
- Cada modificação entregue deve evidenciar quem fez, quando fez, por que fez e o que fez
- A versão entregue não pode destruir o que já está rodando. Planeje suas entregas para que o impacto no ambiente de produção seja **mínimo**.
- Cada versão entregue não completa, porém compilável e rodando é chamada de ***build***
- Nunca pode subir código que não compila. Código ***sangrando***

Atividade prática 1
Perguntas que não querem calar

- Como os exercícios foram divididos entre as pessoas? Por que usaram esta estratégia?
- Quem fez o que?
- Como o exercício foi codificado? Cada exercício virou o que no código? Uma classe? Um método? Outro?
- O código está padronizado? O código compila? O código roda conforme o esperado? Quem homologou?
- Quais foram os problemas encontrados?
- Se tivesse de fazer de novo, o que melhoraria?

Atividade prática 1
Considerações

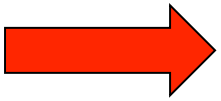
- Os exercícios era independentes uns dos outros.
- O único ponto de acoplamento era o **`main()`** único
- **Acoplamento** significa o quanto uma classe depende da outra para funcionar. E quanto maior for esta dependência entre ambas, dizemos que estas classes elas estão fortemente acopladas. **Devemos sempre diminuir o acoplamento**
- **Coesão** está diz que uma classe deve ter apenas uma única responsabilidade e realizá-la de maneira satisfatória, ou seja, uma classe não deve assumir responsabilidades que não são suas . **Devemos sempre aumentar a coesão.**

Prática 1
Estratégia “Uma Classezona”

01, 02, 03, 04, 05,
06, 07, 08, 09, 10

Prática 1
Estratégia “Uma Classezona”

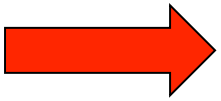
03, 04, 05, 06, 07, 08, 09, 10



01, 02

Prática 1
Estratégia “Uma Classezona”

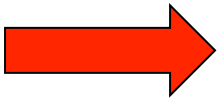
04, 05, 07, 08, 09, 10



01, 02, 03, 06

Prática 1
Estratégia “Uma Classezona”

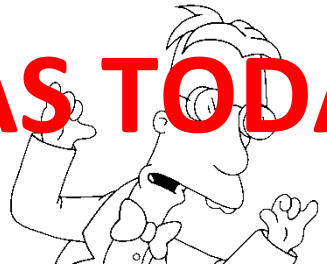
04, 05, 07, 08, 09, 10

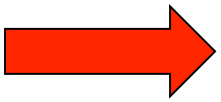


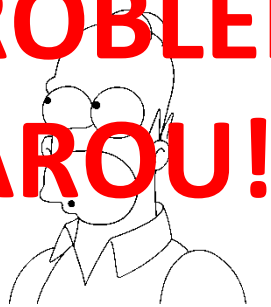
01, 02, 03, 0

Prática 1
Estratégia "Uma Classezona"

APENAS UM MÓDULO EU PROBLEMA MAS TODA A APLICAÇÃO PAROU!







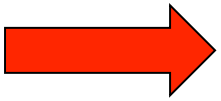
04, 05, 07, 08, 09, 10

01, 02, 03, 0

Prática 1

Estratégia "Uma Classezona"

PARA FAZER A MANUTENÇÃO TODA A APLICAÇÃO SERÁ TIRADA DE PRODUÇÃO!!

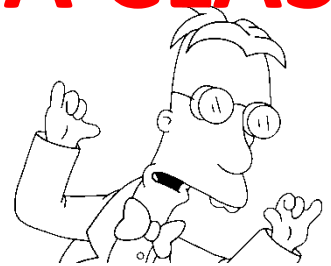


04, 05, 07, 08, 09, 10

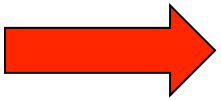
01, 02, 03, 0

Prática 1
Estratégia "Uma Classezona"

UMA CLASSEZONA = BAIXA COESÃO!!



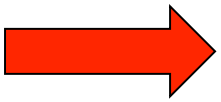
04, 05, 07, 08, 09, 10



01, 02, 03, 0

Prática 1
Estratégia “Cada exercício, uma classe”

Programa

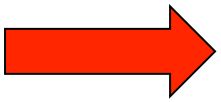


Prática 1
Estratégia “Cada exercício, uma classe”

01	02	03	04	05
06	07	08	09	10

Prática 1
Estratégia “Cada exercício, uma classe”

		03	04	05
06	07	08	09	10



01	02

Prática 1
Estratégia “Cada exercício, uma classe”

		04	05
07	08	09	10

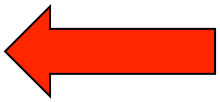
01	02	03
06		

Prática 1
Estratégia “Cada exercício, uma classe”

		04	05
07	08	09	10

01	02	03
✖		

Prática 1
Estratégia “Cada exercício, uma classe”

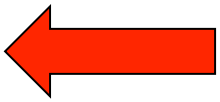


		04	05
07	08	09	10

06

01	02	03

Prática 1
Estratégia "Cada exercício, uma classe"



		04	05
07	08	09	10

06

01	02	03

O PROBLEMA FICA ISOLADO EM UMA CLASSE APENAS

Prática 1
Estratégia “Cada exercício, uma classe”

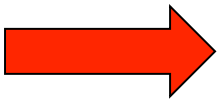
		04	05
07	08	09	10

06.
1

01	02	03

Prática 1
Estratégia “Cada exercício, uma classe”

		04	05
07	08	09	10



01	02	03
06.1		

Prática 1
Estratégia "Cada exercício, uma classe"

