Uma Proposta de Arquitetura para Ambientes de Desenvolvimento Distribuído de Software

Igor Scaliante Wiese, Igor Steinmacher, Rogério Pozza, Elisa Hatsue Moriya Huzita, Éderson Fernando Amorim, Márcia Cristina Dadalto Pascutti

Departamento de Informática, Universidade Estadual de Maringá (UEM)
Av. Colombo 5790, CEP 87020-900 – Maringá – PR – Brasil

{igorwies, igor, elisa, pozza, ederson}@din.uem.br,
pascutti@cesumar.br

Resumo. *O número de empresas que estão distribuindo seus processos de desenvolvimento de software ao redor do mundo é cada vez mais significativo, visando ganho de produtividade, redução de custos e melhorias na qualidade. Neste sentido são propostos ambientes de desenvolvimento distribuído de software(ADDS). Este trabalho tem como objetivo apresentar uma proposta de arquitetura de um ADDS, o DiSEN. Neste sentido é apresentado um estudo dos ambientes encontrados na literatura. Este estudo culmina com a proposta de uma avaliação qualitativa dos ambientes, por meio de um conjunto de características tidas como desejáveis em ambientes de desenvolvimento em que as equipes se encontram distribuídas. Em seguida é apresentada a arquitetura proposta para o DiSEN e como as características enumeradas são suportadas dentro desta arquitetura.*
Palavras-Chave: *Desenvolvimento Distribuído de Software, Arquitetura, Cooperação, Ambiente de Desenvolvimento.*

WISBD – Workshop de Ingeniería de Software y Bases de Datos

1. Introdução

O número de empresas que estão distribuindo seus processos de desenvolvimento de software ao redor do mundo é cada vez mais significativo, visando ganho de produtividade, redução de custos e melhorias na qualidade (PRIKLADNICKI; AUDY, 2004). Desta forma, tem-se criado um cenário onde projetos de software são desenvolvidos com equipes distribuídas, caracterizando assim o desenvolvimento distribuído de software.

O termo ambiente de desenvolvimento de software (ADS) refere-se a um ambiente que automatiza todas as atividades do ciclo de vida de desenvolvimento de software (DART *et al*., 1987). Ambientes de Desenvolvimento de Software (ADS) buscam combinar técnicas, métodos e ferramentas para apoiar o engenheiro de software na construção de produtos de software, abrangendo todas as atividades inerentes ao processo, tais como de gerência, desenvolvimento e controle da qualidade. Dessa forma, os ADS têm ganhado cada vez mais importância (FALBO, 1998).

Baseando-se nesse contexto, este artigo apresentará uma proposta de arquitetura para ambientes de desenvolvimento distribuído de software (ADDS). Esse ambiente fornece suporte ao desenvolvimento distribuído de software, podendo as equipes estar geograficamente distribuídas e trabalharem de forma cooperativa.

A arquitetura proposta adotará o estilo camadas, sendo constituída por três camadas: dinâmica, aplicação e infra-estrutura. A camada dinâmica será responsável pelo gerenciamento da configuração do ambiente; a camada de aplicação suportará as metodologias de desenvolvimento de software, gerenciamento de *workspace*, gerenciamento de agentes e conterá o repositório para armazenamento dos dados e artefatos necessários ao ambiente e a camada da infra-estrutura proverá suporte às tarefas de nomeação, persistência e concorrência e incorporará o canal de comunicação. Os elementos das camadas de aplicação e da infra-estrutura se comunicarão através do canal de comunicação.

Assim, os objetivos do presente trabalho são: (i) revisão bibliográfica das abordagens e propostas encontradas; (ii) proposta de características para análise de ambientes de desenvolvimento distribuído de software (ADDS); (iii) especificação de uma arquitetura de um

ADDS que incorpora metodologias de desenvolvimento e ofereça suporte ao trabalho cooperativo, permitindo a interação de profissionais.

A tarefa de especificar um ADS é bastante complexa, assim não se espera com esse trabalho definir em detalhes como seria a implementação de todos os elementos que constituem o DiSEN (PASCUTTI, 2002) e nem tão pouco mostrar a sua implementação. Isto será alcançado em trabalhos futuros.

O restante do artigo é apresentado da seguinte maneira. A seção 2 apresenta a revisão bibliográfica e a avaliação dos ambientes encontrados. Na seção 3 é proposta a arquitetura para especificação do DiSEN. Na seção 4 é ilustrada a comunicação utilizando o DiSEN. Finalmente, a seção 5 traz as conclusões e os trabalhos futuros.

2. Ambientes de Desenvolvimento Distribuído de Software

Devido ao crescente número de empresas que utilizam processos de desenvolvimento de software de maneira distribuída, é possível encontrar vários projetos de pesquisa que têm como objetivo a criação de ambientes de desenvolvimento distribuído de software. Nesta seção algumas das pesquisas e ambientes encontrados na literatura são apresentados e detalhados. Ao final é proposta uma maneira de classificar qualitativamente tais propostas.

