A Base DELTA e as outras Bases de Software de um Ambiente de Desenvolvimento de Software Cooperativo, Distribuído e Real

Carlos A. M. Pietrobon - Arndt v. Staa
Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro
Rua Marquês de São Vicente 225
22453 - Rio de Janeiro - RJ - Brasil
e-mail: pietrobo@inf.puc-rio.br
arndt@inf.puc-rio.br

Resumo

Este artigo apresenta uma proposta de estrutura de bases de dados para um ambiente real (industrial) de desenvolvimento de software, que seja distribuído e cooperativo. Esta proposta identifica, além das bases utilizadas para construir o software, todas as outras bases de software que podem conter informações que podem ser usadas no sistema em desenvolvimento. Serão também introduzidos os conceitos da Base Delta e dos objetos indiretos. A Base Delta contém informações sobre todas as alterações feitas sobre a base de dados local a cada desenvolvedor. Esperamos que esta base auxilie a integração das partes componentes do software quando for realizada a integração dos trabalhos desenvolvidos pelos diversos desenvolvedores. O objeto indireto contém uma referência para onde o verdadeiro objeto se encontra.

Abstract

This paper presents a database structure appropriate to support a real (distributed and cooperative) software development environment. This proposal identifies the set of databases which are required in addition to those which contain the software being developed. These additional databases contain reusable information. Finally, the Delta Base and the Indirect Object concepts will be introduced. The Delta base stores all modifications made by a developer on his private copy stored in his private database. Using the Delta base, we hope to support more efficiently the integration of the parts developed by each person or group. The Indirect Object contain a reference for the real object location.

1 INTRODUCÃO

A necessidade de suportar adequadamente o desenvolvimento de um sistema de software grande exige, entre outros requisitos, um suporte automatizado para uma equipe de desenvolvedores, trabalhando em várias estações de trabalho, distribuídas geograficamente e conectadas de forma "on-line", ou não.

O ambiente de trabalho dessa equipe exige um complexo sistema de bases de dados inter-relacionadas (pelo menos logicamente), que suporte e organize as atividades de manipulação dos dados [CHOU86]. Particularmente, estamos nos referindo às atividades que envolvem a construção e o gerenciamento de um produto - o software objeto - que é composto de muitos componentes, os quais evoluem ao longo do tempo e estão sujeitos a acessos concorrentes por parte dos diversos desenvolvedores.

Muitas das propostas encontradas na literatura propõem modelos para as bases de dados que são muito restritos, por:

- não considerarem a existência de outras bases de software, além daquelas que armazenam o sistema em desenvolvimento (por exemplo, pode existir uma base com antigos sistemas desenvolvidos que pode ter componentes a serem reaproveitados);
- proporem operações de manipulação dos dados (entre elas, as operações de transferência de dados entre as bases) muito rígidas, que não espelham a realidade de um ambiente de trabalho;
- proporem mecanismos de controle de acesso e consistência que não se adequam bem às atividades de desenvolvimento (trabalho cooperativo) e que restringem a produtividade do desenvolvedor (por exemplo, bloqueio de acesso);
- considerarem os componentes de software armazenados como entidades puramente passivas, que não possuem nenhuma informação que auxilie na sua manipulação;
- considerarem os componentes de software armazenados como estruturas de informação monolíticas, que não podem ser decompostas em estruturas mais elementares.

Neste trabalho, é proposto um modelo de estrutura de bases de dados para um ambiente real (industrial) de desenvolvimento de software. Esta proposta aponta, além das bases utilizadas para construir o software, todas as outras bases de software que podem conter informações que podem ser usadas no sistema em desenvolvimento. Ao descrever estas bases, são indicadas as operações que podem ser realizadas sobre os dados nelas contidos. Essas operações devem espelhar o mais fielmente possível as diversas atividades desenvolvidas em um ambiente real de trabalho. Será também apresentado o conceito da base Delta. Este conceito é introduzido para ajudar a manter a integridade do software quando ocorrem transferências ("check-in") de dados entre as bases privativas a um desenvolvedor e as outras bases do sistema objeto. Esta transferência ocorre quando um desenvolvedor deseja integrar o resultado de seu trabalho com o resto do sistema desenvolvido pela equipe.

Como o ambiente de trabalho é distribuído e cooperativo, estamos considerando a existência de um conjunto de estações de trabalho interligadas suportando as atividades. No entanto,

como eventualmente, os postos de trabalho podem ficar "off-line", introduziremos o conceito de objetos indiretos que nos permitirão manter, mesmo que precariamente, algumas das atividades em curso. Estes objetos facilitarão, também, as transferências de objetos entre as bases.

Com o uso dos conceitos da Base Delta e objetos indiretos, pretendemos dar um passo para atingir o objetivo principal da coordenação de uma equipe de desenvolvimento que, segundo [HARR90], vem a ser aumentar, e não restringir, a produtividade dos desenvolvedores.

Devido as características do ambiente de trabalho cooperativo (arquitetura distribuída, acesso concorrente, etc) e de um software grande (muitos objetos, cada um com várias versões, etc), são necessários modelos de gerência de configuração e de controle de versão que possibilitem manusear os milhares de objetos e coordenar os diversos desenvolvedores [BELK87]. Embora alguns aspectos destes dois mecanismos sejam abordados no trabalho, o modelo completo e detalhado foge ao escopo deste trabalho.

