Adendo ao Relatório de Atividades

Período de Maio de 2000 a Abril de 2002

Francisco C. R. Reverbel

1 Introdução

Este documento, preparado a pedido do Conselho do Departamento de Ciência da Computação, fornece informações adicionais sobre os projetos de software aberto mencionados em meu Relatório de Atividades e sobre minha atuação nesses projetos. Foram-me especificamente solicitados esclarecimentos a respeito dos seguintes pontos:

- relevância dos projetos em si,
- relevância de minhas contribuições para esses projetos,
- meu grau de envolvimento com eles (dimensão dos trabalhos realizados).

O restante deste documento está organizado da seguinte maneira:

- As seções 2 e 3 apresentam alguns fatos sobre projetos de software aberto que são certamente conhecidos por muitos dos conselheiros, mas que talvez não sejam de conhecimento geral no Conselho.
- As seções 4 e 5 abordam respectivamente os projetos JBoss e JacORB, descrevendo minha participação em cada um deles. Estas seções levam em conta os parâmetros apresentados nas seções precedentes.
- A seção 6 conclui este documento e procura desfazer um mal-entendido que meu Plano de Trabalho parece ter causado.

2 Indicadores de Relevância de Projetos de Software Aberto

A relevância de um projeto de software aberto é geralmente associada a um conjunto de indicadores. O vínculo entre a relevância do projeto e os indicadores relacionados a seguir é fortemente sugerido pela análise, *a posteriori*, dos projetos de software aberto mais influentes e bem sucedidos, como Linux, Apache, GNU C/C++, GNU Emacs, Perl, PostgreSQL, FreeBSD, Samba, Sendmail e Bind.

Comunidade de participantes ativa e geograficamente dispersa. Ao longo de sua vida um projeto relevante tende a atrair um grupo de participantes ativos, com capacidade e disponibilidade para oferecer contribuições efetivas. O critério da dispersão geográfica reduz a probabilidade dos participantes terem sido motivados por fatores não necessariamente relacionados com a relevância do projeto. Ele elimina projetos cujos participantes são todos funcionários de uma mesma empresa (que pode ter interesse específico no projeto) ou alunos de um mesmo programa de pós-graduação (possivelmente orientados por um professor que participa do projeto), etc.

Dimensão do projeto. Projetos relevantes tendem a ter certo porte. O número de linhas de código fonte é em geral maior que cem mil e freqüentemente maior que um milhão.

Usuários. Todo projeto relevante tem usuários. A existência de usuários, além de gerar realimentação vital para o ciclo evolutivo do projeto, indica que ele atende a uma necessidade real e que ultrapassou o estado de protótipo. Tipicamente não existe uma contabilização precisa dos usuários de um programa aberto. O volume de usuários é inferido a partir de outros indicadores, como número de downloads e tráfego em listas de usuários. A publicação e comercialização de livros descrevendo o software desenvolvido (freqüentemente escritos por participantes do projeto) e a existência de estruturas comerciais de apoio aos usuários são também fatores relacionados com o número de usuários.

Tráfego em listas de discussão. Todo projeto aberto sério tem pelo menos uma lista de discussão na Internet. As mensagens enviadas para a lista são mantidas num repositório público e acessível a quem quer que seja. Muitos projetos tem mais de uma lista de discussão. Um arranjo típico utiliza uma lista para participantes do projeto, exclusivamente voltada para a discussão de aspectos internos do software coletivamente construído, e outra lista para os usuários desse software. Projetos relevantes têm um certo tráfego de mensagens em suas listas de discussão. Este tráfego varia de projeto para projeto, de dezenas de mensagens por mês até centenas de mensagens por dia.

Existência de literatura. A partir de um certo ponto de sua vida um projeto relevante se torna assunto de livros cujo mercado potencial é o conjunto de usuários do software produzido.

Comercialização de serviços de apoio. A partir de um certo ponto da vida de um projeto relevante surgem empresas que comercializam serviços de apoio aos usuários do software desenvolvido.

