Uma Extensão do Framework ComponentForge para Desenvolvimento Distribuído de Software

João P. F. de Oliveira, Talles Brito, Adriana Oliveira, Sebastião Jr, Glêdson Elias

Departamento de Informática
Universidade Federal da Paraíba (UFPB) – João Pessoa, PB – Brasil
{joaopaulo, talles, drill, sebastiao, gledson}@compose.ufpb.br

Abstract. *Component-Based Development (CBD) and Distributed Software Development (DSD) approaches have been widely adopted and discussed. However, both have limitations related to coordination, cooperation and communication of geographically dispersed teams. In such a context, this paper presents an extension of the ComponentForge framework, which currently provides support to CBD but has limitations related to DSD since does not adopt mechanisms for distributed management of software projects. By integrating DBC and DSD, the proposed extension minimizes issues related to coordination, cooperation and communication of distributed teams.*

Resumo. *Abordagens de Desenvolvimento Baseado em Componentes (DBC) e Desenvolvimento Distribuído de Software (DDS) estão sendo amplamente adotadas e discutidas. Entretanto, ambas apresentam limitações relacionadas à coordenação, cooperação e comunicação de equipes distribuídas. Neste contexto, este artigo apresenta uma extensão do framework ComponentForge, que atualmente provê suporte ao DBC, mas que, entretanto, apresenta limitações relacionadas ao DDS, pois ainda não adota mecanismos de gerenciamento distribuído de projetos. Alinhando DBC e DDS, a extensão proposta tem o potencial de minimizar os problemas relacionados à coordenação, cooperação e comunicação de equipes distribuídas.*

1. Introdução

Nos últimos anos, com o objetivo de reduzir custos e aprimorar a utilização de recursos, organizações especializadas no desenvolvimento de software começaram a adotar abordagens de desenvolvimento distribuído de software (DDS). Essa tendência é favorecida pela necessidade de financiar e utilizar recursos, onde quer que estejam localizados, bem como, pela possibilidade da ágil formação de equipes de desenvolvimento distribuídas, a fim de explorar oportunidades de mercado [1].

Neste contexto, podemos relacionar o Desenvolvimento Baseado em Componentes (DBC) com o DDS. Esta relação é explicitada na forma de um mercado de software, no qual produtores e consumidores, dispersos geograficamente, negociam a aquisição de diversos componentes de software [2].

Entretanto, a separação física ocasiona diversos problemas em vários aspectos. Um dos problemas é coordenar quando, como, onde e por quem um produto será desenvolvido, pois, quando o mesmo deve ser desenvolvido entre diferentes grupos, é necessário ter uma definição precisa de quais são as atividades que devem ser realizadas e quais ainda não foram realizadas. Isto requer um entendimento uniforme do processo

de desenvolvimento adotado [3], bem como, a existência de mecanismos de gerência que forneçam suporte adequado ao desenvolvimento distribuído [4].

Outro problema é inerente à comunicação, ocasionado pelas diferenças de fuso-horário entre diferentes localizações, onde existe uma grande dificuldade de iniciar contatos, bem como, avaliar a disponibilidade de desenvolvedores. Tais problemas de comunicação conduzem para um longo e cíclico período para resolver questões dos projetos, devido à dificuldade de encontrar desenvolvedores que tenham respostas para tais questões, ou, possam indicar outras pessoas para solucioná-las [3].

Neste sentido, ao lidar com o DDS, faz-se necessário um mecanismo estruturado que trate os aspectos temporais, organizacionais e físicos da distribuição e seu impacto no desempenho de projetos distribuídos [5]. Neste contexto, a contribuição deste artigo está na proposta de uma extensão do *ComponentForge* [6], um *framework* arquitetural que atualmente provê suporte a processos de DBC, oferecendo facilidades para armazenar, buscar, recuperar, certificar e negociar artefatos de software, mas que, entretanto, apresenta limitações quanto ao DDS, pois ainda não provê mecanismos que favoreçam o gerenciamento distribuído do projeto como um todo. Esta extensão tem como objetivo oferecer uma visão interdisciplinar e alinhada de DBC e DDS, além de superar os desafios de definir, coordenar e controlar as atividades de equipes dispersas geograficamente, bem como facilitar a comunicação dessas equipes.