Este trabalho está inserido dentro do projeto TOTEM [STAA91] em desenvolvimento na PUC/RJ. Este projeto tem por objetivo desenvolver um meta-ambiente de desenvolvimento de software, bem como estudar ambientes específicos com ele gerados.

2 AS BASES DE SOFTWARE

As bases de dados que contêm informações (representações) sobre um sistema de software são denominadas Bases de Software. A seguir mostramos seu conteúdo, seus diferentes tipos e as operações de tranferência de objetos que podem ser realizadas entre as bases.

2.1 Conteúdo das Bases de Dados

As bases de software armazenam objetos de projeto. Estes objetos representam as diversas partes componentes do sistema objeto sendo manipulado, expressos segundo alguma linguagem de representação, em algum nível de abstração e que possa ser univocamente identificado e manipulado. Um processo em um Diagrama de Fluxo de Dados, um módulo fonte em Pascal ou todo o sistema objeto são exemplos de entidades que são representadas por objetos de projeto.

Como aponta [PARK91], objetos de projeto têm que ser uma coleção de informações. Assim, do ponto de vista implementacional, cada objeto é uma estrutura de dados complexa, composta de diversos atributos e que possui diversas relações com outros objetos.

Os atributos são, por sua vez, estruturas de dados que estão divididas em categorias, de acordo com a classe de informação que armazena. As categorias são as seguintes [PIET91]:

Nome: é a identificação do objeto.

Aliases: são nomes ou identificações alternativas.

Fragmentos de Texto: seqüencia de zero ou mais linhas de texto, contendo código fonte, documentação, etc.

Valores Formatados: são dados estruturados cuja interpretação depende da ferramenta que manipula o objeto (por exemplo, coordenadas onde deve ser exibido o ícone associado ao objeto, no dispositivo de saída).

Relacionamento: estabelecem vínculos entre dois objetos, usualmente, mas não necessariamente, distintos.

Graças aos relacionamentos entre objetos estarem explicitamente armazenados, pode-se visualizar uma base de software como uma rede de objetos de projeto fortemente conectados entre si. Isto traz consigo diversas vantagens. Como exemplo, a construção de uma representação do sistema de software pode ser obtida navegando-se pelos diversos objetos que compõem o software, em busca das informações necessárias, como é feito no editor de estruturas descrito em [STAA90].

Cada objeto de projeto armazenado nas Bases de Software representa um elemento de uma linguagem de representação usada para se construir o software ou um conjunto de elementos. Por exemplo, cada processo de um Diagrama de Fluxo de Dados é representado por um objeto de projeto na base de dados. Já uma folha de fluxo de dados tem associado a si um objeto que faz referências a todos os objetos componentes. Portanto, via de regra, esses objetos são pequenos quando comparados com o software como um todo. O objeto que representa um sistema de milhares de linhas de código pode ser constituído de poucas dezenas de atributos, em particular aqueles que referenciam os objetos componentes. Em termos de memória física, ele pode usar algo em torno de 500 bytes, se tanto. No outro extremo, temos objetos elementares, que não são compostos de outros mais simples, cuja quantidade de informação armazenada também não é muito grande. Por exemplo, o objeto que representa o bloco "configurar prompt" em diagrama de estrutura modular, pode ter apenas uma linha de código armazenada. Além disso, teremos alguns atributos de relacionamento e, possivelmente, mais apenas o atributo nome.

Como o objeto de projeto representa uma parte muito pequena da representação total, e estes objetos estão fortemente relacionados, é muito fácil identificar partes alteradas em operações de atualização. Sob essa ótica, uma alteração implica em atualizar, incluir ou excluir vários pequenos objetos, que são individualmente identificáveis. Estes conceitos serão muito úteis para se entender o que é a base Delta.

Nem toda base possui uma cópia integral do sistema objeto em desenvolvimento. Ela pode possuir apenas alguns objetos. Além disso, a qualidade dos objetos varia conforme a base. Na seção 2.3, abordamos mais detalhadamente estes aspectos.

2.2 As Diferentes Bases

Um ambiente de desenvolvimento de software deve possuir as seguintes bases de dados de projeto de software (daqui para frente chamadas Bases de Software):

- Base de Software Privativa (BSWPriv)
- Base de Software de Projeto Específico (BSWProj)
- Base de Software Pública (BSWPub)

- Base de Software da Organização (BSWOrg)
- Base de Software de Outro Projeto (BSWOutro)
- Base de Software Alienígena (BSWAlien)

Cada uma destas bases podem ser vistas como bases lógicas, compostas de diversos arquivos físicos, fortemente interrelacionados entre si.

2.2.1 A BSWPriv

A BSWPriv é uma base de trabalho onde o desenvolvedor, de fato, constrói o software. De fato, as diversas informações nela armazenadas estão em estado transiente, por estarem sendo alteradas, ou por não apresentarem qualidade suficiente para serem liberadas para outros desenvolvedores, ou usuários.

