Um Gerenciador de Artefatos para Ambiente de Desenvolvimento Distribuído de Software

César Alberto da Silva
Universidade Estadual de Maringá (UEM), Departamento de Informática
Maringá, Paraná, Brasil
cesaralberto91@hotmail.com

e

Elisa Hatsue Moriya Huzita
Universidade Estadual de Maringá (UEM), Departamento de Informática
Maringá, Paraná, Brasil
elisa@din.uem.br

Abstract

The increasing complexity of the activities to be executed in a project of software development demands a greater interaction among the participants of the project, mainly they are dispersed geographically. The artefacts generated by execution of activities during the project development, consist of programs, models and documents that need to be stored. A greater interaction among the people, increase the necessity of exchange of information, artefacts, ideas among them. It also increase, the possibility to occur eventual conflicts when more than one person will be involved in an activity interacting one with other. So, mainly in a distributed software development environment it is import to make available an artefact manager. This paper presents an artefact manager. When is considered the scenario where the team members are in geographically distinct places, it is necessary to provide ways to treat the conflicts that would occur when more than one person try to update the artefacts concurrently. So this proposal intends to include one solution for the treatment of this situation, and so to maintain the artefacts version consistent.

Keywords: Artefact, Conflict, Awareness, ADDS.

Resumo

A crescente complexidade das atividades a serem realizadas em um projeto de desenvolvimento de software exige uma maior interação entre os participantes do projeto, principalmente se estas estiverem geograficamente dispersas. Os artefatos resultantes de atividades realizadas durante o desenvolvimento de um projeto, constituem-se de programas, modelos e documentos textuais que precisam ser armazenados. Com uma maior interação entre as pessoas, aumenta a necessidade de troca de informações, artefatos, idéias entre elas. Aumenta também, a possibilidade de eventuais conflitos quando mais do que uma pessoa estiver envolvida em uma atividade interagindo para gerar e, posteriormente, persistir o artefato resultante. Assim, é importante que esteja disponível, principalmente em um ambiente de desenvolvimento distribuído de software, um gerenciador de artefatos. Este trabalho têm por objetivo construir um gerenciador de artefatos. Em se tratando de um cenário em que os membros da equipe podem estar em locais geograficamente distintos, uma das peculiaridades desse gerenciador é o de incluir uma proposta de resolução para o tratamento de conflitos que podem surgir quando mais de uma pessoa estiver tentando realizar atualizações em artefatos.

Palavras-chaves: Artefato, Conflitos, Percepção, ADDS.

1 INTRODUÇÃO

Os projetos de desenvolvimento de software têm, progressivamente, aumentado de tamanho e complexidade, sendo cada vez mais comum sua realização por equipes de médio porte (entre dez e vinte desenvolvedores) e grande porte (acima de vinte desenvolvedores). Com as facilidades de comunicação proporcionadas pela Internet, a necessidade de experiência em diversas áreas de conhecimento e a pressão por cronogramas mais restritos, acabam fazendo com que alguns projetos sejam desenvolvidos por diversas equipes trabalhando cooperativamente. Além disso, as dificuldades em reunir os especialistas necessários em um mesmo local físico e a delegação do desenvolvimento de determinados componentes para outras empresas, são exemplos de fatores que podem exigir que as equipes participantes de um projeto estejam geograficamente distribuídas [12]. As organizações têm cada vez mais utilizado o desenvolvimento de software remoto como uma facilidade adicional, levando a o que é conhecido como Desenvolvimento Distribuído de Software (DDS) [7] [9].

No entanto, este novo cenário traz desafios para o processo de desenvolvimento. Por exemplo, o trabalho de equipes geograficamente distribuídas dificulta o controle de alterações nos componentes de um projeto em desenvolvimento [19]. Mesmo quando precedida de uma definição precisa das interfaces entre os componentes, a realização de um projeto pode exigir que diversos desenvolvedores alterem, simultaneamente, os mesmos componentes. Esta situação exige a adoção de políticas adequadas para manter a consistência entre os componentes, da versão atual, do projeto ou permitir que essa consistência seja posteriormente restituída.

O desenvolvimento do software deve tratar de duas fontes de complexidade: a complexidade do artefato que está sendo produzido e a complexidade das atividades em torno desse artefato. Modelos de processo de software tentam ajudar as equipes a lidar com a complexidade das atividades, enquanto técnicas focam no artefato [6].

O trabalho cooperativo e a utilização de técnicas de *awareness* ajudam a diminuir as dificuldades para a realização de atividades que são executadas por mais de uma pessoa. Desenvolvedores pertencentes a uma equipe de projeto podem estar dispersos geograficamente, apoiados por decisões estratégicas, motivados por redução de custos e aumento de produtividade. Assim, é desejável que as atividades sejam apoiadas por algum software (*groupware*), que permite minimizar o retrabalho, afim de manter a consistência e uniformidade dos artefatos produzidos.

