LockED: Uma Abordagem para o Controle de Alterações de Artefatos de Software.

Hugo Vidal Teixeira
Leonardo Gresta Paulino Murta
Cláudia Maria Lima Werner
COPPE / UFRJ – Departamento de Sistemas e Computação
Caixa Postal: 68511 – CEP 21945-970 – Rio de Janeiro – RJ
Fax / Telefone: 5521 590-2552
{hvidal, murta, werner}@cos.ufrj.br

Abstract

This work describes an approach to support change control in distributed development of software projects. This approach differs from existing ones because it does not only handle source code, but also other artifacts produced within other stages of the software development process. The proposed approach uses Internet resources for communication and exchange of artifacts among the developers and the server.

1. Introdução

Os projetos de desenvolvimento de software contemporâneos têm progressivamente aumentado de tamanho e complexidade, sendo cada vez mais comum sua realização por equipes de médio porte (entre dez e vinte desenvolvedores) e grande porte (acima de vinte desenvolvedores). Com as facilidades de comunicação proporcionadas pela Internet, a necessidade de experiência em diversas áreas de conhecimento e a pressão por cronogramas mais restritos, alguns projetos são realizados por diversas equipes trabalhando concorrentemente. Ainda mais, as dificuldades de reunir os especialistas necessários em um mesmo local físico e a delegação do desenvolvimento de determinados componentes para outras empresas, são exemplos de fatores que podem exigir que as equipes participantes de um projeto estejam geograficamente distribuídas [5].

A existência de mecanismos eficientes de comunicação, como a Internet, não é suficiente para solucionar os problemas de desenvolvimento concorrente de projetos de software. Ao contrário, o novo cenário traz novos problemas para o processo de desenvolvimento. Por exemplo, o trabalho concorrente de equipes geograficamente distribuídas dificulta o controle de alterações nos componentes de um projeto em desenvolvimento [11]. Mesmo quando precedida de uma precisa definição das interfaces entre os componentes, a realização de um projeto pode exigir que diversos desenvolvedores alterem simultaneamente os mesmos componentes. Esta situação exige a adoção de políticas para manter a consistência entre os componentes da versão atual do projeto ou permitir que essa consistência seja posteriormente restituída.

Atualmente, a grande maioria das abordagens para solucionar esse tipo de problema é focada em código-fonte. Contudo, os sistemas mais complexos demandam um esforço adicional para a sua compreensão, tornando necessário o uso de artefatos que descrevem aspectos não representados pelo código-fonte, agregando informação em um nível de abstração mais elevado. Consideramos como artefatos de software (ou simplesmente artefato) qualquer produto gerado em qualquer etapa do processo de desenvolvimento de software. Desta forma, uma abordagem para solucionar este problema deve permitir o controle de alterações de todos os artefatos de software, como por exemplo: classes (com o seu código-fonte), diagramas de classe, diagramas de seqüência, diagramas de caso de uso, etc.

O objetivo deste artigo é descrever uma abordagem que visa solucionar o problema de controle de alterações de artefatos de software no desenvolvimento distribuído de projetos de software. Inicialmente, desenvolvemos uma abordagem focada no controle de alterações em código-fonte (apoiada pela ferramenta Token), porém esta não se mostrou uma boa alternativa para o entendimento do projeto como um todo. Posteriormente, essa abordagem foi aprimorada para permitir o controle de alterações de todos os artefatos de software produzidos em um projeto (apoiada pela ferramenta LockED, que pode ser considerada uma evolução da ferramenta Token).

Este artigo está organizado em três seções, além desta introdução. A segunda seção trata das abordagens existentes para solucionar o problema de controle de alteração de artefatos de software em ambientes distribuídos. A terceira seção apresenta a abordagem proposta. Finalmente, a seção 4 apresenta as conclusões e perspectivas futuras para este trabalho.

2. Ferramentas de Apoio ao Desenvolvimento Distribuído de Software

As ferramentas para controle de alteração de software disponíveis atualmente são focadas em artefatos de software armazenados em arquivos do sistema operacional (SO). Dentre elas, podemos citar:

- **Code Co-op** [13]: Permite o controle de versões de artefatos em desenvolvimento de forma transparente em relação a localização geográfica dos desenvolvedores, sem a necessidade de um servidor central, armazenando os artefatos de forma distribuída entre os desenvolvedores. Baseia o seu sincronismo em serviços de e-mail, rede local ou disquete. Sua licença de uso custa proporcionalmente ao número de desenvolvedores.

- **PVCS Version Manager** [7]**:** Ajuda a organizar, proteger e gerenciar artefatos de software, utilizando interface gráfica baseada no Windows Explorer. Contém mecanismos para recuperação de erros e alterações perdidas. Sua versão completa permite o controle de versões em ambiente *web*.

- **Visual SourceSafe 6.0** [8]**:** É integrado com todos os ambientes de programação da Microsoft e permite o gerenciamento da equipe juntamente com os artefatos desenvolvidos por ela. Esta ferramenta é altamente escalável e pode ser usada facilmente por equipes com pouca experiência técnica. Existem extensões que permitem o seu uso através da Internet, como por exemplo o SourceOffSite [4].

- **Tlib** [2]**:** Uma ferramenta rápida, que tem a estrutura de armazenamento aberta e permite a importação de projetos que já utilizam outras ferramentas de controle de alterações. Suporta a junção (*merge*) simultânea de diferentes versões do mesmo artefato de código.

- **GP-Version** [12]**:** Idealizado inicialmente para suportar o controle de alterações de código escrito em Delphi e C++ Builder, foi posteriormente estendido para suportar a linguagem Visual Basic. É baseado em extensões add-ins e foi escrito em Delphi.

