Segurança de dados e Auditoria de Sistemas

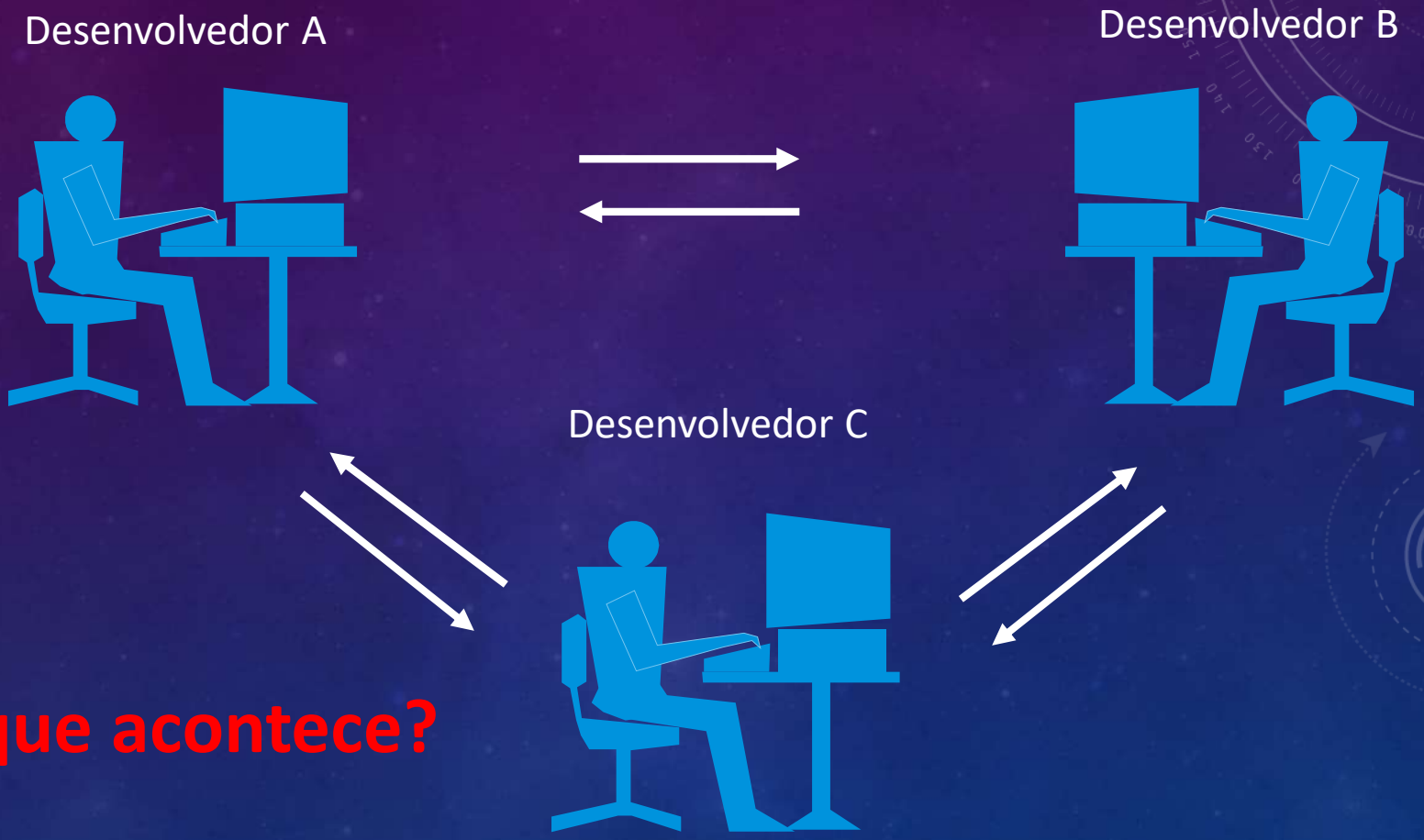
É possível desenvolver uma aplicação segura em uma ambiente não seguro?

CUIDADOS QUE TEMOS TER:

- Gerência de Configuração.
- Distribuição.
- Desenvolvimento.
- Documentação.
- Suporte ao ciclo de Vida.
- Testes de Segurança.
- Avaliação de Vulnerabilidades.

