EVOLUÇÃO DE UMA FERRAMENTA DE GERENCIAMENTO DE CONFIGURAÇÃO DE SOFTWARE ATRAVÉS DA ENGENHARIA REVERSA

Lairce Castanhera Beraldi
FIRP
CP 593, CEP 15.025-400, São José do Rio Preto - SP
beraldi@unirpnet.com.br

Rosely Sanches
ICMSC/USP
CP 668, CEP 13560-970, São Carlos - SP
rsanches@icmsc.sc.usp.br

Fátima Yamashiro
ICMSC/USP
CP 668, CEP 13560-970, São Carlos - SP
fátima@icmsc.sc.usp.br

Abstract

The items that comprise all information produced as parts of the software engineering process are collectively called *Software Configuration*, and constitute an important resource to the professional that works with software since it should be complete an up-to-date. The existence of an appropriate software configuration that reflects to reality depends on an effective *Software Configuration Management* process.
The activities involved in the software configuration management process are complex and the necessity of tools to support them comes into view. Usually these tools are too expensive and not available to every one that needs them. The *Revision Control System (RCS)*, in spite of not being a tool for software configuration management, because it supports only a few activities related with the management process, can be of great help to ease this task. Furthermore, RCS is a public domain system.
Only difficulty using RCS is related to its interface: usually based on command lines. This work aims at providing a graphic interface to RCS so that this system would be more user friendly. To achieve this, firstly the RCS code was submitted to a process of reverse engineering, using the *FUSION-RE* method. This method produces object-oriented analysis models (FUSION's models) from procedural code systems. The reverse engineering process provided a more abstract view of the system that was considered in the development of the graphic interface using XView.

1. INTRODUÇÃO

Muitas organizações encontram grandes dificuldades em atingir metas de custo, qualidade e cronograma no desenvolvimento de software. Compromissos irreais são assumidos, sem que haja uma avaliação sobre a possibilidade de serem cumpridos [Sanches 93]. Dessa maneira, os cronogramas e orçamentos, por não serem baseados em estimativas realistas, muitas vezes, são desobedecidos e quando cumpridos eliminam etapas importantes como testes e revisões.

Freqüentemente, o produto de organizações desse tipo é repleto de erros, a documentação é incompleta e quase sempre inconsistente, não há história de como o produto foi desenvolvido e porque as decisões de projeto e de implementação foram tomadas de determinada maneira. Normalmente isso ocorre porque essas organizações não possuem mecanismos que permitam aprender com a experiência de projetos anteriores e não tem habilidade para efetuar um controle efetivo do processo de desenvolvimento de software. Na maioria dos casos esse processo é improvisado pelos responsáveis durante o decorrer do desenvolvimento [Paulk 93a].

As modificações nos produtos dessas organizações apresentam riscos elevados,

praticamente inevitáveis, de introdução de erros a cada nova mudança [Pressman 95]. Esses erros são conhecidos como efeitos colaterais de mudanças e podem ocorrer no código, nos dados e na documentação [Freedman 90].

Essa situação é típica de organizações que não possuem um ambiente estável para desenvolvimento e manutenção de software [Paulk 93a] e que, segundo o Capability Maturity Model - CMM [Humphrey 89], encontram-se no nível de maturidade denominado "caótico".

Entre as recomendações que o CMM estabelece para melhorar a situação das empresas que se encontram no nível inicial de maturidade, uma recomendação chave é o estabelecimento de um efetivo controle sobre a configuração de software [Bollinger 91, Humphrey 88, Humphrey 89, Paulk 93b, Saiedian 95].

Configuração de Software compreende todos os itens de informação produzidos durante o ciclo de vida de software. O estabelecimento e a manutenção da integridade desses itens de informação constituem o processo de Gerenciamento de Configuração de Software [Dart 90, Pressman 95].

Muitas atividades trabalhosas estão envolvidas no gerenciamento de configuração de software, portanto para que esse gerenciamento possa ser efetuado, é importante orientação e o apoio automatizado de ferramentas. Existem no mercado várias ferramentas de apoio ao gerenciamento de configuração de software [Dart 91]. No entanto, a maioria dessas ferramentas, devido ao alto custo, não é de fácil acesso para a maioria das empresas que delas necessitam. Além disso, de um modo geral, as ferramentas de gerenciamento de configuração de software não possuem uma interface que favoreça a compreensão dos conceitos relacionados a gerenciamento de configuração e que são imprescindíveis para um bom aproveitamento das ferramentas.

Numa tentativa de difundir os conceitos relacionados ao gerenciamento de configuração de software (possibilitando que uma empresa dê os primeiros passos em direção a uma melhoria de qualidade de seus processos de desenvolvimento) e de facilitar o uso de uma ferramenta que apóia as atividades envolvidas no gerenciamento, desenvolveu-se neste trabalho, uma ferramenta denominada Graphic Revision Control System (G~RCS). Essa ferramenta foi desenvolvida tendo como suporte básico o sistema Revision Control System (RCS), de domínio público. Para isso, primeiramente, o RCS foi submetido a um processo de engenharia reversa de modo que sua funcionalidade ficasse transparente. O método utilizado para engenharia reversa foi o FUSION-RE [Penteado 95] o qual, através do Modelo de Análise do método FUSION, fornece uma visão mais abstrata do código. Em posse das informações de funcionalidade do RCS, desenvolveu-se uma interface gráfica (utilizando XView) para facilitar sua utilização, dando origem assim à ferramenta G~RCS.

