Instituição Faculdade Senac – Goiás

Matéria: **Controle de aversão**

Professora: **Ana Flávia**

Aluno: **Paulo Henrique da Silva e Gabriel Ribeiro de Sousa**

Turma: **ADS – Modulo II Turno: Noturno Sala: 21**

Pesquisa sobre o controle de aversão

O sistema de aversão na função pratica da **Ciência da Computação** e da **Engenharia de Software**, é **software** que tem a finalidade de gerenciar deferentes versões no desenvolvimento de um documento. Esses sistemas são comumente utilizados no **desenvolvimento de software** para controlar as diferentes versões – histórico e desenvolvimento – dos **códigos-fontes** e também da **documentação.**

Esse tipo de sistema é muito presente nas empresas e instituições de **tecnologia** e **desenvolvimento de software.** É também é muito comum no desenvolvimento de **software livre.** É útil, em diversos aspectos, tanto para projetos pessoais pequeno e simples como também para grandes projetos comerciais.

Entres os mais comuns encontre-se as soluções livres: **CVS**, **Mercurial**, **Git** e **SVN** (subversion); e as comercias: **SourseSafe**, **TFS**, **PVCS** (serena) e **ClearCase**. O desenvolvimento de software livre utiliza mais **Git** (com repositório no **GitHub**), que vem substituindo o **SVN**, que por sua vez é um sucessor do **CVS**. Muitas empresas também adotam o **Git (como Microsoft com o código fonte do Windows)** ou o **SVN,** embora algumas empresas prefiram uma solução comercial, optando pelo ClearCase (da IBM) ou Team Foundation Server (da **Microsoft**). Optar por uma solução comercial geralmente está relacionada à garantia, pois as soluções livres não se responsabilizam por erros no **software** e perdas de informações. Porém as soluções livres podem ter melhor desempenho e segurança que as comercias.

As soluções comerciais apesar de supostas garantias adicionais, não garantem o sucesso da implementação nem indenizam por qualquer tipo de erro mesmo que comprovadamente advindo do software.

A eficácia do controle de versão de software é comprovada por fazer parte das exigências para melhorias do processo de desenvolvimento de certificações tais como **CMMI** e **SPICE**.

As principais vantagens de se utilizar um sistema de controle de versão para rastrear as alterações feita durante o desenvolvimento de software ou o desenvolvimento de um documento de texto qualquer são:

- **Controle do histórico:** facilidade em desfazer e possibilidade de analisar o histórico do desenvolvimento, como também facilidade no resgate de versões mais antigas e estáveis. A maioria das implementações permitem analisar as alterações com detalhes, desde a primeira versão até a última.
- **Trabalho em equipe:** um sistema de controle de versão permite que diversas pessoas trabalhem sobre o mesmo conjunto de documentos ao mesmo tempo e minimiza o desgaste provocado por problemas com conflitos de edições. É possível que implementação também tenha um controle sofisticado de acesso para cada usuário ou grupo de usuários.
- **Marcação e resgaste de versões estáveis:** a maioria dos sistemas permitem marcar onde é que o documento estava com uma versão estável, podendo ser facilmente resgatado no futuro.
- **Ramificação de projeto:** a maioria das implementações possibilita a divisão do projeto em várias linhas de desenvolvimento, podem ser trabalhadas paralelamente, sem que uma interfira na outra.
- **Segurança:** cada software de controle de versão usa mecanismo para evitar qualquer tipo de invasão de agentes infecciosos nos arquivos. Além do mais, somente os usuários com permissão poderão mexer no código.
- **Rastreabilidade:** Com a necessidade de sabermos o local, o estado e a qualidade de um arquivo; o controle de versão trás todos esses requisitos de forma que o usuário possa se embasar do arquivo que deseja utilizar.
- **Organização:** Com o software é disponibilizado interface visual que pode ser visto todo arquivos controlados, desde a origem até o projeto por completo.

- **Confiança:** O uso repositório remoto ajuda a não perder arquivos por eventos imponderáveis. Além disso e disponível fazer novos projetos sem danificar o desenvolvimento.

Funcionamento básico

A maior parte das informações – com todo o histórico – ficam guardadas num **repositório** (repositor em inglês), num servidor qualquer. Geralmente um cliente pode aceder ao repositório pela **rede** (via **socket**)ou localmente quando o cliente está na mesma **máquina** do servidor.

O repositório **armazena** a informação – um conjunto de documentos – de **modo persistente** num sistema **de arquivo** ou num **banco de dados** qualquer – onde ocasiona um tipo de hierarquia entres arquivos e diretórios. Inúmeros clientes podem se conectar em um repositório, e assim leem e escrevem nesses arquivos. Incrementando dados, um usuário disponibiliza a informação para outros; fazendo a leitura dos elementos, um usuário recebe informação de outros. Além disso é viável armazenar o conteúdo em outros **dispositivos** capazes de " eternizar " e resgatar facilmente a informação.

Cada servidor pode ter vários sistemas de controle de versão e cada sistema pode ter repositórios, limitando-se na capacidade de gerenciamento de software e também e no limite físico do harwdare. Geralmente um repositório possui um **endereço lógico** que permite a conexão do cliente. Esse endereço pode ser um **IP**/ porta uma **URL**, um caminho do sistema de arquivos, etc.

A cada alteração relevante do desenvolvedor é necessário " atualizar " as informações do servidor submetendo(commit em inglês) as alterações. O servidor então guarda a nova alteração junto de todo o histórico mais antigo. Se o desenvolvedor que atualizar sua cópia local é necessário atualizar as informações locais, e para isso é necessário **baixar novidades** do servidor (ou fazer update em inglês).