O restante deste trabalho está organizado da seguinte forma. A Seção 2 introduz o *framework* arquitetural ComponentForge, que oferece um infra-estrutura de suporte a processos de desenvolvimento baseado em componentes. Em seguida, a Seção 3 apresenta uma extensão para o ComponentForge, que, tendo apoio dessa infra-estrutura, possibilitará a coordenação, colaboração e comunicação em projetos distribuídos de software. Já a Seção 4 apresenta uma breve comparação com alguns trabalhos relacionados. Por fim, a Seção 5 apresenta algumas considerações finais.

2. O ComponentForge

O reuso de componentes apresenta alguns fatores inibidores, tais como a carência de componentes, padrões, certificação e métodos para a construção de componentes [7], além da atual dificuldade em identificar, selecionar, negociar e recuperar componentes que atendam aos requisitos especificados [8]. Considerando as limitações expostas, o grupo COMPOSE propôs o ComponentForge [6], um *framework* arquitetural que provê um conjunto de serviços distribuídos, compartilhados, independentes e fracamente acoplados que se comunicam e colaboram entre si através de interfaces e protocolos de comunicação bem definidos, com o objetivo de prover suporte a abordagens de DBC.

O ComponentForge é composto por um conjunto de ferramentas e mais quatro serviços (serviço de busca, serviço de negociação, serviço de certificação e serviço de repositório). No *framework,* o conjunto de ferramentas permite a interação dos usuários dos serviços com os demais elementos do ComponentForge. Além disso, o *framework* foi concebido de forma a permitir a coexistência de diversos *serviços de certificação* que são responsáveis por certificar não somente a qualidade dos artefatos de software, mas também as práticas e os processos adotados pelos produtores. Já o *serviço de negociação* tem a função de prover mecanismos para assegurar que um artefato será adquirido e entregue em conformidade com os modelos de negócios especificados por seu produtor. Para permitir a busca e a recuperação de artefatos armazenados, o ComponentForge define o *serviço de busca*, que consulta o *serviço de repositório* para realizar a indexação dos artefatos e dos seus metadados.

Por fim, para prover a infra-estrutura adotada pelos outros serviços do *framework*, o serviço de repositório atua como um *middleware* de propósito especial, possibilitando o armazenamento, localização, recuperação e gerenciamento de artefatos de software. Vale ressaltar que o serviço de repositório é suportado por uma coleção de entidades cooperantes e distribuídas, denominadas *containers*, que, conjuntamente, provêem facilidades às demais entidades do ComponentForge. Além disso, o serviço de repositório inclui facilidades de controle de versão, gerência de métricas de reuso, como também aspectos de segurança relacionados à autenticação e controle de acesso. A seguir, as facilidades providas pelo serviço de repositório serão descritas, pois as mesmas também são de fundamental importância em um ambiente de DDS.

Esquema de Nomeação. Considerando a natureza compartilhada e distribuída do serviço de repositório, este adota um esquema de nomeação que facilita a organização dos artefatos armazenados e permite a localização e recuperação não ambígua dos mesmos. Neste esquema, os nomes são organizados em uma árvore hierárquica onde as folhas são os artefatos armazenados. Por sua vez, os nós internos correspondem a dois tipos de entidades (*zonas* e *domínios*). Zonas representam produtores de artefatos e suas famílias de produtos, e, domínios são subdivisões das zonas que possibilitam uma melhor organização dos artefatos, cuja função é agrupar um conjunto de artefatos.

Controle de Acesso. O serviço de repositório permite a existência de diferentes usuários que desempenham diferentes papéis e executam tarefas distintas. Para tal, o controle de acesso adota um mecanismo semelhante ao modelo RBAC (*Role-Based Access Control*) [9], em que as operações permitidas são associadas aos papéis, que, por sua vez, são atribuídos aos usuários.

