OdysseyShare: Um ambiente para o Desenvolvimento Cooperativo de Componentes

Cláudia Werner[1], Marco Mangan[1], Leonardo Murta[1], Robson Pinheiro[1], Alessandreia de Oliveira[1], Marta Mattoso[1], Regina Braga[2], Marcos Borges[3]

odyssey@cos.ufrj.br
http://www.cos.ufrj.br/~odyssey/share
[1]Programa de Engenharia de Sistemas - COPPE/UFRJ
Caixa Postal 68511 – CEP. 21945-970 – Rio de Janeiro

[2]CTU – Universidade Federal de Juiz de Fora
[3]– Departamento de Ciência da Computação – IM-NCE/UFRJ

Abstract

This paper presents the preliminary results of the *OdysseyShare* Project, whose aim is to explore the collaborative aspects of component based software development.

1. Introdução

O Projeto *OdysseyShare* tem como objetivo explorar os aspectos colaborativos no Desenvolvimento Baseado em Componentes (DBC), através da construção de mecanismos para o suporte à interação em grupo no ambiente *Odyssey* [WER00].

A proposta do *OdysseyShare* é similar a de outros ambientes como Serendipity [GRU00], Gossip [FAR01] e MILOS [MAU02]. Esses ambientes são motivados pelas necessidades de equipes de desenvolvimento distribuídas, ou virtuais, característicos da globalização do desenvolvimento de software. Entre os requisitos desses ambientes, encontram-se: apoio à encenação de processos em equipes distribuídas, o incentivo à comunicação e à socialização, apoio à atividade individual e em pequenas equipes. Tais ambientes objetivam uma maior eficácia e eficiência no desenvolvimento de software e, sobretudo, uma maior satisfação dos indivíduos que participam desse processo.

No *OdysseyShare*, inicialmente, os processos de desenvolvimento da modelagem de domínio e da aplicação propriamente dita são planejados, utilizando-se uma máquina de processos, conforme descrito na Seção 2. Esta máquina permite a tomada de decisões gerenciais e o controle do fluxo de trabalho das equipes de engenheiros de software envolvidos nas atividades cooperativas de Engenharia de Domínio e de Aplicação suportadas pelo ambiente. Ao estabelecer estes processos definem-se, entre outras coisas, a seqüência de atividades, as ferramentas a serem utilizadas, os papéis dos desenvolvedores e os artefatos consumidos e produzidos. Estes artefatos são recuperados em repositórios de componentes disponíveis local ou remotamente (i.e. na Internet), através de mediadores e ontologias capazes de integrar as informações armazenadas em repositórios distribuídos (Seção 3).

De posse dos artefatos comuns de trabalho, os membros das diferentes equipes podem interagir através da edição colaborativa. Para atender alguns dos requisitos da edição colaborativa, um conjunto de componentes de percepção foi implementado no *OdysseyShare* para suporte a interações remotas e síncronas, de forma que qualquer ferramenta do ambiente possa utilizá-los (Seção 4).

2. Máquina de Processos

Conforme dito anteriormente, a máquina de processos *Charon* tem como objetivo prover coordenação sobre o desenvolvimento cooperativo de software no contexto do *OdysseyShare* [MUR02]. Para isto, é fornecido um ambiente de modelagem de *workflows*, baseado na notação do diagrama de atividades da UML, que permite a criação de processos

compostos (i.e. que contém um *workflow* que define como os seus sub-processos estão relacionados) e processos primitivos (i.e. que define o seu roteiro de execução, as ferramentas que devem ser utilizadas, os papéis de desenvolvedores autorizados, o tempo estimado de execução e os artefatos consumidos e produzidos).

Após a modelagem, os processos são instanciados em um determinado projeto. Essa instanciação é composta por duas etapas. A primeira etapa consiste na simulação do processo, visando a detecção de erros de modelagem e a geração dos tempos previstos de execução dos processos compostos. As técnicas utilizadas para simulação são: Simulação Baseada em Eventos e Simulação de Monte Carlo. A segunda etapa consiste na associação entre os desenvolvedores do projeto e os papéis modelados em cada sub-processo existente.

