DiSEN-U*pdater*: Um Mecanismo de Atualização Dinâmica de Ferramentas para um Ambiente de Desenvolvimento Distribuído de *Software*

Rafael Cassolato de Meneses, Elisa Hatsue Moriya Huzita

Departamento de Informática – Universidade Estadual de Maringá (UEM)
CEP 87020-900 – Maringá – RS – Brasil

{cassolato,emhuzita}@din.uem.br

Abstract. *Throughout software development process, team members demand for tools in order to produce project artifacts. Hence, these tools require updating for bugs fix and addition of new features. The updating process management in a Distributed Software Development Environment (DSDE) introduces challenges related to the updating of available tools in that. This paper aim to present an updating engine, responsible for manage the tools updating process in a DSDE in a dynamic way.*

Resumo. *Durante o processo de desenvolvimento de software, os membros da equipe necessitam de ferramentas para produzir os artefatos do projeto. Conseqüentemente, essas ferramentas necessitam ser atualizadas para correções de falhas e adição de novas características. O gerenciamento do processo de atualização em um Ambiente de Desenvolvimento Distribuído de Software (ADDS) traz desafios em relação à atualização das ferramentas disponíveis nesse ambiente. Este artigo tem por objetivo apresentar um mecanismo de atualização, responsável por gerenciar o processo de atualização de ferramentas num ADDS de forma dinâmica.*

1. Introdução

As organizações têm, cada vez mais, utilizado o desenvolvimento de *software* remoto como uma facilidade adicional, levando ao que é conhecido como Desenvolvimento Distribuído de *Software* (DDS) [Herbsleb e Moitra 2001]. Devido à dispersão de recursos humanos capacitados, muitas organizações encontram no DDS uma alternativa para trabalhar com equipes geograficamente distantes entre si [Kiel 2003].

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*), a fim de manter a consistência e uniformidade dos artefatos produzidos e desta forma minimizar o retrabalho [Silva e Huzita 2008].

Para apoiar a colaboração em DDS, são utilizados sistemas computacionais que oferecem suporte para o desenvolvimento, manutenção e melhorias dos produtos de *software*. Esses sistemas computacionais compõem os chamados Ambientes de Desenvolvimento Distribuído de *Software* (ADDS), cuja meta é permitir o trabalho cooperativo de maneira mais produtiva, auxiliando a comunicação de idéias,

compartilhamento de recursos e coordenação dos esforços de trabalho [Fuks, Raposo e Gerosa 2002].

As ferramentas utilizadas para apoiar o desenvolvimento de produtos de *software* podem sofrer constantes atualizações, seja para correções de falhas, quanto para adição de novas características, muitas vezes o processo de atualização exige uma pessoa especializada para executá-lo. Segundo [Lüer e Hoek 2004], o processo de instalação e configuração de componentes de *software*, quando feito manualmente, aumenta as chances de erro.

Infelizmente, as abordagens tradicionais de atualização geralmente requerem que seja encerrada a execução, aplicadas as atualizações, para então reiniciar o sistema, diminuindo desta forma a disponibilidade do mesmo [Chen et al. 2007].

Esse problema se agrava quando o foco está no contexto de DDS, pois quando se trabalha com equipes geograficamente distribuídas é necessário que haja um membro em cada equipe que seja responsável por executar a atualização nas instâncias dos ambientes dos usuários da sua equipe. Isto aumenta a possibilidade de que o processo de atualização não seja executado de forma correta e no momento exato que deveria ocorrer.

Outro problema está no gerenciamento dos módulos (pacotes) de *software* que cada membro da equipe esteja utilizando em sua instância do ambiente, sendo praticamente impossível ter uma lista atualizada desses.

A atualização dinâmica é uma técnica que permite gerenciar o processo de atualização de *software* enquanto este está sendo executado, não sendo necessário o processo de parar e reiniciá-lo novamente.