GERÊNCIA DE CONFIGURAÇÃO

PROBLEMA DA QUEBRA DE COMUNICAÇÃO



Porque acontece?

PROBLEMA DOS DADOS COMPARTILHADOS

Desenvolvedor A

Desenvolvedor B

Mexer no mesmo código!!

Programa de A

A1 A2 A3

Componente Compartilhado

Programa de B

B1 B2 B3

COMO RESOLVER O PROBLEMA DOS DADOS COMPARTILHADOS ?

PROBLEMA DOS DADOS COMPARTILHADOS - SOLUÇÃO SIMPLISTA

- **Solução simplista:**
  - cada desenvolvedor trabalha em uma cópia “local” do componente.
  - resolve o Problema dos Dados Compartilhados, mas cria um novo problema.

PROBLEMA DA MANUTENÇÃO MÚLTIPLA

PROBLEMA DA MANUTENÇÃO MÚLTIPLA

- Ocorre quando cada desenvolvedor trabalha com uma cópia “local” do que seria o mesmo componente.
- Dificuldade para saber:
  - Que funcionalidades foram implementadas em quais versões do componente.
  - Que defeitos foram corrigidos.
- Evitado através de uma biblioteca central de componentes compartilhados.
  - Nesse esquema, cada componente é copiado para a biblioteca sempre que alterado.

PROBLEMA DA ATUALIZAÇÃO SIMULTÂNEA

PROBLEMA DA ATUALIZAÇÃO SIMULTÂNEA – CENÁRIO 1

- O desenvolvedor A encontra e corrige um defeito em sua versão do componente compartilhado.
- Uma vez corrigido, o componente modificado é copiado para a biblioteca central.
- O desenvolvedor B encontra e corrige o mesmo defeito em sua versão do componente por não saber que A já tinha feito isso.
- O trabalho de A é desperdiçado.

PROBLEMA DA ATUALIZAÇÃO SIMULTÂNEA – CENÁRIO 2

- O desenvolvedor A encontra e corrige um defeito em sua versão do componente compartilhado.
- Uma vez corrigido, o componente modificado é copiado para a biblioteca central.
- O desenvolvedor B encontra e corrige um outro defeito em sua versão do componente, sem saber do defeito corrigido por A.
- O desenvolvedor B copia sua versão do componente para a biblioteca central.
- Além de o trabalho de A ser desperdiçado, a versão do componente que se encontra na biblioteca central continua apresentando um defeito.
- O desenvolvedor A julga o problema como resolvido.

COMO RESOLVER?

- O problema da atualização simultânea não pode ser resolvido simplesmente copiando componentes compartilhados para uma biblioteca central.
- Algum mecanismo de controle é necessário para gerenciar a entrada e saída dos componentes.

O QUE É GERÊNCIA DE CONFIGURAÇÃO?

Gerência de configuração (GC) é o processo de identificar, organizar e controlar modificações ao software sendo construído

A idéia é maximizar a produtividade minimizando os enganos

OBJETIVOS DE GC

- Definir o ambiente de desenvolvimento.
- Definir políticas para controle de versões, garantindo a consistência dos artefatos produzidos.
- Definir procedimentos para solicitações de mudanças.
- Administrar o ambiente e auditar mudanças.
- Facilitar a integração das partes do sistema.

BENEFÍCIOS

- Aumento de produtividade no desenvolvimento.
- Menores Custos de Manutenção.
- Redução de defeitos.
- Maior rapidez na identificação e correção de problemas.

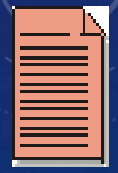
CONCEITOS BÁSICOS

CONFIGURAÇÃO

configuração
do software

ITEM DE CONFIGURAÇÃO

- Um conjunto de itens de hardware e/ou software vistos como uma entidade única para fins de gerência de configuração.
- Um item de configuração está sujeito a mudanças e essas devem obedecer às políticas estabelecidas.
- Normalmente, um item de configuração é estabelecido para cada pedaço de software que pode ser projetado, implementado e testado de forma independente.