Internamente, a instanciação consiste no mapeamento do processo modelado para uma representação Prolog, que povoa a base de conhecimento do projeto. Essa base de conhecimento contém todas as informações sobre a execução do processo e, conseqüentemente, sobre o andamento do projeto. Posteriormente, essas informações são utilizadas e modificadas pelos agentes inteligentes existentes na máquina de processos.

Após a instanciação do processo, os desenvolvedores têm acesso à lista de pendências, permitindo que cada integrante do projeto saiba quais são as suas atividades pendentes. Para a execução de uma determinada atividade pendente, devem ser utilizadas as ferramentas associadas a essa atividade, contando com os recursos de percepção descritos na Seção 4. Além disso, também é possível que decisões sejam tomadas, mudando o curso do desenvolvimento do projeto em função de acontecimentos ocorridos no próprio desenvolvimento.

Durante o desenvolvimento do projeto, o gerente tem acesso a informações, em nível macro, do andamento do processo. Essas informações são descritas utilizando uma notação de cores e sobreposições (Figura 1). A notação de cores utiliza vermelho para indicar que uma determinada atividade foi executada em tempo pior que o previsto, verde para indicar que uma determinada atividade foi executada com tempo dentro do previsto e amarelo para indicar que a atividade ainda está em execução.

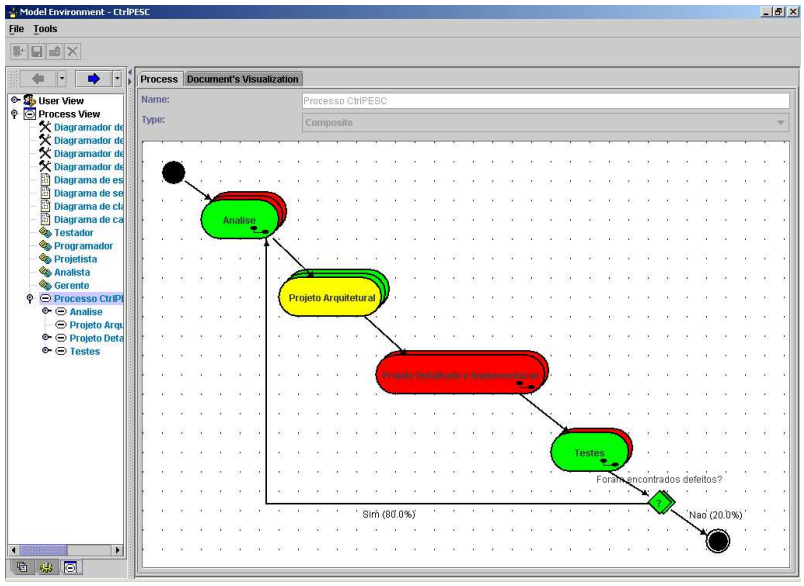

Figura 1 – Processo em execução

A notação de sobreposições indica quantas vezes uma determinada atividade foi executada, repetindo o desenho da atividade de forma sobreposta, com as execuções mais recentes sobrepostas às anteriores.

A flexibilidade da máquina de processos é obtida através da sua arquitetura baseada em agentes inteligentes. Inicialmente, quatro agentes inteligentes foram construídos:

- **Agente de Simulação:** Responsável por verificar o processo modelado quanto a sua corretude, permitindo ou não a sua entrada em execução, e calcular o seu tempo previsto de execução;
- **Agente de Execução:** Responsável por verificar o estado da base de conhecimento, procurando por atividades finalizadas ou decisões que foram tomadas, permitindo o andamento do processo;
- **Agente de Acompanhamento:** Responsável por interagir com o desenvolvedor, indicando as atividades que estão pendentes e as decisões que devem ser tomadas. Também permite a finalização dessas atividades ou a tomada dessas decisões;
- **Agente de Retrocesso:** Responsável por permitir o retorno da base de conhecimento para um determinado instante de tempo no passado, para que seja possível contornar erros na utilização da ferramenta.

