Um Relato de Experiência com uma Infra-Estrutura para Desenvolvimento Distribuído Baseado em Componentes

João Paulo F. de Oliveira, Glêdson Elias

Departamento de Informática
Universidade Federal da Paraíba (UFPB) – João Pessoa, PB – Brasil

joaopaulo@compose.ufpb.br, gledson@di.ufpb.br

Abstract. *Distributed Software Development (DSD) and Component Based Development (CBD) have been recognized as approaches that have the potential to reduce development time, cost and complexity, as well as to improve the software quality. In such a context, the contribution of this paper is to present an experience report on the adoption of a shared, distributed component repository service, which acts as a support infrastructure for component-based distributed development projects.*

Resumo. *Abordagens de Desenvolvimento Distribuído de Software (DDS) e Desenvolvimento Baseado em Componentes (DBC) destacam-se pelo potencial de reduzir a complexidade, o tempo e o custo de desenvolvimento, bem como melhorar a qualidade do software produzido. Nesse sentido, este artigo tem por objetivo apresentar um relato de experiência na adoção de um serviço de repositório compartilhado e distribuído de componentes de software, que atua como uma infra-estrutura de apoio a projetos de desenvolvimento distribuído baseados em componentes.*

1. Introdução

Constantemente, visando resolver ou amenizar os diversos problemas encontrados no desenvolvimento de software, novas soluções e tecnologias são apresentadas aos engenheiros de software. Entretanto, segundo Travassos *et al*. (2002), novos métodos, técnicas, linguagens e ferramentas não deveriam ser apenas sugeridas, publicadas ou apresentadas para a venda sem experimentação. De forma complementar, Babar *et al*. (2004) afirma que mesmo as tecnologias desenvolvidas baseadas em experiências pessoais, observações gerais ou boas práticas não podem assegurar que os resultados esperados sejam obtidos. Portanto, além de desenvolver uma solução tecnológica, é também importante avaliar experimentalmente se a mesma atende aos objetivos e requisitos definidos inicialmente.

Neste contexto, este artigo apresenta um relato de experiência na adoção do serviço de repositório de componentes de software, denominado X-CORE (*X-ARM Component Repository*) (Oliveira, 2007), que foi concebido para atuar como uma infra-estrutura de apoio a projetos de desenvolvimento distribuído baseados em componentes. De forma mais específica, o objetivo do estudo relatado é avaliar as facilidades providas pelo X-CORE no suporte a atividades relacionadas à especificação e desenvolvimento de sistemas baseado em componentes, conduzidas por equipes distribuídas que colaboram entre si, compartilhando e integrando os diferentes artefatos de software produzidos ao longo das diferentes fases do processo de desenvolvimento.

Como contribuição, o estudo relatado aponta os principais benefícios e limitações identificados pelos grupos de desenvolvedores que utilizaram o X-CORE em

um cenário de DDS. Vale ressaltar que os resultados produzidos pelo estudo foram obtidos com base na observação direta e não participativa, bem como em entrevistas com os participantes como meio de reafirmar a percepção encontrada.

Para avaliar a adequação do serviço de repositório X-CORE a projetos integrados de DDS e DBC, o estudo foi conduzido na forma de um estudo de caso devido aos seguintes fatores: tipo de avaliação prática desejada; facilidade de planejamento deste tipo de estudo; e ausência de controle sobre as diversas variáveis que compõem o contexto em que o estudo foi executado. Vale ressaltar que, segundo Mafra e Travassos (2006), um estudo de caso é conduzido com o propósito de investigar uma entidade ou fenômeno dentro de um ambiente específico.

O restante deste trabalho está organizado da seguinte forma. A Seção 2 introduz o *framework* arquitetural *ComponentForge* (Oliveira, 2006), no qual o serviço de repositório X-CORE está inserido. Além disso, essa seção também apresenta as principais funcionalidades providas aos usuários pelos serviços que compõem a camada de acesso do X-CORE. Em seguida, a Seção 3 apresenta os objetivos e a metodologia empregada na condução do estudo de caso. A Seção 4 relata a utilização do X-CORE pelos participantes. Já a Seção 5 apresenta uma análise das atividades em um ambiente de DDS do ponto de vista de coordenação, colaboração e integração. Por fim, a Seção 6 apresenta algumas considerações finais.