Essas ferramentas fazem o controle de modificações em textos, páginas html, código-fonte ou qualquer outro artefato que possa ser relacionado um para um com arquivos do SO. Entretanto, para que modelos possam ser armazenados e alterados sob o controle de uma ferramenta, esta não pode ser baseada em arquivos do SO, pois deve conseguir lidar internamente com o arquivo que contém os modelos a serem controlados.

Apesar da existência de várias ferramentas para controle de alterações em artefatos de software (a nível de código-fonte), como descrito nesta seção, optamos inicialmente por construir uma nova ferramenta, denominada Token. Os principais motivos que nos levaram a construir mais uma ferramenta, ao invés de utilizar uma existente já foram: o seu custo reduzido, por ser baseada em uma plataforma gratuita (Linux, Apache, PHP3 e MySQL), ter como meio de comunicação a Internet, permitindo um desenvolvimento totalmente distribuído, e ser independente de plataforma, tanto no lado cliente como no lado servidor. A sub-seção a seguir descreve mais detalhadamente esta ferramenta.

2.1. Token

A ferramenta Token foi desenvolvida para apoiar o desenvolvimento concorrente de projetos de software, auxiliando na resolução dos problemas de controle de alterações nos componentes do projeto [10]. Suas principais funcionalidades são: o cadastramento dos desenvolvedores participantes do projeto, a troca de informações entre estes desenvolvedores e o controle de alterações nos componentes do projeto. Token foi desenvolvida em ambiente Linux, utilizando a linguagem de script PHP3, o banco de dados MySQL e o servidor *web* Apache. A ferramenta é acessível via um navegador Internet, sendo portanto independente tanto da plataforma cliente quanto da plataforma servidora.

Para facilitar a comunicação entre os desenvolvedores, Token oferece um quadro de mensagens, indexadas por assuntos. Sempre que uma mensagem é enviada através da ferramenta Token, os desenvolvedores cadastrados recebem uma notificação por e-mail. A notificação indica o assunto da mensagem, seu tipo e contém um link para a página de mensagens do Token. Os tipos de mensagem utilizados no Token são: pergunta, resposta, informação e urgente. Na versão atual da ferramenta, os tipos de mensagem são meramente informativos, permitindo ao leitor uma rápida identificação do objetivo da mensagem e a criação de filtros em seu e-mail.

A abordagem adotada para o controle de alterações de artefatos no Token consiste na divisão de um projeto em diversos componentes[1] atômicos de software. Os componentes são acessíveis pelos desenvolvedores através da ferramenta, que permite que apenas um desenvolvedor altere um determinado componente a cada instante. Um componente é o elemento de menor granularidade controlado pela ferramenta.

Token segue a mesma abstração dos protocolos de comunicação das redes de computadores *Token Ring* e *Token Bus* [15]. Nestas redes, diversos computadores compartilham um meio de comunicação comum, que deve ser utilizado por um computador de cada vez. Para sincronizar os acessos à rede, um pacote de dados padronizado, denominado token, permanece em tráfego na rede enquanto nenhuma mensagem é transmitida. Quando um computador deseja transmitir uma mensagem, ele retira o token da rede, envia sua mensagem, e, por fim, restaura o token, permitindo que outros computadores enviem suas mensagens.

Da mesma forma, através da ferramenta Token, os desenvolvedores podem alocar ou desalocar componentes. Cada componente possui um token, que se encontra disponível quando o componente não está reservado para nenhum desenvolvedor. Quando um desenvolvedor decide alterar um componente, ele deve pegar o token do componente. Sendo o token único por componente, apenas um desenvolvedor poderá alterar um determinado componente por vez. Se o componente desejado foi previamente reservado para um desenvolvedor, este deve liberar seu token para que outro desenvolvedor tenha permissão para alterá-lo. Token controla a alocação de componentes no nível lógico, limitando o acesso aos componentes à presença de seus tokens, e no nível físico, disponibilizando os arquivos com o código fonte dos componentes.

A desalocação de componentes é um processo complexo, onde o Token reconstrói a estrutura do projeto a partir dos componentes existentes em sua base e do componente recentemente liberado. Para que um componente seja desalocado, é necessário que ele tenha sido previamente alocado pelo desenvolvedor corrente e que esse desenvolvedor informe o nome do componente, o arquivo contendo seu código fonte e uma descrição das alterações que o componente sofreu.

O diagrama apresentado na Figura 1 exibe a forma de iteração do Token com as suas bases de dados e com o usuário, representado por uma conexão http vinda de uma máquina cliente.

Token está sendo aplicado, com resultados satisfatórios, no desenvolvimento do Projeto Odyssey [16] e do Projeto GOA++ [6]. Entretanto, conforme dito anteriormente, sua abordagem é baseada em controle de versões de código-fonte (como nas demais ferramentas apresentadas nessa seção). A seguir descrevemos uma extensão desta ferramenta que suporta o controle de alterações de artefatos de software de mais alto nível, mais especificamente diagramas UML.

---

[1] No contexto do Token, um componente é um arquivo compactado (ZIP), contendo um ou mais arquivos de código-fonte. Em projetos orientados a objetos, um componente pode conter desde uma classe até um *framework* inteiro, contendo um conjunto de classes.