Um artefato é resultado de uma atividade e pode ser utilizado posteriormente como matéria prima para aquela ou para outra atividade afim de gerar novos artefatos. Dessa forma, uma atividade pode consumir artefatos (de entrada) e gerar novos artefatos (de saída). Artefatos são freqüentemente persistentes, armazenados em repositórios, e possuem versões [18].

Assim, embora o gerenciamento das modificações efetuadas em artefatos seja um fator importante quando estes são compartilhados por várias pessoas em um ambiente de desenvolvimento distribuído de software, as soluções apresentadas na literatura, conforme serão vistas na seção 4, tratam principalmente de resolução de conflitos de artefatos no formato código fonte, deixando uma lacuna no que se refere a artefatos no formato XMI [17].

O objetivo deste trabalho é apresentar uma solução para os problemas, relacionados ao armazenamento de artefatos no repositório, que poderão ocorrer na execução das atividades de um projeto.

O gerenciador de artefatos ora apresentado, está situado, em termos da arquitetura de um ambiente de desenvolvimento, entre o usuário e o repositório de artefatos. O repositório de artefatos é responsável por manter e controlar as versões dos artefatos, podendo ser, neste caso utilizado, por exemplo, o SubVersion[1]. O gerenciador de artefatos adiciona novas políticas de acesso ao repositório de artefatos, facilita a interação dos desenvolvedores com o repositório de artefatos, mantém os desenvolvedores atualizados sobre o andamento das atividades do projeto e permite resolver conflitos de artefatos no formato XMI, em particular os artefatos gerados pela ferramenta Requisite [1].

[1]Version Control System. Disponível em: <http://subversion.tigris.org>.

Este artigo está organizado como segue: nas seções 2 e 3 serão apresentados, respectivamente, os conceitos de desenvolvimento distribuído de software e repositório de artefatos, na seção 4 serão apresentados os trabalhos relacionados, na seção 5 será apresentada o gerenciador de artefatos, o contexto para validação, bem como a forma dos experimentos do gerenciador de artefatos. Finalmente, na seção 6 são apresentadas as conclusões e trabalhos futuros.

2 DESENVOLVIMENTO DISTRIBUÍDO DE SOFTWARE

Ao longo dos anos o software se tornou um componente vital da maioria dos negócios. O sucesso das organizações depende da utilização do software como uma arma competitiva. Na década de 80, muitas organizações começaram a experimentar o desenvolvimento de software remoto como uma facilidade a mais [7].

A crescente globalização do ambiente de negócios e da economia tem afetado, diretamente, o mercado de desenvolvimento de software. Os engenheiros de software vêm reconhecendo, há algum tempo, a profunda influência do desenvolvimento global de software na globalização dos negócios e, através de reações alarmistas estão se movimentando para encontrar um modelo de negócio que possa atender a este mercado. Recentemente, a atenção está se voltando para o entendimento dos fatores que permitem às multinacionais e às corporações virtuais a operar com sucesso ultrapassando as fronteiras, geográficas e culturais, em busca de vantagens competitivas como baixos custos, maior produtividade e melhor qualidade na área de desenvolvimento de software [5] [7].

Este fenômeno é alimentado por fatores tais como: o acesso a uma grande quantidade de mão de obra especializada, redução nos custos do desenvolvimento, presença global e proximidade ao consumidor. Apesar do sucesso de várias equipes globais, as pesquisas revelam que a distância contribui para aumentar a complexidade nos processos organizacionais. Primeiramente, os processos de comunicação e coordenação são afetados pela distância, com consequências diretas na definição, construção, testes e entrega do software ao cliente final, assim como no gerenciamento do desenvolvimento [11].

Devido à necessidade de se manter equipes geograficamente dispersas, várias ferramentas e ambientes têm sido construídos para ajudar no controle e coordenação dessas equipes. O desenvolvimento distribuído de software (DDS) tem sido caracterizado principalmente pela colaboração e cooperação entre equipes que realizam atividades em conjunto, mas estão localizados temporal e fisicamente distantes, acrescentando assim, novos desafios ao processo de desenvolvimento de software [4] [20]. Em particular, o gerenciamento dos artefatos produzidos na execução das atividades do projeto, demanda uma maior atenção por parte do gerente de projeto [11].

3 REPOSITÓRIO DE ARTEFATOS

O repositório de artefatos é utilizado para o controle de versões dos artefatos que são gerados pelos responsáveis pela execução de atividades de um projeto. Os mais utilizados são: SubVersion, CVS[2], ClearCase[3] e SourceSafe[4].

Em um ambiente distribuído de desenvolvimento de software é desejável que as atividades de projeto sejam executadas de forma cooperativa. O acesso ao repositório de artefatos pode ser realizado por qualquer membro da equipe que tenha recebido atribuição para execução de uma atividade em um determinado projeto.

