UNIVERSIDADE DO VALE DO RIO DOS SINOS – UNISINOS
UNIDADE ACADÊMICA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA INTERDISCIPLINAR DE PÓS-GRADUAÇÃO
EM COMPUTAÇÃO APLICADA
NÍVEL MESTRADO

VILSON CRISTIANO GÄRTNER

UM AMBIENTE INTEGRADO PARA APOIO AO DESENVOLVIMENTO DISTRIBUÍDO DE SOFTWARE

SÃO LEOPOLDO
2011

Vilson Cristiano Gärtner

UM AMBIENTE INTEGRADO PARA APOIO AO DESENVOLVIMENTO DISTRIBUÍDO DE SOFTWARE

Dissertação apresentada como requisito parcial para a obtenção do título de Mestre pelo Programa de Pós-Graduação em Computação Aplicada da Universidade do Vale do Rio dos Sinos.

Orientador: Prof. Dr. Sérgio Crespo Coelho da Silva Pinto

SÃO LEOPOLDO

2011

G244i Gärtner, Vilson Cristiano

Um ambiente integrado para apoio ao desenvolvimento distribuído de software / Vilson Cristiano Gärtner. – 2011.
161 f.

Dissertação (mestrado) – Universidade do Vale do Rio dos Sinos, Unidade Acadêmica de Pesquisa e Pós-Graduação, Programa Interdisciplinar de Pós-Graduação em Computação Aplicada, São Leopoldo, 2011.

Orientação: Sérgio Crespo Coelho da Silva Pinto

1. Software 2. Desenvolvimento distribuído de software. 3. Ambiente integrado de desenvolvimento. 4. IDE. I. Título
CDU: 004.4

Ficha catalográfica elaborada por Maristela Hilgemann Mendel – CRB-10/1459

Vilson Cristiano Gärtner

UM AMBIENTE INTEGRADO PARA APOIO AO DESENVOLVIMENTO DISTRIBUÍDO DE SOFTWARE

Dissertação apresentada como requisito parcial para a obtenção do título de Mestre em Computação Aplicada, pelo Programa de Pós-Graduação em Computação Aplicada da Universidade do Vale do Rio dos Sinos – UNISINOS.

Aprovado em: _____/_____/_____.

BANCA EXAMINADORA

_____

Prof. Dr. Sérgio Crespo Coelho da Silva Pinto – Unisinos / RS

_____

Prof. Dr. Jorge Luis Victória Barbosa – Unisinos / RS

_____

Prof. Dr. Ismar Frango – Mackenzie / SP

Dedico esta Dissertação de Mestrado a minha esposa Simone,
aos meus filhos Alann e Lucca e demais familiares.

AGRADECIMENTOS

A realização de um mestrado não é apenas um trabalho acadêmico individual, que exige muito esforço e dedicação ao estudo, pesquisa e elaboração da dissertação. É uma etapa que exige amor, compreensão e apoio de muitas outras pessoas. Nesse sentido, quero agradecer:

A minha amada esposa Simone pelo seu apoio e incentivo incondicionais em todos os momentos e por compreender as privações decorrentes do mestrado. Pelo seu amor, companheirismo e pelo suporte que tem sido para mim desde que nos conhecemos, há vinte e dois anos.

Aos meus queridos filhos Alann e Lucca, pelo seu amor, carinho e por proporcionarem os mais felizes dias de minha vida. Apesar de menos frequentes nos últimos anos, nada me anima e fortalece mais do que os momentos em que nos divertimos juntos.

Aos meus pais Silvio e Noeli, pela sua dedicação, educação e exemplos de vida, fundamentais para a formação do meu caráter.

Aos meus irmãos Jairo e Tatiana, pelo seu carinho e apoio em todos os momentos, e pelas felizes recordações da infância.

À família Kuhn, por compreenderem a minha ausência durante o período de realização do mestrado.

Ao meu professor orientador, Dr. Sérgio Crespo, pela sugestão do tema e por sua ajuda e dedicação durante a realização deste trabalho.

Aos meus colegas de mestrado, pelos ótimos momentos, de estudo e confraternizações, compartilhados nos últimos dois anos.

Aos participantes dos estudos de caso, fundamentais para a conclusão deste trabalho.

À CAPES pelo apoio financeiro.

Aos professores e funcionários do PIPCA e a todos aqueles que, de uma ou outra forma, contribuíram para a realização desta dissertação.

“Aprenda com o ontem, viva o hoje, tenha esperança no amanhã.
O importante é não parar de questionar.”
Albert Einstein

RESUMO

O Desenvolvimento Distribuído de *Software* (DDS) é um modelo de desenvolvimento que vem se intensificando nos últimos anos. Também conhecido como Desenvolvimento Global de *Software* (DGS), esse modelo de desenvolvimento é realizado por equipes em diferentes localizações geográficas.

Entre os fatores que contribuem para esse aumento, está a necessidade de negócio das corporações, que buscam redução de custos, recursos qualificados e necessidade de uma presença global. Em outros casos, se deve ao surgimento de novos movimentos de desenvolvimento de *software*, como a comunidade de *software* livre, um exemplo bem sucedido de DDS.

Apesar da necessidade ou mesmo da conveniência de desenvolver o *software* de forma distribuída, é extremamente difícil fazê-lo com sucesso. A separação física traz uma série de problemas e desafios interessantes que recém estão começando a ser compreendidas: questões estratégicas, questões culturais, comunicação inadequada, gestão do conhecimento, alocação de tarefas, confiança, questões técnicas, entre outros.

Desde que surgiu, o DDS mudou grande parte da tradição do desenvolvimento de *software* e, para manter o seu mercado, as organizações não podem depender das mesmas competências e tecnologias de engenharia de *software* utilizadas no desenvolvimento interno.

Nesse sentido, o presente trabalho tem por objetivo auxiliar na redução dos problemas e dificuldades trazidos por esse modelo de desenvolvimento, através da implementação de um ambiente de desenvolvimento cujas ferramentas foram definidas com base em estudos e trabalhos relacionados ao tema.

Palavras-Chave: Desenvolvimento Distribuído de Software. DDS. Desenvolvimento Global de Software. DGS. Ambiente Integrado de Desenvolvimento. IDE.

ABSTRACT

The Distributed Software Development (DSD) is a development model that has been intensified in recent years. Also known as Global Software Development (GSD), this development model is done by teams in different geographical locations.

Among the factors that have contributed to this increase, there is the corporations' business need of seeking ways to reduce costs, seeking skilled resources and having a global presence. In other cases, it is due to the emergence of new movements in software development, such as the free software community, a successful example of DSD.

Despite the need or even desirability of developing software in a distributed way, it is extremely difficult to do this successfully. Physical separation has a number of interesting problems and challenges that are just beginning to be understood: strategic issues, cultural issues, inadequate communication, knowledge management, task allocation, trust, technical issues, among others.

DSD has changed much of the tradition of software development since it appeared. Organizations cannot rely on the same skills and software engineering technologies used internally to maintain this new market.

In this way, this work aims to help to reduce the problems and difficulties brought by this type of development, through the implementation of a software development environment whose tools were defined based on studies related to the topic.

Keywords: Distributed Software Development. DSD. Global Software Development. GSD. Integrated Development Environment. IDE.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

ABNT: Associação Brasileira de Normas Técnicas

API: *Application Programming Interface*

ASCII: *American Standard Code for Information Interchange*

ATA: Adaptador de Telefone Analógico

CASE: *Computer Aided Software Engineering*

CMM: *Capability Maturity Model*

CT-SE: *Collaborative Technology to Support Software Engineering*

CVS: *Concurrent Version System*

CVW: *Collaborative Virtual Workspace*

DAS: Desenvolvimento Adaptativo de *Software*

DDS: Desenvolvimento Distribuído de *Software*

DGS: Desenvolvimento Global de *Software*

DSDE: *Distributed Software Development Environment*

DSDM: *Dynamic Systems Development Method*

DXP: *Dispersed Extreme Programming*

FDD: *Feature Driven Development*

HTTP: *Hypertext Transfer Protocol*

IDE: *Integrated Development Environment*

IHC: Interação Homem-Computador

IM: *Instant Message*

IP: *Internet Protocol*

IRC: *Internet Relay Chat*

ISO: *International Standards Organization*

JMF: Java *Media Framework*

MSN: Microsoft *Service Network*

NAT: *Network Address Translation*

OSS: *Open Source Software*

PSTN: *Public Switched Telephone Network*

RCS: *Revision Control System*

SCM: *Software Configuration Manager*

SIP: *Session Initiation Protocol*

SMTP: *Simple Mail Transfer Protocol*

SVN: Subversion

TCP: *Transmission Control Protocol*

TLS: *Transport Layer Security*

UA: *User Agent*

UAC: *User Agent Client*

UAS: *User Agent Server*

UDP: *User Datagram Protocol*

URI: *Uniform Resource Identifier*

URL: *Uniform Resource Locator*

USB: *Universal Serial Bus*

VIMEE: *Virtual Distributed Meeting Tool*

VOIP: *Voice Over IP*

VPN: *Virtual Private Network*

WWW: *World Wide Web*

XML: *eXtensible Markup Language*

XMPP: *Extensible Messaging and Presence Protocol*

XP: *Extreme Programming*

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos, tem aumentado significativamente o número de empresas que estão adotando o modelo de Desenvolvimento Distribuído de *Software* (DDS) ou Desenvolvimento Global de *Software* (DGS). Segundo Herbsleb e Grinter (1999), o desenvolvimento de *software* geograficamente distribuído tornou-se uma necessidade de negócio para muitas corporações globais. Isso se deve à necessidade de investimentos em outros países para a localização de *softwares*, aquisição de empresas, além da promessa de desenvolvimento de *software* 24 horas por dia.

O DDS tem causando um grande impacto no mercado e na maneira como os produtos de *software* têm sido modelados, construídos, testados e entregues para o cliente. Inicialmente relatado por Carmel (1999), um dos maiores desafios que o ambiente de negócio apresenta, do ponto de vista do processo de desenvolvimento de *software*, é trabalhar com DDS.

Entre os fatores que tem influenciado o crescimento de DDS, está a disponibilidade de mão de obra e custo mais baixo; maior acessibilidade e evolução dos recursos de telecomunicação; revolução das ferramentas de desenvolvimento; necessidade de existência de recursos globais disponíveis a qualquer hora; proximidade com o mercado local; criação de equipes virtuais para atender às oportunidades de mercado; desenvolvimento quase contínuo (conhecido por *round-the-clock*), utilizando as vantagens relativas aos fusos horários diferentes.

Por outro lado, apesar da necessidade e até mesmo da conveniência do desenvolvimento distribuído de *software*, é extremamente difícil de fazê-lo com sucesso, conforme já havia sido apontado por Herbsleb e Grinter (1999). Problemas de coordenação acontecem frequentemente e levam a atrasos, ineficiência e frustração.

O DDS está se tornando uma norma na indústria de *software* [Damian e Moitra, 2006], e criou uma nova classe de problemas a serem resolvidos pelos pesquisadores da área de desenvolvimento de *software* [Prikladnicki *et al*, 2004]. Porém, projetos dessa natureza trazem dificuldades como diferenças culturais entre integrantes das equipes, problemas de comunicação, alocação de tarefas, confiança, entre outros.

1.1 Motivação

De acordo com Sangwan *et al.* (2006), uma vez havendo uma distância física entre as equipes que estão envolvidas no desenvolvimento de um mesmo *software*, existirão desafios complexos e interessantes que recém estão começando a ser compreendidas.

Para Carmel e Tjia (2005), a colaboração possui um papel fundamental no desenvolvimento de *software*. Os autores afirmam que uma organização terá mais benefícios das tecnologias colaborativas se utilizar um conjunto delas: mensagem instantânea, videoconferência, serviços de áudio de qualidade, repositórios integrados e ambientes de compartilhados.

Cheng *et al*. (2003) acreditam que as ferramentas colaborativas até podem ser utilizadas paralelamente ao IDE, todavia, a integração num único ambiente traz grandes recompensas como: redução do desgaste no processo de desenvolvimento, melhoria no senso de contexto, além de contribuir para a rastreabilidade entre artefatos de colaboração e artefato de código.

Dessa forma, a principal motivação deste trabalho está em possibilitar que equipes distribuídas possam trabalhar colaborativamente no mesmo projeto, modificando o mesmo código fonte de programas e utilizando ferramentas pensadas para atender demandas desse modelo de desenvolvimento de *software* e integradas num mesmo ambiente.

1.2 Questão de Pesquisa

O Desenvolvimento Distribuído de *Software* traz uma série de novos desafios para o cenário de desenvolvimento de *software*. Conforme já mencionado, são desafios como: diferenças culturais entre os integrantes das equipes, problemas de comunicação, problemas na alocação de tarefas, confiança entre as pessoas, entre outros.

A questão de pesquisa central procura responder ao seguinte questionamento: É possível definir mecanismos de Coordenação, Comunicação e Controle para o Desenvolvimento Colaborativo e Distribuído de *software* de forma que um ambiente computacional possa ser integrado a ferramentas de mercado já existentes?

1.3 Objetivos do Trabalho

Este trabalho tem por objetivo a criação de um ambiente integrado e distribuído que auxilie no processo de desenvolvimento de *software* por equipes dispersas geograficamente.

Com o uso das ferramentas propostas neste ambiente, espera-se contribuir para a diminuição dos problemas e dificuldades trazidos pelo DDS, maximizando os benefícios do desenvolvimento colaborativo.

Para isso, este trabalho tem como objetivos específicos:

- Avaliar as tecnologias mais apropriadas para o desenvolvimento das ferramentas do ambiente;
- Criar o ambiente de desenvolvimento, com as seguintes ferramentas:
  - Edição simultânea de código por diversos usuários;
  - Adição de marcações visuais ao texto alterado por usuários remotos, indicando os autores das modificações;
  - Integração com controle de versões e suporte a tarefas comuns na programação como: realce de sintaxe e marcação de erros no código fonte, compilação, depuração e execução do programa;
  - Comunicação através de mensagens de texto com suporte à tradução das mensagens;
  - Comunicação através de voz;
  - Gerenciamento distribuído de tarefas;
  - Controle distribuído de compromissos/agenda;
  - Transferência de arquivos entre os usuários.
- Criar um protocolo de mensagens em formato XML, que será utilizado na comunicação entre os ambientes;
- Garantir que o ambiente seja multiplataforma, possibilitando o seu uso em diferentes sistemas operacionais;
- Manter a arquitetura do ambiente expansível, possibilitando a adição de novas ferramentas e funcionalidades, posteriormente, através do uso de *plugins*;
- Possibilitar a interoperabilidade do ambiente com outros programas e dispositivos, permitindo a interação e o acesso as suas funcionalidades e informações.

1.4 Metodologia

Para alcançar os objetivos traçados na seção 1.3, foi estabelecida a metodologia de trabalho a seguir descrita:

- Definição do problema e objeto de pesquisa da presente dissertação;
- Realização de revisão bibliográfica e análise de estudos já realizados na área, para compreensão do problema;
- Juntamente com a revisão bibliográfica, será realizado um estudo de trabalhos que buscam implementar soluções relacionados ao tema;
- Avaliação e escolha do ferramental tecnológico mais adequado para a implementação do ambiente proposto nesta dissertação;
- Criação de protótipos para validação das tecnologias selecionadas, principalmente as relacionadas à comunicação;
- Implementação da solução proposta, utilizando o ferramental escolhido;
- Elaboração de cenários para a realização de estudos de caso reais, para a verificação e validação da solução desenvolvida.

1.5 Organização da Dissertação

Além deste capítulo de introdução, o presente trabalho possui mais cinco capítulos, organizados da seguinte maneira:

- Capítulo 2: Fundamentação Teórica – apresenta a revisão bibliográfica, abordando conceitos relacionados ao trabalho, como o desenvolvimento de *software*, engenharia de *software* e o desenvolvimento distribuído, descrevendo suas características e desafios.
- Capítulo 3: Trabalhos Relacionados – apresenta quatro trabalhos com assuntos relacionados ao domínio do problema e que implementam soluções que buscam contribuir para esse modelo de desenvolvimento de *software*;
- Capítulo 4: Trabalho Proposto – apresenta detalhadamente as características e organização do ambiente desenvolvido, descrevendo as ferramentas implementadas, sua arquitetura, funcionamento e tecnologias utilizadas;
- Capítulo 5: Estudos de Caso – apresenta a utilização da ferramenta em quatro diferentes cenários e realiza a análise dos resultados obtidos com base em avaliação feita pelos participantes;
- Capítulo 6: Conclusões – descreve considerações finais acerca deste trabalho, analisando os resultados alcançados ao seu término e apresenta indicações de trabalhos futuros a serem desenvolvidos.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo traz uma visão geral de assuntos importantes para o contexto do trabalho, como: desenvolvimento de *software*, engenharia de *software*, processo de *software* e desenvolvimento ágil de *software*. Além disso, oferece uma análise detalhada sobre o desenvolvimento distribuído de *software*, suas características, desafios e casos de sucesso.

2.1 Desenvolvimento de *Software*

Nos dias atuais, o *software* é uma tecnologia importante no palco mundial e indispensável para a área de negócios, engenharia, ciência e atividades do cotidiano. Afeta praticamente todos os aspectos da vida das pessoas, e permite a criação de novas tecnologias, além da extensão de tecnologias já existentes. Está embutido em sistemas de toda a espécie: telecomunicação, industrial, militar, entretenimento, médico, transporte, entre outros. Um *software* de computador é um produto construído, e posteriormente mantido ao longo do tempo, por profissionais conhecidos como engenheiros de *software* [Pressman, 2010]. Para construir um *software*, utiliza-se uma abordagem de engenharia de *software*. Ele é obtido da mesma forma como se constrói qualquer produto bem sucedido, aplicando-se um processo ágil e adaptável que leva a um resultado de alta qualidade e que satisfaz às necessidades de quem vai utilizar o produto.

A importância do *software* de computadores tem passado por significativas mudanças em pouco mais de 50 anos. Nesse sentido, à medida que a importância do *software* vem crescendo, têm sido desenvolvidas, pela comunidade de *software*, tecnologias que facilitem, agilizem e tornem menos dispendiosas a construção e a manutenção de programas de computador de alta qualidade. Essas tecnologias podem ser voltadas para domínios específicos, como, por exemplo, projeto e implementação de sites *web*, enfocar determinado domínio tecnológico como sistemas orientados a objetos ou programação orientada a aspectos, ou ainda ser de uma base mais ampla, como por exemplo, sistemas operacionais como o Linux[1]. Todavia, ainda é necessário desenvolver uma tecnologia de *software* que faça isso tudo, mas é bem pequena a probabilidade que ela surja no futuro [Pressman, 2010].

---

[1] Linux: http://www.linuxfoundation.org

2.2 Engenharia de *Software*

De acordo com Sommerville (2003), a engenharia de *software* é uma disciplina da engenharia preocupada com todos os aspectos da produção de *software*, desde a especificação do sistema, que acontece no estágio inicial, até a sua manutenção, quando ele está em operação. De acordo com essa definição, existem duas fases importantes:

- "Disciplina da engenharia" – os engenheiros aplicam teorias, métodos e ferramentas nas situações apropriadas e de modo seletivo, para fazer com que os produtos funcionem. Estão sempre em busca da solução para os problemas, mesmo quando inexistem teorias aplicáveis e métodos de apoio. Precisam trabalhar, ainda, buscando soluções que estejam dentro de eventuais restrições financeiras e organizacionais;
- "Todos os aspectos da produção de *software*" – além de se dedicar aos processos técnicos de desenvolvimento de *software*, a engenharia de *software* também se dedica a atividades como o gerenciamento de projetos e o desenvolvimento de métodos, teorias e ferramentas que deem apoio à produção de *software*.

Para Pressman (2010), os engenheiros de *software* adotam uma metodologia organizada e sistemática em seu trabalho, uma vez que essa é, normalmente, a maneira mais eficaz de produzir *software* de alta qualidade. Porém, a engenharia tem a ver também com a questão de selecionar o método mais apropriado, e, em algumas circunstâncias, pode ser mais eficaz utilizar uma abordagem mais informal e criativa. Um exemplo de desenvolvimento no qual uma abordagem informal é mais apropriada, é um sistema de comércio eletrônico, que requer uma combinação de habilidades de *software* e projetos gráficos.

Atualmente, existem algumas amplas categorias de *software* que apresentam desafios contínuos para os engenheiros de *software* [Pressman, 2010]:

- ***Software* de sistemas** – trata-se de uma coleção de programas escritos para servir a outros programas. Por exemplo, compiladores, editores e utilitários para o gerenciamento de arquivos, que processam estruturas complexas, porém determinadas; componentes de sistemas operacionais, *software* de rede e processadores de telecomunicações, que processam dados amplamente indeterminados;

- ***Software* de aplicação** – consiste de programas isolados e que tratam de uma necessidade específica do negócio. Essas aplicações processam dados técnicos ou comerciais a fim de facilitar as operações ou gestão, e tomadas de decisões do negócio. O *software* de aplicação é usado para controlar, em tempo real, funções de negócio a exemplo do controle do processo de fabricação e processamento de transações no ponto-de-venda. Além disso, também é usado nas aplicações convencionais de processamento de dados;
- ***Software* científico e de engenharia** – as aplicações desses *softwares* vão da astronomia à vulcanologia, da biologia molecular à manufatura automatizada, da análise automotiva de tensões à dinâmica orbital do ônibus espacial. Antigamente caracterizado por algoritmos que processam números, as aplicações modernas na área científica de engenharia estão se afastando desses algoritmos convencionais. Começaram a adquirir características de tempo real e até de *software* de sistemas, com projeto apoiado por computadores, simulação de sistemas e outras aplicações interativas;
- ***Software* embutido** – é usado para implementar e controlar características e funções para o usuário final e para o próprio sistema, e reside dentro de um produto ou sistema. O *software* embutido pode fornecer função significativa e capacidade de controle, como por exemplo, funções digitais como controle de combustível, mostradores do painel e sistemas de frenagem de um automóvel, ou então, pode realizar funções muito limitadas e particulares, como por exemplo, controlar o teclado de um forno microondas;
- ***Software* para linha de produtos** – projetado para ser usado por muitos clientes diferentes, pode focalizar um mercado limitado e especial, como por exemplo controle de estoque, ou então estar direcionado ao mercado de consumo de massa, como por exemplo planilhas, processamento de texto, entretenimento, aplicações financeiras, gerenciamento de banco de dados;
- **Aplicações da *web*** – estas aplicações cobrem uma vasta gama de aplicativos. Podem ser desde um conjunto de arquivos ligados por hipertexto, contendo texto e gráficos até aplicativos mais elaborados como aplicações de comércio eletrônico. No caso de aplicações mais elaboradas, elas não fornecem apenas funções de computação e conteúdo ao usuário final, mas estão integradas ao banco de dados da empresa e orientadas as suas regras de negócio;

- ***Software* para inteligência artificial** – para resolver problemas complexos que não são passíveis de computação ou análise direta, este tipo de *software* faz uso de algoritmos não-numéricos. Essa área inclui aplicações de robótica, reconhecimento de imagem e voz, sistemas especialistas, redes neurais, jogos, entre outros;
- **Computação Ubíqua** – a verdadeira computação distribuída pode tornar-se realidade logo, graças ao rápido crescimento de redes sem fio. O desenvolvimento de sistemas e *softwares* de aplicação que possibilitem a comunicação entre computadores pessoais, pequenos dispositivos e sistemas empresariais através de grande rede, são desafios para os engenheiros de *software*;
- **NetSourcing** – a *world wide web* está se transformando rapidamente em um fornecedor de conteúdo e também em uma ferramenta de computação. Arquitetar desde aplicações simples para uso pessoal até aplicações sofisticadas que forneçam benefícios para um mercado global, se constituem no desafio para os engenheiros de *software*;
- ***Software* aberto** – é uma tendência crescente, permite que o código-fonte seja distribuído e modificado pelos clientes. Neste caso, o desafio dos engenheiros é a construção de códigos-fonte que sejam auto-descritivos, além de desenvolver técnicas que possibilitem ao cliente e outros desenvolvedores saber o que foi modificado e qual o seu impacto no *software*;
- ***Software* para a "nova economia"** – muitas pessoas de negócio acreditavam que a nova economia estava morta, graças à insanidade ponto-com que tomou os mercados financeiros no final da década de 1990 e o surto que continuou no início dos anos 2000. Todavia, a nova economia está viva e muito bem, porém sua evolução será lenta, caracterizada pela comunicação e distribuição em massa. Dessa forma, o desafio para os engenheiros de *software* é a construção de aplicações que facilitem a comunicação e a distribuição de produtos em massa, usando conceitos que estão recém se formando.

2.3 Processo de *software*

Um produto de *software* é obtido por meio de um conjunto de atividades e resultados associados, conhecidos como processo de *software*. Em sua maioria, essas atividades são executadas por engenheiros de *software* [Sommerville, 2003].

Figura 1: Engenharia de *software* em camadas [Pressman, 2010]

Pressman (2010) afirma que a engenharia de *software* é uma tecnologia em camadas, apresentadas na Figura 1. Segundo ele, a camada de processo é o alicerce da engenharia de *software*. Argumenta ainda que o processo permite o desenvolvimento racional e oportuno de *software*, mantendo unidas as camadas de tecnologia. O processo define um *framework* que deve ser utilizado para efetiva utilização da tecnologia de engenharia de *software*. Os processos de *software* são a base para o controle de projetos de *software*, estabelecendo o contexto no qual: os marcos são estabelecidos; os métodos técnicos são aplicados; a qualidade é assegurada; os modelos, documentos, dados, relatórios, formulários e demais produtos de trabalho são produzidos; as modificações são geridas.

Todos os processos de *software* possuem quatro atividades comuns e que são fundamentais: especificação de *software*, projeto e implementação de *software*, validação do *software* e evolução do *software* [Sommerville, 2003].

2.4 Desenvolvimento Ágil

A engenharia de *software* ágil combina um conjunto de diretrizes de desenvolvimento com uma filosofia, que propõe a satisfação do cliente e a entrega incremental do *software* logo de início. Também são características da filosofia: equipes pequenas e altamente motivadas, utilização de métodos informais, simplicidade no desenvolvimento e uso mínimo de engenharia de *software*. A comunicação ativa e contínua entre desenvolvedores e clientes, e a entrega em substituição à análise e ao projeto, são premissas das diretrizes de desenvolvimento. Numa equipe ágil, a comunicação e a colaboração entre todos são

essenciais e o trabalho é realizado de forma conjunta entre engenheiros e demais *stakeholders*[2]. Essa equipe é auto-organizada e controla o seu próprio destino [Pressman, 2010].

Para que um processo de *software* possa ser considerado como processo ágil, deverá considerar três aspectos [Pressman, 2010]:

1. Tanto os requisitos de *software* quanto as prioridade do cliente serão modificados à medida que o projeto avança. É difícil prever quais requisitos vão persistir e quais serão modificados, bem como prever quais prioridades mudarão;
2. As atividades de projeto e construção são intercalados em muitos tipos de *software*, devendo ser realizadas juntas e os modelos de projeto devem ser comprovados à medida que são criados. Nesse sentido, antes de iniciar a construção é difícil prever quanto de projeto será necessário;
3. Não é possível prever as atividades de análise, construção e testes da forma como se gostaria.

Existem diferentes modelos ágeis de processo, havendo semelhanças tanto nas suas filosofias quanto nas práticas, mas todos seguem o princípio de desenvolvimento ágil de *software* [Pressman, 2010]:

- ***Extreme Programming*** **(XP)** – o paradigma deste modelo de desenvolvimento é a orientação a objetos, e possui um conjunto de regras e atividades que ocorrem no contexto de quatro atividades (Figura 2): planejamento, projeto, codificação e teste [Pressman, 2010]. Segundo [Sommerville, 2003] o XP é uma evolução da abordagem de desenvolvimento incremental.
- A proposta do XP que rege o desenvolvimento do *software* é: entregar ao cliente o *software* que ele deseja e no prazo que ele quiser. O trabalho é feito numa equipe única, todos dedicados a entregar um *software* com qualidade

[2] *Stakeholders*, segundo Kotonya e Sommerville (1998), são pessoas ou organizações que serão afetados pelo sistema e que tem influência direta ou indireta nos requisitos do mesmo. Podem ser usuários finais do sistema, gerentes e outros envolvidos nos processos influenciados pelo sistema, engenheiros responsáveis pelo desenvolvimento e manutenção, clientes da organização ou mesmo órgãos reguladores externos e autoridades certificadoras.

[Audy e Prikladnicki, 2008]. Os principais fatores do XP são: coragem, simplicidade, comunicação e *feedback*;

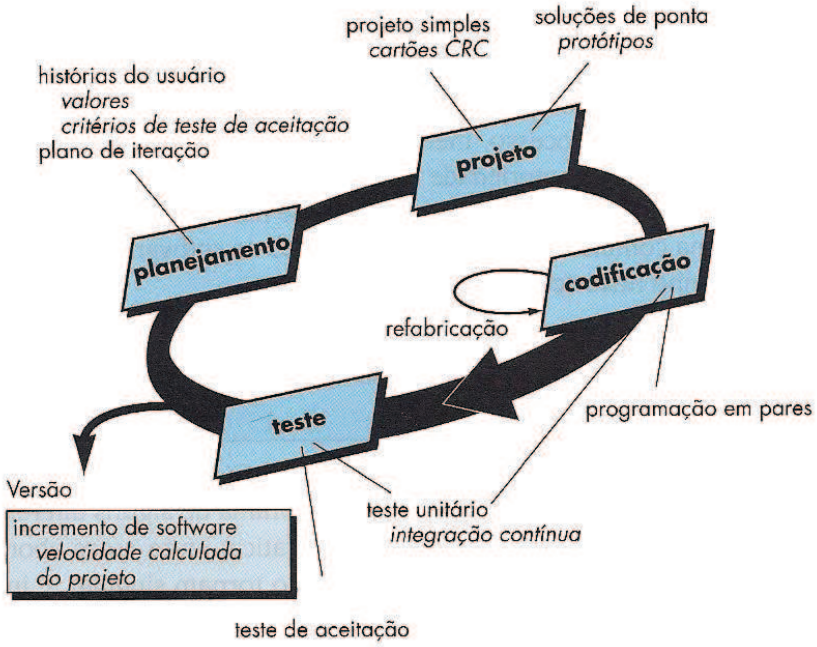


Figura 2: O processo XP [Pressman, 2010]

- **Método de Desenvolvimento Dinâmico de Sistemas** – o DSDM (*Dynamic Systems Development Method*) utiliza a prototipagem incremental em um ambiente controlado de projeto que possibilita construir e manter sistemas com restrições de prazos. Essa abordagem ágil sugere que, a cada iteração, apenas parte do trabalho seja necessário para que haja um avanço para o próximo incremento. O restante do trabalho será completado quando mais requisitos forem conhecidos ou quando solicitações forem requisitadas. As atividades e ciclo de vida deste modelo são: estudo de viabilidade, estudo do negócio, iteração do modelo funcional, iteração de projeto e construção, implementação;
- **Desenvolvimento Adaptativo de *Software* (DAS)** – o DAS foi proposto como uma técnica para a construção de *softwares* e sistemas complexos, apoiado na colaboração humana e auto-organização da equipe. Este método define um ciclo de vida com três fases: especulação, colaboração e aprendizado;
- **Scrum** – as atividades de desenvolvimento propostos pelo *Scrum* são: requisitos, análise, projeto, evolução e entrega. Esse conjunto de "padrões de processo de *software*" se mostraram efetivos para projetos de requisitos

mutantes, com prazos apertados e críticas ao negócio de desenvolvimento. Em cada um desses padrões são realizadas as atividades (Figura 3): Pendência, *Sprints* e Reuniões *Scrum*. Pendência: consiste criação de uma lista de requisitos com suas prioridades, sendo que o gerente de produto pode reavaliar os requisitos e redefinir as prioridades. *Sprint*: são unidades de trabalho, definidos para satisfazer requisitos num determinado prazo, normalmente 30 dias. Reuniões *Scrum*: são reuniões diárias de curta duração, geralmente de 15 minutos, feitas pela equipe. Nela, três perguntas são feitas e respondidas por todos os membros da equipe: "O que você fez desde a última reunião", "Que obstáculos você está encontrando?" e "O que você planeja realizar até a próxima reunião de equipe?";

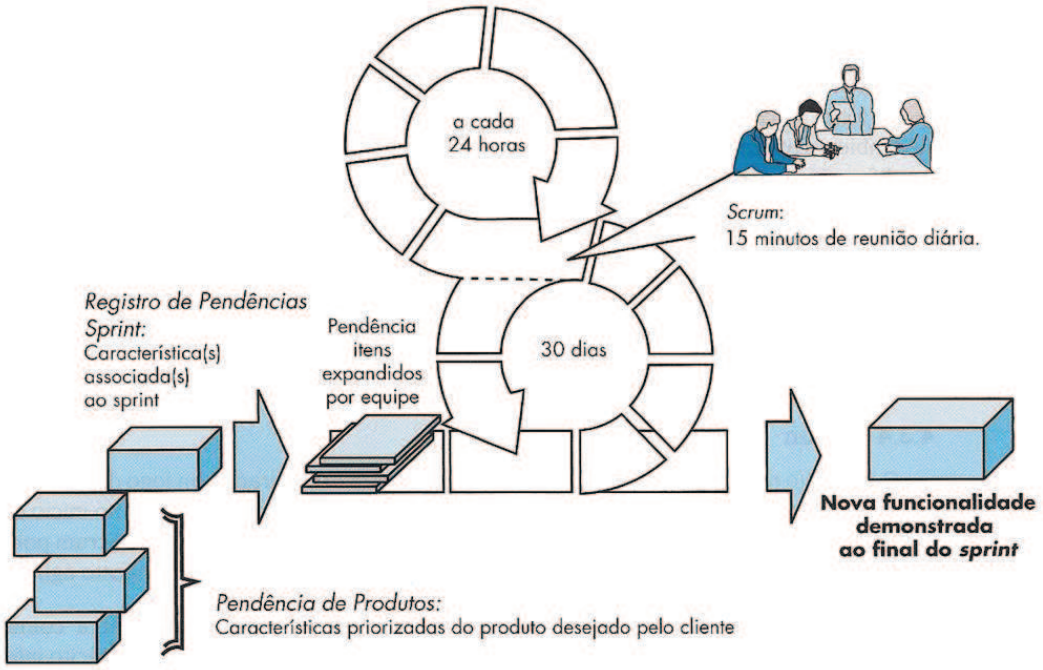


Figura 3: Fluxo de processos Scrum, adaptada de [Pressman, 2010]

- **Desenvolvimento Guiado por Características** – Originalmente concebido como um modelo prático de processo para engenharia de *software* orientada a objetos, o FDD (*Feature Driven Development*) foi melhorado e estendido para ser um processo ágil e adaptativo, passível de ser aplicado a projetos de *software* de tamanho moderado a grande. No FDD, as características são pequenos blocos de funcionalidades, o que facilita a sua descrição pelo

usuário. Da mesma forma, suas representações de projeto e código são mais fáceis de inspecionar. Cada característica é um incremento de *software* passível de entrega. A cada duas semanas a equipe desenvolve características operacionais, que podem ser organizadas em um agrupamento hierárquico, relacionado ao negócio. A hierarquia de características guia o planejamento de projeto, cronograma e monitoramento.

2.5 Desenvolvimento Distribuído de *Software*

O *software* se tornou um componente vital em praticamente qualquer negócio. Cada vez mais, o sucesso depende do uso do *software* como uma ferramenta competitiva [Herbsleb e Moitra, 2001]. Por outro lado, historicamente, a demanda por serviços de *software* tem ultrapassado a disponibilidade de pessoas que os executam. À medida que os computadores proliferam, a demanda por *software* continua a aumentar de forma mais rápida que a disponibilidade de profissionais [Karolak, 1998].

Nas últimas décadas tem-se presenciado uma tendência estável e irreversível em direção à globalização dos negócios, e de modo particular nos negócios de alta tecnologia em *software*. Forças econômicas estão transformando mercados nacionais em mercados internacionais, e criando novas formas de competição e cooperação que ultrapassam as fronteiras de uma nação. Esta mudança tem causado um profundo impacto não somente em marketing e distribuição, mas também na forma como os produtos são concebidos, projetados, construídos, testados e entregues aos clientes [Herbsleb e Moitra, 2001].

Em poucas palavras, Desenvolvimento Distribuído de *Software* (DDS) ou Desenvolvimento Global de *Software* (DGS) pode ser definido como um modelo de desenvolvimento que é realizado por equipes em diferentes localizações geográficas. Em alguns casos, essas equipes podem ser da mesma organização e, em outros, podem haver colaborações ou *outsourcing*[3], que envolvem organizações diferentes [Sangwan *et al*, 2006].

Até o início dos anos 80, aproximadamente 70 a 80 por cento dos programas eram produzidos nos Estados Unidos, e a maioria das necessidades eram atendidas por pessoas de lá. Na metade da década de 1990, a demanda por esses profissionais cresceu de tal forma que não haviam mais recursos para atendê-la (Figura 4). Em função disso, as companhias

[3] *Outsourcing*: quando o desenvolvimento do *software*, ou parte dele, é feito por uma outra empresa [Lamersdorf *et al,* 2009].

passaram a disputar esses profissionais e os custos aumentaram exponencialmente. Como recursos equivalentes surgiram em outros países, o trabalho começou a migrar para fora da empresa. Na essência, a oferta e a procura estavam determinando os custos e, consequentemente, direcionando o desenvolvimento para fora da empresa. As empresas têm considerado economicamente atrativa a terceirização ou mesmo o desenvolvimento conjunto de *software*, e o mercado para a terceirização de *software* fora dos Estados Unidos está estimado entre 200 milhões de dólares e 50 bilhões de dólares [Karolak, 1998].

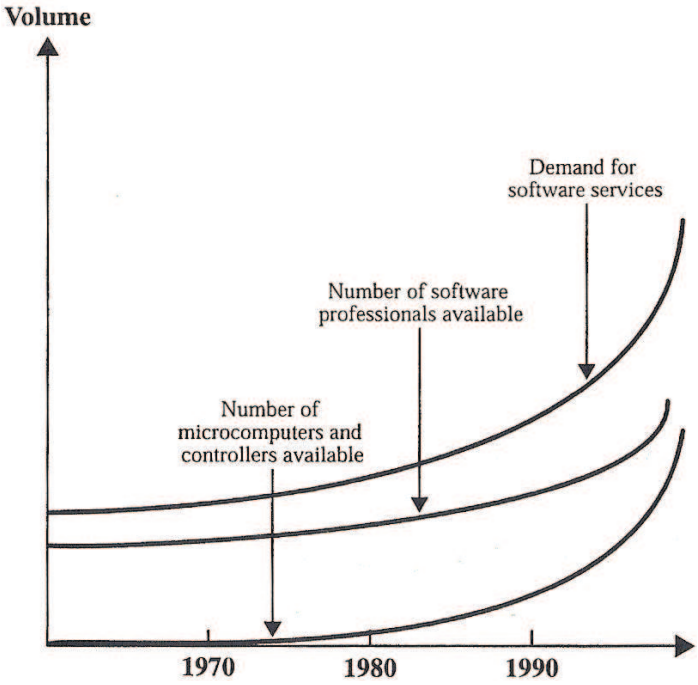


Figura 4: Demanda por profissionais de *software* versus aumento de computadores [Karolak, 1998]

Segundo Herbsleb e Moitra (2001), muitas organizações iniciaram experimentos com o desenvolvimento de *software* em instalações remotas e com terceirização, em busca de redução de custos e recursos qualificados. Vários fatores aceleraram esta tendência como, por exemplo, a necessidade de capitalizar sobre recursos globais e usar recursos escassos, independente de sua localização, com sucesso e de forma competitiva; vantagens de negócio proporcionadas pela proximidade do mercado, incluindo o conhecimento dos consumidores e condições locais, bem como a boa vontade demonstrada pelos investidores locais; a rápida formação de corporações virtuais e equipes virtuais, para explorar as oportunidades do mercado, também foi fator importante; a necessidade de se ter a flexibilidade de capitalizar

em fusões e aquisições; e a pressão para melhorar o tempo de colocação do produto no mercado utilizando-se as diferenças de fuso horário no desenvolvimento "*round-the-clock*".

Em nível executivo e gerencial, Carmel (1999) cita como fatores catalisadores do Desenvolvimento Global de *Software* (DGS): procura por talento especializado em *software*; a colaboração provocada pelas aquisições globais; busca pela redução de custos; a necessidade da companhia de ter uma presença global; necessidade de redução do tempo para colocação do produto no mercado; e a necessidade de estar mais próximo do cliente. Outros fatores importantes foram: o emergente rigor no desenvolvimento necessário em virtude da distância; a atualização interna criada pela diversidade; a habilidade de focar, trazida pela separação; a base de experiência real que os novos sites ganharam; e o novo quadro gerencial de defensores do DGS que influenciam as decisões nas empresas de *software*.