2. Framework Arquitetural ComponentForge

O Desenvolvimento Baseado em Componentes (DBC) tem se destacado na Engenharia de Software em função do potencial de reduzir a complexidade, o tempo e o custo de desenvolvimento, bem como melhorar a qualidade do software produzido (Szyperski, 2002). Entretanto, o reuso de componentes apresenta alguns fatores limitantes. Bass (2000) identifica a carência de componentes, padrões, certificação e métodos para a construção de componentes como os principais fatores inibidores para o uso de componentes. Complementando, Crnkovic (2003) destaca a atual dificuldade em identificar, selecionar, negociar e recuperar componentes que atendam aos requisitos especificados, e, além disso, que possuam alto grau de reusabilidade e qualidade.

Com o objetivo de superar as limitações expostas, o grupo COMPOSE (*Component Oriented Service Engineering*) propôs o *ComponentForge*, um *framework* arquitetural que adota uma abordagem de arquitetura orientada a serviços (SOA – *Service-Oriented Architecture*), na qual um conjunto de serviços distribuídos, compartilhados, independentes e fracamente acoplados comunicam-se e colaboram entre si através de interfaces e protocolos de comunicação bem definidos. O *ComponentForge* é organizado em três camadas (Figura 1). A camada superior é composta por ferramentas de gerenciamento e desenvolvimento. A camada intermediária é composta pelos serviços de busca, negociação e certificação. A camada inferior é definida pelo serviço de repositório, cuja implementação é denominada X-CORE.

O *X-CORE* atua como um *middleware*, provendo a infra-estrutura adotada pelos demais serviços do *framework* para armazenar, indexar, buscar, recuperar, certificar e negociar variados artefatos de software, incluindo códigos executáveis e fonte, especificações de interfaces e componentes, documentações e diagramas. O *X-CORE* é composto por uma coleção de entidades cooperantes e distribuídas, denominadas *containers,* que, conjuntamente, provêem facilidades através de interfaces bem definidas. Como mostra a Figura 2, o projeto arquitetural do *container* está organizado em três camadas, onde cada uma destas provê um conjunto de componentes, que também adotam a abordagem de serviços.

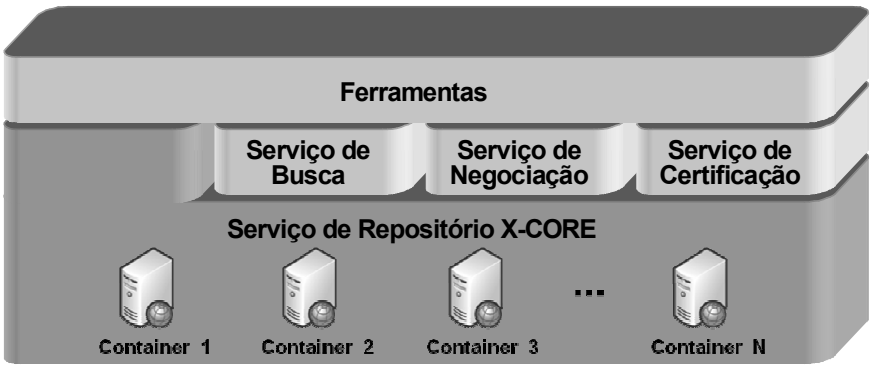

Figura 1. Framework Arquitetural do ComponentForge

Figura 2. Arquitetura em camadas do *container*

A *camada de acesso* permite a interação das *ferramentas* e dos outros serviços do *ComponentForge* com o *container* e com o X-CORE como um todo. A *camada de distribuição* trata de aspectos de distribuição e segurança, tornando possível a descoberta transparente e a segurança das informações armazenadas nos *containers*. Por fim, a *camada de armazenamento* disponibiliza facilidades de persistência, que são adotadas pelos serviços das camadas de acesso e de distribuição, oferecendo meios para armazenar, atualizar e recuperar artefatos. A seguir, por limitação de espaço, apenas as principais funcionalidades dos serviços da *camada de acesso* serão introduzidas, pois é através destes serviços que os diferentes tipos de usuários interagem com o X-CORE.

O *Serviço de Gerenciamento* (Figura 3) provê meios que permitem ao *gerente do container* cadastrar, no seu respectivo *container*, os produtores de software que desejam compartilhar ou até mesmo comercializar seus artefatos. Assim, usando este serviço, o *gerente do container* registra essas entidades em *zonas*, que estão dispostas em uma estrutura hierárquica de nomeação para assegurar unicidade global de nomes.

Figura 3. Serviço de Gerenciamento

O *Serviço de Administração* (Figura 4) possibilita aos produtores de software executar a gestão da sua zona. Nesse serviço, o produtor de software é representado por um ou mais usuários que desempenham o papel de *administrador de zona*. Além disso, para facilitar a organização dos artefatos armazenados em uma zona, os administradores de zona podem subdividi-la em *domínios*, cuja função é agrupar artefatos por área de

aplicação ou produto. Para tal, o serviço de administração possibilita que o *administrador de zona* crie e remova domínios (produtos), assim como registre e remova papéis que serão desempenhados pelos desenvolvedores de software, incluindo a configuração de suas permissões de acesso.

