Suporte a Relacionamentos entre Equipes Distribuídas em um Repositório de Componentes de Software

Yuri Feitosa Negócio[1], Gledson Elias[1]

[1]COMPOSE – Component Oriented Service Enginnering
Departamento de Informática – Universidade Federal da Paraíba
Paraíba, Brasil

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

Abstract. *Taking into account that the integration of Distributed Software Development (DSD) and Component Based Development (CBD) approaches improves the benefits and promises of both, the goal of this paper is to describe the mechanisms adopted by a component repository service for supporting the different types of sourcing relationships that can be established among distributed teams, which are involved in component-based distributed development projects.*

Resumo. *Considerando que a integração de abordagens de Desenvolvimento Distribuído de Software (DDS) e Desenvolvimento Baseado em Componentes (DBC) potencializa os benefícios e as promessas de ambas, este artigo tem por objetivo descrever os mecanismos adotados por um serviço de repositório de componentes para prover suporte aos diferentes tipos de relacionamentos de sourcing que podem ser definidos entre equipes distribuídas envolvidas em projetos de desenvolvimento distribuído baseados em componentes.*

1. Introdução

O impacto da globalização, principalmente na unificação dos mercados, vem aumentando significativamente a cooperação entre empresas de desenvolvimento de software de diferentes países, e, até mesmo, de diferentes regiões de um mesmo país. Neste sentido, as empresas têm adotado equipes geograficamente distribuídas de forma a obter diversas vantagens, tais como: a redução do custo pelo uso de mão de obra mais barata, a contratação de especialistas, e a redução do *time-to-market* usando equipes em diferentes fusos-horário [Herbsleb 2001].

À medida que as equipes passaram a se organizar de forma distribuída, as empresas começaram a estudar meios para obter um melhor desempenho neste novo cenário [Prikladnicki 2006]. Sendo assim, a abordagem de Desenvolvimento Distribuído de Software (DDS) tornou-se nos últimos anos a norma para a indústria de software [Damian 2006]. Apesar dos benefícios, abordagens de DDS ainda estão sujeitas a diversos problemas, dentre os quais podemos destacar aqueles relacionados à comunicação entre equipes e à coordenação das tarefas paralelas.

Para minimizar os tradicionais problemas do DDS, o Desenvolvimento Baseado em Componentes (DBC) tem sido uma alternativa promissora. Originalmente, o DBC emergiu como uma promessa para favorecer o reuso de software, além de oferecer uma série de outras potencias vantagens como, por exemplo, a redução do custo de desenvolvimento e do prazo de entrega dos produtos de software, além da melhoria na qualidade do software [Szyperski 2000].

Conseqüentemente, abordagens de DDS e DBC compartilham benefícios similares e a integração de ambas tem o potencial de minimizar os problemas comuns. Por um lado, DDS facilita a globalização da indústria de software, tornando mais fácil o desenvolvimento de sistemas baseados em componentes em regiões dispersas e possibilitando a compra de componentes a produtores em qualquer lugar do mundo [Kotlarsky 2005]. Por outro lado, em geral, os projetos de DDS são insuficientemente descompostos, resultando na sobreposição e desentendimento de responsabilidades, ocasionando problemas de comunicação e levando ao insucesso do projeto. Portanto, de forma complementar ao DDS, a adoção de uma arquitetura baseada em componentes força os arquitetos e desenvolvedores a encapsular melhor as funcionalidades em componentes coesos e bem documentados [Repenning 2001]. Desta forma, o DBC facilita a decomposição dos sistemas em componentes independentes que se integram através de contratos bem estabelecidos, minimizando a necessidade de comunicação entre equipes e facilitando a coordenação de tarefas.

Nesta direção, atualmente, existem inúmeras forças, tanto técnicas quanto econômicas, que motivam a adoção do desenvolvimento distribuído integrado ao desenvolvimento baseado em componentes [Kotlarsky 2005]. Entretanto, um dos principais requisitos para obter os benefícios do DBC é a existência de um mercado global de componentes de software [Szyperski 2000]. Considerando o crescimento dos mercados *online*, que reduzem o custo inicial de projetos de *offshore outsourcing* para pequenas e médias empresas [Radkevitch 2006], o cenário torna-se favorável à implantação de um mercado global de componentes de software baseado em repositórios distribuídos e compartilhados de artefatos de software produzidos por diferentes empresas geograficamente dispersas.