No entanto, mesmo as atividades sendo executadas cooperativamente, existem momentos aonde os desenvolvedores necessitam trabalhar individualmente. Portanto, é interessante que cada desenvolvedor tenha uma cópia do artefato compartilhado e, de tempos em tempos, sincronizem estas cópias. Dependendo do tempo em que um desenvolvedor fica sem sincronizar sua cópia, esta pode

[2]Concurrent Version System. Disponível em: <http://www.cvshome.org>.
[3]Rational ClearCase. Disponível em: <http://www-306.ibm.com/software/awdtools/clearcase>.
[4]Visual SourceSafe. Disponível em: <http://msdn2.microsoft.com/en-us/vstudio/aa718670.aspx>.

estar muito divergente das demais, acarretando em um esforço de convergência, culminando em retrabalho.

O problema pode surgir quando os artefatos são modificados e enviados novamente para o repositório, pois neste momento pode ocorrer um conflito. Por exemplo, dois membros de uma equipe de desenvolvimento estão trabalhando em uma mesma atividade, os dois solicitam o mesmo artefato no repositório e esse artefato está na primeira versão. Eles estão trabalhando individualmente, cada um fazendo suas alterações. Após as alterações serem efetivadas, chega o momento de submeter o artefato ao repositório. O primeiro que submeter, não terá problema. O artefato será gravado com as modificações efetuadas e então estará na segunda versão. Quando o segundo membro for submeter o artefato, ele não conseguirá, gerando assim um conflito, pois a versão que ele tinha do artefato está diferente daquela que está no repositório atualmente.

Conforme consta em [2], existem no mínimo duas formas de resolver esse problema. A coordenação das ações de cada membro do grupo pode ser realizada de forma pessimista ou otimista. Na forma pessimista, assume-se que as ações serão incoerentes e atingirão um estado inválido e indesejado. Neste caso, a ação de um membro deve bloquear as ações dos demais sobre o mesmo artefato.

Na forma otimista, assume-se que as ações serão coerentes, mesmo ocorrendo de forma isolada, e o resultado final será um estado válido. Neste caso, mais de um membro realiza ações sobre um mesmo artefato. O participante deve solicitar um bloqueio, se deseja trabalhar isoladamente sobre um determinado artefato. No entanto, o uso de bloqueios impossibilita o paralelismo de atividades que concorrem pelo artefato bloqueado. Em geral, os membros adotam a estratégia de geração de cópias do artefato, que são trabalhadas em paralelo e, posteriormente, consolidadas em uma única versão. Assim, por um lado o bloqueio evita a ocorrência de conflitos, mas reduz o paralelismo no trabalho, enquanto que o uso de cópias apenas posterga a detecção dos conflitos, mas possibilita um maior paralelismo de atividades.

Um mecanismo para reduzir o isolamento entre desenvolvedores é a percepção de mudanças entre as cópias do artefato. A informação de mudança deve ser disponibilizada para os desenvolvedores de forma a agilizar a detecção de possíveis conflitos sintáticos e semânticos, permitindo a visualização da evolução das cópias do artefato, tornando possível tomadas de decisões baseadas nas mudanças realizadas.

4 TRABALHOS RELACIONADOS

Nesta seção serão apresentadas algumas ferramentas relacionadas com o tema, encontradas na literatura, que propõem mecanismos para resolver o problema do gerenciamento de versões de artefatos.

A Ferramenta Token e LockED

Token: foi desenvolvida para apoiar o desenvolvimento concorrente de projetos de software, auxiliando na resolução dos problemas de controle de alterações nos componentes do projeto [15]. Suas principais funcionalidades são: o cadastramento dos desenvolvedores participantes do projeto, a troca de informações entre estes desenvolvedores e o controle de alterações nos componentes do projeto. Token foi desenvolvida em ambiente Linux, utilizando a linguagem de script PHP3, o banco de dados MySQL e o servidor Web Apache. A ferramenta é acessível via um navegador Internet, sendo portanto independente tanto da plataforma cliente quanto da plataforma servidora.

Essa ferramenta oferece um quadro de mensagens, indexadas por assuntos. Sempre que uma mensagem é enviada através da ferramenta Token, os desenvolvedores cadastrados recebem uma notificação por e-mail. A notificação indica o assunto da mensagem, seu tipo e contém um link para a página de mensagens do Token. Os tipos de mensagem utilizados no Token são: pergunta, resposta, informação e urgente. Os tipos de mensagem são meramente informativos, permitindo ao leitor uma

rápida identificação do objetivo da mensagem e a criação de filtros em seu e-mail. A ferramenta Token somente se preocupa com o código-fonte do sistema em desenvolvimento, não abrindo espaço para os demais produtos do processo de desenvolvimento no qual a equipe está inserida. Isto exige que o gerente da equipe procure por outras ferramentas, fazendo com que o sucesso do projeto se torne um fator de organização e comprometimento dos seus integrantes.