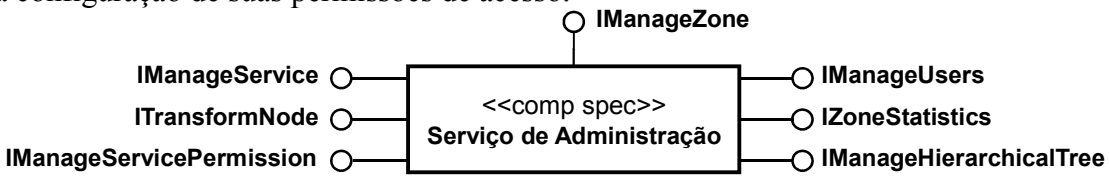

Figura 4. Serviço de Administração

O *Serviço de Desenvolvimento* (Figura 5) possibilita aos desenvolvedores registrar, remover e recuperar artefatos de software durante o processo de desenvolvimento, bem como, controlar as versões desenvolvidas. Além disso, baseado no conceito de visibilidade, os desenvolvedores podem configurar as permissões de acesso para equipes de desenvolvedores externos a sua zona, definindo esquemas de compartilhamento de artefatos entre equipes geograficamente distribuídas. Em complemento, também é possível ao desenvolvedor manipular as informações do modelo de representação de artefatos, denominado *X-ARM* (Schuenck, 2006). Assim, além de permitir registrar informações funcionais, este serviço também permite registrar informações sobre modelos de certificação e de negócio adotados pelos artefatos.

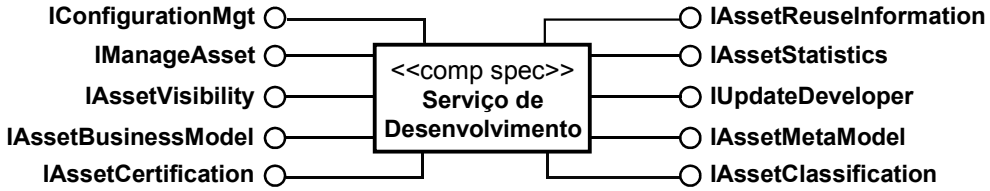

Figura 5. Serviço de Desenvolvimento

Considerando que os demais serviços da camada de acesso (consumidor, qualidade, descoberta e negócio) não foram utilizados no estudo de caso em questão, os mesmos serão descritos de forma bastante resumida. O *Serviço do Consumidor* permite a qualquer usuário acessar e adquirir os artefatos armazenados no repositório, como também os próprios metadados *X-ARM* que os descrevem. Este serviço também permite que os consumidores registrem informações de reuso, tais como as áreas de aplicação, graus de satisfação e comentários sobre o artefato. Vale ressaltar que este serviço permite apenas a manipulação de artefatos sem modelo de negócio, e, portanto, que podem ser livremente recuperados.

O *Serviço de Qualidade* provê meios para que entidades certificadoras registrem e recuperem certificados de qualidade emitidos para artefatos e também para processos de desenvolvimento adotados pelos produtores. O *Serviço de Descoberta* possibilita que sistemas de busca, explorando a estrutura hierárquica de nomeação, possam automaticamente identificar, recuperar e indexar as descrições X-ARM dos artefatos registrados. Por fim, o *Serviço de Negócio* provê meios para que entidades comerciais possam negociar os artefatos de acordo com os seus respectivos modelos de negócio. Assim, para adquirir um artefato com modelo de negócio, o consumidor deve negociar com a entidade comercial, que, por sua vez, recupera o artefato no X-CORE através do serviço de negócio.

A seguir, os objetivos e a metodologia adotada no estudo de caso são apresentados. Vale ressaltar que o estudo de caso se limita a avaliar os serviços de gerenciamento, administração e desenvolvimento do X-CORE, pois são os que provêem funcionalidades a projetos de desenvolvimento distribuído baseados em componentes.

3. Objetivos e Metodologia

Com o objetivo geral de avaliar o uso e a aplicabilidade do X-CORE como uma infra-estrutura de suporte a projetos de desenvolvimento distribuído baseado em componentes, o mesmo foi utilizado na disciplina Desenvolvimento Baseado em Componentes, oferecida no Departamento de Informática (DI) da Universidade Federal da Paraíba (UFPB). Esta disciplina foi ministrada para um grupo de 14 alunos do curso de Bacharelado em Ciências da Computação, que constituíram as equipes de desenvolvimento participantes do estudo de caso. A disciplina teve como escopo principal a apresentação de conceitos e técnicas relacionadas ao DBC, adotando o processo de desenvolvimento *UML Components* (Cheesman, 2001).