Existem diversas bases privativas, uma para cada pessoa ou equipe participante do esforço de desenvolvimento do sistema objeto. Como as informações armazenadas nestas bases não apresentam qualidade satisfatória, elas não podem ser formalmente compartilhadas com outros usuários, ou seja, um desenvolvedor não pode, indiscriminadamente, acessar quaisquer informações armazenadas em outras bases privativas. No entanto, este acesso pode ser feito de modo informal, como será exposto mais à frente.

O responsável pela BSWPriv tem amplos poderes sobre todos os dados nela armazenados, podendo acessá-los, alterá-los e eliminá-los livremente. A exceção para esta última afirmativa consiste naqueles dados que foram importados de outras bases sobre os quais ele não possui autorização de acesso.

A razão de dados diretamente inacessíveis ao desenvolvedor estarem presentes na sua base de trabalho, deve-se a razões de segurança (replicação de dados), eficiência de acesso e ao conceito de contexto de trabalho estável, que deve estar associado a cada desenvolvedor, quando trabalhando em equipe.

O conceito de contexto de trabalho estável reza que o desenvolvedor deve ter uma visão dos dados que seja estável durante um certo período de tempo. Caso contrário, ele teria que se adaptar instantaneamente às modificações feitas nos outros objetos pelos outros membros da equipe. Ter uma uma cópia dos objetos usados, localmente disponível, seria um meio de se conseguir o contexto estável sem prejudicar o paralelismo das atividades.

Um novo objeto (versão) pode ser criado do nada ou importado de qualquer outra base de software. Além disso, uma nova versão de um objeto pode ser derivada a partir da versão de um objeto existente localmente. Neste caso, deve-se atualizar o histórico de versões do objeto considerado.

Finalmente, os objetos podem ser copiados (exportados) para qualquer base de software. Caso a cópia tenha sido requisitada por outro desenvolvedor, o proprietário da BSWPriv deve autorizar a transferência.

2.2.2 A BSWProj

Na BSWProj temos as representações que atingiram um grau de qualidade que permite que elas sejam compartilhadas por um grupo restrito de desenvolvedores, responsáveis pelo desenvolvimento de um projeto específico (sub-projeto) dentro do sistema objeto. Assim, esta base funciona como um meio de comunicação formal entre os desenvolvedores de um projeto específico.

As diversas pessoas que trabalham no sub-projeto, quando desenvolvem uma parte do sub-projeto e esta parte apresenta um nível mínimo de qualidade (o objeto foi testado, mas não exaustivamente, ou seja, não foi feito o teste com todos os outros objetos do sistema considerado), colocam esta parte disponível a esta comunidade restrita, exportando-a da BSWPriv onde ela foi desenvolvida para a BSWProj. Isto vai de encontro com a realidade de desenvolvimento, pois um desenvolvedor pode depender do trabalho de outros e, sendo assim, nada mais natural que ele use os objetos desenvolvidos pelos outros, mesmo que estes objetos ainda venham a ser modificados. Por exemplo, uma pessoa pode estar desenvolvendo um módulo que chama outro desenvolvido por outra pessoa. Até que o módulo chamado esteja disponível na BSWProj, ele terá que usar um módulo "dummy" para simular a chamada. Quando o módulo chamado estiver disponível, ele poderá executar seu módulo de forma mais realista. No entanto, a pessoa deverá estar atenta, pois podem ocorrer erros se o módulo chamado não estiver suficientemente testado e validado.

A BSWProj pode ser uma hierarquia de bases de projeto ou, simplesmente, não existir. Isto depende da estrutura organizacional do ambiente de trabalho. Caso o sistema em desenvolvimento não seja muito grande e a integração entre a equipe seja muito forte, pode-se dispensar esse nível de bases de software. Por outro lado, se o sistema for muito grande e os desenvolvedores não se conhecem, certamente precisaremos de uma hierarquia mais profunda.

A BSWProj tem um responsável que é o Administrador da BSWProj (ADMProj). Ele é, tipicamente, o chefe do sub-projeto. Ele deve cuidar para que os dados introduzidos não corrompam a integridade da base, controlar quem pode acessar quais módulos e definir os modos de acesso (leitura, escrita, acesso para compilação, etc) que cada um pode ter sobre os diferentes objetos copiados. Além disso, somente o ADMProj pode retirar objetos dessa base.

Os objetos de projeto dessa base são considerados estáveis e, conseqüentemente não podem ser alterados. Caso seja necessária alguma modificação, o objeto deve ser exportado para a BSWPriv, alterado e retornado como uma nova versão.

Um objeto instalado nessa base é criado a partir da cópia de um objeto situado em qualquer outra base de software. Essa cópia pode ser requisitada pelo ADMProj ("check-out") ou pelo responsável por outra base ("check-in"). Nesse último caso, o ADMProj pode impedir a operação, caso julgue que a introdução do objeto comprometerá a integridade da base.

2.2.3 A BSWPub

A Base de dados Pública é a base onde estão armazenadas todas as informações (representações) do sistema objeto em desenvolvimento, que atingiram um grau de estabilidade (qualidade) que permita que elas possam ser compartilhadas por todos os outros desenvolvedores, ou mesmo

serem liberadas para o usuário final.