Visibilidade Externa. Para facilitar o compartilhamento de artefatos e controlar o acesso de diferentes equipes de desenvolvedores, o serviço de repositório explora o conceito de visibilidade. Sendo assim, para cada artefato registrado em uma zona pode ser definido um esquema de visibilidade indicando quais desenvolvedores de uma ou várias outras zonas são autorizados ou proibidos de recuperar o artefato.

Controle de Versões. O serviço de repositório oferece um mecanismo que controla as diversas versões de um mesmo artefato, estabelecendo o relacionamento entre as mesmas e o controle das modificações de cada versão. Isto permite que os desenvolvedores pertencentes a um mesmo grupo de trabalho e distribuídos geograficamente participem da produção de um dado artefato.

Informações de Reuso. Com a finalidade de facilitar a tomada de decisão dos consumidores sobre a adoção e uso de um dado artefato, o serviço de repositório gerencia um conjunto de *informações de reuso*. Estas informações identificam o grau de satisfação e comentários dos consumidores que adquiriram o artefato, bem como, as respostas dos desenvolvedores a estes comentários.

Metadados. O serviço de repositório explora um modelo de representação de componentes, denominado X-ARM [10]. Este modelo inclui informações que podem ser exploradas pelos consumidores na identificação, aquisição, instalação e uso de artefatos. Além disso, o X-ARM também representa informações sobre os modelos de certificação e de negócio adotados pelos produtores dos artefatos, viabilizando as funcionalidades providas pelos serviços de certificação, negociação e busca.

3. Uma Extensão do ComponentForge para o DDS

Segundo Cheesman [11], todo projeto de software é composto por dois processos distintos, um processo gerencial e um processo de desenvolvimento. O processo gerencial programa as atividades, planeja entregas, aloca recursos e monitora o

andamento. O processo de desenvolvimento trata aspectos ligados a métodos, técnicas e práticas que empregam pessoas para o desenvolvimento e manutenção de um software e seus artefatos associados (planos, documentos, modelos, código e casos de testes).

Neste contexto, este artigo propõe uma extensão do ComponentForge, que, atualmente, provê um conjunto de serviços que dão suporte ao processo de desenvolvimento, oferecendo facilidades para armazenar, buscar, recuperar, certificar e negociar variados artefatos de software. Com a extensão aqui proposta, o *framework* passa a prover facilidades para a gestão de projetos distribuídos de software, oferecendo mecanismos para a coordenação, colaboração e comunicação. Neste sentido, o *serviço de gerência* é acrescentado ao ComponentForge, como ilustrado na Figura 1.

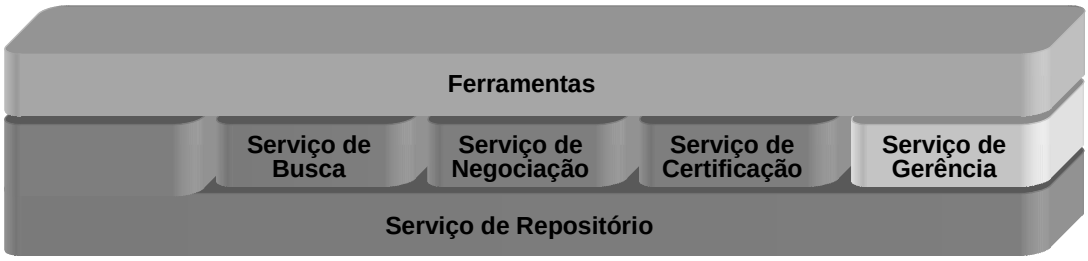

Figura 1. Extensão do Framework ComponentForge

O serviço de gerência, assim como o ComponentForge, adota uma arquitetura baseada em serviços, sendo o mesmo formado por três serviços, como ilustra a Figura 2. O *serviço de gestão* permite criar e coordenar equipes virtuais, possibilitando verificar quais atividades foram definidas e quais os resultados obtidos pelas mesmas no decorrer de um projeto. Durante o desenvolvimento de um projeto de software, problemas podem ser encontrados, logo precisam ser relatados para que soluções possam ser definidas. Neste contexto, o *serviço de manutenção* tem por finalidade documentar esses erros, de forma que possam ser mantidos históricos de informações sobre esses problemas. Por fim, o *serviço de comunicação* permite que um dado desenvolvedor ou uma equipe possa ser contatada de acordo com as possibilidades disponíveis de comunicação. Nas subseções seguintes estes serviços serão descritos.