2. GERENCIAMENTO DE CONFIGURAÇÃO DE SOFTWARE

Um processo de desenvolvimento de software, independente do paradigma de ciclo de vida adotado, inclui as fases de engenharia de sistemas, análise de requisitos, projeto de software, codificação, testes e manutenção [Pressman 95]. Durante o processo de desenvolvimento de software, é produzida uma grande quantidade de itens de informação. Esses itens são chamados *itens de configuração de software* e o conjunto dos mesmos compõem uma *configuração de software*.

Quando o software é construído e posteriormente em suas manutenções, ocorrem muitas alterações na configuração de software. As atividades desenvolvidas para administrar

essas alterações são conhecidas como *gerenciamento de configuração de software* [Buckley 94, Dart 90, Pressman 95].

O gerenciamento de configuração de software é composto por quatro atividades básicas [Bersoft 79, Berlack 92, Capretz 94, Narayanaswamy 87, Pressman 95]:

1. *Identificação dos itens de configuração de software* - Várias tarefas estão associadas com a atividade de identificação dos itens de configuração de software: criação da hierarquia dos itens de informação produzidos durante o ciclo de vida do software, seleção dos itens de informação relevantes para a configuração de software, especificação do relacionamento entre os itens de configuração de software, criação de um esquema de identificação dos itens de configuração de software e planejamento das linhas de referência [Berlack 92].
2. *Controle de configuração de software* - Após a identificação dos itens de configuração de software, deve ser instituído o controle sobre estes itens. O controle da configuração do software é dividido em dois controles básicos: Controle de Mudanças e Controle de Versões.
3. *Relato do "status" da* configuração *de software*. Essa atividade está associada com todas as outras atividades de gerenciamento de configuração e proporciona um meio através do qual a história do ciclo de vida do software pode ser recuperada [Bersoft 84, Narayanaswamy 87].
4. *Auditoria da Configuração de Software* - Essa atividade consiste em verificar e validar a configuração do software [Capretz 92]. Existem 2 tipos de auditoria de configuração de software que são pré-requisitos para estabelecimento das linhas de referência do ciclo de desenvolvimento de software: a Auditoria Funcional e a Auditoria Física [Berlack 92].

3. APRESENTAÇÃO DO SISTEMA RCS

O Revision Control System (RCS) é uma ferramenta de domínio público, desenvolvida na Universidade de Purdue [Tichy 85], cuja principal função é controlar versões de arquivos textos.

O RCS controla grupos de versões. Um grupo de versões é um conjunto de arquivos texto que mantém uma sequência cronológica entre si, e que representam a sequência de evolução do texto inicial. Essas versões são armazenadas em forma de uma árvore, sendo que a primeira versão é chamada *raiz*, a última versão do tronco é chamada *head* e as versões intermediárias formam os ramos.

O RCS foi originalmente projetado para manipular apenas programas, porém percebeu-se que ele também poderia ser útil para controlar qualquer arquivo que fizesse parte da configuração de software. A maior restrição é que o RCS trabalha apenas com arquivos texto e código objeto e sua utilização em arquivos não-texto (binários) pode corromper os dados.

A interface com o usuário consiste apenas de comandos de linha, o que torna difícil, para o usuário principiante, explorar toda a potencialidade do sistema, devido ao grande número de parâmetros. Outra desvantagem é que os resultados ficam todos espalhados na tela, dificultando o seu entendimento [Yamashiro 96].

A versão do RCS que foi estudada apoia apenas algumas funções das atividades de gerenciamento de configuração de software: esquema de identificação, controle de mudanças, controle de versões e relato do "status" da configuração [Yamashiro 96].

a) *Esquema de Identificação do RCS*

A criação de um esquema de identificação dos itens de configuração de software é a única função da atividade de identificação dos itens de configuração de software realizada pelo RCS. O RCS identifica automaticamente cada versão fornecendo um nome, um número de versão, a hora de criação e o autor [Yamashiro 96]. O nome fornecido pelo RCS é decorrente do nome do arquivo. O número de versão é formado por dois campos, sendo o primeiro chamado número de versão e o segundo, número de nível. Dois campos a mais no número de versão indicam que essa versão é uma versão intermediária e pertence a um ramo. O terceiro campo indica o número do ramo e o quarto campo indica o nível da versão dentro desse ramo. Por exemplo (Figura 1), os números de versão 1.3.1.1 e 1.3.1.2 indicam respectivamente a primeira e a segunda versão do primeiro ramo da versão 1.3; as versões do segundo ramo da versão 1.3 são numeradas como 1.3.2.1, 1.3.2.2. A data e a hora no esquema de identificação é capturada do sistema. No RCS, a identificação é anexada no arquivo texto.

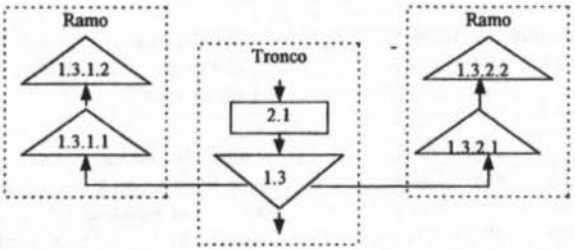


Figura 1: Exemplo de Árvore de Versões com Versões Intermediárias de Itens de Configuração de Software

b) *Controle de Mudanças no RCS*

O controle de mudanças, da atividade de controle de configuração de software, é efetuado parcialmente pela função de controle de acesso do RCS. A função de controle de acesso determina quem têm permissão para acessar e modificar uma versão específica. Isso é feito através de uma lista de acesso, relativa a cada arquivo RCS, contendo os *login names* dos que têm permissão de acesso [Yamashiro 96]. Para efetuar o controle de sincronização, o RCS possui um sistema de *lock*, através do comando *co* com o parâmetro *(-l)*, que bloqueia o objeto que foi recuperado (*checked-out*), impedindo que qualquer outra atualização possa ser feita sobre a mesma versão, até que a versão modificada seja recolocada sob controle do RCS. Isto impede que mudanças paralelas e conflitantes sejam feitas ao mesmo tempo, comprometendo uma à outra.