Neste cenário, onde a integração de abordagens de DDS e DBC potencializa os benefícios e promessas de ambas, a relação entre produtores e consumidores em um mercado global permite o estabelecimento de diversos novos tipos de relacionamentos entre empresas e suas respectivas equipes de desenvolvimento. Desta forma, a infra-estrutura tecnológica de suporte a tal mercado global de componentes deve prover formas eficientes e seguras de acessar os repositórios de componentes e coordenar os relacionamentos e interações entre os produtores e os consumidores de componentes.

No contexto de integração de DDS e DBC, este artigo tem por objetivo descrever os mecanismos adotados pelo serviço de repositório de componentes, denominado X-CORE [Oliveira 2007], para prover suporte aos diferentes e variados tipos de relacionamentos de *sourcing* que podem ser definidos entre equipes distribuídas de diferentes empresas, envolvidas como clientes ou fornecedoras em projetos de desenvolvimento distribuído baseados em componentes. Como inovação, os mecanismos descritos tornam o repositório X-CORE uma infra-estrutura distribuída e compartilhada, que tem o potencial de definir um mercado global de artefatos de software, no qual equipes remotas e independentes colaboram e compartilham artefatos de acordo com os seus respectivos relacionamentos de *sourcing*.

O restante deste artigo está organizado da seguinte forma. A Seção 2 identifica os principais relacionamentos de *sourcing* encontrados na literatura, destacando exemplos no contexto de abordagens de DBC. A Seção 3 apresenta uma breve introdução ao X-CORE, destacando os mecanismos que possibilitam a configuração e gerência dos relacionamentos de *sourcing*. Em seguida, considerando os mecanismos providos pelo X-CORE, a Seção 4 apresenta exemplos de configuração dos diferentes tipos de relacionamentos de *sourcing*. Por fim, a Seção 5 apresenta algumas considerações finais e trabalhos futuros.

2. Relacionamentos em Cenários Integrados de DDS e DBC

Os desenvolvedores de componentes estão buscando novas oportunidades de negócio na criação de componentes para as mais variadas aplicações. Inicialmente, as oportunidades de venda de componentes para terceiros criou um mercado de componentes restrito a componentes de interface de usuário, e, gradativamente, vem se transformando em um mercado de componentes de negócio [Bass 2000].

Paralelamente, por motivos econômicos, diversas empresas estão migrando para projetos de desenvolvimento globalmente distribuído e utilizando *outsourcing* para seus produtos e serviços. Em conseqüência, surgiram diversos mercados eletrônicos de *outsourcing* para produtos de TI: *Elance Online* (www.elance.com), *Prosavvy (*www.prosavvy.com), *Guru* (www.guru.com) e *RentACoder* (www.rentacoder.com). Estes mercados consideram apenas o cenário tradicional de *outsourcing* onde apenas duas partes interagem, ou seja, uma empresa cliente delega o controle sobre uma ou mais atividades para uma empresa fornecedora externa a quem contratou o serviço. Segundo Robbison [Robbison 2004], esta é a forma mais simples de ser operacionalizar cenários de *outsourcing*.

No entanto, além da categoria tradicional de *outsourcing*, existem diversos outros tipos de relacionamento entre clientes e fornecedores. De acordo com Hyder [Hyder 2006] e Gallivan [Gallivan 1999], estes relacionamentos se encaixam nas categorias ilustradas na Figura 1. A seguir, uma breve descrição das categorias é apresentada, destacando exemplos de possíveis cenários no contexto de DBC.

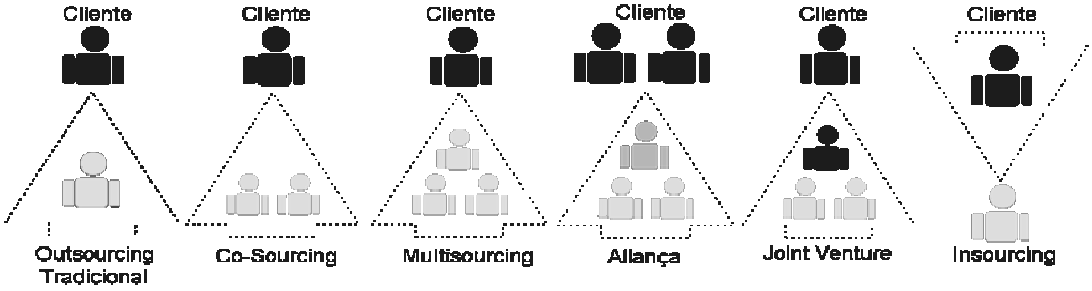


Figura 1. Relacionamentos entre clientes e fornecedores

***Outsourcing* Tradicional.** Nesta categoria, uma empresa cliente contrata uma única empresa fornecedora para desenvolver um determinado serviço ou produto. No contexto de DBC, este tipo de relacionamento pode ser representado através de uma empresa cliente que contrata uma empresa fornecedora, especializada em determinado domínio de aplicação para o desenvolvimento de um ou mais componentes.