Na BSWPub estão armazenadas todas as representações de todos os objetos de software componentes do sistema objeto. Aí estão incluídas as representações relativas às diversas versões de cada objeto componente. Consequentemente, na BSWPub estão todos os dados necessários para se construir qualquer representação relativa a qualquer versão do sistema objeto ou de um sub-sistema componente. Naturalmente, estas afirmativas só serão verdade após ter sido desenvolvida pelo menos uma versão de cada objeto componente.

O objeto que está nessa base é considerado estável e, consequentemente, não pode ser alterado. Caso seja necessário uma modificação, ele deve ser exportado para a BSWPriv, alterado e retornado como uma nova versão. Possivelmente ele terá que ser copiado, também, para uma BSWProj.

Como os objetos são estáveis, eles não podem ser eliminados. No entanto, esta regra é relaxada para permitir que versões muito antigas de um objeto, que não estejam sendo utilizadas, sejam eliminadas. Esta eliminação só poderá ser feita pelo ADMPub.

Qualquer inserção de dados na BSWPub só é permitida após autorização fornecida pelo Administrador da BSWPub (ADMPub), que é o responsável pela integridade das informações nela retida e o destino das informações dela retiçadas.

Os objetos da BSWPub podem ser exportados para qualquer outra base de software. Como no caso da BSWProj, a exportação deve ser autorizada pelo responsável pela base destino. Por outro lado, a base pode importar objetos da BSWPriv ou da BSWProj. No entanto, estes objetos devem ter qualidade assegurada.

2.2.4 A BSWOrg

A Base de Dados da Organização (BSWOrg) armazena software desenvolvido pela firma ao longo de sua existência, no âmbito de um Ambiente Automatizado de Desenvolvimento de Software (AADS). Esta base tem duas funções:

a) Armazenar software já desenvolvido;

b) Permitir e facilitar a reutilização de componentes.

Desta forma, ao se desenvolver uma nova aplicação, muito esforço pode ser economizado pela reutilização de objetos desenvolvidos anteriormente na organização. É evidente que isso já é feito atualmente, só que de uma maneira informal e manual.

De forma semelhante às bases vistas anteriormente, esta base deve ter seu administrador (ADMOrg), que zelará pela integridade da mesma e o destino das informações dela retiradas.

Os objetos que estão nessa base, são considerados estáveis, não podendo ser alterados. No entanto, eles podem ser copiados para uma BSWPriv podendo, nesta última base, serem alterados. Estas cópias modificadas podem ser introduzidas na BSWOrg porém, como uma nova versão do objeto. No entanto, objetos muito antigos podem ser removidas da BSWOrg. Esta atividade só pode ser realizada pelo ADMOrg.

A introdução de objetos na BSWOrg só poderá ser feito a partir da BSWPub de algum projeto. Isto se deve ao fato de que somente os objetos armazenados na BSWPub apresentam

um nível de qualidade satisfatória para serem compartilhados por toda a organização.

Qualquer desenvolvedor reponsável por qualquer base de software pode extrair objetos da BSWOrg, desde que possua autorização para isso. Caso contrário, o objeto pode ser obtido via o ADMOrg, que avaliará a requisição.

2.2.5 A BSWOutro

A Base de Software de Outro Projeto (BSWOutro) é, na verdade, um conjunto de bases de software. Estas bases são aquelas que estão sendo usadas para o desenvolvimento de um outro projeto (BSWPub, BSWProjs e BSWPrivs).

O objetivo principal de se considerar a existência de uma ou mais BSWOutro é o reuso. Em outros projetos sendo desenvolvidos paralelamente ao projeto considerado, podem estar sendo construídos objetos que podem ser reusados. Por outro lado, aplicações diferentes podem, quando prontas, trabalhar comunicando-se via uma interface comum. Por exemplo, aplicações que rodam diretamente em cima do sistema operacional. Se o sistema operacional e as aplicações estiverem sendo desenvolvidas ao mesmo tempo, o desenvolvedor de um dos projetos pode importar rotinas do outro para testar o seu trabalho.

A requisição e a importação de dados de uma BSWOutro não são atividades automáticas. Por questões de segurança e problemas de relacionamento de pessoal, a exportação de dados da BSWOutro para outro projeto só ocorrerá se o dono ou responsável pela base de software que contém o dado autorizar a exportação. No entanto, a localização do objeto desejado poderia ocorrer com apoio automatizado. Colocando de outra forma, um desenvolvedor deveria ter condições de saber quais os objetos que estão sendo desenvolvidos por outras pessoas, em outros projetos.

2.2.6 A BSWAlien

Muitas das aplicações existentes hoje em dia não foram desenvolvidas dentro de um AADS. Isto é um problema porque a utilização de um AADS impõe que todas as representações do software objeto estejam armazenadas segundo um formato que é característico para cada AADS. Isso faz com que as representações de aplicações desenvolvidas fora do AADS considerado sejam vistas como dados alienígenos. Assim, a BSWAlien é a estrutura de armazenamento que contém tais tipos de dados. Para que esses dados possam ser efetivamente utilizados por um AADS, eles precisam ser convertidos para a representação que seja reconhecida pelo AADS considerado. Isto terá que ser feito utilizando-se uma ferramenta de conversão ou, se não for possível, manualmente.