O GENESIS (Generalized Environment for procESs management in cooperatIve Software engineering) (AVERSANO *et al.*, 2003) é um ambiente distribuído que suporta a engenharia de software cooperativa, que mantém as características de um sistema distribuído e que permite o trabalho em grupo. A sua arquitetura é constituída por diferentes subsistemas integrados por meio de um cliente em comum e uma camada lógica de coordenação. O componente Sistema Gerenciador de Recursos (*Resource Management System*) é o responsável por gerenciar recursos humanos e as respectivas alocações para diversos projetos; o Sistema Gerenciador de Artefatos (*Artefact Management System*) é responsável pela criação, modificação, remoção e armazenamento de artefatos; a Máquina de Eventos (*Event Engine*) notifica outros sítios cooperativos e usuários do GENESIS sobre a produção de um artefato ou término de uma atividade, enquanto o Sistema de Comunicação (*Communication System*) permite a comunicação (síncrona e assíncrona).

O Odyssey Share (WERNER *et al.*, 2003) tem como objetivo explorar os aspectos colaborativos, por meio da construção de um mecanismo para suporte à interação em grupos de trabalho, em um ambiente para desenvolvimento baseado em componentes pré-existentes, denominado Odyssey. Este ambiente possui como principais características o apoio à encenação de processos em equipes distribuídas, incentivo à comunicação e à socialização, apoio à atividade individual e em pequenas equipes. O desenvolvimento de componentes entre grupos de trabalho distribuídos envolve o uso de repositórios de diferentes tipos (formato de armazenamento, tipo de acesso, capacidade de consulta, etc.) e independentes entre si. Por esta razão, foi elaborada uma arquitetura, ComPublish (WERNER et. al., 2003), que visa auxiliar os desenvolvedores desses grupos a publicar e recuperar artefatos de software (modelos, diagramas e códigos) na Internet.

Gossip (BABAK, 2000) é parte de um projeto de pesquisa na área de ambientes de desenvolvimento de software, em que foi proposto um framework para suportar a colaboração em projetos de desenvolvimento de produtos distribuídos. Esse framework consiste em três partes: (1) a camada *Product*, tem a função de prover percepção entre os produtos virtuais distribuídos que são compostos de várias partes, (2) a camada *cluster* implementa mecanismos de colaboração para partes dos produtos de grupos dentro do *cluster*, (3) a camada *application* auxilia na integração e utiliza várias ferramentas de desenvolvimento.

O MILOS (*Minimally Invasive Longterm Organizational Support*) (MAURER; MARTEL, 2002; MAURER *et al.*, 2000) propõe suportar a execução de processos e aprendizagem organizacional para grupos virtuais de desenvolvimento de software. O *Milos* é um esforço colaborativo entre o grupo de processo de software da Universidade de Calgary e o grupo de inteligência artificial da Universidade de Kaiserlautern. O *Milos* tem como característica oferecer suporte para processos ágeis (XP e Scrum), provendo colaboração e coordenação tecnológica para o desenvolvimento de software distribuído.

Analisando as propostas de ambientes apresentados acima pode-se enumerar algumas

características necessárias ou possíveis em ambientes de desenvolvimento distribuído. Em CARMEL (1999) estas características são apresentadas como forças centrípetas para a formação de equipes distribuídas e que podem levá-las ao sucesso. Neste trabalho estas forças foram mapeadas para características desejáveis em ambientes de desenvolvimento distribuído de software e são utilizadas para avaliar as abordagens apresentadas, e, em seguida, para definição da arquitetura aqui proposta. A figura sugerida por CARMEL (1999) é apresentada na Figura 1 e sua atualização para características de sucesso de ambientes de desenvolvimento é apresentada na Figura 1(b).

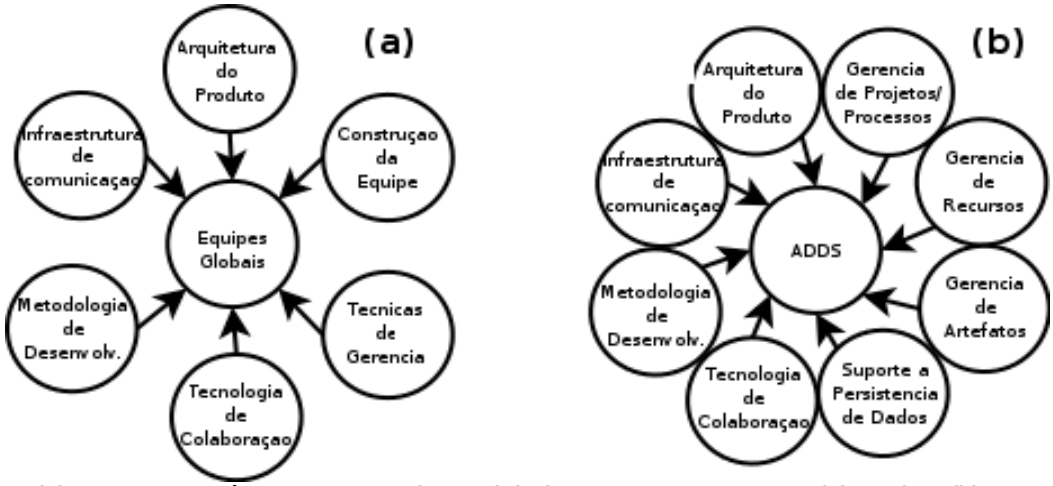

Figura 1. (a) Forças Centrípetas para Equipes Globais propostas por Carmel (1999); e (b) características necessárias a um Ambiente de Desenvolvimento Distribuído de Software (ADDS)