Novos requisitos à máquina de processos podem ser modelados em novos agentes, que consultam e modificam a base de conhecimento em função dos seus objetivos. Cada agente construído pode agir de forma reativa, motivado por eventos vindos do ambiente ou de outros agentes, ou de forma pró-ativa, buscando o seu objetivo mesmo que nenhum evento externo tenha ocorrido.

3. Publicação e Recuperação de Componentes em Repositórios Heterogêneos e Distribuídos

O trabalho em equipe, no contexto do desenvolvimento cooperativo de componentes, pode envolver desenvolvedores em todo o mundo, produzindo e armazenando componentes em repositórios dos mais variados tipos e independentes uns dos outros (diferentes formatos de armazenamento, meios de acesso, capacidades de consulta, etc). Por esse motivo, a localização e recuperação de componentes de software, em especial os disponíveis na Internet, é uma tarefa complexa.

Neste sentido, foi elaborada uma arquitetura, denominada *ComPublish* [SOU01] (Figura 2), que visa auxiliar os desenvolvedores de software do *OdysseyShare* a publicar e recuperar variados artefatos de software na Internet, tal como modelos, diagramas, código fonte ou qualquer outro tipo de artefato utilizado ou produzido nas diferentes etapas do processo de desenvolvimento. Baseada no conceito de mediadores, a *ComPublish* provê uma visão lógica de vários componentes publicados, local ou remotamente, dentro de um mesmo domínio de aplicação. Seus principais serviços são:

- **Descrição e Publicação de Componentes:** A publicação de componentes em repositórios remotos é feita pelo Servidor *Le Select*, que fornece os mecanismos necessários para a abstração da heterogeneidade de diferentes repositórios de componentes a serem publicados na Internet. Através dos serviços de publicação de metadados do *Le Select*, componentes são descritos segundo o padrão XML. Um DTD XML pré-definido é fornecido para cada categoria de componentes, divididos em código, documentos e serviços. Para registrar um componente, o publicador deve fornecer informações como: domínio de aplicação, autor, linguagem, etc. Uma vez publicados, os componentes armazenados no repositório remoto passam a ser referenciados por um mediador específico dentro da camada de integração, através dos metadados exportados de cada componente pelo módulo publicador. Desta forma, cada componente se torna acessível às consultas submetidas por usuários na camada de aplicações. Uma vez descoberto e selecionado, a camada de publicação oferece serviços para transferir o componente remoto para a máquina local do usuário.

- **Serviços de Mediação para Integração de Metadados:** A camada de integração da *ComPublish* é formada por um conjunto de mediadores e tradutores que se comunicam entre si segundo um barramento CORBA. Os tradutores encapsulam a API cliente *Le Select*, que provê os serviços necessários para comunicação com os servidores *Le Select* remotos. Os mediadores, por sua vez, são responsáveis por organizar as informações dos componentes publicados segundo um determinado domínio de aplicação. Os serviços de armazenamento e consulta dos metadados e dos componentes locais a cada mediador são realizados pelo servidor de objetos GOA. Todas as informações de componentes exportadas pelos servidores *Le Select* no padrão XML são convertidas e armazenadas dentro do GOA segundo o modelo orientado a objetos (OO). Dentre os mediadores, existe ainda um mediador especial denominado Gerente de Serviços (GS), que gerencia as informações relacionadas aos demais mediadores desta camada. O GS é responsável por direcionar as consultas submetidas pela camada de aplicações aos mediadores adequados e realizar as ligações ontológicas intra e inter-domínios através do módulo GOS (*GOA Ontology Services*).