Vale ressaltar que o X-CORE não está vinculado a nenhum processo de desenvolvimento de software, embora tenha sido concebido para atender aos requisitos de abordagens de DDS e DBC. Assim, a princípio, o X-CORE pode ser adotado em qualquer processo que tenha como requisito a colaboração e o compartilhamento de artefatos de software produzidos por equipes possivelmente remotas e independentes, que estão engajadas em projetos de desenvolvimento distribuído baseados em componentes. Considerando a natureza colaborativa, compartilhada e distribuída do repositório, o X-CORE tem o potencial de facilitar e possivelmente incrementar o reuso em larga escala de componentes de software desenvolvidos por terceiros.

Para contextualizar um ambiente de DDS, com requisitos de coordenação de atividades, colaboração entre equipes independentes e compartilhamento de artefatos, os alunos participantes foram divididos em diferentes equipes para trabalhar cooperativamente em um dado projeto de software. Durante a formação das equipes, alguns participantes foram designados para exercer os papéis de gerente de projeto, engenheiros de software e programadores, que, segundo Kroll e Kruchten (2003), viabiliza a formação de grupos auto-suficientes e diminui a necessidade de comunicação entre os grupos. Vale ressaltar que, para simular a distância física, as equipes não se comunicavam diretamente, mas apenas através dos artefatos depositados no repositório.

Definido o escopo, a metodologia aplicada para a realização deste estudo de caso foi dividida em três etapas: *configuração do ambiente*, *especificação de componentes*, e *implementação de componentes*. Na etapa de configuração do ambiente, as equipes foram cadastradas no X-CORE pelos gerentes dos *containers*, sendo que cada equipe foi representada por uma zona diferente na hierarquia de nomeação. Para representar o aspecto de distribuição física, as equipes foram configuradas em diferentes *containers*. Em seguida, os papéis dos diferentes usuários (administradores de zona e desenvolvedores) foram configurados. Posteriormente, os usuários foram devidamente registrados. Por fim, os administradores de zona configuram suas zonas, criando a estrutura de domínios para agrupar os artefatos na hierarquia de nomeação. Nesta primeira etapa, foi possível avaliar a realização das atividades do *gerente do container* na configuração das zonas das respectivas equipes, e do *administrador de zona* na criação de papéis, registro de usuários e a criação de domínios.

Na etapa de especificação de componentes, os desenvolvedores foram produzindo e registrando os diversos artefatos que compõem as atividades das fases de requisitos e especificação do processo de desenvolvimento *UML Components*. Nesta segunda etapa, foi possível avaliar o grau de dificuldade encontrado pelo *desenvolvedor* na manipulação dos identificadores dos artefatos, a geração e empacotamento das descrições X-ARM dos artefatos e o registro dos mesmos no serviço repositório.

Na etapa de implementação de componentes, com o objetivo de simular a cooperação de equipes dispersas geograficamente, as equipes fizeram uma permuta dos

seus projetos. Assim, a equipe responsável pela especificação de um dado conjunto de componentes na segunda etapa, ficou responsável pela implementação dos componentes que foram especificados por outra equipe na terceira etapa. Portanto, de forma similar a segunda etapa, os desenvolvedores foram produzindo e registrando os diversos artefatos que compõem as atividades da fase de provisionamento do processo *UML Components*. Nesta terceira etapa, foi possível avaliar, principalmente, o grau de dificuldade para definir esquemas de visibilidade dos artefatos, visando o compartilhando de artefatos entre diferentes equipes. Além disso, questões não funcionais foram observadas, tais como os aspectos de segurança e de distribuição do X-CORE.

Após a conclusão da terceira etapa, entrevistas individuais e de forma estruturada foram realizadas com os integrantes das equipes com o propósito de identificar possíveis problemas funcionais, e, principalmente, aqueles que influenciaram negativamente na produtividade das equipes durante o processo de desenvolvimento.

4. Execução do Estudo de Caso

A infra-estrutura adotada pelas equipes foi composta por três *containers*, que, conjuntamente, proviam as funcionalidades do X-CORE. Para viabilizar o acesso remoto aos *containers*, foi desenvolvida uma ferramenta cliente bastante simples, denominada *X-CORE Client*. Para facilitar o entendimento e uso das funcionalidades do X-CORE, esta ferramenta foi concebida de modo a integrar apenas as funcionalidades requeridas para o desenvolvimento do estudo de caso.