***Co-Sourcing*.** Esta categoria de relacionamento considera que dois fornecedores trabalham conjuntamente para entregar um determinado produto ou serviço a uma empresa cliente. Considerando o contexto de DBC, este relacionamento pode ser operacionalizado através da divisão das fases do processo de desenvolvimento baseado em componentes entre as empresas fornecedoras. Por exemplo, uma empresa cliente *X* necessita de um determinado componente, para tal contrata as empresas fornecedoras *Y* e *Z* para desenvolvê-lo. Neste cenário, a empresa *Y* pode ser responsável pela especificação e implementação do componente, enquanto a empresa Z pode ser responsável pelos testes.

***Multisourcing*.** Nesta categoria de relacionamento, diversos fornecedores entregam produtos ou serviços para um único cliente. Desta forma, considerando que a empresa cliente recebe artefatos de diferentes fornecedores, a responsabilidade de integração e gerenciamento dos múltiplos fornecedores é do próprio cliente. No contexto de DBC,

um exemplo de cenário seria uma determinada empresa cliente decompor sua aplicação em diversos componentes e delegar a cada empresa fornecedora a tarefa de implementar e testar um ou mais componentes.

Aliança. Esta categoria considera a colaboração de múltiplos fornecedores com o objetivo de atender as necessidades de um ou mais clientes. Nesta categoria é possível que um dos fornecedores seja designado como o responsável pelo relacionamento com cada cliente. Ao contrário do cenário anterior (*multisourcing*), onde a empresa cliente mantém o relacionamento com os diversos fornecedores, na aliança, a empresa cliente mantém relacionamento apenas com o fornecedor responsável pela aliança, que, por sua vez, distribui as tarefas associadas ao desenvolvimento dos componentes entre os outros participantes da aliança, e, ao final, realizar a integração dos componentes.

***Joint Venture*.** Nesta categoria, duas ou mais empresas unem seus recursos criando uma nova entidade para executar um ou mais projetos por um determinado período de tempo. No contexto de DBC, um possível cenário pode ser definido por uma empresa cliente que possui a demanda por componentes de um determinado domínio de aplicação, mas, para reter o conhecimento e participar do processo, associa-se a uma empresa fornecedora com conhecimentos especializados, criando uma nova empresa que tem como objetivo desenvolver tais componentes.

***Insourcing*.** Esta categoria considera que um cliente seleciona como fornecedor um grupo ou subsidiária da própria empresa. Entretanto, neste relacionamento, o grupo selecionado como fornecedor é gerenciado como uma entidade externa e compete com outros potenciais fornecedores. No contexto de DBC, um exemplo deste tipo de relacionamento pode ser definido entre o cliente que demanda por um componente especifico, e, após considerar todas as propostas de fornecedores, seleciona aquela apresentada por um dado grupo ou subsidiária da própria empresa com especialistas no domínio do problema.

3. Suporte a Relacionamentos DDS no X-CORE

Aliando os potenciais benefícios do crescente mercado de DDS e DBC, o X-CORE [Oliveira 2007] provê uma infra-estrutura de suporte a processos de desenvolvimento distribuído baseados em componentes de software, disponibilizados em um serviço de repositório compartilhado e distribuído. O serviço de repositório é dito compartilhado porque vários produtores podem registrar seus artefatos. Por outro lado, o serviço repositório é dito distribuído porque é constituído de um conjunto de entidades de software cooperantes e geograficamente dispersas, denominadas *containers*.

Adotando uma arquitetura em camadas e orientada a serviços, o X-CORE tem o potencial de incrementar o reuso de software, provendo suporte ao armazenamento, busca, certificação e negociação de artefatos de software produzidos ao longo dos processos de desenvolvimento. Além disso, o X-CORE inclui facilidades de controle de versão, gerência de métricas de reuso, como também aspectos de segurança relacionados à autenticação, integridade, privacidade e controle de acesso.

Para viabilizar a adoção de processos de desenvolvimento distribuído, o X-CORE provê mecanismos de suporte aos diferentes tipos de relacionamentos de *sourcing* que podem ser definidos entre equipes distribuídas. Para tal, o X-CORE adota um *esquema de nomeação*, que provê transparência de localização e distribuição dos fornecedores e seus respectivos artefatos, como também, define um mecanismo de *visibilidade externa*, que controla como clientes e fornecedores podem compartilhar artefatos. Considerando que o objetivo deste artigo é apresentar os mecanismos de

suporte aos relacionamentos de *sourcing*, a seguir serão descritos os esquemas de nomeação e visibilidade externa.