2.5.1 Fatores que contribuem para o DDS

Segundo Carmel e Tjia (2006), existem seis fatores que convergiram e geraram este fenômeno (Figura 5):

- **Globalização do comércio de serviços**: bem conhecida, a globalização do comércio e, mais recentemente, globalização do comércio de serviços, que está chegando a 2 trilhões de dólares. A abertura das fronteiras iniciou em 1980 à medida que soluções baseadas no mercado ganharam aceitação. O colapso do bloco Soviético impulsionou ainda mais esse processo;
- **Clima favorável aos negócios**: nações que eram adversas, ou na melhor das hipóteses indiferentes, agora estão competindo para atrair investimentos estrangeiros e impulsionar seus setores de *software*. Países oferecem incentivos fiscais e aliviam as regulamentações governamentais;
- **Crescimento da força de trabalho**: com o passar do tempo, tem aumentado o número de engenheiros formados nas universidades e colégios técnicos. No caso da China, por exemplo, anualmente são graduados quatro vezes mais engenheiros que nos Estados Unidos. Num passado, esses profissionais emigravam em busca de trabalho, mas graças às empresas globais, atualmente eles podem permanecer onde estão;
- **Custos de Comunicação**: em menos de dez anos, o custo com a comunicação praticamente desapareceu, e sem limite de uso. Isso gerou um resultado notável, de forma que é quase tão fácil trabalhar com alguém no outro lado do mundo, como é

trabalhar com alguém em outra cidade. No final dos anos 1990 e início de 2000, as taxas de ligações internacionais caíram entre 80% e 90%. Além disso, muitos profissionais da área de *software* utilizam VoIP a custos beirando a zero;

- **Comoditização do *Software***: não é bem entendido por aqueles que estão fora da indústria de *software*, e se refere à uniformização das práticas e ferramentas de desenvolvimento. Pela primeira vez em 50 anos de história do *software*, algumas tarefas estão suficientemente rotinizadas e automatizadas que se tornaram commodities. Essas tarefas praticamente não se diferenciam de um fornecedor para outro, e tem um baixo custo;
- **Diferencial de salários**: esta é, sem dúvida nenhuma, a força dominante. As diferenças salariais levam a custos mais baixos, e as pressões de custo transformaram o desenvolvimento distribuído numa necessidade estratégica.

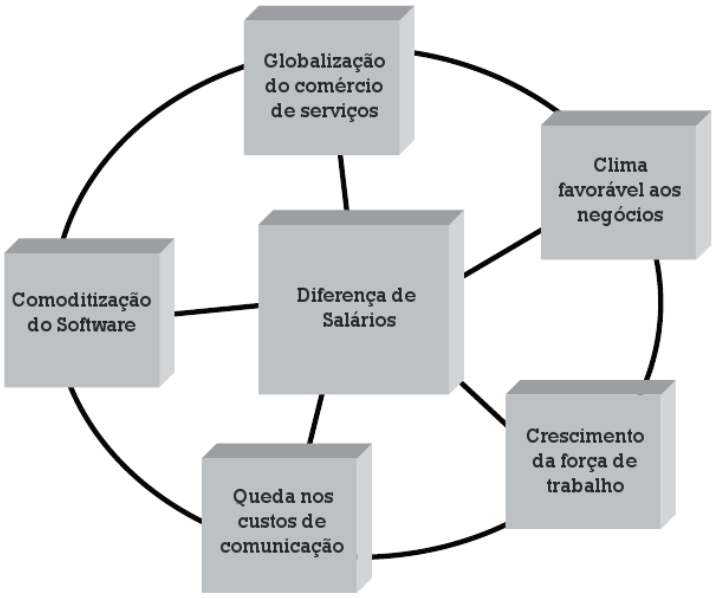


Figura 5: Principais fatores econômicos, de negócio e tecnológicos, adaptada de [Carmel e Tjia, 2005]

O resultado de todo esse processo, é que o desenvolvimento de *software* está cada vez mais se tornando multissítio, multicultural e globalmente distribuído. Engenheiros, gerentes e executivos enfrentam inúmeros desafios formidáveis, em vários níveis, do técnico até o social e cultural [Herbsleb e Moitra, 2001].

O certo é que os tempos mudaram, e mudou também uma grande parte da tradição do desenvolvimento de *software*. Para manter o seu mercado, as organizações não podem

depender das mesmas competências de gerenciamento de engenharia de *software* utilizadas no desenvolvimento interno. Os gerentes devem atualizar suas competências no gerenciamento de pessoas, processos e tecnologia [Karolak, 1998].

2.5.2 Níveis de dispersão

As equipes envolvidas num desenvolvimento distribuído podem estar localizadas num mesmo país ou espalhadas pelo globo. De toda forma, uma vez havendo uma distância física entre as equipes que estão envolvidas no desenvolvimento de um mesmo *software*, existirão desafios complexos e interessantes que recém estão começando a ser compreendidas [Sangwan *et al*, 2006].

A distância física existente entre os envolvidos num mesmo projeto caracteriza diferentes níveis de dispersão. Normalmente, as dificuldades em um cenário de dispersão global é bem diferente das dificuldades encontradas num cenário de dispersão local. Os níveis de dispersão podem auxiliar a caracterizar melhor a distribuição física das equipes do projeto e auxiliar na identificação de possíveis fontes de problemas [Audy e Prikladnicki, 2008]. É importante entender o nível de distância existente e as implicações para as equipes, pois segundo Herbsleb *et al.* (2001 *apud* Audy e Prikladnicki, 2008, p. 48) quando a distância entre os participantes é superior a 30 metros, a frequência de comunicação diminui para um nível idêntico ao de colaboradores que estão distribuídos milhares de quilômetros. Audy e Prikladnicki (2008) definem quatro níveis de dispersão (Figura 6):

- **Mesma localização física** – nesse caso, os atores estão todos no mesmo local (Figura 6-A). As reuniões e interações entre as pessoas acontecem "rosto-a-rosto". Nessa situação, as dificuldades encontradas são as já existentes no desenvolvimento tradicional de *software*;
- **Distância nacional** – neste caso, os atores estão localizados dentro dos limites de um mesmo país (Figura 6-B). Podem acontecer reuniões presenciais em pequenos intervalos de tempo. Todavia, as dificuldades aumentarão caso o país seja muito grande ou ainda, se possuir mais de 1 fuso horário;
- **Distância continental** – neste caso, os atores devem estar localizados num mesmo continente, contudo, podem estar localizados em países diferentes (Figura 6-C). Reuniões presenciais se tornam mais difíceis e a diferença de fuso pode aumentar ainda mais as dificuldades de interação;

- **Distância global** – este caso é caracterizado por seus atores estarem distribuídos tanto em países, quanto em continentes diferentes (Figura 6-D). Nesta situação há um aumento das dificuldades já apresentadas pela distância continental, podendo, inclusive, impedir interações entre as equipes.

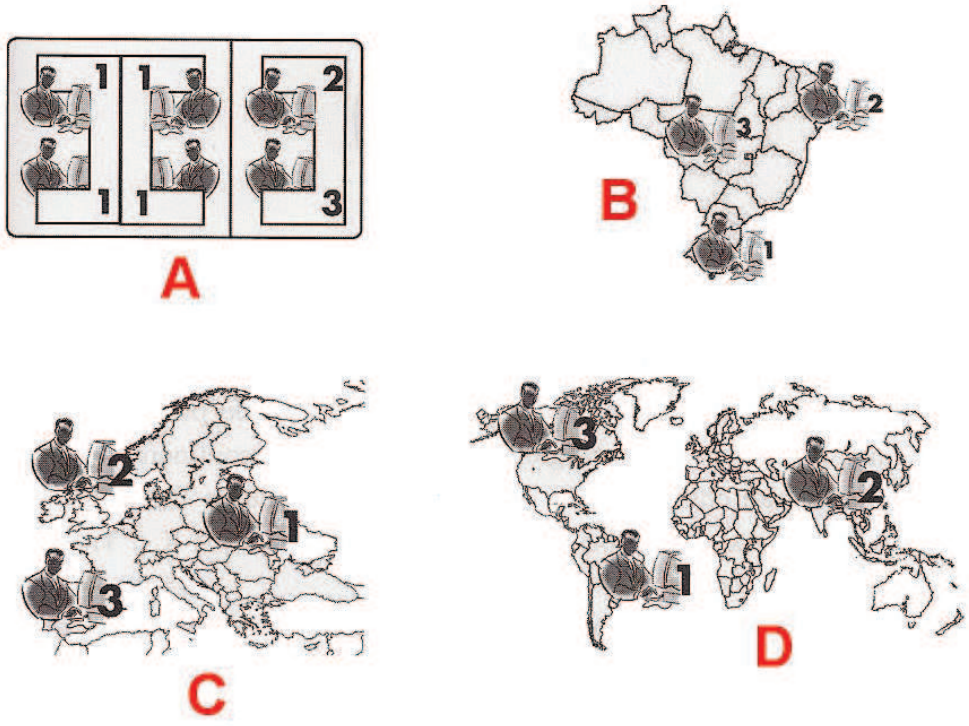


Figura 6: Níveis de dispersão, adaptada de [Audy e Prikladnicki, 2008].

2.5.3 Desafios do DDS

Além dos desafios e dificuldades já presentes no modelo tradicional de desenvolvimento de *software*, Audy e Prikladnicki (2008) afirmam que o DDS traz uma série de novos problemas e desafios. De acordo com Herbsleb e Moitra (2001), a separação física entre os membros de um projeto possui muitos efeitos, e nos mais variados níveis: questões estratégicas, questões culturais, comunicação inadequada, gestão do conhecimento, gestão de projeto e processos e questões técnicas.

Para Carmel (1999) e Carmel e Tjia (2005) o desenvolvimento distribuído de *software* possui cinco forças centrífugas, que dispersam os desenvolvedores mundo afora: choque de cultura, barreiras de coesão, colapso do modelo tradicional de comunicação, de controle e de coordenação (Figura 7).

Audy e Prikladnicki (2008) agrupam esses desafios em: desafios relacionados a pessoas, processos, gestão, comunicação e tecnologia. No contexto deste trabalho, serão analisados os três tipos de desafios que, de alguma forma, podem ser amenizados pela ferramenta proposta: desafios relacionados a pessoas, desafios relacionados à comunicação e desafios relacionados à tecnologia.

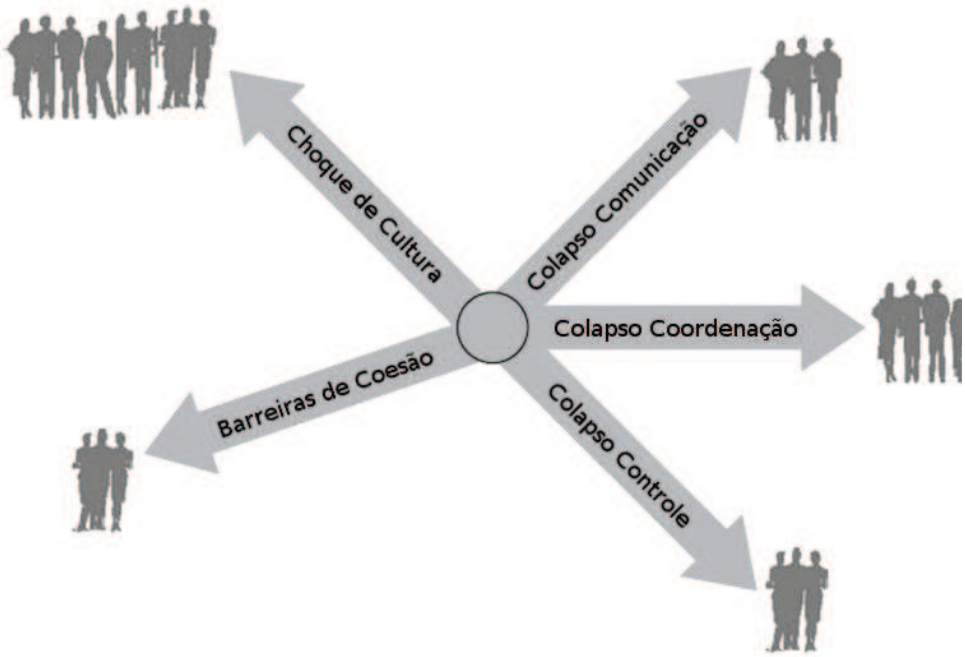


Figura 7: Forças centrífugas [Carmel e Tjia, 2005].

2.5.3.1 Desafios Relacionados a Pessoas

Confiança – a confiança é essencial quando as pessoas dependem umas das outras, para conseguir atingir seus objetivos. Uma equipe na qual os integrantes não confiam um no outro, certamente não funcionará de forma efetiva, ou ainda acabará falhando. Isso significa que é necessário confiar no caráter, habilidade, força e ter segurança na outra pessoa. Como essas qualidades são complexas, levam tempo até se estabelecer. Enquanto membros de equipes locais conseguem estabelecer confiança formal e informalmente, através de interações "rosto-a-rosto", a distância acaba por dificultar o estabelecimento dessa relação de confiança. Também é importante observar que diferentes culturas desenvolvem confiança de forma diferente. Por exemplo, culturas asiáticas geralmente demoram mais tempo para desenvolver confiança que os americanos [Carmel, 1999; Audy e Prikladnicki, 2008].

Diferenças culturais – o desenvolvimento distribuído requer a interação próxima de indivíduos de diferentes culturas. As culturas diferem em várias dimensões críticas como

necessidade de estrutura, atitudes perante hierarquia, sentido da importância do tempo e estilos de comunicação. Enquanto muitas pessoas consideram enriquecedoras essas diferenças, elas também podem levar a mal-entendidos sérios e crônicos, principalmente entre pessoas que não se conhecem bem. Por exemplo, um *e-mail* de alguém em cuja cultura a comunicação tende a ser mais direta, pode parecer áspero ou até mesmo rude para pessoas de outras origens culturais. Pessoas que dão mais importância ao fator tempo, podem parecer rudes no momento de cobranças em relação à interpretação e seriedade dos prazos. Muitas vezes, as diferenças culturais também podem agravar problemas de comunicação. Quando pessoas ficam intrigadas em responder uma mensagem com um tom estranho, muitas vezes elas simplesmente a ignoram ou fazem atribuições maldosas sobre o caráter ou sobre as intenções do emissor [Herbsleb e Moitra, 2001].

Espírito de equipe – Uma boa equipe trás uma série benefícios para a organização. Ela cria uma sinergia de ideias e inovação, é melhor em encontrar problemas e avaliar ideias, proporciona maior acesso à experiência e conhecimento, aumenta a motivação e comprometimento com as tarefas e é uma unidade de trabalho bem flexível. De fato, uma boa equipe nem precisa se comunicar muito, seus membros sabem das fraquezas uns dos outros, já conhecem o processo e compartilham uma visão comum. Em equipes distribuídas, as diferenças culturais e problemas de comunicação podem retardar o processo. A coesão é outro fator importante numa equipe, mas ela se torna mais difícil de ser alcançada em equipes distribuídas. As pessoas podem desconfiar umas das outras em função de estereótipos, maior conversa entre grupos e menos aproximação pessoal. Problemas de comunicação acontecem, provavelmente, em função do idioma. Os integrantes das equipes não conseguem se sentir relaxados, e existe uma constante tensão e estado de alerta em função de mensagens culturais e linguísticas. O tamanho de equipes distribuídas tende a ser maior do que são equipes locais, e isso contribui para a diminuição do espírito de equipe, pois reduz a intimidade e aumenta a complexidade da comunicação. Assim, quanto menor a equipe, menor será a quantidade de *links* de comunicação. Por exemplo, observando a Figura 8 é possível verificar que numa equipe de nove integrantes será necessário gerenciar 36 *links* de comunicação: n*(n-1)/2 [Carmel, 1999; Audy e Prikladnicki, 2008].

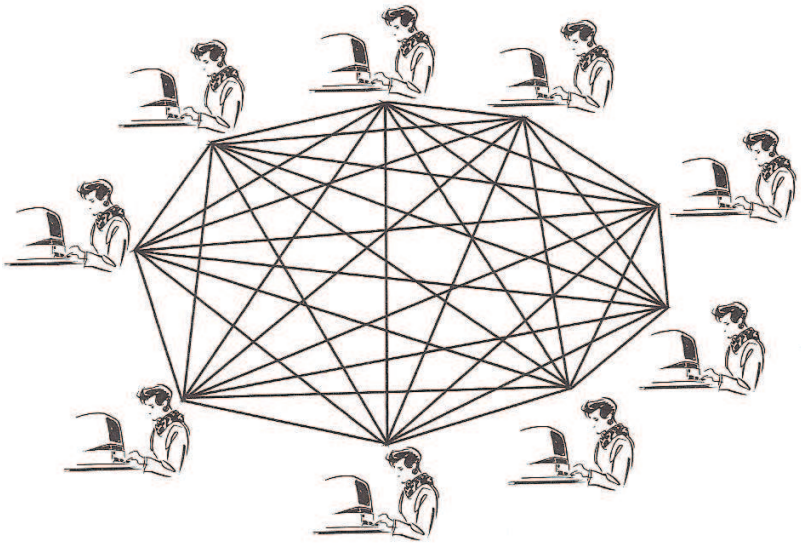


Figura 8: Número de canais de comunicação [Carmel, 1999]

2.5.3.2 Desafios Relacionados à Comunicação

Comunicação inadequada – o desenvolvimento de *software*, principalmente nos primeiros estágios, requer muita comunicação. Dependendo do modelo de *software*, a comunicação e interação entre cliente e equipe podem ser constantes. Conforme Herbsleb e Moitra (2001), projetos de *software* têm duas necessidades de comunicação que são complementares. Uma, a mais formal, é a comunicação oficial, que necessita de uma interface clara e bem compreendida por todos. Para tarefas cruciais como atualizar o status do projeto e resolver questões do projeto, e determinar responsabilidades de tarefas, uma interface imprecisa e mal definida provoca perda de tempo e faz com que os problemas escapem do controle. A segunda forma de comunicação, igualmente vital é a comunicação informal. Desenvolvedores dispersos mantêm poucas conversas informais e espontâneas. Conversas informais de corredor ajudam as pessoas a estarem informadas sobre o que está acontecendo ao seu redor, qual é o status das outras partes do projeto, qual é o conhecimento e em qual área as outras pessoas estão trabalhando, e tantas outras peças importantes para que os desenvolvedores consigam trabalhar juntos e de forma eficiente. Um resultado é que as questões, grandes ou pequenas, que surgem quase que diariamente num projeto de *software*, podem passar desapercebidas ou podem ficar sem ser resolvidas por um longo período de tempo. Quanto mais incerto é o projeto, maior é a necessidade por uma comunicação eficiente [Herbsleb e Moitra, 2001]. De acordo com Carmel e Tjia (2005), os problemas de

comunicação acarretam atrasos, pois perde-se tempo esclarecendo dúvidas, geram retrabalho quando não se entende exatamente a necessidade e, em casos mais graves, podem levar a conflitos quando a mensagem não é bem clara.

Comunicação lateral – algumas equipes globais utilizam estilos de comunicação gerencial na qual os líderes das equipes são os responsáveis pela maior parte da comunicação entre as equipes [Carmel, 1999]. Porém, a comunicação lateral entre os líderes não é o suficiente. Um ambiente profissional com uma supervisão inapropriada, que afunila decisões e conhecimento através das hierarquias formais, simplesmente não é efetiva. A organização moderna e integrada deve promover uma coordenação lateral e uma comunicação lateral (Figura 9). Isso significa comunicação entre os membros das equipes através da organização, ao invés de uma comunicação que siga a hierarquia. É a comunicação lateral que cria uma integração efetiva de tarefas e que rapidamente localiza problemas que vão surgindo. A coordenação lateral é mais flexível, toma decisões mais rapidamente e geralmente melhores, e auxilia na motivação e no aumento de energia de todos os membros da equipe.

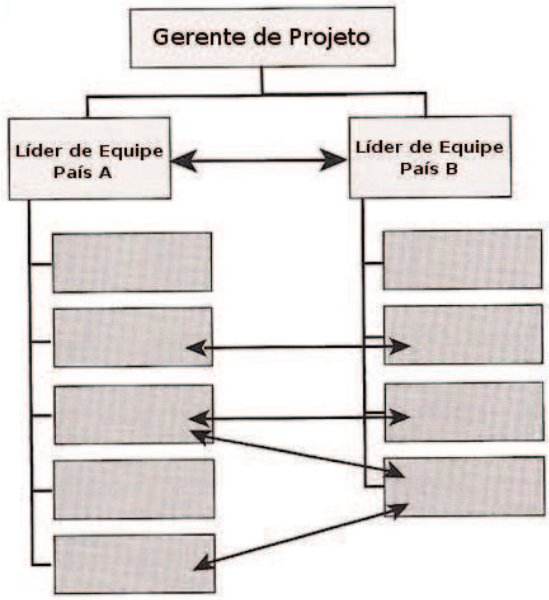


Figura 9: Coordenação e comunicação lateral, adaptada de [Carmel, 1999]

Visão 360º – Na ausência da proximidade física de construções, pessoas e imagens, membros de equipes dispersas precisam sentir como se estivessem próximos aos demais. Isso é chamado de visão 360º [Carmel, 1999]. Cada desenvolvedor pode ver o que as pessoas estão fazendo acima dele, abaixo dele e ao seu redor. Esta dinâmica reduz o sentimento de

isolamento e contribui para a coesão e efetividade da equipe. A visão 360º está intimamente relacionada à noção de transparência organizacional, que requer uma clara definição sobre os papéis individuais, tarefas individuais e tarefas do grupo. Quando esses papéis e responsabilidades não estão claras e transparentes, a confiança e comunicação dentro da equipe diminuirá.

2.5.3.3 Desafios Relacionados à Tecnologia

A tecnologia possui um papel fundamental no DDS. Os desafios relacionados à tecnologia estão divididos em tecnologias de colaboração e telecomunicações.

Tecnologia de colaboração – uma intensiva comunicação eletrônica pode compensar os desafios impostos pela distância e falta de comunicação no DDS. A tecnologia de colaboração auxilia na ampliação da comunicação informal, e possibilita novas maneiras de comunicação formal entre os desenvolvedores dispersos. Também chamada de *groupware*, a tecnologia de colaboração auxilia a criar um ambiente no qual as pessoas dispersas sejam identificadas e localizadas [Audy e Prikladnicki, 2008].

Existem dois tipos principais de tecnologia de colaboração, apresentadas detalhadamente a seguir: tecnologias genéricas de colaboração e tecnologia de colaboração para suporte das atividades de engenharia de *software*.

- **Tecnologias genéricas de colaboração**: compreendem ferramentas como videoconferência, *e-mail*, correio de voz, entre outras, e podem ser agrupadas de acordo com dois critérios: tempo e localização. O critério tempo é utilizado para indicar se a equipe está trabalhando ao mesmo tempo (síncrono) ou em horários diferentes (assíncrono). Já o critério localização, indica se a equipe está trabalhando no mesmo local ou se está dispersa [Audy e Prikladnicki, 2008]. A Tabela 1 apresenta uma relação das ferramentas, agrupadas de acordo com o tempo e localização. Analisando a tabela, é possível perceber que existem ferramentas exclusivas para o ambiente síncrono-local como reuniões e apresentações presenciais, que não podem ser replicadas num ambiente distribuído, sem a utilização de recursos eletrônicos do ambiente síncrono-disperso, como videoconferência ou audioconferência.

  A tecnologia colaborativa genérica não somente facilita a comunicação, como também dá suporte a vários processos, a exemplo de: encontrar e obter informações que auxiliem na resolução de problemas, auxiliar na organização

dos trabalhos da equipe, definição de requisitos e acompanhamento das implementações, entre outros. Essa tecnologia serve como memória e centro de conhecimento da equipe. Nesse centro, são encontrados os documentos das equipes, comunicação, artefatos, código, registros de *bugs*, tarefas, entre outros, contribuindo ainda para que a equipe tenha uma visão 360º. Aliado a isso, esse centro é útil não somente para o caso de equipes distribuídas, mas também quando se trabalha em parceria com outra empresa [Carmel, 1999].

Tabela 1: Tecnologias de colaboração, adaptada de [Audy e Prikladnicki, 2008] e [Carmel, 1999]

		Tempo	
		Síncrono	**Assíncrono**
Localização	**Mesmo Local**	Reuniões Votações Apresentações	Computadores compartilhados
	Dispersos	Telefone *Chat* Videoconferência Audioconferência Quadro eletrônico	E-mail Correio de Voz Correio de Vídeo Listas de discussão Groupware Calendário

- **Tecnologia de colaboração para suporte das atividades de engenharia de *software***: uma colaboração não pode ser alcançada utilizando-se somente a tecnologias colaborativas genéricas. O desenvolvimento de *software* necessita de soluções específicas de acordo cada a tarefa. Essas tecnologias, conhecidas como CT-SE (*Collaborative Technology to Support Software Engineering*), compreendem ferramentas como SCM (*Software Configuration Manager*), CASE (*Computer Aided Software Engineering*), gerenciamento de projeto, entre outros. Além dos objetivos e benefícios das tecnologias genéricas (memória do projeto, repositório e centralizador de documentação, proporcionar uma visão 360º), o CT-SE auxilia a reduzir a duplicação de esforços e redundância, graças ao compartilhamento de vários tipos de conhecimento do projeto; suporta atividades de coordenação e *workflow*, pois possibilita que os membros das equipes possam gerenciar muitas de suas tarefas através de coordenação lateral ou procedimentos eletrônicos de

workflow; e ajuda a garantir a qualidade através de funcionalidades como controle de versões e monitoramento de *bugs* [Audy e Prikladnicki, 2008; Carmel, 1999].

Telecomunicações – dados os elevados custos de coordenação, não deveria acontecer nenhum esforço sério, antes de se estar utilizando uma conexão confiável e de alta velocidade, para todas as formas de comunicação e dados. Nesse sentido, é necessária uma análise cuidadosa para verificar se a infraestrutura atende às necessidades. Apesar de não ser a realidade de todos os países, na grande maioria existe uma estrutura de telecomunicações que permite uma transmissão de áudio e vídeo com um bom desempenho e de forma confiável. As conexões seguras podem ser estabelecidas de diversas formas, desde conexões dedicadas via satélite até VPN (*Virtual Private Network*). Devido ao baixo custo, alta disponibilidade e confiabilidade, as Redes Privadas Virtuais tem crescido nos últimos anos [Audy e Prikladnicki, 2008].

A comunicação síncrona como telefone, áudio conferência, videoconferência, compartilhamento de aplicativo e algumas vezes análise e revisão síncrona *online* do código, pode ajudar a melhorar, inclusive, a qualidade de vida. Profissionais de tecnologia da informação envolvidos em trabalho global, frequentemente reclamam sobre a necessidade de comprometer a vida pessoal para falar com colegas distantes, em outros fusos horários. Além disso, trabalhar num mesmo fuso horário facilita a comunicação síncrona efetiva [Carmel e Agarwal, 2001].

2.6 Ruptura do Modelo Tradicional de Desenvolvimento

Desde o seu surgimento, o Desenvolvimento Distribuído de *Software* trouxe consigo uma série de novos desafios e mudou grande parte da tradição do desenvolvimento de *software*. Gerentes e engenheiros de *software* foram obrigados a atualizar suas competências de gerenciamento de pessoas, processo e tecnologia, tendo em vista o novo paradigma [Karolak, 1998].

Dois movimentos de *software* bem sucedidos de DDS são a comunidade de *software* livre ou OSS (*Open Source Software*) e o Desenvolvimento Ágil de *Software*. Esses modelos usam enfoques não convencionais no DDS, fundiram a comunicação formal e a não formal, e conseguiram superar, com sucesso, os problemas da colaboração distribuída [Carmel e Tjia, 2005].

2.6.1 O caso do *Software* Livre

Uma das experiências mais bem sucedidas de Desenvolvimento Distribuído de *Software* é o *software* livre [German, 2003; Carmel e Tjia, 2005]. Um *software* é livre quando proporciona alguns direitos mínimos aos seus usuários [German, 2003]:

- Direito de fazer cópias do *software*;
- Direito de distribuir as cópias;
- Direito de ter acesso ao código fonte do *software;*
- Direito de fazer melhorias no programa.

Existem inúmeros *softwares* livres e exemplos clássicos são: o sistema operacional Linux, o conjunto de ferramentas de desenvolvimento GNU[4]. Outro exemplo é o projeto Gnome[5], que é um ambiente *desktop* para o sistema operacional Linux.

Um dos principais requisitos para o desenvolvimento distribuído é a concordância em utilizar as mesmas ferramentas para o desenvolvimento de *software*, de forma que todos os integrantes da equipe tenham acesso às mesmas ferramentas. No desenvolvimento privado, isso não chega a ser um problema, pois as empresas estabelecem e disponibilizam o ambiente e ferramentas necessárias. Por outro lado, dada a filosofia existente por trás do *software* livre, as ferramentas utilizadas neste caso geralmente são as disponíveis também em *software* livre [German, 2003].

Vários projetos de *software* livre utilizam as ferramentas GNU (make, gcc, autoconf, automake, vi, emacs, entre outros), CVS para o gerenciamento de configuração de *software*, Bugzilla[6] para o gerenciamento de *bugs*, GNU Mailman para as listas de discussão e o Linux como plataforma de desenvolvimento. Conforme German (2003), mais de 500 pessoas possuem direito de gravação no repositório CVS do projeto Gnome. Para contribuir com o projeto, basta que os desenvolvedores instalem uma versão recente do Linux e o ambiente estará disponível. Dessa forma, o custo é bastante baixo, para não dizer inexistente.

Por outro lado, é necessário observar que, apesar de possuir uma gama de ferramentas para auxiliar no desenvolvimento distribuído, utilizar as ferramentas de forma separada não é tão fácil e intuitivo quanto o proporcionado por uma IDE de desenvolvimento de *software*.

---

[4] GNU: http://www.gnu.org
[5] Gnome: http://www.gnome.org
[6] Bugzilla: http://www.bugzilla.org

Segundo SCACCHI (2002), muitos dos projetos de *software* livre não possuem uma fase tradicional de engenharia de requisitos. Por vezes, os únicos *stakeholders* são os próprios desenvolvedores que também atuam como clientes, investidores, codificadores, testadores e documentadores. Em alguns casos, os requisitos são obtidos através da troca de *e-mails* na lista de discussões. Muitas vezes, surgem requisitos de uma discussão cujo objetivo original não era a análise de requisitos.

A comunicação é um elemento fundamental em projetos de DDS. Em projetos *open source*, ela está baseada em: listas de discussão, utilizadas tanto por desenvolvedores quanto por usuários finais dos *softwares*, são uma fonte de informação que auxiliam nas tomadas de decisão no projeto; *chat*, utilizado para bate papo e conversas entre desenvolvedores; *Websites*, possuem uma grande quantidade de informações a respeito do projeto, direcionado a cada tipo de colaborador: codificadores, testadores, documentadores, tradutores [German, 2004]. Aliado a isso, existem diversos repositórios de códigos fonte, além de blogs e redes sociais.

2.6.2 O Desenvolvimento Ágil de *Software*

O Desenvolvimento Ágil de *Software*, conforme apresentado anteriormente, possui vários exemplos de metodologias. Todavia, segundo Carmel e Tjia (2005), o *XP* (*Extreme Programming*) é a mais conhecida e bem sucedida. As metodologias ágeis surgiram como uma reação às metodologias mais pesadas, moldadas segundo o *CMM* (*Capability Maturity Model*). Os seguidores do *XP* defendem que o trabalho deve ser executado em pequenas equipes locais. Os programadores trabalham em pares, um ao lado do outro, auxiliando, guiando e aconselhando-se mutuamente. Nessa abordagem existe grande contato e muita interação cara-a-cara. Por isso, *XP* é a antítese do desenvolvimento colaborativo à distância.

Todavia, os defensores da metodologia *XP* aprenderam a adaptá-la à realidade do trabalho distribuído, criando inclusive a *DXP* (*Dispersed XP*). Eles aprenderam uma série de lições como, por exemplo: iniciar a equipe num mesmo lugar, para que as pessoas se conheçam; nomear pessoas que viajam fisicamente para visitar os sítios distribuídos; concordar com uma definição de bloco de tempo comum; concordar com o uso de ferramentas de comunicação comum; e concordar em utilizar as mesmas diretrizes e padrões de codificação [Carmel e Tjia, 2005].

A programação realizada por pares distribuídos, ao invés daquela realizada por pares locais que utilizam o mesmo computador, tem sido objeto de estudos a fim de avaliar as

diferenças com relação à produtividade e à qualidade [Stotts *et al*, 2003]. Entre as vantagens estão:

- É necessário que os pares mantenham registros eletrônicos de suas ideias e trabalhos;
- Como os dois programadores não precisam sentar-se lado a lado, não são necessárias mudanças nas posições dos móveis;
- Os membros permanecem mais tempo focados na sua tarefa, reduzindo conversas desnecessárias;
- Por muitas vezes, enquanto um programador faz alterações no código, o outro pode efetuar outras tarefas, como pesquisa em base de conhecimento ou na internet;
- Existe uma redução na necessidade de viagens.

Por outro lado, para que o trabalho realmente traga vantagens, os pares distribuídos devem estar aptos a se comunicar. Além disso, é importante que ambos possam acompanhar as alterações que estão sendo feitas no código, mas que apenas um altere o trecho de código por vez [Schümmer e Schümmer, 2001].

A produtividade e a qualidade do produto de *software* não apresentam diferenças significativas na comparação entre a programação sendo realizada por pares locais ou pares distribuídos [Baheti *et al*, 2002]. Por outro lado, muitas vezes programadores podem acabar por trabalhar de forma isolada ao realizar suas tarefas. Isso pode ocorrer quando há problemas com o canal de comunicação de dados, quando não existe acordo em relação à melhor solução a ser adotada para a resolução de um problema, ou ainda quando as regras não são estabelecidas antes de iniciar a sessão [Canfora *et al*, 2003].

A programação em pares possui vários benefícios: conversando e discutindo, duas pessoas conseguem encontrar um maior número de soluções para um problema, e de forma mais rápida, em comparação a uma pessoa trabalhando sozinha. Enquanto que uma digita o código, a outra pode ir pensando nos impactos que serão causados em outras partes dos programas. Além disso, o par que não está digitando fica revisando constantemente o código, o que proporciona uma revisão natural e eficiente. Por fim, a programação em pares tende a produzir um número menor de linhas, oferecendo melhor qualidades e melhores soluções ao projeto [Cockburn e Williams, 2001].

Williams e Kessler (2000) destacam que, enquanto uma pessoa digita o código o seu par deve ficar atento e participar ativamente. Como ambos estão envolvidos numa atividade comum e com os mesmos propósitos, cada participante estimula o parceiro a ficar focado no trabalho. O resultado é um ganho de produtividade, pois dificilmente um programador interrompe o trabalho para executar outras atividades como navegar na internet, ou ler *e-mails*.

Outros benefícios citados por Cockburn e Williams (2001), são a uniformização do conhecimento e a rapidez no aprendizado. Aliado a isso, está o fato da disseminação do conhecimento e redução do impacto gerado pela saída de um programador-chave.

Analisando-se os benefícios proporcionados pela interação de duas pessoas, acredita-se que havendo uma interação entre mais pessoas, a qualidade final do projeto também tenderá a aumentar. Todavia, entende-se que, se ocorrem problemas quando duas pessoas não se organizam apropriadamente [Canfora *et al*, 2003], o resultado pode ser ainda pior quando mais pessoas estiverem interagindo numa mesma sessão, sem uma organização pré-estabelecida.

2.7 Ferramentas importantes no DDS

A colaboração possui um papel fundamental no desenvolvimento de *software*, e essa importância cresce ainda mais quando o desenvolvimento for distribuído. Uma organização terá mais benefícios das tecnologias colaborativas se utilizar um conjunto delas: mensagem instantânea, videoconferência, serviços de áudio de qualidade, repositórios integrados e ambientes de compartilhados [Carmel e Tjia, 2005].

As ferramentas colaborativas até podem ser utilizadas paralelamente ao IDE, todavia, a integração traz grandes recompensas como: redução do desgaste no processo de desenvolvimento, melhoria no senso de contexto, além de contribuir para a rastreabilidade entre artefatos de colaboração e artefato de código [Cheng et al, 2003].

De acordo com Cheng et al. (2003), as características de colaboração devem ser escolhidas cuidadosamente, para que atendam as necessidades da equipe e também levem em consideração questões como trabalho individual versus o trabalho colaborativo.

2.7.1 Edição Colaborativa

O desenvolvimento de *software* raramente é uma atividade de codificação solitária. Frequentemente é um processo colaborativo, com equipes de programadores trabalhando em conjunto para projetar soluções e produzir código de qualidade. Os membros dessas equipes normalmente olham os códigos uns dos outros, fazem planos conjuntos sobre como proceder, ou ainda corrigir os *bugs* dos códigos [Cheng *et al*, 2003].

De todas as atividades do desenvolvimento de *software*, a que é mais executada utilizando-se *offshore*[7] é a codificação, seguido por testes e manutenção, conforme pode ser observado na Figura 10. Estas atividades estão diretamente envolvidas com a etapa de implementação, no processo tradicional de *software*.

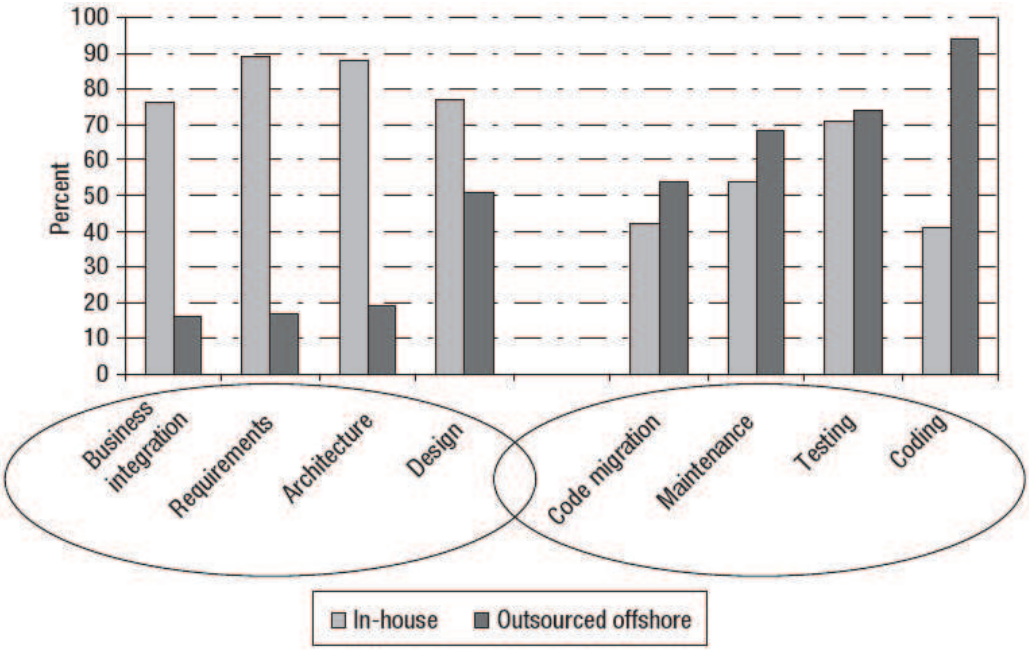


Figura 10: Execução interna versus execução externa [Carmel e Tjia, 2005]

A edição simultânea tem papel fundamental no desenvolvimento distribuído, auxiliando para que se tenha um código padronizado, otimizado e de melhor qualidade final. Segundo Carmel e Agarwal (2001), deve ser possível a análise e revisão síncrona *online* do código.

[7]Offshore: atividades realizadas fora do país, similar à palavra estrangeiro [Carmel e Tjia, 2005].

2.7.2 Controle de Versões

Conforme Cheng *et al*. (2003), ferramentas de controle de versões ou gerenciamento de configuração, se constituem no repositório central da equipe de desenvolvimento de *software*. Elas permitem que desenvolvedores modifiquem, permutem, excluam e unam arquivos de maneira coordenada. Importante no desenvolvimento tradicional de *software*, o controle de versões é indispensável quando pensamos num ambiente distribuído, com vários programadores alterando os mesmos códigos.

Um sistema de controle de versões, atualizado constantemente, auxilia no processo de acompanhamento das mudanças e diminui os enganos. À medida que o projeto avança, é possível acompanhar as mudanças diariamente, além de manter os códigos atualizados. Com a utilização dessa ferramenta, uma equipe consegue enxergar as modificações feitas por outra equipe, mesmo sem utilizar uma sessão de edição colaborativa.

Se por um lado as ferramentas de controle de versões são poderosas, por outro elas acabam por ser muito complexas de usar, especialmente em projetos grandes. Ao integrar essa ferramenta no IDE, etapas extras de gerenciamento do repositório podem ser evitadas. Além disso, uma boa integração poderá passar ao desenvolvedor a impressão que a utilização de uma ferramenta de gerenciamento de configurações é tão fácil quanto manipular outros arquivos locais [Cheng *et al,* 2003].

2.7.3 Comunicação

Num processo de desenvolvimento distribuído, a comunicação é um ponto crucial para o sucesso de um projeto de *software* [Paasivaara e Lassenius, 2004]. O maior problema em ter uma equipe de desenvolvimento distribuída é a falta de uma comunicação eficiente. Os desafios na comunicação como distância física, diferenças culturais e falta de infraestrutura, reduzem significativamente o progresso de um projeto [Herbsleb e Mockus, 2003].

Num projeto não distribuído, é possível conversar com os programadores inclusive na hora do café, podendo-se decidir a respeito da inclusão de uma nova característica ao sistema. Além disso, decisões podem ser feitas em conversas informais de corredor. Já no caso de equipes distribuídas, essa mesma dinâmica não é possível.