O estudo de caso foi realizado considerando dois projetos de software: *sistema de prontuário eletrônico* e *sistema acadêmico*. O sistema de prontuário eletrônico foi concebido com a intenção de ser adotado no hospital universitário da UFPB. O sistema acadêmico foi concebido apenas para o estudo de caso. Além disso, o sistema acadêmico foi decomposto em cinco serviços: *serviço do professor*, *serviço da biblioteca*, *serviço administrativo*, *serviço de matrícula* e *serviço de pesquisa*.

Uma vez definido o escopo dos projetos, na primeira etapa de configuração do ambiente do estudo de caso, os alunos foram divididos em seis pequenas equipes, sendo que cinco delas foram conjuntamente alocadas ao projeto do sistema acadêmico, enquanto a outra equipe foi alocada individualmente ao projeto do sistema de prontuário eletrônico. Assim, o estudo de caso foi conduzido com um projeto conjunto executado por cinco equipes, e outro projeto independente executado por uma única equipe.

O projeto conjunto favoreceu o estudo de caso, pois foi configurado na forma de um projeto de desenvolvimento distribuído baseado em componentes, no qual foi simulada a colaboração de equipes distribuídas que cooperavam para o desenvolvimento de um mesmo projeto de software. Vale ressaltar que o sistema acadêmico foi concebido adotando a arquitetura baseada em serviço (SOA), permitindo assim que cada equipe fosse alocada a um serviço diferente, e, desta forma, tivesse responsabilidades distintas durante o processo de desenvolvimento, gerando e manipulando artefatos diferentes. Além disso, como já mencionado anteriormente, para simular a distância física, as equipes não se comunicavam diretamente, mas apenas colaboravam através do compartilhamento de artefatos depositados no repositório.

A Figura 6 mostra a organização hierárquica e a localização das equipes nos três *containers* adotados no estudo de caso. Como pode ser observado, os nomes das equipes foram utilizados para a criação dos identificadores das zonas (*equipeA*, *equipeB*, *equipeC*, *equipeD*, *equipeE* e *equipeF*). Para criar tais zonas, inicialmente, o gerente de cada container utilizou operações da interface *IManageZone* do serviço de gerenciamento para autorizar a existência das zonas. Em seguida, as zonas criadas foram

cadastradas na zona raiz, permitindo o correto funcionamento do processo de resolução de nomes. Para cadastrar na zona raiz, o administrador de cada zona utilizou operações da interface *IManageHierarchicalTree* do serviço de administração.

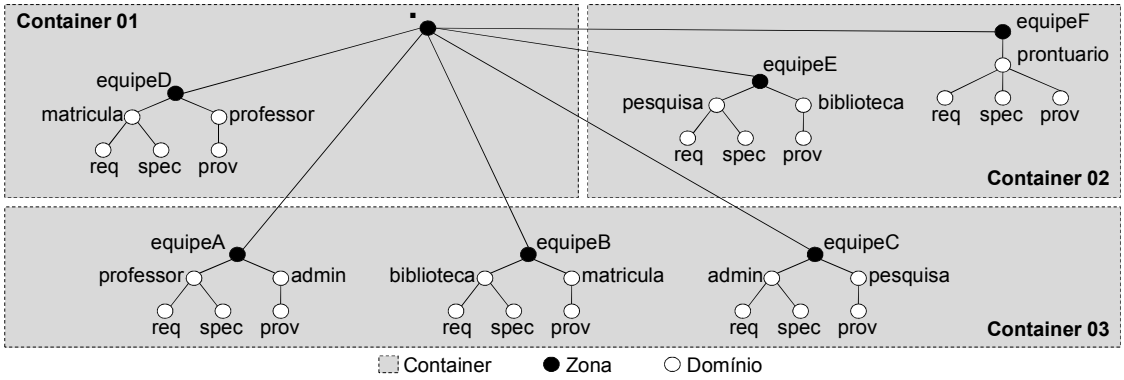

Figura 6. Estrutura hierárquica de nomes nos containers

É possível observar na Figura 6 que, tanto no sistema de prontuário eletrônico quanto em cada serviço do sistema acadêmico, artefatos das fases de requisitos, especificação e provisionamento são agrupados em diferentes domínios (*req*, *spec* e *prov*). Além disso, no caso das equipes *A*, *B*, *C*, *D* e *E*, cada serviço do sistema acadêmico possui dois domínios, configurados nas duas equipes responsáveis pela especificação (*req* e *spec*) e pela implementação (*prov*) do serviço. Para criar esses domínios, os administradores das respectivas zonas utilizaram operações da interface *IManageZone* do serviço de administração.