Esquema de Nomeação. Em função da natureza compartilhada e distribuída do X-CORE, o esquema de nomeação adotado facilita a organização dos artefatos armazenados e permite a localização e recuperação não ambígua dos mesmos. Neste esquema, os nomes são organizados em uma estrutura hierárquica onde as folhas são os artefatos armazenados. Por sua vez, os nós internos correspondem a dois tipos de entidades (zonas e domínios). Cada zona representa uma empresa produtora de software e suas famílias de produtos. Ou seja, a zona é a representação virtual de uma empresa ou grupo de desenvolvimento no mundo real. Além disso, uma determinada zona permite a incorporação de subzonas, podendo assim representar grupos ou subsidiárias da empresa representada pela zona pai. Já os domínios são subdivisões das zonas que possibilitam uma melhor organização dos artefatos. Assim, a função de um domínio é agrupar um conjunto de artefatos relativos a um mesmo projeto de software. Cada zona reside completamente em um servidor geograficamente disperso (*container*). Para prover transparência de localização e distribuição, o X-CORE atua como um *middleware* de propósito especial responsável por resolver de maneira transparente os problemas relativos à distribuição dos *containers* e prover uma série de serviços para seus clientes.

A Figura 2 representa uma possível configuração distribuída do X-CORE, evidenciando sua natureza distribuída. Como pode ser observado, o primeiro *container* mantém a raiz da árvore de nomeação (zona raiz) e a zona *br*. Por sua vez, o segundo *container* mantém a zona *empresaA* que possui a subzona *subsidiariaA*, representando assim a empresa *A* e a sua respectiva subsidiária. Por fim, o terceiro *container* mantém a zona *empresaB* que possui o domínio *projetoA*, representando a empresa B e um de seus projetos de software, cujos artefatos estão inseridos como folhas. Para assegurar unicidade global, os nomes de zonas, domínios ou artefatos são formados pela concatenação dos nomes existentes na parte superior da hierarquia. Por exemplo, na Figura 2, o nome *br.empresaA.subsidiariaA* refere-se a subsidiária da empresa A. Além de definir identificadores globalmente únicos, o esquema hierárquico proposto também provê transparência de localização, pois os nomes não referenciam a localização física, mas apenas a organização lógica de produtores e seus artefatos.

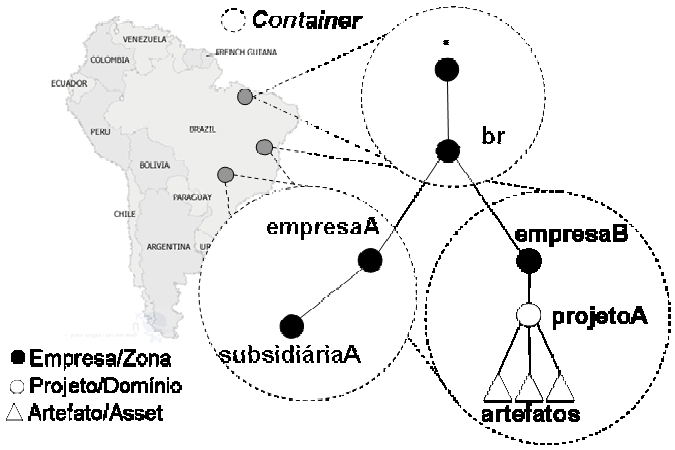


Figura 2. Esquema de nomeação e distribuição

Visibilidade Externa. Para simplificar o compartilhamento de artefatos e controlar o acesso entre diferentes equipes de diferentes empresas (clientes ou fornecedoras), o X-

CORE define o conceito de visibilidade. De forma simplificada, a visibilidade controla como uma equipe, cadastrada em uma zona possivelmente remota, pode acessar (ou não) os artefatos produzidos por outra equipe e armazenados em outra zona. Na configuração da visibilidade, para cada artefato registrado em um projeto de uma determinada empresa, podem ser definidas as permissões de visibilidade, indicando quais outras empresas cujas equipes de desenvolvedores são autorizadas ou proibidas de recuperar o artefato. Portanto, a visibilidade atua como um mecanismo de controle de acesso entre diferentes equipes de desenvolvimento de diferentes empresas, ou até mesmo de subsidiárias da própria empresa.