Se numa equipe local a comunicação já é difícil, num ambiente distribuído o problema fica ainda maior [Paasivaara e Lassenius, 2004]. Para superar os desafios da comunicação, os

projetos em DDS utilizam uma variedade de mídias de comunicação, como telefone, teleconferências, *e-mail* e mensagens instantâneas [Herbsleb e Mockus, 2003].

Asynchronous technologies	Synchronous technologies
E-mail	Voice telephony/Internet telephony
Voice mail/video mail	Audio-conferencing
Online discussion groups	Video-conferencing (meeting room)
Calendaring	Video-conferencing (desktop/web-based)
Collaborative authoring and commenting	Web-audio hybrid meetings
Project Management	IM (Instant Messaging)
Production tools and repositories such as configuration management systems, issue tracking systems, workflow tools, knowledge management	Whiteboard/Screen Sharing

Figura 11: Tipos de tecnologia colaborativa [Carmel e Tjia, 2005]

Atualmente, existe uma variedade de tecnologias tanto síncronas quanto assíncronas, como pode ser observado na Figura 11. Apesar do poder das tecnologias assíncronas, existem fortes razões para a utilização de comunicação síncrona [Carmel e Agarwal, 2001].

A comunicação síncrona auxilia a resolver problemas de comunicação, problemas de entendimento e ajuda a evitar que pequenos problemas se tornem maiores. Pequenas questões podem levar dias para serem resolvidas através de *e-mails*, ao passo que uma rápida conversa pode acabar com o problema. Nesse sentido, a comunicação assíncrona muitas vezes pode atrasar ou complicar a resolução de problemas [Carmel e Agarwal, 2001].

2.7.3.1 Mensagens Instantâneas (IM)

As mensagens instantâneas, segundo Cheng et al. (2003), são muito utilizadas para auxiliar no desenvolvimento de *software*. As mensagens podem ser avisos sobre as últimas atualizações de código, uma discussão sobre alguma falha no programa, ou ainda sobre o código não documentado de um programa. Além disso, podem conter referências específicas do projeto como, por exemplo, URLs, nomes de arquivos e pacotes, e localização do repositório de código, ou ainda ser utilizado para enviar fragmentos de código.

A principal vantagem dessa ferramenta de IM estar integrada num IDE é o fato de que as trocas de mensagens, normalmente, levarão a discussões sobre o código fonte, atualizações, entre outros, relativos ao projeto no qual estão envolvidos. A utilização de programas externos para a troca de mensagens instantâneas tende a dispersar a atenção do usuário para outros

assuntos, que não relacionados ao projeto, uma vez que nesses aplicativos ele também possui contatos pessoais e não somente profissionais [Cheng et al, 2003].

2.7.3.2 Comunicação via Áudio

A comunicação síncrona utilizando mensagens instantâneas é muito mais efetiva que as mensagens assíncronas de *e-mails*. Todavia, em determinadas situações, e principalmente para evitar problemas de comunicação, pode ser mais aconselhável falar diretamente com a pessoa.

Aliado a isso, é bem provável que nem todos os atores envolvidos no processo de desenvolvimento de um *software* disponham dos mesmos meios e dispositivos de comunicação. Nesse sentido, é importante garantir que todos os *stakeholders* consigam ser facilmente contatados, sem que haja a dependência de determinados dispositivos ou formas de comunicação.

Ao garantir um processo de comunicação eficiente, contribui-se para a melhora na comunicação e coordenação lateral, cuja importância é ressaltada por Carmel (1999).

2.7.4 Gerenciamento de Tarefas e Agenda

Utilizar ferramentas para o gerenciamento de tarefas é importante quando se possuem equipes distribuídas. Elas auxiliam na organização e acompanhamento da execução, por exemplo, das tarefas de cada programador.

Quando uma empresa possui desenvolvedores localizados em diferentes fusos horários, a ferramenta de gerenciamento de tarefas poderá auxiliar no processo de desenvolvimento "*round-the-clock*", de forma que, enquanto um programador estiver dormindo, o outro poderá continuar a desenvolver a tarefa. Para isso, o controle de tarefas será essencial, pois através dele o programador poderá, antes de iniciar, ver o status do trabalho e ler observações a respeito da tarefa, deixadas pelo outro programador.

Aliado a isso, é necessária uma ferramenta de agenda que possibilite o controle e agendamento de trabalhos, reuniões e outros eventos conjuntos, para possibilitar que cada integrante da equipe esteja ciente dos eventos previstos no projeto.

Outro aspecto importante dessas ferramentas é o fato de contribuírem para a redução dos problemas e dificuldades relacionados à visão 360º, descrita por Carmel (1999), pois

auxiliam na redução do sentimento de isolamento dos desenvolvedores dispersos, contribuindo, assim, para o aumento da coesão e efetividade da equipe.

3 TRABALHOS RELACIONADOS

Neste capítulo são apresentados trabalhos com assuntos relacionados a esta dissertação. A pesquisa por trabalhos relacionados foi feita com base em publicações e estudos voltados ao tema Desenvolvimento Distribuído de *Software*.

Muitos dos trabalhos encontrados apresentam estudos de casos e propõem soluções baseadas na utilização integrada de diferentes ferramentas, enquanto que outros, contribuem no sentido de implementar soluções que auxiliam a resolver aspectos específicos desse modelo de desenvolvimento.

Dentre os trabalhos analisados, neste capítulo são apresentadas quatro ferramentas desenvolvidas que, de alguma forma, influenciaram o desenvolvimento da aplicação proposta pelo presente trabalho.

A seção 3.1 apresenta o VIMEE, um aplicativo voltado para comunicação síncrona e tomada de decisão em equipe. Na seção 3.2 é apresentado o RemotePP, um *software* com várias formas de comunicação e edição colaborativa por pares distribuídos. A seção 3.3 traz o CVW, um ambiente colaborativo integrado com uma série de ferramentas para o trabalho em grupo. A seção 3.4 descreve o CollabEd, um aplicativo que permite a edição colaborativa por diversos usuários e a comunicação através de *chat*.

3.1 VIMEE

Em seu trabalho, Trindade *et al.* (2008) apresentam o VIMEE (*Virtual Distributed Meeting Tool*), um programa para suportar a comunicação e a tomada de decisão em equipe. Essa ferramenta possibilita uma comunicação síncrona, explícita e formal em ambientes DSDE (*Distributed Software Development Environment*). Segundo os autores, o VIMEE possibilita uma reunião mais produtiva, definindo uma área comum para a realização das reuniões virtuais, nas quais os participantes podem interagir e visualizar a performances uns dos outros. A comunicação entre os participantes ocorre de maneira clara e explícita, e é baseada em procedimentos formais, que guiam o andamento das atividades, indicando quem está habilitado a executar algo. Entre as características do *software*, estão:

- Agendamento de reuniões, com aviso aos participantes;
- Permite a troca síncrona de mensagens de texto;
- Não possui limites quanto ao número de participantes;
- Permite a visualização de artefatos do projeto;

- Permite o compartilhamento de documentos e imagens;
- Possibilita o armazenamento de documentos gerados numa reunião;
- Possibilita a realização de votações, para tomada de decisão.

Para organizar o fluxo dos procedimentos, a ferramenta permite a coordenação das reuniões através do papel de um mediador.

A interface do mediador, apresentada na Figura 12, difere da interface dos demais usuários, e tem por objetivo possibilitar: o início e término de uma reunião (A), bloquear ou liberar algum participante (B), controlar as votações (C) e enviar mensagens (D). Nessa interface, também é controlado qual usuário, previamente inscrito, pode se manifestar (E). Aliado a isso, possibilita a liberação do acesso aos artefatos do projeto (F), além de poder tornar visível a existência de documentos e imagens para os demais participantes.

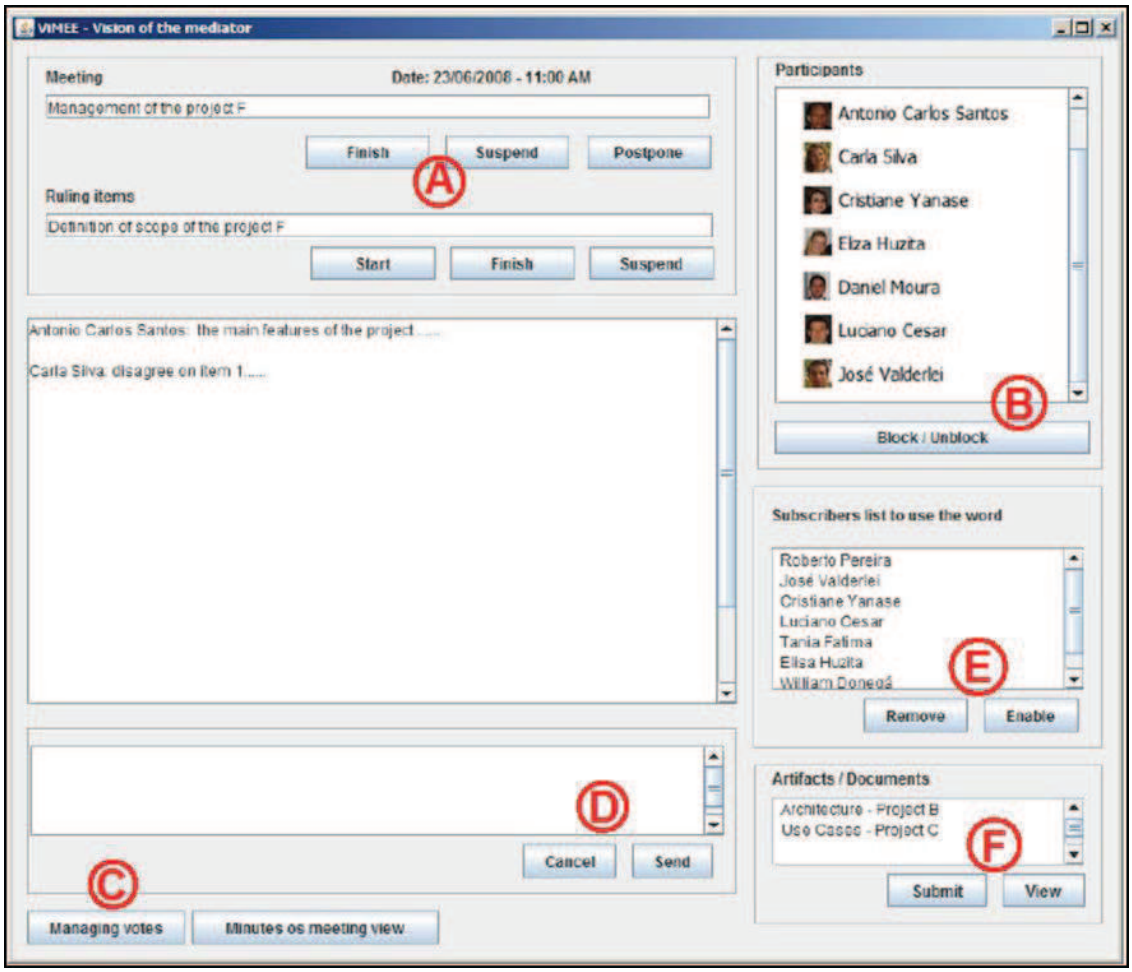


Figura 12: Ambiente do VIMEE, visão do mediador [Trindade et al, 2008]

A interface do participante, por sua vez, permite que a comunicação através de mensagens de texto, participe das votações criadas pelo mediador, visualize os artefatos do projeto e disponibilize documentos.

Em termos técnicos, o VIMEE é um *software* cliente/servidor, e foi desenvolvido utilizando a tecnologia Java J2SE. Sua arquitetura possui uma separação lógica em três camadas: *Application Layer*, responsável por apresentar a interface gráfica e os elementos para interação dos usuários; *Business Layer*, onde estão localizados os objetos de negócio e modelo de dados; e *Infrastructure Layer*, que fornece a infraestrutura para a camada de negócio. Apesar das tentativas de manter contato com os autores, não foi possível ter acesso ao código fonte do aplicativo.

3.2 RemotePP

O RemotePP, cuja interface é apresentada na Figura 13, é um ambiente cooperativo de desenvolvimento com o objetivo de dar suporte à programação em pares distribuídos [Borges *et al*, 2007]. Como o seu objetivo é voltado para a programação em pares, a sessão somente pode ser estabelecida entre dois programadores, um local e outro remoto.

O ambiente possui uma série de ferramentas, a seguir descritas:

- Edição de código-fonte: este módulo permite a edição de programas de forma colaborativa, porém, não simultâneo, ou seja, somente um programador pode alterar o código por vez. Caso a sessão não tenha sido estabelecida com o outro usuário, a edição acontece somente de forma local;
- Quadro branco: funcionalmente similar ao módulo de edição de código-fonte, este módulo permite que ambos os programadores, colaborativamente, editem desenhos e diagramas. Neste caso, a edição pode ser feita simultaneamente por ambos programadores;
- *Chat*: permite a troca de mensagens instantâneas entre os dois usuários;
- Vídeo: este módulo permite que os programadores realizem videoconferência. Para isso, é necessário que exista uma *webcam* instalada e configurada nos computador;
- Voz: possibilita que os programadores se comuniquem por áudio, utilizando o microfone acoplado ao computador;

- Ferramenta externa: permite que um programa externo seja executado, a exemplo de um compilador. As mensagens de saída, oriundas da execução do programa externo, serão exibidas para ambos os programadores.

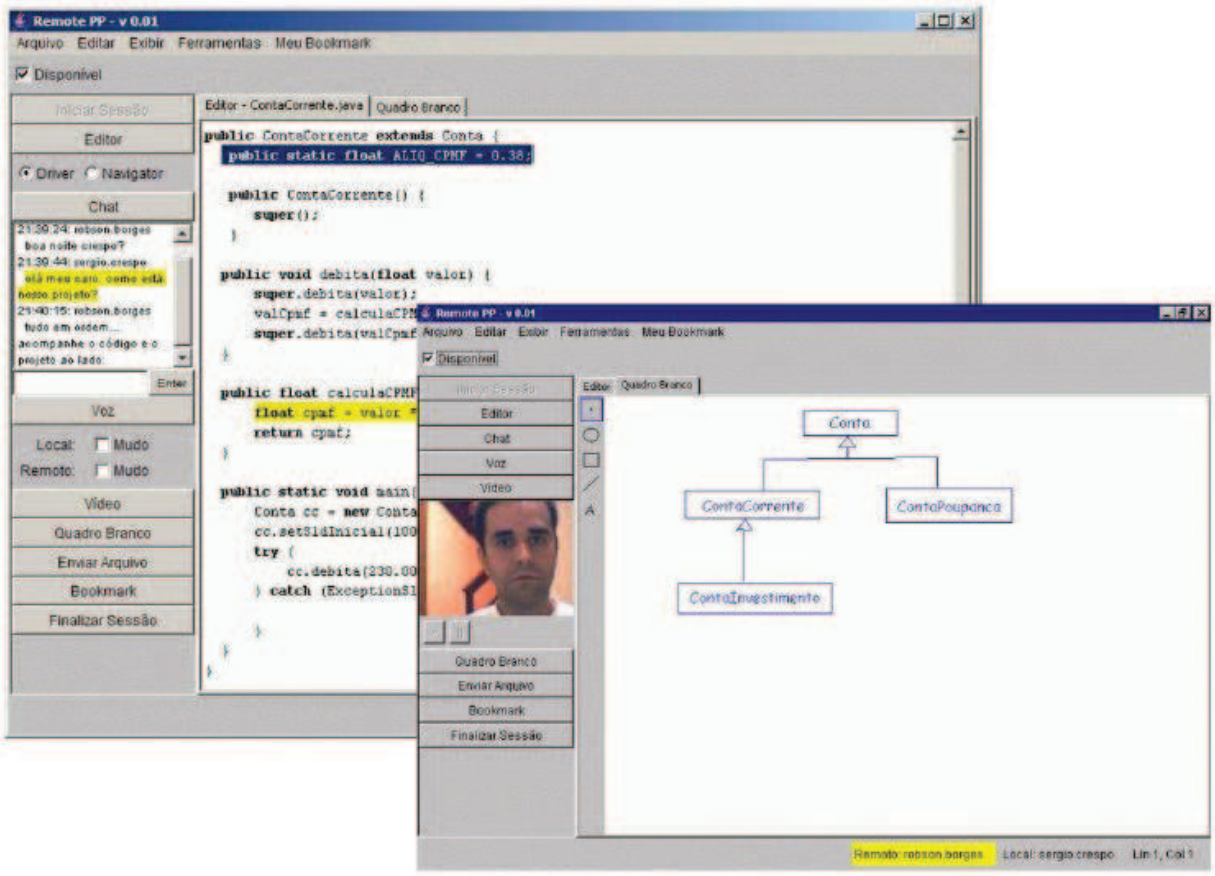


Figura 13: Ambiente do RemotePP [Borges *et al*, 2007]

O programa foi desenvolvido na linguagem Java e sua arquitetura é composta por três camadas: Aplicação, que implementa os módulos do sistema; Interface, responsável pelo ambiente de interação do usuários; e Comunicação, responsável pela troca de informações entre os dois programas.

A conexão entre os usuários é estabelecida através do endereço IP dos computadores. Assim, para que um usuário consiga estabelecer uma conexão ao ambiente remoto, é necessário que os equipamentos se "enxerguem", o que pode ser um fator limitante quando estes estiverem atrás de um *firewall*. O protocolo utilizado para a troca das mensagens não está documentado e as tentativas de manter contato com o autor, para a obtenção do código fonte do aplicativo, não foram bem sucedidas.

Apesar da linguagem de programação ser multiplataforma, a utilização do aplicativo está restrita ao sistema operacional Windows, em função da biblioteca, proprietária e de código fechado, utilizada nos módulos de vídeo e voz.

3.3 CVW

O CVW (*Collaborative Virtual Workspace*) é um ambiente colaborativo integrado, no qual as equipes podem se comunicar, colaborar e compartilhar informações, independentemente da sua localização geográfica [Maybury, 2001]. O CVW utiliza o conceito de salas, onde os usuários se encontram e podem iniciar reuniões virtuais.

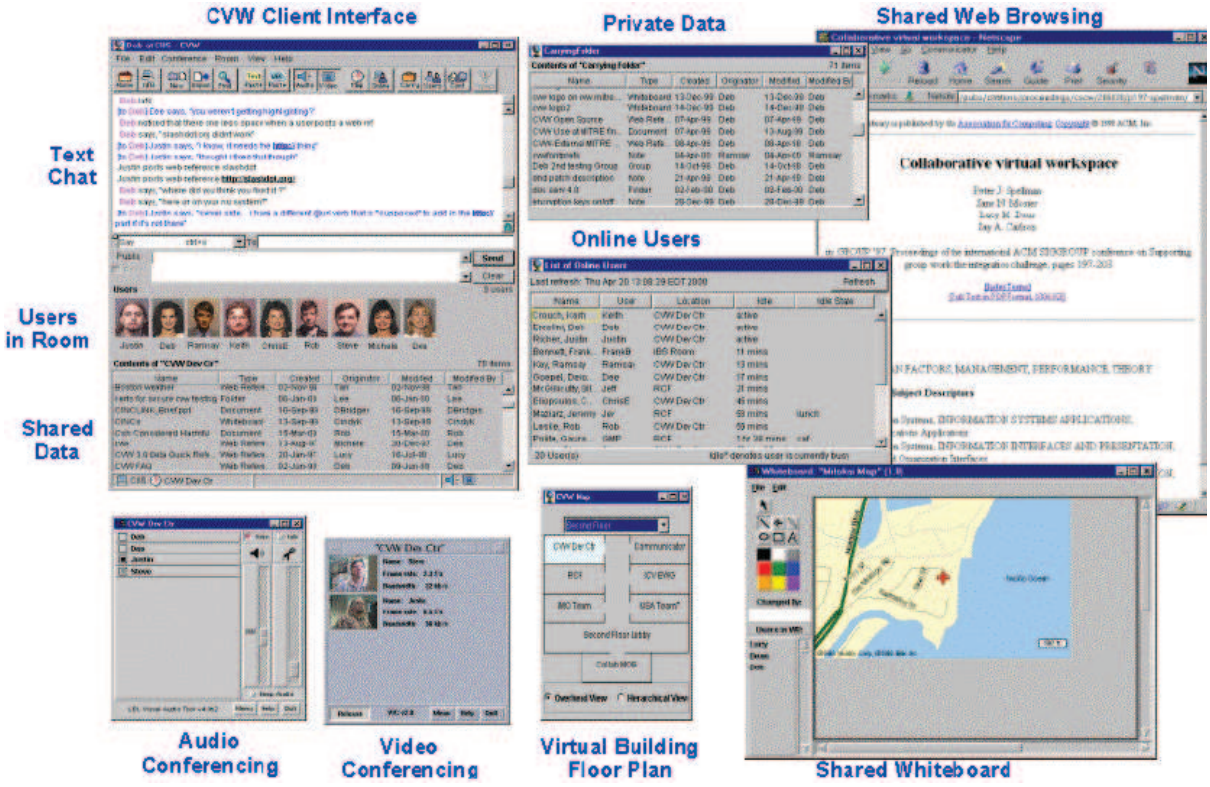


Figura 14: Ferramentas disponibilizadas pelo CVW [Maybury, 2001]

Este aplicativo possui uma série de ferramentas síncronas e assíncronas, apresentadas na Figura 14 e descritas a seguir:

- *Chat*: permite a troca de mensagens de texto;
- Conferência via áudio: ferramenta que possibilita que os usuários se comuniquem utilizando som;
- Videoconferência: possibilita que os usuários se comuniquem através de vídeo;

- Quadro branco: fornece um espaço no qual os participantes podem desenhar de forma colaborativa, similar à ferramenta do RemotePP;
- Planta: permite que o usuário veja a localização real dos seus participantes;
- Navegador: trata-se de um navegador *web* compartilhado, de forma que todos os usuários vejam o mesmo conteúdo.

A arquitetura da aplicação está baseada no modelo cliente/servidor. O servidor, majoritariamente desenvolvido em Java, possui versões para os sistemas operacionais Linux e Solaris, enquanto que o cliente possui clientes para Windows, Linux e PalmOS. Além de Java, funcionalidades nativas dos Sistemas Operacionais e ferramentas externas como a videoconferência estão implementadas em outras linguagens como C, C++ e TCL.

A conexão entre o cliente e o servidor é feita através do uso de *sockets* e os comandos do protocolo de comunicação são definidos como strings de texto. O projeto parece estar descontinuado, sendo que a última modificação foi feita no ano de 2001.

3.4 CollabEd

Este aplicativo, apresentado na Figura 15, é um editor desenvolvido sobre uma plataforma de colaboração, segundo Lawrence (2009). De fato, no *website* onde o programa está disponível para *download*, também está disponível um aplicativo de desenho que utiliza o mesmo conceito.

O design da plataforma compreende três categorias essenciais: conexão, manutenção e *replay* das interações. A arquitetura do aplicativo é baseada no modelo cliente/servidor. Tanto o servidor quanto o cliente são desenvolvidos na linguagem Java e, assim, são multiplataformas. Além de ser executado como uma aplicação independente, o CollabEd pode ser acoplado a editores de texto já existentes, através *plugins*. Já existem *plugins* disponibilizados para os editores Netbeans, Eclipse e JEdit. Todavia, após efetuar a instalação no ambiente Netbeans, não foi possível localizar o aplicativo e suas funcionalidades. Dessa forma, acredita-se que esse *plugin* não é compatível com as versões atuais desse IDE. Aliado a isso não foi possível ter acesso aos códigos fontes desses *plugins*, somente aos fontes do aplicativo *standalone*.

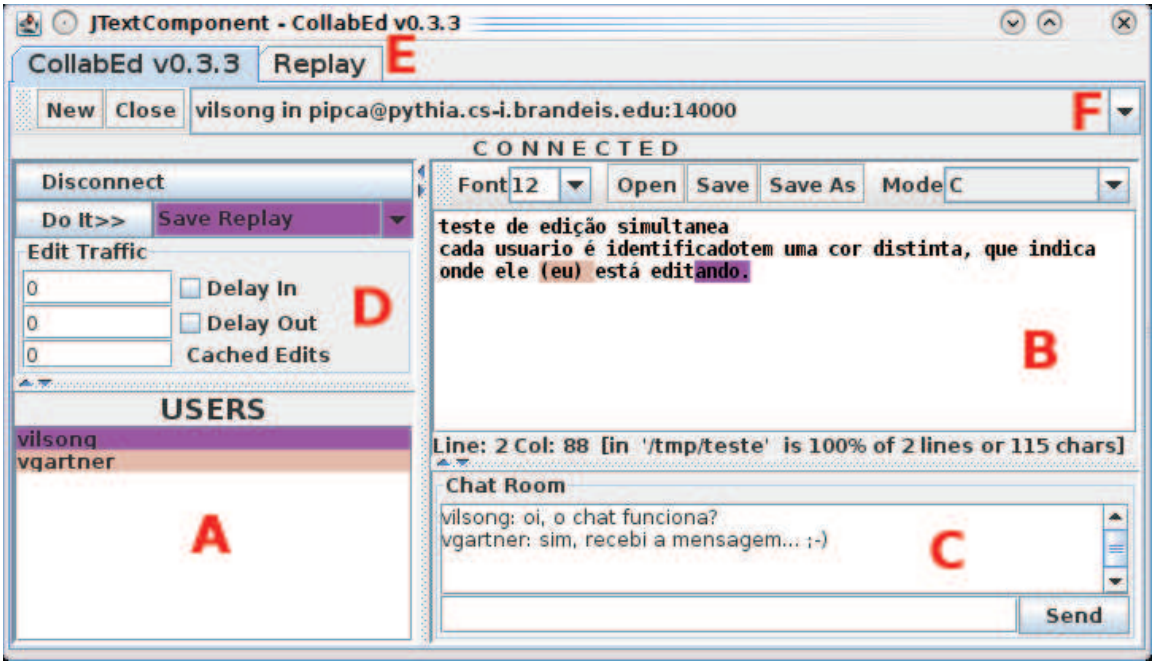


Figura 15: Ambiente do CollabEd

Ao iniciar o aplicativo cliente, é necessário informar a *URL* ou *IP*, e respectiva porta, onde o servidor está sendo executado. Após efetuar a conexão, o usuário poderá verificar, através da lista de usuários, os demais participantes que estão conectados nesse servidor e sessão (Figura 15-A). Quando um usuário sair da sessão ou um novo ingressar, essa lista será atualizada automaticamente pelo programa. Os usuários ativos na sessão poderão se comunicar através de *chat* (Figura 15-C), onde poderão trocar mensagens de texto. As mensagens enviadas serão vistas por todos os usuários conectados.

A edição simultânea é feita no espaço indicado na Figura 15-B. Uma característica dessa ferramenta é o fato de que os usuários podem alterar o mesmo texto. Ou seja, se um usuário estiver alterando a coluna 2 e linha 2, outro usuário também poderá fazê-lo, o que pode levar a certa confusão. Como é possível observar na figura, o texto que está sendo alterado é destacado com a cor do usuário que está fazendo essa alteração. À medida que um usuário alterar o texto, imediatamente as alterações são apresentadas aos demais participantes. Todavia, esse comportamento pode ser configurado no painel indicado pela Figura 15-D. Nessas configurações pode-se indicar se o aplicativo não deve: enviar aos demais participantes as alterações feitas localmente; não aplicar no texto local as alterações feitas pelos outros usuários; ou ainda, que o programa crie um *cache* e, somente quando ele estiver cheio, aplique as alterações no texto. O campo de seleção, indicado na Figura 15-F, apresenta os demais servidores nos quais podem ser abertas sessões.

Neste aplicativo, o servidor é o encarregado de manter o histórico das alterações efetuadas nos clientes. Graças a este histórico, mesmo que os clientes mantenham as alterações em um buffer, as mesmas serão aplicadas corretamente pelo servidor, que se encarregará de atualizar a posição onde as alterações devem acontecer no documento remoto.

Conforme mencionado anteriormente, uma das características principais desse ambiente é o fato dele possuir em sua plataforma, um componente de *replay*. Graças a essa característica, um usuário poderá executar passo-a-passo todas as modificações que ocorreram no texto. Da mesma forma, todas as modificações podem ser salvas em arquivos, possibilitando que o passo-a-passo possa ser revisto posteriormente. As funcionalidades de replay são acessadas através da aba destacada na Figura 15-E.

Entre os trabalhos futuros mencionados pelos autores, está a utilização de XML para compor as mensagens do protocolo. Atualmente, as mensagens enviadas pelos ambientes são objetos serializados com as informações.

3.5 Avaliação final dos aplicativos

Todos os aplicativos apresentados neste capítulo demonstram sua importância e contribuição para a essência da proposta, que é o Desenvolvimento Distribuído de *Software*.

O primeiro trabalho apresentado, VIMEE, está mais focado no aspecto de comunicação, possibilitando o gerenciamento de reuniões e trazendo grande contribuição no que tange a procedimentos formais que guiam e organizam o andamento dos trabalhos. Possui uma interface clara e bem compreendida pelos participantes, aspecto importante para tarefas cruciais como atualizar o status e resolver questões do projeto, citados Herbsleb e Moitra (2001).

O segundo trabalho descreve um aplicativo com ferramentas importantes para o desenvolvimento distribuído. Essas funcionalidades atendem perfeitamente o modelo de desenvolvimento *Dispersed XP*, descrito por Carmel e Tjia (2005). Todavia, levando em consideração as demais características e demandas do DDS, a restrição de somente permitir a colaboração entre duas pessoas, aliado aos aspectos técnicos descritos na análise da ferramenta, acabam por limitar a sua utilização. Ainda assim, em termos de ferramentas, este trabalho é o que mais se assemelha com a proposta desta dissertação.

Já o terceiro trabalho apresentado, apesar de ser mais antigo e estar descontinuado, possui um conjunto de ferramentas importantes no contexto deste trabalho. Através de sua análise foi possível observar ferramentas até então não encontradas em outros trabalhos, como

por exemplo, o "*Virtual Building Floor Plan*" que implementa o sentimento de visão 360º, cuja importância é descrita por Carmel (1999). Além disso, outras características presentes neste aplicativo foram consideradas importantes em trabalhos futuros para esta dissertação.

O quarto trabalho destaca-se por apresentar uma solução especialmente voltada para a edição colaborativa de códigos de programas, permitindo não apenas dois, mas inúmeros participantes. A abordagem de edição colaborativa proposta por esse trabalho é a que mais se aproxima da ferramenta de edição proposta por esta dissertação. Todavia, enquanto que esse trabalho utiliza uma arquitetura cliente/servidor, a proposta da dissertação é utilizar uma arquitetura distribuída, sem a necessidade de se conectar a um servidor central que receba, analise e distribua as alterações dos códigos. Aliado a isso, está o fato deste aplicativo não utilizar XML em suas mensagens.

4 TRABALHO PROPOSTO

Este capítulo apresenta o trabalho proposto nesta dissertação. Como resultado final, espera-se a disponibilização de um ambiente de desenvolvimento de *software*, chamado de IdDE (*Integrated and Distributed Development Environment*), com um espectro de ferramentas que auxiliem na minimização dos problemas inerentes ao processo de Desenvolvimento Distribuído de *Software*. Além de implementar ferramentas relevantes no contexto de DDS, pretende-se que o ambiente também seja expansível, permitindo a futura integração de novas funcionalidades e ferramentas.

Neste capítulo serão apresentados, de forma detalhada, o ambiente e suas ferramentas, sua arquitetura, tecnologias utilizadas, além do processo de comunicação e o protocolo IdDE.

4.1 Visão Geral

Olhando sob a perspectiva de um programador, um Ambiente Integrado de Desenvolvimento (IDE) possibilita a escrita de código e fornece uma série de funcionalidades importantes para o processo de desenvolvimento de *softwares*. Nesse sentido, para alcançar os objetivos propostos, acredita-se que, ao invés de desenvolver o ambiente desde o princípio, o ideal seja integrar as ferramentas colaborativas num IDE já existente. Dessa forma, tem-se o benefício de contar com todas as facilidades e características já presentes neste ambiente, o que torna a ferramenta mais completa. Igualmente, acredita-se que esta decisão também contribua positivamente tanto no processo de desenvolvimento distribuído, quanto na aceitação e adoção da ferramenta pelos desenvolvedores.

A definição das ferramentas que serão implementadas ocorreu com base na revisão bibliográfica, o que possibilitou que se tivesse uma visão mais abrangente, além de uma melhor compreensão das dificuldades e necessidades trazidas pelo DDS. Outrossim, a análise de trabalhos relacionados auxiliou na compreensão dos benefícios trazidos pelas soluções propostas.

As funcionalidades que serão implementadas no ambiente proposto neste trabalho são: edição simultânea e colaborativa de código, versionamento de código, *chat* utilizando mensagens de texto e suporte à tradução, controle distribuído de tarefas, agenda, troca de arquivos e comunicação através de áudio (Figura 16).

Aliado a isso, também é objetivo do presente trabalho a criação de um protocolo XML aberto e documentado, que será utilizado para comunicação e negociação entre os ambientes.

Por negociação, entende-se o envio de mensagens entre os ambientes, contendo instruções ou comandos, avisos, solicitação de participação numa sessão de edição compartilhada, entre outros. O protocolo é detalhado na seção 4.4, juntamente com o processo de comunicação entre os ambientes.

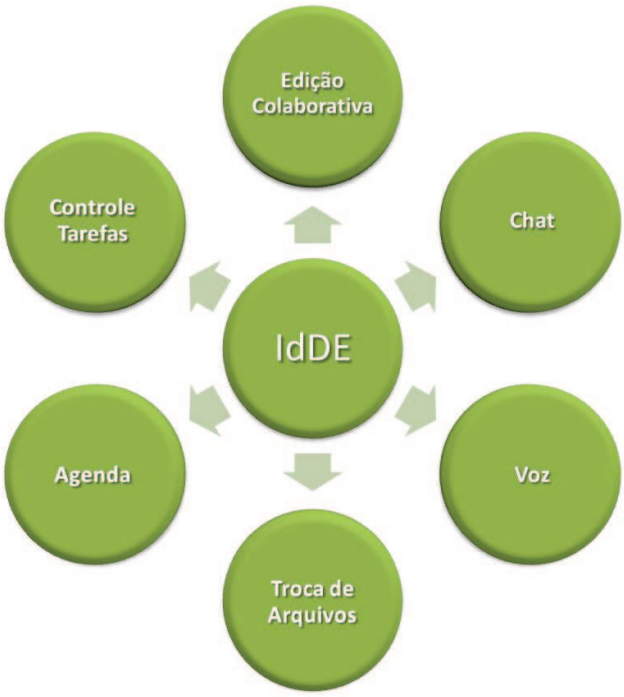


Figura 16: Ferramentas do ambiente IdDE

Um aspecto importante, em se tratando de desenvolvimento distribuído, é a observância do sistema operacional utilizado nos equipamentos dos usuários. Tão importante quanto a criação de uma ferramenta que auxilie no DDS, é garantir que os usuários possam utilizá-la no sistema operacional de sua preferência. Nesse sentido, pretende-se que o IdDE possa ser executado tanto no Sistema Operacional Windows[8] quanto Linux[9].

Graças às tecnologias adotadas, descritas na seção 4.2, o IdDE poderá se comunicar também com outros dispositivos, não estando limitado apenas ao ambiente proposto. A Figura 17 mostra, de forma genérica, os dispositivos que poderão ser utilizados. Observando a figura, percebe-se que a comunicação via áudio poderá ocorrer entre usuários do ambiente IdDE e dispositivos de telefonia convencional, telefonia móvel, telefones IP, softphones, ATAs e *smartphones*. Já a edição compartilhada poderá ocorrer entre aplicativos que implementem o protocolo de comunicação e negociação do IdDE. O chat, por sua vez, poderá ser utilizado em

[8] http://www.windows.com
[9] http://www.linuxfoundation.org

dispositivos que possuam *softwares* que utilizem o mesmo protocolo de comunicação adotado ambiente.

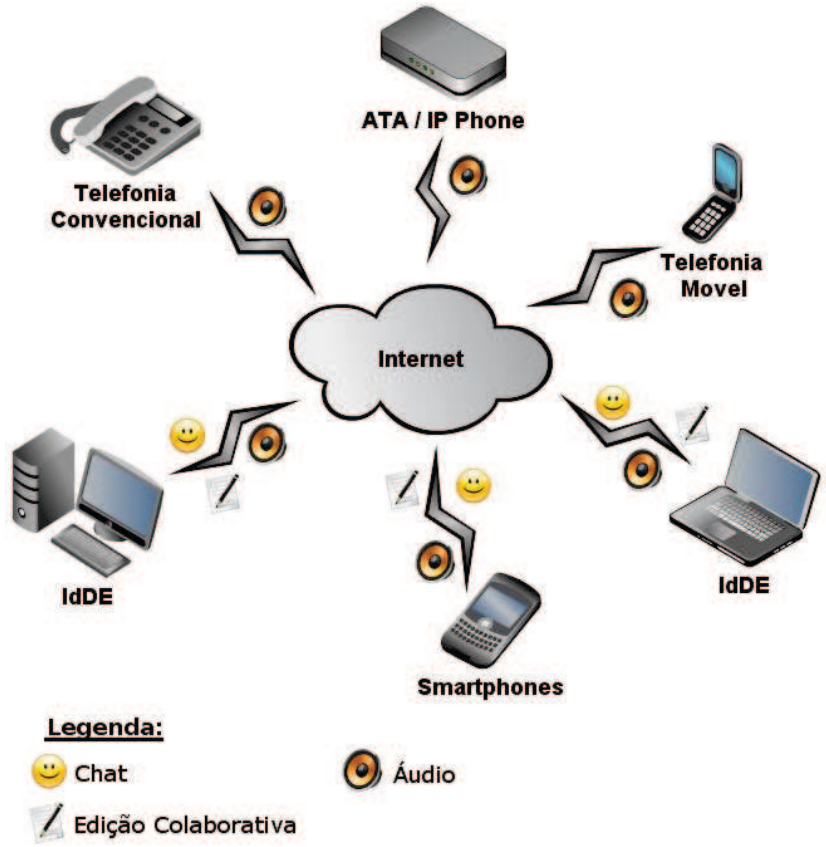


Figura 17: Formas de comunicação

A Tabela 2 apresenta o agrupamento de funcionalidades de acordo com os dispositivos e *softwares*, demonstrando a interoperabilidade do ambiente. Analisando a tabela, é possível observar que, além do ambiente IdDE, todas as demais funcionalidades poderiam ser desenvolvidas em *smartphones*. Isso se deve graças a possibilidade do desenvolvimento de aplicativos com suporte aos protocolos do ambiente nesses dispositivos. Assim, por exemplo, poderia se utilizar esse dispositivo móvel para acompanhar a, ou mesmo colaborar na, edição de um código fonte de um programa. Também é possível observar, na tabela, que o áudio é a ferramenta que pode ser utilizada na maior quantidade de dispositivos e/ou *softwares*.

Tabela 2: Funcionalidades versus dispositivos e *softwares*

Funcionalidade	Edição Compartilhada	Chat	Áudio	Controle Tarefa	Agenda	Troca Arquivos
IdDE	✓	✓	✓	✓	✓	✓

Funcionalidade	Edição Compartilhada	Chat	Áudio	Controle Tarefa	Agenda	Troca Arquivos
Smartphone	✓	✓	✓	✓	✓	✓
Celular			✓			
Telefone Fixo			✓			
Softphone / IPPhone			✓			
ATA			✓			
Software Chat		✓				✓
Software que implementar protocolo IdDE	✓	✓		✓	✓	✓
Software que implementar SIP			✓			

Por outro lado, é importante observar que a interoperabilidade não está restrita aos dispositivos e *softwares* apresentados na figura e tabela anteriores. Nesse sentido, todo aplicativo que utilize o protocolo XMPP para a troca de mensagens, poderá se comunicar com o ambiente. Esta comunicação engloba tanto o envio de mensagens de *chat*, quanto mensagens de negociação, desde que encapsulem o protocolo de comunicação do IdDE. Da mesma forma, aplicativos e dispositivos que utilizem o protocolo SIP poderão se comunicar via áudio com usuários do ambiente.

4.2 Tecnologias Adotadas

Tão importante quanto a definição das ferramentas que serão desenvolvidas no projeto, é a opção tecnológica para seu desenvolvimento.

Durante os primeiros meses de desenvolvimento deste trabalho, ocorreu intensa atividade de pesquisa, estudo e experimentação. O objetivo dessa fase foi o de identificar e validar as tecnologias necessárias para conseguir atingir os propósitos. Outro aspecto importante desse período foi a sua contribuição no sentido de proporcionar uma familiarização com as tecnologias escolhidas. Nesse sentido, foram criados protótipos, que permitiram identificar dúvidas e antecipar problemas que poderiam se tornar fatores complicadores ou ainda inviabilizar o projeto.

4.2.1 Linguagem de Programação – Java

Uma das primeiras decisões tomadas foi a escolha da linguagem de programação. Como o objetivo é o de possibilitar que o ambiente seja executado nas mais variadas plataformas operacionais, a linguagem de programação adotada foi Java[10], que, além de multiplataforma, é uma linguagem largamente utilizada e difundida.