Assim, torna-se necessário a utilização de um mecanismo de atualização dinâmica em um ambiente de desenvolvimento. Com isto, é possível manter uma lista atualizada dos módulos que cada usuário possui em seu ambiente; garantir que um usuário utilize somente os módulos que lhe foram atribuídos pelo seu gerente (ou responsável), o que impossibilita que um usuário malicioso tenha, em sua instância, um módulo que não lhe tenha sido atribuído a permissão de uso.

Esse artigo apresenta o DiSEN-*Updater*, um mecanismo de atualização dinâmica de ferramentas para um ADDS. Descreve-se a solução adotada em sua implementação para minimizar os problemas referentes à atualização dinâmica num ADDS, bem como o serviço gerenciador do repositório responsável por manter a lista de módulos que cada usuário deve ter disponível em sua instância do ambiente.

O artigo está organizado em quatro seções, além dessa introdução. A Seção 2 apresenta os trabalhos relacionados e as características ausentes nos mesmos, que motivaram o desenvolvimento do DiSEN-*Updater*. A Seção 3 fornece uma visão geral do contexto onde o DiSEN-*Updater* está inserido. A Seção 4 apresenta o DiSEN-*Updater* e as características implementadas em relação à atualização dinâmica do ambiente. Por fim, a Sessão 5 apresenta as conclusões.

2. Trabalhos Relacionados

Os seis trabalhos apresentados foram escolhidos por possuírem algumas das seguintes características: fazer uso de módulos de *software*, utilizar de repositórios de pacotes de *software*, tratar do processo de atualização, estar inserido no contexto de ambientes distribuídos, fazer uso da associação de ferramentas por perfil de usuário (baseado no LDAP[1]).

O GATI (Gerência de Ambiente de Tecnologia da Informação) é um ambiente integrado para administração de diretórios distribuídos, desenvolvido para gerenciar ambientes de TI [Cruz et al. 2004]. Este faz associação com base nas ferramentas disponíveis para cada usuário de um ambiente coorporativo. Porém, não existe uma garantia de que no momento em que um usuário tem ferramentas adicionadas ou removidas do seu perfil as mesmas sejam instaladas em sua estação, além do mesmo não oferecer suporte à atualização dinâmica.

O SOFA 2.0 [Bureš et al. 2008] é um modelo de componentes que trabalha com a atualização dinâmica dos componentes instalados em cada nó do ambiente distribuído através dos *SOFAnodes*. Os componentes ficam armazenados em repositório, porém não existe critério algum de qual componente, cada SOFAnode pode estar utilizando em sua estação.

A IDE Eclipse [Eclipse 2009], utiliza *plugins* em seu ambiente, no entanto, não é possível monitorar quais *plugins* cada usuário tem instalado em sua estação [Yap et al. 2005], além de existirem aqueles que necessitam de algum conhecimento específico do desenvolvedor para realizar sua instalação, caso não se encontre em um repositório.

A plataforma NetBeans [Netbeans 2009], também utiliza módulos de *software*, juntamente com um repositório que armazena-os para posterior utilização no ambiente. Além de não oferecer suporte à atualização dinâmica, esta plataforma não oferece alguma opção para que possam ser associados os módulos de *software* ao perfil de cada usuário.

Já o gerenciador de pacotes Synaptic [Synaptic 2009] foi desenvolvido para ser executado apenas em ambientes Linux. Apesar de fazer um ótimo gerenciamento das dependências entre os pacotes de *software* e, em alguns casos, atualizar o sistema sem que este necessite ser reiniciado, é necessário que o usuário da estação tenha privilégio de administrador da estação de trabalho, o que não é uma boa prática [Santos e Oliveira 2004], além de que qualquer usuário pode acessar o repositório de pacotes e instalar quaisquer pacotes disponíveis neste.

