UMA FERRAMENTA DE APOIO AO DESENVOLVIMENTO DE SOFTWARE BASEADO EM COMPONENTES

Moisés Pfaffenseller
moises@logycware.com.br
Universidade de Sta. Cruz do Sul
Av. Independência, 2293
Sta. Cruz do Sul/RS - 96815.290

Matheus Pfaffenseller
matheus@logycware.com.br
Universidade de Sta. Cruz do Sul
Av. Independência, 2293
Sta. Cruz do Sul/RS - 96815.290

Eduardo Kroth
kroth@dinf.unisc.br
Universidade de Sta. Cruz do Sul
Av. Independência, 2293
Sta. Cruz do Sul/RS - 96815.290

RESUMO

A programação orientada a objetos e as novas gerações de linguagens para desenvolvimento propiciam um alto grau de modularização e grande flexibilidade, destacando-se conceitos como componentes e reuso de software. Contudo, observa-se que estas ferramentas de modelagem de dados e de processos não possuem uma técnica específica para o gerenciamento de componentes, cuja utilização tende a agilizar o processo de construção de software, uma vez que propicia técnicas seguras de trabalho em grupo para o desenvolvimento de software baseado em componentes, com políticas de manutenção e ferramentas de auxílio ao reuso de software, facilitando também a visualização de hierarquias entre as classes e suas características. O presente trabalho apresenta uma ferramenta que engloba estas funcionalidades, provendo aos usuários da tecnologia de componentes, um gerenciador capaz de suprir as principais deficiências desta técnica de desenvolvimento.

Palavras-chave: gerenciamento de componentes, políticas de manutenção, reuso de software, versões.

1. Introdução

O desenvolvimento de software é um processo intrinsecamente difícil e consumidor de recursos pessoais e financeiros. Como os sistemas têm se tornado cada vez maiores, há um grande trabalho a ser feito no sentido de buscar mecanismos para reduzir sua complexidade.

Existem diversas ferramentas para modelagem de dados e de processos, e algumas de auxílio ao controle de versões de componentes e desenvolvimento de softwares em grupo. Contudo, observa-se que estas ferramentas apresentam uma falta de interoperabilidade entre si, e entre as linguagens de desenvolvimento, ou seja, não há nenhuma ferramenta que integre opções de modelagem, controle de versões, trabalho em grupo e políticas de manutenção com a abrangência de novos conceitos, como componentes e reuso de software.

O problema enfocado neste trabalho é o gerenciamento de componentes, objetivando a organização dos mesmos e conseqüentemente a otimização do processo de construção de software, propiciando técnicas seguras para desenvolvimento de sistemas em grupos de trabalhos, com políticas de manutenção e ferramentas de auxílio ao reuso de software, facilitando também a visualização de hierarquias entre classes de componentes, suas funções, características e fundamentações de cada item.

Baseando-se nestas necessidades, foi desenvolvida uma ferramenta de apoio ao desenvolvimento de software baseado em componentes, tendo como principais características:

- implementar técnicas de armazenamento e busca aprimorando o reuso;
- apresentar a cadeia de dependências entre componentes;
- armazenar informações sobre os componentes, como histórico de manutenções, funções e especificações;
- prover um gerenciamento de versões dos componentes;
- definir políticas de manutenção, controlando os acessos aos componentes com níveis de permissões aos desenvolvedores.

2. Repositório de componentes

O repositório de componentes armazena algumas informações técnicas sobre os componentes, como: classes, hierarquia, métodos, propriedades e eventos; e algumas analíticas como: a função do componente, qual o seu contexto, abrangência, etc. Os desenvolvedores ligados ao repositório têm acesso aos componentes e às suas informações, permitindo a sua utilização e métodos para pesquisa de componentes, objetivando o seu reuso.

2.1 Informações armazenadas no repositório

Para uma fácil visualização da funcionalidade de cada componente, o repositório contém algumas informações, que além de prover um eficiente mecanismo para indexação e recuperação de componentes, armazena dados importantes, que garantem uma boa manutenção dos mesmos [KRO99].

Características: Conforme [GAM95] fica evidenciado como as principais características acerca de componentes as seguintes especificações: *(i) contexto,* procura dar uma idéia geral do contexto em que se aplicam; *(ii) problema,* expõe informações sobre o problema que se propõe a resolver; *(iii) solução,* sintetiza a solução implementada para o problema exposto;

Informações técnicas: São informações úteis para manutenções e análise dos componentes, como: nome, propriedades do arquivo, classificação, criador, etc.