Atualmente, a linguagem Java é utilizada para desenvolver aplicações de larga escala, sistemas para a *web*, além de possibilitar o desenvolvimento para dispositivos móveis entre outras aplicações [Deitel e Deitel, 2006].

4.2.2 IDE – Netbeans

Após o estabelecimento das características do ambiente e a definição da linguagem de programação, o passo seguinte foi a escolha do IDE que seria utilizado como base para a integração das ferramentas propostas no presente trabalho.

Cheng *et al.* (2003) advertem, porém, que integrar os aspectos de colaboração num IDE aumenta o número de desafios, incluindo requisitos para expansão do ambiente, acesso aos modelos estruturais do IDE, interoperabilidade e infraestrutura de rede. Nesse sentido, mantendo presente o propósito de oferecer ao usuário uma ótima experiência em termos de programação, além de possibilitar a expansão do ambiente, optou-se pela plataforma Netbeans [Netbeans, 2010]. Este ambiente já oferece todas as funcionalidades necessárias para a edição de código fonte, além de disponibilizar inúmeras outras ferramentas, a exemplo de: formatação de código, dicas e sugestões de códigos automáticos, compilação, depuração, criador visual de formulários, suporte à internacionalização, suporte a diversas linguagens de

[10]Java: http://www.java.com/pt_BR

programação, entre outros. Além disso, novas funcionalidades podem ser desenvolvidas e integradas ao ambiente, através da utilização de *plugins*. Aliado a isso, este *software* atende o requisito de poder ser executado nos dois sistemas operacionais desejados (Windows e Linux).

O Netbeans possui uma arquitetura modular, como pode ser observado na Figura 18. Um módulo é um componente de *software* auto-suficiente, que pode se comunicar com outros módulos através de interfaces bem definidas e que escondem as suas implementações. Todas as características de um aplicativo são criadas dentro de um módulo e, assim, ficam claramente definidos os limites e responsabilidades de cada módulo sobre o aplicativo [Petri, 2010]. Dessa forma, a ferramenta proposta, IdDE, será desenvolvida como um módulo do Netbeans, em concordância com as regras e requisitos determinados por esse ambiente.

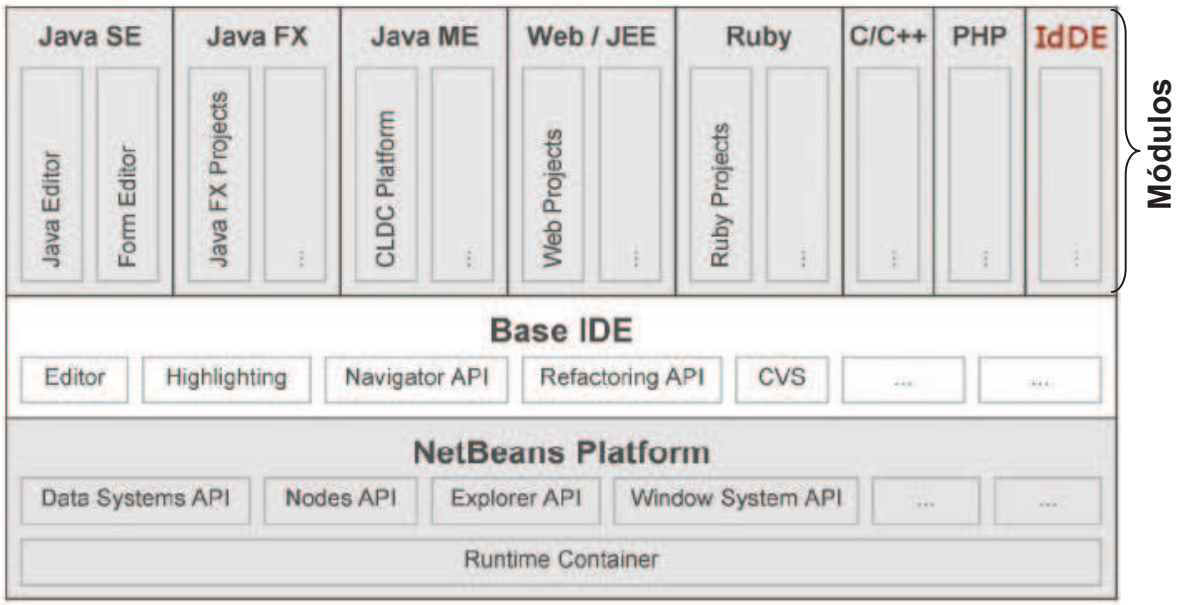


Figura 18: Arquitetura modular do Netbeans, adaptada de [Böck, 2009]

Por outro lado, é sabido que existem aplicativos com características similares às do Netbeans, como por exemplo, o Eclipse[11]. Entretanto, a opção pelo Netbeans se deve à familiaridade e experiências anteriores com essa IDE. Além disso, mesmo antes do início do desenvolvimento da aplicação, foram identificados possíveis participantes para os estudos de caso, e os mesmos também utilizavam esse ambiente, fato que auxiliaria nessa etapa.

[11] http://www.eclipse.org

Todavia, é importante ressaltar que, apesar de o IdDE ser desenvolvido como um módulo do Netbeans, o mesmo também poderá ser facilmente integrado a outros editores, conforme será visto no na seção 4.3, que trata sobre a arquitetura do ambiente.

4.2.3 Protocolos de Comunicação e Transporte

Um dos elementos fundamentais para o funcionamento do ambiente proposto é a comunicação. Nesse sentido, antes de iniciar o desenvolvimento propriamente dito do ambiente, foram criados pequenos protótipos cliente/servidor para testar e validar aspectos inerentes à comunicação. Com o auxílio desses protótipos, foram identificados os primeiros desafios que, de certa forma, acabaram por direcionar as decisões seguintes. Tais desafios estavam, primariamente, relacionadas em compreender e superar questões relativas à segurança de rede, como *firewall* e NAT. Como pretende-se que usuários de uma rede local possa se comunicar com usuários distribuídos em outras redes, através da internet, isso deverá acontecer de forma transparente, sem necessitar conhecimentos avançados nessa área.

Os resultados obtidos nos testes práticos com os protótipos indicaram que a melhor opção seria utilizar um protocolo já existente, e cujas portas de comunicação fossem conhecidas e difundidas, ao invés de criar um novo protocolo de transporte. Assim, além de contribuir para o processo de comunicação entre redes heterogêneas, a utilização de um protocolo conhecido permitiria a utilização de servidores já existentes, ampliando assim, inclusive, a interoperabilidade do IdDE com outros aplicativos que utilizem o mesmo protocolo.

Conforme descrito nas seções 4.2.3.1 e 4.2.3.2, ficou estabelecida a utilização de dois protocolos: XMPP e SIP. O protocolo SIP para comunicação via áudio, e o protocolo XMPP para as mensagens de texto entre usuários e o transporte das mensagens do protocolo de comunicação do ambiente IdDE.

4.2.3.1 Protocolo XMPP

O protocolo XMPP (eX*tensible Messaging and Presence Protocol*) foi escolhido por ser um protocolo aberto e largamente utilizado para a comunicação. Exemplos bastante

conhecidos de seu uso são o Google Talk[12] e Facebook[13], além de inúmeros outros servidores públicos disponíveis na internet. Aliado a isso, é possível também implantar um servidor privado em redes locais, o que aumenta o sigilo e segurança das informações.

Este protocolo originou-se a partir do Jabber e utiliza o XML para a comunicação [XMPP, 2010]. Além do uso de XML, o XMPP utiliza alguns conceitos importantes na sua estrutura [Miller, 2001]:

Identidade – identifica cada recurso do sistema e é composto por três partes: usuário, servidor e recurso. O usuário e servidor são formatados da mesma forma como no e-mail convencional: usuario@servidor. Já o recurso, normalmente identifica o aplicativo que está sendo utilizado. A identidade final é algo como: usuario@servidor/aplicativo.

Presença – esta propriedade é fundamental, pois através dela um usuário fica sabendo o status dos demais usuários. O status pode ser: conectado, desconectado, ocupado, disponível, entre outros.

Roster – esta é outra característica essencial nos serviços de mensagens instantâneas, e é mais conhecido como lista de contatos ou lista de amigos. Através da lista é possível ter acesso às informações dos contatos de um usuário, como por exemplo, o status dos seus amigos: se estão conectados, desconectados, entre outros.

A tecnologia XMPP utiliza uma arquitetura cliente/servidor descentralizada, similar às arquiteturas utilizadas nas redes WWW e de *e-mail* [Saint-Andre *et al*, 2009]. A Figura 19 mostra uma representação simplificada dessa arquitetura. Nela, pode-se observar que cada servidor possui três clientes conectados. Estes clientes, por sua vez, podem se comunicar tanto com usuários do mesmo servidor quanto com usuários de outros servidores, uma vez que os mesmos estão conectados entre si. Essa característica de possibilitar a integração de servidores distintos é chamada de XMPP *Federation*.

Outra peculiaridade interessante desse protocolo é a possibilidade de efetuar conexões simultâneas em diversos programas, utilizando o mesmo usuário e senha. Diferentemente do que ocorre com protocolos que desconectam o usuário quando este efetua *login* em outro aplicativo, o XMPP permite a conexão simultânea e efetua a entrega das mensagens em todos os programas. Esta característica permite que se acompanhem as mensagens enviadas entre os ambientes.

---

[12] Google Talk: http://www.google.com/talk
[13] Facebook: http://www.facebook.com

As portas de comunicação utilizadas pelo protocolo XMPP são a 5222 TCP, para conexão não segura e 5223 TCP para conexões seguras utilizando TLS.

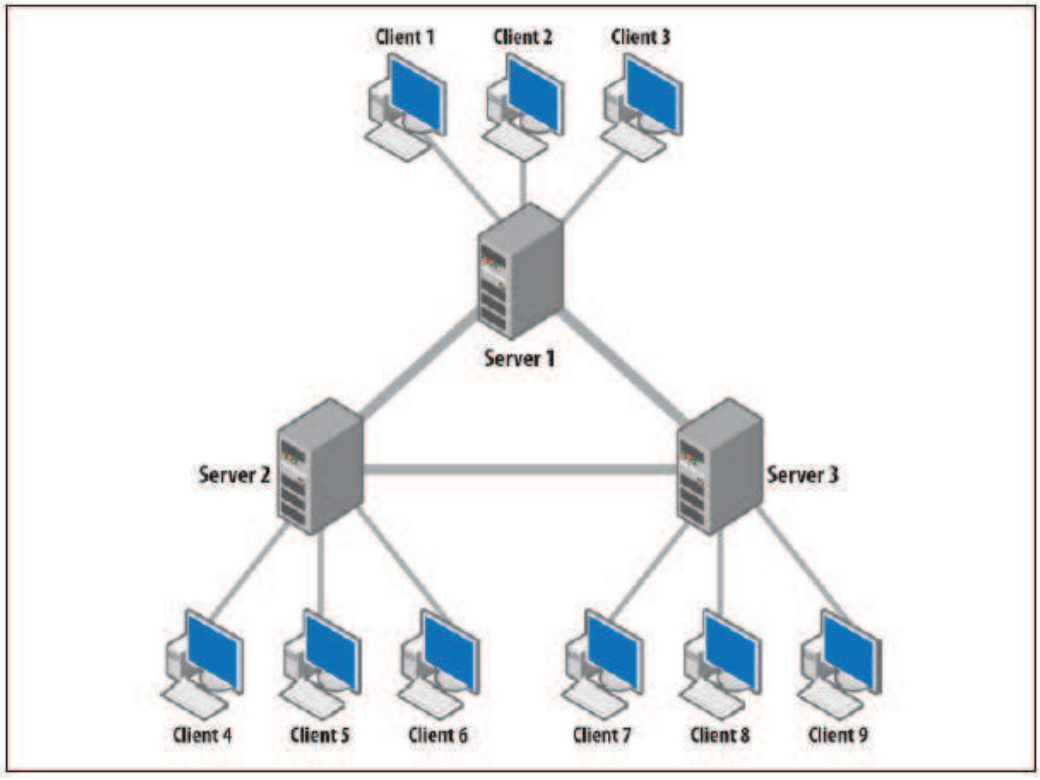


Figura 19: Arquitetura da tecnologia XMPP [Saint-Andre et al, 2009]

Uma vez que esteja estabelecida uma sessão entre o cliente e o servidor, ambos poderão trocar três tipos básicos de mensagens, também chamadas de *Stanzas* [Saint-Andre *et al*, 2009]. A Tabela 3 descreve e exemplifica essas *stanzas*.

Tabela 3: XMPP *Stanzas*

Stanza	**Objetivo**	**Exemplo**
<message>	Utilizado para as mensagens instantâneas (IM), alertas, notificações e outras aplicações desse tipo.	<message to=’vgartner@gmail.com’ from=’vilson@vgdata.net/IdDE’ type=’chat’> <body>E aí, tudo bem? </body> </message>
<presence>	Está é uma característica importante para aplicativos de tempo real. Ela é utilizada	<presence from="vilson@vgdata.net/IdDE"> <show>away</show>

Stanza	Objetivo	Exemplo
	para anunciar a disponibilidade do usuário para comunicação.	<status>Tô com fome: no almoço</status> </presence>
<iq>	Esta *stanza* (Info/Query) fornece uma estrutura para interações do tipo solicitação/resposta, similar aos métodos PUT e GET do HTTP.	<iq from="vilson@vgdata.net/IdDE" id="rr82a1z7" to="vgartner@gmail.com" type="get"> <query xmlns="jabber:iq:roster"/> </iq>

A *stanza <message>* é utilizada para o envio das instruções do protocolo do ambiente IdDE, descrito na seção 4.4. O exemplo da *stanza <iq>* mostra a solicitação da lista de contatos do usuário vilson@vgdata.net para o usuário vgartner@gmail.com.

A comprovação das funcionalidades e do potencial do protocolo XMPP foi obtida através da criação de um protótipo inicial. Através deste protótipo, foi possível enviar mensagens de texto, nas quais estavam encapsuladas mensagens do protocolo do IdDE. Todavia, este protótipo também apontou um problema significativo. Apesar de o protocolo XMPP permitir a comunicação via áudio, as bibliotecas disponíveis na linguagem Java possuem um suporte muito incipiente a este recurso, o que impossibilitou o estabelecimento de comunicação via áudio.

Face ao problema identificado, o qual inviabilizou a utilização deste protocolo para a comunicação via áudio, iniciou-se uma nova etapa de testes e experimentações em busca de outro protocolo que pudesse ser utilizado e que tivesse maior suporte de bibliotecas na linguagem de programação Java.

4.2.3.2 Protocolo SIP

Dadas as dificuldades de comunicação via áudio no protocolo XMPP, foram feitos testes e experimentos para avaliar o protocolo SIP (*Session Initiation Protocol*), utilizado em comunicações do tipo VoIP.

Segundo Davidson *et al.* (2006), SIP é um protocolo que controla a negociação, modificação e encerramento de uma sessão multimídia interativa. As sessões multimídia podem ser de áudio, vídeo, *chat* ou mesmo sessões de jogos. Além disso, o SIP possui

extensões para possibilitar mensagens instantâneas, presença e notificação de eventos. Nos testes realizados, a comunicação via áudio foi bem sucedida, demonstrando se tratar de uma opção viável para utilização no IdDE. Todavia, o suporte a mensagens instantâneas, presença e notificação de eventos não se mostrou tão eficaz quanto no protocolo XMPP, o que impossibilitou o seu uso como protocolo único para todos os serviços do IdDE. Dessa forma, a solução encontrada para suprir as necessidades propostas neste trabalho, foi a utilização de ambos os protocolos, XMPP e SIP.

De acordo com Sinnreich e Johnston (2006), as comunicações IP utilizam endereços similares aos e-mails, a exemplo de sip:user@servidor.com, chamados de SIP URIs. Pode-se observar que se trata do mesmo tipo de formatação utilizada no protocolo XMPP. Para os casos nos quais a comunicação deva acontecer com um número de telefone, o formato do endereço SIP é: sip:+555198765432@servidor.com. Neste endereço, +55 representa o país, 51 o código de área e 98765432 o número do telefone. Esta formatação se aplica tanto a telefones convencionais quanto de telefonia móvel.

Existem diversos tipos de dispositivos que podem ser utilizados na comunicação SIP, como por exemplo: computadores, telefones IP, ATAs, dispositivos móveis, além de telefones convencionais e celulares (PSTN). Todavia, é importante observar que, para efetuar chamas oriundas da, e para a rede PSTN (rede pública de telefonia comutada), faz-se necessária sua interconexão com a rede SIP. Esse papel é desempenhado pelos gateways de empresas prestadoras de serviços de telefonia [Sinnreich e Johnston, 2006]. A Figura 20 exemplifica a estrutura e interconexão das redes PSTN e SIP, e seus respectivos dispositivos.

Dispositivos IP com suporte a SIP podem chamar uns aos outros diretamente, desde que saibam o endereço. Assim, uma chamada de telefone IP pode ser feita diretamente entre dois ou mais telefones SIP ou computadores. Inclusive, pequenas conferências podem ser realizadas por vários usuários se estes se conectarem a um dispositivo que atue como ponte da conferência. O papel de ponte pode ser desempenhado por um dos telefones SIP, que passará a ser tanto participante da conferência, quanto ponte da conferência. [Sinnreich e Johnston, 2006].

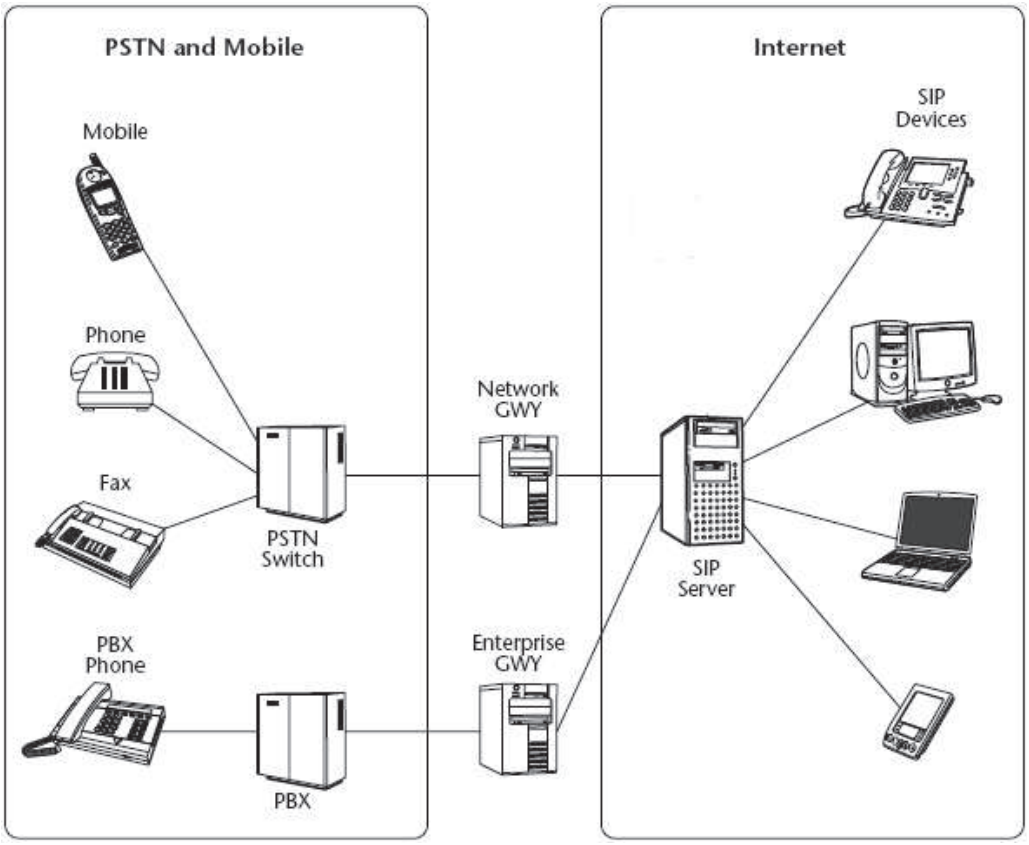


Figura 20: Interconexão de redes SIP e PSTN, adaptada de [Sinnreich e Johnston, 2006]

Os elementos básicos do protocolo são, de acordo com Davidson *et al. (*2006):

- *User Agents* (UA) – trata-se de uma função lógica que, na rede SIP, inicia ou responde às solicitações de transações SIP. O *User Agent Client* (UAC) é quem faz requisições SIP ou recebe respostas SIP, enquanto que o *User Agent Server* (UAS) aceita as requisições SIP e gera as respostas SIP;
- *Proxy* – é uma entidade intermediária na rede SIP, responsável por encaminhar requisições SIP para os respectivos UAS, ou outro *proxy;*
- *Registrar Server* – é um UAS que recebe solicitações de autenticação e é responsável por manter atualizadas as informações dos UA's;
- *Redirect Server* – é um UAS que direciona o UAC a contatar um URI alternativo.

As mensagens, chamadas de requisições SIP, são enviadas pelo cliente para o servidor, com o objetivo de realizar uma operação. O protocolo é baseado em mensagens de texto, similares ao HTTP e ao SMTP. Na Figura 21 é apresentado um exemplo de uma mensagem

SIP. Conforme pode ser observado no SIP URI (sip:john@192.190.132.31), esta requisição, do tipo INVITE, é destinada ao usuário John, que está conectado num dispositivo com o endereço IP 192.190.132.31. O envio ocorre através de UDP e, como a porta não foi informada, é utilizada a porta padrão 5060. É importante observar que a comunicação também pode ocorrer através de TCP, portas 5060 e 5061, esta última para conexão segura via TLS.

```
INVITE sip:john@192.190.132.31 SIP/2.0
Via: SIP/2.0/UDP 10.11.12.13;branch=z9hG4bK776asdhds
Max-Forwards: 70
To: "John" <sip:john@192.190.132.31>
```

Figura 21: Exemplo de mensagem do protocolo SIP [Hersent, 2010]

A RFC 3261[14] define seis requisições [Davidson *et al,* 2006]:

- INVITE: método utilizado para convidar o destinatário a participar de uma sessão;
- ACK: utilizado para indicar que o UAC recebeu a resposta a um INVITE que enviou;
- OPTIONS: utilizado para consultar as funcionalidades suportadas pelo UAS;
- BYE: utilizado para finalizar a sessão estabelecida;
- CANCEL: utilizado para cancelar uma requisição em andamento, a exemplo de um INVITE;
- REGISTER: utilizado para registrar um usuário num servidor SIP.

Normalmente, apenas o estabelecimento de uma sessão é feita através da porta 5060. Quando a sessão está estabelecida, são utilizadas portas entre 10.000 e 20.000 para a transmissão do áudio em tempo real.

A Figura 22 mostra o fluxo de mensagens trocadas entre os usuários e o servidor, a partir do momento em que um usuário solicita o estabelecimento de uma sessão de voz com outro usuário.

[14]RFC 3261: http://www.ietf.org/rfc/rfc3261.txt

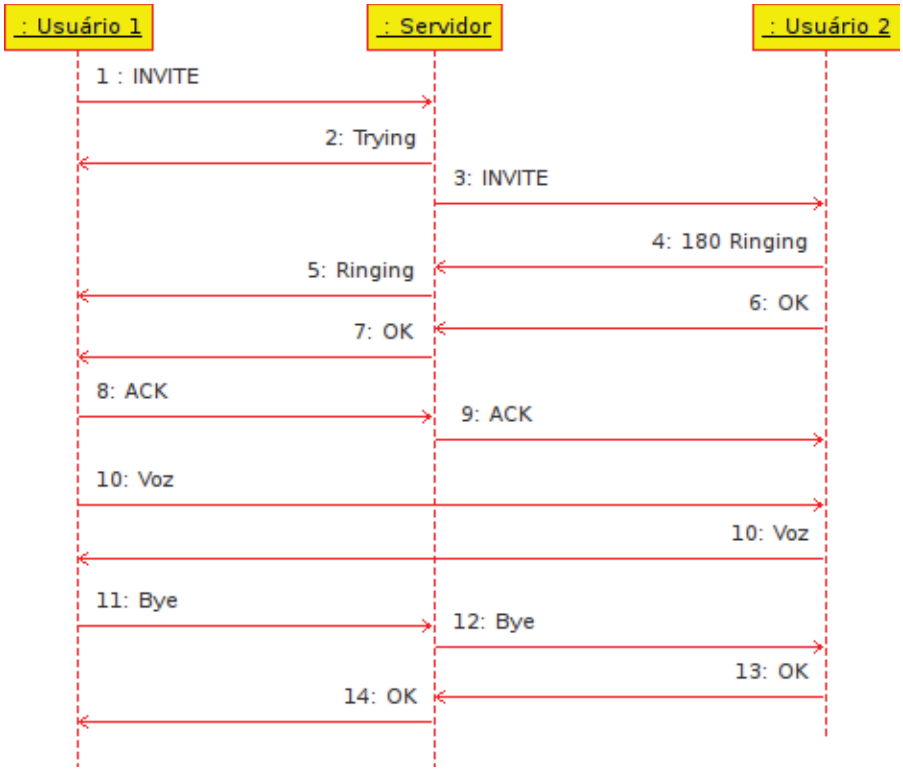


Figura 22: Fluxo de uma chamada SIP

A análise do fluxo deve ser feita de cima para baixo, e a ordem das mensagens acompanha as numerações:

1. O Usuário 1 envia uma mensagem ao servidor, convidando o Usuário 2 a estabelecer uma sessão.
2. O servidor responde informando que a solicitação está em andamento.
3. O servidor encaminha a solicitação ao Usuário 2.
4. Se o Usuário 2 estiver conectado, o servidor receberá a resposta dando conta que ele está sendo chamado.
5. O servidor encaminha a informação ao Usuário 1.
6. Se o Usuário 2 aceitar a sessão, a respectiva mensagem é enviada ao servidor.
7. A mensagem de aceite é então repassada ao Usuário 1.
8. O Usuário 1 envia uma mensagem indicando que recebeu a resposta relativa ao seu comando INVITE (1).
9. A mensagem é repassada ao Usuário 2, para que na sequencia se estabeleça a sessão.
10. A sessão está estabelecida e os usuários podem conversar.
11. O Usuário 1 envia uma mensagem para finalizar a sessão.
12. O servidor encaminha a mensagem de fim de sessão para o Usuário 2.

13. O Usuário 2 responde positivamente e encerra a sessão.
14. O servidor repassa a mensagem ao Usuário 1, e também encerra a sessão.

4.3 Arquitetura do Sistema

Conforme já mencionado anteriormente, o sistema possui uma arquitetura modular e é desenvolvido utilizando a estrutura do Netbeans. A Figura 23 mostra de forma mais detalhada a arquitetura do ambiente. A interface e funções de interação com o usuário serão criadas independentes do IDE, porém, serão integradas ao mesmo através de sua arquitetura de *plugins*. Dessa forma, em cada IDE que se queira utilizar o ambiente de desenvolvimento, será necessário criar a camada de *plugin* (na figura, representado por "Interface"). Essa camada será a única parte do ambiente que terá alguma dependência do editor. Assim, interação do usuário com o ambiente se dará através das opções disponibilizadas pelo IdDE, e a interação deste com os editores será através da arquitetura de *plugins* dos mesmos.

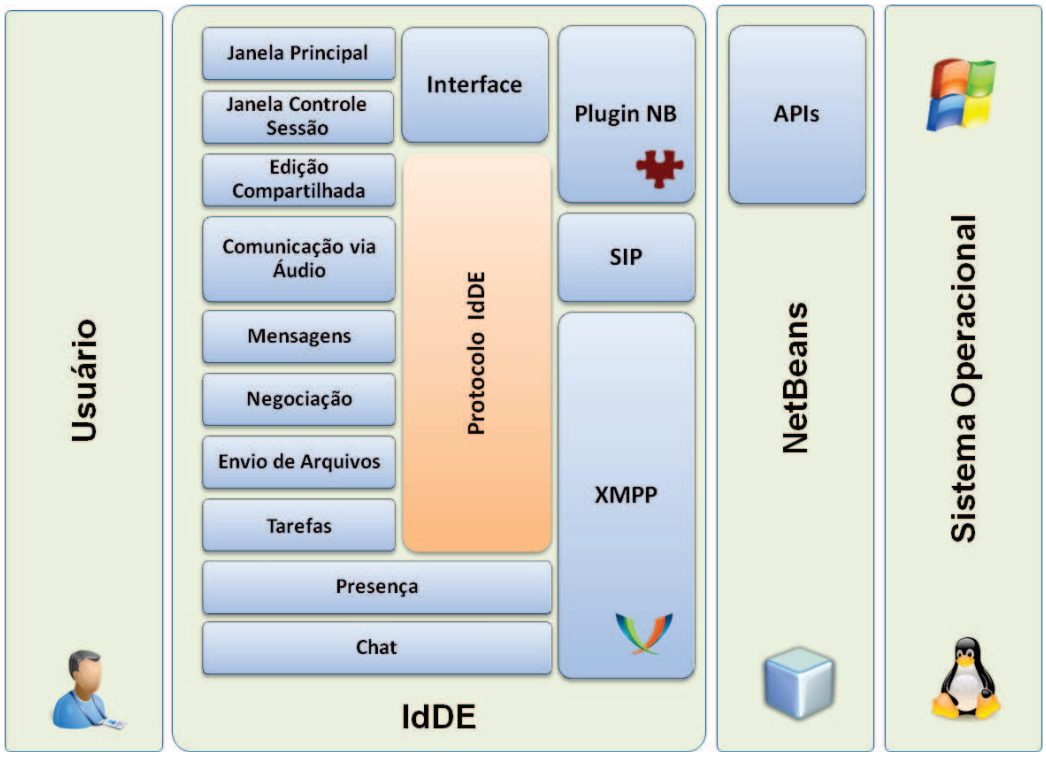


Figura 23: Arquitetura do sistema

Analisando a figura, é possível observar que as únicas funcionalidades do IdDE que estão diretamente vinculadas à camada de *plugins* do Netbeans, e utilizando as funcionalidades de sua API, são a edição compartilhada e a camada de interface, que está

relacionada às janelas Principal e Controle de Sessões de edição. Essas janelas são criadas quando o módulo é carregado no Netbeans, o que ocorre durante a inicialização deste aplicativo. Assim, pode-se concluir que as funcionalidades de edição e controle de interface são as únicas que possuem dependência direta do Netbeans e, dessa forma, deverão ser reescritas para utilização do ambiente em outro IDE.

Observando a figura, pode-se verificar também que (praticamente todas) as funções acionadas pelo usuário fazem uso do Protocolo IdDE. Isso se deve ao fato de que o ambiente envia mensagens de negociação para o ambiente remoto, solicitando ou informando a ação do usuário local. Pode-se concluir assim que, para que um aplicativo possa se comunicar com o IdDE, deve implementar, obrigatoriamente, esse mesmo protocolo. Exemplificando, caso um usuário queira enviar uma nova tarefa para outro usuário, o aplicativo utilizado por aquele usuário deverá criar uma mensagem do protocolo IdDE. Por fim, a figura mostra que as funcionalidades de presença e *chat* interagem diretamente através de XMPP, não havendo a troca de mensagens ou utilização do protocolo IdDE.

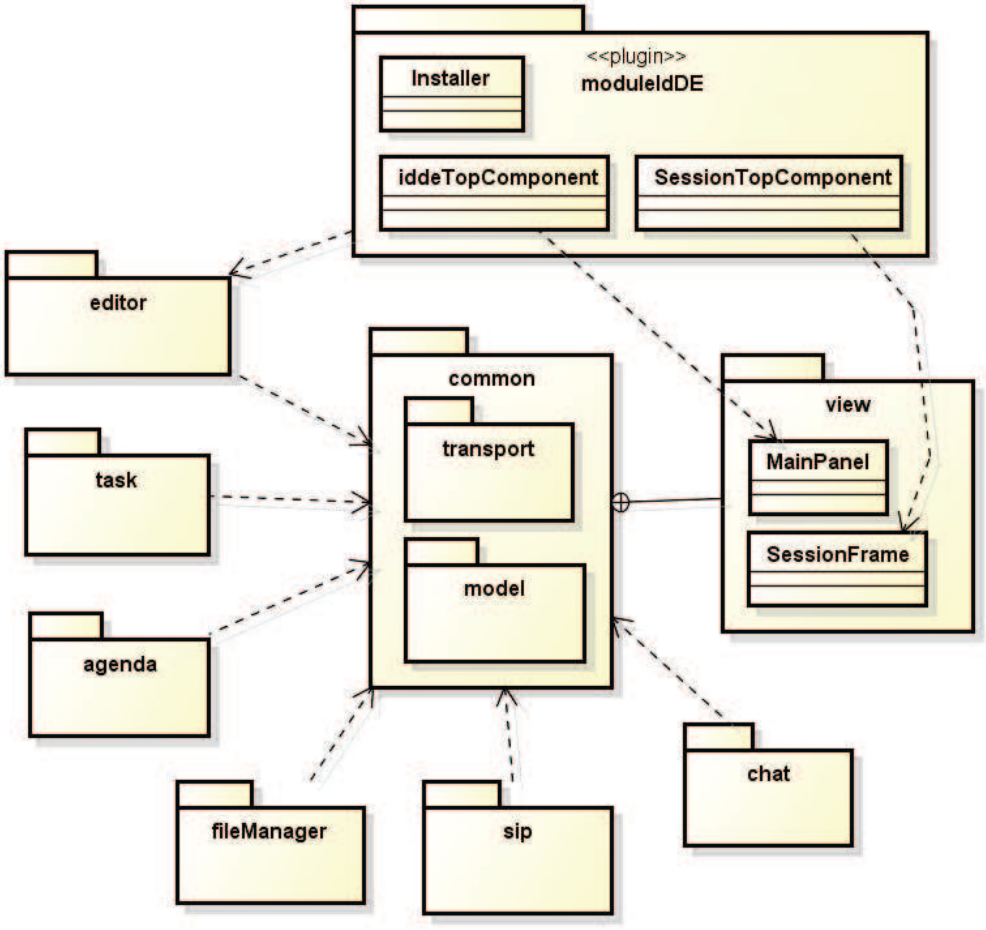


Figura 24: Estrutura de pacotes de classes do IdDE

A Figura 24 procura representar a visão lógica (macro) da organização do sistema e suas ferramentas. No diagrama pode-se observar que os pacotes de classes foram organizados de forma a refletir a arquitetura modular apresentada na Figura 23.

O nome de cada pacote faz referência à funcionalidade que implementa, e o detalhamento dessas funcionalidades é apresentado na Tabela 4. Todos os pacotes possuem subpacotes e o seu detalhamento é feito na descrição das ferramentas do ambiente, nas próximas seções.

Tabela 4: Pacotes e suas respectivas funcionalidades

Pacote	Funcionalidade
editor	Possui as classes responsáveis pela implementação da edição compartilhada e simultânea.
task	Neste pacote estão as classes responsáveis pelo gerenciamento de tarefas.
agenda	Agrega as classes relacionadas à agenda de compromissos.
fileManager	Classes responsáveis pela transferência de arquivos entre os usuários.
sip	Possui as classes que implementam a comunicação via áudio.
common	Este pacote agrega funcionalidades comuns, necessárias aos demais pacotes. Este pacote possui os subpacotes *view*, *transport* e *model*.
common::view	Neste pacote estão localizadas as classes responsáveis pela implementação da tela principal e de configuração do IdDE (*MainPanel*), e da tela com informações das sessões de edição colaborativa ativas (*SessionFrame*).
common::transport	Agrega as classes de definição de mensagens do protocolo IdDE, além de criação e interpretação das mensagens desse protocolo.
common::model	Possui as classes que implementam as funcionalidades relacionadas ao XMPP, como conexão, envio de mensagens e presença.
moduleIdDE	Neste pacote ficam localizadas as classes responsáveis pela

Pacote	Funcionalidade
	integração do IdDE ao Netbeans e pela interação com o documento de edição de código fonte. As principais classes são: *Installer*, responsável pela "instalação" e ativação do módulo no Netbeans; *IddeTopComponent*, responsável pela apresentação da tela principal do IdDE no Netbeans; e *SessionTopComponent*, responsável por apresentar a tela com informações das sessões de edição compartilhada existentes.

4.4 Comunicação e Protocolo IdDE

Conforme descrito anteriormente, a comunicação no ambiente IdDE utiliza dois protocolos: XMPP e SIP. O protocolo SIP é utilizado exclusivamente para comunicação via áudio, enquanto que o protocolo XMPP é utilizando tanto para trocas de mensagens instantâneas (*chat*) e presença quanto para comunicação e troca de mensagens de protocolo do próprio ambiente.

Como são utilizados dois protocolos, é necessária a utilização de dois servidores (serviços) distintos, específicos para cada tipo protocolo. No IdDE, os papéis de servidores XMPP e SIP podem ser desempenhados por servidores disponíveis na internet, ou ainda é possível implementar servidores privados que forneçam os serviços. Exemplos de *softwares* conhecidos são OpenFire[15] para XMPP e Asterisk[16] para SIP, todavia, como se tratam de protocolos abertos e altamente documentados, é possível também implementar um servidor desde o princípio.

Graças à adoção de dois protocolos, é possível utilizar diferentes configurações de servidores: pode-se utilizar um único equipamento ou ainda pode-se separar os serviços em distintos equipamentos, o que contribui também para o aumento da escalabilidade da solução. A Figura 25 exibe a configuração na qual ambos os serviços estão instalados no mesmo equipamento (server.vgdata.net). Nesta situação tanto a autenticação do XMPP quanto SIP acontecerá no mesmo computador. A única diferença será em relação às portas, uma vez que a porta padrão do SIP é a 5060 e a porta padrão do XMPP é a 5222.

[15] http://w ww.igniterealtime.org/projects/openfire/
[16] http://www.asterisk.org

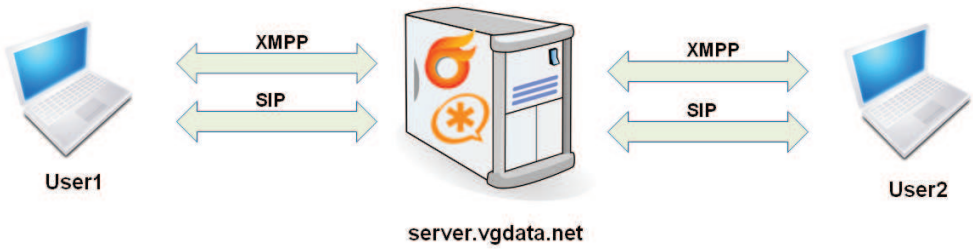


Figura 25: Configuração utilizando um servidor

Já a Figura 26 exemplifica a utilização de dois equipamentos. Nesta situação, o serviço XMPP é provido pelo servidor xmpp.vgdata.net enquanto que o serviço SIP é provido pelo servidor sip.vgdata.net. A utilização de dois servidores possibilita ainda que os mesmos estejam em redes ou locais distintos.

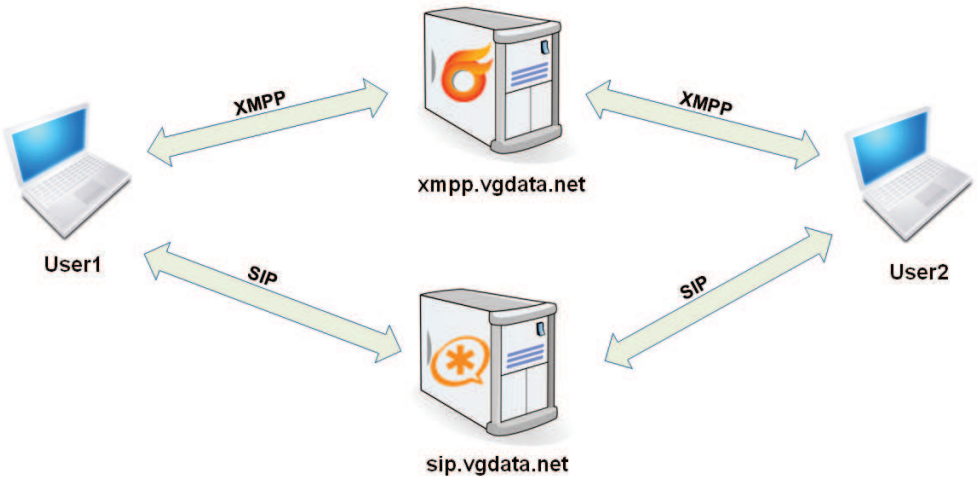


Figura 26: Configuração utilizando dois servidores

Independente da configuração utilizada, o aspecto mais importante, todavia, é garantir que o usuário consiga, efetivamente, conectar aos serviços. Nesse sentido, caso o(s) servidor(es) esteja(m) fora da rede local, deve-se certificar que as respectivas portas de comunicação estejam liberadas ou, caso contrário, o usuário não poderá acessar os serviços.

Além da existência dos servidores XMPP e SIP, é necessária também a criação dos usuários. O IdDE considera que os usuários já existem e não possibilita a sua criação. Esta tarefa deve ser executada utilizando-se as ferramentas apropriadas.

Para implementar a comunicação no IdDE, foram utilizadas as bibliotecas: Smack[17] para a comunicação XMPP, Peers[18] para comunicação SIP, além de partes de código do projeto Shortalk[19]. A estruturação do código, conforme discutido anteriormente, foi feita de acordo com os módulos do aplicativo. A Figura 27 apresenta a organização dos pacotes e classes principais relacionadas com o envio e recebimento de mensagens.