		04	05
07	08	09	10

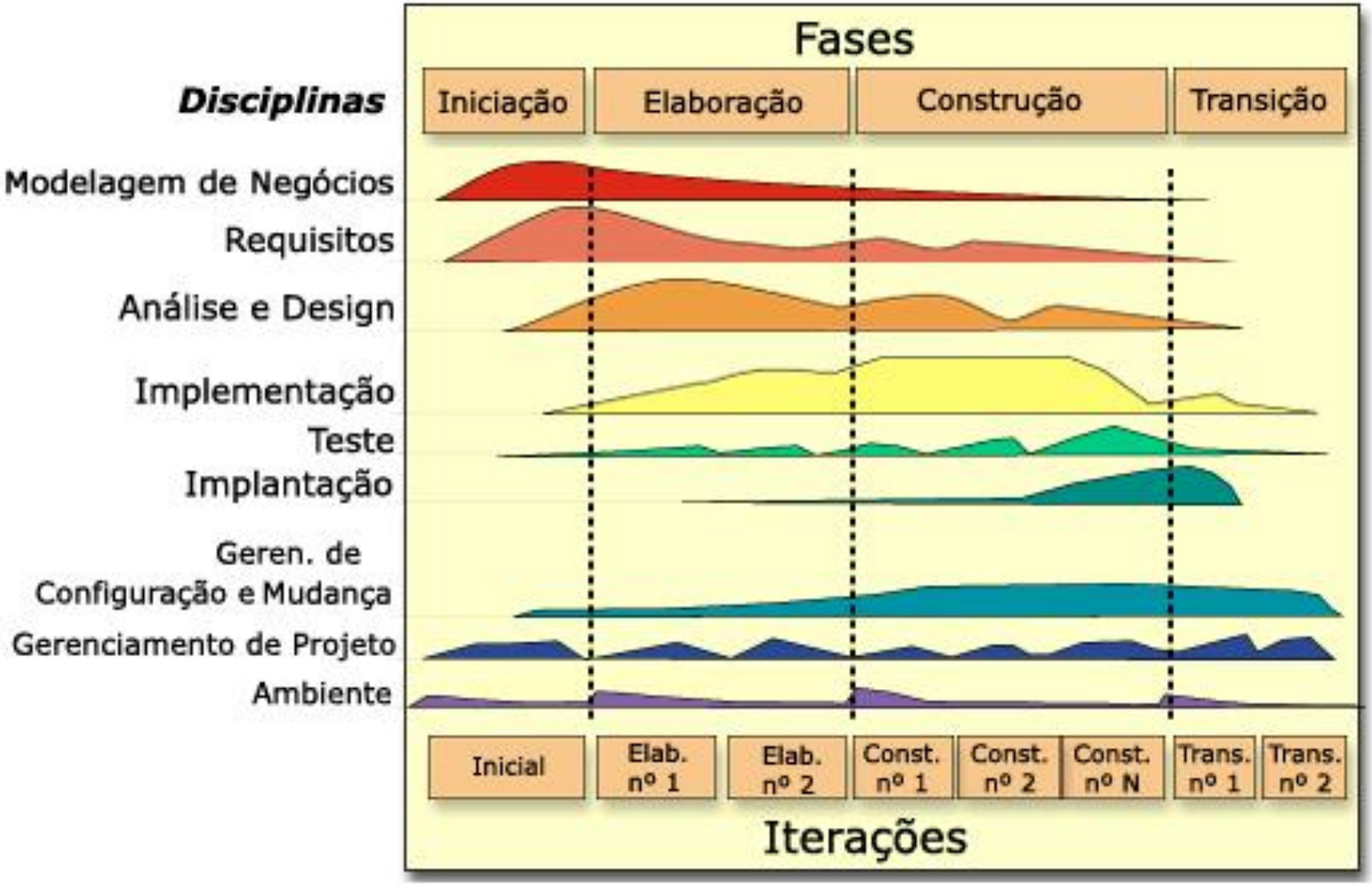
01	02	03
06.1		

ALTA COESÃO

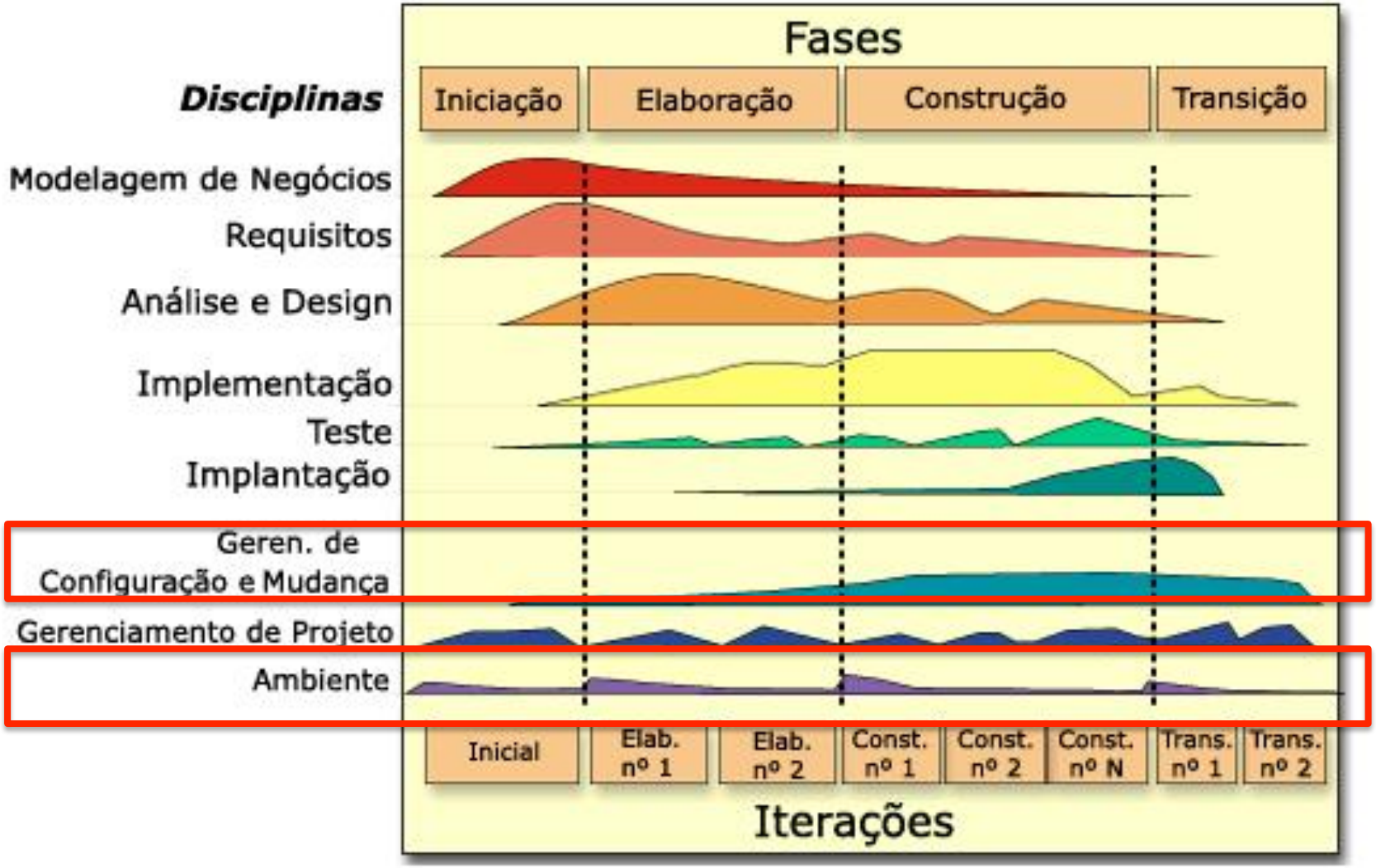
BAIXO ACOPLAMENTO

GCS e o processo de software

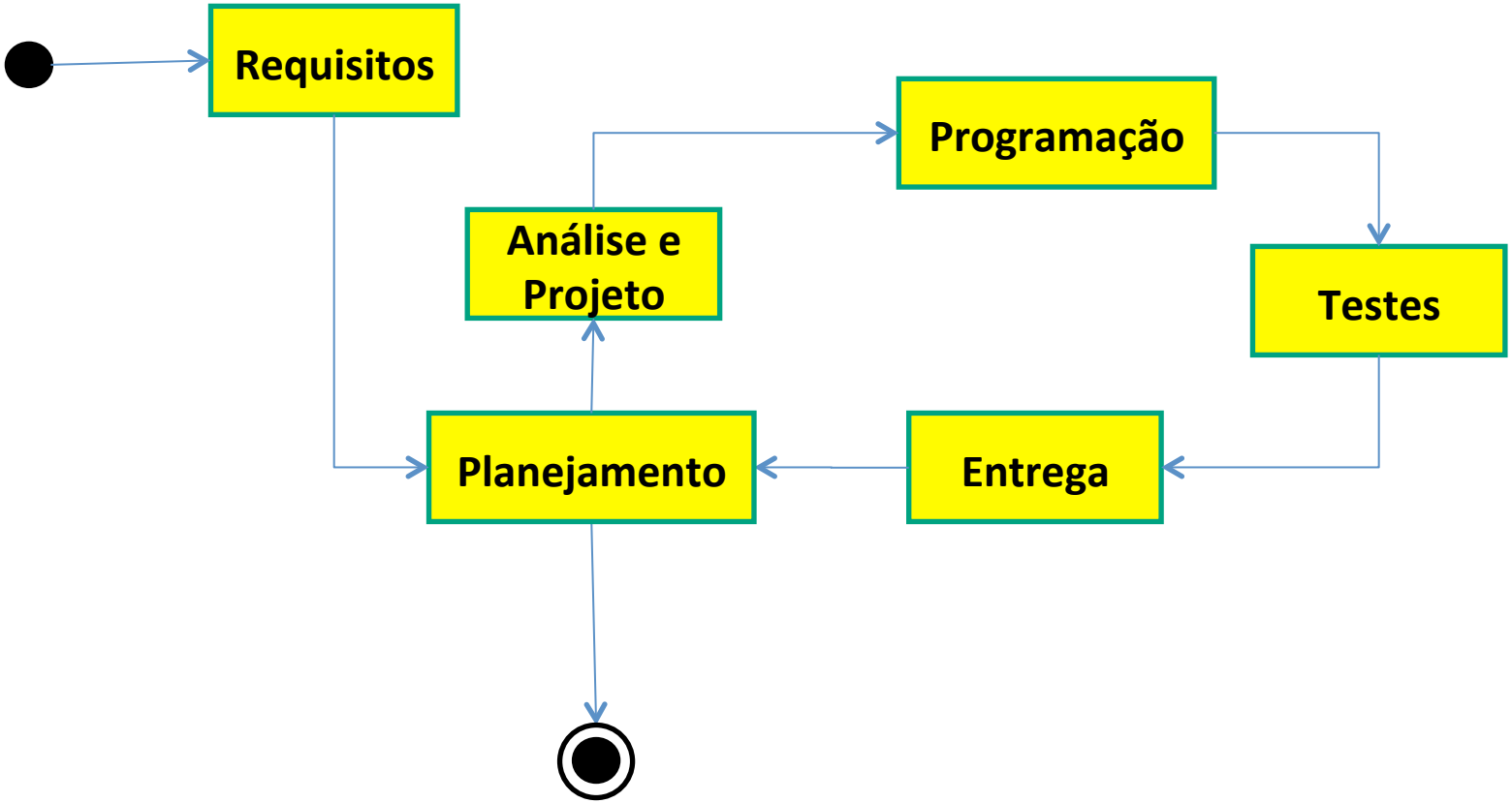
Rational Unified Process (RUP)



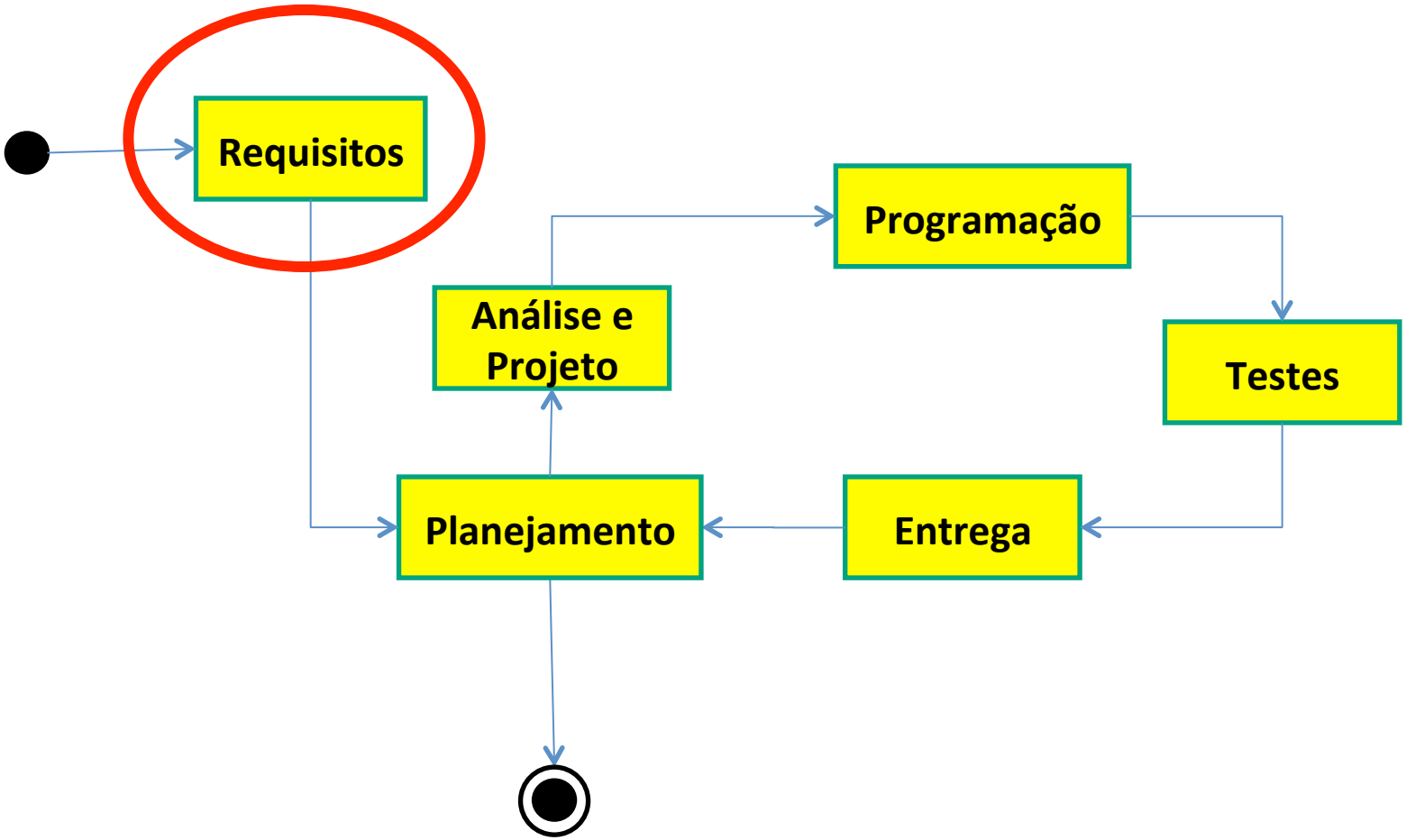
GCS no RUP


Fases
Disciplinas
Iniciação
Elaboração
Construção
Transição
Modelagem de Negócios
Requisitos
Análise e Design
Implementação
Teste
Implantação
Geren. de Configuração e Mudança
Gerenciamento de Projeto
Ambiente
Inicial
Elab. nº 1
Elab. nº 2
Const. nº 1
Const. nº 2
Const. nº N
Trans. nº 1
Trans. nº 2
Iterações