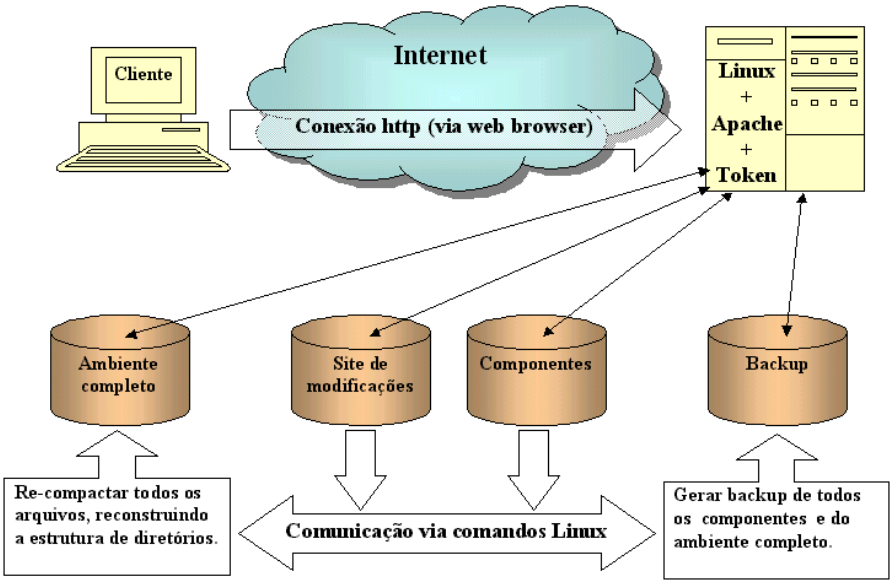

Figura 1: Diagrama de iteração do Token

3. A ferramenta LockED

Segundo os mais conhecidos princípios da engenharia de software, o desenvolvimento de sistemas é uma atividade muito mais complexa do que parece. A lógica envolvida por trás das funcionalidades da aplicação, juntamente com todas as suas classes, funções e interações torna o entendimento do problema uma tarefa muito complexa e trabalhosa. Da mesma forma, um dos erros mais cometidos pelos desenvolvedores de sistemas é o fato deles considerarem o código do sistema o único produto do processo que merece atenção especial. Ao negligenciarem os modelos que compõem a arquitetura da aplicação, e que foram a base da análise e do projeto, eles estão perdendo valiosas informações sobre a complexa lógica que explica todo o comportamento do software. De fato, estão criando um problema que os futuros mantenedores do produto precisam enfrentar, que é a dificuldade do rastreamento do código e o seu correto entendimento.

Como vimos anteriormente, a ferramenta Token somente se preocupa com o código-fonte do sistema em desenvolvimento, não abrindo espaço para os demais produtos do processo de desenvolvimento no qual a equipe está inserida. Isto exige que o gerente da equipe procure por outras ferramentas, fazendo com que o sucesso do projeto se torne um fator de organização e comprometimento dos seus integrantes.

Analisando por outra perspectiva, a cooperação existente em todo o trabalho de equipe muitas vezes presencia grandes contratempos. A maior parte das dificuldades concentra-se na dispersão da informação e na maneira com a qual esta é compartilhada pelo grupo. Muitas pesquisas já contribuíram para o assunto, trazendo métodos, ferramentas e práticas de trabalho que facilitam a coordenação dos membros da equipe [14, 3]. Entretanto, a complexidade envolvida neste trabalho existe e jamais deixará de ser considerada um desafio.

Observando através de um cenário mais específico, o desenvolvimento de sistemas dentro de um determinado domínio de aplicação apresenta particularidades sobre o trabalho em equipe. No contexto da engenharia de domínio [1], a modelagem do domínio e sua posterior evolução exigem um grande esforço tanto da equipe de desenvolvimento quanto dos especialistas na área. Os produtos resultantes desta análise (artefatos do domínio) precisam estar disponíveis para todos os desenvolvedores, apoiando as suas discussões e fundamentando as decisões sobre o desenvolvimento de um sistema em particular. De fato, quando a equipe está toda concentrada em um único local – cenário típico de pequenos projetos – as interações entre estes membros são razoavelmente fáceis de serem controladas. Por outro lado, quando a equipe se apresenta geograficamente distribuída, a concorrência sobre as alterações dos artefatos de software requer mais cautela.

Dentro deste contexto, foi desenvolvida a ferramenta LockED, a qual visa controlar e impor ordem sobre a criação e as alterações de artefatos do domínio, disponibilizando-os para alocação e desalocação. Seguindo o mesmo princípio da ferramenta Token, um usuário somente pode alterar um artefato quando este já estiver previamente alocado ao mesmo através da ferramenta LockED. A idéia básica é que haja uma base oficial de informações sobre o domínio em um servidor conectado a Internet / Intranet, a qual será acessada por toda a equipe. Cada integrante trabalhará em sua própria estação de trabalho utilizando a infra-estrutura de reutilização provida pelo seu ambiente de modelagem, realizando atividades de modelagem, consultas e instanciando aplicações a partir destes artefatos.

Neste trabalho, integramos a ferramenta LockED a infra-estrutura de reutilização de software fornecida pelo projeto Odyssey, a qual é descrita brevemente na próxima seção.

3.1. A Infra-Estrutura Odyssey

O uso de técnicas de reutilização nas fases iniciais do desenvolvimento de aplicações baseado em componentes facilita a reutilização de componentes em fases mais avançadas do desenvolvimento (ex. implementação). A infra-estrutura Odyssey visa apoiar uma reutilização efetiva durante todo o desenvolvimento de software, provendo métodos, ferramentas e procedimentos adequados para a especificação de modelos e aplicações do domínio.

A infra-estrutura Odyssey pode ser vista como um arcabouço onde modelos conceituais do domínio, modelos de projeto (arquiteturas de software específicas do domínio) e modelos implementacionais (*frameworks*) são especificados para um determinado domínio, disponibilizando componentes reutilizáveis neste domínio. Durante o processo de desenvolvimento de aplicações, estes componentes são refinados e adaptados ao contexto de um projeto específico.

Os modelos de domínio constantes da infra-estrutura são especificados e posteriormente evoluídos segundo as atividades de um processo para construção de componentes reutilizáveis, denominado Odyssey-ED (BRAGAet al., 1999). Estes componentes são reutilizados durante o desenvolvimento de aplicações através de um processo denominado Odyssey-EA [9].