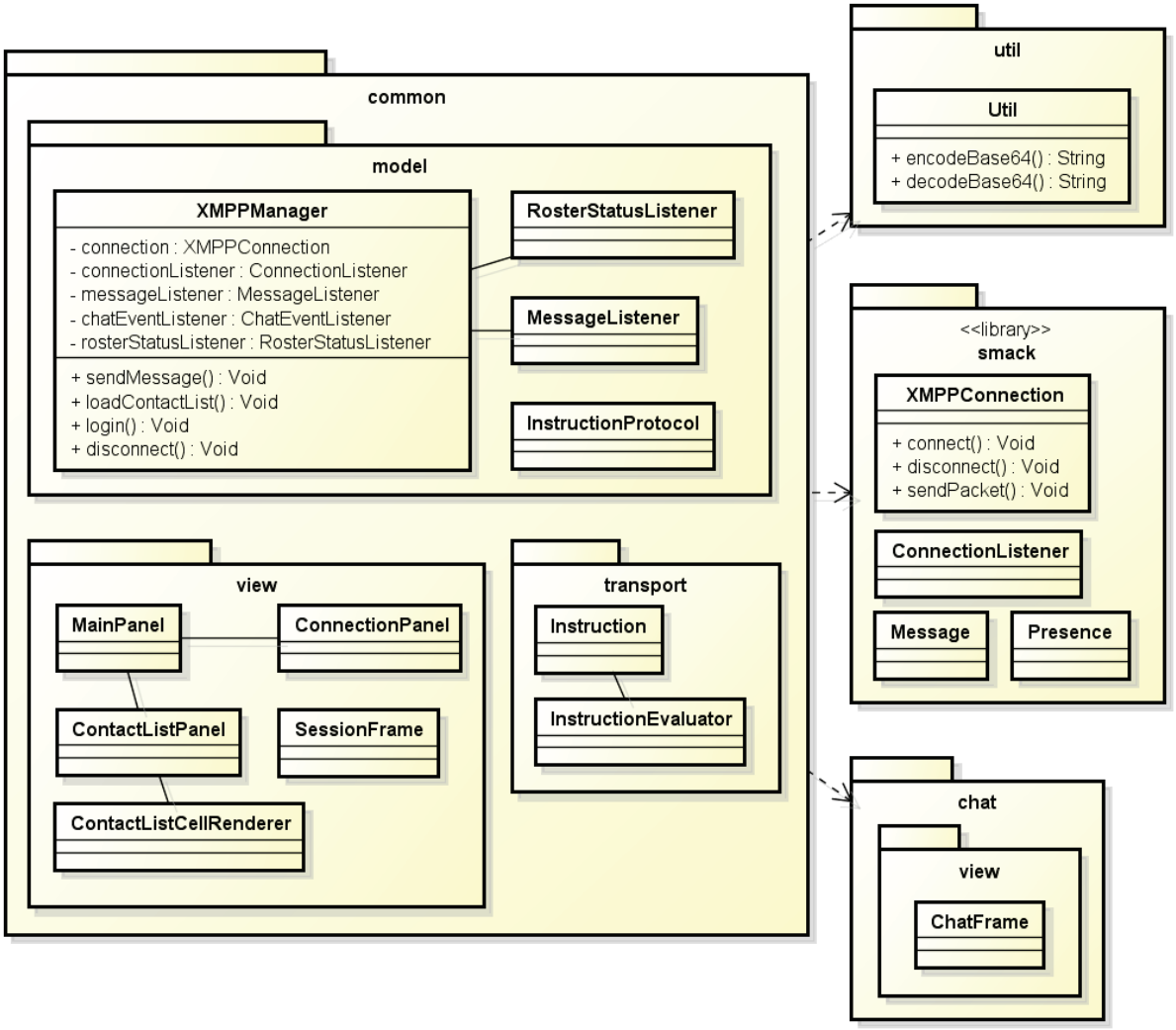


Figura 27: Pacotes e classes para comunicação XMPP

O nome "*common*" foi atribuído ao pacote principal em função das suas classes, que são comuns e utilizadas por todas as ferramentas para o envio das mensagens entre os

[17] http://www.igniterealtime.org/projects/smack/
[18] http:// peers.sourceforge.net
[19] http://code.google.com/p/shortalk/

ambientes. A classe *XMPPManager*, do subpacote *model*, é uma das principais classes. Como pode-se perceber pelos seus atributos, ela mantém e monitora as informações da conexão (atributos *connection* e *connectionListener*), monitora o status de presença (*rosterStatusListener*) e mensagens (*messageListener* e *chatEventListener*) dos outros usuários. A função e utilização dos demais pacotes e classes estão exemplificadas no processo de criação, envio e recebimento de mensagens, descritos a seguir.

É importante salientar que, para o funcionamento do IdDE, o único serviço imprescindível é o XMPP. Caso o serviço SIP não esteja disponível, não será possível utilizar a comunicação via áudio. Todavia, caso não se disponha de um servidor XMPP, será inviável a utilização do ambiente, pois este serviço é utilizado na comunicação entre os ambientes, transportando as mensagens do protocolo IdDE.

A mensagem XMPP enviada de um ambiente para outro, encapsula, na sua *Stanza* <*message*>, uma mensagem do protocolo IdDE que contém as respectivas instruções. Este protocolo foi estruturado utilizando XML, o que torna a mensagem clara e de fácil compreensão, além de contribuir na interoperabilidade com outros aplicativos.

Exemplificando, quando um usuário efetuar uma alteração no código fonte de um programa em seu ambiente, será enviada uma mensagem XMPP aos usuários remotos contendo um XML com a mensagem do protocolo IdDE similar à Figura 28.

```
<IdDE>
  <code>2</code>
  <arg0>File.java</arg0>
  <arg1>2i</arg1>
  <arg2>125</arg2>
  <arg3>5</arg3>
  <arg4>while</arg4>
</IdDE>
```

Figura 28: Mensagem do protocolo IdDE

Nesta mensagem, pode-se observar a estrutura de *tags* utilizadas para definição de todas as mensagens do protocolo. À exceção da *tag* <code>, que sempre é utilizada para identificar o código da instrução, as informações das demais *tags* variam, dependendo do código. Além disso, cada instrução utiliza apenas as tags necessárias e, assim, algumas instruções podem, por exemplo, utilizar apenas as *tags* <code> e <arg0>. A Tabela 5 descreve algumas mensagens comuns do protocolo IdDE, utilizadas por todas as ferramentas. As

demais mensagens do protocolo são apresentadas adiante, juntamente com as ferramentas nas quais são utilizadas.

Tabela 5: Mensagens comuns do protocolo IdDE

Objetivo	*Tag*	Conteúdo da *Tag*
Este código é enviado quando foi recebida alguma requisição contendo um código inválido ou não conhecido.	*code*	-2
	arg0	Código da requisição recebida
	arg1	Não utilizado
	arg2	Não utilizado
	arg3	Não utilizado
	arg4	Não utilizado
Esta mensagem é enviada em resposta a uma instrução, avisando que ocorreu um erro durante a execução da instrução recebida anteriormente.	*code*	-1
	arg0	Código da requisição recebida
	arg1	Não utilizado
	arg2	Não utilizado
	arg3	Não utilizado
	arg4	Não utilizado

Apesar de fundamental, pode-se perceber que o papel do servidor XMPP fica restrito a fornecer a informação de presença dos demais usuários e ao transporte das mensagens entre os usuários. Não existe nenhum tipo de processamento adicional feito pelo servidor.

Quando um ambiente pretende enviar uma mensagem, este deverá seguir a sistemática exibida no fluxograma da Figura 29. De acordo com o fluxograma, quando se deseja enviar uma mensagem do protocolo, o primeiro passo é criar um objeto (classe *Instruction*, pacote *common::transport*) com a respectiva mensagem. Em seguida, este objeto é convertido numa *string* XML. Para esta tarefa, o IdDE utiliza a biblioteca Xstream[20]. A etapa seguinte é verificar se a mensagem deve ser codificada. Em caso positivo, ocorre a codificação utilizando o método base64 (método *encodeBase64* da classe *Util::Util*). Conforme será descrito adiante, essa codificação é necessária em alguns casos, como por exemplo, quando se está utilizando o servidor do Google. O passo seguinte é criar a mensagem do protocolo

[20] http://xstream.codehaus.org/

XMPP (classe *Message* da biblioteca Smack), contendo a string de instrução do protocolo IdDE anteriormente convertida. Em seguida, a mensagem é enviada utilizando-se a conexão ativa (atributo connection, classe *XMPPManager*, pacote *common::model*) com o servidor XMPP. Caso ocorra algum erro, é apresentada uma mensagem ao usuário, solicitando se este deseja que seja feita uma nova tentativa de envio. Em caso positivo, é repetido o processo de envio, caso contrário, ocorre o cancelamento do envio.

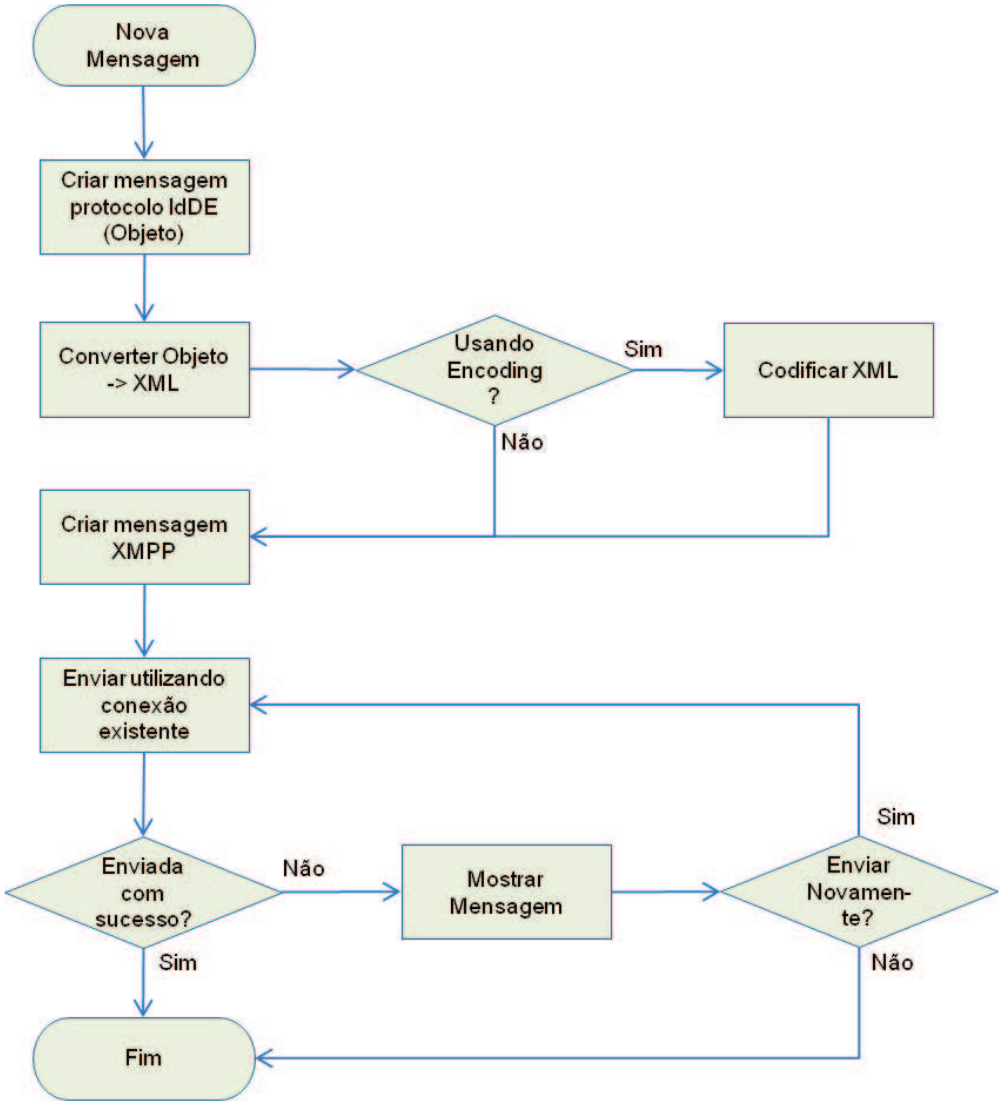


Figura 29: Fluxograma de criação e envio de mensagem no IdDE

Já o recebimento de uma mensagem pelo ambiente, obedece à sequência de atividades apresentada na Figura 30. A classe *XMPPManager* mantém e monitora as informações relacionadas à comunicação XMPP. Quando uma nova mensagem é recebida, é acionado o

método *processMessage* da classe *MessageListener* (pacote *common::model*). Este método verifica se a mensagem é válida e, em caso positivo, propaga a mensagem que é então capturada pelo método *modelPropertyChange* da classe *ContactListPanel* (pacote *common::view*). Esta classe, como o nome dá a entender, é responsável pela exibição da lista e status dos contatos do usuário. Além dos contatos, ela efetua também a análise de todas as mensagens recebidas. Estas mensagens podem ser de presença dos usuários, mensagens de *chat* ou ainda, mensagens do protocolo IdDE. A determinação do tipo de mensagem recebida é feita pelo método *modelPropertyChange*. Caso seja uma mensagem de *chat*, será acionado o método *receivedMessage* da classe *ChatFrame* (pacote *chat.view*), responsável pela exibição da janela de *chat*. Por outro lado, se a mensagem for uma instrução do protocolo IdDE, será invocado o método *processMessage* da classe *InstructionEvaluator* (pacote *common::transport*). Este método, então, analisará o código da instrução recebida e executará os procedimentos necessários para atender a solicitação recebida.

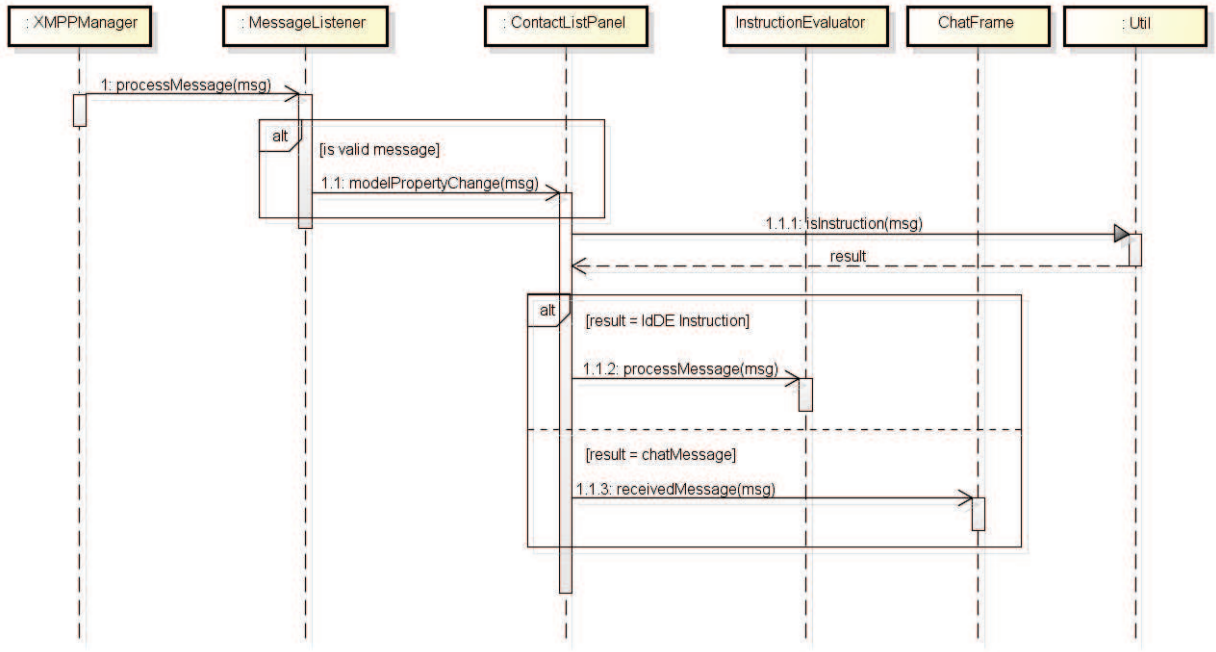


Figura 30: Sequência de atividades do recebimento de uma mensagem

4.5 O Ambiente IdDE

A interface do IdDE, utilizada para interação pelo usuário final, foi uma preocupação constante durante o desenvolvimento do projeto. Pelo fato de o ambiente ser integrado ao Netbeans, o desenvolvimento de suas interfaces teve que considerar tanto os aspectos de IHC

quanto seguir as definições e regras do Netbeans. Objetivava-se que o ambiente desenvolvido fosse de fácil acesso pelo usuário final e, ao mesmo tempo, não "atrapalhasse" sua experiência de uso ou fugisse do padrão visual estabelecido pelo Netbeans.

Para atingir esses objetivos, as interfaces desenvolvidas no IdDE foram encapsuladas no sistema de janelas do Netbeans, através da utilização de suas APIs. Na Figura 24 essas interfaces são identificadas pelas classes *MainPanel* e *SessionFrame* do pacote *common::view*. Essas janelas são criadas automaticamente quando o *plugin* do IdDE é carregado, o que ocorre durante a inicialização do Netbeans.

Como resultado final, obteve-se uma interface composta de duas janelas que ficam acopladas ao Netbeans, como pode ser observado na Figura 31.

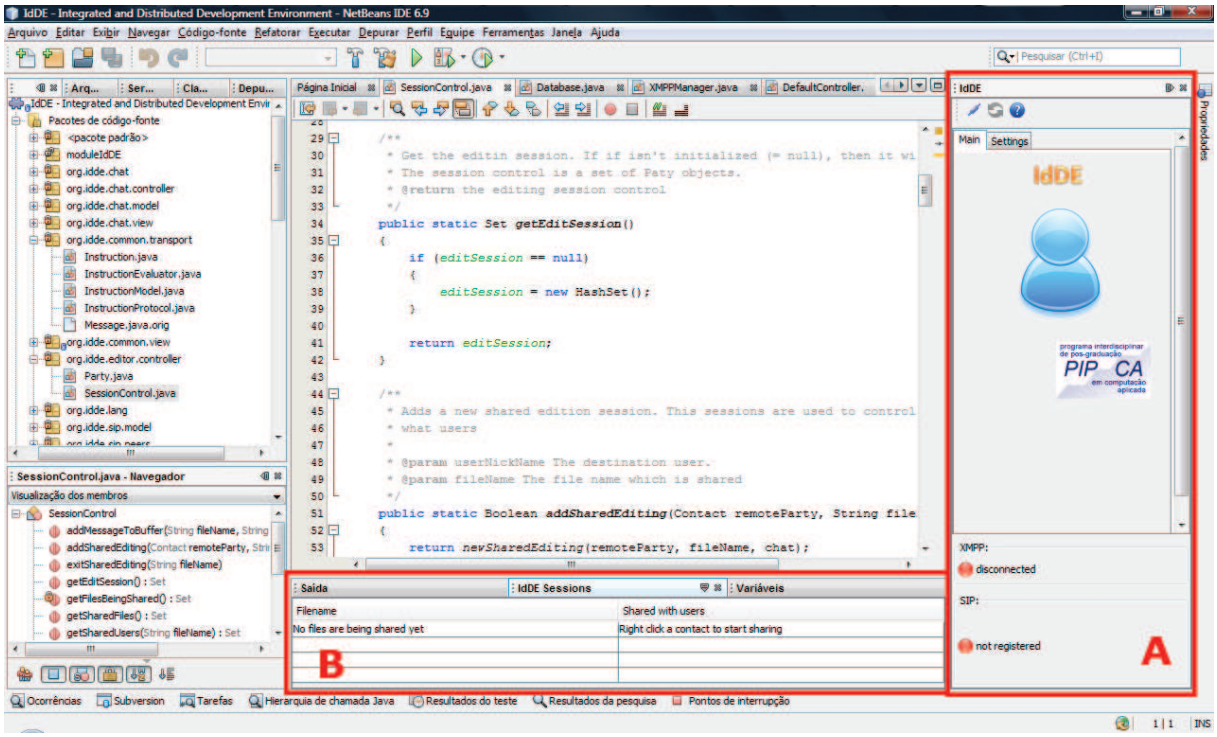


Figura 31: Integração do IdDE no Netbeans

Na Figura 31 estão destacadas ambas as janelas do IdDE: a janela principal (A) e a janela de sessões de edição (B). Como está integrado ao Netbeans, o IdDE dispõe dos mecanismos deste ambiente, como por exemplo, a possibilidade de minimizar as janelas, o que amplia o espaço de edição, o que pode ser observado na Figura 32. Mesmo estando minimizadas, as funcionalidades do aplicativo permanecem inalteradas.

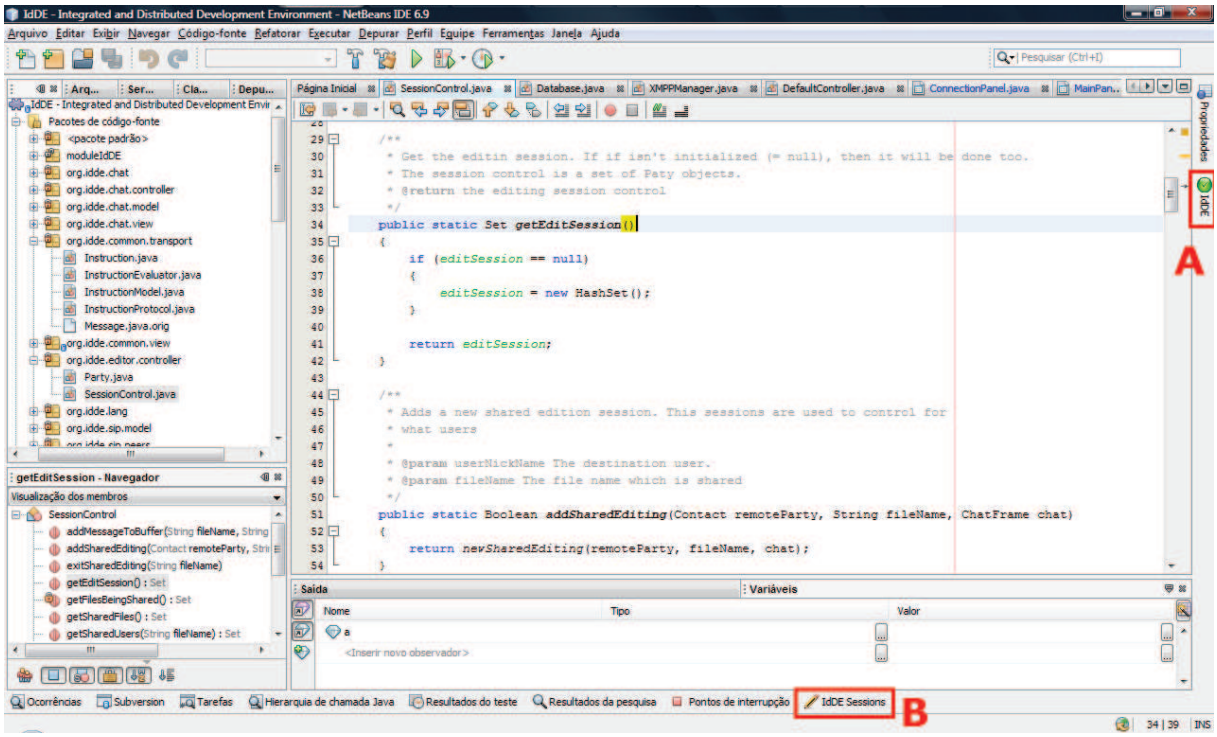


Figura 32: Minimização das janelas do IdDE

Uma das funções da janela principal é possibilitar a configuração do IdDE. Essas opções estão localizadas na aba "*Settings*". Clicando nesta aba, o ambiente apresenta a interface exibida na Figura 33, na qual devem ser configurados:

- A: os dados para conexão com o servidor XMPP. Esta é a conexão que o ambiente utilizará para se comunicar com outros ambientes;
- B: os dados para conexão com o servidor SIP. Esta é a conta que o ambiente utilizará para a comunicação via áudio, tanto para chamadas SIP quanto PSTN;
- C: ícone que indica o status da conexão com o servidor XMPP;
- D: ícone que indica o status da conexão com o servidor SIP;
- E: barra de ferramentas, que possui os botões de conectar, atualizar lista de contatos e o botão que exibe informações sobre a versão do IdDE.

Acessando a aba "*Behavior*", é apresentada a tela exibida na Figura 33-F, com outras opções de configuração. Nessa tela são informados:

- *Phone Numbers*: indicam os números de telefone móvel e convencional do usuário para receber contatos de áudio via PSTN;

- *Use encoded messages* (Base64): essa opção indica se o ambiente deve codificar as mensagens antes de enviá-las. Esta codificação se mostrou necessária, por exemplo, quando se utiliza o servidor do Google para a troca de mensagens. Durante o desenvolvimento percebeu-se que após a décima ou décima primeira mensagem enviada, ocorriam erros de transmissão. Passados após alguns segundos, porém, a comunicação voltava ao normal. Quando passou-se a utilizar mensagens codificadas com esse servidor, o problema deixou de ocorrer;
- *Encoding charset*: é utilizado para indicar o conjunto de caracteres que deve ser utilizados na codificação das mensagens;
- *Messages Buffer*: este campo é utilizado para indicar o tempo (em segundos) que o ambiente deverá manter o buffer de alterações de código antes de enviá-las aos usuários remotos. A opção em utilizar uma informação medida em tempo, e não em quantidade de caracteres, deve-se ao fato de considerar que, se um usuário ficar muito tempo sem efetuar uma alteração, essa informação poderá ficar desatualizada. Testes indicaram, ainda, que a utilização de buffer não é interessante, pois como nenhum servidor é utilizado para intermediar a troca de mensagens, caso algum usuário utilize um buffer de tempo e outro não utilizar, a ordem das alterações poderá ficar incorreta, causando discrepâncias entre os códigos dos ambientes. Nesse sentido, essa funcionalidade será reavaliada em futuras versões do aplicativo;
- *Allow only ASCII chars*: este campo deve ser utilizado para indicar se o sistema deve permitir somente o uso caracteres ASCII básicos (sem acentuação, cedilha), ou se eles devem ser permitidos na edição colaborativa. Esta opção deve ser habilitada em situações nas quais os equipamentos possuem sistemas operacionais diferentes. Nos testes efetuados, ao utilizar um notebook com o sistema operacional Linux e outro com Windows Vista, percebeu-se que as mensagens acentuadas enviadas por um ambiente não eram recebidos corretamente pelo outro. Não foi possível identificar se o problema está relacionado ao conjunto de caracteres do sistema operacional, suporte do Java ou outro problema. Em casos como esse, é aconselhável efetuar testes prévios para validar a possibilidade de usar ou não caracteres acentuados;

- *Log changes to DB*: ativando esta opção o IdDE passa a armazenar todas as modificações feitas num documento compartilhado em uma tabela na base de dados. A base de dados utilizada, SQLite[21], está embarcada junto à ferramenta e, dessa forma, não necessita de nenhuma configuração ou instalação adicional no Sistema Operacional.

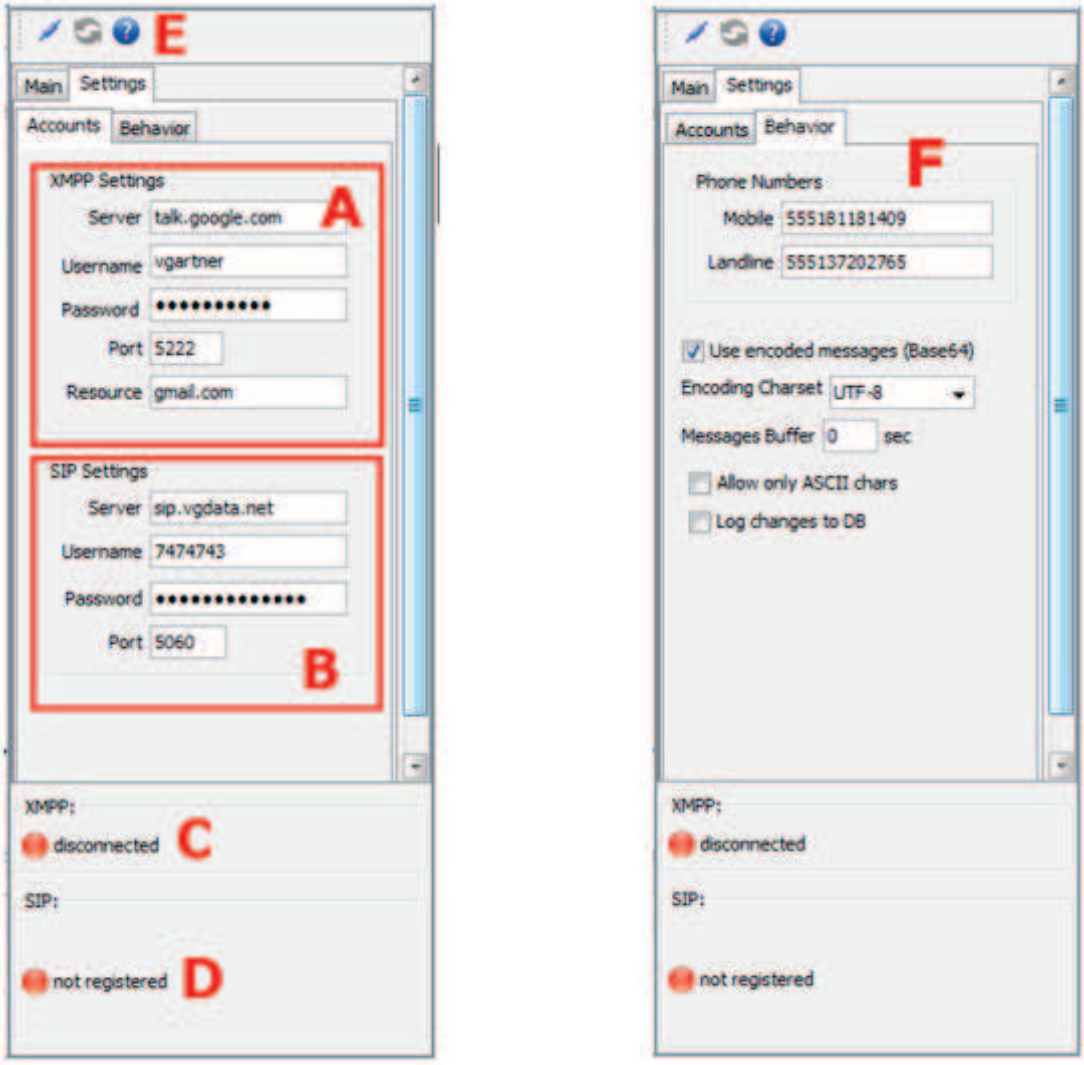


Figura 33: Configuração do IdDE

Ao efetuar *login* (pressionando o botão da barra de ferramentas), o sistema tentará acessar os servidores configurados e autenticar as credenciais informadas na configuração. Caso a autenticação seja bem sucedida, o conteúdo da janela principal será modificado,

[21] http://www.sqlite.org/

conforme mostra a Figura 34. Além de apresentar a lista de contatos do usuário (A), os ícones indicadores do status dos serviços passarão a ser verdes (B), e a barra de ferramentas (C) terá o botão de conectar substituído pelo botão de desconectar, e o botão para atualizar os contatos será habilitado.

Juntamente com a lista de contatos, é apresentada a informação atualizada sobre a sua presença. Essa informação é importante, pois usuários em sítios diferentes compartilham relativamente pouco contexto e tendem a ter pouco conhecimento sobre o que outros usuários estão fazendo, se estão disponíveis para comunicação e quais são suas preocupações iniciais. Esta falta de informação contextual torna difícil iniciar um contato e, muitas vezes, leva a mal-entendidos [Herbsleb, 2007].

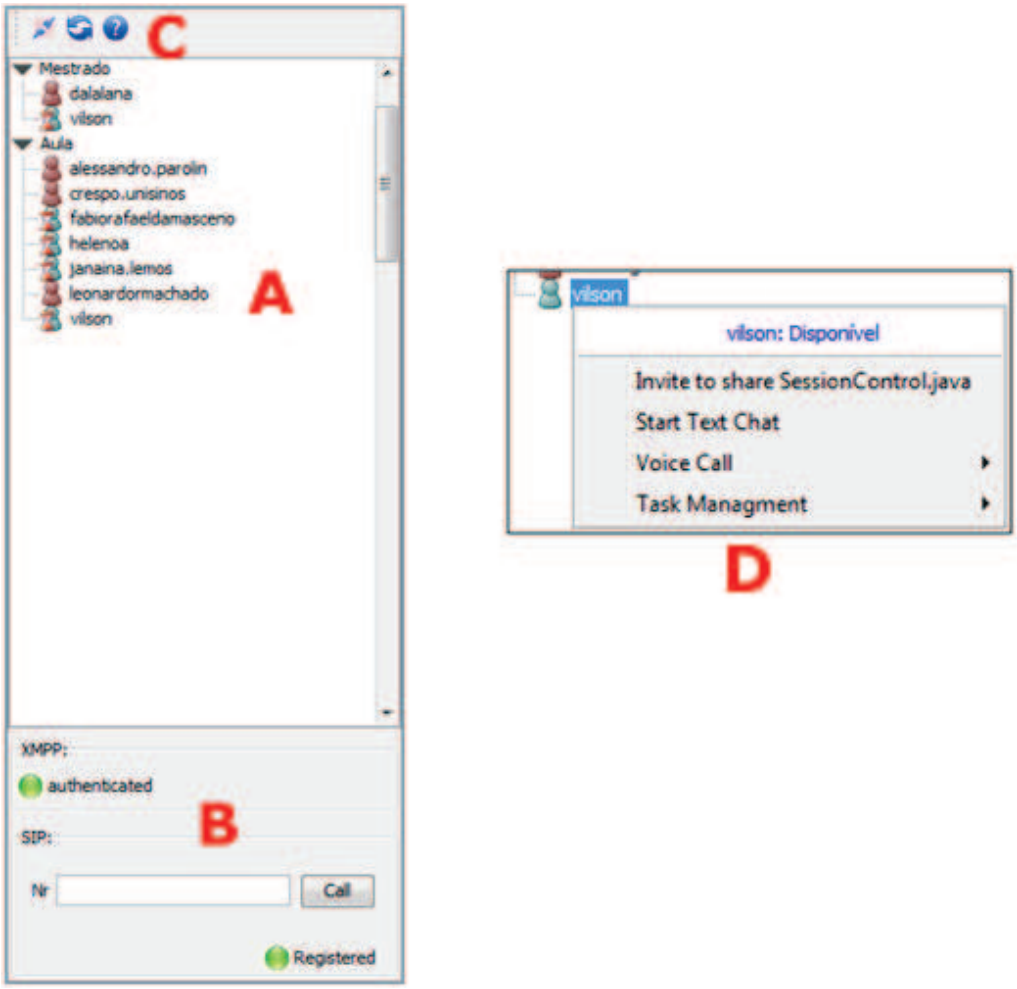


Figura 34: Lista de contatos e menu de contexto

Após a comunicação ter sido estabelecida com sucesso entre o ambiente e o servidor, é possível iniciar o uso das ferramentas disponibilizadas pelo IdDE. Para isso, basta clicar com

o botão direito do *mouse*, sobre um contato e o sistema abrirá um menu de contexto com o acesso às ferramentas, exemplificado na Figura 34-D.

4.6 Edição colaborativa

A ferramenta de edição colaborativa do ambiente IdDE permite que vários usuários acessem e modifiquem o mesmo código fonte, simultaneamente. Assim, quando um usuário fizer uma alteração no código fonte em seu ambiente, as mesmas alterações serão replicadas nos ambientes remotos. Dessa forma, além de permitir a edição simultânea, a ferramenta também poderá auxiliar no aprendizado de desenvolvedores iniciantes, possibilitando que estes acompanhem a programação de usuários mais experientes.

Um aspecto importante da edição colaborativa é o controle e ordenação das mensagens enviadas pelos ambientes. Levando em consideração aspectos como velocidade e *delay* da rede, em situações onde houver vários usuários alterando simultaneamente um arquivo, existe a possibilidade da ordem de recebimento das mensagens ficar incorreta. Caso isso ocorra, poderão ocorrer inconsistências, fazendo com que os usuários tenham diferentes versões do código em seus ambientes.

Para impedir que este tipo de problema ocorra, foi implementado o algoritmo dOPT (*Distributed Operational Transformation*), publicado originalmente por Ellis e Gibbs (1989). Com este algoritmo, todas as operações originadas por um usuário são ordenadas e enviadas para os demais usuários, onde devem ser executadas na mesma ordem. Quando uma mensagem é recebida, informações auxiliares são examinadas para verificar se a mensagem pode ser executada imediatamente. Caso não possa, a mesma é colocada na fila de espera, até que as mensagens faltantes sejam recebidas e executadas.

4.6.1 Estabelecimento da sessão de edição colaborativa

Como visto anteriormente, após estabelecer a conexão com o servidor XMPP, será possível iniciar a utilização do ambiente. Para iniciar uma sessão de edição colaborativa, deve-se clicar com o botão direito do mouse sobre o usuário desejado, na lista de contatos, e escolher a opção "*Invite to share*", mostrada a Figura 34-D. Para que seja possível compartilhar um arquivo, é necessário que ele já esteja aberto no Netbeans.

O processo completo do convite para edição colaborativa é apresentado pelo fluxograma da Figura 35. Após selecionar o usuário, o ambiente verifica se o arquivo já está

numa sessão com este mesmo usuário e, se estiver, exibe uma mensagem avisando o usuário e finaliza o processo. Caso contrário, será criada a mensagem de convite do protocolo IdDE, descrita na Tabela 6, e enviada ao usuário remoto. A criação desta mensagem segue os procedimentos apresentados no fluxograma da Figura 29. Após enviar o convite, o IdDE mostra ao usuário a mensagem da Figura 36 (A), solicitando que este não modifique o arquivo antes de receber uma resposta do usuário remoto. Isso se deve ao fato de o convite enviado conter também o conteúdo do arquivo que será compartilhado e, caso ocorra alguma modificação, as informações dos dois ambientes serão diferentes. Este procedimento foi adotado tendo em vista que, se o usuário aceitar o convite, não será necessária uma nova troca de mensagens para o envio desse conteúdo, contribuindo também para a diminuição do fluxo de mensagens. Em trabalhos futuros, pretende-se impedir que o usuário possa fazer as alterações no documento, travando sua edição até o recebimento da resposta.

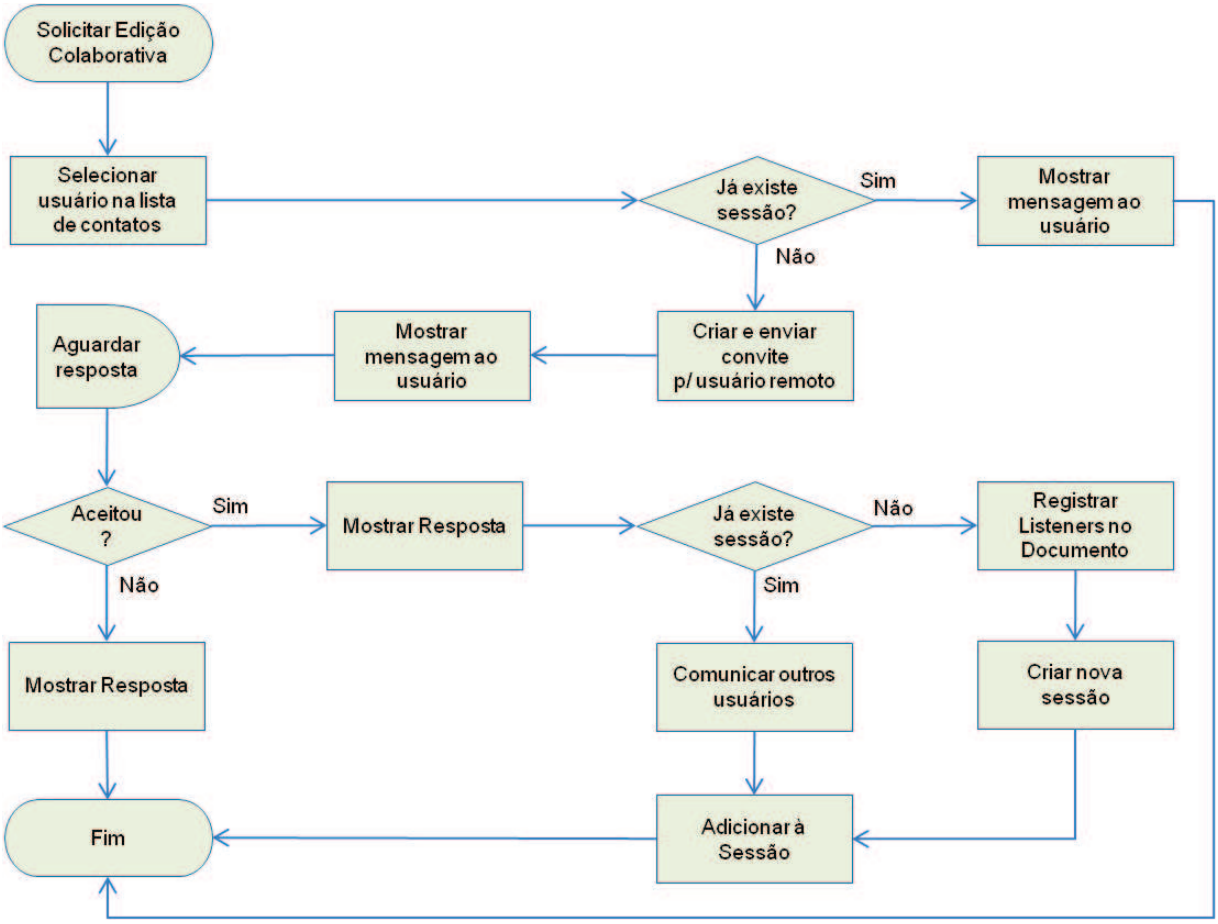


Figura 35: Fluxograma de convite para nova edição colaborativa

Quando o usuário remoto não aceitar o convite, é mostrada a mensagem da Figura 36 (C) informando o fato e o fluxo é encerrado. Por outro lado, se a resposta for positiva, a

mensagem exibida será a da Figura 36 (B), avisando que a sessão foi estabelecida. Neste caso, o sistema verificará se o arquivo já está sendo compartilhado com outros usuários. Caso já exista uma sessão, será enviada uma mensagem do protocolo IdDE a todos os usuários, solicitando que estes adicionem o novo usuário as suas sessões. Entretanto, se ainda não existir uma sessão para este arquivo, o IdDE adicionará um *listener*[22] ao documento, responsável por monitorar as alterações no documento (descrito adiante), criará uma nova sessão para o arquivo compartilhado e, finalmente, registrará o arquivo na nova sessão, concluindo o processo.