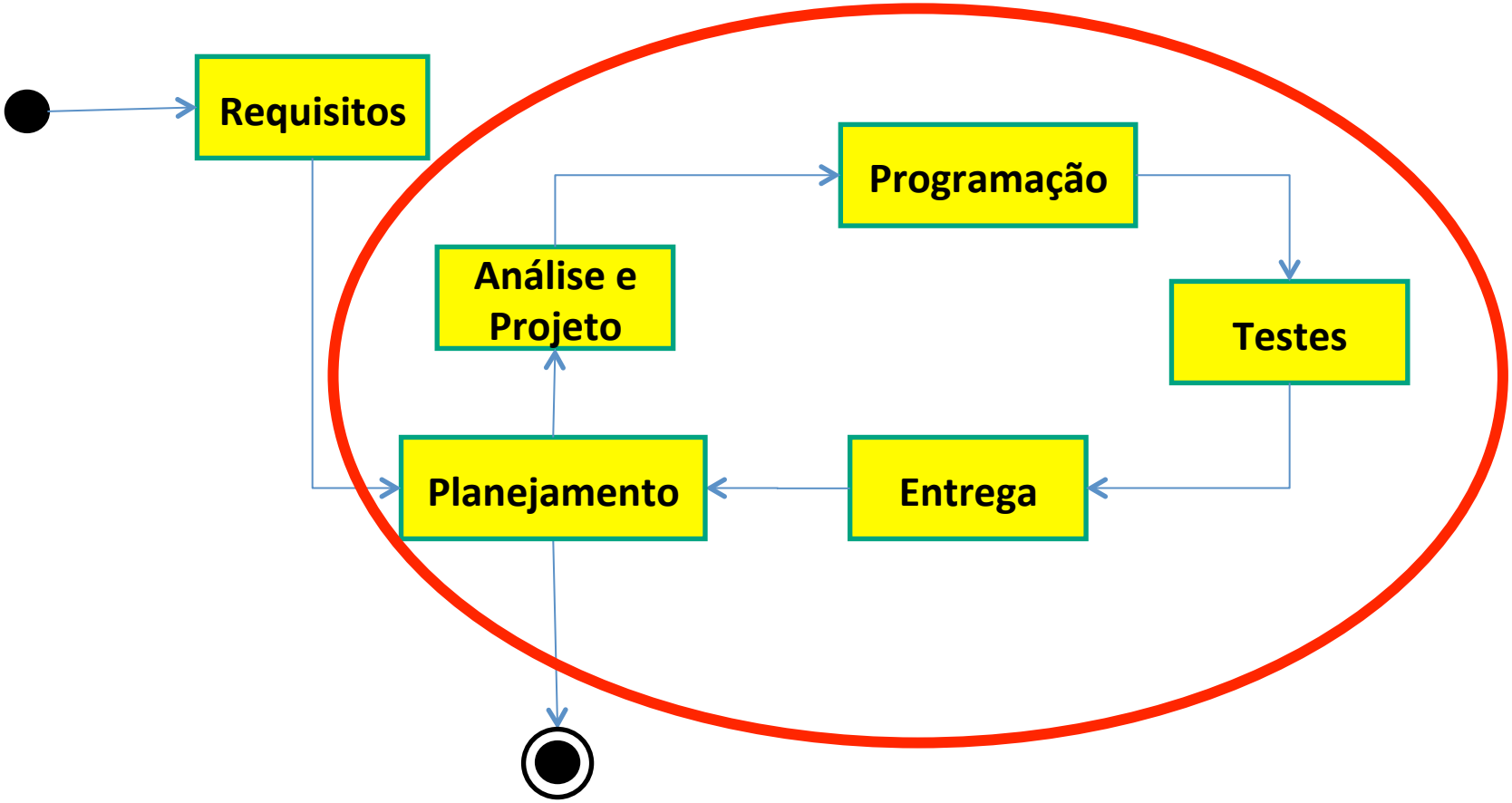
Processo UEAno
Desenvolvimento de TCCs



Processo UEAno



Processo UEAno



Cada iteração durará uma semana!

Processo UEAno

Processo UEAno

Quais artefatos (itens de configuração) cada disciplina cria?

Atividade prática 2

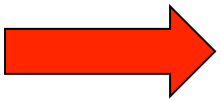
Atividade prática 2

- Grupo de 3 a 5 pessoas
- 30 min para resolver tudo.
- Defina um ambiente de desenvolvimento para Java (problemas simples Java SE)
- Utilize Powerpoint e Word
- Use esquemas e desenhos
- Defina um processo de desenvolvimento utilizando apenas os recursos (ambiente de desenvolvimento + pessoas) que tem em sala
  - Quais os passos?
  - Quais os perfis? Quem faz o que?
  - Quais os artefatos (itens de configuração) utilizados em seu processo?
- Nomeie e defenda seu processo para os outros alunos

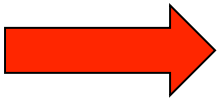
Versionamento

Versionamento

Programa



Versionamento



01	02	03	04	05
11	12	13	14	15
21	22	23	24	25
31	32	33	34	35
41	42	43	44	45

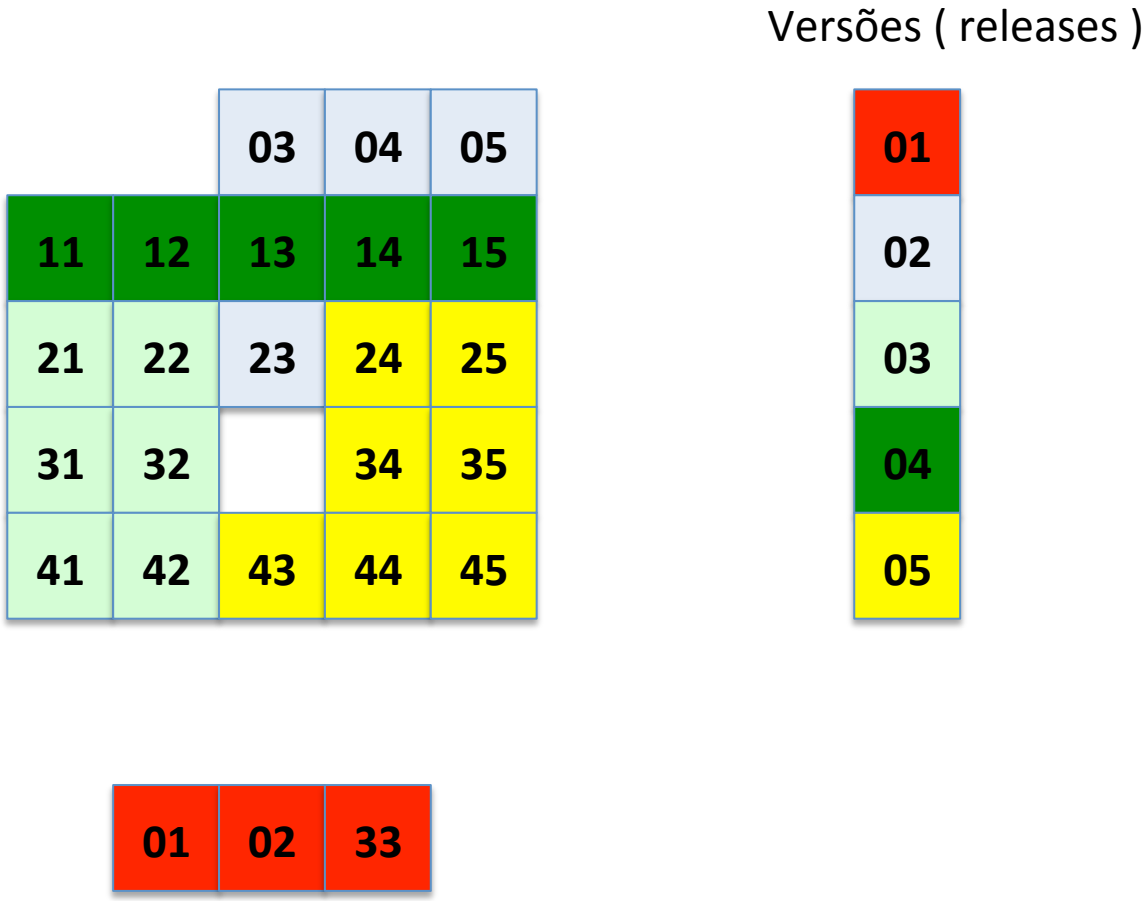
Versionamento

01	02	03	04	05
11	12	13	14	15
21	22	23	24	25
31	32	33	34	35
41	42	43	44	45

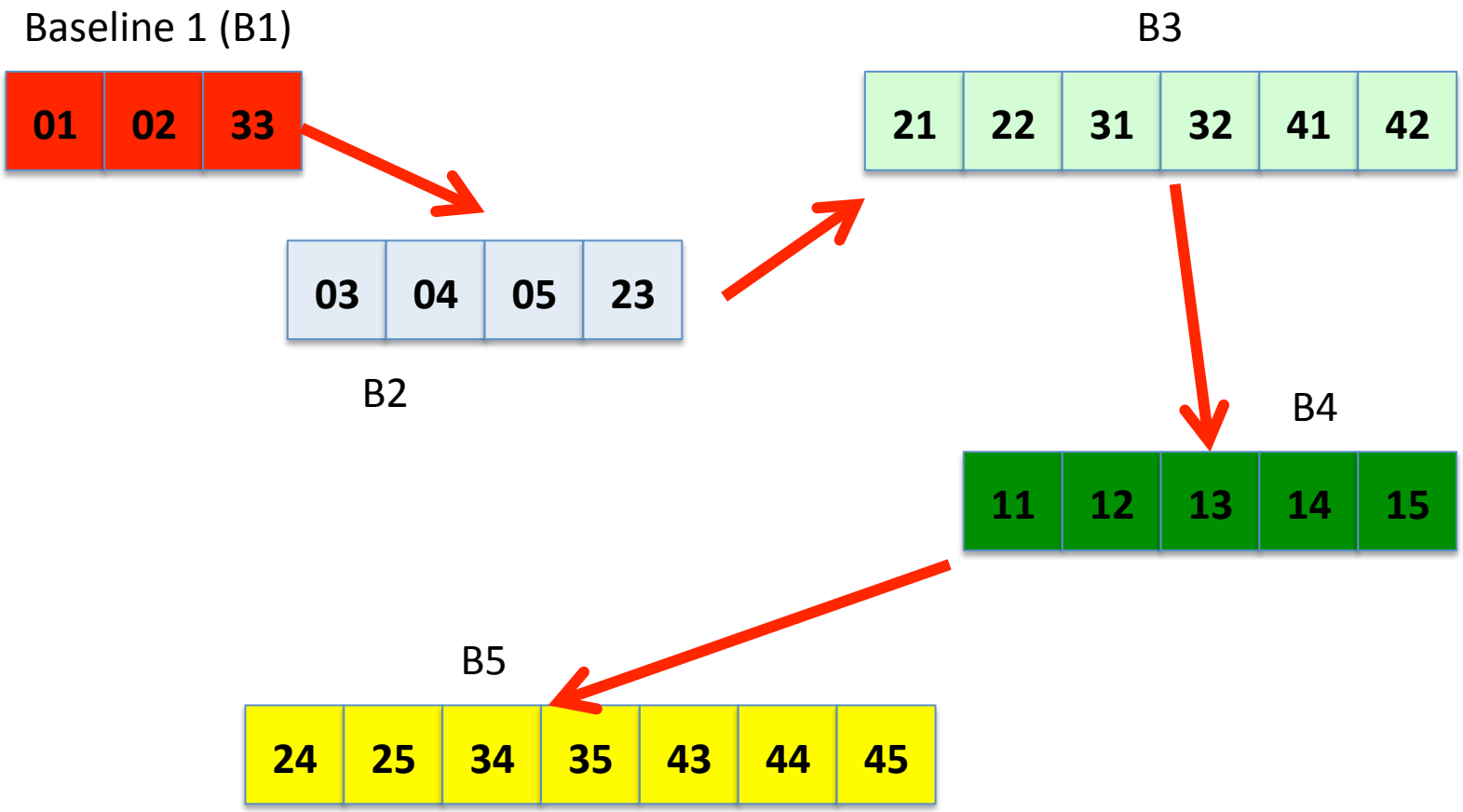
Versões (baselines)