c) *Controle de Versões no RCS*

O controle de versões, da atividade de controle de configuração de software, é realizada completamente pelo RCS. O RCS armazena as versões em forma de uma árvore, onde há uma raiz (normalmente a versão 1.1), um tronco e ramos (Figura 2). A estrutura de ramos é muito útil, porque permite o armazenamento de versões intermediárias às existentes, o que muitas vezes é necessário. Através dessa estrutura, pode-se armazenar temporariamente uma versão em um ramo e mais tarde uni-la a uma versão do tronco através do comando

rcsmerge. Por questão de economia de espaço e para obter um tempo de acesso razoavelmente rápido, o RCS armazena as versões na forma de deltas, utilizando o conceito de delta negativo para os troncos (armazena-se integralmente a versão mais recente do tronco e os deltas necessários para recuperar as versões mais antigas) e o conceito de delta positivo para os ramos (armazena-se integralmente a versão mais antiga e os deltas necessários para recuperar as versões mais recentes). A forma de armazenamento com deltas negativos para as versões do tronco diminui o tempo de acesso, visto que de acordo com uma pesquisa, efetuada na Universidade de Purdue, 95% das operações de *check-out* são realizadas sobre a última versão. A forma de armazenamento dos deltas positivos para as versões dos ramos permite que as versões intermediárias sejam recuperadas a partir das versões do tronco, não precisando armazenar nenhuma versão intermediária inteira. A interface com o usuário não permite a visualização dos deltas. Os deltas empregados pelo RCS são baseados em linha, o que significa que as únicas alterações permitidas são a inserção e remoção de linhas. Mesmo que apenas um caracter em uma linha seja alterado, a linha inteira é considerada alterada. O comando *rcsdiff*, que mostra as diferenças existentes entre duas versões, funciona de acordo com esse conceito, comparando linha por linha e mostrando as linhas que diferem entre as versões.

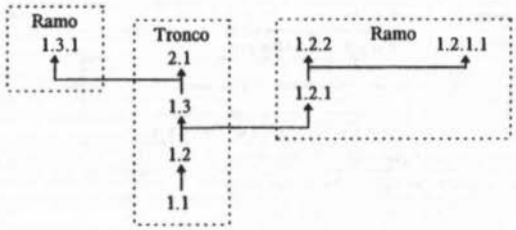


Figura 2: Árvore de Versões de um Item de Configuração de Software no RCS.

d) *Relato do "Status" da Configuração no RCS*

A atividade de relato do "status" da configuração de software é atendida parcialmente pelo RCS. Através do uso de *keywords* colocadas como identificadores no arquivo, em forma de comentário, o RCS realiza parcialmente o relato do "status" da configuração . Essas *keywords* são expandidas (substituídos pelas informações correspondentes) através comando *co*. Ao recuperar-se *(check-out)* uma versão, as informações são atualizadas automaticamente. Por exemplo, a *keywords Log* acumula todas as mensagens de *log* que são solicitadas durante o *check-in*, permitindo que se mantenha a história completa de uma versão diretamente dentro dessa versão. Para recuperar-se as informações, inicialmente o usuário insere *strings* da forma *$keyword$* no arquivo. Ao realizar um *check-out*, o comando *co* expande essas *strings* por *strings* da forma *$keyword:valor$*. Se uma versão contendo *keywords*, assim expandidas, é novamente *checked-in*, o campo valor será substituído durante o próximo *check-out*, isto é, os valores de *keywords* são atualizados automaticamente a cada *check-out*. O RCS também permite o uso de *keywords* no código objeto. Para expandir uma *keyword* no código objeto, basta colocá-la em uma *string* de caracteres literais. Por exemplo, na linguagem C, isto pode ser feito do seguinte modo: *static char rcsid[] = "Id";*. O comando *ident* extrai uma cópia das *keywords* de qualquer arquivo de trabalho, inclusive do código objeto, devidamente expandidos e atualizados pelo

comando *co*. O sistema armazena automaticamente algumas importantes informações de controle, mesmo que se decida pela não utilização de *keywords*. Porém, essas informações não são visíveis no próprio texto da versão, como as *keywords*, e nem podem ser extraídas pelo comando *ident*. Para serem visíveis é necessário utilizar o comando *rlog*.

Um resumo das atividades de gerenciamento de configuração de software que são atendidas pelo RCS é apresentado no Quadro 1.