O Brechó [2]é um sistema de informação para a *Web* que fornece mecanismos de documentação, armazenamento, busca e recuperação de componentes [Fernandes et al. 2007]. Esse sistema é uma evolução do mecanismo de carga dinâmica do ambiente Odyssey[3] que foi projetado e implementado para prover uma solução mais eficiente e

[1] LDAP (*Lightweight Directory Access Protocol*): é um mecanismo que atua como suporte para a realização de atividades como nomeação, localização e segurança, dentre outras, relacionadas à gestão da infra-estrutura de recursos (usuários, impressoras, servidores, etc.) nas organizações [CRUZ et al., 2004].
[2] Projeto Brechó: In: http://reuse.com.ufrj.br/brecho
[3] Projeto Odyssey: In: http://reuse.cos.ufrj.br/odyssey

flexível para o mecanismo de carga do ambiente [Murta et al. 2004]. No entanto, é necessário que haja intervenção do usuário na instalação dos componentes, demandando assim por uma pessoa especializada envolvida no processo de atualização.

3. Contexto

Este trabalho está inserido no contexto do ambiente DiSEN (*Distributed Software Engineering ENvironment).* Este Ambiente Distribuído de Desenvolvimento de *Software* (ADDS), busca combinar técnicas, métodos e ferramentas para apoiar todas as atividades inerentes ao processo de construção de produtos de *software*, tais como gerência, desenvolvimento e controle da qualidade [Huzita et al. 2007]. .

A arquitetura do ambiente DiSEN proposta por Pascutti (2002), se apresenta em 3 camadas (camada de dinâmica, camada de aplicação e camada de infra-estrutura). Essa arquitetura é constituída por gerenciadores, suportes e um supervisor de configuração dinâmica que provê uma infra-estrutura para realização do desenvolvimento distribuído.

Posteriormente, Schiavoni (2007), propôs a criação de um *framework* denominado FRADE (*Framework to Infrastructure of Distributed Software Development Environment*), para especificar a infra-estrutura que capacita o ambiente a gerenciar a comunicação entre os diversos participantes.

O *framework* FRADE, apresentado na Figura 1, possui uma divisão lógica nas camadas de: aplicação, negócios, infra-estrutura e comunicação. A divisão em camadas possibilita representar as funcionalidades de um sistema em diferentes níveis de abstração. A opção pela arquitetura em camadas também foi feita devido à facilidade de manutenção.

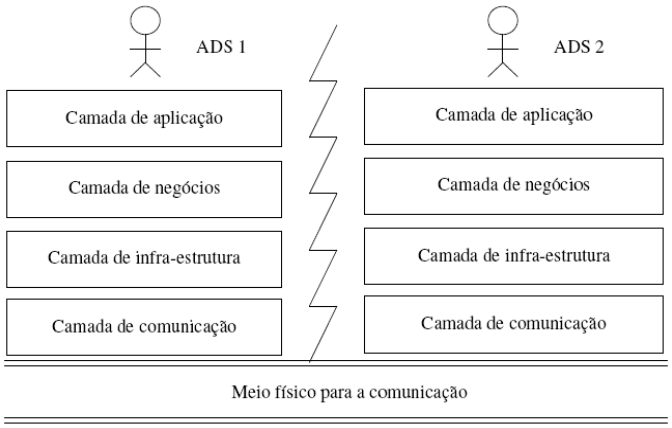

Figura 1: Arquitetura do *Framework* FRADE (SCHIAVONI, 2007).

4. Mecanismo de Atualização – DiSEN-*Updater*

O mecanismo de atualização DiSEN-*Updater* é constituído de um conjunto de componentes do DiSEN, com a finalidade de gerenciar todo o processo de atualização deste ambiente de uma forma dinâmica e transparente para o usuário, sem que haja a necessidade de qualquer tipo de intervenção do mesmo.