LockED: descreve uma abordagem que visa solucionar o problema de controle de alterações de artefatos de software no desenvolvimento distribuído de projetos de software. LockED visa controlar e impor ordem sobre a criação e as alterações de artefatos do domínio, disponibilizando-os para alocação e desalocação [24]. Seguindo o mesmo princípio da ferramenta Token, um usuário somente pode alterar um artefato quando este já estiver previamente alocado ao mesmo através da ferramenta LockED. A idéia básica é que haja uma base oficial de informações sobre o domínio em um servidor conectado à Internet/Intranet, a qual será acessada por toda a equipe. Cada integrante trabalhará em sua própria estação de trabalho utilizando a infra-estrutura de reutilização provida pelo seu ambiente de modelagem, realizando atividades de modelagem, consultas e instanciando aplicações a partir destes artefatos.

Ferramentas para Apoio à Percepção

Existem diversas propostas para apoiar a percepção em edição colaborativa. A ferramenta CO2DE [13] é um editor gráfico e síncrono de diagramas UML, baseado em uma metáfora de "máscaras", representando versões. Outro editor colaborativo de artefatos UML é D-UML [3]. Tukan [22] é um ambiente síncrono e distribuído de programação Smalltalk. O ambiente SAMS [14] permite a edição colaborativa através de interações síncrona, assíncrona e multi-síncrona. NetEdit [27] é um editor colaborativo multi-síncrono de documentos texto acessível pela Web.

A ferramenta Palantír [21] complementa sistemas de gerência de configuração, fornecendo aos desenvolvedores pertencentes à sessão de colaboração informações sobre os espaços de trabalho dos demais. Koblylinski et. al (2002) apresentam uma proposta de sistema de percepção, que permite que colaboradores monitorem atividades de outros sobre artefatos de software.

Em termos de mecanismos de percepção, CO2DE, D-UML e Tukan possuem suporte à interação síncrona, sendo que CO2DE provê também suporte assíncrono para percepção. Palantír, SAMS, NetEdit e a abordagem descrita em [10] contemplam mecanismos de percepção em interações multi-síncrona, além do suporte síncrono e assíncrono à percepção. Apoiando a percepção em modelos UML, tem-se CO2DE e D-UML.

Tukan utiliza como metáfora ícones usados em boletins meteorológicos; mudanças com alto impacto são associadas à ícones de tempo instável. Palantír fornece informações sobre severidade de mudanças sobre o artefato compartilhado através de diversas formas gráficas de representação (por exemplo, barra de progresso).

MAIS (Multi-synchronous Awareness InfraStructure)

A ferramenta está inserida no contexto do Projeto OdysseyShare [16], baseada na arquitetura cliente-servidor. Os eventos coletados são armazenados em um espaço de tuplas, tendo como implementação utilizada a especificada por JavaSpaces [23], visível a todos os desenvolvedores. Quando novos eventos são gerados, os desenvolvedores são notificados e os obtém do espaço de tuplas.

Os eventos são coletados do ambiente Odyssey, onde MAIS está inserida, e são apresentados aos desenvolvedores na forma de mensagens, descritas textualmente. Os desenvolvedores visualizam os eventos gerados, quem os gerou, além dos elementos do modelo envolvidos nos eventos. Eventos gerados pelo próprio desenvolvedor são apresentados em uma lista diferente dos gerados pelos demais.

Cada desenvolvedor pode tomar ciência sobre a ocorrência de determinado evento. Isto evita que o desenvolvedor seja alertado sobre a existência de eventos que já tenha tomado ciência em um momento anterior.

Augur: Combina Informações de Artefatos e Atividades

Augur é uma ferramenta desenvolvida por Froehlich e Dourich (2004) que permite a visualização de processos distribuídos do desenvolvimento do software. Augur gera representações visuais dos artefatos e das atividades do desenvolvimento do software e permite que os colaboradores explorem o relacionamento entre eles. O Augur é projetado para os colaboradores que participam no processo do desenvolvimento do software.

Froehlich e Dourich (2004) mostram que os artefatos podem carregar informações sobre as atividades em andamento ou concluídas, assim, tornando possível o próprio artefato extrair informações sobre as atividades que possuem dependências.

5 GERENCIADOR DE ARTEFATOS

Nesta seção serão apresentadas, o gerenciador de artefatos e como ele tratará a ocorrência de conflitos; a sua implementação; o contexto no qual serão realizados os experimentos do gerenciador de artefatos; a ferramenta Requisite e o método de avaliação do gerenciador de artefatos.

5.1 O Gerenciador de Artefatos

Os trabalhos pesquisados, na literatura, mostraram a necessidade de um gerenciador de artefatos, em um ambiente de desenvolvimento de software, devido à possibilidade de compartilhar artefatos entre os membros da equipe. Quando se pretende executar uma atividade que é realizada por um grupo de pessoas, é muito provável que surgirão alguns problemas. Esses problemas podem ser tanto de relacionamento entre os membros daequipe, como problemas na execução da mesma.

Sendo assim, durante a execução de uma atividade, podem ocorrer conflitos entre versões de um artefato. Dentre as ferramentas analisadas, na seção 4, percebeu-se que elas oferecem suporte apenas para resolver conflitos de artefatos no formato código fonte, e não oferecem um mecanismo para resolver conflitos em artefatos no formato XMI.