As atividades destes processos são suportadas por um conjunto de ferramentas, dentre elas, ferramentas para definição dos modelos, ferramentas para armazenamento e recuperação dos modelos, e ferramentas para navegação e reutilização dos modelos. A Figura 2 exibe a ferramenta de diagramação de casos de uso em funcionamento. O ambiente Odyssey foi desenvolvido na linguagem de programação Java e está equipado com toda a infra-estrutura necessária à modelagem dos domínios e das aplicações que a equipe precisa para executar suas atividades.

Os principais usuários desta infra-estrutura são o engenheiro do domínio, o especialista do domínio e o engenheiro de software responsável pelo desenvolvimento de aplicações no domínio. O engenheiro do domínio e o especialista utilizam a infra-estrutura, principalmente, para especificar e evoluir os conceitos do domínio. O engenheiro de software utiliza a infra-estrutura para obter conhecimento sobre o domínio da aplicação e reutilizar este conhecimento na especificação de sua aplicação.

A infra-estrutura Odyssey disponibiliza três formas de persistência das suas informações: Serialização, GOA++ (Gerente de Objetos Armazenados) e MOR (Mapeador Objeto Relacional). A Figura 3 exibe o relacionamento entre o Odyssey e suas formas de armazenamento.

A Serialização é um mecanismo da linguagem Java que permite o armazenamento de objetos e todos os demais objetos relacionados (recursivamente) em um arquivo do sistema operacional. Essa abordagem, apesar de mais simples, não é recomendada para o armazenamento de uma grande quantidade de informação.

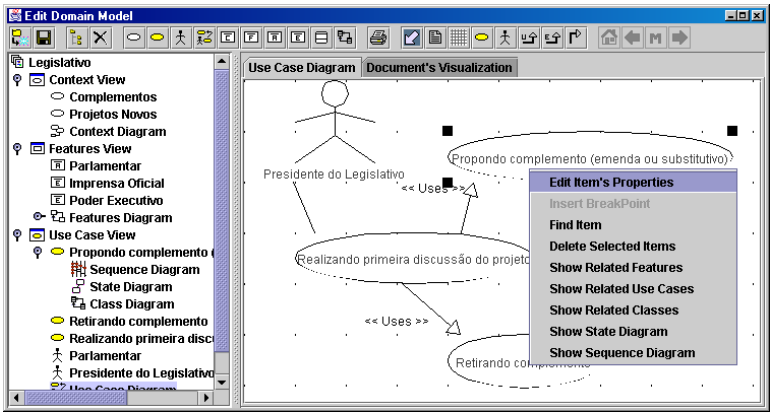

Figura 2: Ferramenta de diagramação de casos de uso

Uma outra opção para o armazenamento no Odyssey é o GOA++. A utilização do GOA++ como repositório de informações do Odyssey representa uma boa opção pois funciona no mesmo paradigma do Odyssey (Orientação a Objetos). No entanto, por questões de flexibilidade, foi ainda disponibilizado o MOR, que é uma interface genérica de armazenamento, de forma que os dados possam ser armazenados em qualquer SGBD com interface para ODBC (vide Figura 3). Assim, o armazenamento não fica dependente de um só tipo de SGBD.

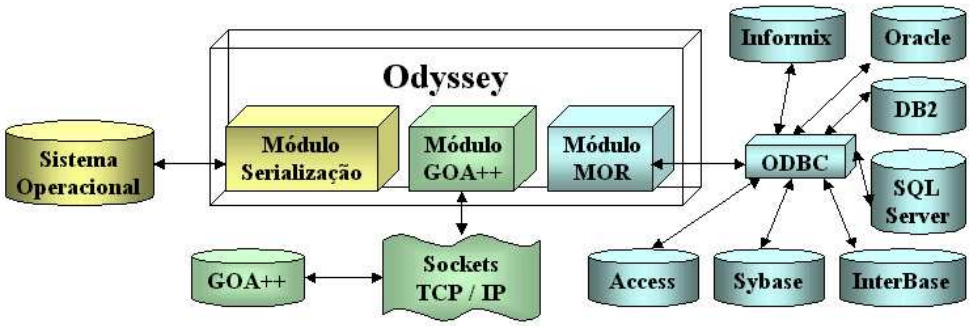

Figura 3: Formas de armazenamento no Odyssey

3.2. Arquitetura do LockED

Pelo fato da ferramenta LockED representar uma extensão da ferramenta Token, suas arquiteturas apresentam muitas características em comum, como a presença do servidor e dos clientes se comunicando via rede Internet ou intranet. A comunicação das estações de trabalho Odyssey com a ferramenta LockED é feita utilizando-se arquivos exportados e importados por ambos. A Figura 4 ilustra a interação entre as ferramentas.

A realização do processo está baseada na alocação e desalocação dos artefatos do domínio. Para entendermos melhor como funciona este mecanismo, vamos acompanhar o processo desde o início. Suponha que uma equipe precise trabalhar com o domínio de telecomunicações e nada se encontra modelado. Primeiramente, alguém deverá criar o domínio no ambiente Odyssey, modelar o que desejar, e exportá-lo através do sistema LockED. A partir deste ponto, todos os outros membros da equipe poderão adquiri-lo no servidor (alocando o que desejarem) e importá-lo em suas estações de trabalho. Deste ponto em diante, cada usuário trabalhará em sua própria estação de trabalho, modificando o que estiver alocado em seu nome, além de poder criar novos artefatos. Quando o usuário desejar, ele poderá exportar o domínio (para um arquivo) fornecendo-o posteriormente ao sistema LockED, de forma que este atualize a sua base de dados. A arquitetura interna do LockED é descrita na Figura 5.