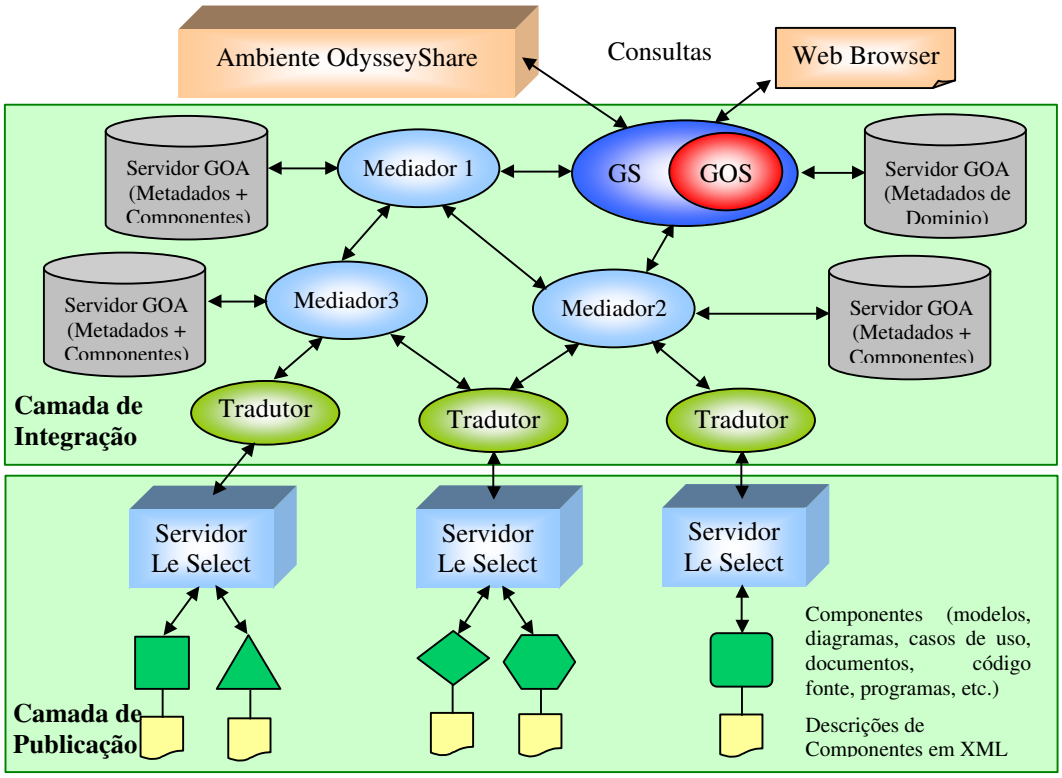

Figura 2 – Arquitetura *ComPublish*

- **Gerenciamento de Ontologias e Integração de Termos de Diferentes Domínios:** GOS é o módulo responsável por realizar as ligações ontológicas intra e inter-domínios. Este módulo realiza os mapeamentos necessários entre termos do mesmo e de diferentes domínios e mantém as informações atualizadas a respeito do domínio. GOS recebe do GS solicitações de serviços ontológicos, como por exemplo, criação de ontologias e inclusão de termos ontológicos, descoberta de informações sobre componentes relacionados ao termo em questão e estabelecimento de relacionamentos intra e interontológicos. GOS utiliza o algoritmo de *forward chaining* e algumas regras (armazenadas no GOA) com o objetivo de gerar novos relacionamentos intra e interontológicos, a partir de relacionamentos ontológicos já existentes.

4. Componentes de Percepção

Assumindo que boa parte do tempo do desenvolvimento de software é gasta na edição de documentos e que, com freqüência, a complexidade de um software extrapola as capacidades de um único indivíduo, sendo necessária a participação de diversos desenvolvedores na equipe, o apoio à edição colaborativa de documentos torna-se um requisito fundamental para o suporte à colaboração no desenvolvimento de software.

A edição de um documento por diversas pessoas é uma atividade comum, mas a edição colaborativa com o uso de editores colaborativos não é realizada com freqüência. Em grande parte, a razão para isso é que as facilidades naturais encontradas na edição de uma folha de papel sobre uma mesa, onde dois ou mais participantes expõem suas idéias e rascunham um documento, não se encontram presentes durante a edição através de uma ferramenta colaborativa. Portanto, o problema da percepção é introduzido pela necessidade de um melhor apoio à edição colaborativa.