Apesar das dificuldades para se usar os dados depositados na BSWAlien, esta base deverá, pelo menos, fornecer um catálogo com informações sobre os dados armazenados de forma a facilitar a localização de informações que podem estar sendo necessárias à alguém.

A BSWAlien diferencia-se da BSWOrg pelo fato da primeira conter dados estruturados de forma totalmente diferentes da que é usada no AADS correntemente em uso na organização. De resto, ela deve ser conceitualmente idêntica à BSWOrg, possuindo as mesmas funções. Com

o passar do tempo, os dados armazenados na BSWAlien devem ser migrados para a BSWOrg.

Este catálogo poderia ser criado e mantido por um Administrador da BSWAlien (ADMAlien). Além disso, é claro, ele zelaria pelo conteúdo da base e do destino das informaçẽs dela retiradas.

2.3 Transferências de objetos entre as bases

A transferência de objetos entre as bases se faz por meio da cópia do referido objeto da base origem para a base destino. Esta transferência pode ocorrer de duas formas: através da operação *check-in* ou via a operação *check-out*.

A operação *check-in* pode ser vista como uma operação de exportação, onde o proprietário de uma base origem envia dados para uma base destino. Cabem aqui duas observações. Primeiro, o responsável pela base destino pode recusar receber o dado pois ele pode não apresentar qualidade satisfatória. Segundo, a introdução deste dado pode provocar, ou não, a criação de uma nova versão do referido objeto na base destino.

A geração de uma nova versão durante o *check-in* depende de porque foi feito o *check-out* do referido objeto. Por exemplo, se duas ou mais pessoas tiraram cópias de um mesmo objeto para desenvolverem funcionalidades diferentes, mas complementares, quando elas forem introduzidas na base de onde foram copiadas, todas as cópias deverão ser integradas para gerar uma única versão possuindo todas as funcionalidades desenvolvidas separadamente. Por outro lado, existem casos onde as diversas cópias não devem ser integradas em uma única versão. Neste caso, cada cópia gerará uma versão diferente. É o que ocorre quando desenvolvemos um mesmo produto para diferentes clientes

A operação *check-out* pode ser vista como uma operação de importação, onde o responsável por uma base destino requisita objetos de uma base origem. Deve-se apontar que a requisição pode ser indeferida caso o requisitante não possua autorização para acessar os dados ou o responsável pela base origem não permita a transferência. Além disso, no nosso modelo, sempre que um objeto for importado, será gerado uma nova versão para o mesmo

Na figura 1 são mostradas as diferentes possibilidades de transferência de objetos entre as bases. Note que, dependendo do referencial considerado (qual a base de referência), as diversas transferências apresentadas podem ser consideradas como conseqüência de um *check-in* ou *check-out*.

As tranferências entre a BSWOrg e as outras bases usadas na construção do sistema objeto apresenta uma particularidade: embora seus objetos possam ser exportados para qualquer uma das três bases, ela só pode receber objetos da BSWPub. Isto deve-se ao fato de que somente os objetos públicos atingiram um nível de qualidade satisfatório.

As transferências entre a BSWOutro e as bases do sistema objeto também apresentam uma característica que deve ser apontada: A BSWPub não pode receber objetos da BSWOutro. A razão é que nada garante que o objeto importado apresente, dentro do ambiente de desenvolvimento do sistema objeto, qualidade suficiente pra ser tornado público para toda a comunidade.

Observando-se as BSWPrivs percebe-se que existe tranferência de dados entre elas. Isto modela uma realidade dos ambientes de desenvolvimento: pessoas trocam informações entre

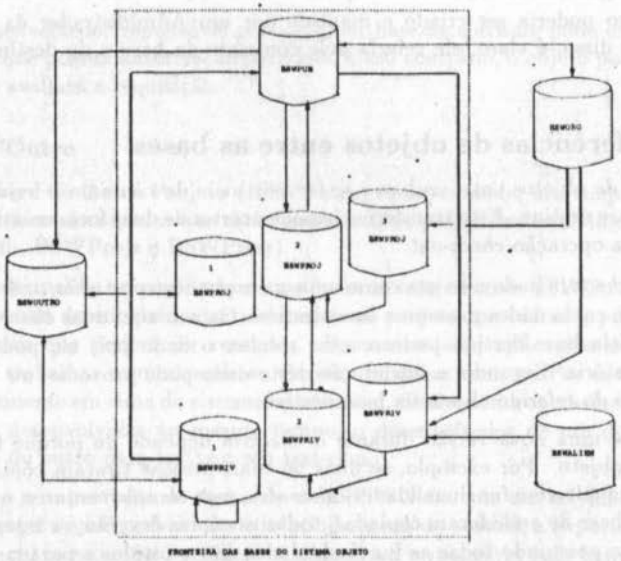

Figure 1: As Bases de Software

si, mesmo que informalmente, sem nenhum apoio direto das ferramentas de gerenciamento das bases de dados. No nosso caso, para irmos de encontro a esta realidade, permitiremos que tais transferências sejam feitas com o apoio dos gerenciadores das bases. No entanto, a responsabilidade pelas consequências de tais operações é deixada a cargo do desenvolvedor.