O fato de o ambiente Odyssey ter sido construído na linguagem Java foi um fator decisivo na construção e integração do sistema LockED, o qual foi desenvolvido na linguagem Java e JSP (Java Server Pages). A tecnologia JSP se apresentou como uma vantagem considerável nesta implementação, uma vez que esta possibilita a construção de sistemas *web* totalmente compatíveis com a linguagem Java. Assim, aproveitamos

toda a estrutura provida pelo Odyssey para disponibilizar e armazenar os artefatos no servidor (isto inclui o acesso às bases de dados GOA, MOR e serialização).

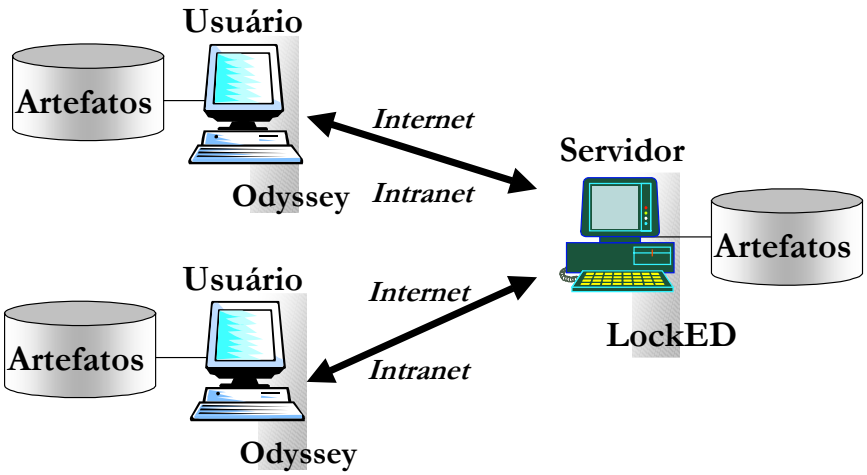

Figura 4: Visão geral da comunicação entre as ferramentas

A comunicação entre o servidor e todas as estações de trabalho é feita através da importação e exportação de arquivos. Estes arquivos, na verdade, são os domínios do ambiente Odyssey que são serializados para um arquivo temporário. Pelo fato de grande parte do sistema LockED ser constituído pelas classes do Odyssey, ambos os sistemas são capazes de abri-los e computá-los.

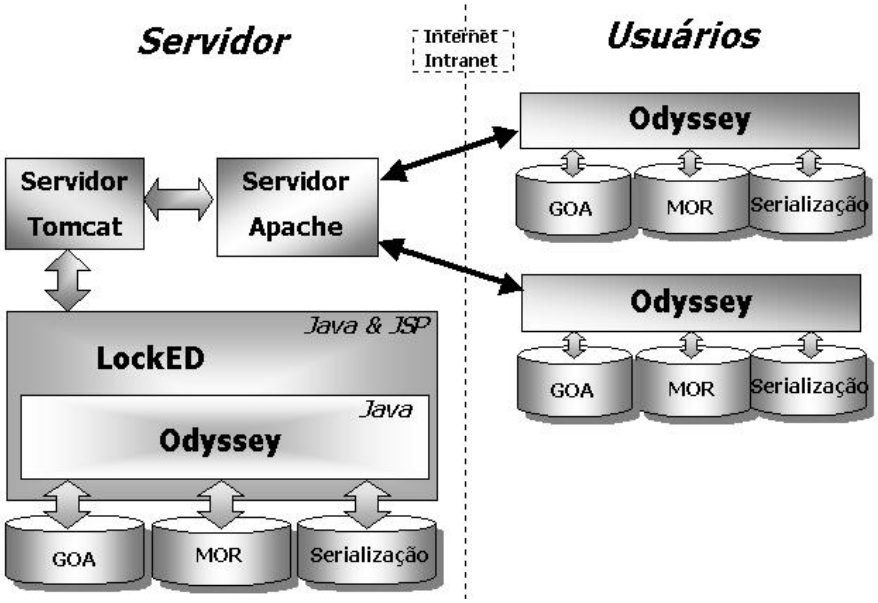

Figura 5: Arquitetura interna do LockED

Para disponibilizar o sistema na Internet / Intranet, foi necessário utilizar o servidor de JSP Tomcat, o qual deve estar acompanhado do servidor Apache que tratará as solicitações dos usuários.

3.3. Interface do LockED

A interface do sistema LockED é gerada dinamicamente a partir das páginas em JSP que estão presentes no servidor. Isto reflete em uma utilização consistente das informações inseridas no sistema, mantendo todas as páginas constantemente atualizadas.

Todos os usuários da ferramenta possuem um *login* e uma senha de acesso, fazendo com que estes sejam identificados e reconhecidos pelo sistema. A identificação de cada usuário é fundamental, na medida em que eles estão sempre alocando e desalocando artefatos da base de dados do servidor. Portanto, o sistema é capaz de associar um usuário a um artefato toda vez que uma alocação é feita.

Após o *login* inicial, o sistema retorna a tela que pode ser vista na Figura 6. O menu presente no lado esquerdo da tela possui *links* para todas as funcionalidades do sistema.

Figura 6: Página principal da ferramenta LockED

3.3.1. Alocação de Artefatos

A alocação dos artefatos pode ser feita seguindo o link "Alocação" presente no menu. Feito isso, o usuário deverá escolher em qual domínio ele pretende realizar a operação. Após a escolha, todos os artefatos pertencentes ao domínio estarão apresentados na tela. O fato interessante desta apresentação é a maneira com a qual os artefatos são apresentados para o usuário. Isto é feito da mesma forma que o ambiente Odyssey os apresenta em Java, porém o LockED o faz em HTML (acompanhado de Javascript). Um exemplo desta tela com a árvore de elementos semânticos está ilustrado na Figura 7.

Todos os artefatos que não estão alocados possuem uma caixa de seleção ao seu lado que serve para o usuário indicar o que ele pretende alocar dentro do domínio.