Quadro 1: Atividades de Gerenciamento de Configuração de Software apoiadas pelo RCS

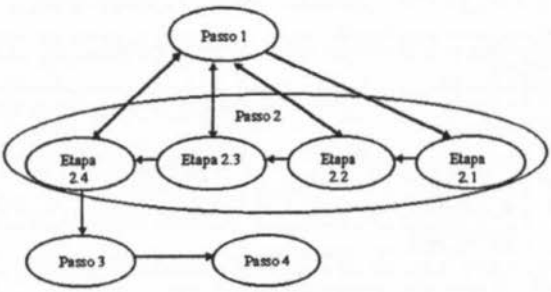
ATIVIDADE 1. Identificação dos Itens de Configuração de Software	
a) Criação da Hierarquia dos Itens de Informção produzidos durante o Ciclo de Vida de Software	**Não**
b) Seleção dos Itens de Informação Relevantes para a Configuração de Software	**Não**
c) Especificação dos Relacionamentos entre os Itens de Configuração de Software	**Não**
d) Criação de um Esquema de Identificação dos Itens de Configuração de Software	**Sim**
e) Planejamento das Linhas de Referência	**Não**
ATIVIDADE 2. Controle de Configuração de Software	
a) Controle de Mudanças	**Parcialmente**
b) Controle de Versões	**Sim**
ATIVIDADE 3. Relato do "Status" da Configuração de Software	
Relato do "Status" da Configuração de Software	**Parcialmente**
ATIVIDADE 4. Auditoria da Configuração de Software	
a) Auditoria Funcional da Configuração de Software	**Não**
b) Auditoria Física da Configuração de Software	**Não**

4. O MÉTODO *FUSION-RE* DE ENGENHARIA REVERSA

O principal objetivo do método FUSION-RE de engenharia reversa [Pentedo 95], desenvolvido no ICMSC-USP, é a produção dos modelos da fase de análise do método FUSION (um modelo de análise orientado a objetos), a partir do código fonte (desenvolvido sem a tecnologia de orientação a objetos) do sistema sumetido a engenharia reversa e de informações existentes sobre tal sistema. O método é composto de quatro passos, resumidamente apresentados no Quadro 2. A sequencia ideal para a execução dos passos é a apresentada na Figura 3, sendo que o passo 1 e as etapas 2.2, 2.3 e 2.4 são conduzidos com bastante interseção e interação.

Quadro 2 : O Método FUSION-RE de Engenharia Reversa

PASSO 1. REVITALIZAR A ARQUITETURA DO SISTEMA COM BASE NA DOCUMENTAÇÃO EXISTENTE
Objetivo: Obter informações relacionadas à arquitetura do sistema para o seu entendimento. **Documentos Produzidos:** Lista de todos os procedimentos, sua descrição e a relação chama/chamado por.
PASSO 2. RECUPERAR O MODELO DE ANÁLISE DA SOLUÇÃO ATUAL
Objetivo: Obter um modelo de análise considerando somente os aspectos físicos.
Etapa 2.1. Definir Temas
Objetivo: Modelar em temas as informações armazenadas relativas às entradas, saídas, armazenamento permanente e temporário **Documentos Produzidos:** Lista de Temas
Etapa 2.2. Desenvolver o Modelo de Objetos
Objetivo: Elaborar um modelo com as classes e seus relacionamentos, extraídos dos tipos abstratos de dados que compõem a base de dados do sistema. **Documentos Produzidos:** Modelo de Objetos, lista de elementos de dados, procedimentos associados às classes, lista das anomalias existentes.
Etapa 2.3. Definir o Ciclo de Vida
Objetivo: Mostrar o comportamento global do sistema. **Documentos Produzidos:** Modelo de Ciclo de Vida.
Etapa 2.4. Abstrair Operações e Desenvolver o Modelo de Operações
Objetivo: Obter as operações realizadas pelo sistema. **Documentos Produzidos:** Modelo de Operações.
PASSO 3. ABSTRAIR O MODELO DE ANÁLISE DO SISTEMA
Objetivo: Obter um modelo de análise do sistema considerando os aspectos do domínio da aplicação.
Etapa 3.1. Desenvolver o Modelo de Objetos
Objetivo: Elaborar um modelo de objetos considerando as classes e seus relacionamentos que devem ser tratados pelo sistema.
Documentos Produzidos: Modelo de Objetos. Para cada classe: lista dos atributos e métodos.
Etapa 3.2. Elaborar o Modelo de Ciclo de Vida
Objetivo: Fornecer uma visão global do comportamento do sistema a partir da abstração realizada. **Documentos Produzidos:** Modelo de Ciclo de Vida.
Etapa 3.3. Especificar o Modelo de Operações
Objetivo: Descrever como as operações devem ser realizadas. **Documentos Produzidos:** Modelo de Operações.
PASSO 4. MAPEAR O MODELO DE ANÁLISE DO SISTEMA PARA O MODELO DE ANÁLISE DO SISTEMA ATUAL
Objetivo: Descrever a Relação entre os Modelos de Análise do Sistema atual e novo. **Documentos Produzidos:** Mapeamento das Classes e dos Métodos do MAS para o MASA.



Passo 1: Revitalização da Arquitetura
Passo 2: Recuperação do Modelo de Análise
Etapa 2.1: Definição de Temas
Etapa 2.2: Desenvolvimento do Modelo de Objetos
Etapa 2.3: Definição do Ciclo de Vida
Etapa 2.4: Desenvolvimento do Modelo de Operações
Passo 3: Abstração do Modelo de Análise
Passo 4: Mapeamento

Figura 3: Sequencia Ideal para a Execução dos Passaos do Método FUSION-RE

5. GRAPHIC REVISION CONTROL SYSTEM (G~RCS)

Para elaboração da interface gráfica, primeiramente, o RCS foi submetido a um processo de engenharia reversa, usando o Método FUSION-RE. O resultado desse processo de engenharia reversa foi uma visão mais abstrata do sistema RCS a qual foi utilizada no desenvolvimento da interface gráfica.