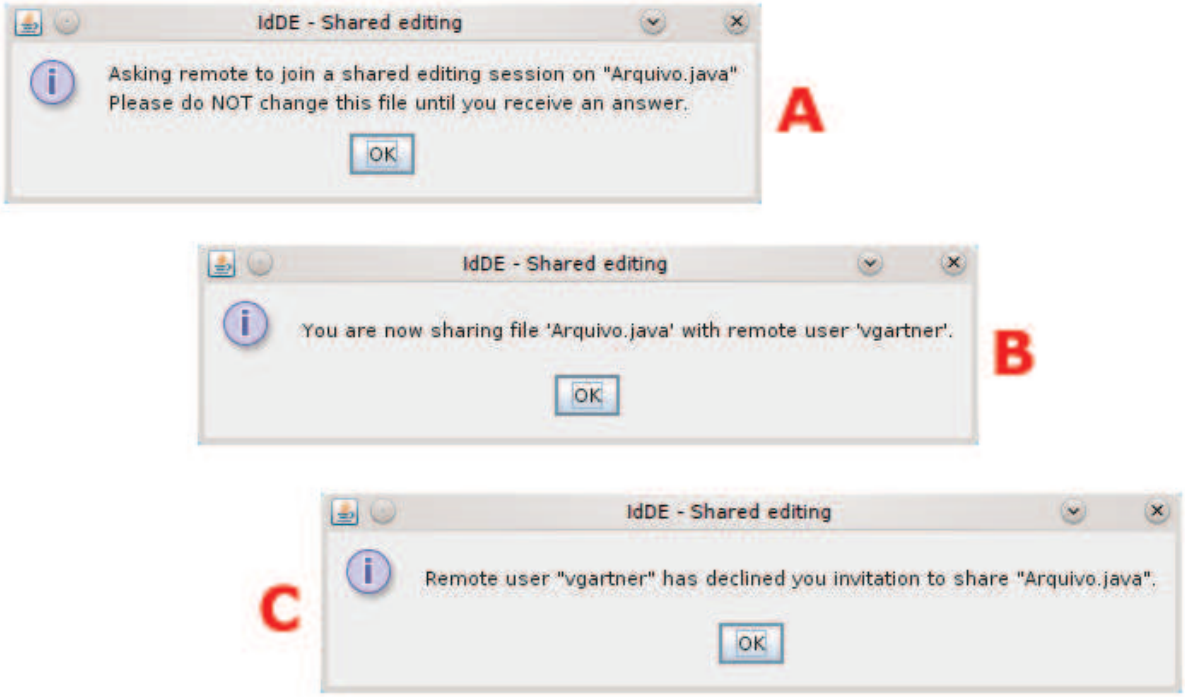


Figura 36: Mensagens do convite de nova sessão

O fluxo de eventos que ocorrem no usuário remoto, que recebe o convite, está representado na Figura 37. Quando o ambiente recebe a mensagem e identifica como sendo um convite para colaborar na edição de um arquivo, o primeiro passo é verificar se este arquivo já não está numa sessão.

Se este arquivo já estiver numa sessão, será enviada uma mensagem para o usuário remoto informando essa situação. Se nenhuma sessão existir, será mostrada a mensagem da

---

[22] Um *listener* é executado quando o usuário interage com a interface de usuário, de um programa Java, que gera um evento [Deitel e Deitel, 2006].

Figura 38 (A). Caso o usuário não aceite o convite, o IdDE envia a resposta negativa para o usuário que originou o convite, mostra a mensagem da Figura 38 (C) e avisa o usuário local que nenhuma sessão foi estabelecida, encerrando o processo em seguida.

As mensagens de resposta ao convite recebido estão descritas na Tabela 6.

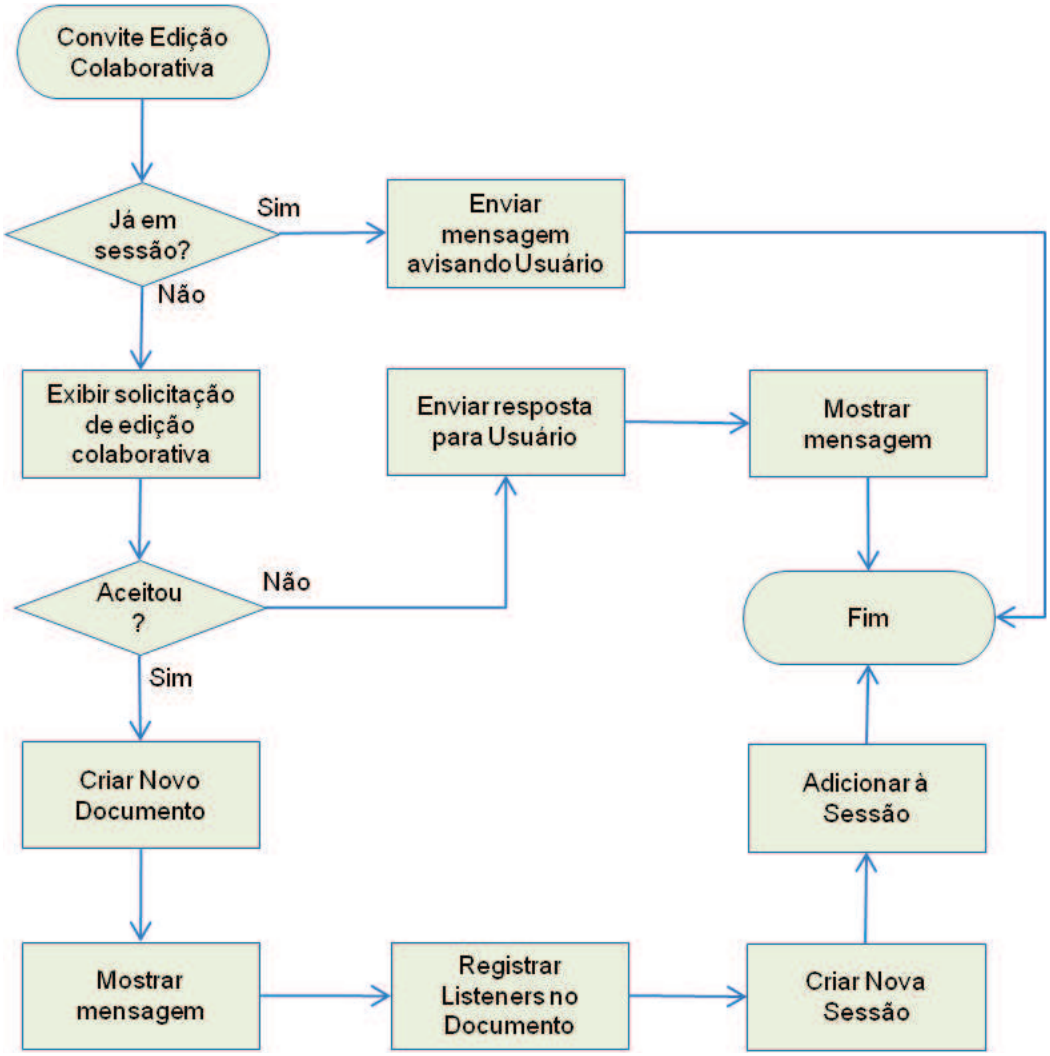


Figura 37: Fluxograma recebimento convite edição

Contudo, caso o usuário aceite o convite, o IdDE inicia um novo documento no Netbeans e armazena o conteúdo do arquivo recebido em um arquivo na pasta "*home*" do usuário, com o nome recebido na mensagem do protocolo. Se o arquivo já existir, ele será sobrescrito. Posteriormente, este arquivo poderá ser salvo em qualquer outra pasta, bastando utilizar as funções de gravação de arquivos do próprio Netbeans.

Após efetuar a criação do documento, é apresentada ao usuário uma mensagem como a da Figura 38 (B), com a respectiva informação. Em seguida, é adicionado um *listener*

(encarregado de monitorar as alterações) ao novo documento. Por fim, é criada uma nova sessão para o controle da edição compartilhada, o arquivo é adicionado à sessão e o fluxo é encerrado.

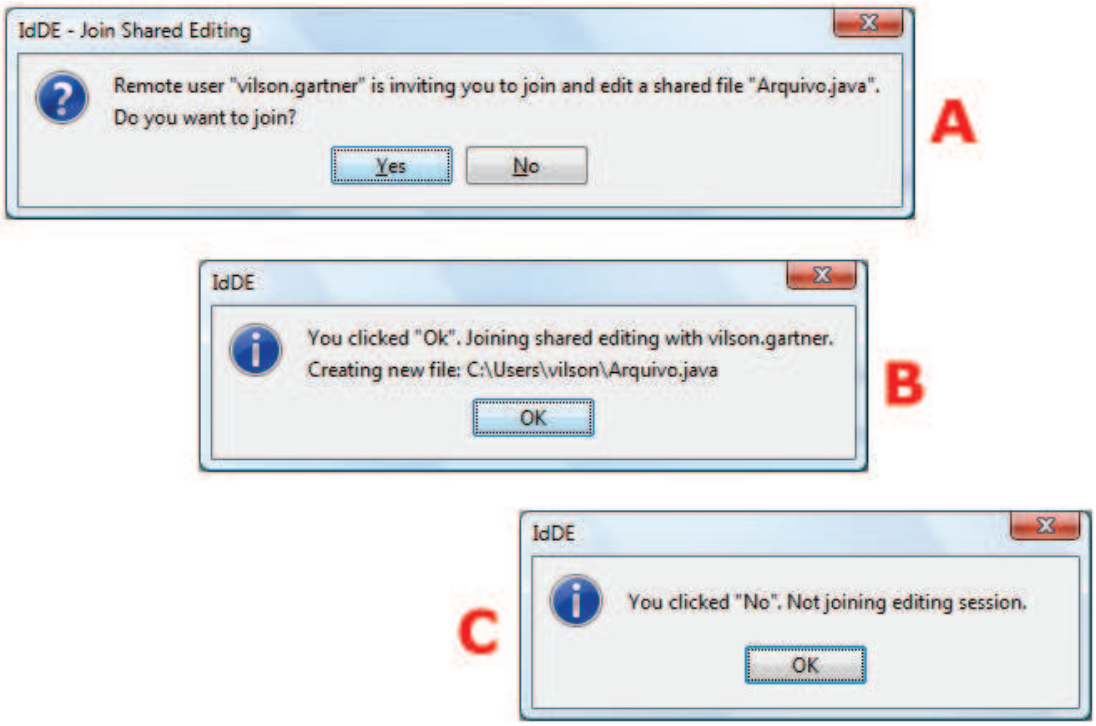


Figura 38: Mensagens do recebimento de convite para edição

A ferramenta de edição colaborativa é a que mais utiliza a troca de mensagens entre os ambientes. A Tabela 6 descreve as mensagens do protocolo IdDE envolvidas com o processo de negociação para estabelecimento e manutenção da sessão de edição colaborativa.

Tabela 6: Instruções do Protocolo IdDE de negociação da sessão

Objetivo	*Tag*	Conteúdo da *Tag*
Este código é enviado pelo ambiente quando um usuário deseja estabelecer uma sessão de edição compartilhada com o usuário remoto. No convite, o ambiente também já envia o conteúdo do arquivo (arg1), para ser adicionado ao novo documento que será criado pelo ambiente remoto.	*code*	1
	arg0	Nome do arquivo que será compartilhado
	arg1	Conteúdo do arquivo

Objetivo	*Tag*	Conteúdo da *Tag*
As mensagens com este código indicam que o arquivo já está numa sessão no ambiente remoto que recebeu o convite.	*code*	1e
	arg0	Nome do arquivo
Mensagem enviada em resposta ao convite, avisando que o usuário aceitou participar da sessão. Ao receber esta mensagem, o ambiente remoto passará a enviar as mensagens de alteração no documento.	*code*	1a
	arg0	Nome do arquivo que está sendo compartilhado
Mensagem enviada para informar que o usuário não aceitou o convite para participar da edição colaborativa de um documento.	*code*	1r
	arg0	Nome do arquivo que seria compartilhado
Este código é utilizado nas mensagens que informam usuários remotos que um novo usuário está participando da edição compartilhada. Ao receber esta mensagem, o novo usuário será adicionado às sessões e passará a receber as mensagens de alterações no documento.	*code*	1j
	arg0	Nome do arquivo da sessão
	arg1	Nome do usuário que está entrando na sessão
Código utilizado nas mensagens que indicam que um usuário está deixando a sessão.	*code*	16
	arg0	Nome do arquivo compartilhado

4.6.2 Monitoramento de alterações no documento

Após estabelecer a comunicação entre os ambientes e adicionar os *listeners* necessários aos documentos, todas as modificações feitas por um usuário serão enviadas aos demais usuários participantes da sessão e serão replicadas naqueles ambientes.

O processo de monitoramento das alterações de estado do documento e o seu envio para os demais usuários são apresentados no fluxograma da Figura 39. Quando o ambiente detecta uma modificação, o primeiro passo é verificar se o documento está numa sessão de edição colaborativa. Caso ele não esteja numa sessão, o processo é concluído e nenhuma ação

é tomada. Entretanto, se ele estiver sendo compartilhado, é necessário avaliar o tipo de alteração ocorrida e compor a instrução apropriada para os demais usuários participantes.

As modificações no estado podem ser alteração de texto, com a inclusão ou exclusão de caracteres, ou a mudança na posição do cursor. Quando ocorrer uma mudança na posição do cursor, o ambiente verifica o novo posicionamento e envia uma mensagem aos ambientes remotos, para que estes atualizem essa informação.

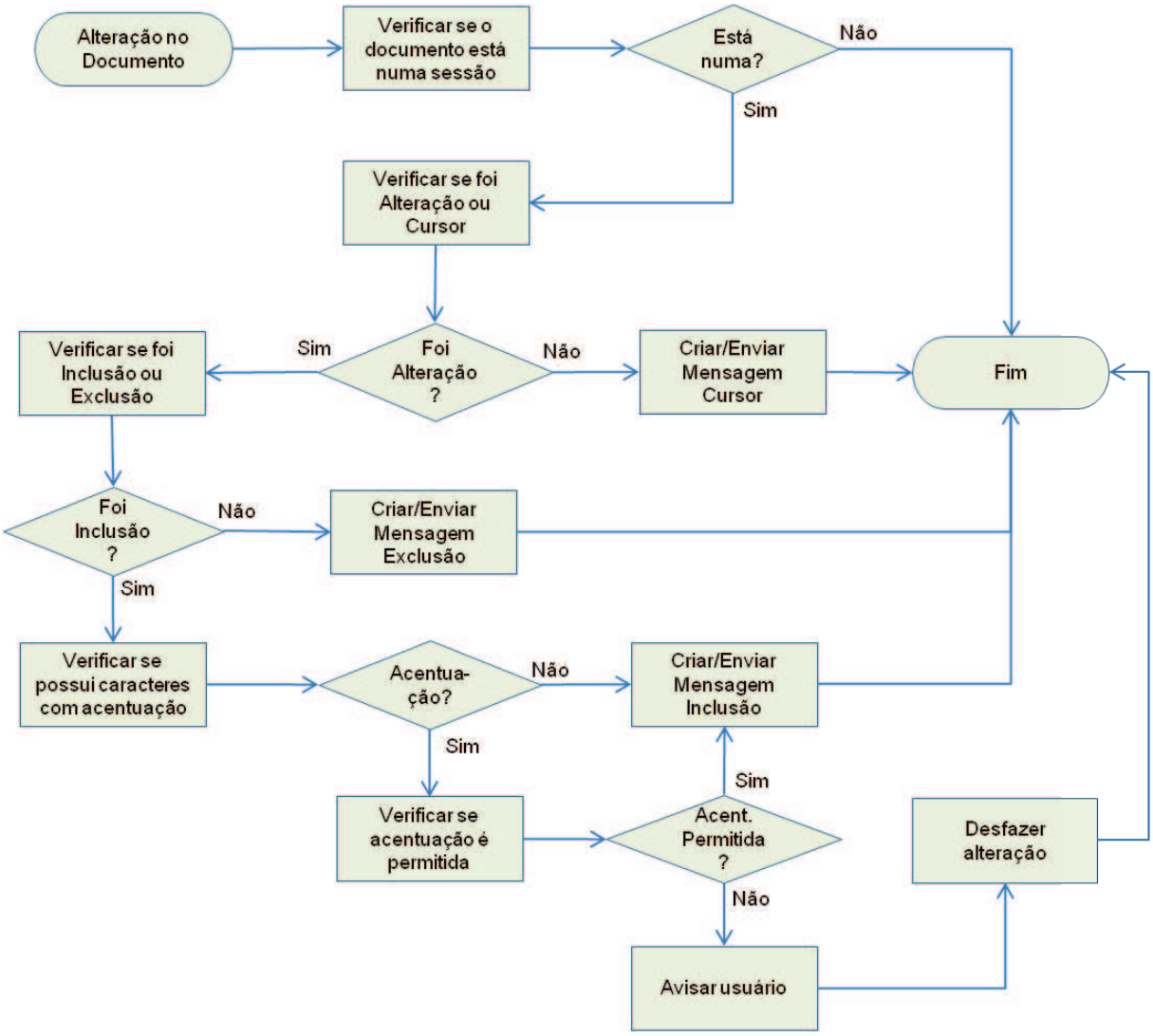


Figura 39: Fluxograma de monitoramento do documento e envio de alterações

Quando ocorrer a alteração de texto, o IdDE verifica se houve inclusão ou exclusão de texto. Em caso de exclusão de um trecho de código, o único procedimento a ser executado é a criação e o envio da respectiva mensagem para os ambientes remotos. Por outro lado, quando um novo caractere é adicionado ao documento, serão necessárias verificações adicionais. O

primeiro passo é verificar se o(s) caractere(s) adicionado(s) possui(em) acentuação ou cedilha. Se algum desses símbolos estiver presente, será necessário verificar a opção assinalada na tela de configuração (antes de efetuar o *login*). Se na configuração estiver indicado que o uso desses caracteres não é permitido, é apresentada uma mensagem ao usuário alertando o fato e a alteração no documento será desfeita. Porém, se estes caracteres não estiverem presentes, ou eles forem permitidos, será criada (e enviada) a instrução de inclusão para os demais ambientes e o processo será finalizado.

É importante salientar que a inclusão de texto também pode ser feita utilizando o processo de copiar e colar. Neste caso, a mensagem de inclusão que será enviada para os demais usuários conterá todos os caracteres adicionados, não sendo envida uma mensagem por caractere.

O recebimento de uma mensagem de alteração segue a sequência indicada na Figura 30. Neste tipo de mensagem, o método *processMessage* executa a rotina de atualização do documento. Uma situação particularmente importante que este método deve observar é a existência dos *listeners* no documento, que ficam monitorando as alterações. Antes de efetuar as modificações (inclusão ou exclusão de caracteres) ou mudar a posição do cursor, estes *listeners* devem ser desabilitados, caso contrário os mesmos desencadearão o processo de envio das alterações para os demais usuários, gerando um *loop*.

As mensagens do protocolo IdDE utilizadas pelos ambiente para comunicar as alterações e mudanças no cursor estão descritas na Tabela 7.

Tabela 7: Mensagens do protocolo IdDE para alterações no código e cursor

Objetivo	***Tag***	**Conteúdo da *Tag***
Este código é utilizado em mensagens que comunicam uma alteração no conteúdo de um arquivo. O conteúdo da *tag* <arg1> indica que a operação de alteração a ser feita é de inclusão de conteúdo. A posição de inserção (arg2) é uma informação numérica e deve indicar a posição onde o conteúdo deve ser inserido. A posição inicia em 0 (zero) – primeira posição no texto – e o número máximo que indique a última posição no texto.	*code*	2
	arg0	Nome do arquivo no qual deve ser aplicada a alteração
	arg1	2i
	arg2	Posição onde a *string* deve ser inserida.

Objetivo	*Tag*	Conteúdo da *Tag*
A informação de <arg3> poderia ser opcional, pois ela poderia ser obtida através da verificação de <arg4>. Todavia, essa informação é enviada para que o receptor possa validar se o conteúdo de <arg4> possui a mesma quantidade de caracteres informadas em <arg3>.	*arg3*	Tamanho da *string* a ser inserida
	arg4	*String* que deve ser inserida no código
Esta mensagem também indica que houve uma alteração no documento, entretanto, a informação da *tag* <arg1> indica se tratar de uma exclusão de caracteres.	*code*	2
	arg0	Nome do arquivo
	arg1	2d
	arg2	Posição onde deve ser iniciada a exclusão
	arg3	Quantidade de caracteres que devem ser excluídos
Mensagem utilizada para indicar que houve mudança na posição do cursor.	*code*	15
	arg0	Nome do arquivo
	arg1	Número que indica a posição do cursor
Esta mensagem é utilizada para enviar aos ambientes remotos o conteúdo completo do arquivo. Assim, os ambientes remotos ficarão com o mesmo conteúdo do ambiente local. Esta mensagem é utilizada para sincronizar os conteúdos, caso ocorra algum erro.	*code*	17
	arg0	Nome do arquivo onde o conteúdo deve ser substituído
	arg1	Conteúdo do arquivo

4.6.3 Características visuais do ambiente

Quando recebe instruções de outros usuários, além de efetuar a alteração no código, o IdDE adiciona elementos visuais ao ambiente, como pode ser observado na Figura 40, que permitem localizar e identificar tanto as alterações quanto seus autores. Essas informações são importantes, pois auxiliam o usuário na análise e revisão síncrona *online* do código alterado, o que é essencial, segundo Carmel e Agarwal (2001).

Os recursos visuais implementados utilizam-se das APIs nativas do Netbeans. Dessa forma, numa adaptação do *plugin* para outro IDE, estas funcionalidades deverão ser reescritas utilizando as funcionalidades do novo ambiente.

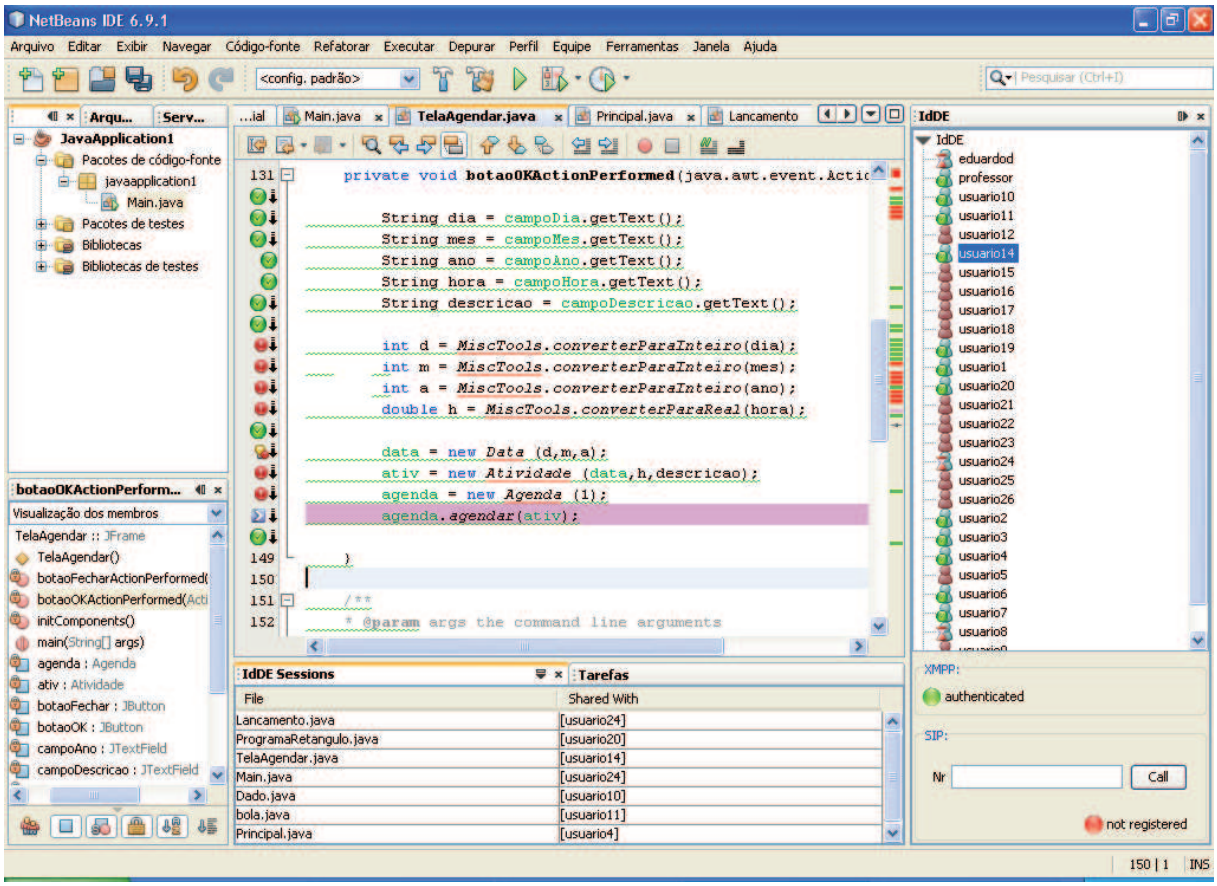


Figura 40: Texto alterado e marcações visuais

As marcações visuais adicionadas pelo IdDE ao Netbeans são a marcação do texto alterado e a indicação da posição do cursor do usuário remoto. A Figura 41 mostra detalhadamente essas marcações.

Quando o ambiente recebe uma mensagem de alteração de código de um usuário remoto, essas modificações são imediatamente aplicadas no código local e marcadas através da utilização de uma sublinha ondulada em verde. Além disso, as linhas que sofreram alterações são marcadas com um ícone com a forma de um círculo verde (Figura 41-A). Quando o usuário desloca o ponteiro do mouse sobre este ícone, são exibidas informações detalhadas como dados inseridos, coluna e autor, como mostra a imagem.

Outra marcação importante adicionada ao documento é a indicação da linha na qual está posicionado o cursor do usuário remoto. Essa marcação é representada por um ícone azul com uma seta (Figura 41-B). Como é possível que vários usuários editem o documento ao

mesmo tempo, para saber qual a posição de um determinado usuário, basta que se desloque o ponteiro do mouse sobre o ícone e o ambiente apresentará o nome do usuário, conforme pode ser observado na imagem. Quando o usuário remoto deslocar o cursor em seu ambiente, essa informação é igualmente atualizada no ambiente local. Nos estudos de caso realizados, pode-se perceber que essa característica contribui para o aumento da interatividade e experiência do usuário.

Em casos nos quais o código fonte é muito extenso, são criadas barras de rolagem no documento, da mesma forma como ocorre em outros editores. Nestas situações, pode acontecer de o usuário remoto fazer alterações na parte do texto que não está visível para o usuário local. Assim, para que este tenha ciência das alterações que estão sendo feitas no código, são criadas também as marcações indicadas pela Figura 41 (C). Nestas marcações, as cores identificam o tipo de ocorrência e, deslocando o mouse sobre as mesmas, são exibidas informações detalhadas, a exemplo do que ocorre quando se desloca o mouse sobre os ícones. Adicionalmente, é possível clicar nessas marcações, fazendo com que o editor traga para a área visível o trecho de código com a respectiva ocorrência.

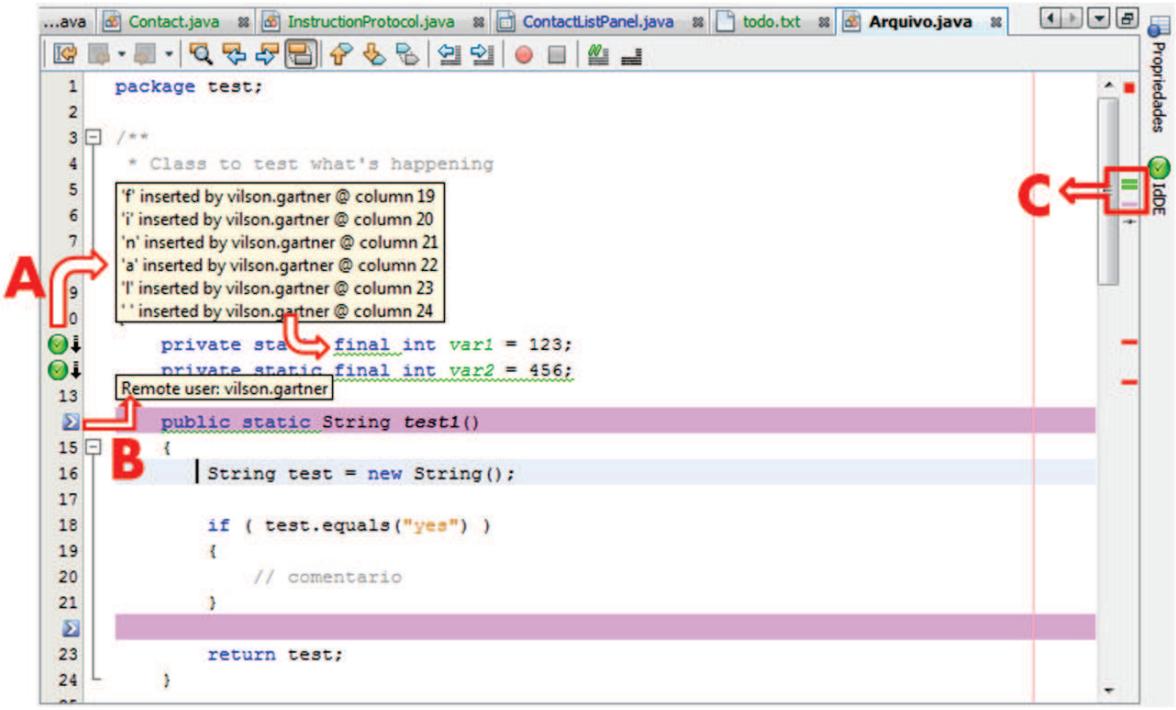


Figura 41: Marcação do texto alterado e posição do cursor

4.6.4 Funcionalidades adicionais

Considerando que, teoricamente, é possível estabelecer um número ilimitado de sessões, é importante proporcionar ao usuário a informação de quais arquivos estão sendo

compartilhados e quais usuários estão participando dessas sessões. Caso o usuário participe de muitas sessões de edição colaborativa, essas informações passam a ser ainda mais importantes.

Para auxiliar o usuário nesse aspecto, foi criada uma janela específica para manter as informações das sessões de edição colaborativa existentes, exibida Figura 42. Essa janela é composta por uma tabela, contendo o nome do arquivo compartilhado e os usuários participantes da sessão. Toda vez que uma nova sessão de edição colaborativa é estabelecida, uma nova linha é adicionada à tabela. Da mesma forma, quando um novo usuário passa a fazer parte da sessão, seu nome é adicionado àquela informação.

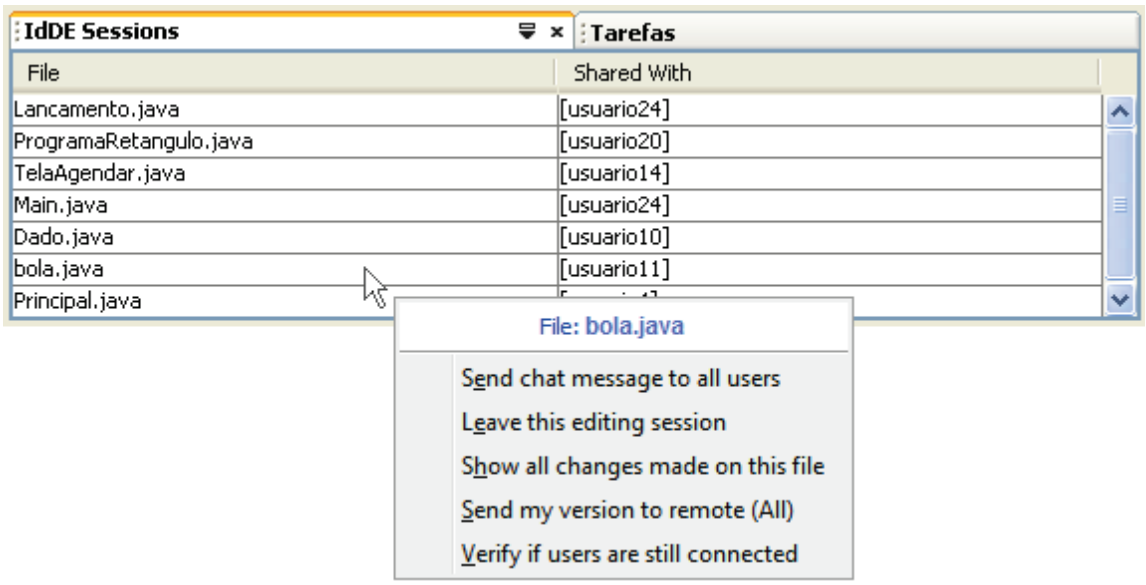


Figura 42: Janela de controle de sessões e menu de contexto

Além de exibir as informações sobre os arquivos e usuários das sessões, essa janela também dá acesso a outras funcionalidades relacionadas à sessão. Clicando com o botão direito sobre uma linha da tabela, é apresentado um menu de contexto, com as funções a seguir descritas:

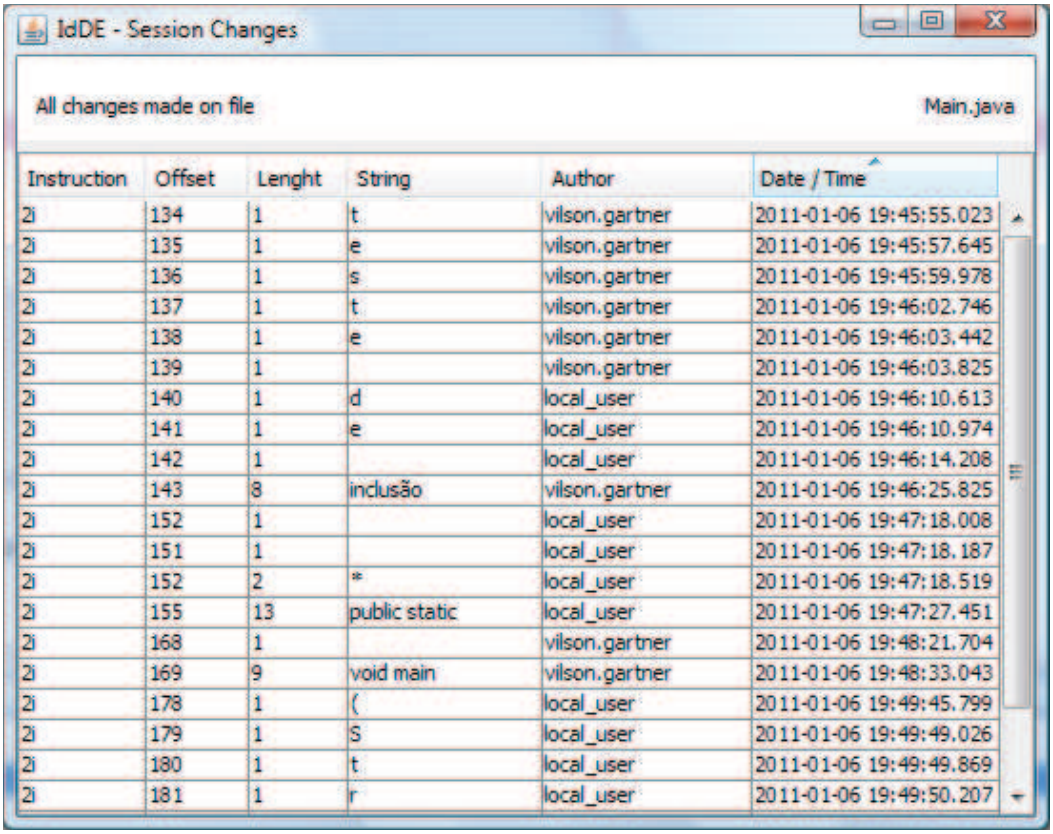
- "***Send chat message to all users***": através desta opção pode-se enviar uma mensagem de texto para todos os usuários participantes da sessão;
- "***Leave this editing session***": esta opção deve ser utilizada quando se deseja sair de uma sessão de edição colaborativa. Outras formas de sair de sessões são: finalizar a conexão com o servidor XMPP, fechar o documento ou ainda fechar o Netbeans. Em todos estes casos o ambiente envia uma mensagem alertando os ambientes remotos que o usuário está deixando a sessão.

- "***Show all changes made on this file***": acessando esta opção é apresentada ao usuário uma janela com todas as alterações feitas na sessão de edição selecionada. As informações são apresentadas numa tabela, conforme pode ser observado na Figura 43.

  Os dados constantes nessa tabela são: código do protocolo indicando o tipo de alteração ocorrida (inclusão ou exclusão), posição onde a modificação aconteceu, a informação incluída, o autor e momento da alteração. Quando se trata da exclusão de informações, o campo de informação incluída fica em branco.

  As informações da tabela podem ser ordenadas pressionando-se o título da coluna. Dessa forma, as informações podem ser agrupadas permitindo, por exemplo, uma análise para verificar as alterações feitas por um mesmo autor, tipo de alteração ou seguindo a ordem que as alterações ocorreram.

  Além de fornecer o histórico das alterações efetuadas, essa funcionalidade permite que se efetuem outras análises posteriores como, por exemplo, avaliar a lógica de programação utilizada por um usuário na resolução de um problema ou implementação de uma nova funcionalidade a um programa. Além disso, com base nessas informações é possível implementar uma ferramenta que permita a execução passo-a-passo de todas as alterações que ocorreram num arquivo, podendo-se simular todo o processo desde o princípio.
- "***Send my version to remote (All)***": esta opção é utilizada para enviar o código do ambiente local para os demais participantes da sessão. Esta opção era especialmente útil quando o ambiente ainda não implementava o controle da ordem das mensagens;
- "***Verify if users are still connected***": através desta opção é possível verificar se todos os usuários participantes da sessão continuam conectados. Caso um usuário remoto tenha sido desconectado de forma abrupta, ele permanecerá na lista de usuários da sessão e o ambiente continuará a enviar as mensagens de alterações. Assim, esta opção auxilia a eliminar usuários que não estiverem mais conectados.



Instruction	Offset	Lenght	String	Author	Date / Time
2i	134	1	t	vilson.gartner	2011-01-06 19:45:55.023
2i	135	1	e	vilson.gartner	2011-01-06 19:45:57.645
2i	136	1	s	vilson.gartner	2011-01-06 19:45:59.978
2i	137	1	t	vilson.gartner	2011-01-06 19:46:02.746
2i	138	1	e	vilson.gartner	2011-01-06 19:46:03.442
2i	139	1		vilson.gartner	2011-01-06 19:46:03.825
2i	140	1	d	local_user	2011-01-06 19:46:10.613
2i	141	1	e	local_user	2011-01-06 19:46:10.974
2i	142	1		local_user	2011-01-06 19:46:14.208
2i	143	8	inclusão	vilson.gartner	2011-01-06 19:46:25.825
2i	152	1		local_user	2011-01-06 19:47:18.008
2i	151	1		local_user	2011-01-06 19:47:18.187
2i	152	2	*	local_user	2011-01-06 19:47:18.519
2i	155	13	public static	local_user	2011-01-06 19:47:27.451
2i	168	1		vilson.gartner	2011-01-06 19:48:21.704
2i	169	9	void main	vilson.gartner	2011-01-06 19:48:33.043
2i	178	1	(	local_user	2011-01-06 19:49:45.799
2i	179	1	S	local_user	2011-01-06 19:49:49.026
2i	180	1	t	local_user	2011-01-06 19:49:49.869
2i	181	1	r	local_user	2011-01-06 19:49:50.207

Figura 43: Alterações feitas na sessão de edição

4.6.5 Arquitetura da ferramenta de edição colaborativa

A organização do código da ferramenta de edição colaborativa é representada pelo diagrama da Figura 44. O pacote *moduleIdDE*, que é responsável pelo integração do ambiente ao Netbeans, também possui as classes responsáveis pelo monitoramento dos eventos no documento. A classe *DocumentChangesListener* monitora as alterações (inclusão e exclusão de caracteres) que ocorrem no documento e a classe *DocumentCaretListener* é a responsável por fazer o monitoramento das alterações na posição do cursor. Também neste pacote estão as classes: *EditAnnotation*, responsável pela criação das características visuais no ambiente (marcações de alterações feitas e seus respectivos autores), e a classe *CaretPosAnnotation* responsável pelas marcações visuais relativas à posição do cursor dos usuários remotos.

