

# GERENCIAMENTO DE CONFIGURAÇÕES DE SOFTWARE USANDO A FERRAMENTA "GFORGE"

Artur Sampaio - ICMC/USP  
Agosto de 2010

## SCM - Definição

- ❑ Conjunto de práticas adotadas para administração de códigos fonte, controle de alterações e distribuição de pacotes de software.
  - Gerenciar diferentes versões dos produtos de *software*, controlando, auditando e reportando as alterações realizadas
  - Garantir a integridade, confiabilidade e reprodutibilidade do desenvolvimento de produtos de *software*, desde o projeto até a distribuição

2

## SCM - Visão geral

- ❑ Rastrear e controlar alterações realizadas no *software*
  - Controlar revisões (VCS)
  - Rastrear defeitos (*Issue/Ticket Tracking*)
  - Manter histórico da evolução do produto
- ❑ Estabelecer *baselines* (distribuições estáveis)
- ❑ Garantir aderência ao processo de desenvolvimento

3

## A edição concorrente - 1



4

## A edição concorrente - 2



5

## A edição concorrente - 3



6

## VCS (Version Control Systems) - Controle de revisões

- Gerenciamento de alterações em arquivos
  - Alterações recebem um identificador único, para posterior referência
  - Permite resgatar versões anteriores dos arquivos
  - Permite a criação de linhas de desenvolvimento paralelas (*branches*)
  - Auxilia o usuário no processo de *merge* (mesclagem) de diferentes versões do arquivo
  - Utilização eficiente de espaço, armazenando apenas as alterações entre versões

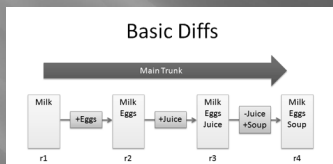
7

## VCS - Ferramentas

- Gratuitos / comunitários
  - CVS (Concurrent Version System)
  - SVN (Subversion)
  - Git (Linus Torvalds)
- Comerciais
  - IBM Rational ClearCase
  - Borland Starteam
  - Microsoft Visual SourceSafe

8

## VCS Diff



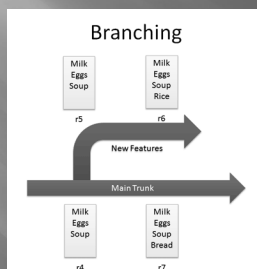
9

## VCS Compare

```
63         md.update(password.getBytes("UTF-8")); //step 3
64         System.out.println("...");
65     } catch (Exception e) {
66         System.out.println(e.getMessage());
67     }
68     byte raw[] = md.digest(); //step 4
69     //String hash = new String(raw); //step 5
70     String hash = new String(Base64.encode(raw));
71     ApplicationLog.debug("SHA1 encoded: " + hash);
72     //System.out.println("SHA1 encoded: " + hash);
73     return hash; //step 6
74 }
75
76 public boolean validateLogin(String username, String password) {
77     if (username.length() == 0) {
78         ApplicationLog.debug("username is blank and invalid");
79         return false;
80     }
81     if (password.length() == 0) {
82         ApplicationLog.debug("password is blank and invalid");
83         return false;
84     }
85     Password instance = new Password();
86     String encryptedPassword = encryptPassword(password);
87     if (encryptedPassword.equals(instance.getPassword())) {
88         ApplicationLog.debug("password is valid and correct");
89         System.out.println("...");
90         return true;
91     } else {
92         ApplicationLog.debug("password is incorrect");
93         System.out.println("...");
94         return false;
95     }
96 }
97
98 private String encryptPassword(String password) {
99     try {
100         ApplicationLog.debug("...");
101         return Base64.encode(password.getBytes());
102     } catch (Exception e) {
103         ApplicationLog.debug("...");
104         return null;
105     }
106 }
107 }
```

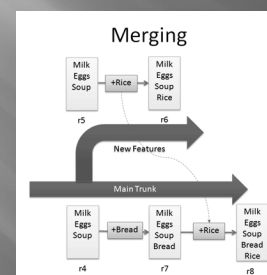
10

## VCS Branch



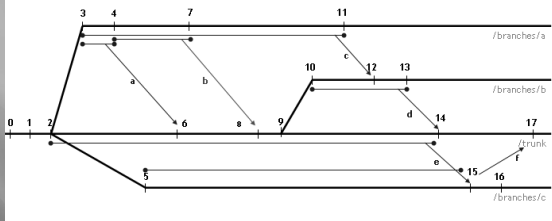
11

## VCS Merge



12

## Histórico de mudanças



13

## Bug Tracking

- Registro e rastreamento de requisitos (defeitos existentes ou melhorias propostas)
  - Auxilia na distribuição de atividades entre membros da equipe
  - Atribui graus de prioridade a cada requisito
  - Acompanha o ciclo de vida de cada requisito, registrando seu histórico e estado
  - Suporta o processo de desenvolvimento adotado pela equipe