Como a engenharia reversa parte de informações já existentes, apresenta-se no Quadro 3 as informações relativas ao RCS, com base nas quais o método foi aplicado.

Quadro 3: Informações Disponíveis sobre o RCS

Ambiente Operacional:	- Estações de Trabalho com Sistema Operacional UNIX.
Tamanho:	- Cerca de 18.000 linhas de código.
Linguagem:	- "C" Padrão.
Estruturação:	- Modularizado por função (23 módulos).
Documentação Interna:	- Linhas de comentário pouco explicativas e em pequeno número.
Documentação Externa:	- Manual "On-line" com informações pouco explicativas; - Instruções de uso do sistema elaboradas no ICMSC-USP [Yamashiro 96].
Usuários do RCS:	- Apenas os pesquisadores envolvidos nesta pesquisa.

5.1. Desenvolvimento da Ferramenta *Graphic Revision Control System (G~RCS)*

A ferramenta Graphic Revision Control System (G~RCS) foi desenvolvida no Laboratório de Engenharia de Software do Instituto de Ciências Matemáticas de São Carlos, Universidade de

São Paulo (LABES ICMSC-USP). Sua implementação foi feita utilizando o *toolkit Xview*. Esse *toolkit* fornece suporte interativo para o desenvolvimento de aplicações gráficas em sistema *X Window*, através de um conjunto de rotinas prontas que implementam as técnicas de interação mais comuns entre usuário e máquina (menus, *scrollbars*, *panel-buttons*, etc). Visando facilitar o desenvolvimento de aplicações em *Xview*, existem diversos editores de projeto de interface gráfica. Para a elaboração das telas G~RCS foi utilizado o *GUIDE da Sun Microsystems*, que gera código em uma linguagem de interface gráfica (GIL). A partir desta linguagem é gerado o código fonte C++ e XView através do Generate C++ and XView Source Code (gxv++).

Para o desenvolvimento da ferramenta, optou-se por não modificar o código fonte do RCS e sim utilizar seus comandos executáveis. Para isso, linhas de comandos são montadas a partir das opções da interface escolhidas pelo usuário e são executadas em uma janela *TTY* (janelas que emulam terminais) do *XView*. Essa ferramenta facilita a entrada de dados no RCS, porém a saída continua sendo visualizada através de uma janela do terminal.

De um modo geral, as informações que constam das telas (apresentadas parcialmente nas Figuras 4, 5, 6 e 7) foram obtidas dos modelos produzidos pela engenharia reversa.

As informações já existentes sobre o sistema RCS não proporcionava uma visão geral, abstrata, das funções deste sistema. Através do Modelo de Objetos do MAS foi possível visualizar a funcionalidade do RCS de forma clara. Além disso, esse modelo de objetos foi necessário para elaboração do Modelo de Ciclo de Vida do MAS que foi o único utilizado diretamente no desenvolvimento da interface. Por exemplo, o trecho do modelo de ciclo de vida do MAS, descrito abaixo, foi utilizado para o desenvolvimento da tela apresentada na Figura 4.

```
armazenar_versão = ( [-r . [versão]] || [(-l | -u) . [versão]] || [-q . [versão]] || [-f . [versão]] ||
[-k . [versão]] || [-N . nome] || [-M . [versão]] || [-x . sufixos] || [-V . n] || [-m . mensagem] ||
[-s . estado] || [-t . (-texto)] || [-d . [data]] || [-w . login]) . lista_arquivos . #mensagem
```

Para o desenvolvimento da tela da Figura 4, foram consideradas cada uma das opções apresentadas na descrição formal da operação *armazenar_versões*. Opções descritas como mutuamente exclusivas (*-l* que significa recuperar uma versão bloqueando e *-u* que significa recuperar sem bloquear) são implementadas em XView, utilizando desabilitação.

O número de versão ([versão]) que é apresentado como opcional só deve aparecer uma vez quando a operação *armazenar_versões* é executada. Para resolver este problema ele foi colocado na tela como se fosse uma opção.

6.2. Apresentação da Ferramenta G~RCS

Nesta seção, apresentam-se algumas telas da ferramenta G~RCS, mostrando como as principais funçoes de gerenciamento de configuração de software, apoiadas pelo RCS, foram implementadas na ferramenta G~RCS. Os programas relativos à construção da interface encontra-se no documento de programas [Documento2 96]

Na Figura 5 apresenta-se a tela inicial da ferramenta e na Figura 6 é apresentada a tela mostrando os comandos.