O controle da sessão é implementada pela classe *SessionControl*, do pacote *editor::controller*. Entre os métodos importantes dessa classe estão: *getEditSession*, responsável por iniciar uma nova sessão de edição colaborativa; *addSharedEditing*, responsável por adicionar um novo arquivo e usuário a uma sessão; *getSharedFiles*, método utilizado para retornar os arquivos que estão numa sessão de edição; *getSharedUsers*,

utilizado para retornar os usuários que participam de uma sessão; *pingAlive*, utilizado para verificar se os usuários continuam na sessão; *exitSharedEditing*, executado quando um usuário sair de uma sessão compartilhada; e *addMessageToBuffer*, método utilizado para enviar uma mensagem do protocolo para os demais usuários da sessão. As informações das sessões existentes são mantidas no atributo *editSessions*. Os objetos armazenados neste atributo são do tipo *Party*, que mantém a informação do usuário (classe *common::model::Contact*), nome do arquivo e canal de comunicação aberto com esse usuário (classe *chat::view::ChatFrame*).

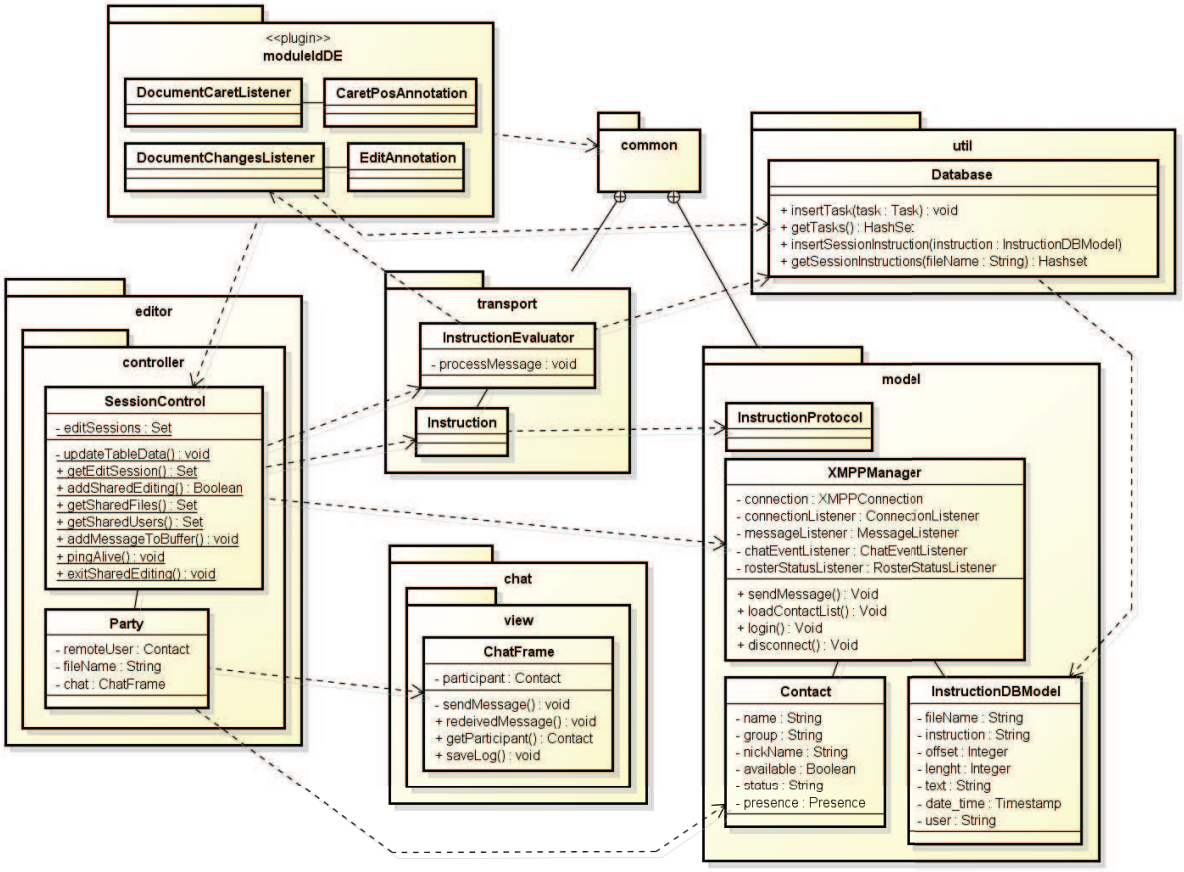


Figura 44: Arquitetura da ferramenta de edição colaborativa

Quando uma mensagem de alteração ou posição do cursor deve ser enviada, essa informação é enviada a o método *addMessageToBuffer*, responsável pelo controle do buffer das mensagens a serem enviadas. O tempo de buffer é controlado pela configuração indicada na configuração do ambiente (Figura 33-F). Para o envio da mensagem, é utilizado o atributo connection, da classe *common::model::XMPPManager*. A classe *InstructionEvaluator*, do

pacote *common::transport* é a responsável pela análise e verificação da instrução recebida de um usuário remoto.

As informações sobre as alterações efetuadas num documento são armazenadas no banco de dados utilizando o método *insertSessionInstruction* da classe *Util::Database*. Nessa mesma classe existe o método *getSessionInstructions*, utilizado para a obtenção das alterações feitas num documento.

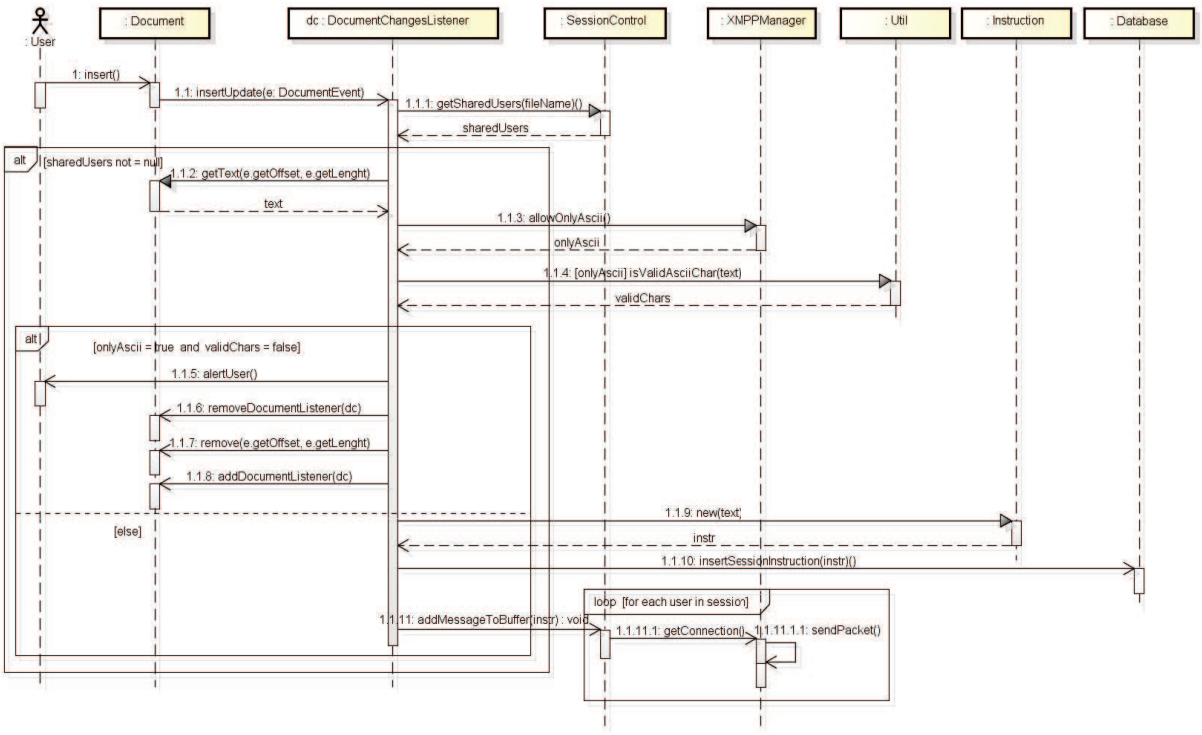


Figura 45: Diagrama de sequência do monitoramento e envio de mensagem de inclusão

O diagrama da Figura 45 procura representar, de forma um pouco mais detalhada, a sequência em que são executadas as atividades no momento em que ocorre a inclusão de texto no documento, bem como as classes e métodos envolvidos. Quando um usuário efetua uma inclusão, o *DocumentChangesListener* registrado no documento é acionado imediatamente. A primeira providência deste método é verificar se o arquivo está em alguma sessão de edição colaborativa. Para isso, executa o método *getSharedUsers* da classe *SessionControl*. O retorno deste método é um objeto do tipo *Set* contendo os usuários da sessão. Quando este retorno for igual a "*null*", indica que não existe uma sessão estabelecida para o arquivo. Caso contrário (existe uma sessão), será necessário obter a *string* incluída. Para isso, é executado o método *getText* no objeto *Document*, passando como parâmetros a posição onde ocorreu a inserção e a quantidade de caracteres inseridos. O passo seguinte é a verificação relativa ao uso de

caracteres ASCII básicos, conforme descrito anteriormente. Caso a string contenha caracteres não permitidos, é apresentada uma mensagem ao usuário informando o fato e, em seguida, o *listener* é removido do documento (para não entrar em *loop*), a *string* inserida pelo usuário é removida do documento e o *listener* é adicionado novamente. Por outro lado, se a string for válida, é criada uma instrução do tipo *Instruction* com as informações da inserção. Essa informação é então armazenada no banco de dados e enviada para *addMessageToBuffer* da classe *SessionControl*, que se encarrega de enviar a mensagem XMPP com a instrução encapsulada para os demais usuários da sessão, utilizando a conexão existente com o servidor, armazenada em *XMPPManager*.

As atividades executadas na exclusão de código estão representadas na Figura 46. Como é possível observar, neste processo não existem verificações adicionais. Assim que o *DocumentChangesListenter* detecta uma exclusão no documento, ele verifica se o documento está numa sessão. Se estiver, cria uma instrução (*Instruction*) de exclusão e armazena a informação na base de dados. Em seguida, executa o método *addMessageToBuffer* que envia a instrução para os demais participantes da sessão, da mesma forma como acontece na inclusão.

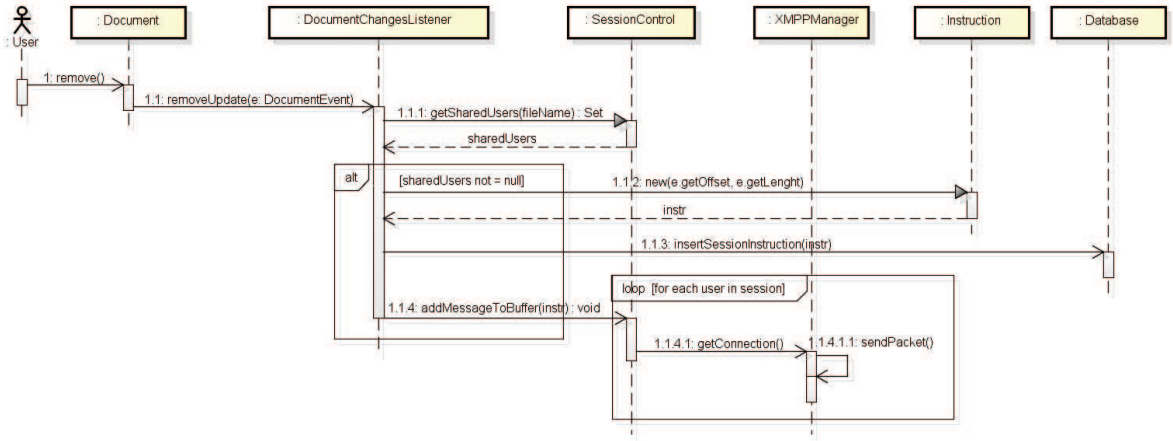


Figura 46: Diagrama de sequência da exclusão de código

A sequência de atividades executadas no monitoramente da mudança de posição do cursor, está representada na Figura 47.

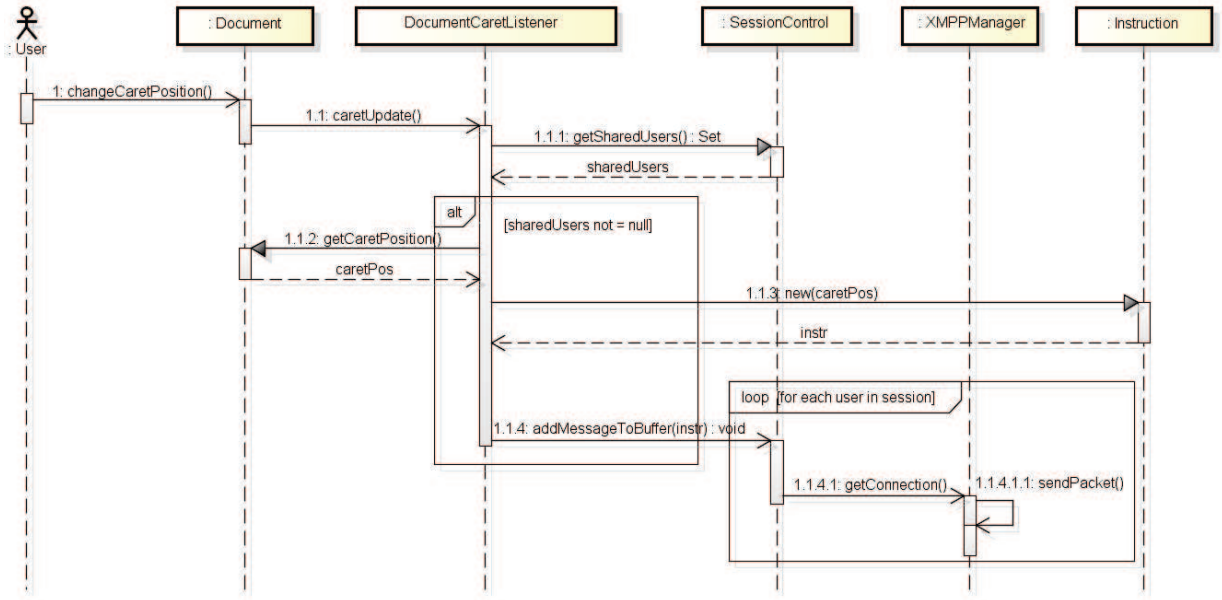


Figura 47: Sequência de atividades da mudança de posição do cursor

Quando ocorre uma mudança na posição do cursor é executado o método *caretUpdate*, da classe *DocumentCaretListener*, que está registrada no documento ativo. Se o documento estiver sendo compartilhado, é obtida a posição do cursor através do método *getCaretPosition*, da classe *Document*. Em seguida, é criada uma instrução de posicionamento do cursor (classe *Instruction*). Por fim, a instrução é encapsulada numa mensagem XMPP e enviada pelo método *addMessageToBuffer*, da mesma forma como ocorre com a inclusão e exclusão de caracteres.

4.7 Controle de versões

No contexto do IdDE, as sessões de edição colaborativa permitem o acompanhamento das alterações em tempo real, enquanto que o uso da ferramenta de controle de versões mantêm uma visão macro das alterações efetuadas ao longo do tempo. O controle de versões está integrada diretamente no Netbeans e, dessa forma, todas as alterações feitas num arquivo automaticamente ficam registradas num repositório local, controlado pelo Netbeans. Além disso, esse IDE também possui suporte e integração com outros repositórios como: Subversion[23], Mercurial[24] e CVS[25].

[23] http://subversion.tigris.org/
[24] http://mercurial.selenic.com/
[25] http://savannah.nongnu.org/projects/cvs/

A Figura 48 apresenta a visualização de diferenças entre duas versões de um mesmo arquivo. Como é possível observar, o Netbeans apresenta uma tabela contendo as versões do arquivo, permitindo a sua seleção (A). Ao clicar numa determinada versão, o programa exibe do lado esquerdo (B) a versão do código naquele momento e, do lado direito da janela (C), o estado atual do código do arquivo. Para facilitar a visualização das diferenças, adiciona ainda marcações informando onde e como o texto foi modificado. Essa funcionalidade de visualização é nativa do Netbeans, não sendo necessária nenhuma implementação adicional no IdDE.

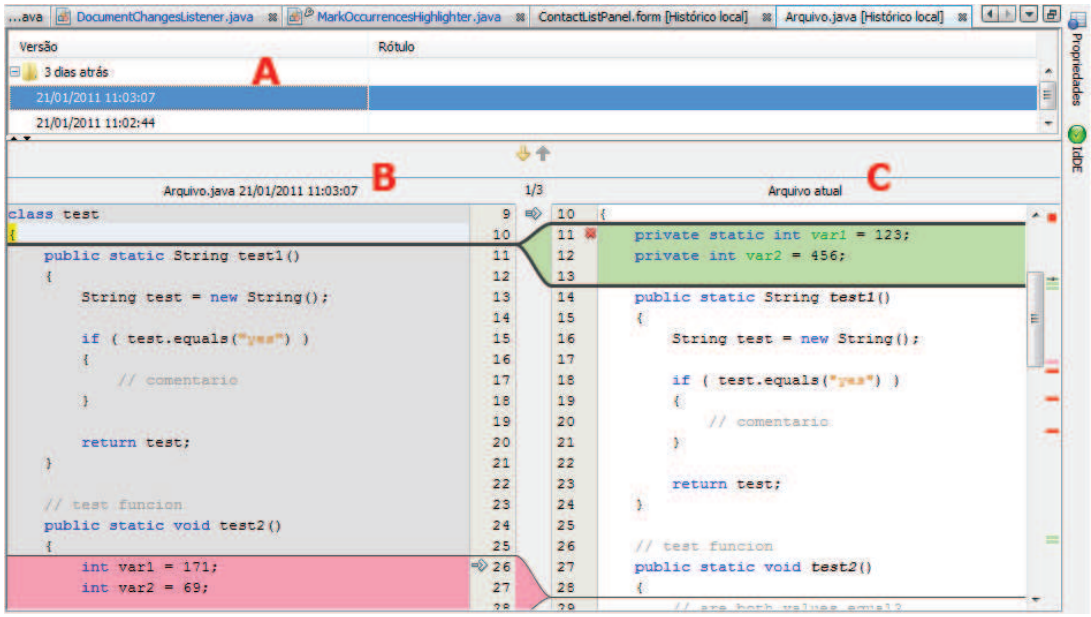


Figura 48: Visualização de diferenças

4.8 Mensagens Instantâneas (IM)

O objetivo da ferramenta de comunicação do IdDE é possibilitar a comunicação através de mensagens instantâneas de texto, a exemplo de MSN[26], Google Talk[27], entre outros. Para que seja possível a comunicação com outros usuários é necessário que eles também estejam conectados ao servidor XMPP. Além de se comunicar com usuários do ambiente IdDE, também é possível trocar mensagens com outros aplicativos que utilizam esse

[26]MSN: http://br.msn.com
[27]Google Talk: http://www.google.com/talk

protocolo para comunicação, e que estejam conectados ao mesmo servidor, conforme apresentado na Tabela 2.

4.8.1 Interface com o usuário

Para iniciar a comunicação via chat no ambiente, deve-se efetuar um duplo clique sobre o respectivo contato da lista (Figura 34). Outra forma, é pressionar o botão direito no contato e selecionar a opção apropriada no menu de contexto. Diferentemente do que ocorre com as outras ferramentas, para estabelecer uma comunicação via chat não ocorre nenhuma negociação adicional utilizando o protocolo IdDE. Dessa forma, a comunicação pode ser estabelecida com outros aplicativos, além do IdDE.

A janela de chat, apresentada na Figura 49-A, possui três elementos de interação com o usuário: campo para a digitação da mensagem, o botão para o envio e o botão para tradução.

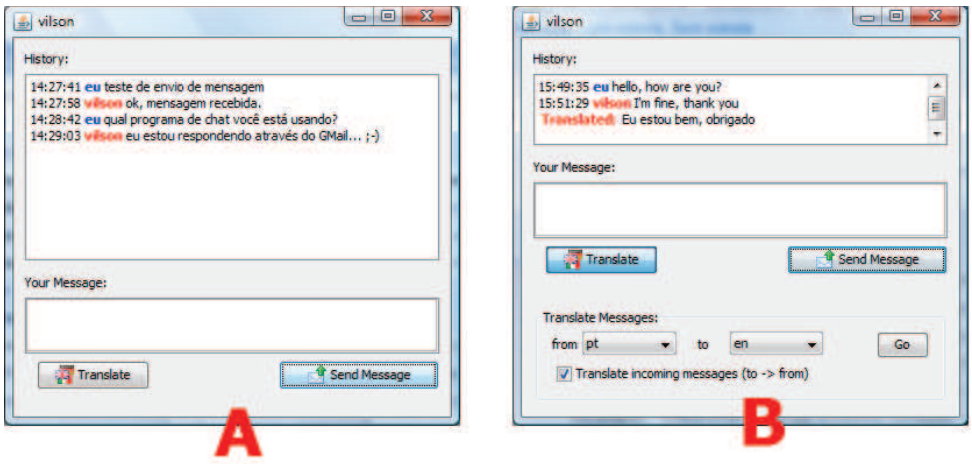


Figura 49: Janela de *chat*

Para possibilitar que equipes de idiomas nativos diferentes consigam se comunicar sem problemas, o IdDE auxilia no processo de tradução de mensagens. Assim, se um usuário que somente fala português precisar se comunicar com outro que somente fala inglês, ele poderá utilizar a ferramenta de tradução. As opções de tradução são exibidas quando é pressionado o botão "*Translate*", como mostra a Figura 49-B. Para traduzir a mensagem escrita, antes de enviar, deve-se selecionar o idioma de origem e o idioma de destino e, por fim, pressionar o botão "*Go*". Quando a opção "*Translate incoming messages*" estiver habilitada, o IdDE automaticamente fará a tradução das mensagens recebidas, apresentando tanto a mensagem original quanto sua tradução, no painel de histórico de mensagens. Nesta

situação, o IdDE inverte os idiomas, ou seja, a origem passa a ser a opção selecionada no campo "*to*" e o destino será a opção selecionada em "*from*". Para efetuar a tradução, o IdDE utiliza o sistema de tradução do Google[28].

Todas as mensagens enviadas e recebidas são armazenas em um *log* (arquivo texto), na pasta de arquivos do IdDE. Em trabalhos futuros, pretende-se incorporar o acesso ao histórico das mensagens, diretamente na tela do chat ou através do menu de contexto.

4.8.2 Arquitetura da ferramenta de mensagens instantâneas

A Figura 50 apresenta a arquitetura da ferramenta de mensagens instantâneas.

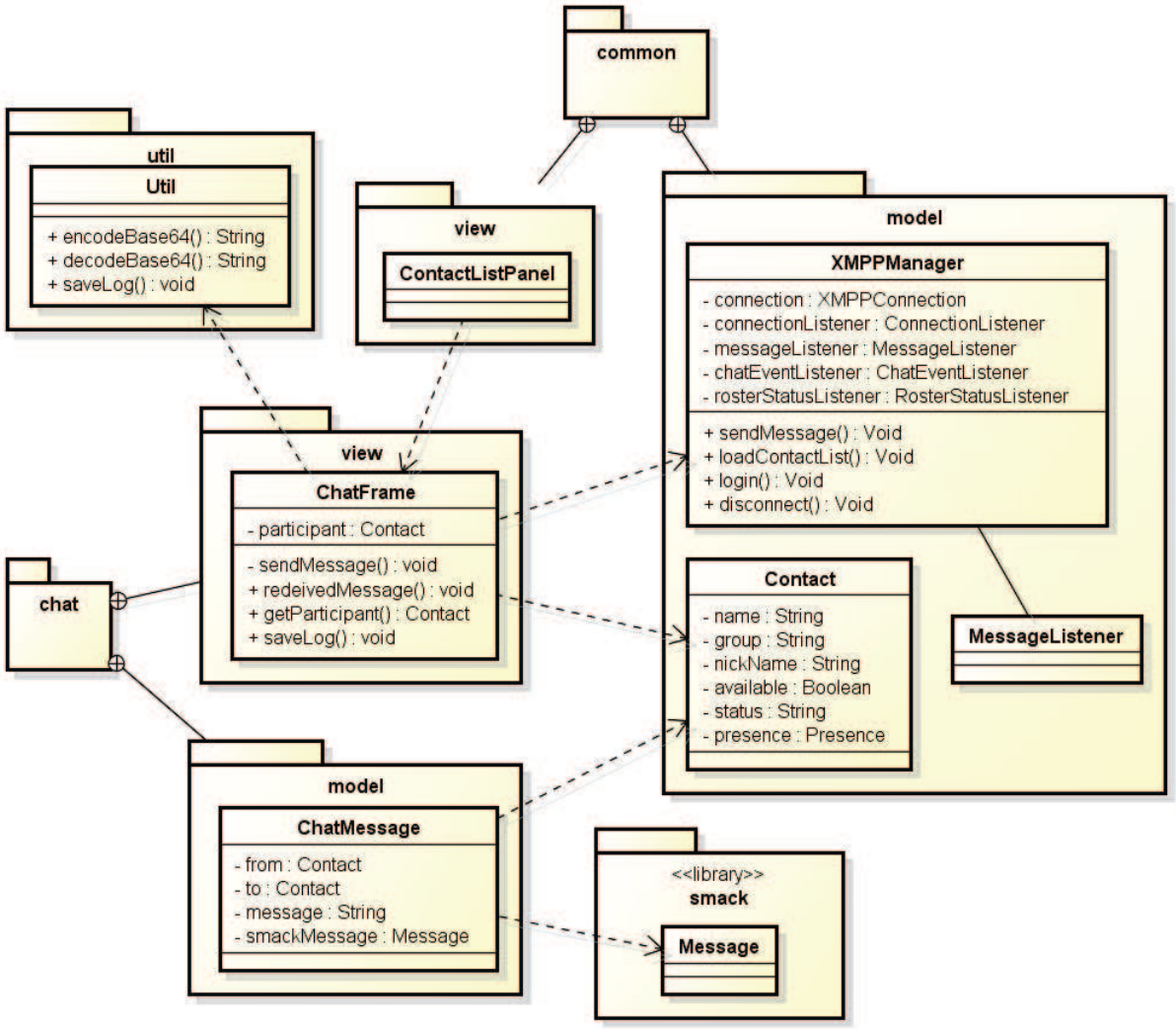


Figura 50: Arquitetura da ferramenta de chat

[28] http://translate.google.com

As classes responsáveis por esse módulo se encontram no pacote chat, o qual é subdividido em *view* e *model*. No subpacote *view* é encontrada a classe *ChatFrame*, responsável por criar a interface e pela troca de mensagens. Para cada chat aberto com um usuário, é instanciado um novo objeto dessa classe. A informação do usuário com o qual estão sendo trocadas mensagens fica armazenada no atributo *participant*. O método público *receivedMessage* é acionado pela classe *ContactListPanel*, quando uma nova mensagem de chat é recebida de um contato. O método *sendMessage* é o responsável pelo envio das mensagens de chat. Para este envio, é utilizado o método *sendMessage* da classe *XMPPManager*. Para o transporte das mensagens entre as classes, é utilizado um objeto do tipo *ChatMessage*. Para efetuar a gravação do *log* de mensagens, é executado o método *saveLog*, que interage com o método de mesmo nome da classe *Util*.

As atividades de recebimento de mensagens de chat seguem a sequência representada na Figura 30.

4.9 Comunicação via Áudio

O módulo de comunicação via áudio do IdDE, além de permitir a comunicação entre os usuários do ambiente, possibilita a comunicação com outros dispositivos, a exemplo de telefones móveis, *smartphones* ou mesmo telefones fixos.

Exemplificando, se um gerente desejar contatar um desenvolvedor, poderá fazê-lo utilizando o seu *smartphone*. Igualmente, se um desenvolvedor precisar entrar em contato com um cliente, para saber mais informações sobre determinado requisito de *software*, poderá, através da ferramenta de comunicação de áudio, contatar diretamente o telefone fixo ou celular do cliente.

A Tabela 2 apresenta os dispositivos que podem ser utilizados na comunicação através de áudio. É importante observar que, para que seja possível a comunicação via áudio entre os ambientes, somente é necessária a existência de um servidor SIP. Por outro lado, para que a comunicação possa ocorrer com telefones celulares e telefones fixos (que não utilizam SIP), é necessário que o servidor SIP esteja interconectado com a rede PSTN, conforme mostra a Figura 20.

Além de estabelecer a comunicação direta com outro usuário, o ambiente permite também que sejam estabelecidas conferências com a participação de mais usuários. Todavia, esta funcionalidade está restrita ao SIP, não sendo possível com equipamentos da PSTN.

4.9.1 Interface com o usuário

Para efetuar uma chamada de áudio para outro usuário, também conectado ao servidor SIP, basta inserir o número do ramal no campo da tela principal do IdDE, destacado na Figura 51-A. O usuário remoto não precisa, necessariamente, estar utilizando o ambiente de desenvolvimento. A comunicação pode ocorrer via *smartphone* ou outro aplicativo/dispositivo que utilize o protocolo SIP.

Para efetuar uma chamada para um número de telefone fixo ou celular, além de inserir o respectivo número é necessário a aquisição de créditos junto a empresas prestadoras de serviço VoIP.

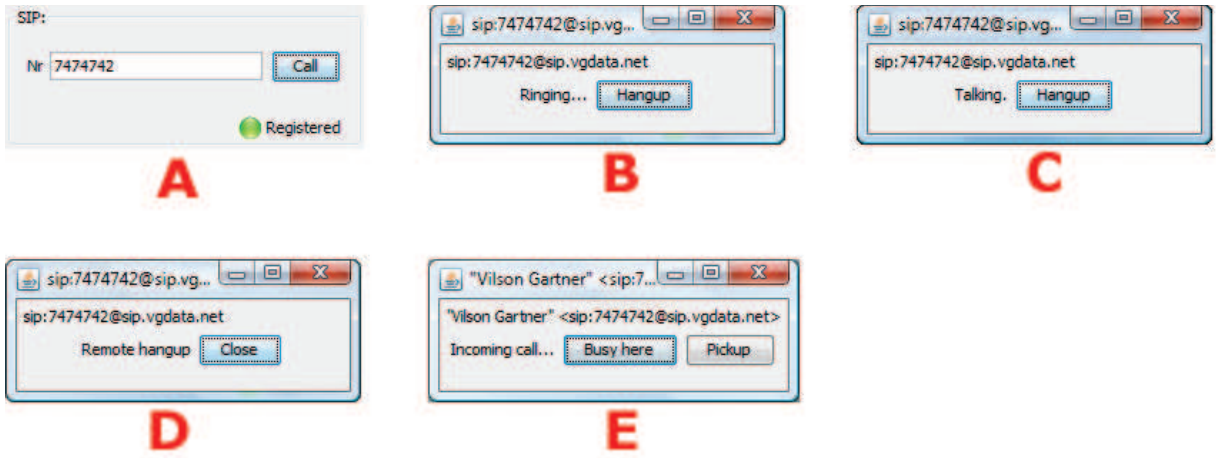


Figura 51: Telas da comunicação via áudio

Ao pressionar o botão "*Call*", o IdDE apresenta uma mensagem indicando a conexão com o dispositivo remoto está sendo feita. Quando a conexão for bem sucedida e o dispositivo remoto estiver tocando, o usuário será notificado através da mensagem apresenta da Figura 51-B. Quando o usuário atende a ligação, o ambiente alerta o usuário apresentando a mensagem da Figura 51-C. Se o usuário remoto não aceitar a ligação, é exibida a mensagem Figura 51-D. Esta mesma mensagem é exibida quando o usuário remoto desligar durante uma ligação estabelecida. Quando é recebida uma solicitação para conversa via áudio, o IdDE exibirá a mensagem da Figura 51-E, possibilitando que se aceite ou rejeite a ligação. Em ocasiões onde algum erro ocorre, o sistema apresenta uma mensagem informando o motivo.

Nas situações que não se conhece os números de contato do usuário, pode-se utilizar o IdDE para obter essas informações. Para isso, basta clicar com o botão direito do mouse sobre o respectivo contato e acessar as opções no menu de contexto, conforme indicado na Figura

52. É importante observar que, neste caso, mesmo que a conexão de áudio seja estabelecida utilizando o servidor SIP, é necessário que o usuário remoto esteja conectado ao servidor XMPP, pois, são necessárias trocas de mensagens do protocolo IdDE entre os ambientes.

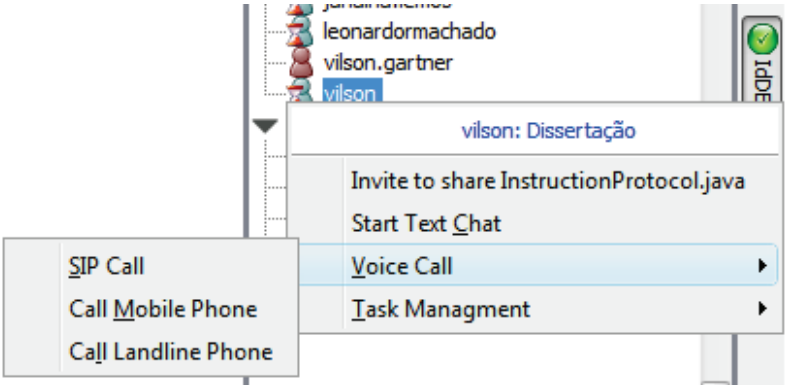


Figura 52: Menu de contexto e opções de chamada de áudio

4.9.2 Mensagens do protocolo IdDE

O diagrama de atividades da Figura 53 apresenta o fluxo do processo para a obtenção do número de contato do usuário remoto. Inicialmente, o usuário deverá localizar o contato na lista e clicar com o botão direito do mouse para abrir o menu de contexto. Em seguida, deverá selecionar a opção de acordo com o número que deseja saber (SIP, fixo ou celular). Ao executar a seleção, o ambiente local prepara a instrução do protocolo IdDE e envia ao ambiente remoto, através de uma mensagem XMPP. O ambiente remoto, ao receber a mensagem, analisa a instrução e então busca a informação nas configurações do usuário. O passo posterior é criar uma instrução de resposta utilizando o protocolo IdDE e enviar a resposta através de uma XMPP ao usuário que originou a solicitação. O ambiente de onde partiu a solicitação recebe a resposta, que é então exibida ao usuário. Quando a informação solicitada não for encontrada pelo ambiente remoto, a mensagem de resposta avisará este fato.

As informações dos números de telefone e ramal SIP devem ser previamente configuradas, conforme descrito anteriormente e mostrado na Figura 33-F.

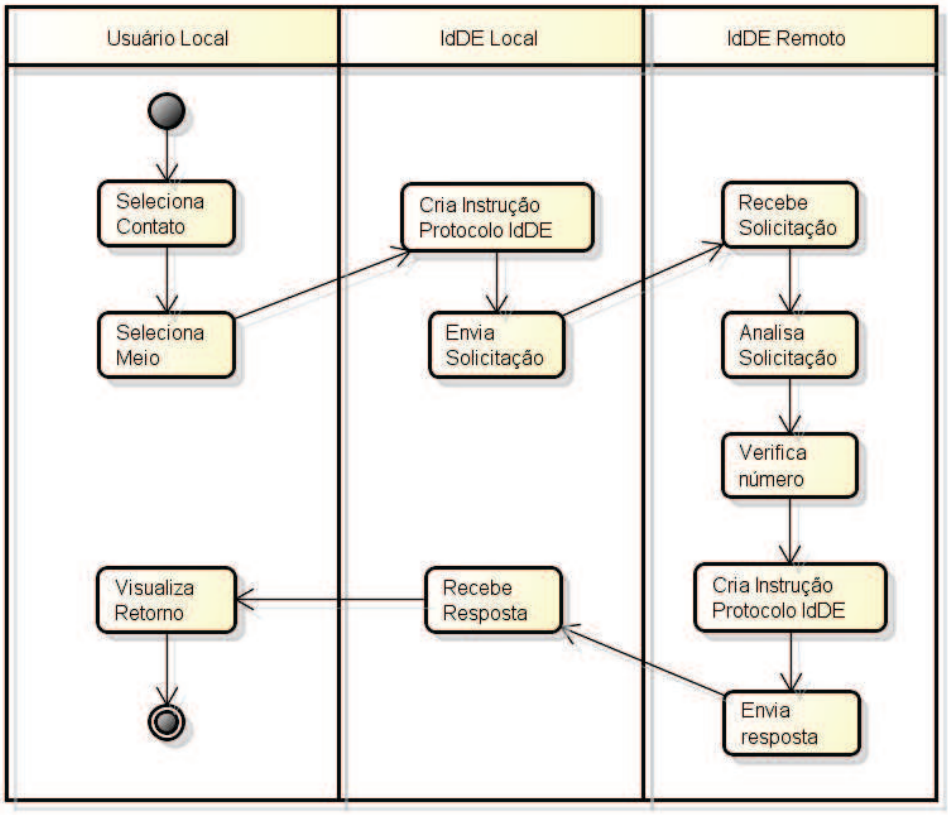


Figura 53: Diagrama de atividades para obter número de contato

A Tabela 8 descreve, detalhadamente, as instruções do protocolo IdDE utilizadas nas mensagens de negociação entre os ambientes, para a obtenção dos números de telefone e ramal SIP.

Tabela 8: Instruções do protocolo IdDE utilizados pela ferramenta de áudio

Objetivo	*Tag*	Conteúdo da *Tag*
Mensagem utilizada para solicitar o número do ramal SIP do usuário remoto.	*code*	4
Mensagem de resposta com o número do ramal SIP.	*code*	5
	arg0	Número do ramal
Mensagem para solicitar o número do telefone celular.	*code*	6
Mensagem com a resposta do número de telefone celular.	*code*	7
	arg0	Número do telefone celular
Mensagem utilizada para solicitar o número do	*code*	8

Objetivo	*Tag*	Conteúdo da *Tag*
telefone fixo do usuário.		
Mensagem de resposta com o número de telefone fixo.	*code*	9
	arg0	Número do telefone fixo
Mensagem indicando que o número solicitado não está configurado.	*code*	10
	arg0	Código da solicitação original recebida

4.9.3 Arquitetura da ferramenta de comunicação via áudio

A Figura 54 procura representar a arquitetura da ferramenta. Nessa visão lógica da implementação, são apresentados os principais pacotes e classes, bem como suas dependências.

A tela principal da ferramenta SIP, a partir da qual são originadas as chamadas, é implementada pela classe *sip::view::MainSIPFrame*, e é adicionada à tela principal do ambiente por meio da classe *common::view::MainPanel*, no momento da inicialização do ambiente.

Igualmente como ocorre nas demais ferramentas do IdDE, o gerenciador da conexão XMPP (*common::model::XMPPManager*) é o responsável pelo envio das mensagens (método *sendMessage*) de negociação do protocolo IdDE, relativas às solicitações do número de ramal SIP e números telefônicos, bem como suas respectivas respostas. A solicitação dessas informações é feita através do menu de contexto na lista de contatos, implementada pela classe *common::view::ContactListPanel*.

No pacote *sip* são feitas as implementações relativas à comunicação SIP propriamente dita. Quando a tela principal dessa ferramenta é criada, é instanciado também um objeto da classe *SIPManager* (*sip::model*). Este objeto é responsável por controlar as janelas de mensagens das ligações (atributo *callFrames*) e monitorar o recebimento de chamadas através implementação da interface *sip::core::useragent::SipListener* da biblioteca Peers, utilizada para a comunicação SIP. O atributo *userAgent* (classe *peers::sip::core::UserAgent*) é responsável por implementar funcionalidades como: efetuar o registro (login), controle dos pacotes de comunicação, entre outros.

No pacote *sip::view* está localizada a classe *CallFrame*, que é a responsável por implementar os controles visuais comuns às telas de mensagens. Esta classe também é a

responsável por fazer as chamadas ao ramais/telefones remotos, por meio do objeto *sipRequest* (classe *peers::transport::SipRequest*). Já o objeto *callListener* (classe *sip.model.SIPManager*) é encarregado de interceptar as ligações que estão chegando.