Figura 7: Árvore de Artefatos do LockED

Caso um artefato já esteja alocado a um usuário, a caixa de seleção não aparece. Entretanto, neste caso, aparecerá, ao lado do nome do artefato, o nome do seu dono entre parênteses e um cadeado simbolizando a restrição sobre o acesso àquele item. Após o usuário selecionar o que ele deseja alocar, o sistema fornece a ele um arquivo para *download*, o qual contém todo o domínio que deverá ser importado na infra-estrutura Odyssey, presente na estação de trabalho do usuário. A importação do arquivo invoca um algoritmo de junção detalhado na seção 3.4.

3.3.2. Desalocação de Artefatos

Uma vez tendo o domínio em sua base local, cada integrante tem a permissão para criar e alterar artefatos conforme lhe foi alocado, isto é, o Odyssey só permitirá que um item seja alterado quando este estiver alocado ao nome do usuário. As constantes mudanças realizadas por cada membro da equipe criarão bases de dados distintas em cada estação. A unificação destas bases é feita pelo sistema LockED no momento em que os usuários fornecem o domínio de volta, através do arquivo exportado pelo Odyssey. Os algoritmos de unificação das bases estão também descritos na seção 3.4.

Após completar suas mudanças, o usuário deverá devolver o domínio ao servidor do LockED, o que resultará na desalocação e atualização de todas as informações sobre os artefatos alocados.

3.4. Detalhamento Técnico

A complexidade envolvida neste sistema está na comparação dos artefatos existentes nas diferentes bases de dados que estão espalhadas pela equipe e no servidor. Toda vez que um arquivo é importado pelo Odyssey ou exportado para o LockED, uma série de comparações recursivas são realizadas, visando efetuar uma junção das árvores de elementos semânticos existentes em ambas as bases.

Estas árvores de elementos semânticos são, na verdade, um conjunto de artefatos que descrevem um domínio. Elas funcionam como se fossem "arquivos e diretórios" de um sistema operacional. A Figura 8 ilustra como esta árvore é representada dentro da estrutura Odyssey.

Existe uma série de detalhes que foram considerados durante a construção dos algoritmos, como o fato de um usuário não poder perder suas mudanças quando este importa um domínio mais atualizado em sua estação de trabalho.

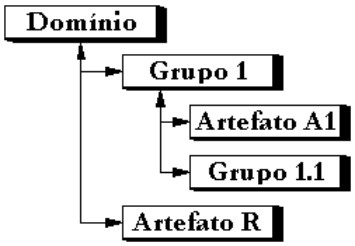

Figura 8: Exemplo de árvore de elementos semânticos

Estes detalhes contribuíram para o desenvolvimento de dois algoritmos básicos:

1. *Importar um arquivo do Odyssey no LockED*
2. *Importar um arquivo do LockED no Odyssey*

Esta distinção precisou ser feita porque a base que está no servidor sempre será mais importante daquelas mantidas pelos integrantes em suas estações de trabalho. Assim, ambos os algoritmos possuem um foco nas informações do servidor, mas sempre considerando a importância das modificações que o usuário possui em sua própria máquina.

Cada nó de uma árvore de elementos semânticos possui dois atributos fundamentais para o processo de alocação e desalocação de artefatos: *criação* e *dono*. O atributo *criação* é uma variável boleana que serve para informar se aquele nó foi criado recentemente ou não. Sempre que um artefato é criado no ambiente Odyssey, este possuirá o valor *verdadeiro*. Entretanto, toda vez que um domínio é importado pelo servidor, ele atribui o valor *falso* a esta variável, indicando que aquele nó já está na base oficial (servidor). O outro atributo, *dono*, é

uma variável da classe Usuário que representa o dono do nó em questão. Sempre que um usuário aloca um artefato da arvore, seu código é inserido nesta variável, representando a associação.

Em função destas duas variáveis, a infra-estrutura Odyssey tem uma missão peculiar no controle dos acessos e modificações nos domínios. Não se pode permitir que um usuário apague um artefato que não lhe pertence. Esta restrição é fundamental para manter a consistência do domínio e para garantir a integridade da informação que lá está representada.

Apresentamos na Figura 9 o algoritmo para importação, utilizado quando um usuário exporta o domínio presente em sua estação de trabalho e o importa no sistema LockED que está no servidor.

```
(1) Faça o Upload da árvore
(2) Verifique se o domínio existe. Caso não exista, crie-o.
(3) Para cada nó X da árvore cujo dono é o usuário, faça:
   (3.1) busque o mesmo nó (antigo) na base do LockED
   (3.2) Se encontrar o nó X:
      (3.2.1) coloque o novo nó no lugar do antigo, preservando o seu pai
      e os seus descendentes
   (3.3) Senão (o nó foi criado)
      (3.3.1) busque o pai do nó X criado na árvore do LockED
      (3.3.2) insira o nó X criado no pai encontrado
      (3.2.2) Retire o nome do usuário do nó X (variável dono)
(4) Para cada nó Y presente na árvore da base LockED cujo dono ainda é o
   usuário, faça:
   (4.1) Apague o nó Y
```

Figura 9: Algoritmo de importação do Odyssey para o LockED

Para entendermos melhor este algoritmo, apresentamos na Figura 10 um exemplo onde o usuário *Hugo* aloca um artefato no domínio que está no servidor e o importa em sua estação de trabalho.