CONFIGURAÇÃO DE SOFTWARE

BASELINE

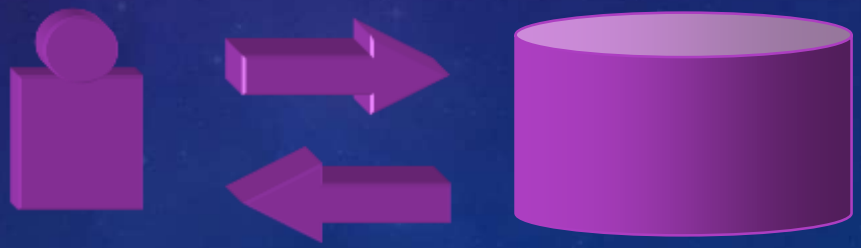
- Uma especificação ou produto que foi formalmente revisado e aceito.
  - Serve como base para os passos posteriores do desenvolvimento.
- A configuração do software em um ponto discreto no tempo.
- Só pode ser modificado através de procedimentos formais (solicitações de mudança).
- Um artefato ou conjunto de artefatos só se torna um item de configuração depois que um baseline é estabelecido.

RAZÕES PARA CRIAR UM BASELINE

- **Reprodutibilidade** – a habilidade de reproduzir uma versão anterior do sistema.
- **Rastreabilidade** – Estabelece uma relação predecessor-sucessor entre artefatos do projeto (projeto satisfaz requisitos, código implementa projeto, etc.).
- **Geração de Relatórios** – A comparação dos conteúdos de dois *baselines* ajuda na depuração e criação de documentação.
- **Controle de Mudanças** – referencial para comparações, discussões e negociações.

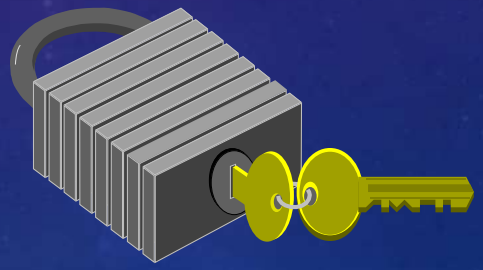
REPOSITÓRIO

- Local (físico e lógico) onde os itens de um sistema são guardados.
- Pode conter diversas versões do sistema.
- Utiliza mecanismos de controle de acesso.

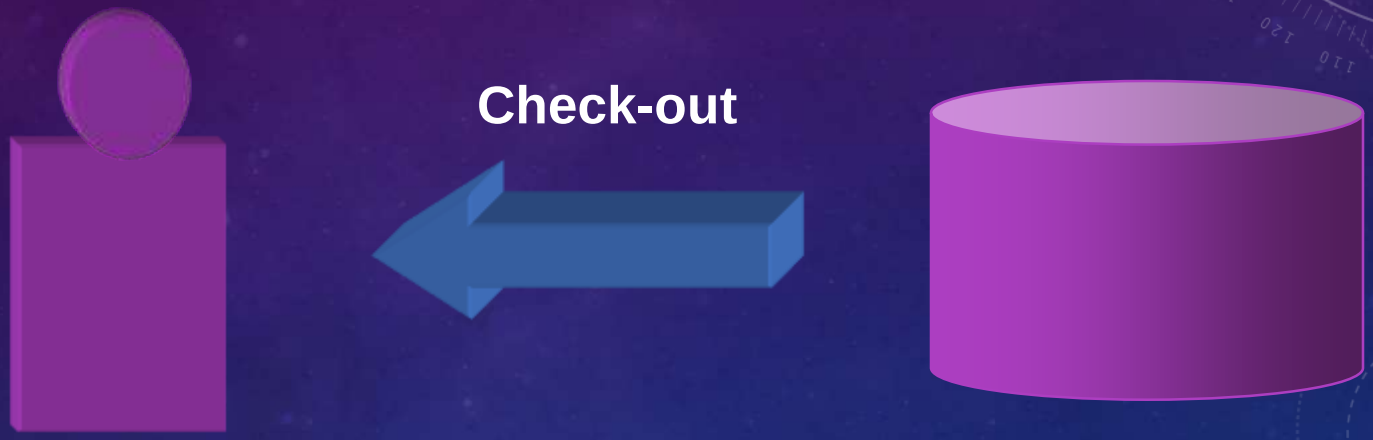


LOCK

- Resolve a Atualização Simultânea.
- Garante que apenas o usuário que detém o lock pode alterar o arquivo.
- Problema: “serializa” o trabalho dos desenvolvedores.