No X-CORE, cada artefato possui um conjunto de permissões de visibilidade, identificando zonas ou domínios cujos desenvolvedores podem ou não recuperar o artefato. Nas permissões de visibilidade, com as primitivas *allow* e *deny*, é possível permitir ou proibir de forma hierárquica e recursiva o acesso de desenvolvedores das zonas ou domínios especificados.

A Figura 3 ilustra a configuração das permissões de visibilidade para o artefato *br.empresaB.projetoA.artefatoA*. Neste exemplo, duas empresas brasileiras (*empresaA* e *empresaB*) estão cadastradas. A empresa *A* possui três subsidiárias, representadas pelas subzonas *subA*, *subB* e *subC*. Já a empresa *B*, embora não possua subsidiárias, ela possui um projeto, representando pelo domínio *projetoA*, que por sua vez possui o artefato *artefatoA*. Como ilustrado, a primeira permissão indica que o artefato pode ser acessado pelos desenvolvedores da empresa *A* e de todas as suas subsidiárias, pois a aplicação da permissão é realizada de forma recursiva. Entretanto, as duas permissões seguintes proíbem o acesso dos desenvolvedores das subsidiárias *B* e *C*, respectivamente. Portanto, as permissões de visibilidade permitem que o artefato possa ser recuperado por desenvolvedores da empresa *A* e da sua subsidiária *subA*.

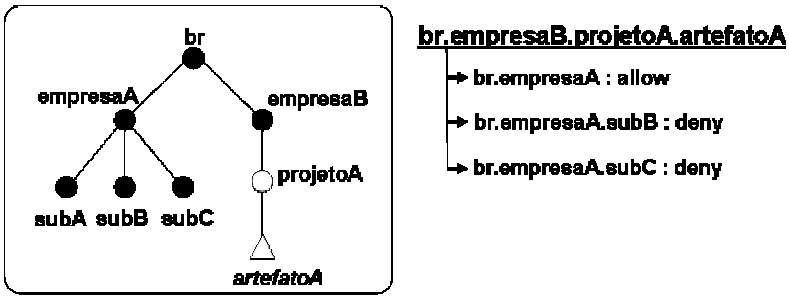


Figura 3. Esquema de visibilidade

É importante ressaltar que apenas os desenvolvedores da empresa responsável pelos artefatos é que podem alterá-los, ou seja, a empresa externa é autorizada apenas a recuperar determinado artefato. Além disso, as permissões de visibilidade somente podem ser modificadas pelos desenvolvedores da própria zona a qual o artefato pertence. Caso não possua visibilidade definida, o artefato pode ser recuperado por qualquer desenvolvedor de outras zonas. Embora seja permissiva, esta regra foi adotada para maximizar o reuso, simplificando a inserção de artefatos em projetos que adotam o modelo *open source.*

4. Configurando Relacionamentos DDS no X-CORE

Como descrito anteriormente, no X-CORE, o esquema de nomeação e o mecanismo de visibilidade externa permitem configurar os diferentes tipos de relacionamentos entre empresas clientes e fornecedores, envolvidas em projetos de desenvolvimento

distribuído de software. Desta forma, para evidenciar o potencial do X-CORE de suportar os diferentes tipos de relacionamentos DDS, a seguir serão apresentados exemplos de configuração de relacionamentos entre clientes e fornecedores.

Outsourcing Tradicional. No exemplo da Figura 4a, as empresas *eA* e *eB* definem um relacionamento de *outsourcing* tradicional, no qual *eA* é a empresa cliente e *eB* a empresa fornecedora. Neste caso, todos os artefatos do projeto *p* produzidos pelos desenvolvedores da empresa *eB* devem ser disponibilizados para os desenvolvedores da empresa *eA*. Para configurar este relacionamento no X-CORE, todos os artefatos gerados no projeto *p* devem possuir uma permissão *allow* para a zona *br.eA*.

Multisourcing. Na Figura 4b, as empresas *eA*, *eB* e *eC* definem um relacionamento de *multisourcing*, no qual *eA* é a empresa cliente, enquanto *eB* e *eC* são as empresas fornecedoras. Neste caso, os artefatos do projeto *p* produzidos pelos desenvolvedores das empresas *eB* e *eC* devem ser disponibilizados aos desenvolvedores da empresa *eA*. No X-CORE, este relacionamento é configurado definindo a permissão *allow* para a zona *br.eA*, tanto nos artefatos do projeto *br.eB.p* quanto nos artefatos do projeto *br.eC.p.* Como a empresa cliente é responsável pela integração dos artefatos, as empresas fornecedoras não precisam compartilhar artefatos entre si.