Informações adicionais: informações estruturais dos componentes como eventos, métodos, e propriedades (atributos), além de suas dependências (relação dos arquivos utilizados).

Palavras-chave: Um dos métodos utilizados para a busca e recuperação de componentes é a utilização de palavras-chave associadas a cada componente.

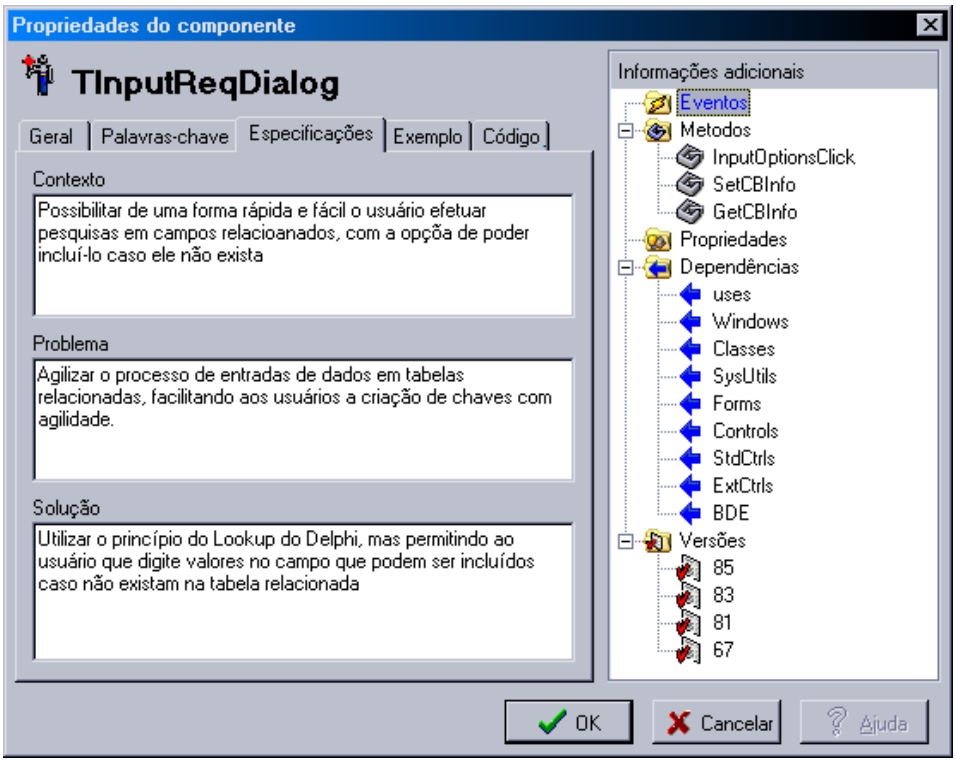

Figura 1 – Interface do software (propriedades do componente)

Exemplo de utilização: Para facilitar a compreensão das funcionalidades dos componentes por parte dos desenvolvedores, criou-se esta propriedade.

Código fonte: Para cada versão, atual ou obsoleta, do componente criada no repositório, armazenam-se os respectivos arquivos do componente. Seu armazenamento se torna necessário para que seja possível a atualização dos componentes nas máquinas dos desenvolvedores e também para que se possa restaurar versões antigas.

Classificação de Componentes: Uma boa forma para organizar a base de pesquisa é separar os componentes conforme o tipo de funções a que ele se destina. Dessa forma, baseada em um estudo sobre *Patterns* [MAR98], inicialmente pode-se identificar alguns padrões como: componentes de estrutura, banco de dados, interface, para desenvolvimento distribuído e de domínio do problema. Contudo, para não restringir a classificação dos componentes às citadas acima, o sistema permite um cadastro de classificações.

2.2 Representação da Hierarquia

Outra funcionalidade oferecida pelo repositório é a representação dos componentes existentes, sua identificação e a exibição da sua cadeia hierárquica, atributos e funções. Uma má escolha da técnica de diagramação inibe o raciocínio enquanto uma boa escolha agiliza o trabalho e melhora os resultados [MAR95]. Quando várias pessoas trabalham no desenvolvimento de um sistema, uma técnica de diagramação formal é uma ferramenta essencial para a troca de informações e idéias entre os desenvolvedores.