De acordo com esses indicadores, um projeto de software aberto pode ser considerado relevante mesmo que não esteja alinhado com o estado da arte em sua área. (O Linux é freqüentemente enquadrado nessa categoria, por não empregar uma arquitetura tipo microkernel.) Como não contempla a questão da inovação tecnológica, esse conjunto de parâmetros é certamente incompleto para a avaliação da relevância de atividades de desenvolvimento de software aberto desenvolvidas num ambiente acadêmico. Ele deve ser complementado por

uma análise do grau de inovação e das contribuições para o estado da arte presentes no software desenvolvido.

3 Atribuição de Créditos em Projetos de Software Aberto

O modo como os participantes de um projeto de software aberto se organizam internamente varia bastante de projeto para projeto. Praticamente todos os projetos bem sucedidos adotaram como organização social alguma forma de meritocracia. O mérito de cada participante é baseado no valor de suas contribuições para o projeto. O funcionamento da organização repousa portanto sobre o reconhecimento coletivo (que necessariamente inclui uma avaliação) da contribuição de cada um.

A correta atribuição de créditos pelas contribuições dos participantes é uma condição imprescindível para a harmonia interna da estrutura social do projeto. É extremamente difícil uma apropriação indevida do crédito pelo trabalho realizado por outrem. Na improvável hipótese de que tal apropriação ocorra, é virtualmente impossível que ela passe despercebida. Além disso, num projeto com certa importância, mesmo a cessão voluntária de crédito (prática muito difícil de se eliminar num ambiente acadêmico tradicional) é inviabilizada pelo fato dos trabalhos serem realizadas em público. A história completa das modificações nos arquivos fonte é mantida num repositório acessível a todos os participantes (e a toda e qualquer pessoa, em muitos projetos). Esse repositório é utilizado diariamente pelos membros ativos do projeto. Em muitos projetos o repositório de arquivos fonte é configurado para enviar automaticamente uma mensagem informativa a todos os participantes sempre que algum arquivo for modificado. Tais mensagens contém os nomes dos arquivos modificados, as alterações efetuadas e o participante responsável pelas alterações.

A quase totalidade dos projetos tem pelo menos duas formas básicas de reconhecimento pelas contribuições de seus participantes:

Direito de escrita no repositório de arquivos fonte. É a forma mais elementar de reconhecimento. Um "*committer*" é um participante que tem o direito de efetivar alterações ("*commit*") no repositório de fontes do projeto. Novos participantes somente se tornam *committers* depois de demonstrarem um certo nível (que varia bastante entre projetos) de dedicação e de competência. Do ponto de vista de segurança, é indesejável a existência de *committers* que não estejam efetivamente contribuindo para o projeto. Por esse motivo, muitos projetos adotam alguma política de revogação do direito de escrita de participantes inativos.

Divulgação do nome do participante. Os nomes dos participantes são geralmente publicados na página do projeto, com grau de destaque proporcional à importância de suas contribuições.

Além dessas formas elementares de reconhecimento, projetos com uma estrutura organizacional mais complexa adotam esquemas mais elaborados para definir algum tipo de diferenciação hierárquica entre seus participantes.

4 O projeto JBoss

4.1 Descrição Geral

O projeto JBoss (`http://www.jboss.org/`), iniciado em março de 1999, produziu um servidor de aplicações baseado no padrão *Java 2 Enterprise Edition* [7] e evolui na direção de um *Web Operating System* [1]. Um servidor de aplicações atua como um *container* no qual usuários implantam componentes [6], que correspondem a aplicações ou a partes de aplicações executadas num ambiente distribuído. Por meio de um mecanismo de invocação remota de métodos, o servidor de aplicações disponibiliza para clientes remotos os serviços dos componentes nele implantados. Além disso, o servidor oferece a esses componentes um conjunto de serviços básicos, como persistência, transações e segurança.

Todo o software produzido pelo JBoss é disponibilizado livremente, sob a *GNU Library General Public License*. Pacotes com "versões oficiais" destinadas aos usuários são publicados em `http://sourceforge.net/project/showfiles.php?group_id=22866`. O repositório de trabalho do JBoss, contendo todos os arquivos fonte gerados ao longo da vida do projeto, é acessível (para leitura) a toda e qualquer pessoa e pode ser visualizado em `http://cvs.sourceforge.net/cgi-bin/viewcvs.cgi/jboss/`.