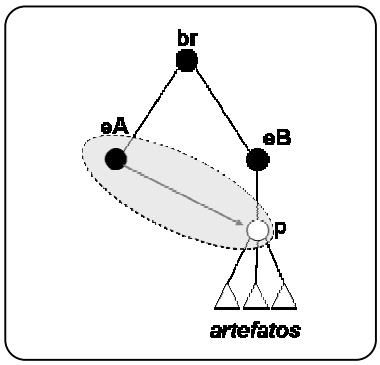


Figura 4a. Oursourcing

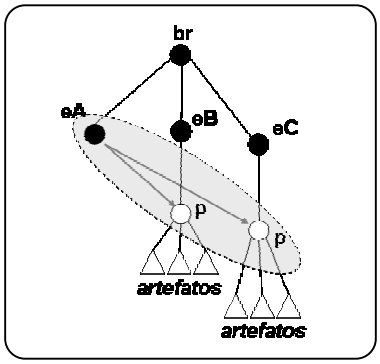
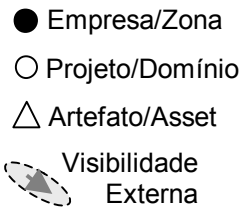


Figura 4b. Multisourcing



Co-Sourcing. No exemplo da Figura 5a, a empresa cliente *eA* contrata as empresas fornecedoras *eB* e *eC* para desenvolverem componentes do projeto *p* de forma cooperada, definindo assim um relacionamento de *co-sourcing*. No entanto, neste caso, o contrato especifica que a empresa *eB* é responsável pelo desenvolvimento dos componentes, enquanto a empresa *eC* é responsável pelos testes. Desta forma, os desenvolvedores da empresa *eC* precisam ter acesso aos artefatos gerados pela equipe da empresa *eB*. Para tal, todos os artefatos gerados pelos desenvolvedores da empresa *eB* no projeto *p* devem possuir uma permissão *allow* para a zona *br.eC.* Além disso, em função do contrato de *sourcing*, todos os artefatos do projeto *p* produzidos pelos desenvolvedores das empresas fornecedoras contratadas (*eB* e *eC*) devem ser acessíveis aos desenvolvedores da empresa cliente (*eA*). Assim, todos os artefatos gerados no projeto *p* pelas empresas *eB* e *eC* devem possuir a permissão *allow* para a zona *br.eA*.

Em uma possível variação desse cenário, mostrada na Figura 5b, as empresas fornecedoras definem um relacionamento interdependente, no qual cada empresa realiza o desenvolvimento de uma parte dos componentes e os testes da outra parte. Neste caso, as empresas *eB* e *eC* precisam acessar os artefatos produzidos pela outra. Para tal, os artefatos gerados pelos desenvolvedores da empresa *eB* devem possuir uma permissão *allow* para a zona *br.eC*, enquanto os artefatos gerados pelos desenvolvedores da empresa *eC* também devem possuir a permissão *allow* para a zona *br.eB*. Em relação à empresa cliente *eA*, a configuração das permissões não precisa ser modificada.

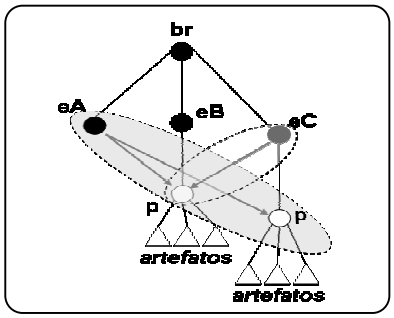


Figura 5a. Co-sourcing Simples

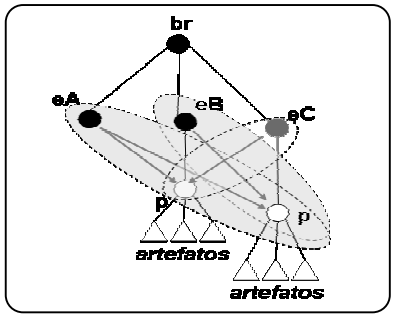


Figura 5b. Co-souring Interdependente

Aliança. A Figura 6a ilustra uma aliança entre as empresas fornecedoras *eB, eC* e *eD* para atender a empresa cliente *eA*. Neste caso, a empresa *eB* é a responsável por representar a aliança, sendo portanto também responsável pela integração dos artefatos e componentes gerados. Desta forma, os desenvolvedores da empresa *eB* devem ter acesso aos artefatos produzidos pelos desenvolvedores das empresas *eC* e *eD* no projeto *p*. Assim, os artefatos produzidos pelas empresas *eC* e *eD* devem possuir a permissão *allow* para a zona *br.eB*. Por outro lado, considerando que a empresa *eB* é a responsável pela integração dos artefatos, os desenvolvedores da empresa cliente *eA* precisam ter acesso apenas aos artefatos de empresa *eB*. Para tal, os artefatos integrados pela empresa *eB* devem possuir a permissão *allow* para a zona *br.eA*.