CHECK-OUT



CHECK-OUT

- Recupera a (última) versão de um item de configuração guardada no repositório.
  - Escrita
    - Verifica que ninguém detém o lock do item de configuração.
    - Obtém o lock do item.
    - Cria uma cópia, para edição, no cliente.
  - Leitura
    - Verifica que alguém já detém o lock.
    - Cria uma cópia, apenas para leitura, no cliente.

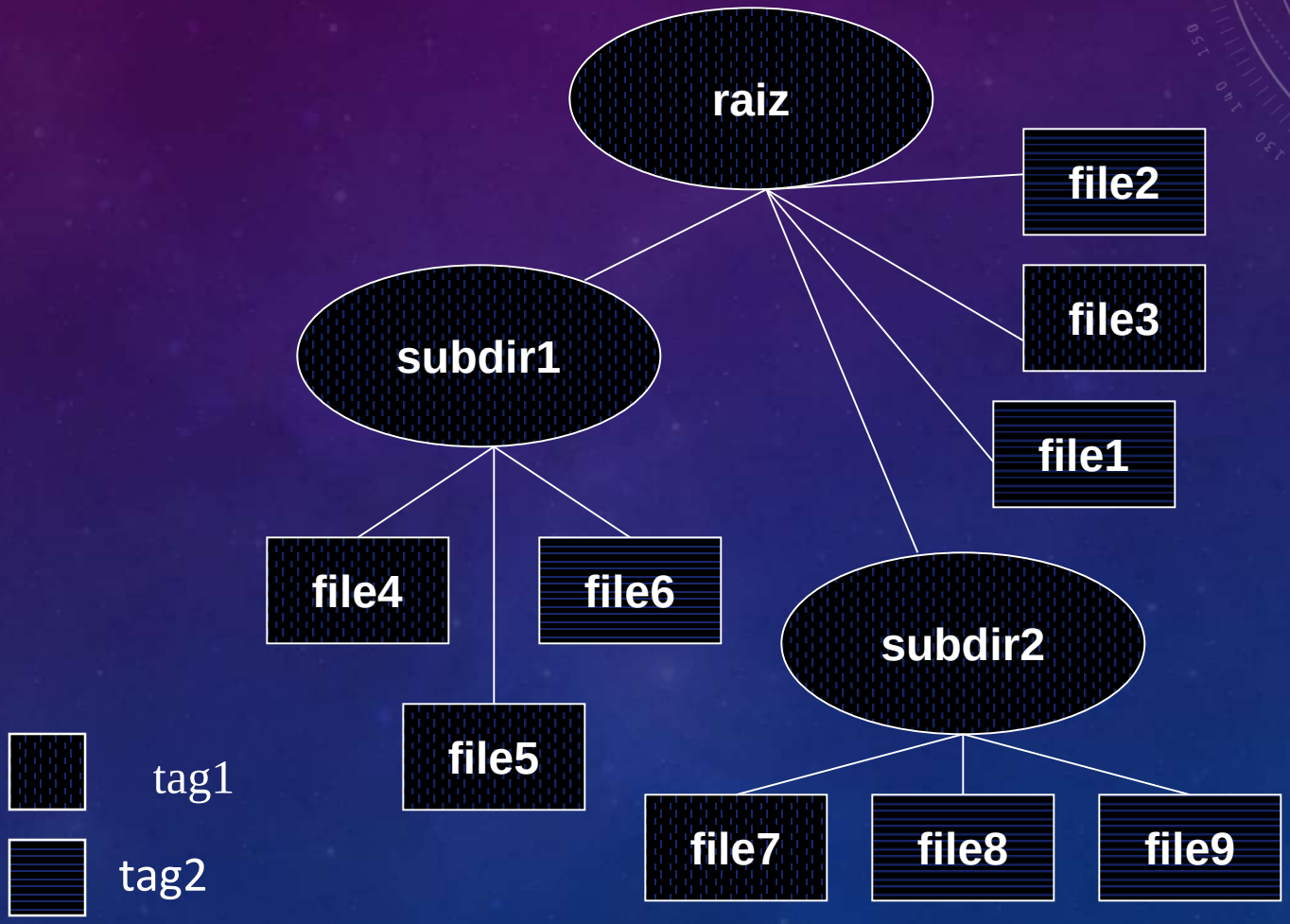
BUILD

- Representa uma versão ainda incompleta do sistema em desenvolvimento, mas com certa estabilidade.
- Costuma apresentar limitações conhecidas.
- Espaço para integração de funcionalidades.
- Inclui não só código fonte, mas documentação, arquivos de configuração, base de dados, etc.
- A política de geração dos builds deve ser bem definida na estruturação do ambiente

TAGS

- Rótulos que são associados a conjuntos de arquivos.
- Um tag referencia um ou mais arquivos em um ou mais diretórios.
  - Costuma-se usar tags para:
    - Denominar projeto rotulando todos os arquivos associados ao projeto.
    - Denominar uma versão do projeto (um build ou release) rotulando todos os arquivos associados ao build ou release.

TAGS - EX


raiz
file2
file3
file1
subdir1
file4
file6
subdir2
file5
file7
file8
file9
tag1
tag2