4.2 Relevância Tecnológica

A tecnologia de servidores de aplicações é hoje o que há de mais avançado em termos de *middleware*. Praticamente todos os grandes fornecedores de software investiram nessa área e hoje comercializam produtos desse tipo a preços na casa dos milhares de dólares por CPU. Embora tais produtos já estejam no mercado (geralmente no topo da linha de sistemas de software de cada fornecedor), a atividade na área continua muito intensa. Quem estiver acompanhando os seminários de Sistemas de Software[1] oferecidos por este departamento no presente semestre deve ter notado que dois dos seminários (dentre os quatorze previstos até o final do semestre) abordaram sistemas desse tipo: o Interstage da Fujitsu e o Websphere da IBM, ambos competidores do JBoss.

A competição tecnológica entre os fornecedores de servidores de aplicações (grandes empresas, em sua maioria) é muito forte. Mesmo assim, o JBoss tem se destacado como o servidor pioneiro em recursos e inovações como "implantação quente" (*hot deployment*), *proxies* dinâmicos, gerenciamento dinâmico (e possivelmente remoto), carga remota do próprio servidor e, finalmente, acesso a um mesmo componente através de múltiplos protocolos (JRMP, IIOP, SOAP). Sua recente vitória no *2002 JavaWorld Editors' Choice Awards*[2], na categoria "*Best Java Application Server*" (tradicionalmente disputada apenas por sistemas comerciais) parece ter sido motivada em parte pelas inovações que introduziu e em parte por sua popularidade entre os usuários de servidores de aplicações. Na avaliação da JavaWorld, o JBoss superou os sistemas desenvolvidos por empresas como IBM (um dos três finalistas), BEA Systems (o outro finalista), Borland, Computer Associates, Fujitsu, Hewlett-Packard, Hitachi, Iona, Macromedia, NEC, Oracle, Sun Microsystems e Sybase.

[1] A programação de seminários está disponível em `http://www.ime.usp.br/~rcastro/seminars/`.

[2] Vide `http://www.javaworld.com/javaworld/jw-03-2002/jw-0326-awards-p3.html`.

4.3 Relevância como Projeto de Software Aberto

4.3.1 Comunidade de participantes

A comunidade de participantes do JBoss está distribuída por vários continentes (predominantemente América do Norte, Europa e Oceânia, com representação bem menor da Ásia e da América do Sul) e inclui seis doutores: Marc Fleury, Scott Stark, Bela Ban, Gerald Brose, Christoph Jung e Francisco Reverbel. Ela é estruturada da seguinte maneira:

- Um grupo de *committers*, hoje com 57 pessoas, tem o direito de alterar o repositório de fontes do projeto. A relação atualizada de *committers* está disponível em `http://sourceforge.net/project/memberlist.php?group_id=22866`.
- A página de créditos do JBoss (`http://www.jboss.org/developers/team.jsp`) inclui os novos participantes, que ainda não tem o status de *committers*, bem como os ex-participantes, que deixaram de ser *committers*.
- Os *core developers* são um subconjunto dos *committers*, com aproximadamente 30 pessoas, que aparecem em destaque na página de créditos do JBoss. Os *core developers* que deixaram o projeto permanecem na página de créditos, com o mesmo destaque, sob a legenda "*retired*".
- O *JBoss Group*, um subconjunto dos *core developers* com aproximadamente 15 pessoas, é o núcleo responsável pela arquitetura geral e pelas principais decisões do projeto.
- O JBoss tem um líder (Marc Fleury) e um vice-líder (Scott Stark).

4.3.2 Dimensão do projeto

A versão 3.0 do JBoss tem pouco mais de meio milhão de linhas de código Java. A tabela abaixo mostra a distribuição dessas linhas pelos diferentes módulos do projeto:

admin	9.163
blocks	901
catalina	4.236
cluster	23.550
common	22.704
connector	15.590
console	913
iiop	15.381
j2ee	8.040
jboss.net	6.327
jetty	90.420
jmx	63.218
management	12.773
messaging	39.730

naming	2.944
pool	2.558
security	15.191
server	87.152
system	11.340
testsuite	66.204
varia	6.094
total	504.429

4.3.3 Usuários

O software do JBoss é disponibilizado através do sítio SourceForge, que mantém estatísticas imparciais de downloads. Conforme essas estatísticas[3], nos últimos seis meses o número mensal de downloads do JBoss oscilou entre 78.445 e 207.970. Esses números sugerem que o JBoss é o servidor de aplicações com base de usuários mais ampla. A já mencionada vitória do JBoss no *2002 JavaWorld Editors' Choice Awards* parece estar relacionada com a amplitude da base de usuários do sistema e com sua excelente aceitação por parte dos usuários.