Como dito anteriormente estas características são utilizadas para avaliar qualitativamente as propostas encontradas na literatura. Tal avaliação é apresentada na Tabela 1.

Tabela 1. Avaliação dos Ambientes de Desenvolvimento Distribuído de Software de Acordo com as características enumeradas

Características	***GENESIS***	***Odissey Share***	***Gossip***	***MILOS***
Arquitetura de Produtos	--	Foco no desenvolvimento baseado em componentes	--	--
Suporte a Metodologias de Desenvolvimento	Definidas através de processos de *workflow*	Baseado na engenharia de domínio (Catalysis)	--	Processos ágeis (XP, Scrum)
Suporte a gerência de Recursos	Sistema Gerenciador de Recursos	Máquina de Workflow (Charon)	--	Alocados pelo Sistema de *Workflow*.
Gerência de Artefatos	Sistema Gerenciador de Artefatos	Sistema Gerenciador de Artefatos (GOA++)	Motor de Awareness de Produtos	--
Suporte a Persistência de Dados	Banco de Dados relacional, não existe especificação de tal característica.	Banco de Dados relacional, arquivos serializados (streams)	Motor de *Awareness* de Produtos	OODBMS
Gerência de Projetos/Processos	Máquina de Workflow e Ferramenta de Gerenciamento de Projetos	Máquina de Processos, Máquina de Workflow (Charon), Motor de *Awareness* de Produtos	Motor de *Awareness* de Processos	Workflow para coordenação e ferramentas externas para definir processos e controlar projetos.
Infraestrutura de Comunicação	Sistema de Comunicação - Síncrona/Assíncrona	Sistema de Comunicação - Síncrona/Assíncrona	Sistema de Comunicação - Síncrona/Assíncrona	Infraestrutura sobre o CORBA
Tecnologia de Colaboração	Máquina de Eventos (Avisos de Alterações)	Ájax (mensagens), Teleapontador, radar, espaço de trabalho compartilhado. Serviço de percepção para interações (Ariane)	Motor de *awareness workspace* e motor de *awareness* de participante	Colaboração assíncrona (agenda e avisos de conclusão)

Na próxima seção é apresentada a arquitetura sugerida para um ambiente de desenvolvimento distribuído de software, e como esta arquitetura atende às necessidades

explicitadas pelas características citadas acima.

3. Arquitetura Proposta

A fim de sugerir uma arquitetura que dê suporte ao desenvolvimento distribuído de software e atenda às características citadas na seção 2, foi criado o projeto DiSEN. O DiSEN é um ambiente de desenvolvimento de software distribuído, incorporando a tecnologia de agentes segundo o padrão da FIPA (*Foundation for Intelligent Physical Agents*). Segundo Pascutti (2002), a arquitetura do DiSEN foi projetada para utilizar, dentre outras, a MDSODI (GRAVENA, 2000; HUZITA, 1999), que é uma metodologia para desenvolvimento de software que leva em consideração algumas características identificadas em sistemas distribuídos, tais como concorrência, paralelismo, comunicação, sincronização e distribuição. Em linhas gerais, o objetivo do DiSEN é fornecer o suporte necessário para o desenvolvimento do software distribuído; a equipe poderá estar distribuída em locais geográficos distintos e trabalhando de forma cooperativa com uma metodologia para desenvolvimento de software distribuído.