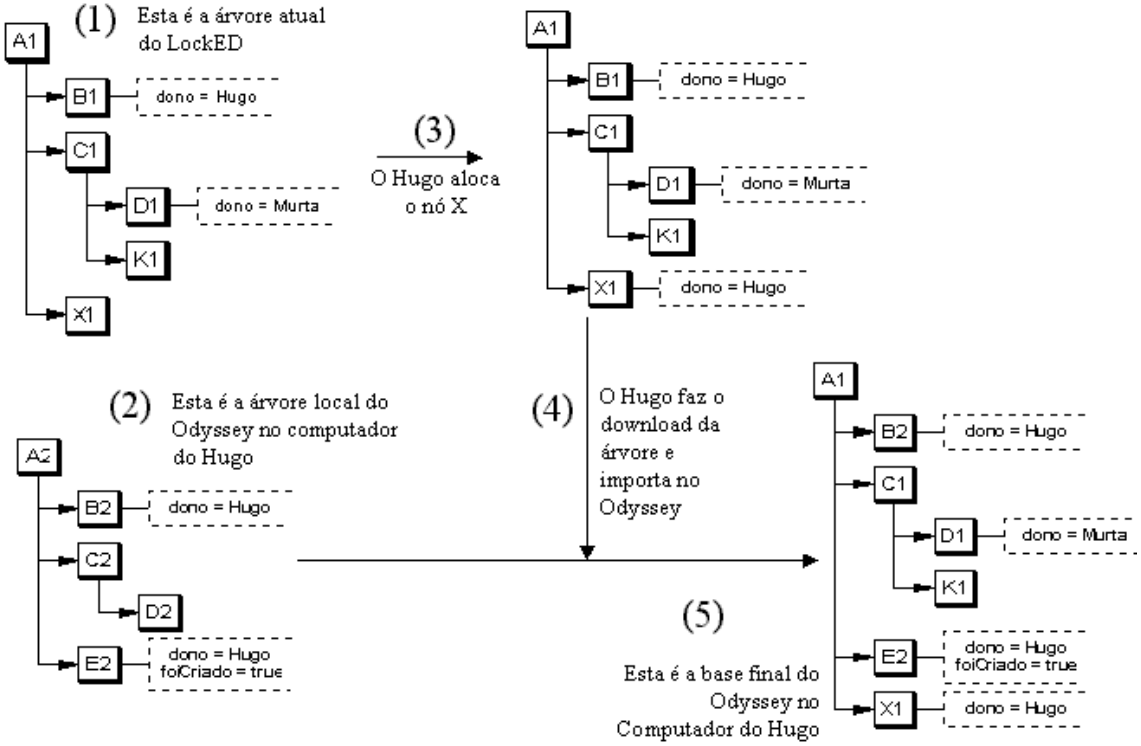

Figura 10: Exemplo de alocação de artefatos no LockED

Na Figura 11 apresentamos o algoritmo criado para tratar a importação do caso em que o usuário faz o download do domínio no servidor e o importa em sua estação de trabalho.

```
(1) Faça o import da árvore pelo arquivo serializado
(2) Para cada nó X da base do Odyssey cujo dono é o usuário,  faça:
    (2.1)  Se o nó X foi criado (X.foiCriado == true)
        (2.1.1)   Busque o pai do nó criado na base do LockED
        (2.1.2)   Insira o nó criado no pai encontrado.
    (2.2) Senão, faça:
        (2.2.1) Busque o nó X na base do LockED
        (2.2.2) Copie X por cima do nó encontrado, preservando o seu pai e
        seus descendentes
(3) Substitua a base do Odyssey pela base serializada.
```

Figura 11: Algoritmo de importação do LockED para o Odyssey

Visando compreender melhor o algoritmo que trata da junção dos domínio dentro do servidor, apresentamos na Figura 12 uma situação onde dois usuários interagem com os artefatos de um domínio, efetuando modificações nas bases e atualizando-as no LockED.