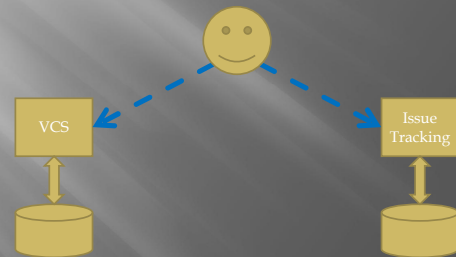
14

## Bug Tracking - Ferramentas

- Gratuitos / comunitários
  - Bugzilla
  - MantisBT
  - Trac
- Comerciais
  - IBM Rational Clear Quest

15

## Ferramentas não integradas



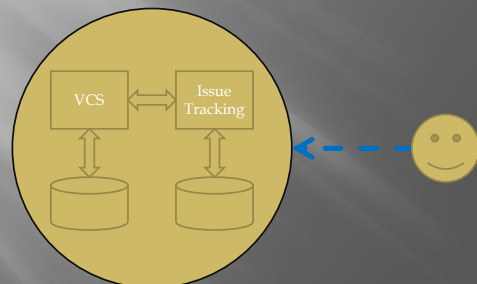
16

## Ferramentas não integradas

- Processo centrado no usuário
  - Usuário é responsável por manter consistência entre VCS e tracker
  - Não há ligação entre os artefatos *commitados* e a atividade do tracker
  - Boa prática: cada *commit* recebe como comentário o número do tracker associado, tentando garantir rastreabilidade

17

## Ferramentas integradas



18

## Ferramentas integradas

- ❑ Processo centrado em ferramentas
  - Consistência automática entre VCS e *tracker*
  - Possibilidade de só permitir *commits* com *tracker* associado
  - Criação automática vínculo entre artefatos *commitados* e atividades de *tracker*
- ❑ Sourceforge
  - Só hospeda aplicações *open source*
- ❑ JIRA
  - Produto comercial

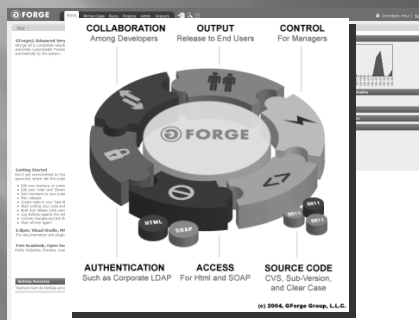
19

## GForge

- ❑ Ambiente colaborativo para desenvolvimento de software
  - Derivado do SourceForge; mesma equipe de desenvolvimento
  - Suporta múltiplos projetos e múltiplos usuários
  - Integra módulos de VCS, tracking, comunicação, fórum e baselines
  - Realiza levantamento de métricas de produtividade por projeto e por membro

20

## Tela inicial do GForge



21

## Listagem de projetos

Captura de tela da interface de listagem de projetos do GForge. O cabeçalho mostra 'GForge' e 'Home - No Bug, Search, Reports, Admin, Control'. Abaixo, há uma tabela com colunas para Nome, Descrição, Status e Data. Alguns projetos listados incluem 'GForge', 'GForge CVS', 'GForge Sub-versions', etc.

22

## Resumo de um projeto

Captura de tela do resumo de um projeto no GForge. O cabeçalho mostra 'SGPC' e 'Atividade'. A tabela principal tem colunas para Hora, Tipo de Atividade e Por. O resumo mostra uma lista de atividades realizadas em agosto de 2019, todas atribuídas a Artur Sampaio.

23

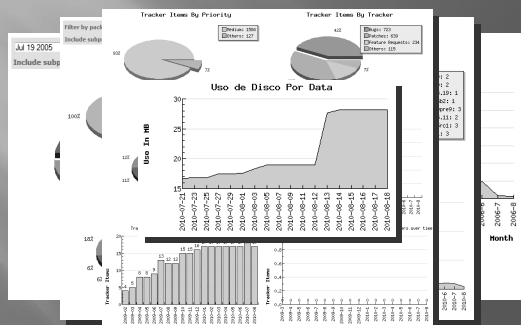
## Listagem de BUG 's

Captura de tela da interface de listagem de bugs do GForge. O cabeçalho mostra 'GForge' e 'Home - No Bug, Search, Reports, Admin, Control'. Abaixo, há uma tabela com colunas para ID, Resolvido, Prioridade, Dependência, Status, Criado por, Data de criação, Data de atualização e Data de fechamento. Alguns bugs listados incluem 'GForge', 'GForge CVS', 'GForge Sub-versions', etc.

24



## Relatórios



31

## Conclusões

- ❑ Práticas de SCM são fundamentais para a maturidade de todo e qualquer produto de software
- ❑ É necessário o uso de ferramentas adequadas
- ❑ Ferramentas isoladas dependem muito da disciplina do desenvolvedor
- ❑ Ferramentas integradas garantem consistência no processo de desenvolvimento e simplificam a gestão do projeto

32

## Perguntas

## Quero saber mais

- ❑ <http://gforge.org/>
- ❑ <http://subversion.tigris.org/>
  
- ❑ [artur@icmc.usp.br](mailto:artur@icmc.usp.br)

*"The only constant in software development is change."*

33

34