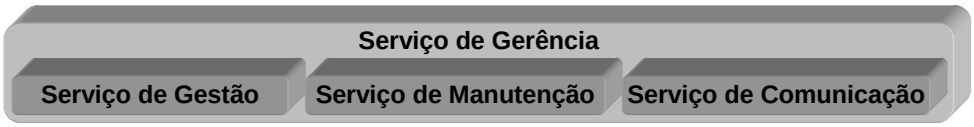

Figura 2. Composição do serviço de gerência

3.1 Serviço de Gestão

O serviço de gestão permite a criação de organizações virtuais compostas por um conjunto de produtores distribuídos. Além disso, o serviço de gestão provê funcionalidades para a coordenação dessas equipes, bem como a definição das atividades que devem ser realizadas pelas mesmas.

Formação de Organizações Virtuais. O DDS é caracterizado pela formação de *Organizações Virtuais* (OVs), que são compostas por equipes virtuais, isto é, equipes dispersas geograficamente. O objetivo da OV é desenvolver um sistema de software com base nos requisitos do cliente. Uma OV pode ser formada por uma ou por um conjunto de empresas ou desenvolvedores que cooperam enquanto durar o projeto [12].

Neste contexto, para o ComponentForge, uma OV é a empresa ou equipe que é responsável pelo projeto, bem como, pela seleção e coordenação das equipes virtuais. Assim, o serviço de gestão possibilita que esta empresa constitua uma OV. Além disso,

a OV, ao interagir com o serviço de gestão, poderá desenvolver e gerenciar o projeto como um todo. Vale ressaltar que estas funcionalidades são suportadas pela infra-estrutura colaborativa provida pelo serviço de repositório.

Para que uma OV possa coordenar seus projetos e as equipes possam executar suas atividades, tanto a OV quanto as equipes deverão estar registradas em zonas no serviço de repositório. Assim, no registro de um novo projeto, são definidos os usuários que desempenharão o papel de *gerente do projeto*, bem como, são configurados domínios, na zona que representa a OV, nos quais serão armazenados os artefatos gerados durante o desenvolvimento do projeto.

A Figura 3 ilustra um cenário em que equipes estão dispersas geograficamente e como seus relacionamentos são mapeados no serviço de repositório na constituição de uma OV. Neste cenário, a *empresa B*, por meio do serviço de gestão, resolve criar uma OV e registrá-la no serviço de repositório na zona *br.empresaB*. Desta forma, a empresa B poderá formar uma parceria com a *empresa A* (*com.empresaA*) e com a *empresa C* (*br.empresaC*) para desenvolver o projeto, aqui definido como Projeto *Alpha*.