Joint Venture. Na Figura 6b, as empresas *eA* e e*B* estabelecem um relacionamento de *joint venture* para formar uma terceira entidade *eC* responsável pelo desenvolvimento do projeto *p*. Neste caso, ao invés de explorar o mecanismo de visibilidade, o relacionamento pode ser estabelecido apenas usando a estrutura hierárquica definida pelo esquema de nomeação. Para tal, uma zona (*br.eC*) pode ser criada no espaço de nomes para representar a nova entidade *eC*, e, em seguida, os desenvolvedores das empresas *eA* e *eB* que foram selecionados para atuar no projeto *p* são também registrados na zona *br.eC*.

Insourcing. No exemplo da Figura 6c, o cenário considera que a empresa *eA* repassa o projeto *p* para uma de suas subsidiárias (*seA*), por exemplo, em função da existência de especialistas no domínio do problema. Neste caso, para representar o *insourcing*, é preciso explorar a estrutura hierárquica do esquema de nomeação e as permissões de acesso do mecanismo de visibilidade. Para tal, um domínio é criado na zona da subsidiária (*eA.seA*) para representar o projeto *p*. E, em seguida, para ficar em conformidade com um relacionamento de *insourcing*, todos os artefatos do projeto *p* produzidos pelos desenvolvedores da subsidiária *seA* devem ser disponibilizados para os desenvolvedores da empresa *eA*. Para configurar este relacionamento, todos os artefatos gerados no projeto *p* devem possuir uma permissão *allow* para a zona *br.eA*.

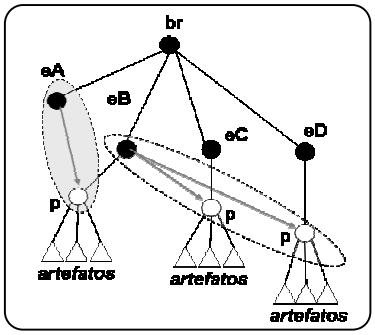


Figura 6a. Aliança

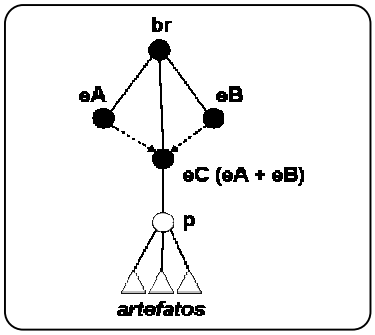


Figura 6b. Joint Ventrure

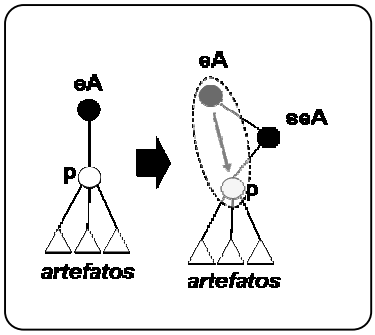


Figura 6c. Insourcing

4. Considerações Finais

A evolução da Internet e o avanço da infra-estrutura de comunicação global vem facilitando a formação de empresas geograficamente dispersas, como também o estabelecimento de contratos e relacionamentos de *sourcing* entre diferentes empresas localizadas em diversas parte do mundo [Hyder 2006]. Neste contexto, a principal contribuição deste artigo é evidenciar o potencial e a flexibilidade provida pelo serviço de repositório X-CORE para representar e operacionalizar os principais tipos de relacionamentos de *sourcing* que podem ser estabelecidos entre diferentes equipes distribuídas, a saber: *outsourcing tradicional*, *co-sourcing*, *multisourcing*, *aliança*, *joint ventrue* e *insourcing*.

No entanto, é importante destacar que os relacionamentos de *sourcing* apresentados e configurados no X-CORE não são os únicos possíveis, mas caracterizam os principais cenários identificados na literatura e indústria de software. Em função do avanço na adoção de abordagens de DDS, certamente, novos tipos de relacionamentos de *sourcing* ou até mesmo variações dos tipos apresentados passarão a existir. Desta forma, a medida que novos tipos de relacionamentos forem identificados, os mecanismos providos pelo X-CORE serão gradativamente aplicados para proporcionar um melhor grau de validação da proposta.