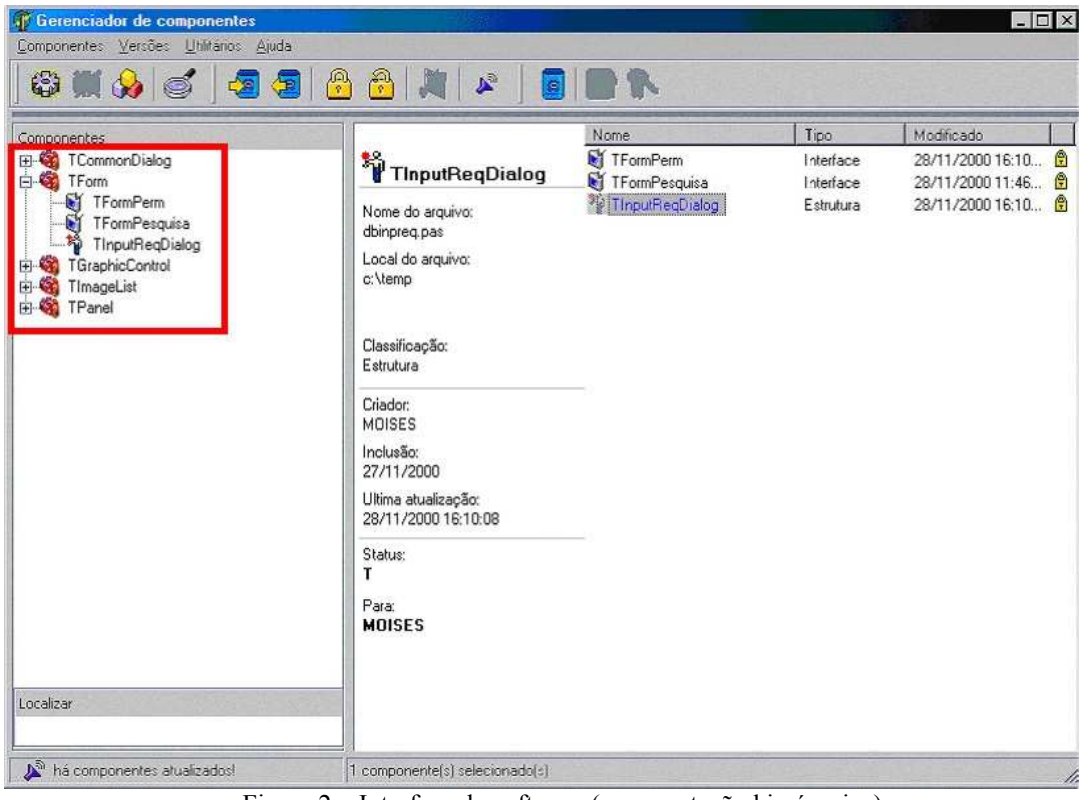

Figura 2 – Interface do software (representação hierárquica)

A diagramação do problema envolve a parte da estrutura hierárquica dos componentes e quais as propriedades, métodos e eventos que esse componente está herdando do seu pai. Cada componente é representado como um nó da árvore e cada nível da árvore como uma descendência hierárquica de suas classes, conforme pode ser visualizado na figura 2.

Um fator importante a ser citado é a relação de dependência que pode ocorrer entre os componentes, o que implica em demonstrar ao desenvolvedor quais os componentes que podem afetar o seu funcionamento, e também quais podem ser afetadas por utilizarem-no, prevenindo desta forma eventuais mudanças que afetem o funcionamento de um sistema.

3. Controle de versão

Para conseguir trabalhar em uma equipe de desenvolvimento de aplicações baseadas em componentes é necessário que se tenha um controle sobre as versões dos componentes. Não utilizar um controle de versões ou ainda utilizar-se de forma incorreta, pode criar problemas que prejudiquem o cronograma do projeto e a qualidade do trabalho [HAV99].

Existem hoje no mercado diversas ferramentas que implementam o controle de versão, automatizando o processo de forma a agilizar e facilitar o trabalho do desenvolvedor. Dentre elas pode-se citar *Borland TeamSource*, *Intersolv PVCS Version Manager*, *Microsoft Visual SourceSafe, Rational Clear Case* e *Concurrent Version System*. Ao analisar as duas primeiras ferramentas, o *PVCS* e o *TeamSource*, pode-se verificar que são ferramentas com finalidades bastante genéricas, tratando os dados em nível de arquivos, sem a preocupação com qual a sua utilidade e suas ligações. Neste trabalho implementou-se o controle de versões de uma forma mais direcionada ao estudo de componentes, proporcionando assim um enriquecimento da ferramenta para este objetivo específico.