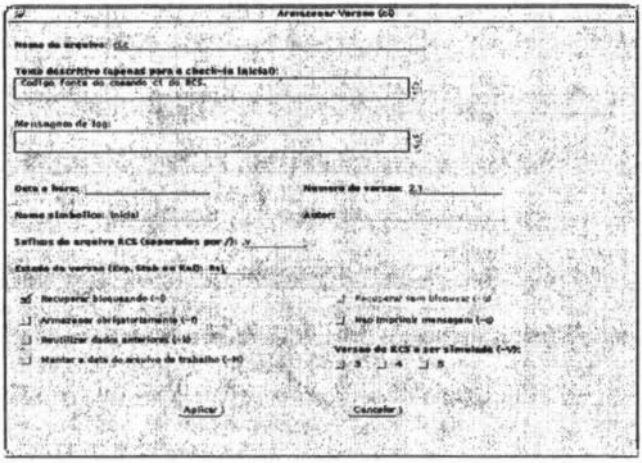


Figura 4: Tela Contendo um Exemplo de Informações Fornecidas pelo Usuário

Figura 5: Tela Inicial da Ferramenta G~RCS

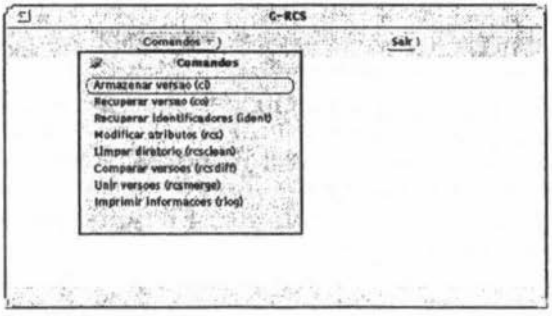


Figura 6: Tela de Apresentação dos Comandos da Ferramenta G~RCS

A função de **controle de versões** da atividade de **controle de configuração de software** é efetuada através dos comandos: *armazenar_versão, recuperar_versão, comparar_versões e unir_versões*

A seguir, são apresentados exemplos de utilização do comando armazenar_versão. Através destes exemplos é possível explicar como outras funções de gerenciamento de configuração de software, apoiadas pelo RCS, foram implementadas.

Um exemplo de uma tela pronta para que o comando seja executado é apresentada na Figura 7. Deve-se notar que por serem mutuamente exclusivas, ao selecionar-se a opção "recuperar bloqueando", a opção "recuperar sem bloquear" foi desabilitada. O mesmo ocorreria com a opção "recuperar bloqueando", se "recuperar sem bloquear" fosse selecionada.

Considerando a interação exibida na tela do comando armazenar_versão, verifica-se que:

- a função de criação de um esquema de identificação dos itens de configuração de software, da atividade de identificação dos itens de configuração de software, está representada pela opção número de versão. Se essa opção não for preenchida, o próprio sistema adiciona o número de versão.
- a função de controle de mudanças, da atividade de controle de configuração de software, é atendida, parcialmente, através do controle de acesso, pela opção recuperar bloqueando. Uma versão pode ser armazenada com ou sem bloqueio.
- a atividade de relato do "status" da configuração de software é atendida, parcialmente, através da utilização de *keywords* com informações como as opções texto descritivo (apenas para o check-in inicial) e mensagem de *log*.

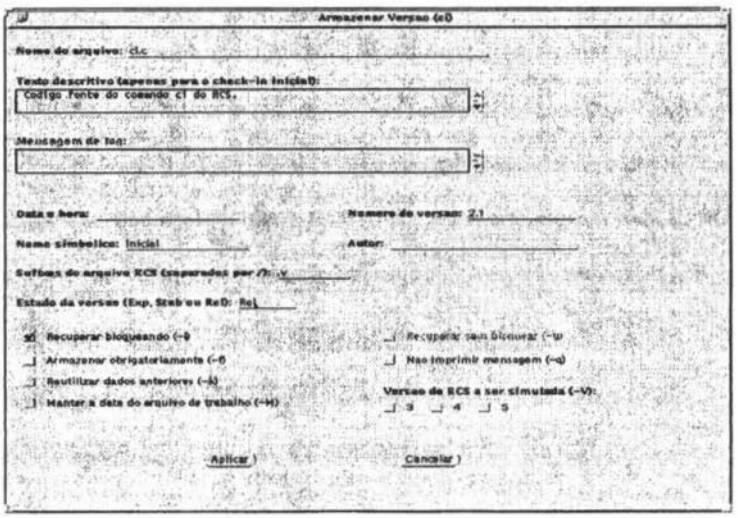


Figura 7:Tela Contendo um Exemplo de Informações Fornecidas pelo Usuário

6. ROTEIRO PARA IMPLANTAÇÃO DE GERENCIAMENTO DE CONFIGURAÇÃO DE SOFTWARE

Para facilitar a compreensão das atividades de gerenciamento de configuração de software, visando uma mais efetiva utilização da ferramenta G-RCS, foi elaborado um roteiro de tarefas para preparação, implantação e controle do processo de gerenciamento de configuração de software (Quadro 4).

Quadro 4: Roteiro para Implantação de Gerenciamento de Configuração de Software

1ª Tarefa: Definição dos Procedimentos Administrativos para Implantação de Gerenciamento de Configuração de Software
Nesse passo deve-se definir os procedimentos administrativos para um efetivo controle da configuração de software.
1ª Etapa: Criar os Cargos de Controlador de Alteração, Administrador da Configuração e Autoridade Controladora de Mudança
Deve-se especificar as qualificações necessárias, as tarefas e as responsabilidades.
2ª Etapa: Definir os Procedimentos de Alteração
Deve-se definir o processo de mudança a ser seguido; criar os formulários de pedido, aprovação e incorporação de alteração.
3ª Etapa: Definir os procedimentos de auditoria
Deve-se definir quando devem ser feitas as auditorias - baseadas em que, quem serão os responsáveis e como devem ser registrados os seus resultados.
2ª Tarefa: Identificação dos Itens de Configuração de Software
A identificação dos itens que irão compor a configuração do software, a especificação dos relacionamentos entre eles e o planejamento das linhas de referências serão consolidados nesse passo.
1ª Etapa: Identificar os itens de informação que serão produzidos.
Nessa etapa deve-se, primeiramente, verificar qual modelo de ciclo de vida está sendo utilizado para o desenvolvimento de software e dentro desse modelo quais itens de informação são gerados. Esses itens devem ser apresentados de forma hieraquica.
2ª Etapa: Selecionar dentre os itens de informação, quais são relevantes para compor a configuração de software.
Utilizando-se as informação produzidas na primeira etapa, deve-se selecionar os itens de informação que devem ser controlados no desenvolvimento e manutenção do software. Para os itens selecionados, deve-se definir os formatos de apresentação dos itens de informação escolhidos para compor a configuração.
3ª Etapa: Especificar os relacionamentos entre os itens de configuração de software.
Deve-se especificar como cada item de configuração se relaciona com os demais, definindo-se procedimentos para controle desse inter-relacionamento.