01
02
03
04
05

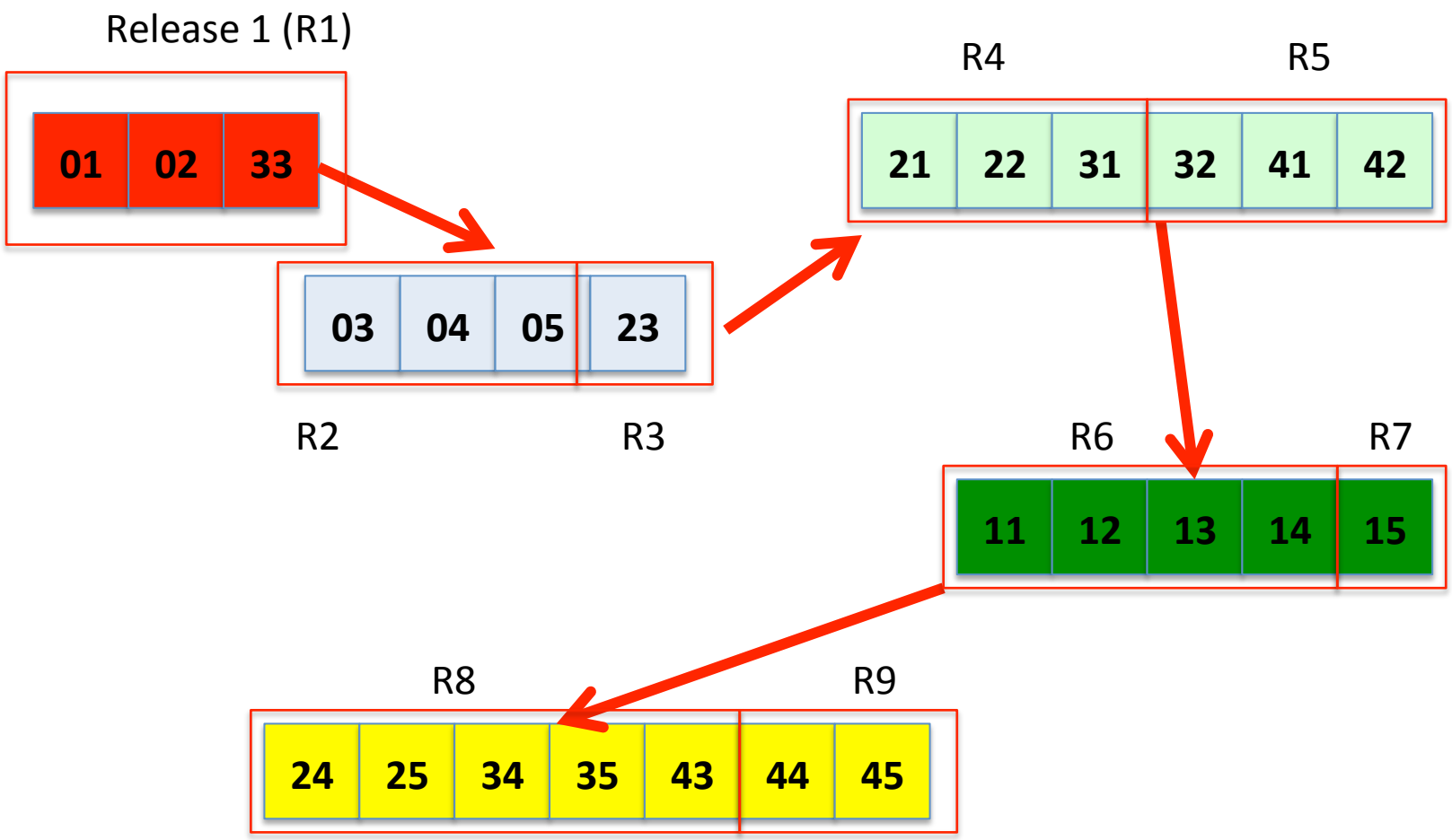
Versionamento



Versionamento



Versionamento



Como escolher a funcionalidade prioritária?

1. Maior risco técnico
2. Selecionado pelo cliente
3. Selecionado pelo gerente para agregar mais valor
4. Caminho crítico
5. Selecionado pela equipe

Nenhum comando "mágico" do controle de versão irá salvar um planejamento ruim!

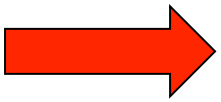
Como escolher a funcionalidade prioritária?

1. Quantos membros sua equipe possui?
2. O quanto é possível paralelizar?

Nenhum comando "mágico" do controle de versão irá salvar um planejamento ruim!

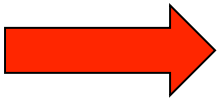
Versionamento

		03	04	05
11	12	13	14	15
21	22	23	24	25
31	32		34	35
41	42	43	44	45



01	02	33

Versionamento

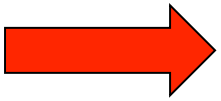


11	12	13	14	15
21	22	23	24	25
31	32		34	35
41	42	43	44	45

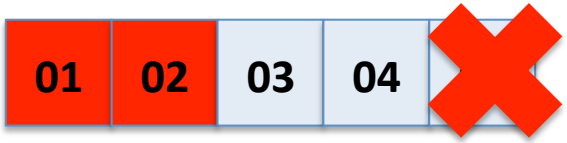
01	02	03	04	05

33

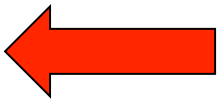
Versionamento



11	12	13	14	15
21	22	23	24	25
31	32		34	35
41	42	43	44	45



Versionamento



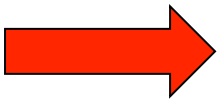
05

11	12	13	14	15
21	22	23	24	25
31	32		34	35
41	42	43	44	45

01	02	03	04

33

Versionamento



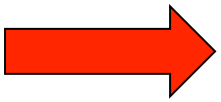
11	12	13	14	15
21	22	23	24	25
31	32		34	35
41	42	43	44	45

05.
1

01	02	03	04

33

Versionamento



11	12	13	14	15
21	22	23	24	25
31	32		34	35
41	42	43	44	45

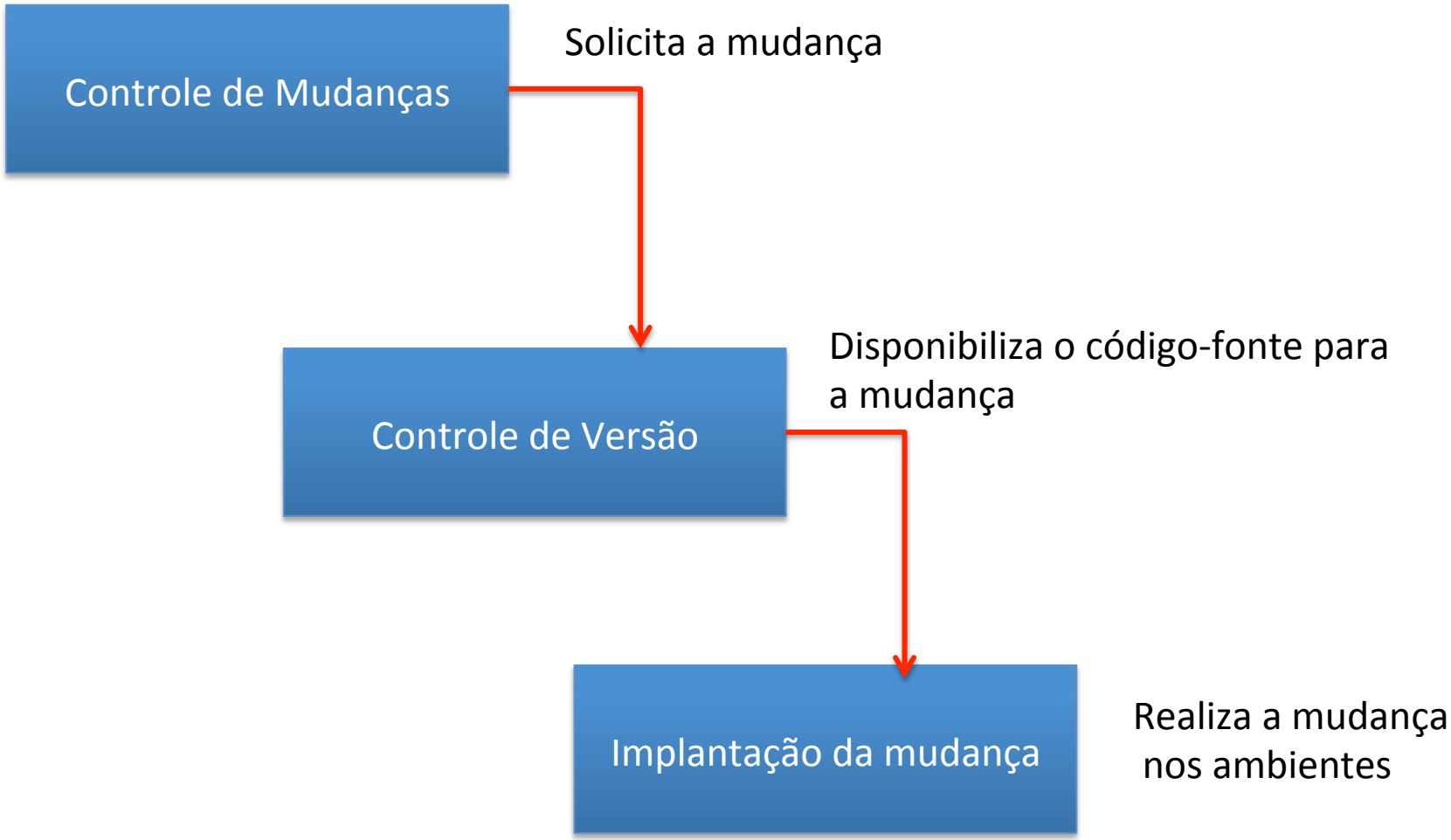
05.
1

01	02	03	04

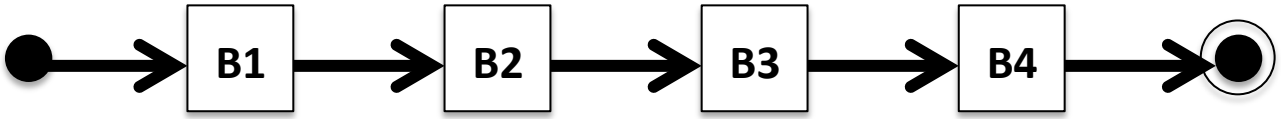
33

A nova versão irá provocar erros no usuário?
A nova versão irá provocar erros nos futuros módulos?