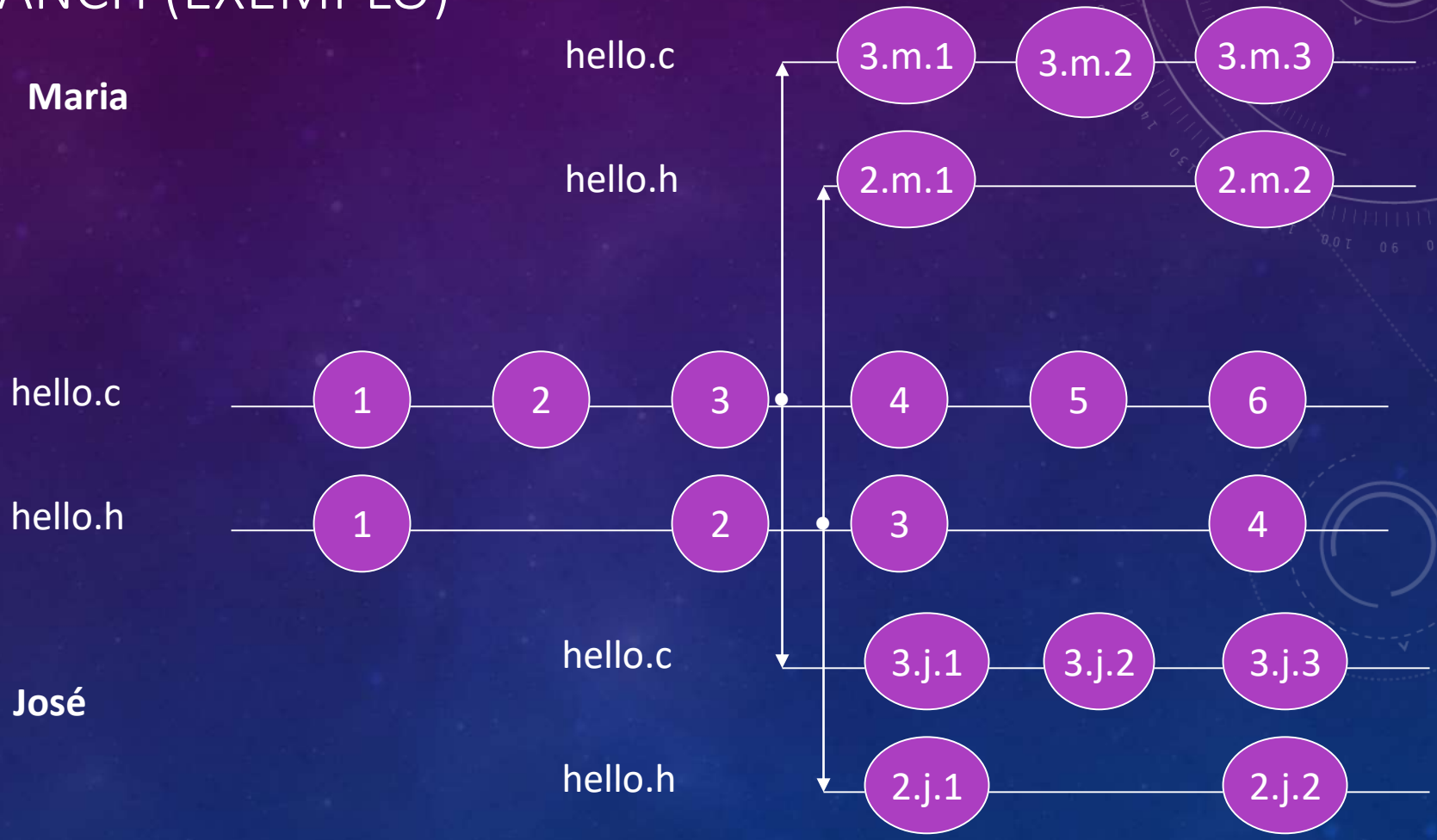
BRANCH

- Criação de um fluxo alternativo para atualização de versões de itens de configuração.
- Recurso muito poderoso.
- Devem existir regras bem definidas para criação de branches:
  - Por que e quando devem ser criados?