4.3.4 Tráfego em listas de discussão

O JBoss tem um conjunto de foros de discussão para usuários (cada fórum é voltado para um assunto específico, como projeto de componentes, persistência, etc.), uma lista de discussão para os participantes do projeto e outra lista para distribuição de mensagens de alteração no repositório de arquivos fonte. Tanto os foros como as listas de discussão estão publicamente disponíveis em `http://www.jboss.org/forums.jsp`. O volume de tráfego nesse conjunto de foros e listas é da ordem de cem mensagens por dia.

Além dessas listas, que são abertas, há também uma lista de discussão fechada, exclusiva para os membros do JBoss Group. Como a maior parte das discussões técnicas ocorre na lista aberta, o tráfego na lista fechada é baixo.

4.3.5 Literatura

O livro [5] descreve detalhadamente a versão 2.4 do JBoss. A arquitetura de microkernel do JBoss é descrita no último capítulo do livro [3].

4.3.6 Disponibilidade de serviços de apoio a usuários

O JBoss Group LLC, uma empresa sediada em Atlanta, comercializa serviços de apoio a usuários do JBoss.

[3] Vide `http://sourceforge.net/project/stats/index.php?report=months&group_id=22866`.

4.4 Minhas Contribuições

Participo do projeto JBoss como *committer*, *core developer*, membro do JBoss Group e responsável pelo sub-projeto IIOP.

4.4.1 O JBoss/IIOP

O sub-projeto IIOP gerou o módulo que converte o JBoss num servidor de aplicações CORBA. O módulo JBoss/IIOP permite que componentes implantados no JBoss recebam invocações remotas de métodos através do protocolo padronizado por CORBA, o *Internet Inter-ORB Protocol*. Os serviços dos componentes implantados no servidor JBoss ficam portanto acessíveis a clientes escritos nas várias linguagens de programação suportadas por CORBA. A relevância do módulo JBoss/IIOP advém do fato dele permitir a utilização do JBoss em ambientes distribuídos heterogêneos. Uma descrição sucinta desse módulo está disponível em `http://www.jboss.org/developers/projects/jboss/iiop.jsp`.

No projeto do JBoss/IIOP o maior desafio foi transformar o JBoss num servidor de aplicações CORBA sem abrir mão das características de facilidade de uso que o tornaram popular entre os desenvolvedores de componentes. O JBoss/IIOP não requer um compilador IDL nem etapas adicionais para geração de stubs e esqueletos. A geração de stubs IIOP é automaticamente efetuada em tempo de implantação de um componente. Clientes escritos em Java podem carregar remotamente stubs IIOP a partir do servidor JBoss. A carga remota de stubs IIOP é dinâmica e transparente para o código dos clientes. Ela se baseia em informações de *codebase* embutidas nas referências CORBA exportadas pelo JBoss/IIOP.

O módulo JBoss/IIOP inclui ainda um serviço de nomes CORBA e um repositório de interfaces. Ambos rodam na mesma máquina virtual que o servidor JBoss. As interfaces *home* dos componentes são registradas com o serviço de nomes. O repositório de interfaces trata chamadas `_get_interface_def()` sobre as interfaces dos componentes.

Hoje o módulo JBoss/IIOP tem pouco mais de quinze mil linhas de código Java. Aproximadamente metade desse módulo foi escrita por Ole Husgaard, um engenheiro de software que atua como consultor independente na Dinamarca. Eu escrevi a outra metade. O código fonte do JBoss/IIOP pode ser visto na Internet, usando um navegador, a partir da URL `http://cvs.sourceforge.net/cgi-bin/viewcvs.cgi/jboss/contrib/iiop/`. Todos os arquivos fonte incluem a informação de autoria.

O sub-projeto IIOP requereu ainda um certo número de alterações (algumas delas não triviais) em outros módulos do JBoss. Tais alterações foram efetuadas por mim.