Após configurar as zonas e domínios, os administradores de zonas cadastraram os *desenvolvedores* de cada equipe. Para tal, utilizaram operações da interface *IManageUsers* do serviço de administração para definir papéis, atribuir permissões e associar os *desenvolvedores* aos seus respectivos papéis. Para o estudo de caso, os desenvolvedores só tiveram permissão de executar operações nas interfaces *IManageAsset*, *IAssetVisibility*, *IConfigurationMgmt*, *IUpdateDeveloper* e *IAssetMetaModel* do serviço de desenvolvimento, pois são as interfaces que provêem as funcionalidades necessárias para o estudo em questão.

Concluída a etapa inicial de configuração do ambiente, deu-se início a segunda etapa de especificação de componentes, na qual os desenvolvedores seguiram as atividades definidas nas fases de requisitos e especificação do processo *UML Components*, produzindo, reusando e registrando no X-CORE os seus respectivos artefatos. Para registrar e reusar os artefatos, os desenvolvedores utilizaram operações da interface *IManageAsset* do serviço de desenvolvimento. No entanto, antes de registrar cada artefato produzido, os desenvolvedores geravam manualmente as descrições X-ARM dos artefatos, e, em seguida, empacotavam as descrições e o próprio artefato. Somente após gerar o pacote do artefato é que as operações do serviço de desenvolvimento eram utilizadas para registrar o artefato.

Vale ressaltar que, nesta etapa de especificação de componentes, as equipes produziram e registram mais de 60 artefatos, tais como casos de uso, especificações de interfaces e componentes, e diagramas de interação. Por limitação de espaço, não é possível apresentar detalhes destes artefatos produzidos e registrados.

Como mencionado na seção anterior, a fim de simular a cooperação de equipes dispersas geograficamente, na etapa de implementação de componentes, as equipes foram permutadas, de modo que cada equipe alocada à especificação de um dado serviço do sistema acadêmico foi realocada à implementação de outro serviço do mesmo

sistema. Por exemplo, a equipe *A* que era responsável pela especificação do serviço do professor, foi realocada para a implementação do serviço administrativo. No entanto, no caso do sistema de prontuário eletrônico, como apenas uma equipe atuava neste sistema, a mesma ficou responsável pela especificação e implementação do sistema.

Conseqüentemente, em função da permuta de responsabilidades entre as equipes, foi necessário configurar o compartilhamento de artefatos entre equipes. Por exemplo, a equipe *A* que foi responsável pela implementação do serviço de administração precisou acessar os artefatos gerados pela equipe *C* referentes à especificação deste serviço.

Vale ressaltar que o esquema de permissões de acesso, definido na etapa de configuração do ambiente, não contempla mecanismos de controle de acesso entre equipes de desenvolvedores alocadas a diferentes zonas. Neste caso, o mecanismo de *visibilidade externa* provido pelo X-CORE teve que ser adotado para gerenciar o acesso a artefatos entre diferentes equipes. Portanto, uma vez que as equipes estavam alocadas a zonas distintas e precisavam reusar artefatos entre si, os desenvolvedores configuraram esquemas de visibilidade para os artefatos. Para tal, os desenvolvedores utilizaram operações da interface *IAssetVisibility* do serviço de desenvolvimento.

A Figura 7 ilustra dois esquemas de visibilidade configurados no estudo de caso. A Figura 7a mostra a configuração do esquema de visibilidade padrão dos artefatos do domínio *equipeE.pesquisa*. Nesta configuração, os desenvolvedores da equipe *C*, alocados para a implementação do serviço de pesquisa, têm acesso ao artefato modelo conceitual de negócio (*bcm 1.0*) gerado pela equipe *E*, que foi responsável pela especificação do serviço de pesquisa. A Figura 7b mostra a configuração do esquema de visibilidade padrão dos artefatos do domínio *equipeC.admin*. Nesta configuração, os desenvolvedores da equipe *A*, alocados para a implementação do serviço administrativo, têm acesso ao artefato interfaces de negócio (*BusinessInterfaces-1.0*) gerado pela equipe *C,* que foi responsável pela especificação do serviço administrativo.