  - Quais os passos?
  - Quando retornar ao fluxo principal?

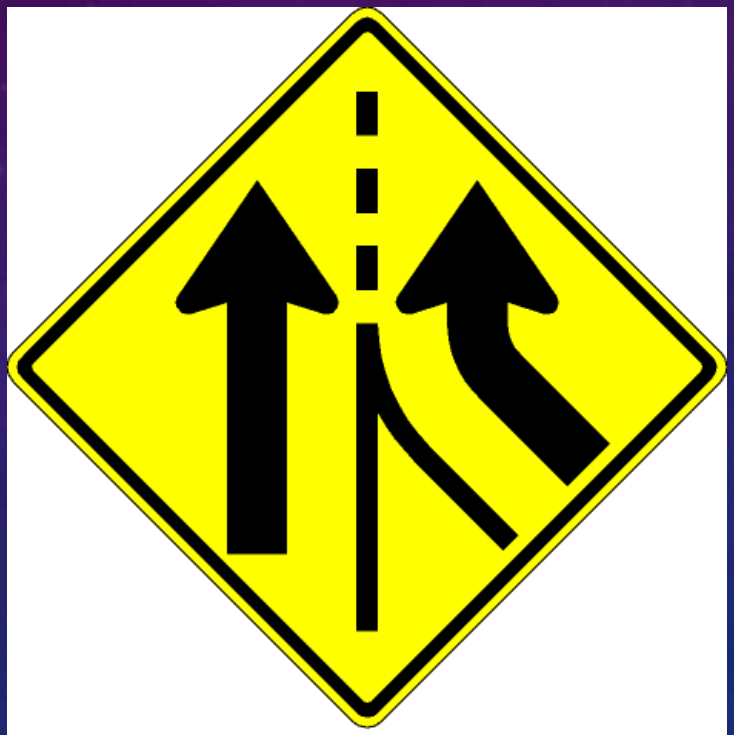
BRANCH

- Uso de lock inviabiliza a criação de branches.
- Branches normalmente se originam de correções em versões anteriores.

BRANCH (EXEMPLO)