Finalmente, deve-se esclarecer o que é transferido entre as bases que contém o sistema em desenvolvimento. Uma proposta seria transferir uma versão completa do sistema. Outra, seria transferir apenas os objetos a serem alterados ou já manipulados.

No primeiro caso, embora tenhamos algumas vantagens, tais como, o desenvolvedor poder trabalhar mesmo quando a rede cair, pode ser uma solução proibitiva em termos de consumo de memória. No segundo caso, o desenvolvedor ficará demasiadamente dependente de uma rede física de comunicação, o que pode interromper suas atividades, caso haja problemas com a rede.

Como nenhuma das duas soluções é plenamente satisfatória, propomos uma mais flexível. Segundo esta proposta, as transferências devem ser feitas considerando as necessidades do desenvolvedor, os recursos disponíveis e aspectos gerenciais (por exemplo, dados muito usados em um projeto localizado em uma cidade devem ser colocados em uma base de projeto nesta cidade e não em uma outra cidade). Assim, podemos transferir todo o sistema, só os objetos a serem modificados ou um outro subconjunto qualquer dos objetos existentes.

Caso, nem todos os objetos que serão usados em uma base sejam copiados para essa base,

terão que ser criadas referências que permitam localizar e acessar os objeto não disponíveis localmente. Isto se dará por meio dos objetos indiretos, discutidos a seguir.

3 Objetos Indiretos

Sempre que copiarmos alguns objetos para uma base, possivelmente, alguns desses objetos farão referências à outros objetos localizados na base origem. Enquanto a rede funcionar não existe problemas. Entretanto, quando houver uma falha, todo o trabalho pode ser interrompido pois o sistema de apoio encontrará referências não satisfeitas. Usando objetos indiretos esperamos resolver tais problemas.

Um objeto indireto representa o objeto referenciado que não se encontra presente na base considerada. O objeto indireto possui uma referência para onde está realmente o objeto requerido. Além disso, possui outras informações, tais como o tipo do objeto, a localização de objetos equivalentes, etc. Isto nos fornece duas vantagens:

- o desenvolvedor pode saber exatamente quais são os objetos que ele não tem disponível localmente;
- os objetos existentes localmente, que usam um objeto não disponível localmente, ficam independentes da localização desse objeto. Se a referência principal, do objeto indireto, for nula, sabe-se que o objeto existe mas não está acessível.

Observe que o uso de objetos indiretos nos fornece um mecanismo poderoso de manipulação de objetos e de trabalho. Quando for perdido o contato com a BSWProj correspondente ou com a BSWPub, pode ser feito um redirecionamento para uma outra BSWProj ou BSWPriv que contenha os objetos referenciados e que ainda estão acessíveis. Uma outra situação, onde o uso deste objeto seria útil, é quando ocorre movimentação de dados entre as bases. Se um objeto existente na BSWPRoj é promovido para a BSWPub e retirado da BSWProj, basta redirecionar a referência contida no objeto indireto. Quem usa este objeto fica alheio a essa alteração.

Por fim, com o uso destes objetos, podemos continuar trabalhando, de forma limitada, quando existe a perda de comunicação. Por exemplo, se um diagrama de estrutura modular for composto de alguns objetos não disponíveis, o sistema avisa ao usuário que ele não pode acessá-los e permite que ele trabalhe em cima dos outros blocos disponíveis.

4 A BSW Delta

Ambientes de trabalho cooperativo apresentam, entre outros, o problema do compartilhamento dos dados de projeto. Para enfrentar tais dificuldades, várias soluções têm sido propostas. Entre elas, podemos citar duas bastante usadas:

- acesso serial ao objeto. Neste caso, um desenvolvedor só pode acessar um objeto do sistema em desenvolvimento, se ele não estiver sendo acessado por mais nenhuma outra pessoa. Esta técnica peca por limitar a produtividade uma vez que uma pessoa tem que

esperar pelo término do trabalho da outra para acessar o objeto. Em outras palavras, não existe paralelismo de atividades. Um outro problema, mais grave, é que esta técnica não garante a consistência do sistema. Por exemplo, suponha que uma pessoa faça uma alteração no sistema. Posteriormente, outra pessoa vem e altera os componentes manipulados pela primeira pessoa. Se este último voltar a acessar o sistema, ele encontrará uma situação diferente da que ele conhecia;

- uso de múltiplas cópias do objeto. Toda vez que um novo acesso é feito ao objeto, uma cópia do mesmo é feita para a base de dados do requisitante.

Nós optamos por esta segunda maneira de tratar o problema, pelos seguintes motivos: aumenta o paralelismo das atividades (aumentando a produtividade), aproxima-se bastante da realidade de um ambiente de trabalho, fornece um contexto de trabalho estável para o desenvolvedor, permite que o desenvolvedor trabalhe mesmo estando desconectado do resto do sistema e, por replicar dados, permite que dados sejam recuperados, em caso de pane, em algum nodo da rede de bases.