Além disso, na maioria das vezes, as ferramentas encontradas no mercado (NetBeans, Eclipse, SmartSVN[5], entre outras) disponibilizam mecanismos ou plugins que auxiliam o desenvolvedor à resolver os conflitos em artefatos do tipo código fonte. Os artefatos de código fonte são gerados pelo próprio desenvolvedor, o que não acontece com os artefatos no formato XMI, que são gerados por ferramentas. Por isso, os artefatos de tipo código fonte são mais fáceis de serem entendidos no momento de resolver eventuais conflitos.

Portanto resolver conflitos de artefatos no formato XMI se torna uma tarefa não trivial para os desenvolvedores.

A figura 1 simula uma situação real, onde dois membros de uma equipe estão alocados para realizar a mesma atividade. Assim, eles compartilharão os mesmos artefatos. Cada um possuirá em seu computador, no repositório local, uma cópia dos artefatos que estão no repositório central.

No repositório central estão armazenadas todas as versões dos artefatos, sendo a última versão considerada como válida. O SubVersion é uma das alternativas, encontradas no mercado, utilizada para controlar as versões dos artefatos no repositório central. Como repositório local é usado o sistema de arquivos do sistema operacional.

O gerenciador de artefatos é responsável em atender as solicitações dos usuários para acesso aos artefatos. Portanto, ele deverá fornecer facilidades, aos usuários, para manipulação dos artefatos como: requisitar um artefato, submeter um artefato, excluir um artefato, fornecer informações a

[5]The Smart Subversion-Client. Disponível em: <http://www.syntevo.com/smartsvn/>.

respeito do membro que fez a última modificação no artefato, quando foi realizada a última modificação.

De um modo geral, pode-se identificar duas situações no desenvolvimento de software.

A primeira situação (figura 1a): os desenvolvedores podem trabalhar de maneira cooperativa. Neste caso, as dificuldades para a realização de uma atividade seriam menores, pois cada um saberia o que o outro está fazendo. Entretanto, assim mesmo poderia ocorrer conflito, caso os dois realizassem modificações em uma parte comum do artefato. O ideal seria que eles cooperassem, e que só um efetivasse as modificações decididas por ambos.

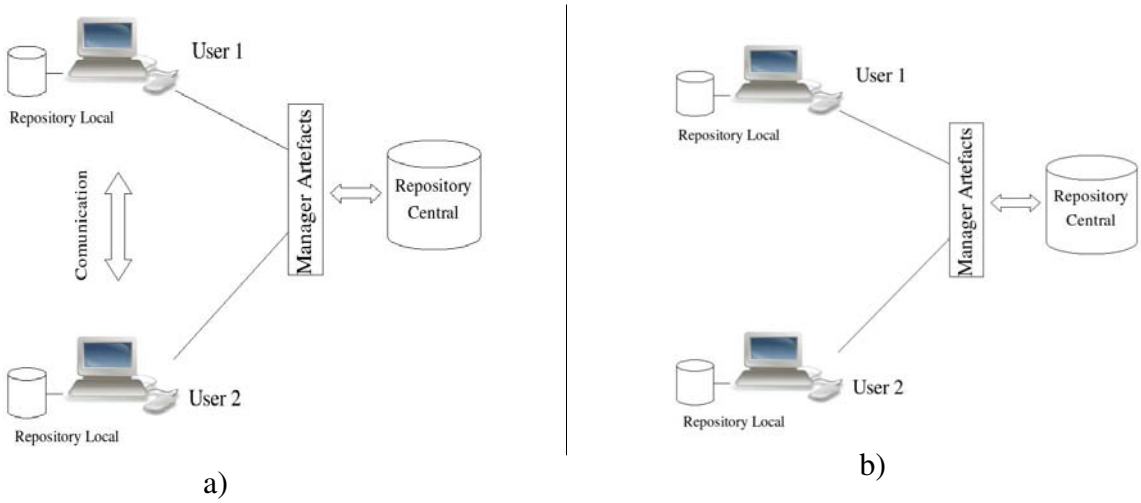

Figura 1a: Compartilhamento de Artefatos com comunicação entre os usuários. Figura 1b: Comportilhamento de Artefatos sem comunicação entre os usuários.

A segunda situação (figura 1b): os desenvolvedores podem trabalhar isoladamente, não havendo cooperação entre eles. Dessa maneira a probabilidade de haver divergência entre as versões de cada um, é maior. Neste caso, o primeiro que submeter a nova versão do artefato, ao repositório central, não terá problema. O conflito surgirá quando o segundo usuário tentar submeter a sua versão do artefato. Ele não conseguirá, pois a versão atual do artefato, no repositório central, é diferente da versão de quando ele fez a requisição do mesmo.