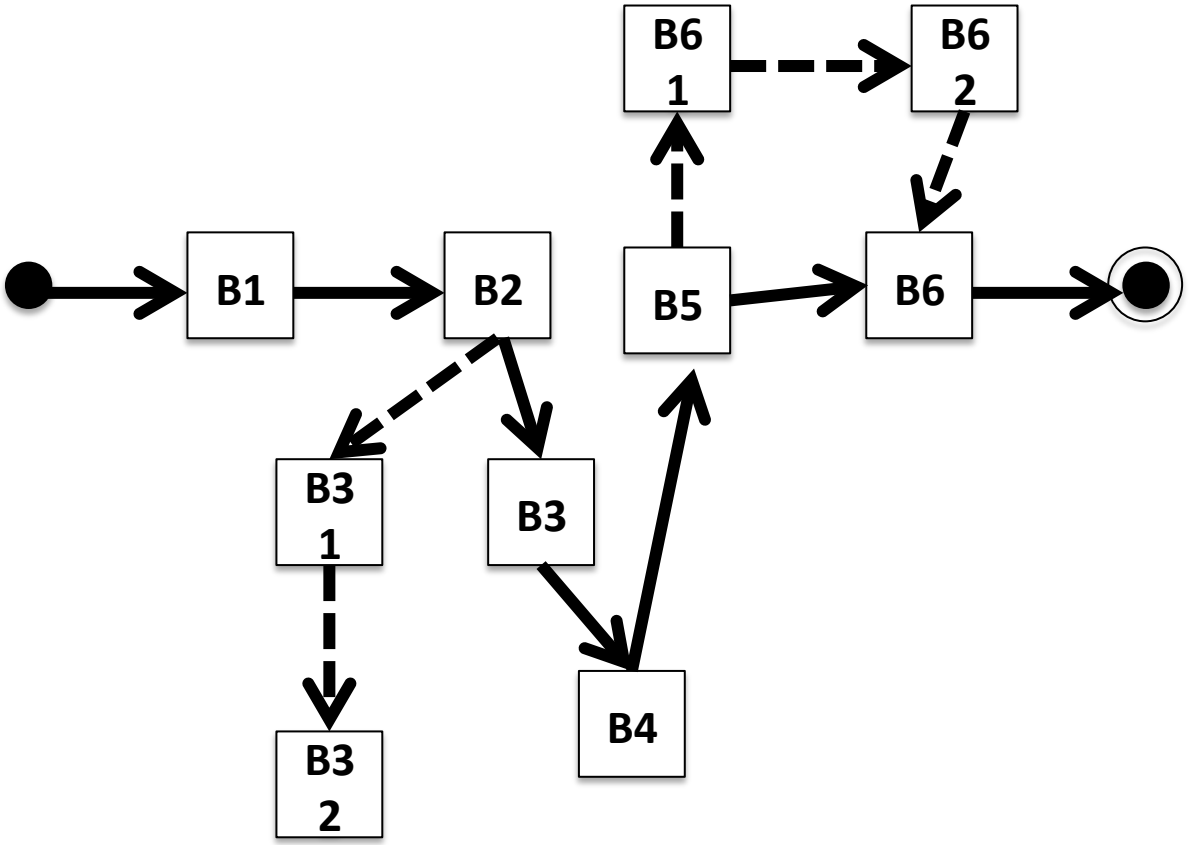
Ferramentas GCS



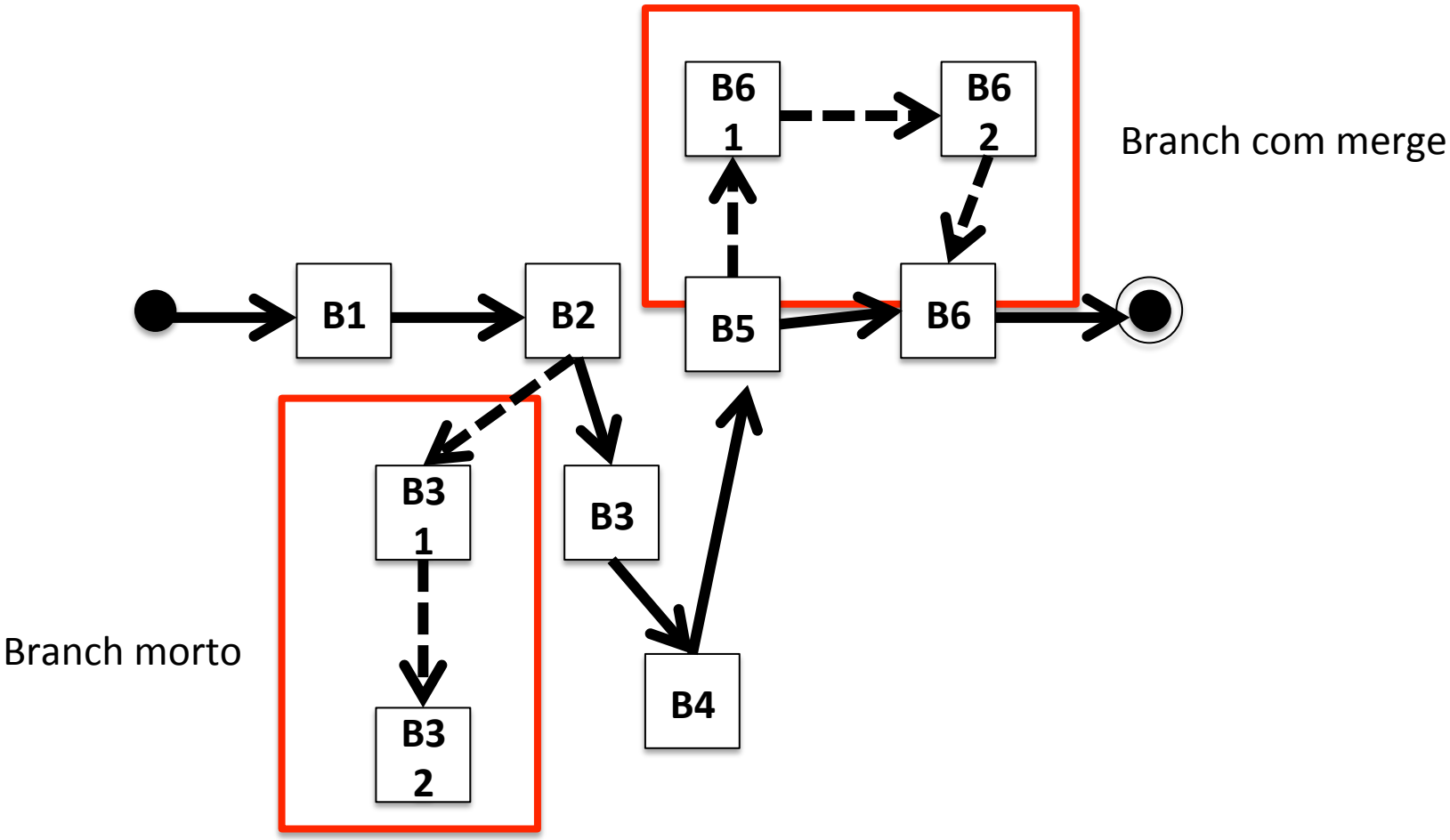
Mainline (planejada)



Mainline (executada)



Mainline (executada)



Atividade prática 3

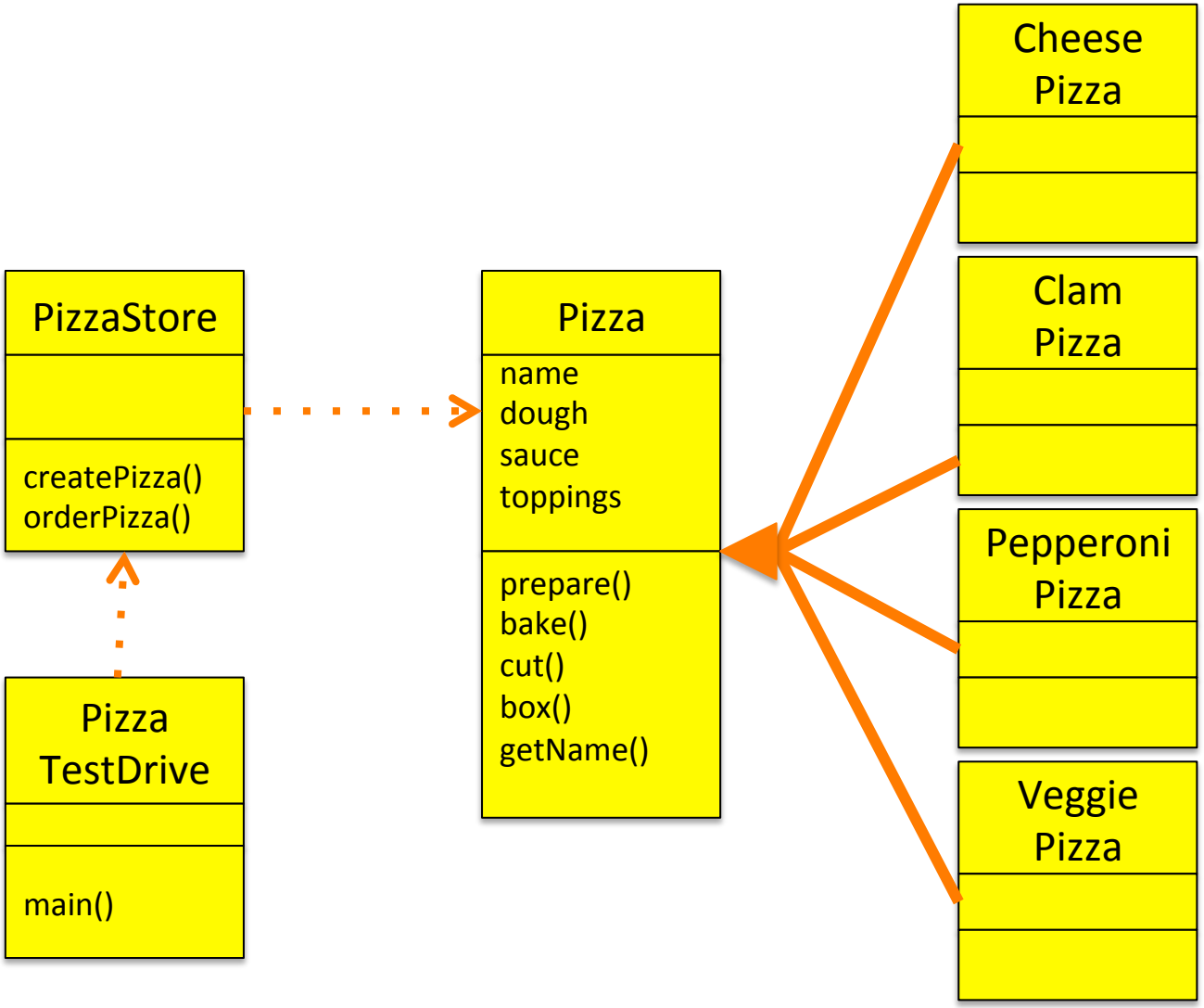
Atividade prática 3

- Criar uma aplicação PizzaStore para gerenciar uma pizzaria
- A pizzaria vende pizzas tipo Pepperoni, Veggie, Clam e Cheese
- Para rodar a aplicação, crie uma classe PizzaTestDrive

Atividade prática 3

- Use o processo desenvolvido na Prática 4
- Defina as *baselines*
- Defina as *releases*
- Defina a padronização de código
- Crie uma projeto chamado **pizza**
- Vamos simular um dia de trabalho! A cada **6 min**, cada equipe tem de entregar uma versão **compilável** da aplicação

Atividade prática 3

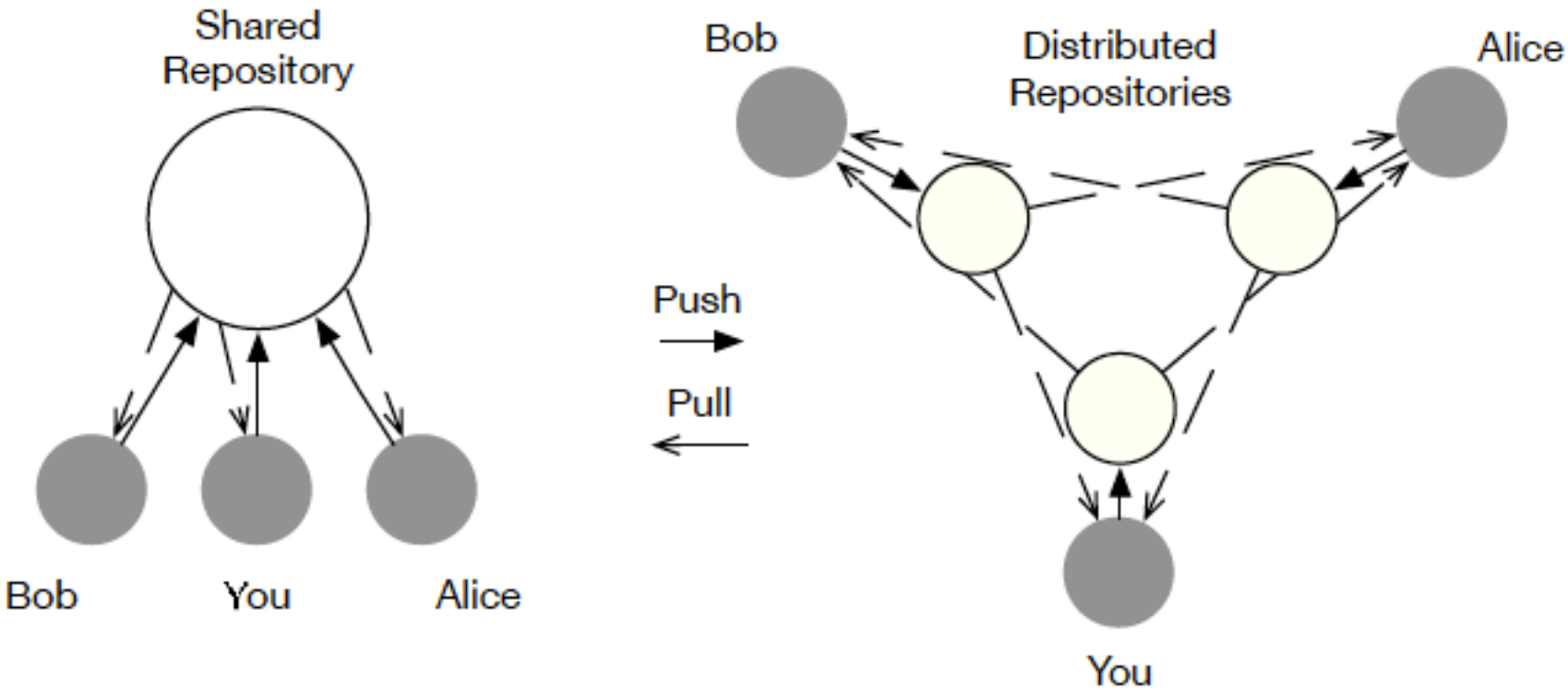


Introdução ao Git

GIT

- Sistema de controle de versão (SCV) distribuído (http://git-scm.com/)
- Software Livre criado por Linus Towards para manutenção e gerenciamento do código-fonte do Kernel do Linux (http://www.youtube.com/watch?v=4XpnKHJAok8)
- Quem usa
  - Microsoft, Facebook, Microsoft, Twitter, Linkedin, Netflix, Erlang...
- Simples, fácil de usar e bom desempenho
- **GIT não é mágico ou médico! Ele não vai consertar maus projetos e arquiteturas ruins!!**