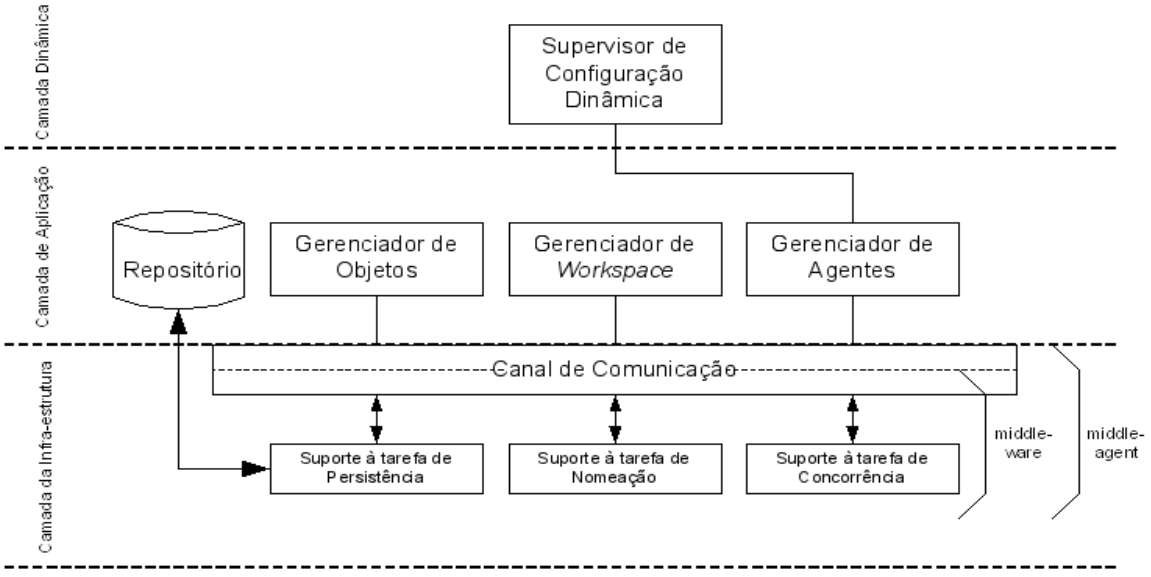

Figura 2. Arquitetura do DiSEN (Pascutti, 2002)

A arquitetura proposta para o DiSEN possui três camadas, que são descritas abaixo e ilustradas na Figura 2:

- *Camada Dinâmica:* responsável pela inserção, remoção e configuração dos componentes de software e serviços em tempo de execução*;*
- *Camada de Aplicação*: oferece suporte às metodologias de desenvolvimento de software (MDSODI, RUP, XP), o repositório para armazenamento dos artefatos e informações necessários ao ambiente e os gerenciadores de *workspace*, objetos e agentes;
- *Camada de Infra-estrutura:* define o alicerce da arquitetura, provendo suporte às tarefas de nomeação, persistência e concorrência, além de incorporar o canal de comunicação.

Como pode ser observada na Figura 2 cada uma das camadas contém módulos específicos. Estes módulos serão detalhados nas sub-seções a seguir.

3.1. Gerenciador de Objetos

Uma das principais tarefas dos ambientes de desenvolvimento de software é armazenar, estruturar e controlar, além do próprio software, os artefatos gerados durante o desenvolvimento. Esses artefatos são, por natureza, mais complexos que os itens tratados por sistemas de banco de dados tradicionais, pois são mais estruturados e requerem operações complexas. O Gerenciador de Objetos é constituído pelo gerenciador de atividades, gerenciador de recursos, gerenciador de artefatos, gerenciador de projetos, gerenciador de processos e gerenciador de versões e

configurações.

3.1.1. Gerenciador de Atividades

Este gerenciador será responsável pelo gerenciamento das atividades, atividades estas que são parte de um processo de software. Atividades incorporam e implementam procedimentos, regras e políticas e têm como objetivo gerar ou modificar um dado conjunto de artefatos. As atividades estão associadas a atores, ferramentas e artefatos. Uma atividade necessita de recursos (por exemplo, máquinas), é escalonada, monitorada e atribuída a atores, sendo que estes, por sua vez, podem utilizar ferramentas para executá-la. Uma atividade também pode ser executada somente por ferramentas automatizadas sem intervenção humana.

3.1.2. Gerenciador de Recursos

Este gerenciador será o responsável por fornecer o suporte ao gerenciamento de recursos. Os recursos necessários à realização das atividades podem ser materiais (computador, impressora, sala de reunião), ferramentas (programas de computador destinados ao suporte ou automação de parte do trabalho relacionado com uma atividade), ou recursos humanos. O gerenciador será responsável pelo gerenciamento de todos os recursos mantendo a atualização do estado de cada um, bem como a busca por um recurso similar quando um que for solicitado por uma determinada atividade estiver bloqueado ou em uso.

3.1.3. Gerenciador de Artefatos

Um artefato é um produto criado ou modificado durante um processo. Um artefato é resultado de uma atividade e pode ser utilizado posteriormente como matéria prima para aquela ou para outra atividade a fim de gerar novos artefatos. Dessa forma, uma atividade pode consumir artefatos (de entrada) e gerar novos artefatos (de saída). Outros artefatos são freqüentemente persistentes e possuem versões. No DiSEN um artefato pode ser um diagrama, modelo, manual, código fonte ou código objeto, entre outros.

3.1.4. Gerenciador de Versão e Configuração

Os artefatos são sensíveis ao tempo. Isto é, durante o desenvolvimento de software é inevitável que alguns artefatos sejam produzidos em várias versões. Em um ambiente de desenvolvimento de software cooperativo, as modificações efetuadas nos artefatos devem ser gerenciadas de forma efetiva, pois os mesmos são compartilhados por muitos profissionais. Para efetivar o envolvimento de vários profissionais nas tarefas que constituem o ciclo de vida de software, um mecanismo de versões de artefatos deverá estar disponível e o gerenciador de versão e configuração terá essa atribuição.