3.1 Modelos de gerenciamento do controle de versões

De uma maneira geral, a atividade principal do controle de versões fica a cargo do trabalho em grupo. Visualizado este problema, verifica-se a necessidade de uma técnica que vislumbre uma solução adequada para o processo. Uma das dificuldades encontradas no controle de versões de componentes em comparação aos tradicionais, é que num único arquivo podem ser encontrados vários componentes. A idéia então é tratar todas as operações relacionadas ao seu arquivo, como no caso de bloqueios (*lock* e *unlock*) [1] e atualizações (*check-in* e *check-out*) [1]. A contribuição para um bom funcionamento deste método, é que normalmente se encontram no mesmo arquivo, componentes que possuem referências entre si.

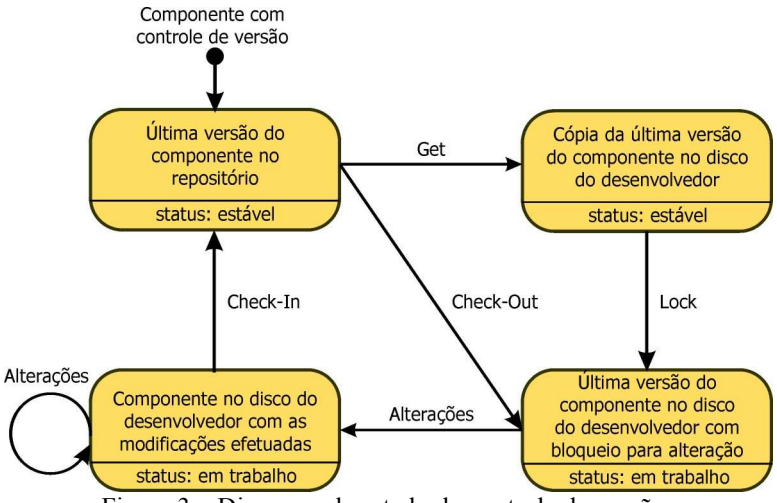

Figura 3 – Diagrama de estado do controle de versões

[1] Os termos *Lock*, *Unlock*, Get, *Check-in* e *Check-out* foram extraídos de [HAV99]

3.2 Análise do modelo proposto

Analisando os modelos para controle de versões encontrados [INP99], conclui-se que o modelo serial é mais viável para o controle de versões de componentes, uma vez que ele proporciona uma maior segurança no gerenciamento das atualizações dos componentes entre o repositório e a cópia local do desenvolvedor.

A operação de inclusão de componentes no controle de versão é feita através da indicação do arquivo onde está o código fonte dos componentes, através deste são identificados as classes nele contidos, bem como suas propriedades, métodos e eventos, entre outras informações citadas na seção 2. Armazenada também a informação da localização dos arquivos no disco local do desenvolvedor, para futuras atualizações.

A Figura 3 mostra o diagrama de estado do controle de versões adotado. Para atualizar a base de componentes local executa-se uma operação de *Get*, que busca as últimas versões existentes no repositório. Para modificar um componente, este deve ser bloqueado através da operação de *lock*, que marca o arquivo a que pertence o componente com o estado de *em trabalho*. A operação de *get* e *lock* pode ser simplificada através de outra, o *check-out*. Ao término do trabalho é executada a operação de *check-in*, que atualiza o componente no repositório, dando-lhe uma nova versão, ou mantém-se a mesma versão.

3.3 Propriedades das versões

As versões de um mesmo componente estão ligadas através de um relacionamento de derivação linear. Cada versão é criada como sucessora de outra, e esta é considerada sua antecessora [NOR96]. Para cada versão também é estabelecido um atributo de *status,* que identifica o seu estado, e partir do seu valor, quais operações poderão ser realizadas sobre este componente. As determinações dos valores para o atributo status de cada versão foram identificadas como: *em trabalho*, *estável* ou *obsoleto*.

Um componente pode ser alterado freqüentemente com um mesmo objetivo, o que acaba gerando muitas versões praticamente iguais, obtendo-se assim uma replicação de dados desnecessária. Para solucionar este problema foi implantado o conceito de *Etiquetas*, com o qual várias alterações são armazenadas em uma única etiqueta, mantendo sempre as informações da última atualização como a versão atual [HAV99].

4. Políticas de manutenção

O princípio das políticas de manutenção é a definição de dois atores, com papéis distintos no relacionamento com o repositório: o administrador e o desenvolvedor. O administrador do repositório tem como função principal gerenciar o trabalho dos desenvolvedores, definindo regras, direitos e permissões que possibilitem controlar as atividades pré-definidas conforme o diagrama de use-case da Figura 4.