O DiSEN-*Updater* trata os pacotes de *software* que conterão as ferramentas e suas dependências como módulos de *software*. Um módulo contém um conjunto de

componentes de *software* e cada componente é um arquivo com extensão JAR (*Java ARchive*). Esses módulos contém ferramentas, que são disponibilizadas aos usuários do ambiente, e as bibliotecas que as mesmas necessitam para operar corretamente. As ferramentas que estão presentes nesses módulos são atualizadas dinamicamente dentro do ambiente.

Além de disponibilizar esses módulos, esse mecanismo é responsável por efetuar a atualização dinâmica desses no ambiente do usuário. Com isto**,** as dependências dos mesmos é gerenciada e, também, é garantido que o usuário utilize em seu ambiente, somente os módulos que foram definidos pelo seu gerente ou membro da equipe responsável.

A Figura 2, apresenta o mapeamento dos componentes do DiSEN-*Updater* no ambiente DiSEN os quais são descritos nas subseções a seguir:

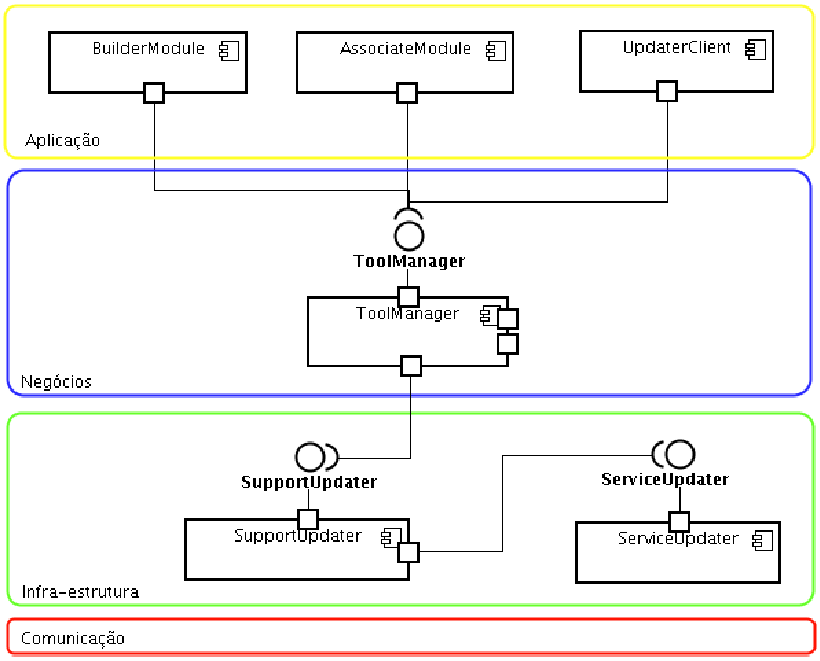

Figura 2: Mapeamento dos Componentes do DiSEN-*Updater* no ADDS DiSEN.

4.1. DiSEN-*BuilderModule*

O DiSEN-*BuilderModule* é um componente que fornece uma ferramenta para construir (*build*) módulos de *software* de uma forma simples e rápida. Para construir um novo módulo, o usuário tem a opção de selecionar: a) O arquivo descritor do módulo (arquivo contendo as informações básicas do módulo); b) O componente (Arquivo JAR) principal; c) Os arquivos JAR (bibliotecas externas) que o módulo necessita para funcionar corretamente; d) Os módulos dependentes desse módulo, que já estão disponíveis no repositório.

Depois que o usuário selecionar o conjunto de artefatos que comporão o novo módulo e confirmar a criação do mesmo na GUI, o DiSEN-*ToolManager* se incumbe de gerenciar o processo para que os dados desse módulo possam ser persistidos na base de dados.

4.2. DiSEN-*AssociateModule*

Após a construção dos módulos, com o auxilio do DiSEN-*BuilderModule*, é necessário que os módulos disponíveis no repositório sejam associados ao perfil de cada usuário do ambiente, o que é realizado pelo gerente de projetos. Para realizar essas associações, este deve ter disponível em sua instância do ambiente o DiSEN-*AssociateModule*.