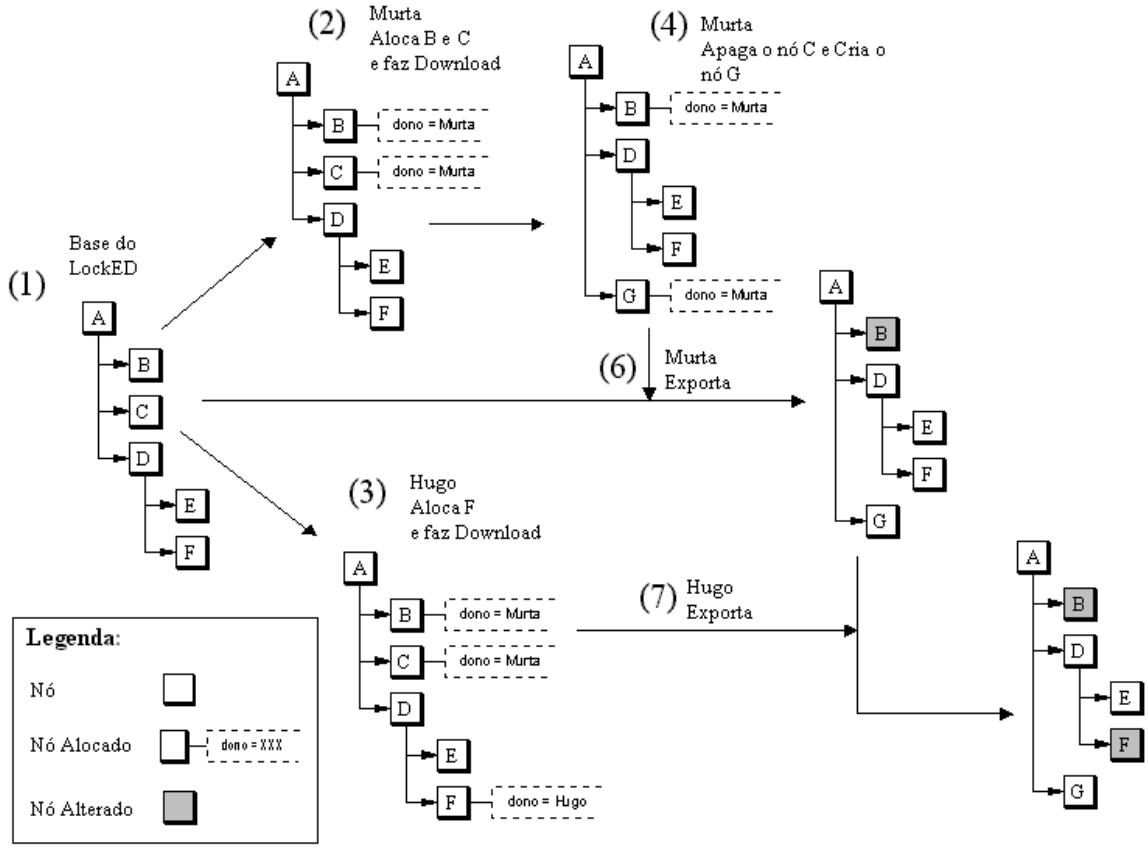

Figura 12: Exemplo de interação entre usuários e o servidor

4. Conclusões e Perspectivas Futuras

Apresentamos neste artigo a ferramenta LockED, que foi implementada pelo grupo de reutilização de software da COPPE/UFRJ, com o objetivo de controlar as modificações realizadas em artefatos de software em um processo onde a equipe encontra-se geograficamente dispersa. A ferramenta foi desenvolvida utilizando-se a linguagem Java, combinada com a tecnologia JSP, sendo disponibilizada em um servidor *web*

conectado à Internet/Intranet. Esta abordagem contribuiu com inúmeras vantagens tanto para a equipe como para a consistência das informações manipuladas, apresentando como benefícios diretos:

1. A consistência e a integridade da informação;
2. A unificação de todos os artefatos na base de dados oficial no servidor, o qual representa o ponto de contato entre todos os participantes da equipe;
3. A visibilidade sobre a alocação dos artefatos;
4. A facilidade de acesso à ferramenta pela Internet/Intranet;
5. A flexibilidade sobre a ordem de alocação e desalocação dos artefatos.

Este sistema foi integrado à infra-estrutura de reutilização de software Odyssey, a qual apresenta suporte para a modelagem de domínios e aplicações voltados para o desenvolvimento de software. A utilização do sistema LockED combinado com o Odyssey pode melhorar muito a maneira com a qual estamos acostumados a desenvolver software. Isto porque não será mais necessário administrar o código-fonte dos sistemas em desenvolvimento com se fosse um produto do processo, visto que podemos gerá-lo a partir dos artefatos que descrevem a aplicação, os quais são modelados e armazenados em bases de dados do Odyssey.

Neste momento, o sistema LockED está em fase de experimentação pela equipe de desenvolvimento do Projeto Odyssey (cerca de 10 pessoas envolvidas, entre docentes, alunos de doutorado, mestrado e iniciação científica), não existindo ainda resultados estatísticos positivos ou negativos sobre sua utilização.

Como trabalho futuro, já estudamos a possibilidade de adaptação desta abordagem para a utilização num contexto de trabalho cooperativo, criando um ambiente multi-usuário voltado para a modelagem dos domínios e aplicações de software.

Referências Bibliográficas

[1] Braga, R.; Werner, C. “Odyssey-DE: um Processo para Desenvolvimento de Componentes Reutilizáveis”, X CITS, Curitiba, Maio, 1999.

[2] Burton System Software; Site na Internet em http://www.burtonsys.com/

[3] Crnkovic,I; “Distributed Development Project using WWW”, Proceedings on 19th International Multimedia Conference Opatija, Croácia, Maio, 1997.

[4]Giant Technologies Limited; Site na Internet em http://www.giant-technologies.com/sourceoffsite/

[5] Mair, Q.; “Technical Issues in the Design of a Virtual Software Corporation”; ECSCW'97 OOGP workshop; 1997.

[6] Mattoso, M.; Werner, C.; Braga, R.; Pinheiro, R.; Murta, L.; Almeida, V.; Costa, M.; Bezerra, E.; Soares, J.; Ruberg, N.; "Persistência de Componentes num Ambiente de Reuso", XIV Simpósio Brasileiro de Engenharia de Software, Sessão de Ferramentas, João Pessoa, outubro 2000.

[7] Merant; Site na Internet em http://www.merant.com/products/pvcs/vm/index.asp

[8] Microsoft Corporation; Site na Internet em http://msdn.microsoft.com/ssafe/

[9] Miler Jr., N.; Werner, C.; Braga, R. “O uso de Modelos de Features na Engenharia de Aplicações”, IDEAS´00, Cancun, México, Abril, 2000, pp.85-96.

[10] Murta, L.G.P.; Barros, M.; Werner, C.; "Token: Uma Ferramenta para o Controle de Alterações em Projetos de Software em Desenvolvimento", XIV Simpósio Brasileiro de Engenharia de Software, Sessão de Ferramentas, João Pessoa, outubro 2000.

[11] Pressman, R. S.; "Engenharia de Software"; MAKRON Books; 1995.

[12] Quality Software Components; Site na Internet em http://www.qsc.co.uk/gpv/gpversion.htm

[13] Reliable Software; Site na internet em http://www.relisoft.com/co_op/

[14] Schooler E. M., “Conferencing and collaborative computing”, Multimedia Systems, Vol. 4, 1996, 210-225

[15] Tanenbaum, A.S.; “Computer Networks”; 3rd edition; Prentice Hall; Março, 1996.

[16] Werner, C.; Braga, R.; Mattoso, M.; Murta, L.; Costa, M.; Pinheiro, R.; Oliveira, A.; "Infra-estrutura Odyssey: estágio atual", XIV Simpósio Brasileiro de Engenharia de Software, Sessão de Ferramentas, João Pessoa, outubro 2000.

[17] Conradi, R; Westfechtel, B; “Version models for software configuration management”; ACM Computing Surveys; junho 1998, pp. 232-282.