Segundo nosso modelo, sempre que um indivíduo resolve alterar uma parte do sistema, ele recebe uma cópia local do objeto (ou objetos) que ele deseja alterar. Quando ele terminar suas modificações, o(s) objeto(s), deve(m) ser reintroduzido(s) na base de onde foi(foram) importado(s).

O retorno do objeto modificado à base de onde foi extraído é problemático porque a base origem pode ter sido alterada devido ao "check-in" realizado por outro desenvolvedor. Note que, se não houvesse outras pessoas trabalhando, não haveria conflito na volta pois a base que possuia a versão original não teria sido alterada por mais ninguém. Toda e qualquer inconsistência causada pelas alterações teriam sido resolvidas a nível das bases privativas.

Como neste tipo de ambiente de trabalho o "check-in" é inevitável e os conflitos resultantes do "check-in" também são inevitáveis, propomos um modelo que permite conviver mais naturalmente com esta realidade. Assim, introduzimos o conceito da Base de Software Delta (BSWDelta) e fazemos um uso mais sofisticado do controle de versão e configuração.

Quando um desenvolvedor deseja trabalhar sobre um ou mais objetos de software já desenvolvidos, ele receberá uma cópia local dos objetos desejados. Desta forma, ele possui localmente uma cópia de todos os objetos que ele quer diretamente alterar. No entanto, como esses objetos podem ser compostos, eles podem referenciar objetos localizados em outra base de software. A menos de restrições de segurança sobre algumas das representações, ele poderá acessar quaisquer uma delas.

Nos trabalhos encontrados na literatura corrente, qualquer acesso de atualização feito sobre estas cópias locais , efetivamente modifica a cópia. Isso acarreta o seguinte problema: quando as diferentes representações forem inseridas na BSWPub ou BSWProj, poderá ser muito complicado garantir a consistência global da base receptora. Isto ocorrerá porque será preciso identificar as alterações feitas e localizar as partes e funcionalidades impactadas. As soluções vistas na literatura consistem em usar um comparador de arquivos que detectará porções diferentes entre duas representações. Muitos autores simplesmente não abordam como solucionar o

problema.

No nosso trabalho, propomos dividir a BSWPriv em duas sub-bases de software: a Base Cópia (BSWCop) e a Base Delta (BSWDelta). Na BSWCop estão todos os objetos que foram copiados. Na BSWDelta são armazenadas todas as alterações feitas pelo desenvolvedor. Desta forma, as representações armazenadas na BSWCop são mantidas intactas sejam quais forem as alterações feitas pelo desenvolvedor.

A BSWDelta possui três tipos de informação: os atributos novos, os atributos alterados e os atributos eliminados. Estes atributos devem estar estruturados por objeto, de forma a facilitar as operações de acesso. Caso seja criado um novo objeto, então só existem atributos para este objeto na BSWDelta. Caso seja eliminado um objeto, então é armazenado, na BSWDelta, o seu atributo identificador, com uma marca indicando que o objeto foi descartado. Caso seja feita uma alteração, então é armazenado tantos atributos quanto forem os atributos alterados.

O que é armazenado na BSWDelta não é apenas um pedaço de representação (por exemplo, algumas linhas de código fonte ou pedaços de um Diagrama de Fluxo de Dados) que é diferente do objeto original. É informação fortemente tipada tendo uma semântica bem determinada dentro do software. Por exemplo, se o desenvolvedor retirou um depósito de dados de um Diagrama de Fluxo de Dados, ele não retirou apenas um ícone da representação gráfica. Ele retirou uma instância de um objeto que ocorria no objeto que representa uma folha de fluxo de dados. Desta forma, muitas operações tem que ser disparadas como resposta a esta eliminação. Por exemplo, retirar a representação gráfica que esteja sendo exibida, informar aos outros objetos que mantinham alguma relação com o referido objeto que ele não existe mais, etc. Estas operações dependem do tipo do objeto e do tipo de alteração (remoção, modificação e inclusão de atributos).

Graças a idéia de se armazenar os atributos e não apenas uma alteração da representação, como vista externamente pelo desenvolvedor, esperamos facilitar muitas atividades, notadamente aquelas envolvidas com a manutenção da consistência do software.

A visão que o desenvolvedor possui da sua base privativa é obtida pela navegação do conteúdo de ambas as bases. Este processo é transparente tanto a um observador externo quanto a uma ferramenta que venha a trabalhar sobre esta representação. O algoritmo, simplificado, deste processamento seria o seguinte:

```
Enquanto (existem objetos a serem processados) faça
   Enquanto (existem atributos para um dado objeto) faça
      Se (um mesmo atributo ocorre em ambas as bases) ou
         (atributo só ocorre na BSWDelta)
         então processe o atributo da BSWDelta
                  ...
         senão processe o atributo na BSWCop
                  ...
   FimEnquanto
   ...
```

FimEnquanto

Quando o desenvolvedor validar sua visão da base de objetos, ele pode levar seus objetos para uma base compartilhada. Para isso, ele terá que processar a BSWCop e a BSWDelta. Os objetos existentes na BSWDelta são copiados. Já, os objetos da BSWCop só serão copiados se eles não existirem na base compartilhada.