O componente DiSEN-*AssociateModule*, localizado na camada de aplicação do ambiente DiSEN, provê uma GUI ao gerente possibilitando-o associar ao perfil de cada membro da equipe, os módulos que estes devem utilizar em sua instância. Este componente também garante que, no momento exato em que as alterações forem feitas no perfil do usuário, este será notificado de que seu ambiente será atualizado. O gerente é responsável por atribuir também um tempo para o usuário salvar as alterações nos artefatos que o mesmo está produzindo antes do seu ambiente ser atualizado.

4.3. DiSEN-*UpdaterClient*

O componente DiSEN-*UpdaterClient* está presente em todas instâncias do ambiente DiSEN. Ele fornece uma GUI para o usuário selecionar quais módulos ele deseja instalar em seu ambiente, porém somente os módulos de uso facultativo podem ser selecionados pelo usuário, isso porque é de responsabilidade do mecanismo de atualização gerenciar a instalação/remoção dos módulos obrigatórios.

Como o Synaptic [Synaptic 2009], este componente serve de GUI ao usuário, no entanto, o responsável por gerenciar todo o processos de verificação de dependências dos módulos e adição/remoção de ferramentas da GUI do usuário de forma dinâmica é DiSEN-*ToolManager*.

4.4. DiSEN-*ToolManager* (Gerenciador de Ferramentas)

O Gerenciador de Ferramentas está localizado na camada de negócios do ambiente DiSEN, cujas funcionalidades estão implementadas no componente DiSEN-*ToolManager*.

É responsabilidade de um Gerenciador de Ferramentas [Schiavoni 2007]: a) Carregar (*load*) e configurar as ferramentas na GUI do ambiente do usuário; b) Gerenciar a configuração no momento da instalação/remoção de uma ferramenta no ambiente do usuário; c) Disponibilizar as novas ferramentas a serem integradas ao ambiente instanciado e integrá-las ao *front-end* do usuário; d) Verificar se há dependências com outros módulos e resolver essas dependências antes de instalar ou remover uma ferramenta ou um conjunto destas, e disponibilizar a sua GUI no ambiente do usuário. Essas dependências podem ser tanto de outros módulos que já estão disponíveis no repositório, quanto de bibliotecas que essa ferramenta necessita para operar corretamente.

O DiSEN-*ToolManager* é responsável por intermediar as operações executadas pelo usuário com as ferramentas disponíveis na camada de aplicação com a infra-estrutura de atualização.

4.5. DiSEN-*SupportUpdater* (Suporte de Atualização)

Para a instalação, remoção e atualização das ferramentas do ambiente do usuário, é utilizado o suporte à atualização. O Suporte à Atualização, localizado na camada de infra-estrutura, provê a lista de módulos disponíveis no *workspace* do usuário e aqueles disponíveis no ambiente. O componente que implementa as funcionalidades do suporte à atualização é o DiSEN-*SupportUpdater*, que é responsável por instalar, remover e atualizar as ferramentas dinamicamente no ambiente.

O DiSEN-*SupportUpdater* também é responsável por administrar o *workspace* do usuário, fornecendo somente as ferramentas e suas dependências para que o DiSEN-*ToolManager* possa disponibilizar à GUI do usuário.

No momento que o DiSEN-*SupportUpdater* envia um inventário com a lista de módulos presentes no *workspace* do usuário ao DiSEN-*ServiceUpdater* (Serviço de Atualização), este serviço compara essa lista com os módulos que foram cadastrados para o seu uso (junto ao SGBD) e verifica quais módulos devem ser instalados ou removidos do ambiente do usuário.

Além de atualizar as estações de trabalho, o DiSEN-*SupportUpdater* é quem envia os novos módulos criados para o serviço de atualização (DiSEN-*ServiceUpdater*), que por sua vez realizará a persistência dos dados desse no banco de dados. Esse processo possibilita que a atualização do repositório de componentes seja efetuada pelo próprio ambiente.