Uma solução apresentada, pelo gerenciador de artefatos proposto, é a utilização de bloqueio (trava) por um período de tempo para cada atividade. O tempo de bloqueio é definido pelo gerente de projeto. Essa flexibilidade é permitida em razão de cada projeto possuir suas peculiaridades. Por exemplo, uma atividade que seja mais complexa, o tempo de bloqueio pode ser aumentado para que tenha tempo suficiente para o desenvolvedor realizar as modificações necessárias.

O gerenciador de artefatos, possui também um mecanismo para resolver conflitos de artefatos no formato XMI, constituindo-se assim em uma importante contribuição. A subseção 5.4 explica como os experimentos serão realizados para validação do gerenciador de artefatos.

Uma outra funcionalidade oferecida pelo gerenciador de artefatos, ora apresentado, é permitir a combinação de informações das atividades e das versões dos artefatos. Com isso, pode-se ter conhecimento de várias informações sobre o progresso do projeto como um todo. Por exemplo, a partir do mapeamento das atividades com os artefatos, pode-se ter uma visão do progresso das atividades, quando e quais foram os desenvolvedores que tiveram participação na elaboração de um artefato em específico; para quem o artefato está alocado; quem está aguardando a liberação do artefato; quais são as atividades que dependem do artefato que está sendo produzido, dentre outros.

5.2 A Implementação

O gerenciador de artefatos está sendo implementado como um componente do ambiente DiSEN. A partir dos estudos realizados, puderam ser identificadas como funcionalidades importantes deste gerenciador:

- armazenar o artefato;
- recuperar o artefato;
- excluir o artefato;
- requisitar a trava;
- liberar a trava;
- definir tempo para alocação da trava;
- listar alterações do artefato;
- oferecer suporte para resolver conflitos.

O suporte à resolução de conflitos está sendo implementado por meio de um mecanismo e será integrado à ferramenta Requisite para os experimentos. Assim, os artefatos gerados pela ferramenta Requisite poderão ser armazenados e recuperados no repositório central. Além disso, se ocorrerem conflitos com a versão do artefato, o mecanismo deverá fornecer facilidades para que o desenvolvedor possa resolvê-los. Caso ocorram conflitos, a ferramenta apresenta, visualmente, os elementos que foram adicionados ou removidos no artefato. Os elementos que estão apresentando divergência, nas versões do artefato, serão visualizados com cores distintas, para que o desenvolvedor perceba com maior facilidade.

5.3 Contexto Considerado para Realização dos Experimentos com o Gerenciador de Artefatos

O DiSEN (Distributed Software Engineering ENvironment), um ambiente de desenvolvimento de software distribuído, incorporando a tecnologia de agentes segundo o padrão da FIPA (Foundation for Intelligent Physical Agents), utiliza a MDSODI [8], uma metodologia para desenvolvimento de software que leva em consideração algumas características identificadas em sistemas distribuídos, tais como concorrência, paralelismo, comunicação, sincronização e distribuição.

O objetivo do DiSEN é fornecer o suporte necessário para o desenvolvimento distribuído de software; a equipe poderá estar distribuída em locais geográficos distintos e trabalhar de forma cooperativa usando uma metodologia para desenvolvimento distribuído de software.

Requisite

A Requisite é uma ferramenta nativa do ADDS (Ambiente de Desenvolvimento Distribuído de Software) DiSEN que tem por objetivo auxiliar na modelagem de requisitos, prover um meio de comunicação entre os stakeholders, prover apoio à rastreabilidade e à documentação de requisitos no DiSEN [1].

As funcionalidades de criação do diagrama de caso de uso, criação de atores e casos de uso, bem como a recuperação e a persistência do modelo (salvar modelo), foram reimplementadas por Wiese (2006), o que facilita a importação e exportação de artefatos.

Na figura 2 são apresentadas as funcionalidades da ferramenta Requisite.

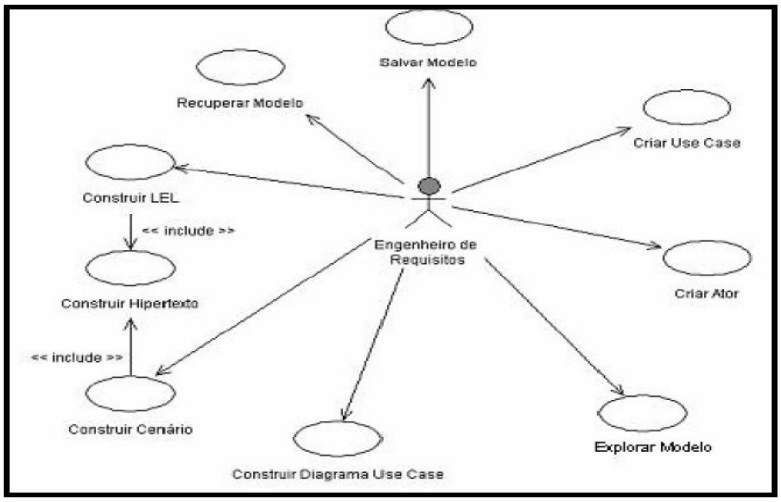

Figura 2: Funcionalidade da Requisite
Fonte: BATISTA (2003)

5.4 Método de Avaliação

A avaliação do gerenciador de artefatos será feita de acordo com a engenharia experimental [25]. O gerenciador de artefatos será integrado ao DiSEN, que oferece uma estrutura necessária para a realização dos experimentos.