Em uma sessão de edição colaborativa em tempo real, define-se como informação de percepção (*awareness information*) a informação que aumenta a percepção de cada participante em relação às ações dos outros usuários. Diversos mecanismos de colaboração têm sido propostos na literatura com o objetivo de aumentar a quantidade e a qualidade da informação de percepção disponível aos participantes [GUT99]. Tais mecanismos exploram algum tipo de representação gráfica que permite transferir informações a respeito da atividade dos demais participantes para cada participante. A informação de percepção completa a comunicação explícita oferecida por um canal de voz ou bate-papo disponível durante a edição do documento. Dois exemplos desses mecanismos são a visão em radar e o teleapontador (Figura 3).

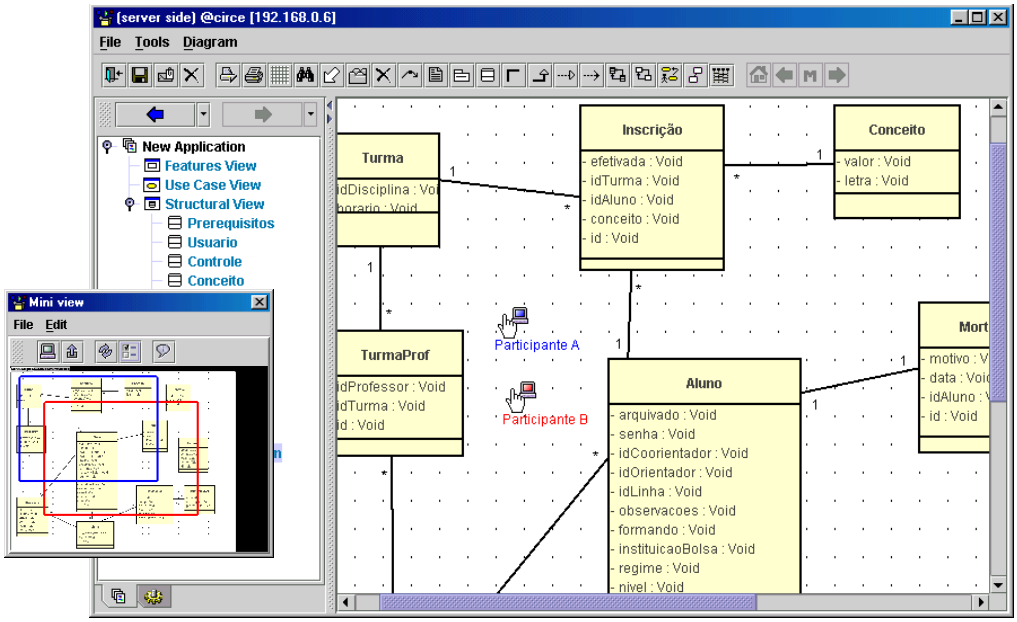

Figura 3 - Teleapontadores e visão de radar sobre um diagrama de classes da UML

A visão de radar (*radar view*) é um serviço de percepção comum em editores gráficos e textuais que auxilia a determinar a posição corrente de edição em um documento grande. Por exemplo, programas como *Acrobat Reader*, *Aladdin Ghost View* e *Microsoft PowerPoint* implementam esse serviço. No contexto de *groupware*, a visão de radar exibe informações sobre o posicionamento tanto do usuário local como dos demais usuários do grupo. O teleapontador (*telepointer*) apresenta a posição do cursor do mouse, ou de outro dispositivo de indicação, de cada usuário no grupo. Teleapontadores são populares em teleconferências, mas são vistos raramente fora do contexto de *groupware*.