Considerando a natureza compartilhada e distribuída do X-CORE, os cenários avaliados sinalizam a viabilidade da adoção desta infra-estrutura como plataforma para integração de abordagens de DDS e DBC. No entanto, em função dos estudos de caso conduzidos com o X-CORE [Oliveira 2008], identificamos alguns pontos que precisam ser melhorados. Por exemplo, para simplificar a configuração dos relacionamentos DDS, observamos a necessidade de aumentar a granularidade da configuração da visibilidade externa. Originalmente, a visibilidade foi concebida para ser configurada por artefato. Mas, após os estudos de caso desenvolvidos, percebe-se que o ideal seria declarar a visibilidade por projeto, simplificando a inserção dos artefatos pelos desenvolvedores. Entretanto, a configuração por artefato ainda seria adotada, possibilitando o controle de acesso diferenciado a determinados artefatos específicos, caso desejado.

Considerando a crescente importância do desenvolvimento global de software, modelos robustos, processos, métodos e ferramentas são necessárias para organizar eficientemente a realização das tarefas [Damian 2006]. Neste contexto, considerando que a automação do processo e o suporte a negociação pode melhorar a colaboração entre as empresas e reduzir o tempo necessário para fechar contratos (*time-to-contract)* [Laurikkala 2002], como evolução da plataforma X-CORE, estão sendo concebidas ferramentas que auxiliam a criação de contratos eletrônicos (*e-contracting)* para casos de *outsourcing*, possibilitando a configuração automática dos relacionamentos de DDS diretamente no X-CORE. Tais ferramentas adotam técnicas de assinatura digital na descrição de contratos baseados em XML, representando assim as responsabilidades e as obrigações dos diversos atores participantes.

5. Referências

Bass, L.; Buhman, C.; Comella-Dorda, S.; Long, F.; Robert, J.; Seacord, R.; Wallnau, K. (2001). "Volume I: Market Assessment of Component-Based Software Engineering", *Technical Note* CMU/SEI-2001-TN-007, Carnegie Mellon Software Engineering Institute (SEI).

Damian, D; Moitra, D. (2006) "Global Software Development: How Far Have We Come", IEEE Software, Volume 23, Issue 5, September/October.

Gallivan, M. J. (1999). "Analyzing IT Outsourcing Relationships as Alliances among Multiple Clients and Vendors". Proceedings of the 32nd Hawaii International Conference on System Sciences, IEEE, Hawaii, EUA.

Herbsleb, J.; Moitra, D. (2001) "Global Software Development". IEEE Software, Volume 18, Issue 2, March.

Hyder, E. B.; Heston, K. M.; Paulk, M. C. (2006). "The eSourcing Capability Model for Service Providers (eSCM-SP)", v. 2.01. Carnegie Mellon University.

Kotlarsky, J. (2005). "Management of Globally Distributed Component-Based Software Development Projects", Information and Decision Sciences, Rotterdam School of Management Erasmus, Rotterdam, The Netherlands.

Laurikkala, H.; Tanskanen, K. (2002). "Managing Contracts In Virtual Projects Supply Chains", 3rd IFIP Working Conference on Infrastructures for Virtual Enterprises, in Collaborative Business Ecosystems and Virtual Enterprises. Ed. L.M. Camarinha-Matos, Kluwer Academic Publishers.

Oliveira, J. P. F.; Brito, T.; Rabelo, S. Jr.; 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, Paraíba, Brasil.

Oliveira, J. P. F.; Elias, G. (2008). "Um Relato de Experiência com uma Infra-Estrutura para Desenvolvimento Distribuído Baseado em Componentes", II Workshop em Desenvolvimento Distribuído de Software. Campinas - SP, Brasil.

Prikladnicki, R. (2006). "Uma Análise Comparativa de práticas de Desenvolvimento Distribuído de Software no Brasil e no exterior". 20º Simpósio Brasileiro de Engenharia de Software. Florianópolis, Santa Catarina, Brasil.

Radkevitch, U.L.; Heck, H. W. G. M.; Koppius, O.R., (2006). "Leveraging Offshore IT Outsourcing by SMEs through Online Marketplaces", Research Paper ERS-2006-046-LIS Revision, Erasmus Research Institute of Management (ERIM). Rotterdam, The Netherlands.

Repenning, A.; Ioannidou, A.; Payton, M.; Ye, W.; Roschelle, J. (2001). "Using Components for Rapid Distributed Software Development". IEEE Software, Vol. 18. Issue 2, March/April.

Robbison, M.; Kalakota, R. (2004). "Offshore Outsourcing: Business Models, ROI and Best Practices". Mivar Press.

Szyperski, C. (2000). "Component Software: Beyond Object-Oriented Programming", ACM Press/Addison-Wesley Publishing Co.