A Requisite utiliza um metamodelo para a criação dos elementos no diagrama, dessa maneira, é possível mapear os objetos (elementos do diagrama de caso de uso descritos no metamodelo) e realizar neles as transformações necessárias e aplicar os filtros de importação e exportação, para a criação de arquivos XMI [26].

O gerenciador de artefatos será utilizado para auxiliar no controle das versões dos artefatos criados pelos usuários do DiSEN. Inicialmente, os experimentos estão sendo realizados utilizando o SubVersion, mas a estrutura do gerenciador de artefatos fornece uma interface para a utilização de outros repositórios de versões, por exemplo o CVS, ClearCase ou SourceSafe.

6 CONCLUSÕES

Este artigo apresenta um gerenciador de artefatos que oferece apoio à redução de conflitos. Assim, como contribuições deste trabalho pode-se destacar: facilidade para o acesso ao repositório de versões como, requisitar um artefato, submeter um artefato, mostrar a diferença entre duas versões de um artefato, exibir quando foi a última vez que um artefato foi modificado, quem fez a última modificação no artefato, quem fez mais modificações no artefato; melhor controle de alocação do artefato ao desenvolvedor (utilizando controle por tempo); a possibilidade de resolver conflitos entre versões de artefatos no formato XMI; e o uso de técnicas de *awareness,* para manter os membros da equipe atualizados sobre as modificações realizadas nos artefatos e sobre o andamento do projeto como um todo.

No contexto do desenvolvimento distribuído de software, quando um desenvolvedor requisita para trabalhar em uma atividade, o artefato é locado para o desenvolvedor daquela atividade. Quando o desenvolvedor concluir a atividade, ele desaloca o artefato. Se existisse uma atividade que fosse desenvolvida em paralelo, ou que fosse atribuída para mais de um desenvolvedor, e eles precisassem trabalhar com esse artefato, somente o primeiro que alocasse o artefato teria a permissão para alterar o artefato. Este requisito é atingido pela ferramenta LockED, ora apresentada, mas não oferece controle algum quanto a forma como ocorrem essa alocação e resolução de conflitos. O gerenciador apresentado possui este diferencial.

Para haver um melhor controle na alocação dos artefatos, o gerenciador de artefatos permite que seja controlado o tempo de alocação de um artefato para um desenvolvedor. Quando um desenvolvedor solicitar um artefato e esse já estiver alocado, o desenvolvedor irá para uma fila e assim que o artefato fosse liberado (pelo tempo excedido ou pela livre desalocação do desenvolvedor) o primeiro da fila é alocado para o uso do artefato.

Um outro problema se refere aos artefatos no formato XMI. A manipulação desse tipo de artefato pelo desenvolvedor, não é trivial, quando comparada com um artefato de código fonte. Para resolver esse tipo problema, o gerenciador de artefatos, apresenta um mecanismo para auxiliar o desenvolvedor a resolver conflitos entre duas versões de um artefato, distinguindo com cores, os objetos do diagrama de caso de uso, que estão divergentes em cada versão.

Como trabalhos futuros:

- Problemas de relacionamento entre os membros da equipe.
- Criação de um mecanismo para gerenciamento de repositórios distribuídos.
- Implementar um mecanismo para tratar conflitos de outros modelos da UML, além de use case.

Agradecimento: Os autores agradecem o Conselho Nacional de pesquisa (CNPq) pelo apoio financeiro ao projeto de pesquisa. Número do Processo: 506511/2004-9.

REFERÊNCIAS

[1]BATISTA, S. M. Uma ferramenta de apoio á fase de requisitos da MDSODI no contexto do ambiente DiSEN. 83 f. Dissertação (Mestrado) – Programa de Pós- Graduação em Informática, Universidade Federal do Paraná, Curitiba, 2003.

[2]BORGHOFF, U. M., SCHLICHTER, J. H. Computer-Supported Cooperative Work: Introduction to Distributed Applications. *Springer*, USA. 2000.

[3]BOULILA, N., DUTOIT, A. H., BRUEGGE, B. *D-Meeting: an Object-Oriented Framework for Supporting Distributed Modelling of Software*. In: Int. Workshop on Global Soft. Development, Int. Conf. on Soft. Eng., pp. 34-38, Maio, EUA. 2003.

[4]CARMEL, E. Global Software Teams – Collaborating Across Borders and Time- Zones. Prentice Hall, USA, 1999, 269p.

[5]FREITAS, A., MAIA, A., NUNES, D. Um Modelo para Interação entre Processos de Software. In: Congresso Brasileiro de Computação, CBCOMP, 4., 2004, Itajaí. *Anais*. Itajaí: Univali, 2004. p. 149 p 154.