4.4.2 Outras contribuições

Além de liderar o sub-projeto IIOP e escrever parte do módulo JBoss/IIOP, participei de decisões mais gerais do projeto. Neste tipo de atividade, considero que minha contribuição mais relevante foi na definição de uma arquitetura de "múltiplos invocadores" para os componentes implantados no servidor JBoss. Essa arquitetura, que defini em conjunto com Marc Fleury e Bill Burke, permite que um mesmo componente receba invocações remotas através de diferentes protocolos: JRMP, IIOP e SOAP.

5 O projeto JacORB

5.1 Descrição Geral

O JacORB é uma implementação livre do padrão CORBA do OMG, escrita em Java e voltada para o ambiente Java. Distribuído sob a *GNU Library General Public License*, este software foi desenvolvido com o apoio da *Freie Universität Berlin*, por Gerald Brose e outros.

5.2 Relevância Tecnológica

CORBA é o padrão para interoperabilidade entre aplicações distribuídas em ambientes heterogêneos. Esse padrão define um mecanismo de invocação remota de métodos que oferece transparência de localização e independência de plataformas de hardware e software, bem como independência das linguagens usadas para se implementar o objeto chamador e o objeto chamado.

5.3 Relevância como Projeto de Software Aberto

Sendo mais antigo que o JBoss, o JacORB já é mencionado na literatura como exemplo de projeto de software aberto bem sucedido [4].

5.3.1 Comunidade de participantes

O JacORB foi desenvolvido por um grupo de dezessete pessoas, distribuídas por vários países, com predominância de alemães e americanos. A relação de participantes se encontra em `http://www.jacorb.org/authors.html`. Gerald Brose é o fundador e líder do projeto.

5.3.2 Dimensão do projeto

A versão 1.4 do JacORB tem pouco mais 140 mil linhas de código Java. Isso não inclui definições de interfaces IDL, nem tampouco código Java mecanicamente gerado a partir de interfaces IDL.

5.3.3 Usuários

Não existem estatísticas de downloads do JacORB. Uma relação parcial de usuários se encontra em `http://www.jacorb.org/customers.html`. Essa relação é certamente muito incompleta, pois nela somente aparecem os usuários que solicitaram sua inclusão. Nota-se que o JacORB é utilizado como base para vários projetos de pesquisa acadêmicos, inclusive no Brasil (na Universidade Federal de Santa Catarina).

Aqui no IME-USP, o JacORB tem sido usado nos trabalhos práticos da disciplina Sistemas de Objetos Distribuídos. Essa disciplina está sendo oferecida no presente semestre, pelo professor Fábio Kon. Pelo menos um dos trabalhos práticos deste semestre[4] empregou o JacORB.

[4]Vide `http://www.ime.usp.br/~kon/MAC5759/EP1.html`.

5.3.4 Tráfego em listas de discussão

O JacORB tem uma lista de discussão cujas mensagens são publicamente arquivadas em `http://lists.spline.inf.fu-berlin.de/pipermail/jacorb-developer/`. Essa lista é voltada para os usuários do sistema. O tráfego é da ordem de uma centena de mensagens por mês.

5.3.5 Literatura

Como o JacORB é uma implementação fiel do padrão CORBA, não faz muito sentido a existência de livros específicos sobre ele. Textos sobre programação CORBA em Java geralmente mencionam o JacORB. Um exemplo é o livro [2], cujo primeiro autor é o líder do projeto JacORB. Esse texto está sendo utilizado no IME-USP neste semestre, pela disciplina Sistemas de Objetos Distribuídos.

5.3.6 Disponibilidade de serviços de apoio a usuários

Três empresas comercializam serviços de apoio aos usuários do JacORB:

- Object Computing International (OCI), USA.
- PrismTechnologies, UK.
- Huihoo, China.

5.4 Minhas Contribuições

Minha participação no JacORB foi motivada por uma necessidade do projeto JBoss: precisávamos de um ORB que pudesse ser livremente distribuído como parte de nosso servidor de aplicações e que implementasse completamente certas características de CORBA essenciais para o JBoss. Embora o JacORB fosse o ORB que mais se aproximasse do preenchimento de nossas necessidades, ele ainda não tinha uma característica crucial para o JBoss: a capacidade de transmitir e receber *RMI valuetypes* (objetos Java passados por valor de acordo com as convenções do sistema Java RMI).