A existência do DiSEN-*SupportUpdater* na infra-estrutura do DiSEN garante que a adição ou remoção de módulos seja realizada de forma transparente. Isso é possível devido ao uso da alteração dinâmica no carregador de classes da *Java Virtual Machine* (JVM). O carregador de classes (*class loader*) é responsável por localizar e importar os dados binários das classes.

Por instalar e remover os módulos dinamicamente na instância do ambiente do usuário, se garante a disponibilidade do serviço que essa instância esteja oferecendo.

4.6. DiSEN-*ServiceUpdater* (Serviço de Atualização)

O Serviço de Atualização (DiSEN-*ServiceUpdater*), localizado na camada de infra-estrutura do DiSEN, tem o papel principal no DiSEN-*Updater*. Todas as operações que são executadas no repositório são gerenciadas por esse serviço.

No ambiente DiSEN o *middleware* Sequoia é utilizado para o gerenciamento dos SGBDs [Silva e Huzita 2008]. O Sequoia é um *middleware* que oferece *clustering*, balanceamento de carga, serviços de gerenciamento de falhas, distribuição e replicação de dados entre várias estações, além de fornecer suporte para manutenção e recuperação de dados *on-line* [Continuent 2008]. Deste modo, a utilização do *middleware* Sequóia permite o armazenamento das ferramentas utilizadas no ambiente em repositórios distribuídos e, os problemas que ocorreriam durante a sincronização desses repositórios deixam de existir [Silva e Huzita 2008].

Quando o DiSEN-*SupportUpdater* requisita um novo módulo ao DiSEN-*ServiceUpdater*, este por sua vez, constrói o arquivo do módulo e o envia de volta como resposta da requisição. Esse arquivo contém: a) O componente do módulo (arquivo

JAR); b) Um diretório com as bibliotecas que este depende; c) Arquivo descritor do módulo.

O descritor do módulo é diferente do arquivo descritor informado pelo usuário através da GUI do DiSEN-*BuilderModule*. Este arquivo contém as informações básicas do módulo e, também, as informações sobre as dependências que o mesmo tem de outros módulos e bibliotecas (*external libs*), para que assim o DiSEN-*SupportUpdater* possa fazer o gerenciamento do *workspace* do usuário.

O que diferencia o DiSEN-*ServiceUpdater* de um repositório estático, é o fato deste possuir a característica de ter um serviço atuando de forma ativa, garantindo que o acesso às ferramentas disponíveis no repositório seja feito de forma centralizada, onde qualquer acesso a esse repositório do ambiente é feito por esse serviço.

Também se garante que qualquer usuário do ambiente terá um módulo instalado no seu ambiente somente se este usuário tiver permissão de utilizá-lo, permissão esta que é atribuída pelo seu gerente por meio do DiSEN-*AssociateModule*.

4.7. Características Similares e Vantagens do DiSEN-*Updater* em Relação aos Trabalhos Relacionados.

O DiSEN-*BuilderModule* está localizado na camada de aplicação do DiSEN, oferecendo apoio à construção de novos módulos para o ambiente, persistindo-os em um repositório. Essa característica de criação de módulos/*plugins*, é encontrada na plataforma NetBeans [Netbeans 2009] para a criação de módulos (.nbm), na IDE Eclipse [Eclipse 2009] para a criação de *plugins* e na biblioteca Brechó [Fernandes et al. 2007] para o gerenciamento do cadastro de novos componentes.

Somente no sistema Brechó, é efetuado o controle de qual usuário poder criar um componente e disponibilizá-lo para o ambiente. Já no NetBeans e no Eclipse qualquer usuário poderá criar um módulo ou um plug-in para o Eclipse. No ambiente DiSEN, somente os usuários autorizados, podem estar disponibilizando novos módulos ao ambiente. Para um novo módulo ser disponibilizado no ambiente é necessário que além de seu arquivo principal, exista um arquivo descritor e as bibliotecas que o mesmo necessita para o seu correto funcionamento. Por essa razão, o DiSEN-*Updater* trata esse conjunto de componentes como um único *módulo de software*.