[6]FROEHLICH, J. and DOURISH, P. Unifying Artifacts and Activities in a Visual Tool for Distributed Software Development Teams. Proceedings of the International Conference on Software Engineering ICSE 2004 (Edinburgh, UK), 387-396. 2004.

[7]HERBSLEB, J. D., MOITRA, D. Guest Editors' Introduction: Global Software Development, IEEE Software, vol. 18, no. 2, pp. 16-20, March/April, 2001.

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

[9]KIEL, L. Experiences in Distributed Development: A Case Study, In: Workshop on Global Software Development at ICSE 2003", Oregon, EUA. 2003.

[10]KOBYLINSKI, R., CREIGHTON, O., DUTOIT, A., BRUEGGE, B. Building awareness in distributed software enginering: Using issues as context. In: International Workshop on Distributed Software Development, Int. Conf. on Soft. Eng., Orlando, EUA. 2002.

[11]LANUBILE, F., DAMIAN, D., OPPENHEIMER, H. L. Global Software Development: Technical, Organizational, and Social Challenges. *ACM SIGSOFT Software Engineering Notes*. Volume 28 Number 6. November 2003.

[12]MAIR, Q. Technical Issues in the Design of a Virtual Software Corporation; ECSCW'97 OOGP workshop. 1997.

[13]MEIRE, A. P., BORGES, M. R. S., ARAÚJO, R. M. Supporting Collaborative Drawing with the Mask Versioning Mechanism. *Lecture Notes in Computer Science*, Vol. 2806, p.p. 208-223, Berlim, Alemanha. 2003.

[14]MOLLI, P., SKAF-MOLLI, H., OSTER, G., JOURDAIN, S. SAMS: Synchronous, Asynchronous, Multi-Synchronous Environments. In: 7th Int. Conf. on Computer Supported Cooperative Work in Design. CSCWD´2002, pp. 80-84, Rio de Janeiro, Brasil. 2002.

[15]MURTA, L. G. P., BARROS, M., WERNER, C. Token: Uma Ferramenta para o Controle de Alterações em Projetos de Software em Desenvolvimento, XIV Simpósio Brasileiro de Engenharia de Software, Sessão de Ferramentas, João Pessoa, outubro 2000.

[16]ODYSSEY, Projeto Odyssey. 2004. Disponível em: <http://www.cos.ufrj.br/~odyssey>. Acesso em 05 de fevereiro de 2007.

[17]OMG XMI. XMI Specification 2.1. 2005. Disponível em: <http://www.omg.org/technology/documents/formal/xmi.htm>. Acesso em 27 de Março de 2007.

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

[19]PRESSMAN, R. S. Software Engineering: A Practitioner's Approach. Mc Graw Hill, 5 ed., EUA, 2001.

[20]PRIKLADNICKI, R., LOPES, L., AUDY, J. L. N., EVARISTO, R.. Desenvolvimento Distribuído de Software: Um Modelo de Classificação dos Níveis de Dispersão dos Stakeholders. In: I SBSI - Simpósio Brasileiro de Sistemas de Informação, 2004, Porto Alegre. I SBSI - Simpósio Brasileiro de Sistemas de Informação, 2004. v. 1. p. 253-262.

[21]SARMA, A., NOROOZI, Z., VAN DER HOEK, A. Palantír: Raising Awareness among Configuration Management Workspaces. In: Proceedings of the 25th International Conference on Software Engineering (ICSE 2003), pp. 444-454, Maio, EUA. 2003.

[22]SCHUMMER, T.; SCHUMMER, J. Support for Distributed Teams in eXtreme Programming. In: Proceedings of eXtreme Programming and Flexible Processes Software Engineering - XP2000, Addison Wesley. 2000.

[23]SUN, JavaSpaces. Disponível em: <http://java.sun.com/developer/Books/JavaSpaces>. Acesso em 09 de março de 2007.

[24]TEIXEIRA, H. V., MURTA, L. G. P., WERNER, C. M. L. LockED: Uma Abordagem para o Controle de Alterações de Artefatos de Software, In: IV Workshop Ibero-americano de Engenharia de Requisitos e Ambientes de Software (IDEAS'01), pp.348-359, San José, Costa Rica. 2001.

[25]TRAVASSOS, G. H.; GUROV, D.; AMARAL, E. A. G. G. Introdução à engenharia de software experimental. Relatório Técnico RT-ES-590/02. Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ, 2002. 52 p.

[26]WIESE, I. S. Um Modelo de Interoperabilidade para Ambientes de Desenvolvimento Distribuído de Software. 90 f. Dissertação (Mestrado) – Programa de Pós- Graduação em Ciência da Computação, Universidade Estadual de Maringá, Maringá, 2006.

[27]ZAFFER, A., SHAFFER, C., EHRICH, R., PEREZ, M. NetEdit: A Collaborative Editor, TR-01-13, Computer Science, Virginia Tech, EUA. 2001.