(continua)

Quadro 4: Roteiro para Implantação de Gerenciamento de Configuração de Software (continuação)

4ª Etapa: Criar um esquema de identificação dos itens de configuração de software.
O esquema deve identificar claramente e unicamente cada item de configuração. Para tanto, deve ser utilizados nomes significativos.
5ª Etapa: Planejar as linhas de referência.
Deve-se planejar em quais tempos do ciclo de vida do software as alterações nos itens de configuração devem ser efetuadas formalmente.
3ª Tarefa: Controle da Configuração de Software
Nesse passo deve-se definir os procedimentos de controle da configuração de software.
1ª Etapa: Definir a lista de acesso aos itens de configuração
Deve-se definir a lista relacionando itens de configuração com usuários que a eles têm acesso.
2ª Etapa: Escolher ferramentas de apoio.
Deve-se escolher uma ou mais ferramentas que darão apoio automatizado a abordagem de controle de versão.
3ª Etapa: Definir os procedimentos para controlar diferentes versões de objetos de configuração
Deve-se escolher os atributos, associados a cada versão dos itens de configuração de software, que serão usados para construir versões específicas de um sistema e os mecanismos do processo de construção.
4ª Tarefa: Relato do "Status" da Configuração de Software
Nesse passo, deve-se definir os procedimentos gerais para o registro e comunicação da situação da configuração de software.
1ª Etapa: Definir como será registrada a situação da configuração de software
A maneira como serão guardadas as informações da configuração de software devem ser definidas nessa etapa.
2ª Etapa: Definir como serão e quais serão os relatórios da situação da configuração de software
Nessa etapa deve-se definir quais relatórios são importantes e como eles devem ser apresentados.
3ª Etapa: Definir como será feita a comunicação da situação da configuração de software.
Nessa etapa, deve ser especificado o caminho seguido pela informação, para que ela chegue a seu destino no tempo desejado.
5ª Tarefa: Implantar e Monitorar o Processo de Gerenciamento de Configuração de Software
Nessa tarefa, devem ser aplicadas as atividades definidas nas quatro tarefas anteriores. O sucesso do gerenciamento de configuração de software depende da efetividade desta etapa.

7. CONCLUSÕES

No contexto de melhoria de qualidade de processo de software, ressalta-se neste trabalho a relevância de um efetivo processo de gerenciamento de configuração de software para o estabelecimento e a manutenção da integridade dos itens de informação que constituem a configuração de um sistema.

Uma grande ameaça à qualidade de software são as mudanças, pois elas têm potencial para introduzir erros ou criar efeitos colaterais que propagam erros [Pressman 95].

Uma maneira de avaliar e controlar o impacto das mudanças nos itens de informação, produzidos durante todo o ciclo de vida do software, é através de um efetivo processo de gerenciamento de configuração de software.

No entanto, a institucionalização de um processo de gerenciamento de configuração de software é difícil, pois exige um conhecimento conceitual relacionado a este processo e além disso, devido a complexidade das tarefas, torna-se necessário o apoio automatizado de ferramentas.

O sistema *Revision Control System (RCS)* é uma ferramenta de domínio público que pode auxiliar o gerenciamento de configuração de software, pois atende, plenamente, uma das principais tarefas que é o controle de versões e parcialmente outras tarefas de gerenciamento de configuração. Esse sistema, porém, é muito pouco utilizado devido a falta de conhecimento dos benefícios que ele proporciona e, principalmente, devido a sua interface que, por ser baseada em comandos de linha, torna difícil sua utilização.

Visando uma maior qualidade de software através da implantação de gerenciamento de configuração, neste trabalho, desenvolveu-se a ferramenta *Graphic Revision Control System (G~RCS)* que consiste de um sistema para auxiliar o gerenciamento de configuração, apoiado no RCS, com uma interface gráfica. Além disso, desenvolveu-se um roteiro genérico para implantação de processo de gerenciamento de configuração em uma organização, no qual procurou-se colocar as principais atividades administrativas e técnicas que devem ser cuidadas para que o processo se torne efetivo.

Para o desenvolvimento da interface, sentiu-se a necessidade da aplicação de engenharia reversa para que informações importantes sobre o RCS pudessem ser recuperadas. Optou-se, então, pelo Método FUSION-RE de engenharia reversa, visto que o mesmo parte de código não orientado a objeto, produzindo abstrações através de modelos orientados a objeto (modelos FUSION).

O método FUSION-RE mostrou-se particularmente útil para a obtenção de uma visão geral da funcionalidade do sistema RCS, o que muito auxiliou o desenvolvimento da interface. No entanto, a aplicação desse método exigiu um grande conhecimento sobre o sistema RCS (adquirido através do uso exaustivo do sistema) e de um profundo conhecimento da linguagem na qual o sistema RCS foi escrito (linguagem C padrão).