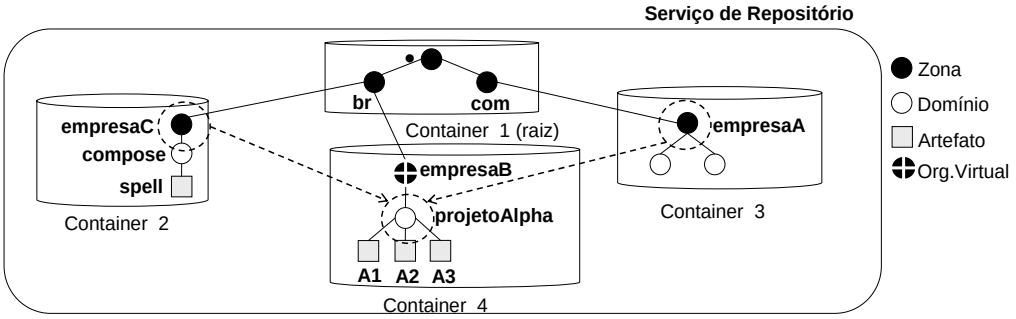

Figura 3. Formação de uma Organização Virtual no serviço de repositório

Neste caso, quando registrado o projeto *Alpha*, o serviço de gestão deve criar um domínio (*br.empresaB.projetoAlpha*) no serviço de repositório, no qual os desenvolvedores da empresa *A* e empresa *C* poderão interagir com a empresa *B* para o desenvolvimento do projeto. Neste cenário, ao final do projeto, como pode ser observado na Figura 3, um conjunto de artefatos (*A1*, *A2* e *A3*) é gerado como resultado das atividades dos desenvolvedores das zonas remotas *com.empresaA* e *br.empresaC*.

Gestão das Atividades. O gerente de projeto deve definir quais atividades serão realizadas e quais artefatos devem ser gerados. Para tal, o serviço de gestão adota o conceito de *workflow*, que é a automação de um processo de negócio, por inteiro ou por partes, durante o qual documentos, informações e atividades são passadas de um participante para outro, para que estes desenvolvam ações respeitando um conjunto de regras procedimentais, visando um objetivo comum [13].

As atividades podem ser executadas em seqüência ou simultaneamente, por diferentes indivíduos. Nesse sentido, o serviço de gestão permite ao gerente do projeto definir e controlar completamente os *workflows*. Logo, uma vez que um processo para o desenvolvimento é definido, o serviço de gestão pode certificar que as atividades ocorrem na seqüência própria e que os usuários são informados para que possam executar suas tarefas. Desta forma, os usuários podem ser coordenados remotamente, e, também, é permitido aos mesmos identificar quais artefatos servem de entrada para as suas atividades e quais artefatos devem ser gerados ao fim de suas atividades.

Como mencionado na Seção 2, o serviço de repositório adota o X-ARM como modelo de representação. Além de descrever as funcionalidades dos artefatos, o X-ARM também representa informações sobre o processo de desenvolvimento adotado.

Assim, definido um processo e seus *workflows*, o serviço de gestão permite que o mesmo seja representado em forma de um artefato que poderá ser reutilizado em outros projetos. Desta maneira, para um novo projeto, o serviço de gestão somente precisa que o gerente de projeto informe o artefato que representa o processo, para que automaticamente sejam configuradas as atividades a serem desenvolvidas.

Vale ressaltar que o serviço de repositório provê uma infra-estrutura para que o serviço de gestão configure esquemas de visibilidade para os artefatos de entrada das atividades, bem como, possibilita o armazenamento dos artefatos de saída produzidos nos *workflows*. Além disso, em uma OV, é bastante comum a existência de diversas equipes virtuais trabalhando em conjunto, dividindo as atividades de desenvolvimento e compartilhando os artefatos produzidos. Logo, os mecanismos de visibilidade e controle de versões de artefatos do serviço de repositório auxilia na gerência dessa produção.

Gestão das Equipes. É responsabilidade do gerente do projeto a coordenação das equipes através do serviço de gestão. Para tal, o serviço de gestão adota o modelo *RBAC* de controle de acesso. Porém, diferentemente do serviço de repositório, que associa aos papéis as operações permitidas aos usuários, o serviço de gestão associa aos papéis as atividades que as equipes podem executar dentro do projeto. Desta maneira, caso uma equipe não faça mais parte do projeto, o gerente do projeto pode associar as atividades para outra equipe. Vale ressaltar que o modelo de controle de acesso no serviço de gestão é mapeado para o modelo de controle de acesso no serviço de repositório, pois, para cada atividade, o serviço de gestão deve configurar no serviço de repositório as permissões necessárias à realização da mesma. Além disto, ao utilizar os mecanismos de autenticação e controle de acesso do serviço de repositório, o serviço de gestão pode garantir que uma atividade somente será realizada pelo desenvolvedor designado.