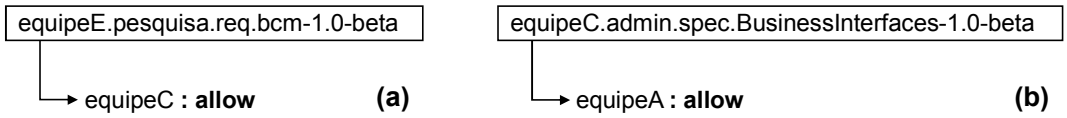

Figura 7. Esquemas de visibilidade

Após a configuração da visibilidade, deu-se início a etapa de implementação de componentes, na qual os desenvolvedores seguiram as atividades definidas na fase de provisionamento do *UML Components*, reusando artefatos das fases anteriores para produzir e registrar no X-CORE os novos artefatos da fase de provisionamento. Para reusar e registrar os artefatos, os desenvolvedores utilizaram operações da interface *IManageAsset* do serviço de desenvolvimento. Além disso, antes de registrar cada artefato produzido, os desenvolvedores também geravam manualmente as descrições X-ARM dos artefatos, e, em seguida, empacotavam as descrições e o próprio artefato.

Nesta etapa de implementação de componentes, as equipes produziram e registram diversos artefatos, referentes à implementação dos componentes do sistema acadêmico e do sistema de prontuário eletrônico. Novamente, por limitação de espaço, não é possível apresentar detalhes destes artefatos produzidos e registrados.

Vale ressaltar que o *UML Components* é um processo iterativo, ou seja, ao final de cada iteração as equipes podem avaliar os resultados e, caso necessário, modificar e registrar novas versões dos artefatos. Por exemplo, durante o estudo de caso, novos requisitos foram identificados para o sistema de prontuário eletrônico, forçando a equipe *F* a gerar e registrar novas versões dos artefatos usando operações das interfaces *IManageAsset* e *IConfigurationMgmt* do serviço de desenvolvimento.

5. Análise do Estudo de Caso

Como mencionado na Seção 3, o estudo de caso foi dividido em três etapas. Na primeira etapa, foram avaliadas as funcionalidades providas ao *gerente de container* e *administrador de zona*. Na segunda e terceira etapas, foram avaliadas as funcionalidades providas aos *desenvolvedores*, dando ênfase na terceira etapa à cooperação entre equipes distintas. A seguir, do ponto de vista de coordenação, colaboração e integração das atividades em um ambiente de DDS, serão identificadas as limitações e os pontos positivos identificados durante o estudo de caso.

Na primeira fase, os participantes relataram que a principal dificuldade foi a usabilidade da interface gráfica da ferramenta X-CORE *Client*, pois a mesma não foi de fácil entendimento e em certos momentos dificultava o uso do X-CORE. No entanto, é importante ressaltar que o X-CORE *Client* não foi desenvolvido para ser uma ferramenta *CASE*, mas apenas para evitar a adoção de ferramentas de acesso a serviços Web de mais baixo nível de abstração. Considerando que a usabilidade do ferramental de apoio tem alto impacto no sucesso de um projeto, ferramentas *CASE* propriamente ditas deverão ser integradas em uma próxima etapa da evolução do X-CORE.

Do ponto de vista de coordenação, no tocante as atividades do *gerente de container* e *administrador de zona*, os participantes relataram que, em geral, não encontraram dificuldades na criação de zonas, domínios e papéis, bem como no registro dos usuários. No entanto, ficou evidente no estudo de caso a necessidade de uma ferramenta de configuração e gestão de projetos, que, com base na definição de alto nível da arquitetura do sistema e na alocação das equipes às diversas fases do processo de software, seja capaz de configurar automaticamente zonas, domínios, papéis e usuários no X-CORE, além de gerenciar o andamento das várias atividades paralelas.

Na segunda e terceira etapas, quanto ao papel de *desenvolvedor*, os usuários relataram que não encontraram dificuldades para registrar artefatos e gerenciar as diferentes versões. No entanto, encontraram certa dificuldade em gerar manualmente as descrições X-ARM dos artefatos. Tal dificuldade era esperada em função da grande quantidade de informações manipulada pelo X-ARM. Para minimizar esta dificuldade, a ferramenta *X-Packager* (Schuenck, 2006) foi desenvolvida para facilitar a geração semi-automática das descrições X-ARM. No entanto, os participantes optaram por não usar a ferramenta, pois o *plug-in* somente é disponibilizado para a *IDE Eclipse*, e muitos participantes realizaram a etapa de implementação na *IDE NetBeans*.

Na terceira etapa, as equipes foram permutadas, criando assim a necessidade de compartilhamento e integração de artefatos. Nessa etapa, os artefatos registrados tiveram esquemas de visibilidade configurados. Do ponto de vista do compartilhamento, segundo os participantes, não houve dificuldade de configurar os esquemas de visibilidade e compartilhar os artefatos. No entanto, do ponto de vista de integração, a principal dificuldade foi a implementação das especificações, pois a maioria dos participantes não tinha experiência suficiente na implementação de componentes. Logo, a deficiência não está relacionada ao X-CORE, mas ao perfil dos desenvolvedores.