A engenharia reversa foi efetuada com a intenção de recuperar informações que seriam utilizadas no desenvolvimento de uma interface gráfica abrangendo as entradas e as saídas do sistema RCS. Entretanto, constatou-se que para o desenvolvimento da interface gráfica, da parte referente às saídas, seria necessário a realização de uma completa reengenharia no código referente às saídas do sistema RCS. Esta reengenharia deveria remodularizar o código de maneira que saídas e processamento ficassem independentes.

A ferramenta G~RCS e o roteiro de implantação de gerenciamento de configuração de software foram submetidos a um pequeno experimento empírico no qual constatou-se que

pessoas inexperientes com os conceitos de gerenciamento de configuração de software não conseguiam utilizar a ferramenta efetivamente. Sendo assim, para que os benefícios advindos através de gerenciamento de configuração sejam conseguidos é necessário que exista, primeiramente, um treinamento introduzindo os conceitos de gerenciamento de configuração de software.

8. REFERÊNCIAS BIBLIOGRÁFICAS

[Berlack 92] BERLACK, H. R. *Software Configuration Management*. 1.ed. New York, John Wiley, 1992

[Bersoft 79] BERSOFT, E.H.; HENDERSON, V. D.; SIEGEL, S. G. Software Configuration Management: A Tutorial. *IEEE Computer*, v.12, n.1, p.6-14, 1979

[Bersoft 84] BERSOFT, E.H. Elements of Software Configuration Management. *IEEE Transactions on Software Engineering*, v.se-1.0, n.1, p.79-87, 1984

[Bollinger 91] BOLLINGER, N.; DART, S, . A. A Critical Look at Software Capability Evaluations. *IEEE Software*, v.8, n.4, p.25-41, 1991

[Buckley 94] BUCKLEY, F. J. Implementing a Software Configuration Management Environment *IEEE Computer*, v.27, n.2, p.56-61, 1994

[Capretz 92] CAPRETZ, M. A. M. COMFORM - A Software Maintenance Method Based on Software Configuration Management Discipline. In: *Conference on Software Maintenance*, Orlando, 1990. Proceedings. p.183-92

[Capretz 94] CAPRETZ M. A. M.; MUNRO M. Software Configuration Management Issues in the Maintenance of Existing Systems. *Software Maintenance: Research and Pratice*, v.6, p.1-14, 1994

[Dart 90] DART, S. A. *A Spectrum of Funtionality CHARETTE in Configuration Management Systems*. Pittsburgh, Software Engineering Institute, Carnegie Mellon University, 1990. (Tech.report CMU/SEI-90-TR-11)

[Dart 91] DART, S. A. Concepts in Configuration Management Systems. In: *International Workshop on Software Configuration Management*, 3, Trondheim, 1991. Proceedings. p.1-18

[Documento1 96] Documento de Trabalho referente à Engenharia Reversa do Sistema Revision Control System (RCS) - Disponível na Biblioteca do ICMSC

[Documento2 96] Documento de Trabalho referente ao Código Fonte da Ferramenta G~RCS - Disponível na Biblioteca do ICMSC

[Freedman 90] FREEDMAN, D. P.; WEINBERG, G. M. *Handbook of Workthroughs, Inspection, and Technical Reviews*. 3.ed. Dorset House, 1990

[Humphrey 88] HUMPHREY, W. S. Characterizing the Software Process - A Maturity Framework. *IEEE Software*, v.5, n.2, p.73-79, 1988

[Humphrey 89] HUMPHREY, W. S. *Managing the Software Process*. 1.ed. Massachesetts. Addison-Wesley, 1989

[Narayanaswamy 87] NARAYANASWAMY, K.; SCACCHI, W. Maintaining Configurations of Evolving Software. *IEEE Transactions on Software Engineering*, v.se-13, n.3, p.324-34, 1987

[Paulk 93a] PAULK M. C. *et al. Capability Maturity Model for Software*. versão 1.1, Pittsburgh, Software Engineering Institute, Carnegie Melon University, 1993. (CMU/SEI-93-TR-24)

[Paulk 93b] PAULK M. C. *et al.* *Key Practices of the Capability Maturity Model.* versão 1.1. Pittsburgh, Software Engineering Institute, Carnegie Mellon University, 1993. (CMU/SEI-93-TR-25)

[Penteado 95] PENTEADO, R.; GERMANO, F.; MASIERO, P.C. Engenharia Reversa Orientada a Objetos do Ambiente Statsim: Método utilizado e resultados obtidos. In: *Simpósio Brasileiro de Engenharia de software,* 9, Anais, 1995, Recife, UFPe, p.345-60, 1995

[Pressman 95] PRESSMAN, R. S. *Engenharia de Software.* 3.ed. Rio de Janeiro. Makron Books, 1995

[Saiedian 95] SAIEDIAN, H. and KUZARA, R. SEI Capability Maturity Model's Impact on Contractors. *IEEE Computer*, v.28, n.1, p.16-26, 1995

[Sanches 93] SANCHES, R. A Influência do Software e de seu Processo de Manutenção no Esforço de Manutenção, São Paulo, 1993. Tese (Doutorado), Faculdade de Economia, Administração e Contabilidade, Universidade de São Paulo.

[Tichy 85] TICHY, W. F. RCS - A System for Version Control. *Software Pratice and Experience*, v.15, n.7, p.637-54, 1985

[Yamashiro 96] YAMASHIRO, F; BERALDI, L. C.; SANCHES, R. *Instruções de Uso de Um Sistema de Controle de Versões RCS - Revision Control System.* São Carlos, Instituto de Ciências Matemáticas de São Carlos - Universidade de São Paulo, 1996. p.39 (Relatórios Técnicos do ICMSC-USP, 44)