3.1.5. Gerenciador de Projetos

Segundo Conradi (1994), um projeto é a instância de um processo com objetivos e restrições específicos. Pode-se dizer que um projeto é um esforço para desenvolver um produto de software, ou seja, envolve uma estrutura organizacional, prazos, orçamentos, recursos e um processo de desenvolvimento. O Gerenciador de projeto será responsável por gerenciar os projetos criados, bem como as métricas e estimativas associadas a cada projeto.

3.1.6. Gerenciador de Processos

O processo de desenvolvimento de software (processo de software) pode ser compreendido como o conjunto de todas as atividades necessárias para transformar os requisitos do usuário em software, segundo Lima (1998). Um processo de software é formado por um conjunto de passos de processo, parcialmente ordenados, relacionados com conjuntos de artefatos, pessoas, recursos, estruturas organizacionais e restrições e tem como objetivo produzir e manter os produtos de software finais requeridos. O gerenciador de processos preocupa-se em analisar e verificar modelos de processo, obtendo informações durante a execução desses para que possam ser usados posteriormente. É também função deste gerenciador prover, de acordo com as diretivas do processo, a arquitetura do produto a ser desenvolvido (inicialmente está previsto suporte a arquiteturas baseadas em componentes e MDA).

3.2. Gerenciador de Workspace

O Gerenciador de *Workspace* proverá funcionalidades para criar um ou mais *workspace* com dados de um repositório sendo que, no caso do DiSEN, isso será feito através do canal de comunicação e do suporte à tarefa de persistência, conforme mostrado na arquitetura da Figura 2. No caso mais simples, isso consiste em copiar um arquivo simples, porém mais freqüentemente uma configuração inteira pode ser copiada do repositório para o *workspace* possibilitando ao desenvolvedor ter todos os módulos e documentos necessários para o sistema à sua disposição localmente. Assim, o desenvolvedor não tem de decidir o que deveria ser mantido globalmente no repositório e o que é necessário localmente para carregar suas mudanças. Além disso, ele está isolado das alterações das outras pessoas para o repositório – e outras pessoas estão isoladas das suas alterações. Isso significa que ele tem um controle completo do seu mundo e sabe exatamente o que foi alterado e por que. Quando o desenvolvedor termina de efetuar suas modificações, ele precisa adicionar os módulos e documentos modificados no repositório. Essa operação, no caso mais simples, consiste em adicioná-las ao repositório, usando a funcionalidade do gerenciador de controle de versão.
Em cada *workspace* existirão agentes locais que auxiliarão durante a execução das atividades, agentes de interação que auxiliarão nos trabalhos cooperativos e agentes de sistema que auxiliarão na coleta de métricas do sistema. Esses agentes interagirão com os demais enviando mensagens através do canal de comunicação.

3.3. Gerenciador de agentes

O gerenciador de agentes será o responsável pela criação, registro, localização, migração e destruição de agentes e consiste na região de agente (RA) e no servidor de ontologia. O DiSEN deverá ter um servidor de ontologia, que tem como função armazenar as ontologias necessárias para que os agentes possam se comunicar no ADS.

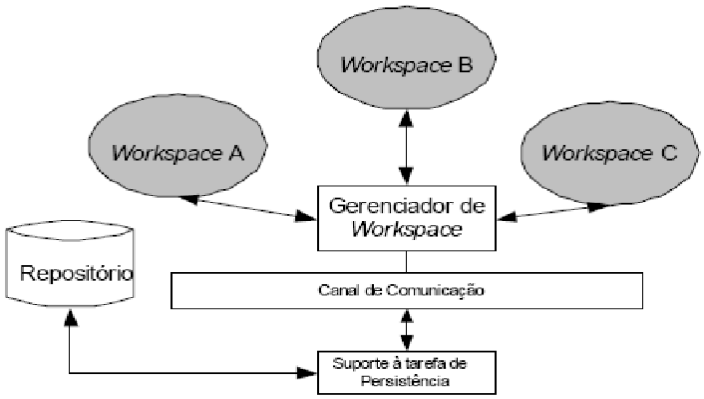

Figura 3. Comunicação inter-*workspaces* e dos *workspaces* com o Repositório

3.4. Canal de Comunicação

Toda troca de informação entre os elementos do DiSEN obrigatoriamente deverá ser por intermédio do canal de comunicação, que será responsável pela comunicação entre as camadas da infra-estrutura (*middleware* e *middle-agent*) e a camada de aplicação. Em uma aplicação que envolva somente objetos, a comunicação será estabelecida pelo *middleware*. Já no caso em que envolver também agentes, esta será gerenciada pelo *middle-agent.* Neste caso, além dos serviços convencionais de nomeação, por exemplo, entende-se que outras propriedades, tais como: colaboração, negociação, coordenação e mobilidade, devem ser consideradas. Existem também na literatura várias estruturas que oferecem o suporte ao desenvolvimento de agentes.