SCV distribuído



GITHUB

GitHub

Explore GitHub | Search | Features | Blog | Sign up for free | Sign in

Features

Project Management | Code Hosting | Community

Every repository on GitHub comes with the tools needed to manage your project. Open to the community for public projects – secured for private projects.

Collaboration
manage large teams with ease

Git Powered Wikis
clone, push, branch and tag it

Integrated Issue Tracking
bug tracking tied to commits

Code Review
living discussions about code

Instalando o GIT e GITHUB

- **GIT**
  - http://git-scm.com/downloads
- **GITHUB**
  - https://help.github.com/articles/set-up-git

Configurando o GIT

- % git config --global user.name ”Jucimar Jr"
- % git config --global user.email ”jucimar.jr@gmail.com"
- % git config --global color.ui auto

Criando repositório

- **Em uma pasta nova**
  - `% mkdir /src/projeto`
  - `% cd /src/projeto`
  - `% git init`
- **Em uma pasta já existente**
  - `% cd /src/projetojaexistente`
  - `% git init`
  - `% git add .`
  - `% git commit –m “the game has begun”`
- **Se não compartilhar**
  - `...`
  - `% git init --bare`
  - `...`

Clonando repositório

- **Clonando na rede local**
  - `% cd src/`
  - `% git clone git@192.0.0.1:projeto`

- **Clonando do github.com**
  - `% cd src/`
  - `% git clone git://github.com/jucimarjr/projeto.git`

Check-in e check-out

- **Baixando as modificações do repositório**
  - `% cd pastarepositorio/`
  - `% git pull`

- **Subindo as modificações para o repositório**
  - `% cd pastarepositorio/`
  - `% git push`

Trabalhando com GIT

- **Verificando a situação**
  - `% git status`
- **Renomeando/movendo arquivos**
  - `% git mv arquivo1.txt arquivo2.txt`
- **Apagando arquivos/pastas**
  - `% git rm arquivo1.txt`
  - `% git rm –r pasta/`
- **Commitando**
  - `% git add .`
  - `% git commit –m “mensagem explicativa”`
- **Resolvendo conflitos**
  - Abrir o arquivo marcado e consertar
  - `% git status`

GIT no dia-a-dia

1. Baixar as atualizações realizadas por outros desenvolvedores
2. Analisar o que foi alterado
3. Analisar o que será desenvolvido hoje de acordo com o planejamento
4. Pegar uma *user story* para codar
5. Codar, Codar, Codar como se não houvesse amanhã
6. Commitar o que foi codado
7. Voltar ao passo 3 até acabar a tarefas diárias
8. Subir os commits para o repositório central

GIT no dia-a-dia

1. Baixar as atualizações realizadas por outros desenvolvedores
   1. **% git pull**
2. Analisar o que foi alterado
3. Analisar o que será desenvolvido hoje de acordo com o planejamento
4. Pegar uma *user story* para codar
5. Codar, Codar, Codar como se não houvesse amanhã
6. Commitar o que foi codado
   1. **% git commit –m "mensagem"**
7. Voltar ao passo 3 até acabar a tarefas diárias
8. Subir os commits para o repositório central
   1. **% git push**

Atividade prática 4

Atividade prática 4

- Ampliar a aplicação PizzaStore (modificar o código existente)
- Uma franquia em Chicago e outra em Nova York
- Cada uma delas faz a pizza “do seu jeito” do seu jeito. Mas devem obedecer um método geral
- Dica: Usar *Factory Method*

Atividade prática 4

- Definir o ambiente de desenvolvimento, teste e produção
  - Desenvolvimento -> Teste -> Produção
- Definir o ambiente de teste
  - **Dica**: como faço para instalar o programa no ambiente de teste? Como faço para levar o que passou no teste para produção? O que fazer com o que não passar no teste?
- Definir *baselines* e *releases*

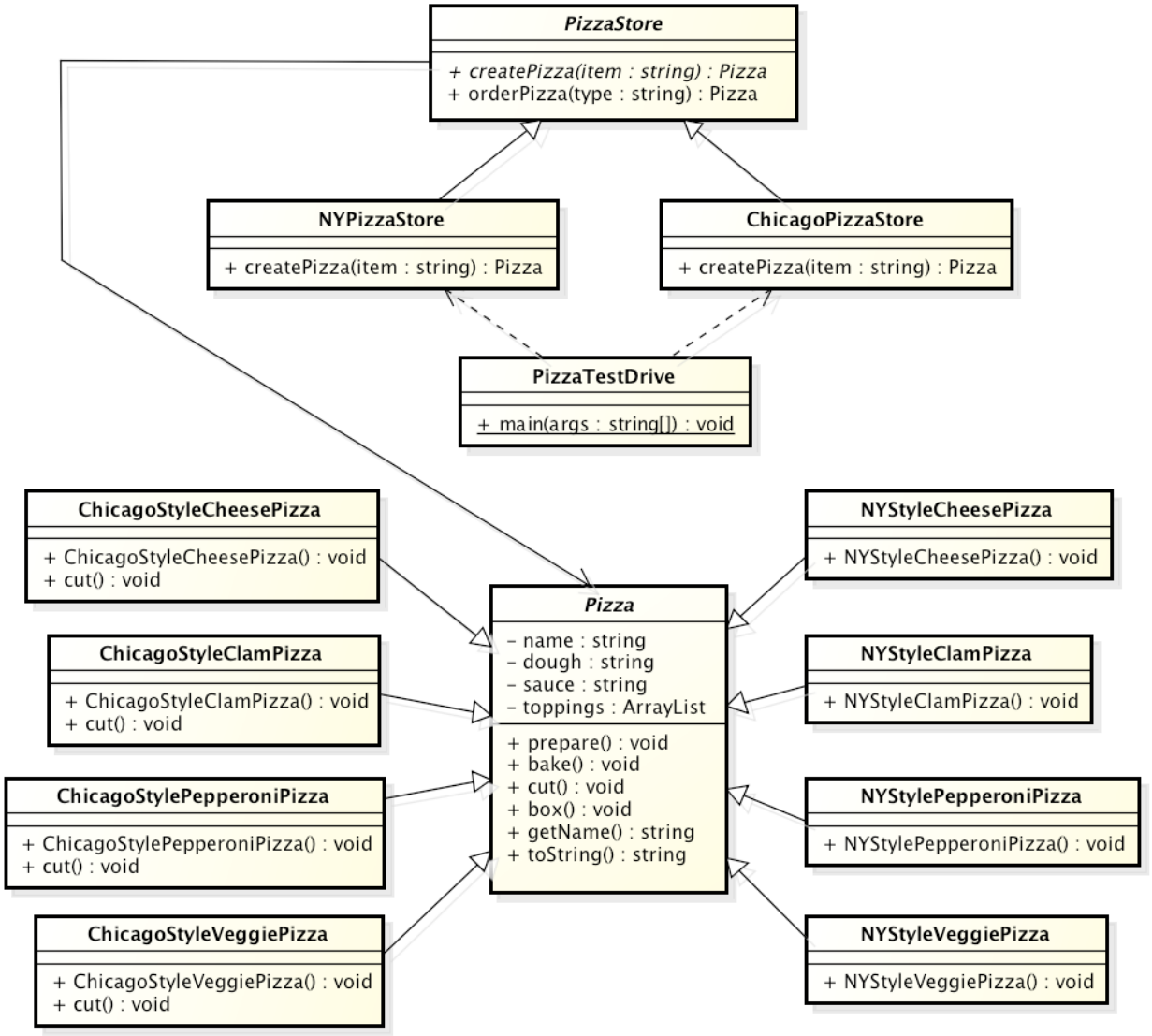
Atividade prática 4

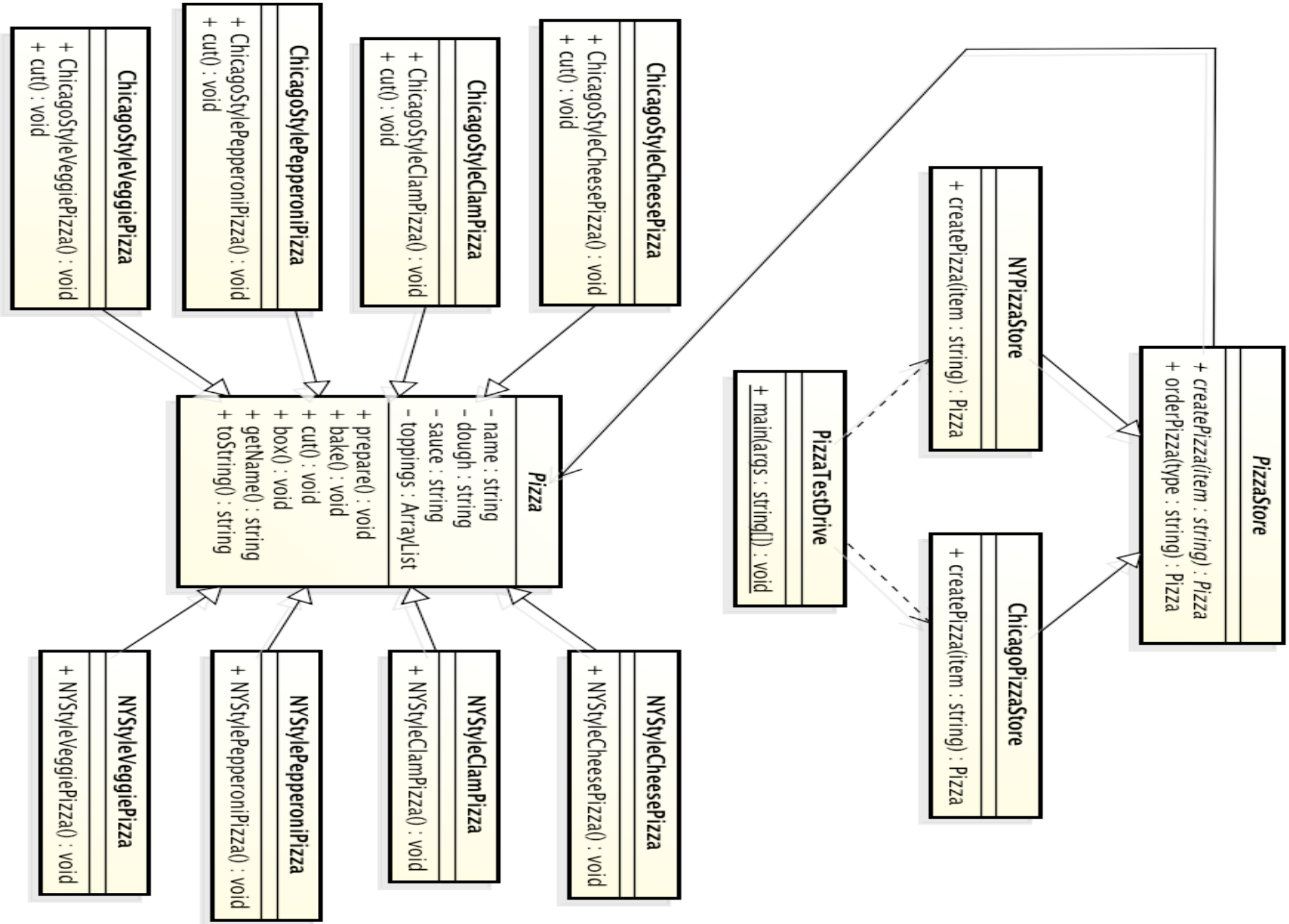
- Configurar o git na sua máquina
- Criar um usuário no GitHub.Com
  - Ícone https://pt.gravatar.com/
- Criar um repositório no github: **`pizza`**
- Clonar para sua máquina (um clone por desenvolvedor)
- Colocar os arquivos da prática 2
- Subir os arquivos
- Verificar o status usando o github local
- Codificar a nova atividade
  - Dias de 6 min
  - Máximo 1h