3.2 Serviço de Manutenção

Repositórios de *bugs*, também conhecidos como sistemas de *bug tracking*, provêem um banco de dados de relatórios sobre os problemas encontrados durante o desenvolvimento de um projeto de software. Segundo Singer [14], o repositório de *bugs* serve como um importante depósito de informações, pois como ele é geralmente mantido atualizado pelos desenvolvedores, acaba representando o estado atual do sistema de forma mais exata do que a documentação, que é suscetível a ser menos consistente. Além disso, com o sistema de *bug tracking* é possível que decisões de projeto possam ser tomadas com base na análise das informações relatadas pelos desenvolvedores e pelo acompanhamento da evolução dos projetos.

Neste contexto, o serviço de manutenção provê meios para que as requisições sobre problemas relatados no projeto possam ser registradas, tanto pelos gerentes das OV, quanto pelos desenvolvedores. Além disto, possibilita ao gerente de projeto decidir por quem e quando tais requisições devem ser tratadas, bem como, identificar e descartar requisições duplicadas ou irrelevantes. É importante destacar que a utilização do serviço de manutenção potencializa a qualidade no desenvolvimento de software, pois permite que *bugs* sejam identificados e corrigidos mais rapidamente.

3.3 Serviço de Comunicação

A prática de reutilização ainda é muito dependente de comunicação entre as equipes de desenvolvimento. A documentação é suficiente para descrever o conhecimento de experiências, mas a comunicação direta é necessária para resolver diversos problemas e estabelecer relações de confiança entre os desenvolvedores [14]. Além disso, existem vários problemas relacionados a questões de comunicação com equipes distribuídas. Um deles diz respeito a como saber da disponibilidade de outros desenvolvedores. [3].

Neste contexto, o serviço de comunicação permite identificar a disponibilidade de um dado usuário, e, além disso, indicar o melhor meio de comunicação disponível. Por exemplo, caso um gerente precise entrar em contato com um desenvolvedor e o mesmo esteja *on-line*, o serviço de comunicação indica uma ferramenta de comunicação síncrona (*chat*). Caso contrário, indica uma ferramenta assíncrona (fóruns, *e-mail*). Além disso, o serviço de comunicação também permite o compartilhamento de agendas, de forma que possam ser marcadas reuniões virtuais entre desenvolvedores distribuídos.

4. Trabalhos Relacionados

Esta seção apresenta uma breve comparação das funcionalidades providas pelo *framework* proposto neste artigo com os principais repositórios disponíveis, que provêem suporte ao DDS, tais como o *OSCAR* [15] e o *SourceForge* [16].

Distribuição e escalabilidade são requisitos fundamentais para promover a cooperação entre equipes distribuídas. Desta forma, além do *framework* proposto, somente o *OSCAR* provê uma arquitetura distribuída, porém restrito a organização que o utilizar. Ou seja, o OSCAR não permite que organizações distintas compartilhem artefatos. Por outro lado, apesar do *SourceForge* provê suporte para equipes distribuídas, sua arquitetura é centralizada.

Além disso, o *OSCAR* e o *SourceForge* não adotam os conceitos de Organização Virtual ou de equipes virtuais, ou seja, não permitem a formação de equipes geograficamente dispersas. No máximo, permitem que usuários remotos acessem seus repositórios. Assim, para controlar os usuários e as entidades autorizadas a recuperar e manipular artefatos, o *OSCAR* adota um simples esquema de *login* e senha, além de histórico (*log*), que armazena quem acessou, qual artefato foi acessado e qual foi o propósito do acesso. Já o *framework* proposto, assim como o *SourceForge*, adotam o modelo de controle de acesso denominado RBAC (*Role-Based Access Control*), no qual as entidades são classificadas em diferentes papéis e diferentes permissões de acesso são associadas a cada papel desempenhado pelos diferentes grupos.

Planos e processos são mecanismos vitais de coordenação em projetos de software. Entretanto, tratar eventos não antecipados requer mecanismos flexíveis de comunicação. Neste contexto, o *framework* proposto, por meio do *serviço de comunicação*, possibilita identificar a disponibilidade dos desenvolvedores e selecionar o melhor mecanismo de comunicação. Já o *SourceForge* só disponibiliza mecanismos assíncronos, o que pode gerar um certo atraso na comunicação das equipes. Por fim, o *OSCAR* não oferece nenhum apoio para a comunicação entre produtores.