3.5. Considerações Finais

Esta seção apresentou o projeto da arquitetura de um ambiente de desenvolvimento de software distribuído. Essa arquitetura é baseada no estilo camadas, sendo composta pelas camadas dinâmica, de aplicação e da infra-estrutura. A camada dinâmica é responsável pelo controle e gerenciamento da configuração do ambiente, bem como dos serviços que podem ser acrescentados ao ambiente em tempo de execução. A camada de aplicação possui um gerenciador de objetos, um gerenciador de *workspace* e um gerenciador de agentes, contendo também o repositório para armazenamento das informações necessárias ao ambiente. E, a camada da infra-estrutura provê funcionalidades e suporte para as tarefas de nomeação, de persistência e de concorrência e, também, incorpora o canal de comunicação. A comunicação entre as camadas de aplicação e da infra-estrutura se dá através do canal de comunicação, ou seja, este canal é responsável pela comunicação entre os elementos da arquitetura.

Com relação às características citadas na seção 2 o DiSEN as atende da seguinte maneira:

- *Arquitetura de Produtos*: Componentes / MDA
- *Suporte a Metodologias de Desenvolvimento*: Camada de Aplicação oferece suporte para que seja utilizada qualquer metodologia.
- *Suporte a Gerência de Recursos*: Provido pelo Gerenciador de Recursos.
- *Gerência de Artefatos*: Provido pelo Gerenciador de Artefatos em conjunto com o Módulo de Suporte à Tarefa de Persistência.
- Suporte a Persistência de Dados: Provido pelo Módulo de Suporte à Tarefa de Persistência.
- *Gerência de Projetos/Processos*: Provido pelos gerenciadores específicos: Gerenciador de Processos, Gerenciador de Projetos e Gerenciador de Atividades.
- *Infraestrutura de Comunicação*: Canal de Comunicação.
- *Tecnologia de Colaboração*: Provido pelo Gerenciador de Workspaces

4. Exemplo de comunicação do DiSEN

O Ambiente de Desenvolvimento de Software poderá estar fisicamente distribuído, ou seja, pode-se ter um ou vários atores trabalhando em um local A, outro(s) em um local B e outro(s) em um local C. Cada local poderia ser visto como uma LAN (*Local Area Network*), contendo pelo menos uma RA, pelo menos um *workspace*, o gerenciador de objetos, o supervisor de configuração dinâmica, o canal de comunicação e o repositório. A comunicação entre regiões num mesmo local se dará através do canal de comunicação. As LAN's de cada local poderiam estar interligadas através de uma WAN (*Wide Area Network*), possibilitando a comunicação entre locais distintos.

Caso seja necessário armazenar algum objeto no repositório ou buscar informações a respeito de algum objeto armazenado, a comunicação também se dará pelo canal de comunicação que acionará o suporte à tarefa de persistência e este, por sua vez, irá acessar o repositório para atender à solicitação feita. Além disso, se for necessária a reconfiguração do ADS ou o acréscimo/solicitação de algum serviço, o agente monitor enviará uma mensagem para o gerenciador de agentes que enviará uma mensagem para o canal de comunicação a fim de executar a tarefa solicitada conforme mostra a Figura 4.

O canal de comunicação será, então, o elemento do DiSEN que será responsável por gerenciar toda a comunicação num local. Como se trata de um ambiente distribuído, este canal permitirá que um elemento do ambiente que pode ser um objeto ou um agente invoque transparentemente outro elemento local ou remotamente, sendo que, neste último caso, a comunicação poderia ser feita através de uma WAN. O canal intercepta a chamada e é responsável por encontrar um elemento no ambiente que possa implementar a requisição, passa-lhe os parâmetros e retorna seus resultados.