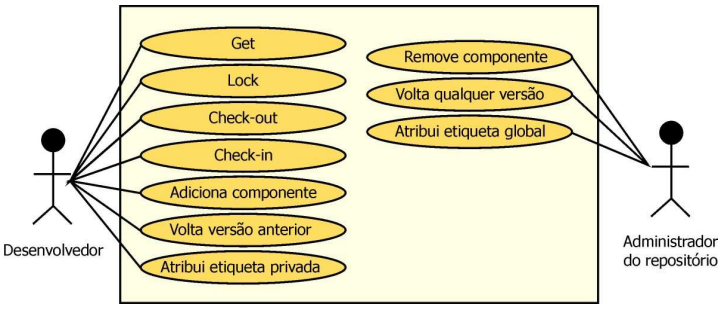

Figura 4 – Diagrama de use-case do controle de versões

Conforme mostrado no diagrama, o desenvolvedor pode executar as tarefas de *get*, *lock*, *unlock*, *check-out*, *check-in*, adição de componente, voltar versão ante-

rior e atribuir etiqueta privada. Já o administrador do repositório pode efetuar todas as operações do desenvolvedor, além de remover componentes, voltar qualquer versão e atribuir etiquetas globais.

A definição de regras permite ao administrador do repositório, estabelecer o tempo que o desenvolvedor pode ficar com um componente bloqueado, automatizando o processo de envio de mensagens para este, avisando-o quando ultrapassar o tempo determinado e obrigando-o a uma renovação do bloqueio, conforme o caso.

5. Conclusões

A inspiração para elaboração deste projeto surgiu das dificuldades encontradas por grupos de trabalho na utilização do desenvolvimento baseado em componentes (DBC), que proporciona muitos benefícios aos seus usuários em relação às demais técnicas, mas que deve ser bem elaborada e executada, tendo como apoio ferramentas gerenciais que solucionem suas deficiências, do contrário pode tornar-se complicada sem fornecer bons resultados.

O trabalho propõe-se a apresentar soluções para o controle de versões e políticas de manutenção, promovendo restrições de acesso em diversos níveis de usuários, bloqueio de componentes que estejam sendo modificados, objetivando o aumento do grau de reuso de componentes, com informações importantes sobre cada um deles, facilitando o trabalho de busca e manutenção.

Como sugestão para trabalhos futuros pode-se indicar a possibilidade de se criar mecanismos para a interligação entre os componentes e os sistemas nos quais eles estão sendo utilizados, além uma ampliação das políticas de manutenção, criando-se mecanismos de controle adaptáveis a cada equipe de trabalho, como tempo de bloqueio, prazos de atualização a serem cumpridos, entre outros.

Referências Bibliográficas

[GAM95] GAMMA, E., HELM, R., JOHNSON, R., VLISSIDES, J. *Design Patterns elements of reusable object-oriented software*. Addison-Wesley, 1995. 395p.

[HAI97] HAINES, G. CARNEY, D., FOREMAN, J. *Component-Based Software Development / COTS Integration.* Acessado em 21 abr. 2000. Disponível na Internet http://www.sei.cmu.edu/str/descriptions.

[HAV99] HAVEWALA, Aspi. *The version control process*. Dr. Dobb's Journal, São Francisco, n. 299, p 100-111, mai. 1999.

[INP99] INPRISE Corporation. Arquivo de Ajuda Borland TeamSource. Versão 1.0.5.38, 1999.

[KRO99] KROTH, E. et al. Software Assistente no Uso de Componentes. **Proceeding of the** XII-Simpósio Brasileiro de Engenharia de Software - Sessão de Ferramentas, outubro, 1999.

[MAR95] MARTIN, J., ODELL, J. *Análise e Projeto Orientados a Objeto*. Makron Books do Brasil Edit. Ltda., São Paulo, 1995.

[MAR98] MARTIN, ROBERT C. *Pattern Languages of Program Design 3.* Addison-Wesley, 1998. 632p.

[NOR96] NORONHA, MARILENE A. *Um mecanismo para controle de versões num sistema de documentos.* Acessado em 16 mai. 2000. Disponibilizado na Internet http://www.inf.ufrgs.br/gdoc/versoes.html.

[POS92] POSTON R., SEXTON M. *Evaluating and Selecting Testing Tools*. *IEEE Software,* 1992.

[VAL95] VALETTO, G., KAISER, G.E. "Enveloping Sophisticated Tools intoComputer-Aided Software Engineering Environments," 40-48. *Proceedings of 7th IEEE International Workshop on CASE.* Toronto, Ontario, Canadá, Julho, 1995. Los Alamitos, CA: IEEE Computer Society Press, 1995.