Começando no GITHUB

- git clone https://github.com/usuario/pizza.git
- cd /pizza
- Copiar os arquivos pra pasta pizza
- git init
- git add .
- git commit –m “the game has begun”
- git remote add origin https://github.com/usuario/pizza.git
- git push –u origin master

Atividade prática 3




PizzaStore
+ createPizza(item : string) : Pizza
+ orderPizza(type : string) : Pizza
NYPizzaStore
+ createPizza(item : string) : Pizza
ChicagoPizzaStore
+ createPizza(item : string) : Pizza
PizzaTestDrive
+ main(args : string[]) : void
ChicagoStyleCheesePizza
+ ChicagoStyleCheesePizza() : void
+ cut() : void
ChicagoStyleClamPizza
+ ChicagoStyleClamPizza() : void
+ cut() : void
ChicagoStylePepperoniPizza
+ ChicagoStylePepperoniPizza() : void
+ cut() : void
ChicagoStyleVeggiePizza
+ ChicagoStyleVeggiePizza() : void
+ cut() : void
Pizza
- name : string
- dough : string
- sauce : string
- toppings : ArrayList
+ prepare() : void
+ bake() : void
+ cut() : void
+ box() : void
+ getName() : string
+ toString() : string
NYStyleCheesePizza
+ NYStyleCheesePizza() : void
NYStyleClamPizza
+ NYStyleClamPizza() : void
NYStylePepperoniPizza
+ NYStylePepperoniPizza() : void
NYStyleVeggiePizza
+ NYStyleVeggiePizza() : void

Atividade prática 3

Trabalhando com GIT

- **Vendo o que foi mudado**
  - `% git status`
- **Ignorando arquivos**
  - Abrir o arquivo .gitignore
  - Acrescentar arquivos que não devem subir pro repositório
- **Descommitando**
  - `% git reset HEAD – arquivo.txt`

Branches

- **Criando um branch**
  - `% git branch novo_branch`
- **Vendo os branches disponíveis**
  - `% git branch`
- **Vendo os branches remotos disponíveis**
  - `% git branch -r`
- **Vendo todos os branches**
  - `% git branch –a`
- **Trocando de branch**
  - `% git checkout novo_branch`

Branches

- **Baixando um branch**
  - `% git pull origin novo_branch`
  - `% git checkout novo_branch`
- **Subindo novo branch**
  - `% git push origin novo_branch`
- **Fazendo merge de um branch**
  - `% git checkout master`
  - `% git merge novo_branch`

Atividade prática 5

Atividade prática 5

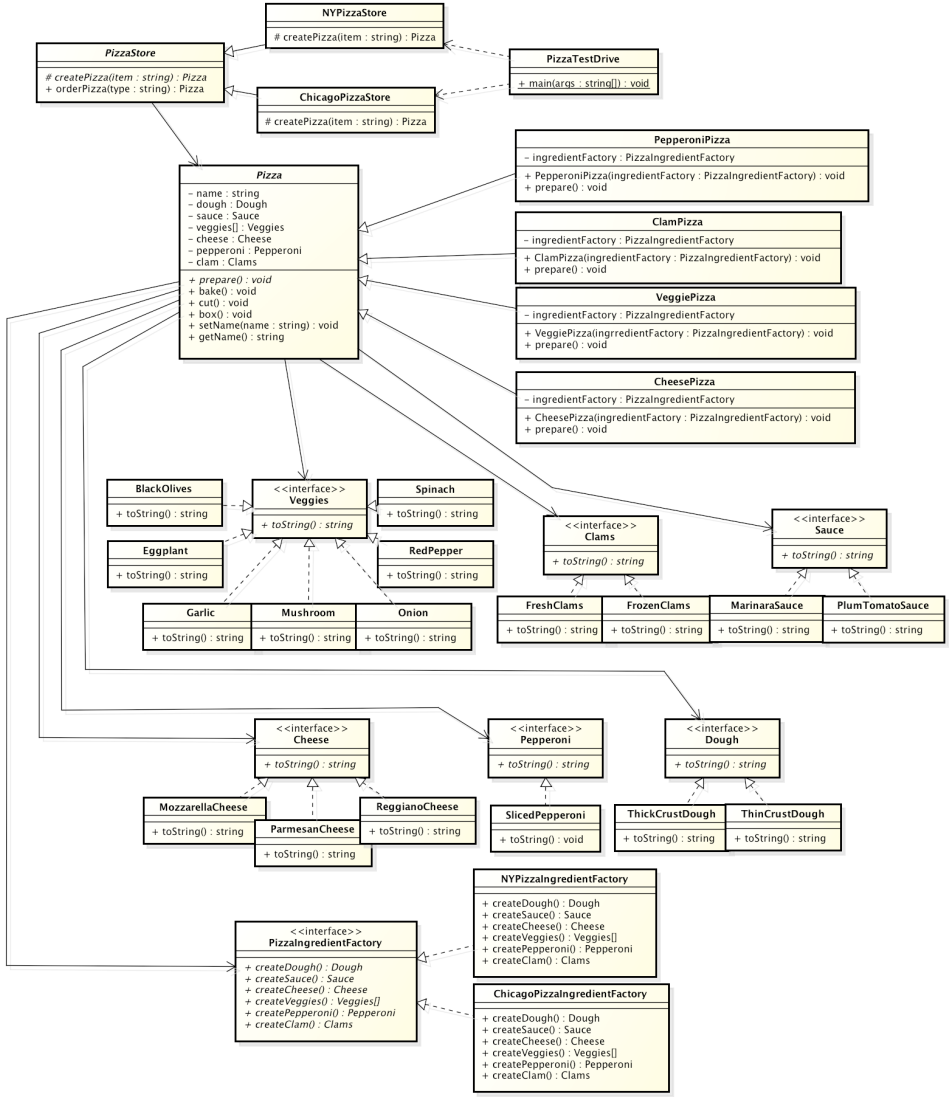
- Ampliar a aplicação PizzaStore (modificar o código existente)
- Devido ao estrondoso sucesso das pizzas-orientadas-a-objetos, verificou-se um crescimento muito grande da demanda
- Cresceu tanto que tornou-se mais barato produzir os ingredientes que comprá-los
- Modifique o código da aplicação para suportar a nova demanda

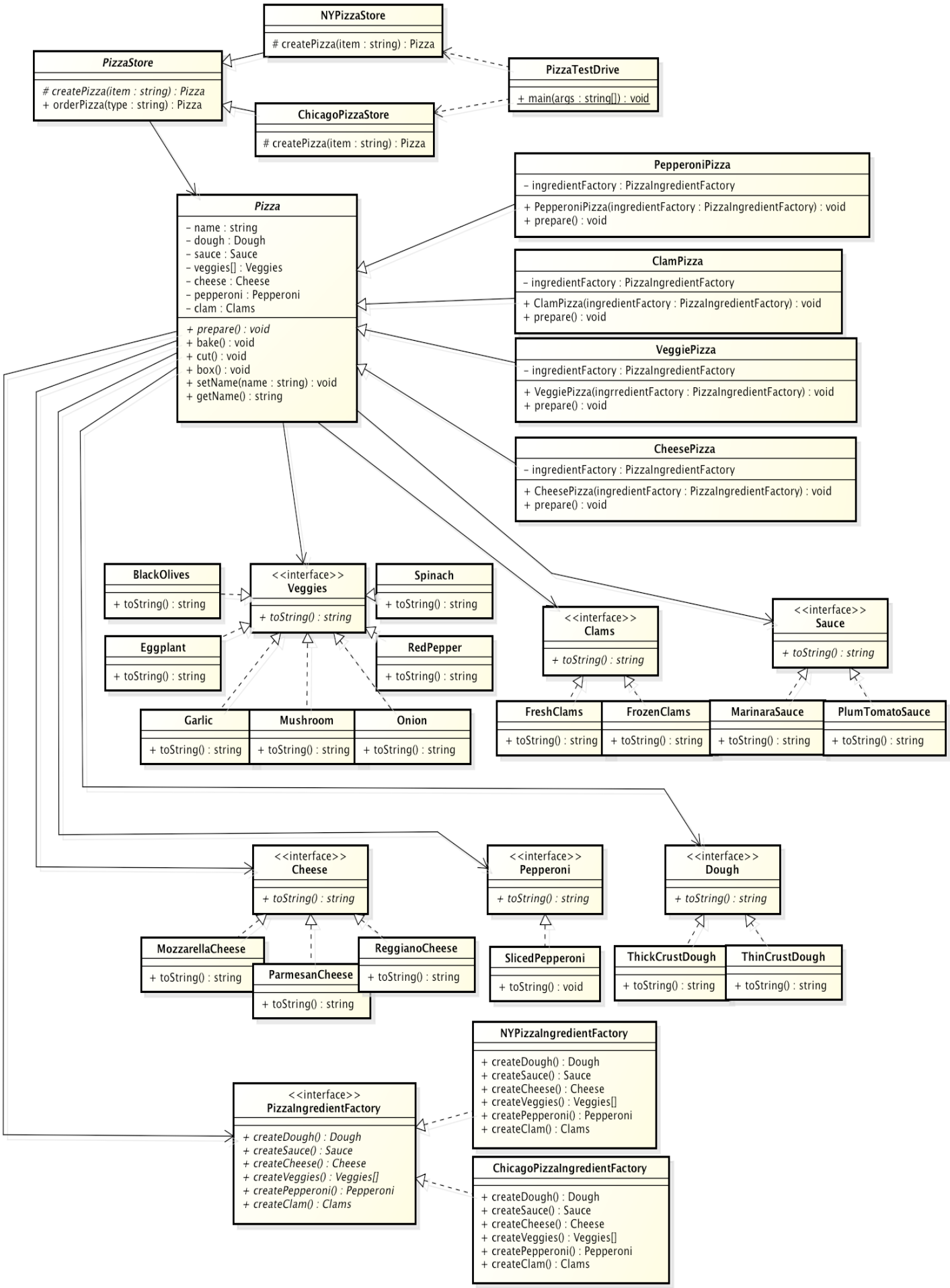
- Dica: Usar *Abstract Factory*

Atividade prática 5

- Definir o ambiente de desenvolvimento,teste e produção
- Definir as *baselines* e *releases*
- Criar um branch para cada *release*
- Fazer o *merge* de cada *release* com o *master*
- Codificar a nova atividade
  - Dias de 6 min
  - Máximo 1h