O ambiente GATI apresentado por [Cruz et al. 2004], utiliza do serviço de controle de usuários existente no serviço de diretórios do LDAP, esse serviço fornece uma maneira de encontrar, identificar e controlar os usuários e recursos disponíveis na rede [Howes, Simith e Good 2003]. Por meio deste serviço, o GATI gerencia as ferramentas que cada usuário do ambiente utiliza em sua estação de trabalho.

Em um ADDS é necessário que seja feito esse mesmo controle de quais módulos cada usuário utiliza em seu ambiente, afim de garantir que um usuário utilize somente os módulos que foram associados ao seu perfil, evitando que um membro não autorizado da equipe tenha acesso a dados privilegiados que uma determinada ferramenta possa exibir. Para realizar esse controle no DiSEN, o DiSEN-*AssociateModule* é o componente que possibilita o gerente associar ao perfil de cada membro da equipe, quais módulos este deve utilizar.

Dentre os gerenciadores de pacotes para as distribuições Linux, muitos gerenciadores têm *front-ends* GUI, dentre eles, o Synaptic [Synaptic 2009]. Esses gerenciadores são, geralmente, apenas uma interface para as funcionalidades fornecidas pelos gerenciadores de linha de comando [Cappos et al. 2008], como o apt-get [Apt-get 2009]. O DiSEN-*UpdaterCliente* tem funcionalidade parecida com o Synaptic, no entanto, este componente provê uma GUI, para que o usuário selecione os módulos a serem instalados/removidos, sendo o DiSEN-*ToolManager* quem realmente faz o gerenciamento das dependências entre os módulos disponíveis no ambiente. Diferentemente da biblioteca Brechó, o gerenciamento da instalação dos módulos no ambiente DiSEN é feito de forma totalmente automatizada pelo DiSEN-*ToolManager*.

5. Conclusões

Este artigo apresentou o DiSEN-*Updater* e explorou as características de implementação em relação a atualização dinâmica de ferramentas em um Ambiente de Distribuído de Desenvolvimento de *Software* (ADDS).

Um dos maiores problemas de se atualizar uma instância do ambiente em um ADDS é conseguir garantir que durante o processo de atualização dessa instância não seja interrompido o fornecimento do serviço ao ambiente. Com o uso da técnica da atualização dinâmica é possível que o DiSEN-*Updater* atualize as ferramentas utilizadas no ambiente do usuário sem que o mesmo necessite parar as suas atividades ou, que ocorra a interrupção de algum serviço que esteja sendo fornecido ao ambiente.

O mecanismo de atualização dinâmica de ferramentas fornece o suporte para além de gerenciar o processo de atualização do ADDS, também fornece ferramentas a serem atribuídas a cada membro da equipe para que este utilize em sua estação.

Para o DiSEN-*Updater* está sendo estruturado um estudo de caso qualitativo, tendo em vista a necessidade de observar se as funcionalidades foram atendidas após a implementação dos componentes do mesmo. Anteriormente, o DiSEN não oferecia suporte para o gerenciamento do processo de atualização dinâmica das ferramentas do ambiente.

Referências Bibliográficas

Apt-get (2009). Debian APT toll. Disponível em: http://www.debian.org/doc/manuals/apt-howto. Acesso em Abril 2009.

Bureš, T.; Děcký, M.; Hnětynka, P.; Kofroň, J.; Parízek, P.; Plášil, F.; Poch, T.; Šerý, O.; Tůma, P. CoCoME in SOFA. Springer: CoCoME. p.388-417, 2008.

Cappos, J.; Samuel, J.; Baker, S.; Hartman, J. H. A Look In the Mirror: Attacks on Package Managers. In: CCS'08, Alexandria, Virginia, USA. October 27–31, 2008.