Apesar dos efeitos positivos da utilização desses mecanismos sobre o desempenho dos participantes, sua aplicação fica restrita a aplicações desenvolvidas por grupos de pesquisa em CSCW [GUT99]. A questão da adoção destes e outros mecanismos disponíveis, porém pouco utilizados na prática, é atualmente um dos maiores desafios de CSCW. A proposta de componentização de mecanismos de percepção parte da premissa de que é possível reutilizar um mesmo mecanismo em diversas aplicações, adotando técnicas de DBC apropriadas. Cada componente realiza uma separação de interesses (*concerns*), uma segmentação vertical que realiza completamente o mecanismo, de forma que a aplicação integrada sofra o mínimo de alterações.

Durante a edição, duas ou mais instâncias de execução de uma aplicação formam uma única sessão colaborativa através do compartilhamento de eventos de interface. Os eventos são obtidos através do sistema de janelas (*windowing system*). Essa técnica de implementação é conhecida como arquitetura para colaboração flexível [BEG99]. Com isso, os componentes podem ser utilizados em qualquer aplicação que utilize o sistema de janelas.

A proposta dos componentes de percepção cumpre então dois objetivos: (a) oferecer informações de percepção para permitir a edição colaborativa de documentos no ambiente *OdysseyShare* e (b) apresentar uma alternativa para a adoção de mecanismos de CSCW através da criação de componente de percepção que podem ser integrados no projeto de um editor(es) já existente(s). A implementação atual dos componentes foi realizada na plataforma *Java*, usando os serviços do sistema de janelas oferecido pela *Java Foundation Classes* e as facilidades para programação distribuída da invocação remota de métodos (RMI - *Remote Method Invocation*).

5. Conclusões

Além dos trabalhos aqui apresentados, diversos outros trabalhos estão sendo desenvolvidos no contexto do Projeto *OdysseyShare*, dentre eles estão a geração da arquitetura de componentes a partir da modelagem funcional da aplicação, o suporte à percepção em bancos de dados orientados a objetos e o controle de concorrência de acesso a componentes através de agentes.

Agradecimentos. Ao CNPq e CAPES pelo apoio financeiro e a todos aqueles que participam deste projeto de pesquisa, por seu trabalho e dedicação.

Referências Bibliográficas

[BEG99] Begole, J.; Rosson, M.B.; Shaffer, C.A. “Flexible Collaboration Transparency: Supporting Worker Independence in Replicated Application-Sharing Systems”, ACM Transactions on Computer-Human Interaction, v. 6, n. 2, 1999, pp. 95-132.

[FAR01] Farshchian, B. A. “Integrating Geographically Distributed Development Teams Through Increased Product Awareness”, Information Systems Journal, v. 26, n. 3, Maio 2001, pp. 123-141.

[GRU00] Grundy, J.C.; Mugridge, W.B.; Hosking, J.G. “Constructing component-based software engineering environments: issues and experiences”, Information & Software Technology, v.42, n.2, 2000, pp. 103-114.

[GUT99] Gutwin, C; Greenberg, S. “The effects of workspace awareness support on the usability of real-time distributed groupware”, ACM Transactions on Computer-Human Interaction, v. 6, n.3, 1999, pp. 243-281.

[MAU02] Maurer, F.; Martel, S. “Process Support for Distributed Extreme Programming Teams”, Draft paper, Disponível em http://sern.ucalgary.ca/~milos/papers/2002/ MaurerMartel2002.pdf.

[MUR02] Murta, L.G.P.; “Charon: Uma máquina de processos extensível baseada em agentes inteligentes”, Dissertação de M.Sc., COPPE, UFRJ, Rio de Janeiro, 2002.

[SOU01] Souza, R. et al. "Software Components Reuse Through Web Search and Retrieval", International Workshop on Information Integration on the Web Technologies and Applications, Rio de Janeiro, Abr. 2001, pp.12-18.

[WER00] Werner, C. et al. "Infra-estrutura Odyssey: estágio atual", SBES’2000, Caderno de Ferramentas, João Pessoa, Out. 2000, pp. 366-369.