Atividade prática 5




NYPizzaStore
createPizza(item : string) : Pizza
PizzaStore
createPizza(item : string) : Pizza
+ orderPizza(type : string) : Pizza
PizzaTestDrive
+ main(args : string[]) : void
ChicagoPizzaStore
createPizza(item : string) : Pizza
PepperoniPizza
- ingredientFactory : PizzaIngredientFactory
+ PepperoniPizza(ingredientFactory : PizzaIngredientFactory) : void
+ prepare() : void
Pizza
- name : string
- dough : Dough
- sauce : Sauce
- veggies[] : Veggies
- cheese : Cheese
- pepperoni : Pepperoni
- clam : Clams
+ prepare() : void
+ bake() : void
+ cut() : void
+ box() : void
+ setName(name : string) : void
+ getName() : string
ClamPizza
- ingredientFactory : PizzaIngredientFactory
+ ClamPizza(ingredientFactory : PizzaIngredientFactory) : void
+ prepare() : void
VeggiePizza
- ingredientFactory : PizzaIngredientFactory
+ VeggiePizza(ingrredientFactory : PizzaIngredientFactory) : void
+ prepare() : void
CheesePizza
- ingredientFactory : PizzaIngredientFactory
+ CheesePizza(ingredientFactory : PizzaIngredientFactory) : void
+ prepare() : void
BlackOlives
+ toString() : string
<<interface>>
Veggies
+ toString() : string
Spinach
+ toString() : string
Eggplant
+ toString() : string
RedPepper
+ toString() : string
Garlic
+ toString() : string
Mushroom
+ toString() : string
Onion
+ toString() : string
<<interface>>
Clams
+ toString() : string
<<interface>>
Sauce
+ toString() : string
FreshClams
+ toString() : string
FrozenClams
+ toString() : string
MarinaraSauce
+ toString() : string
PlumTomatoSauce
+ toString() : string
<<interface>>
Cheese
+ toString() : string
<<interface>>
Pepperoni
+ toString() : string
<<interface>>
Dough
+ toString() : string
MozzarellaCheese
+ toString() : string
ParmesanCheese
+ toString() : string
ReggianoCheese
+ toString() : string
SlicedPepperoni
+ toString() : void
ThickCrustDough
+ toString() : string
ThinCrustDough
+ toString() : string
NYPizzaIngredientFactory
+ createDough() : Dough
+ createSauce() : Sauce
+ createCheese() : Cheese
+ createVeggies() : Veggies[]
+ createPepperoni() : Pepperoni
+ createClam() : Clams
<<interface>>
PizzaIngredientFactory
+ createDough() : Dough
+ createSauce() : Sauce
+ createCheese() : Cheese
+ createVeggies() : Veggies[]
+ createPepperoni() : Pepperoni
+ createClam() : Clams
ChicagoPizzaIngredientFactory
+ createDough() : Dough
+ createSauce() : Sauce
+ createCheese() : Cheese
+ createVeggies() : Veggies[]
+ createPepperoni() : Pepperoni
+ createClam() : Clams

Atividade prática 6

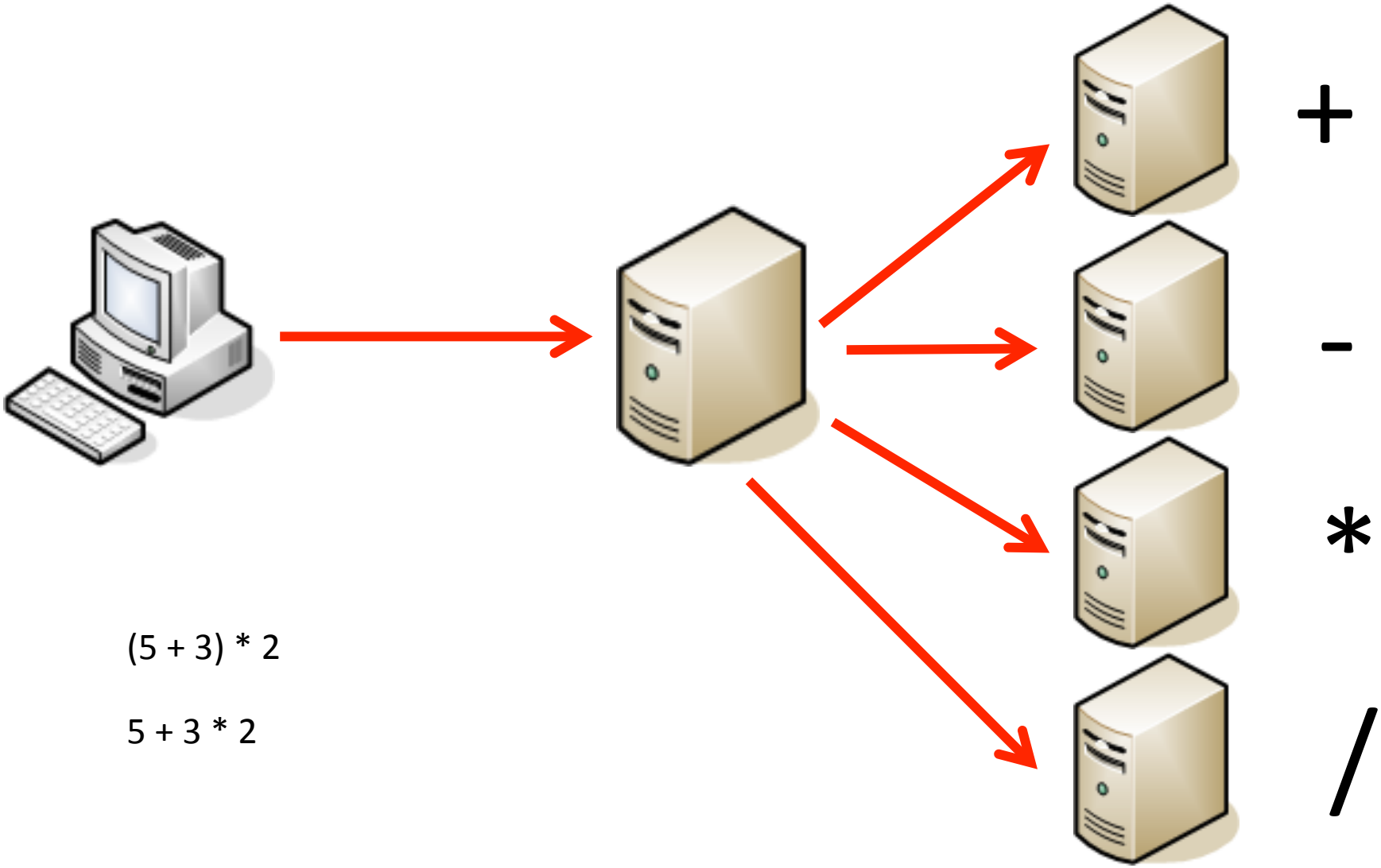
Projeto Final

Projeto Final

- Ambiente de desenvolvimento
- Ambiente de teste
- Ambiente de produção
- Planejamento do versionamento do projeto
- Programa pronto
- Artefatos de GCM gerados durante o desenvolvimento

- **BONUS**: automação do processo

Projeto Final
Calculadora Distribuída



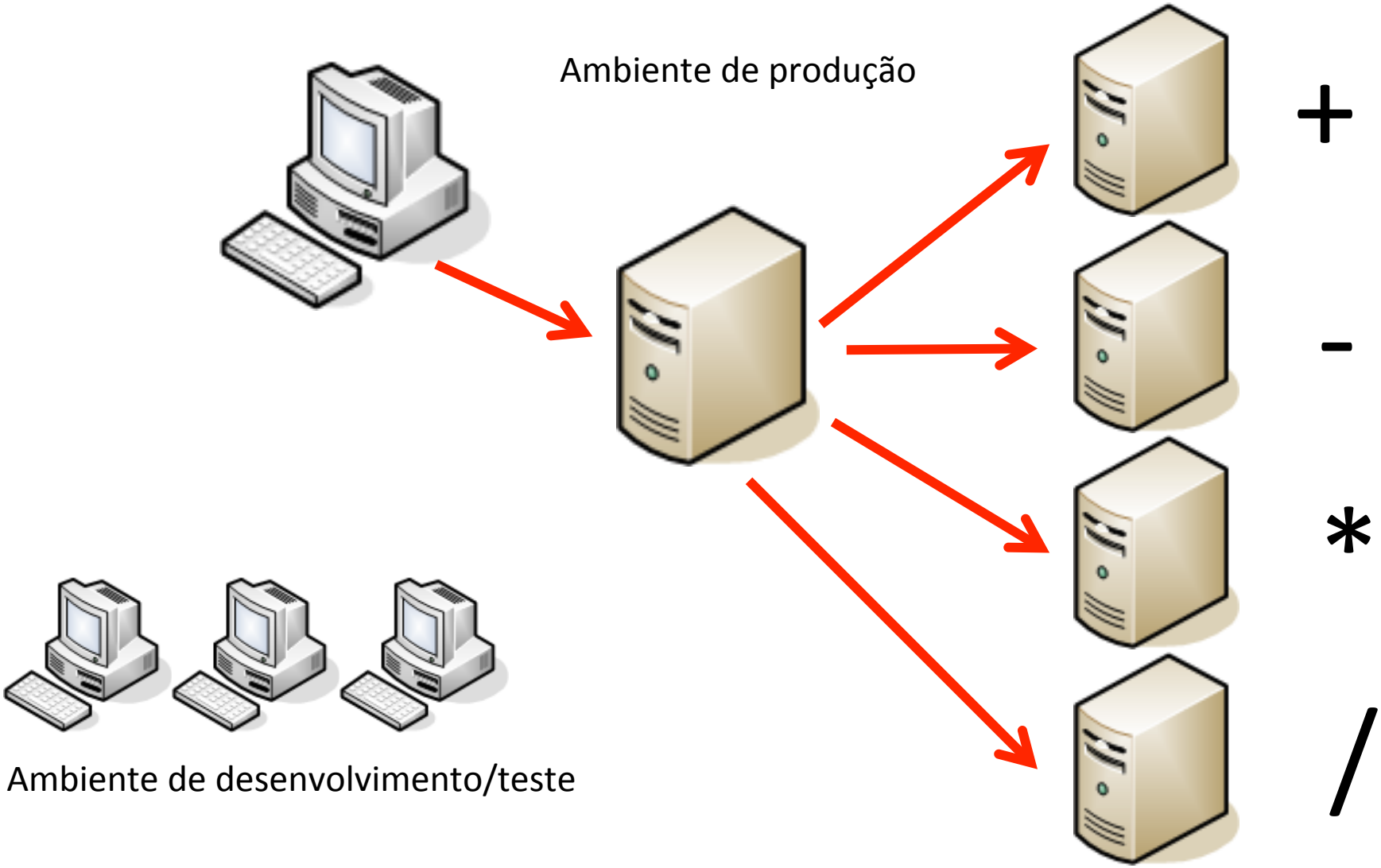
(5 + 3) * 2

5 + 3 * 2

Projeto Final
Calculadora Distribuída

http://www.ruby-lang.org/pt/

Projeto Final
Livros de Ruby

http://www.wowebook.com/search/?q=ruby

Calculadora Distribuída