Chen, H.; Yu, J.; Chen, R.; Zang, B.; Yew, P. (2007) POLUS: A POwerful Live Updating System. In: 29th international conference on Software Engineering.

Continuent (2007), Middleware Sequoia. Disponível em: http://sequoia.continuent.org. Acesso em novembro de 2008.

Cruz, F. W., Santos, G. A., Medeiros, R. D. F., Pereira, L. A., Braga, M. C. Diener, R. Uma Ferramenta para a Administração de Serviços de Diretório Distribuídos

Baseados no OpenLDAP. In: 5o Fórum Internacional Software Livre, Porto Alegre, 2004.

Eclipse (2009). Eclipse.org. Disponível em: http://www.eclipse.org/. Acesso Jun/ 2009.

Fernandes, P. C. C., Prudencio, J. G. G., Marinho, A., Lopes, M. A. M., Murta, L. G. P., Werner, C. M. L. (2007) "Carga dinâmica de Componentes via Biblioteca Brechó". In: SBCARS, Sessão de Ferramentas, Campinas.

Fuks, H., Raposo, A. B., Gerosa, M. A. (2002) Engenharia de Groupware: Desenvolvimento de Aplicações Colaborativas, XXI Jornada de Atualização em Informática, Anais do XXII Congresso da Sociedade Brasileira de Computação.

Herbsleb, J. D., Moitra, D. (2001) "Guest Editors" Introduction: Global Software Development", IEEE Software, vol. 18, no. 2, pp. 16-20, March/April.

Howes, T. A., Simith. M. C., Good, G. S. (2003) Understanding and Deploying LDAP Directory Services, 2nd Edition, Addison Wesley.

Huzita, E. H. M., Tait, T. F. C., Colanzi, T. E., Quinaia. M. A. (2007) "Um Ambiente de Desenvolvimento Distribuído de Software – DiSEN". In: *Proceedings(CD)* SBES – I WDDS, João Pessoa.

Kiel, L. (2003) "Experiences in Distributed Development: A Case Study, In: Workshop on Global Software Development at ICSE 2003", Oregon, EUA.

Lüer, C. and Hoek, A. V. D. A. (2004) JPloy: User-Centric Deployment Support in a Component Platform. In *Proceedings* of CD 2004. p. 190-204.

Murta, L. G. P., Vasconcelos A., Blois, A. P., Lopes, M. A. M., Melo Jr C., Mangan M. A. S., Werner C. M. L. (2004) "Run-time Variability though Component Dynamic Loading". In: "Run-time Variability through Component Dynamic Loading". In: XVIII SBES, Sessão de Ferramentas, pp. 67-72, Brasília, DF, Brasil, Outubro.

NetBeans (2009). NetBeans.org. Disponível em: http://www.netbeans.org. Acesso em Fevereiro 2009.

Pascutti, M. C. D. (2002) "Uma proposta de arquitetura de um ambiente de desenvolvimento de software distribuído baseado em agentes". Dissertação (Mestrado) – Programa de Pós-Graduação em Computação, UFRGS, Porto Alegre.

Santos, M. T. and Oliveira, R. A. R. (2004) Distribuição de Software em Ambiente Corporativo para Plataforma Livre. 5º Fórum Internacional de Software Livre (FISL).

Schiavoni, F. L. (2007) "FRADE – Framework para infra-estrutura de um Ambiente Distribuído de Desenvolvimento de Software". Dissertação de Mestrado – Programa de Pós-Graduação em Ciência da Computação, UEM.

Silva. C. A.; Huzita, E. H. M. (2008) "DiSEN-SCV - Uma Estratégia para replicação de repositórios e alocação de artefatos". In: SBES - II WDDS, 2008, Campinas. Anais (cd) Workshop Desenvolvimento Distribuído de Software.

Synaptic (2009). Synaptic Package Manager. Disponível em: http://www.nongnu.org/synaptic. Acesso em Abril de 2009.