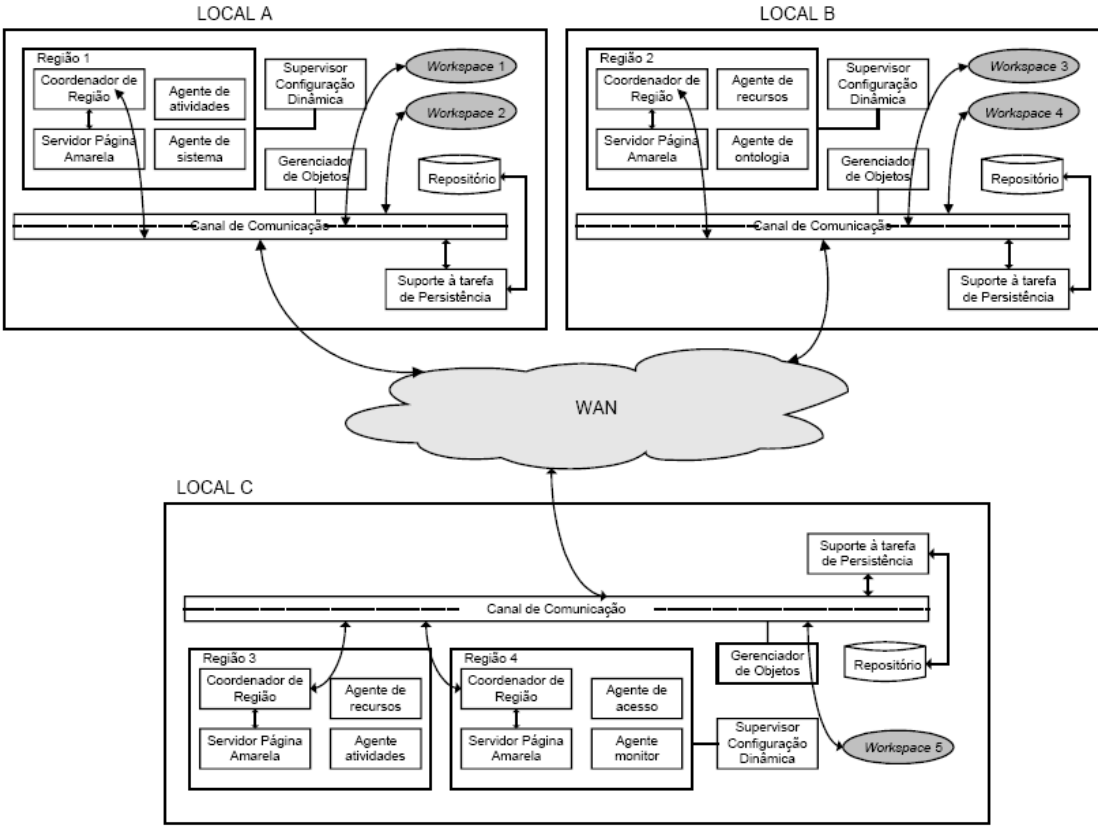

Figura 4. Exemplo de Comunicação Utilizando o DiSEN

Esta seção apresentou um cenário de interação entre vários workspaces do ambiente DiSEN, o que permitiu verificar como os gerenciadores previstos em seu projeto estarão atuando e oferecendo o apoio necessário ao trabalho cooperativo e distribuído no desenvolvimento de software.

5. Conclusões e Trabalhos Futuros

Este artigo apresenta um estudo relacionado a Ambientes de Desenvolvimento Distribuído de Software (ADDS), visando propor uma arquitetura de um ambiente de desenvolvimento distribuído de software (DiSEN). Para tanto, primeiramente foi realizado um levantamento bibliográfico e uma análise das abordagens e propostas de ADDS encontrados.

A avaliação dos ambientes encontrados foi realizada de maneira qualitativa, por meio da avaliação de algumas características desejáveis em ADDSs. As características foram enumeradas com base nas forças centrípetas para a formação de equipes distribuídas propostas por Carmel (1998) e nas características encontradas nos próprios ambientes estudados.

De acordo com esta avaliação é possível tirar algumas conclusões com relação aos ambientes descritos. Dentre os ambientes apresentados o único que apresenta propostas para suportar todas as características desejáveis é o *Odissey Share*. É importante ressaltar a grande gama de mecanismos de colaboração encontrados nesse ambiente. O Odissey Share deixa a desejar quando trata-se da especificação do mecanismo de persistência de dados. O projeto GENESIS não possui definições relacionadas à arquitetura de produtos, deixando uma característica sem suporte. Além de não possuir tal suporte, o GENESIS possui também um mecanismo de suporte à persistência sem definições muito claras, sendo tal suporte feito apenas por um banco de dados relacional. O Gossip apresenta todos os seus mecanismos de suporte às características desejáveis baseados em um mecanismo central de *awareness.* Estes mecanismos possuem definições gerais, algumas vezes sendo muito vagas. Não foram encontradas definições com relação às características de suporte à Arquitetura de Produtos, Metodologia de Desenvolvimento e Gerência de Recursos. Por último, o projeto MILOS não apresenta preocupação com o suporte à Arquitetura de Produtos e

Gerenciamento de Artefatos, e apenas define seu suporte à Persistência como sendo um banco de dados Orientado a Objetos, não havendo um módulo de suporte a tal tarefa. O MILOS se destaca pelo suporte à Metodologia de Desenvolvimento, sendo totalmente voltada para processos ágeis (XP, Scrum).

Após análise dos ambientes encontrados na literatura é apresentada uma proposta de arquitetura para ADDS, o DiSEN. Tal arquitetura é baseada no estilo arquitetural em camadas, sendo composta pelas camadas dinâmica, de aplicação e da infra-estrutura. A proposta visa permitir a construção de um ambiente de software que apóie o desenvolvimento cooperativo de software, onde as equipes estejam distribuídas cooperando na realização das tarefas do processo de software, e que atenda às características definidas na seção 2. A definição de tais características pode ser claramente observada na arquitetura do DiSEN, ilustrada na Figura 2 e são explicitadas na subseção 3.5.