5. Considerações Finais

O Desenvolvimento Distribuído de Software (DDS) e o Desenvolvimento Baseado em Componentes (DBC) destacam-se como abordagens que têm o potencial de tornar realidade a redução da complexidade, do tempo e do custo de desenvolvimento, bem como melhorar a qualidade do software produzido, além da possibilidade de obtenção de recursos em âmbito global. Entretanto, por um lado, abordagens de DBC possuem limitações quanto à certificação da qualidade e a disponibilidade de componentes. Por outro lado, abordagens de DDS ainda precisam vencer desafios relacionados à coordenação, cooperação e comunicação das equipes distribuídas.

Neste contexto, como principal contribuição, este artigo propõe uma extensão do *framework ComponentForge*, de forma que, além apoiar processos de DBC, possa prover suporte a processos de gerenciamento de DDS. Assim, a extensão do *ComponentForge* passa a prover uma infra-estrutura colaborativa que oferece suporte à formação e gestão de múltiplas equipes distribuídas, que, de forma coordenada, podem

desenvolver suas atividades segundo procedimentos pré-definidos e controlados. Além disso, o *ComponentForge* provê não apenas uma ferramenta de comunicação, mas um serviço que faz com que a informação chegue a pessoa certa em tem hábil, sendo de fundamental importância no apoio à cooperação entre equipes distribuídas.

Atualmente, os conceitos e funcionalidades do *framework* estendido já estão com grau de maturidade relativamente satisfatório. No momento, o grupo COMPOSE está iniciando a fase de especificação detalhada das interfaces dos serviços de gestão, manutenção e comunicação. Após a especificação destes serviços, a implementação atual será estendida para contemplar as facilidades de DDS descritas neste artigo.

Referências Bibliográficas

[1] Herbsleb, J.D.; Moitra, D. (2001) “Global software development”. Software, IEEE, Vol.18, Iss.2, Pages:16-20

[2] Wallnau, K. C.; Hissam, S. A.; Seacord, R. C. (2001), “Building Systems from Commercial Components”, SEI Series in Software Engineering, Addison-Wesley

[3] Herbsleb, J.D.; Grinter, R.E. (1999) “Architectures, coordination, and distance: Conway's law and beyond” Software, IEEE, Vol.16, Iss.5, Pages:63-70

[4] Meyer, B. (2006) "The Unspoken Revolution in Software Engineering" *Computer*, vol. 39, no. 1, pp. 124, 121-123.

[5] Damian, D. and Moitra, D. (2006) “Global Software Development: How Far Have We Come?” IEEE Software, Vol.23, p. 17-19.

[6] 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.

[7] 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.

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

[9] Ferraiolo, D. F.; Sandhu, R.; Gavrila, S.; Kuhn, D. R.; Chandramouli, R. (2001) “Proposed NIST Standard for Role-Based Access Control”. ACM Transactions on Information and Systems Security, Vol. 4, No. 3, pp. 224-274.

[10]Schuenck, M. (2006) “X-ARM: Um Modelo de Representação de Artefatos de Software”. Dissertação de Mestrado, DIMAp-UFRN, Natal-RN.

[11]Cheesman, J.; Daniels, J. (2001) “UML Components: A Simple Process for Specifying Component Based Software”. Addison-Wesley.

[12]Cyrillo, L. C. (2005) “GESPRODS – Um Modelo de Gestão de Projetos Distribuídos de Software”. Dissertação de mestrado apresentada à Escola Politécnica da Universidade de São Paulo.

[13]Hollingsworth, D. (1995) “Workflow Management Coalition. The Workflow Reference Model”. Document number TC00-1003.

[14]Singer, J.(1998) “Pratices of Software Maintenance”. In 14th IEEE International Conference on Software Maintenance (ICSM'98). Bethesda, MD, USA

[15]Boldyreff, C.; Nutter, D.; Rank, S. (2002) “Open-Source Artifact Management”. Workshop Open Source Software Engineering, USA.

[16]SourceForge Enterprise Edition (2007).http://www.vasoftware.com/sourceforge.