Do ponto de vista de colaboração, na terceira etapa, identificou-se a necessidade de mecanismos mais robustos de gerência de configuração, incorporando facilidades de *bug tracking* para complementar o controle de versão adotado no X-CORE, provendo uma base de dados sobre problemas encontrados durante a execução do projeto de software. Além disso, o estudo de caso mostrou que a comunicação baseada exclusivamente em artefatos gera a necessidade de mais iterações no processo de desenvolvimento, retardando assim a conclusão das tarefas. Desta forma, o X-CORE também deve evoluir no sentido de integrar facilidades que viabilizem a comunicação

direta entre as equipes participantes (*e-mail*, *chat*), resolvendo assim possíveis conflitos e estabelecendo relações de confiança entre os desenvolvedores.

6. Considerações Finais

No contexto de projetos de desenvolvimento distribuído baseados em componentes, prover uma infra-estrutura completa, integrada e adequada a todos os atores envolvidos é um desafio. Neste contexto, apesar das limitações apontadas, podemos considerar satisfatórias as facilidades providas pelo X-CORE para projetos integrados de DDS e DBC.

Apesar do grande número de serviços e facilidades que compõem o X-CORE, a avaliação realizada evidencia que as equipes foram capazes de entender e utilizar o serviço de repositório sem maiores dificuldades e sem a necessidade de utilização de outras ferramentas de suporte, tais como *CVS* ou *Bugzilla*. Além disso, em relação aos aspectos de distribuição e segurança, o X-CORE mostrou-se efetivo e seguro, provendo transparência de localização, autenticação, integridade e confidencialidade.

Por fim, é importante destacar que os resultados obtidos não podem ser generalizados, pois, segundo Mafra e Travassos (2006), os dados obtidos em um estudo de caso apenas apontam os efeitos em uma situação em particular, não podendo ser generalizado para todos os outros contextos. Assim, como trabalho futuro, mais estudos de caso devem ser realizados, principalmente, em ambientes de fábricas de softwares.

Referências

Babar, M. A.; Zhu, L; Jeffery, R. (2004) "A framework for Classifying and Comparing Software Architecture Evaluation Methods". In: Proceedings of the Australian Software Engineering Conference, pp 309-318, Melbourne, Australia.

Bass, L.; Buhman, C.; Dorda, S.; Long, F; Robert, J.; Seacord, R.; Wallnau, K. (2000). "Market Assessment of Component-Based Software Engineering". SEI, Technical Report CMU/SEI-2001-TN-007.

Cheesman, J.; Daniels, J. (2001). "UML Components". Boston: Addison-Wesley.

Crnkovic, I. (2003) "Component-Based Software Engineering –New Challenges in Software Development". Information Technology Interfaces, pp. 127-133, Croatia.

Kroll, P. ; Kruchten, P. (2003). "The Rational Unified Process Made Easy: a Practitioner's Guide to the RUP". Addison-Wesley.

Mafra, S. N.; Travassos, G. H. (2006). "Estudos Primários e Secundários em Engenharia de Software". Relatório Técnico – ES - 687/06. Programa de Engenharia de Sistemas e Computação – COPPE/UFRJ. Rio de Janeiro – RJ.

Oliveira, J. P. F.; Santos, M. S.; Elias, G. (2006). "ComponentForge: Um Framework Arquitetural para Desenvolvimento Distribuído Baseado em Componentes". VI Workshop de Desenvolvimento Baseado em Componentes. Recife-PE.

Oliveira, J. P. F.; Brito, T.; Júnior, S. R.; Elias, G. (2007). "Um Serviço de Repositório Compartilhado e Distribuído para Suporte ao Desenvolvimento Baseado em Componentes". 22º Simpósio Brasileiro de Engenharia de Software, João Pessoa-PB.

Schuenck, Michael. (2006). "X-ARM: Um Modelo de Representação de Artefatos de Software". Dissertação de Mestrado, DIMAp-UFRN, Natal-RN.

Szyperski, Clemens. (2002) "Component Software: Beyond Object-Oriented Programming". 2th Edition, Addison-Wesley.

Travassos, G. H.; Gurov, D.; Amaral, E. A. G. (2002). "Introdução a Engenharia de Software Experimental". Relatório Técnico – ES – 590/02. Programa de Engenharia de Sistemas e Computação – COPPE/UFRJ. Rio de Janeiro – RJ.