Uma das contribuições e um diferencial desta arquitetura é a utilização da tecnologia de agentes na definição de alguns componentes da camada dinâmica e de aplicação. Estes agentes estão sendo propostos segundo as especificações da FIPA, uma organização que está se propondo a estabelecer um padrão a ser seguido para o desenvolvimento de software baseado em agentes. Outra característica importante desta arquitetura é a incorporação da MDSODI, uma metodologia para desenvolvimento de software que leva em consideração algumas características identificadas em sistemas distribuídos, tais como: paralelismo, concorrência, comunicação, sincronização e desenvolvimento de software, a MDSODI propõe também as representações adequadas para a especificação do processo.

Atualmente estão sendo definidos os mecanismos de colaboração, que deve contar com elementos de *awareness* síncronos e assíncronos, permitindo um alto grau de cooperação dentro do ambiente. Também está sendo proposto um mecanismo de persistência, que utiliza conceitos de bancos de dados distribuídos, fragmentação vertical e horizontal e replicação de dados. Por último, está em desenvolvimento o mecanismo de interoperabilidade de dados, baseado em MDA, que tem como objetivo permitir que usuários utilizem ferramental diferente sem haver perdas relacionadas à cooperação.

Agradecimentos

Agradecimentos especiais ao CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico - Brasil) pelo suporte financeiro ao projeto DiSEN e ao Grupo de Engenharia de Software e Sistemas Distribuídos da Universidade Estadual de Maringá.

Referências

AVERSANO, L.; CIMITILE, A.; LUCIA, A.; STEFANUCCI, S.; VILANN, M. L., Workflow Management in the GENESIS Environment. **Research Center on Software Technology**, Department of Engineering, University of Sannio, 2003.

BABAK, F. A., Gossip: An Awareness Engine for Increasing Product Awareness in Distributed Development Projects, In: International Conference on Advance Information Systems Engineering, Stockholm, 2000. **Proceedings...** p. 264-278.

CARMEL, E. **Global Software Teams**: Collaborating Across Borders and Time-Zone". Prentice Hall, EUA, 1999, 269p.

CONRADI, R. et al. EPOS: Object-Oriented Cooperative Process Modeling. In: FINKELSTEIN, A. et al. (Ed.). **Software Process Modeling and Technology**. Taunton: Research Studies Press, 1994. p. 33-70.

DART, S.; ELLISON, R.; FEILER, P.H.; HABERMANN, A.N., Software Development Environments, In: **Computer**, Novembro/1987. p. 18-28.

FALBO, R.A. **Integração de Conhecimento em um Ambiente de Desenvolvimento de**

Software. Tese de Doutorado, COPPE/UFRJ, Rio de Janeiro, 1998.

GOLDMANN, S.; MÜNCH, J.; HOLZ, H. Distributed Process Planning Support with MILOS. In: **Journal of Software Engineering and Knowledge Engineering**, Outubro, 2000.

GRAVENA, J.P., **Aspectos Importantes de uma Metodologia para Desenvolvimento de Software com Objetos Distribuídos.** Trabalho de Graduação, Universidade Estadual de Maringá, 2000.

HUZITA, E.H.M., **Uma Metodologia par a Desenvolvimento Baseado em Objetos Distribuídos Inteligentes**. Projeto de pesquisa, Universidade Estadual de Maringá, Departamento de Informática, 1999.

LIMA, C. A. **Um Gerenciador de Processos de Software para o Ambiente PROSOFT.** 1998. 197 p. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre. 1998

MAURER, F.; MARTEL, S., Process Support for Distributed Extreme Programming Teams. In: Internacional Conference on Software Engineering, Orlando, 2002. **Proceedings...** 2002.

MAURER, F.; DELLEN, B.; BENDECK, F.; GOLDMANN, S.; HOLZ, H.; KÖTTING, B.; SCHAAF, M. Merging Project Planning and Web-Enabled Dynamic Workflow Technologies. **IEEE Internet Computing.** v. 4, maio/2000. p. 65 - 74 .

PASCUTTI, M.C.D., **Uma proposta de arquitetura de um ambiente de desenvolvimento de software distribuído baseado em agentes**. 2002, 102 p. Dissertação (Mestrado) - Programa de Pós-Graduação em Computação, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2002.

PRIKLADNICKI, R.; AUDY, J.. MuNDDoS: Um Modelo de Referência para Desenvolvimento Distribuído de Software. In: 18 SBES - Simpósio Brasileiro de Engenharia de Software, 2004, Brasília. **Proceedings...** 2004.

WERNER, C.; MANGAN, M.; MURTA, L.; PINHEIRO, R.; MATTOSO, M.; BRAGA, R.; BORGES, M. OdysseyShare: an Environment for Collaborative Component-Based Development. In: IEEE International Conference on Information Reuse and Integration, 2003, Las Vegas. 2003. **Proceedings...** v. 1. p. 61-68.