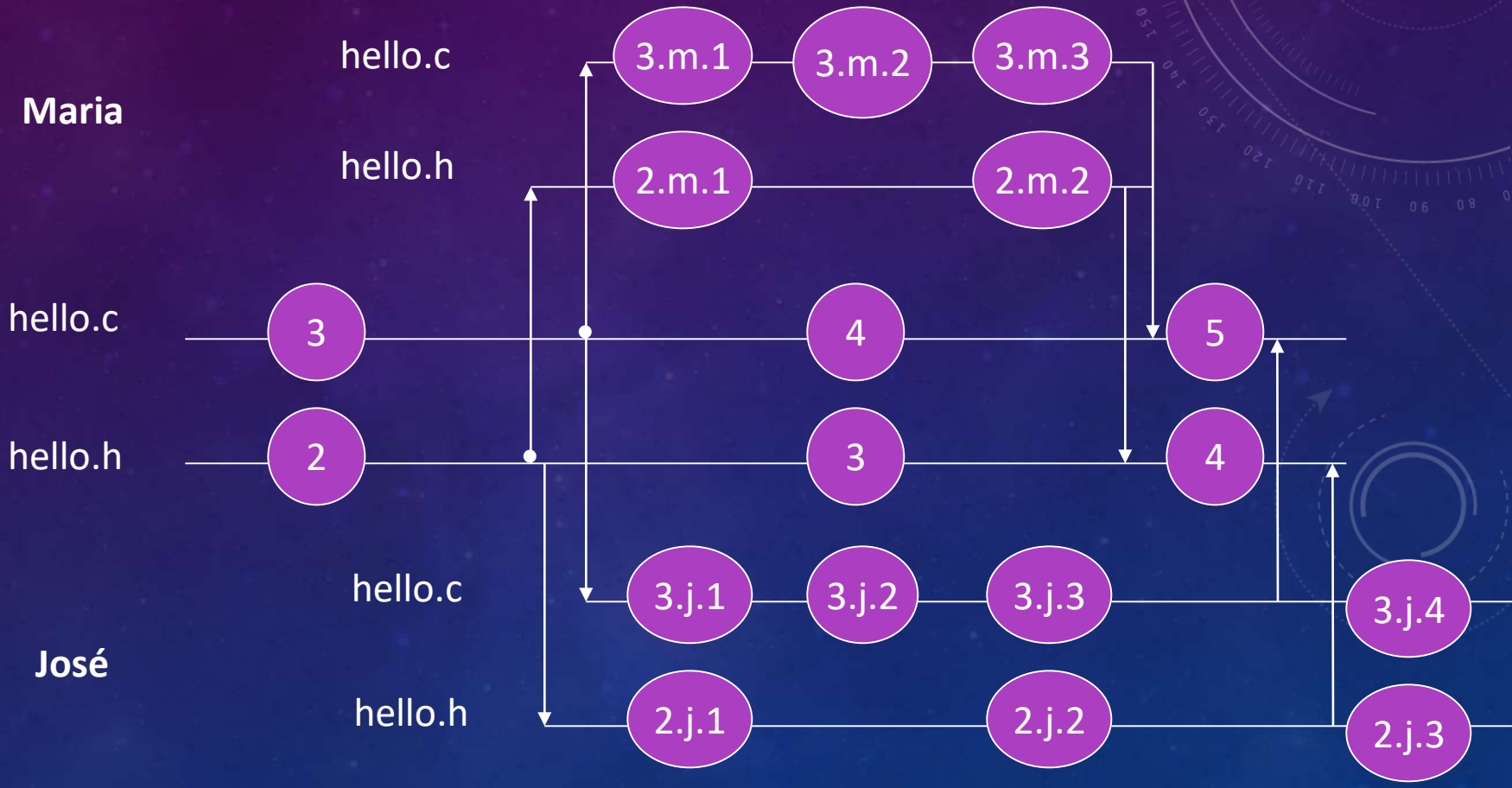


MERGE



W4-3
Added Lane

MERGE (EXEMPLO)



OPORTUNIDADES CRIADAS COM GC

- Reuso de itens de software
  - Artefatos
  - Componentes
- Automação de processo
  - Construção de *builds*
  - Geração de releases
  - Testes
  - Integração
- Aumento da produtividade das equipes.
- Redução de retrabalho.
- Melhoria do acompanhamento do projeto.

DISTRIBUIÇÃO

DOCUMENTAÇÃO

- Qual a importância da documentação de sistemas?

POR QUE A AVERSÃO A DOCUMENTAR?

- Cultura
  - Medo de controle e responsabilização.
  - Cultura do improviso (jeitinho).
  - Orientação para o produto e não, o processo.
  - Heroísmo para consertar em vez de planejar.
  - Estimativas de prazo e custo impostas, não realistas.
  - Pressão de tempo e não, boas práticas.
  - Fracassos anteriores, informações desatualizadas.
  - Falta de integração entre diversos criadores e potenciais usuários da documentação.