A introdução desses objetos na base compartilhada pode, no entanto, não ser tão trivial quanto sugerido, pois conflitos decorrentes da existência de outros desenvolvedores acessando a base compartilhada podem ocorrer e tem que ser resolvidos.

A estratégia de se dividir a BSWPriv em duas bases visa facilitar o processo de integração entre as bases. Isto ocorrerá porque todas as alterações a serem comparadas estão isoladas na BSWDelta. No entanto, a eficiência da solução só é possível graças ao modelo de objetos de projeto adotado. Como estes objetos são unidades de informação altamente estruturadas, interrelacionadas e de tamanho relativamente pequeno (muitos representam diretamente um elemento da linguagem de representação, tais como um fluxo de dados), acreditamos que a técnica nos permitirá integrar de forma bastante eficiente duas representações, aumentando a produtividade dos desenvolvedores.

5 CONCLUSÕES

Neste trabalho apresentamos algumas propostas para incrementar o funcionalismo de AADS que suporte desenvolvimento de software de forma cooperativa, em uma base distruibuída, podendo o desenvolvedor estar total ou parcialmente desconectado do resto das bases de software.

As soluções aqui apresentadas foram baseadas na existência de um conceito mais sofisticado de objetos de projeto. Como vimos, ele é composto de diversos atributos e possui fortes vínculos físicos com os outros objetos existentes. Além disso, essas propostas fazem uso de mecanismos de controle de versão e configuração para controlar a evolução das representações e as transferências de objetos entre as bases. Esses mecanismos fogem ao escopo deste trabalho e serão apresentados em [PIET93], a ser publicado.

Os objetos de projeto, com as características citadas, já estão implementados para um ambiente de trabalho mono-usuário. Ele agora está sendo estendido para suportar o controle de versão e configuração e, é claro, o trabalho cooperativo em um ambiente distribuído.

6 REFERÊNCIAS BIBLIOGRÁFICAS

[**BELK87**] BELKHATIR, N.; ESTUBLIER, J. - EXPERIENCES WITH A DATABASE OF PROGRAMS, ACM SIGPLAN Not, 22(1), pp. 84-91, 1987.

[**CHOU86**] CHOU, H.T.; KIM, W. - A UNIFYING FRAMEWORK FOR VERSION CONTROL IN A CAD ENVIRONMENT, In: International Conference on Very Large Data Base,12, Kyoto, Aug.25-28, 1986. Proceedings. Los Altos, Morgam Kaufmann, p. 336-44, 1986.

[**DIAS91**] DIAS, E. Z. V.; MAGALHÃES G. C. - MVC: UM MODELO PARA CONTROLE

DE VERSÕES E CONFIGURAÇÕES, Anais do V Simpósio Brasileiro de Engenharia de Software, Ouro Preto, MG, 23 a 25 de Outubro, p. 93-106, 1991.

HARR90] HARRISON, W. H.; OSSHER, H.; SWEENEY, P. F. - COORDINATING CONCURRENT DEVELOPMENT, Proceedings of CSCW'90, p. 157-168, out., 1990

LIMA92] LIMA, M. J. D. - O MODELO DE CONTEXTOS ANINHADOS PARA DOCUMENTOS MULTIMÍDIA: Definição e Implementação, Dissertação de Mestrado, Departamento de Informática, PUC/RJ, 1992.

PARK91] PARK, H. J.; LEE, D.; LEE,K.; CHON, K. - CONFIGURATION MANAGEMENT OF OBJECT GROUPS, The Australian Computer Journal, p. 148-158, vol. 23, no. 4, nov., 1991.

PERR88] PERRY, D.; KAISER, G.E. - MODELS OF SOFTWARE DEVELOPMENT ENVIRONMENTS, IEEE, p. 60-8, 1988.

PIET91] PIETROBON, C.A.M.; STAA, A.v. - OBJETOS DE PROJETO E ENGENHARIA DE SOFTWARE: UM ENFOQUE PRÁTICO, Anais do V Simpósio Brasileiro de Engenharia de Software, Ouro Preto, MG, 23 a 25 de Outubro, p. 61-74, 1991.

PIET93] PIETROBON, C.A.M. - TRABALHO COOPERATIVO SUPORTADO POR AMBIENTES AUTOMATIZADOS DE DESENVOLVIMENTO DE SOFTWARE - Tese de Doutorado - Departamento de Informática - PUC-RJ - em desenvolvimento, defesa prevista para 1993.

PRICE89] PRICE, R. T.; GOLENDZIMMER, L. G. - BANCO DE DADOS PARA APLICAÇÕES NÃO CONVENCIONAIS, IV Escola Brasileira Argentina de Informática, Argentina, 170pp.,Jan. 1989.

STAA90] STAA, ARNDT, v. - UM META-EDITOR DE ESTRUTURAS, Anais do V Simpósio Brasileiro de Engenharia de Software, Águas de São Pedro, SP, Outubro, p. 193-202, 1990 .

STAA91] STAA, ARNDT, v. - META-AMBIENTE DE DESENVOLVIMENTO DE SOFTWARE, Comunicação pessoal, 1991.