A colaboração se iniciou com discussões sobre como se deveria implementar *RMI valuetypes*. Participei ativamente dessas discussões e sugeri à equipe do JacORB a forma de implementação de *RMI valuetypes* que foi adotada. Depois de ter contribuído ativamente nos testes e na depuração dessa implementação, verifiquei que ela ainda não contemplava um aspecto do tratamento de *RMI valuetypes* desejável para o JBoss: a possibilidade de se carregar remotamente o código das classes de *RMI valuetypes*, possibilidade essa sugerida mas não requerida pela especificação CORBA. Escrevi então código Java para implementar isso.

Fiz boa parte desse trabalho me considerando um colaborador eventual do projeto, mas os demais participantes não viram a situação da mesma forma. Num certo ponto eles resolveram me dar direito de escrita no repositório de fontes do JacORB, provavelmente cansados de aplicar ao repositório as modificações que volta e meia eu submetia.

Como meu trabalho no JacORB envolveu principalmente modificações e adição de código a arquivos fonte previamente existentes, não sei quantificar minha contribuição em termos de linhas de código.

Minha atuação teve ainda o efeito de aproximar as equipes dos dois projetos (JBoss e JacORB), levando Gerald Brose a participar também do JBoss.

6 Conclusão

Gostaria de concluir este documento retificando um mal-entendido causado pela utilização infeliz, em meu Plano de Trabalho, de um termo ("*paperware*") que talvez tenha melindrado o parecerista anônimo. Usei esse termo sem a intenção de "condenar a produção de artigos descrevendo somente protótipos ou projetos de sistemas de software, em vez de sistemas reais e de utilidade comprovada", como entendeu o parecerista. O que eu disse é que "o risco de se produzir somente *paperware* diminuirá bastante caso se consiga de alguma forma vincular a pesquisa em sistemas de software ao desenvolvimento de sistemas reais." É evidente que protótipos e projetos ("*paperware*") tem grande importância para a pesquisa científica e não faz nenhum sentido condenar sua produção. Mas considero altamente desejável que se produza também algo além de *paperware*. Não acho que esse tipo de consideração deva ser aplicada à produção individual de um pesquisador (longe de mim condenar alguém que não tem as mesmas prioridades que eu), mas à comunidade de sistemas de software como um todo. Nesse sentido, tenho a pretensão de acreditar que estou contribuindo para o desempenho global da comunidade brasileira de sistemas de software com um trabalho que muito poucos estão fazendo. Seria o caso de se refletir se esse tipo de trabalho é desejável ou não. Caso se conclua que é, seria também o caso de se pensar em formas de avaliação que não prejudiquem as pessoas interessadas em desenvolver atividades sérias dessa natureza.

Referências

[1] William J. Bolosky, Richard P. Draves, Robert P. Fitzgerald, Christopher W. Fraser, Michael B. Jones, Todd B. Knoblock, and Rick Rashid. Operating System Directions for the Next Millennium. Microsoft Research, 1999. Available at `http://research.microsoft.com/research/sn/Millennium/mgoals.html`.

[2] Gerald Brose, Andreas Vogel, and Keith Duddy. *Java Programming with CORBA: Advanced Techniques for Building Distributed Applications, 3rd ed.* Wiley, 2001. ISBN 0471376817.

[3] Juha Lindfors, Marc Fleury, and The JBoss Group. *JMX: Managing J2EE Applications with Java Management Extensions*. Sams, 2002. ISBN 0672322889.

[4] Douglas C. Schmidt and Adam Porter. Leveraging open-source processes to improve the quality and performance of open-source software. In *1st Workshop on Open Source Software Engineering*, ICSE 23, May 2001.

[5] Scott Stark, Marc Fleury, and The JBoss Group. *JBoss Administration and Development*. Sams, 2002. ISBN 0672323478.

[6] Sun Microsystems. *Enterprise JavaBeans Specification, Version 2.0*, August 2001.

[7] Sun Microsystems. *Java 2 Platform Enterprise Edition Specification, v1.3*, July 2001.

São Paulo, 5 de junho de 2002

Francisco C. R. Reverbel