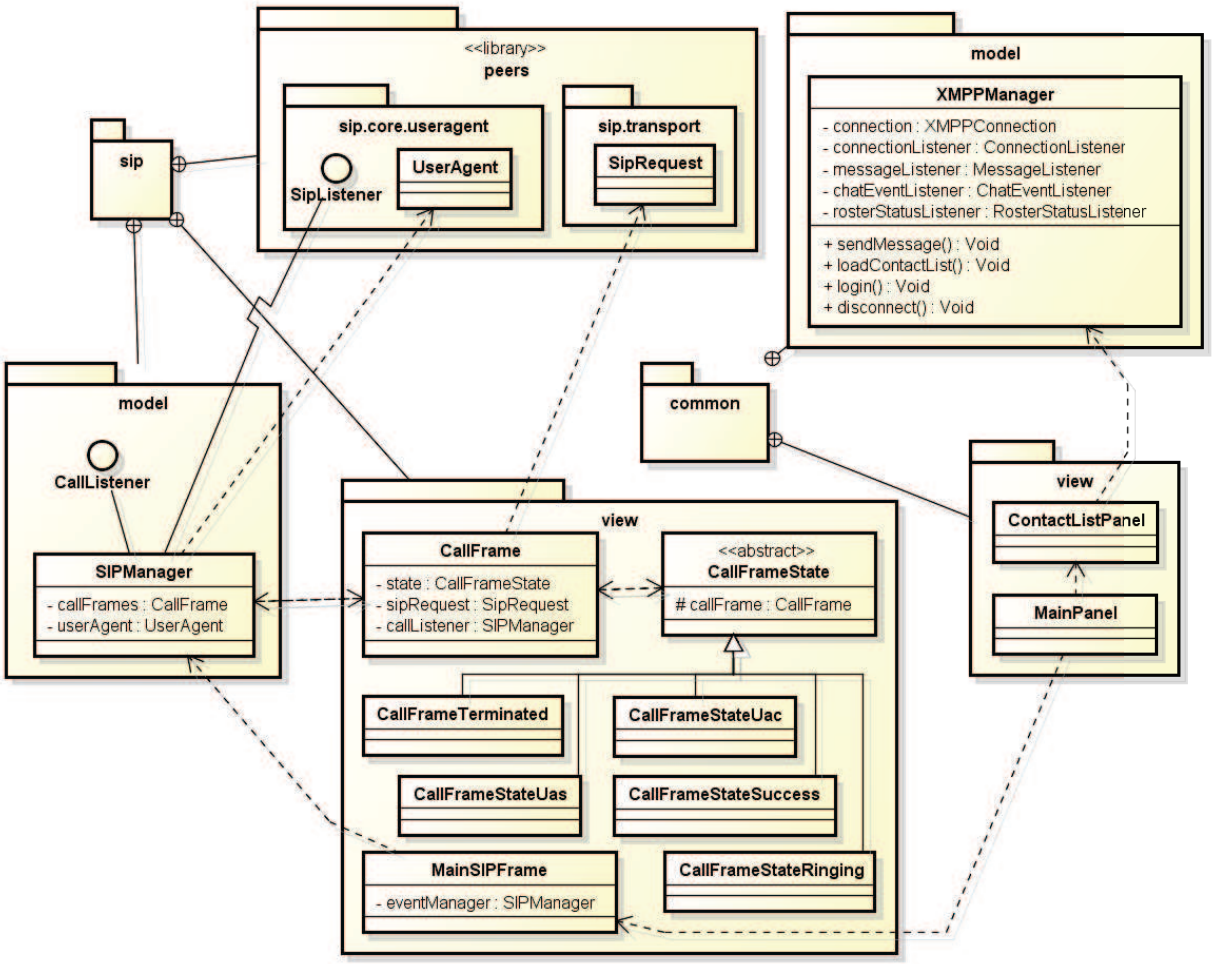


Figura 54: Arquitetura da ferramenta de áudio

Nesse pacote também ficam as classes responsáveis pela implementação da telas de mensagens dos estados das chamadas, apresentadas na Figura 51, entre as quais estão:

- *CallFrameStateUac*: essa classe apresenta ao usuário as informações que a chamada está sendo estabelecida, quando este tela que é apresentada ao usuário quando este solicita uma chamada de áudio;
- *CallFrameStateRinging*: implementa a tela que indica que o dispositivo remoto está chamando o usuário;
- *CallFrameStateSuccess*: essa classe implementa a tela que alerta o usuário que a comunicação está estabelecida;

- *CallFrameStateTerminated*: implementa a tela que informa que a conexão foi terminada pelo usuário remoto;
- *CallFrameStateUas*: essa classe é responsável por implementar a tela que é apresentada ao usuário quando um usuário remoto está solicitando o estabelecimento de uma sessão de áudio;
- *CallFrameState*: é uma classe abstrata que possui algumas implementações abstratas para as classes anteriores. Um dos atributos importantes dessa classe é *callFrame*, que é uma instância da tela de apresentação das mensagens, utilizadas pelas outras classes.

4.10 Gerenciamento de Tarefas

Através da ferramenta de controle de tarefas do IdDE, será possível atribuir tarefas aos programadores, acompanhar sua execução e progresso, permitindo que os programadores adicionem observações às mesmas.

No modelo proposto pelo ambiente, as tarefas não ficam armazenadas num banco de dados centralizado, mas sim, distribuídas de forma descentralizada no banco de dados de cada usuário. Dessa forma, quando um usuário quiser saber o detalhamento das tarefas de outro, basta acessar a opção apropriada no ambiente.

Em trabalhos futuros, pretende-se estudar a viabilidade da integração desse módulo a ferramentas *online* como, por exemplo, o Google *Calendar*[29]. Essa integração possibilitaria que outros *stakeholders* que não possuam acesso ao ambiente de desenvolvimento possam acessar as informações relativas às tarefas.

4.10.1 Interface com o usuário e fluxo de mensagens

As funcionalidades de gerenciamento também são acessadas através do menu de contexto na lista de contatos, conforme mostra a Figura 55. No menu, existem três opções disponíveis, descritas a seguir:

- *Add task for this user*: esta opção deve ser utilizada para adicionar uma nova tarefa para o usuário. A Figura 56 apresenta a tela de inclusão de tarefa;

---

[29]Google Calendar: http://www.google.com/calendar

- *View user's tasks*: Esta opção deve ser utilizada para ver as tarefas existentes para o usuário remoto. Nessas solicitações, o ambiente remoto responde com uma mensagem de *chat*, listando as tarefas do usuário;
- *Open task manager*: Essa opção apresenta uma tela com as tarefas do usuário local. Através dessa tela é possível alterar informações.

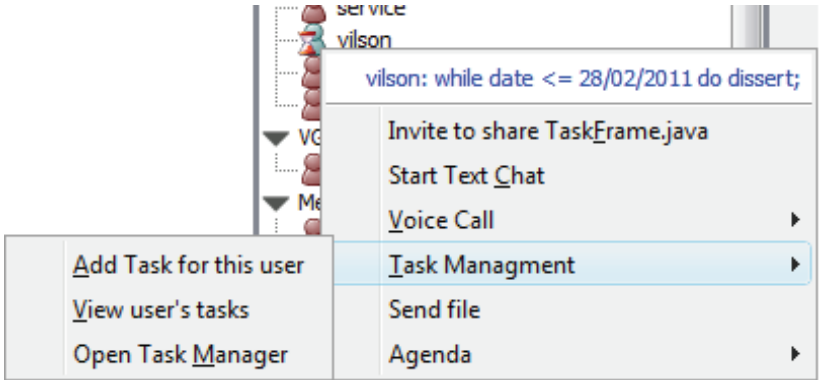


Figura 55: Menu de contexto e opções relacionadas a tarefas

Os campos da tela de inclusão de uma tarefa, apresentada na Figura 56, são:

- *Name*: nome resumido, que indica o assunto da tarefa;
- *Project*: nome do projeto ao qual a tarefa se refere;
- *Due Date*: data final para entrega da tarefa;
- *Priority*: campo de seleção utilizado para indicar a prioridade desta tarefa. As prioridades podem ser: Lowest, Low, High, Highest;
- *Full Task Description*: descrição completa da tarefa.

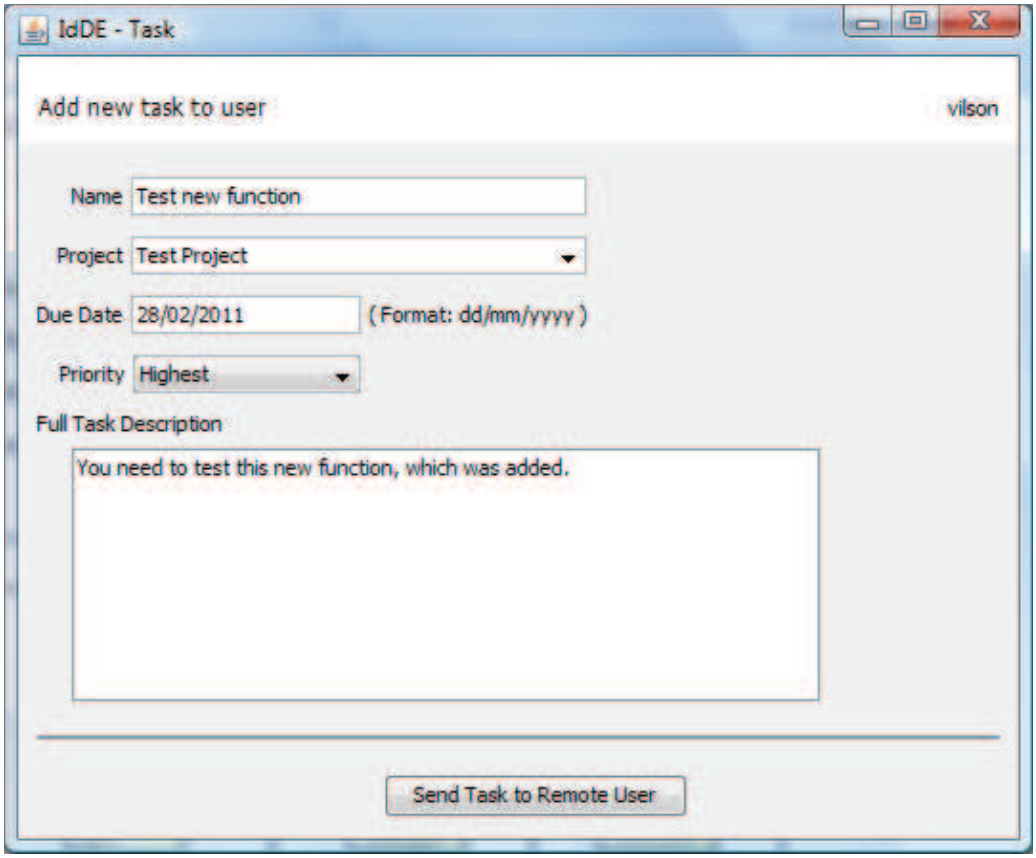


Figura 56: Tela para inclusão de tarefa

O processo de criação e envio de uma nova tarefa para um usuário segue o fluxo de atividades representadas pela Figura 57. Após selecionar a opção apropriada no menu de contexto, será apresentado ao usuário a tela na qual devem ser informados os dados da nova tarefa (Figura 56).

Ao pressionar o botão de envio da tarefa para o usuário remoto, o ambiente local criará a instrução do protocolo IdDE relativa à nova tarefa, encapsulará essa instrução numa mensagem XMPP e enviará ao ambiente remoto.

Quando o ambiente remoto receber a mensagem XMPP, seguindo a sequência de atividades apresentadas no diagrama da Figura 30, analisará a instrução e, detectando que se trata da criação de uma nova tarefa, apresentará as informações da nova tarefa ao usuário, conforme apresentado na Figura 58.

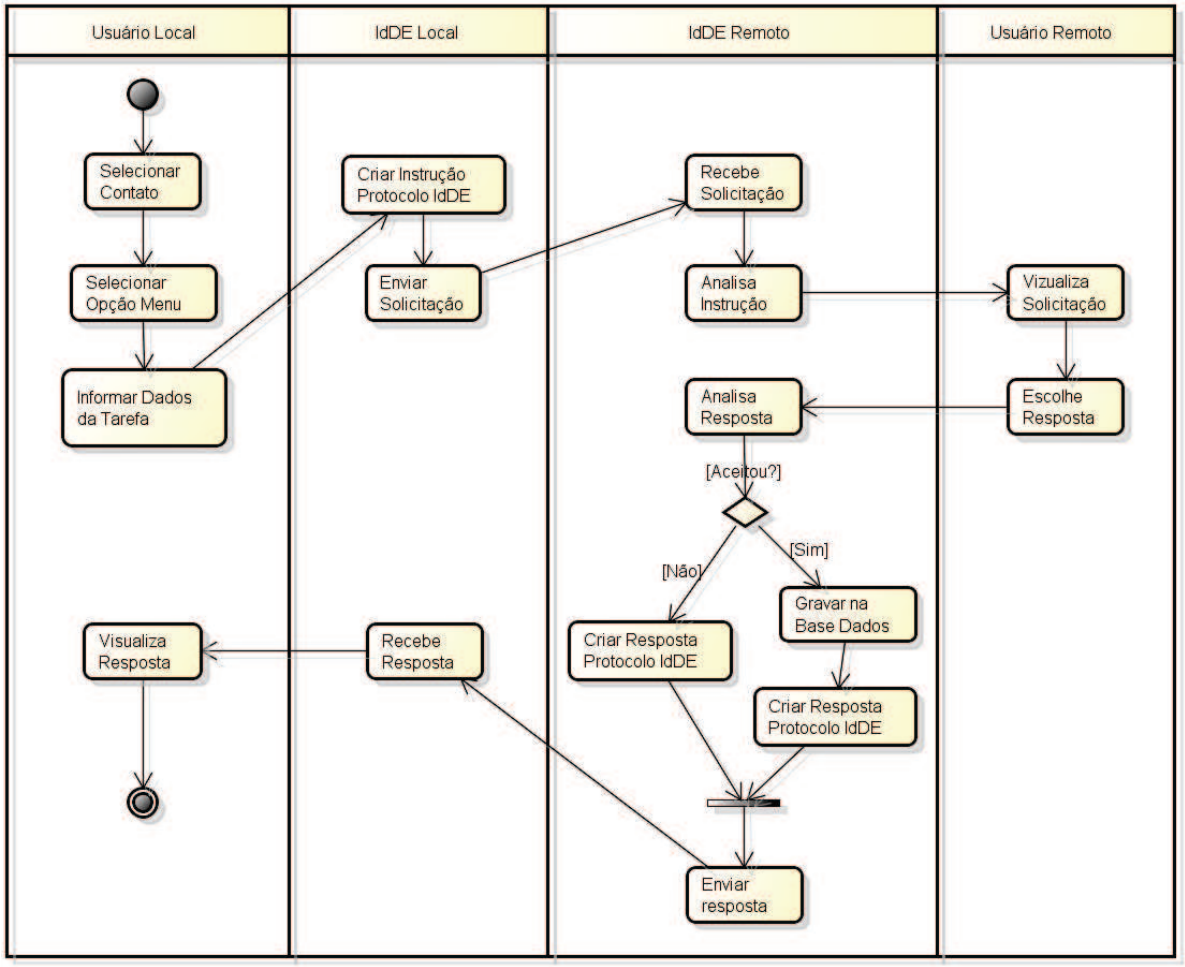


Figura 57: Fluxo de atividades para inclusão de tarefa

Diante da solicitação, o usuário deverá aceitar ou rejeitar a nova tarefa. Se aceitar, o ambiente armazenará a tarefa no banco de dados local (SQLite) e criará a mensagem de resposta. Caso contrário, apenas criará a resposta para o usuário remoto e nenhuma informação será armazenada.

Por fim, o ambiente envia a mensagem ao ambiente de onde partiu a instrução. Ao receber a resposta, este apresenta ao usuário a informação sobre o resultado do envio da nova tarefa. É importante observar que as mensagens de reposta são enviadas diretamente através da interface de *chat*, o que possibilita que os usuários já se comuniquem em caso de não haver concordância com o resultado da criação da tarefa.

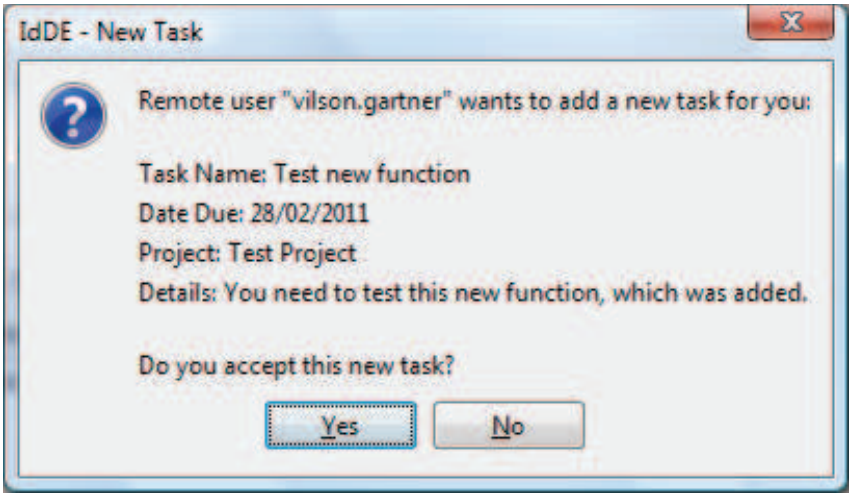


Figura 58: Mensagem solicitando a criação de nova tarefa

4.10.2 Instruções do protocolo IdDE

Nas mensagens de criação de novas tarefas, os dados da tarefa são adicionadas à *tag* <arg0> da instrução do protocolo IdDE. As informações dessa tarefa também são definidas utilizando XML, como pode ser observado na Figura 59. As *tags* que definem a tarefa são: *name*, *project*, *date*, *priority* e *description*.

```
<IdDE>
  <code>14</code>
  <arg0>
    <task>
      <name>Test new function</name>
      <project>Test Project</project>
      <dateDue>2011-02-28 00:00:00.0 BRT</dateDue>
      <priority>Highest</priority>
      <description>
        You need to test this new function, which was  added.
      </description>
    </task>
  </arg0>
</IdDE>
```

Figura 59: Instrução IdDE com XML da tarefa

As mensagens de resposta com as tarefas do usuário não são enviadas utilizando uma instrução do protocolo IdDE. Neste caso, as tarefas existentes são enviadas através de mensagens XMPP e apresentadas na interface de *chat*.

Tabela 9: Códigos do protocolo IdDE utilizados na negociação de tarefas

Objetivo	*Tag*	Conteúdo da *Tag*
Mensagem utilizada para enviar uma nova tarefa para o usuário remoto.	*code*	14
	arg0	XML com as informações da nova tarefa.
Mensagem de resposta indicando que o usuário remoto aceitou a tarefa.	*code*	12
Mensagem de resposta indicando que o usuário não aceitou a tarefa.	*code*	13
Mensagem utilizada para solicitar a relação de tarefas do usuário remoto.	*code*	11

4.10.3 Arquitetura da ferramenta de gerenciamento de tarefas

A arquitetura da ferramenta de tarefas está representada na Figura 60. O pacote relacionado com a inclusão de uma nova tarefa é *task*. Nele, é definida a tela de digitação da tarefa (classe *task::view::TaskFrame*) e a classe que define as informações de uma tarefa (classe *task::model::Task*). A tela de digitação de tarefa é instanciada quando o usuário acessa o menu de contexto na lista de contatos (*common::view::ContactListPanel*).

A classe *util::Database* possui o método *insertTask*, responsável pela inclusão da tarefa na respectiva tabela no banco de dados e o método *getTasks*, utilizado para obter as tarefas do usuário do banco.

O processo de envio e recepção das mensagens de tarefas segue os fluxos descritos nas ferramentas anteriores, utilizando as mesmas classes e métodos. A única diferença está relacionada aos códigos das instruções do protocolo IdDE.

A classe *chat::view::ChatFrame* é utilizada pelo método *InstructionEvaluator* (classe *common::transport*) para a apresentação das tarefas, quando for recebida uma mensagem solicitando as tarefas do usuário.

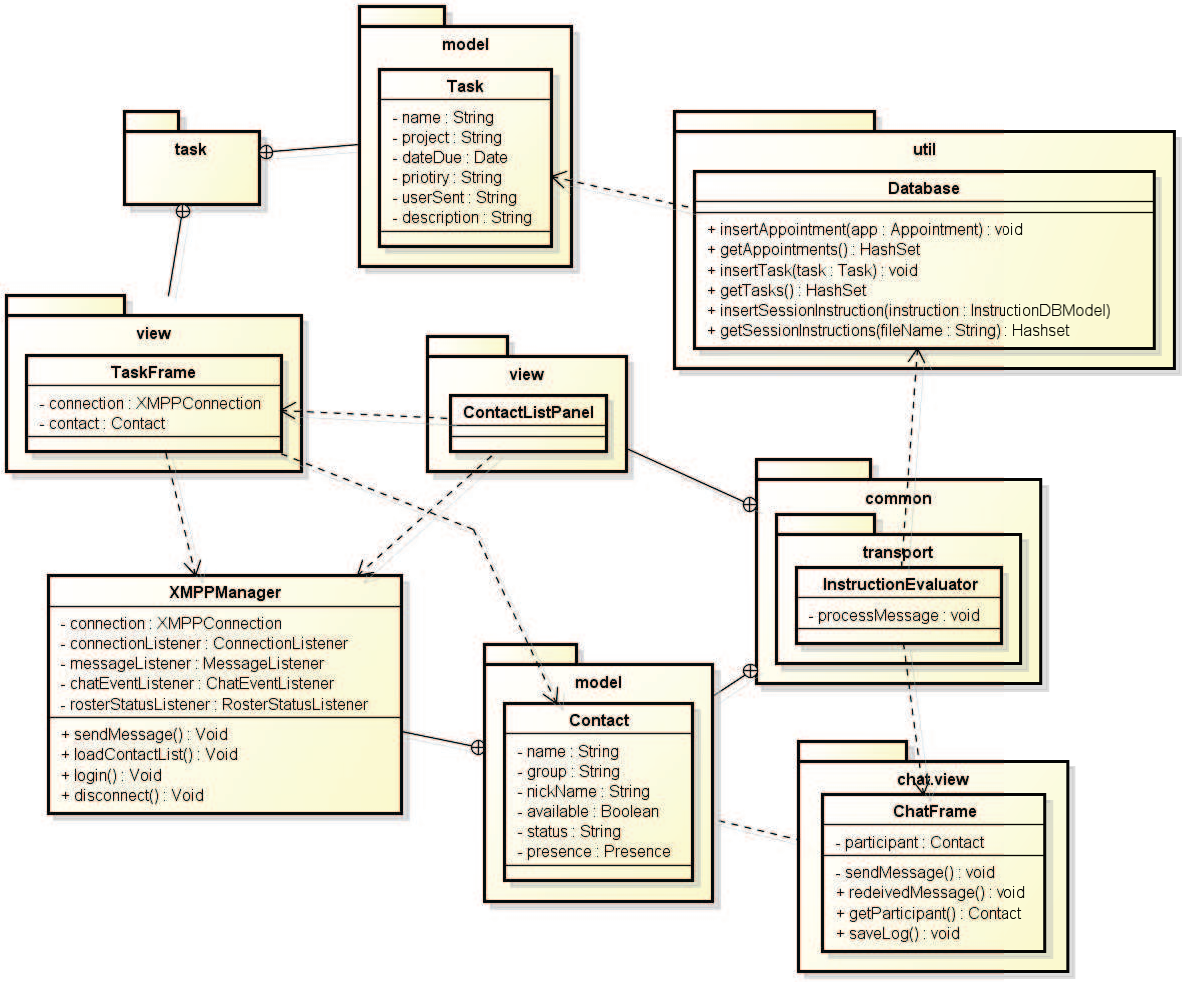


Figura 60: Arquitetura ferramenta de tarefas

4.11 Agenda

A ferramenta de agenda e complementar ao controle de tarefas, e tem por objetivo o controle dos compromissos do usuário. Através deste módulo será possível efetuar o controle da agenda de trabalhos, marcar reuniões, entre outros. Além disso, a ferramenta possibilitará que outros usuários tenham acesso à agenda do usuário, para verificar sua disponibilidade para uma eventual reunião. Assim, um usuário poderá adicionar novos compromissos à agenda de usuários remotos, incluindo, inclusive, informações sobre a pauta das reuniões, possibilitando que os participantes estejam preparados, com todas as informações necessárias, antes do encontro.

4.11.1 Interface com usuário e fluxo de mensagens

O acesso às funcionalidades da agenda ocorre como nas demais ferramentas: através do menu de contexto nos contatos. As opções do menu, relativas à agenda, estão descritas a seguir:

- *Add new appointment for this User*: esta opção é utilizada para acessar a tela para adicionar um novo compromisso à agenda do usuário;
- *View user's appointments*: acessando essa opção é possível ver os compromissos agendados para o usuário. Os compromissos do usuário são exibidos na tela de *chat*;
- *Open agenda manager*: através desta opção o usuário pode acessar os seus compromissos, podendo incluir excluir ou alterar essas informações.

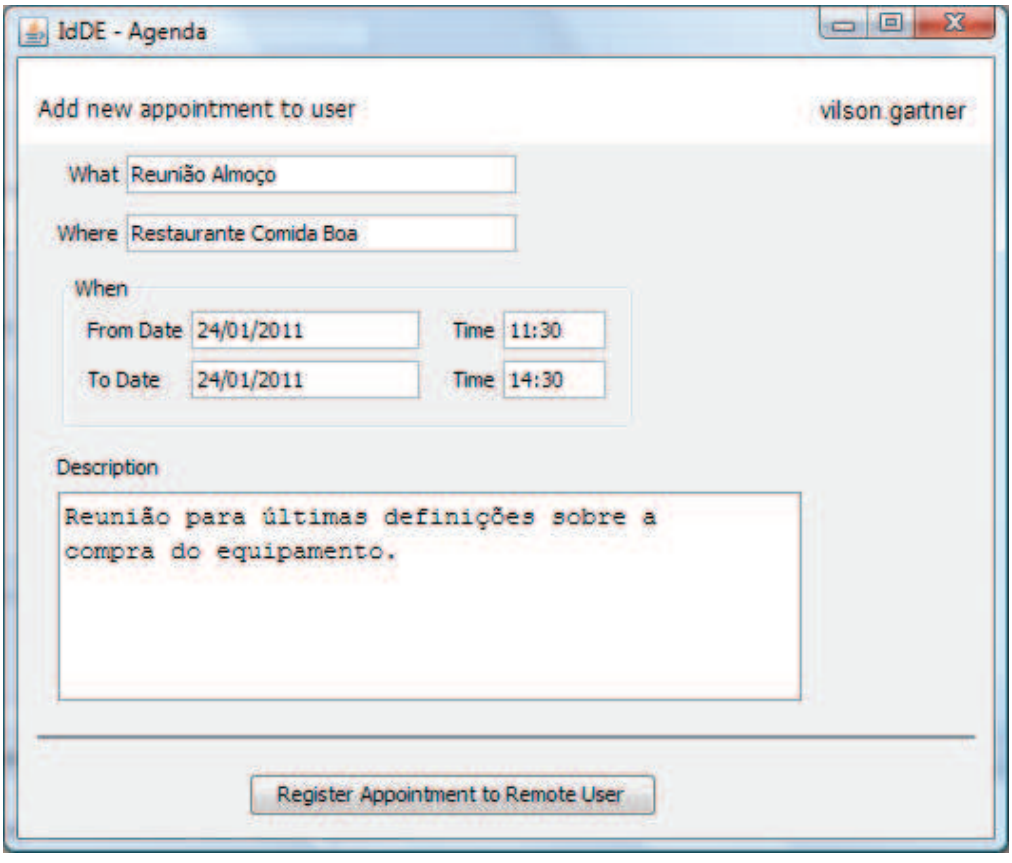


Figura 61: Tela de inclusão de compromisso na agenda

A Figura 61 apresenta a tela de inclusão de compromissos na agenda de um usuário remoto. Os campos dessa interface estão descritos a seguir:

- *What*: descrição sucinta sobre o compromisso;

- *Where*: neste campo deve ser informado o local do compromisso;
- *When*: nestes campos devem ser informadas as datas de início e fim, bem como os horários de início e fim do compromisso;
- *Description*: descrição detalhada do compromisso.

O processo de criação de um novo compromisso para um usuário remoto segue o mesmo fluxo das atividades da ferramenta de tarefas, apresentado na Figura 57. A única diferença está relacionada ao fato de armazenar as informações em outra tabela, e na criação da instrução de resposta, a qual utiliza outros códigos do protocolo IdDE.

4.11.2 Protocolo IdDE

Da mesma forma como acontece na ferramenta de tarefas, o IdDE utiliza uma mensagem XML na tag *<arg0>* para enviar as informações de um compromisso. A Figura 62 apresenta um exemplo da formatação do XML com esses dados.

```
<agenda>
 <what>Reunião Almoço</what>
 <where>Restaurante Comida Boa</where>
 <date_from>2011-01-24 00:00:00.0 BRT</date_from>
 <time_from>11:30</time_from>
 <date_to>2011-01-24 00:00:00.0 BRT</date_to>
 <time_to>14:30</time_to>
 <description>Reunião para últimas definições sobre a
   compra do equipamento.</description>
</agenda>
```

Figura 62: XML com dados do compromisso

A Tabela 10 detalha as mensagens do protocolo IdDE utilizadas na comunicações da ferramenta de agenda de compromissos.

Tabela 10: Códigos do protocolo IdDE utilizados na ferramenta de agenda

Objetivo	*Tag*	Conteúdo da *Tag*
Mensagem para criar um novo compromisso para o usuário remoto.	*code*	18
	arg0	XML com as informações da nova tarefa.

Código utilizado na mensagem de resposta indicando que o usuário remoto aceitou o compromisso.	*code*	19
Código utilizado para indicar que o usuário não aceitou o compromisso.	*code*	20
Código das mensagens que solicitam a relação de compromissos do usuário remoto.	*code*	21

4.11.3 Arquitetura da ferramenta de agenda

A Figura 63 mostra a organização da ferramenta. As classes relacionadas com as funcionalidades da agenda ficam agrupadas no pacote *agenda*.

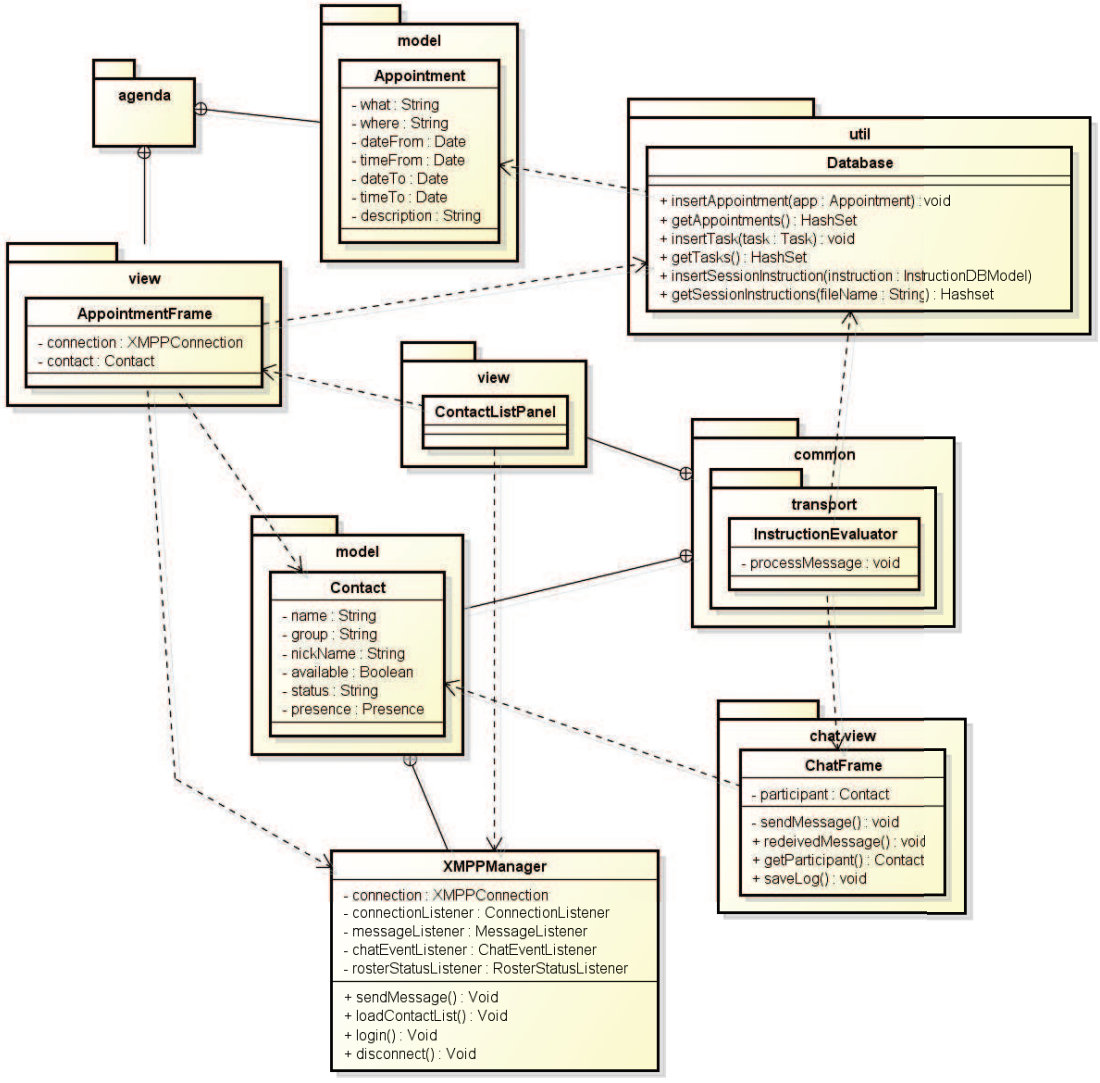


Figura 63: Arquitetura da ferramenta de agenda

A classe *agenda::view::AppointmentFrame* é a responsável pela apresentação da tela de inclusão de novo compromisso. Já a classe *agenda::model::Appoinment* possui as definições do objeto compromisso.

A gravação dos compromissos no banco de dados é realizada pelo método *insertAppointment* (classe *util::Database*) e a leitura dessas informações da base é realizada pelo método *getAppointments*.

As classes dos demais pacotes possuem as mesmas funcionalidades já descritas na ferramenta de controle de tarefas.

4.12 Transferências de arquivos

Durante o processo de desenvolvimento de *software*, é corriqueira a tarefa de compartilhamento de arquivos como artefatos de desenvolvimento, documentos, relatórios, entre outros. Normalmente esses arquivos são enviados através de ferramentas assíncronas como *e-mail* e acredita-se que, com a integração dessa funcionalidade no IdDE, possa haver uma contribuição para a diminuição da dispersão do usuário, analisada por Cheng *et al*. (2003).

4.12.1 Processo de envio

O fluxo de atividades executadas no envio de um arquivo está representado na Figura 64. Para enviar um arquivo para um usuário, após selecionar a respectiva opção no menu de contexto, será necessário selecionar o arquivo. Essa seleção é feita através de um diálogo padrão do Windows para seleção arquivos.

Após indicar o arquivo a ser enviado, o ambiente criará uma mensagem contendo a instrução do protocolo IdDE para o ambiente remoto, solicitando que o usuário confirme se deseja, ou não, receber o arquivo. Diante da resposta do usuário, o ambiente remoto criará a resposta correspondente e enviará ao ambiente de onde partiu a solicitação.

Se a resposta recebida for positiva (o usuário deseja receber o arquivo), o ambiente enviará o arquivo para o ambiente de destino, utilizando o protocolo XMPP. Quando receber o arquivo, além de salvá-lo na pasta "*home*" do usuário, o ambiente remoto também exibirá uma mensagem alertando o usuário. Ao mesmo em tempo que o ambiente local envia o arquivo, ele cria uma mensagem para avisar o usuário sobre o processo.

Quando a transferência do arquivo entre os dois ambientes é concluída, os usuários são avisados e o processo é finalizado.

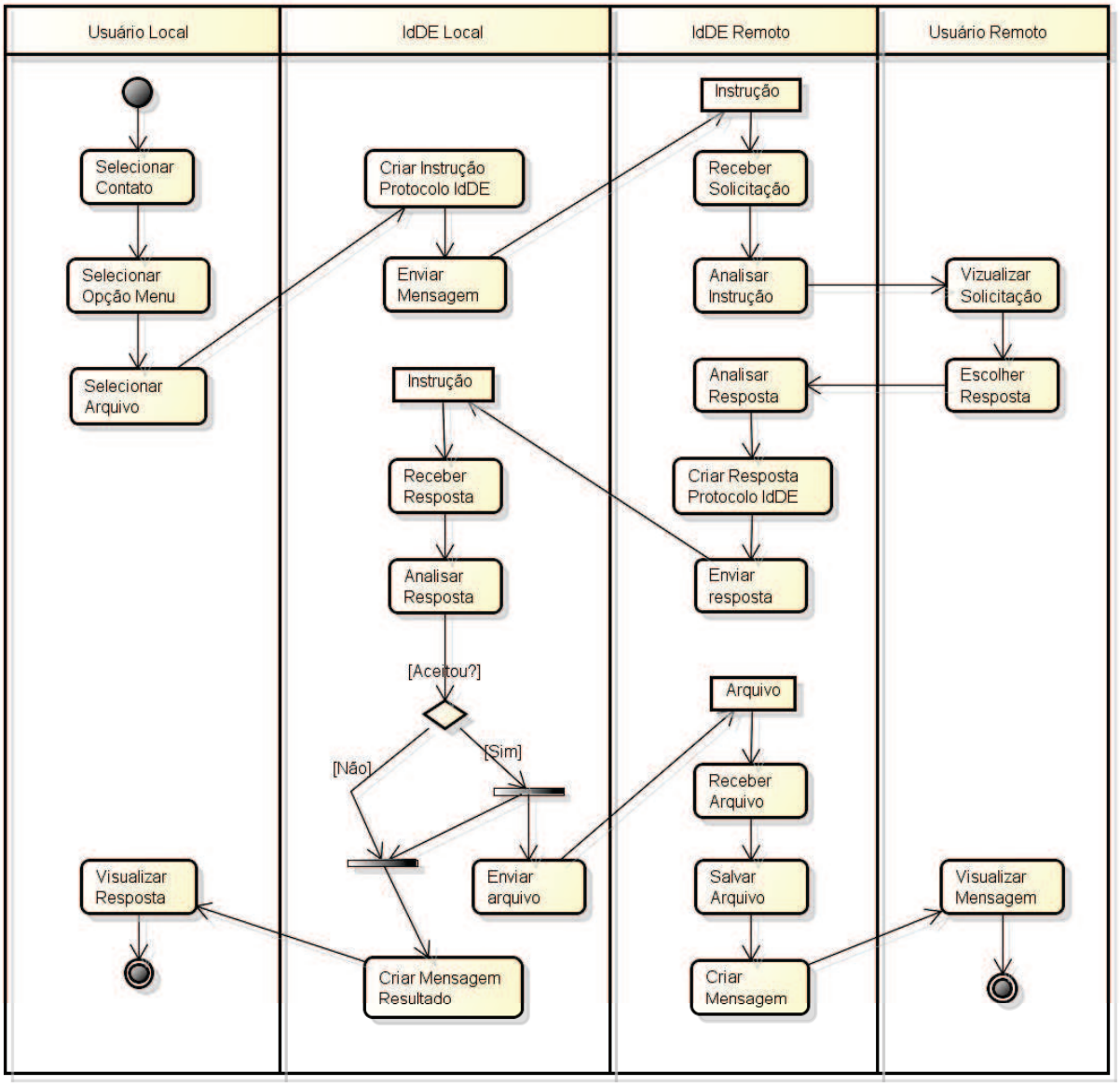


Figura 64: Fluxo de atividades para o envio de arquivo

4.12.2 Instruções do protocolo IdDE

As mensagens utilizadas pela ferramenta de envio de arquivos estão descritas na Tabela 11.

Tabela 11: Mensagens do protocolo IdDE utilizados no envio de arquivos

Objetivo	*Tag*	Conteúdo da *Tag*
Mensagem utilizada quando o usuário deseja enviar	*code*	22

Objetivo	*Tag*	Conteúdo da *Tag*
um arquivo.	*arg0*	Nome do arquivo que o usuário deseja enviar.
Utilizado em mensagens de resposta que indicam que o usuário aceitou receber o arquivo.	*code*	23
	arg0	Nome do arquivo
Valores utilizados nas mensagens de resposta que indicam que o usuário não aceitou receber o arquivo.	*code*	24
	arg0	Nome do arquivo

4.12.3 Arquitetura da ferramenta de transferência de arquivos

A arquitetura da ferramenta está representada no diagrama da Figura 65. As funcionalidades de envio e recebimento de arquivos estão implementados na classe *fileManager::controller::fileTransfer*. Essa classe possui entre seus métodos: *sendFile*, utilizado para o envio de arquivos; *receiveFile*, utilizado para o recebimento de arquivos e *monitorTransfer*, responsável por monitorar o progresso e conclusão da transferência do arquivo.

Para o envio do arquivo, é utilizada a conexão ativa e armazenada na classe *common::model::XMPPManager* (atributo *connection*). O método *sendMessage* dessa classe somente é utilizado para o envio das mensagens de negociação para início da transferência do arquivo.

O transporte dos arquivos é realizado através das classes do pacote *smackx*, uma extensão da biblioteca *smack* (utilizada para o envio das mensagens). Entre as principais classes desse pacote estão: *FileTransferManager*, responsável pelo gerenciamento da transferência dos arquivos; *OutgoingFileTransfer*, utilizada para o envio de arquivos; *FileTransferListener*, utilizada para aguardar o início do recebimento; *IncomingFileTransfer*, utilizada para o recebimento de arquivos.

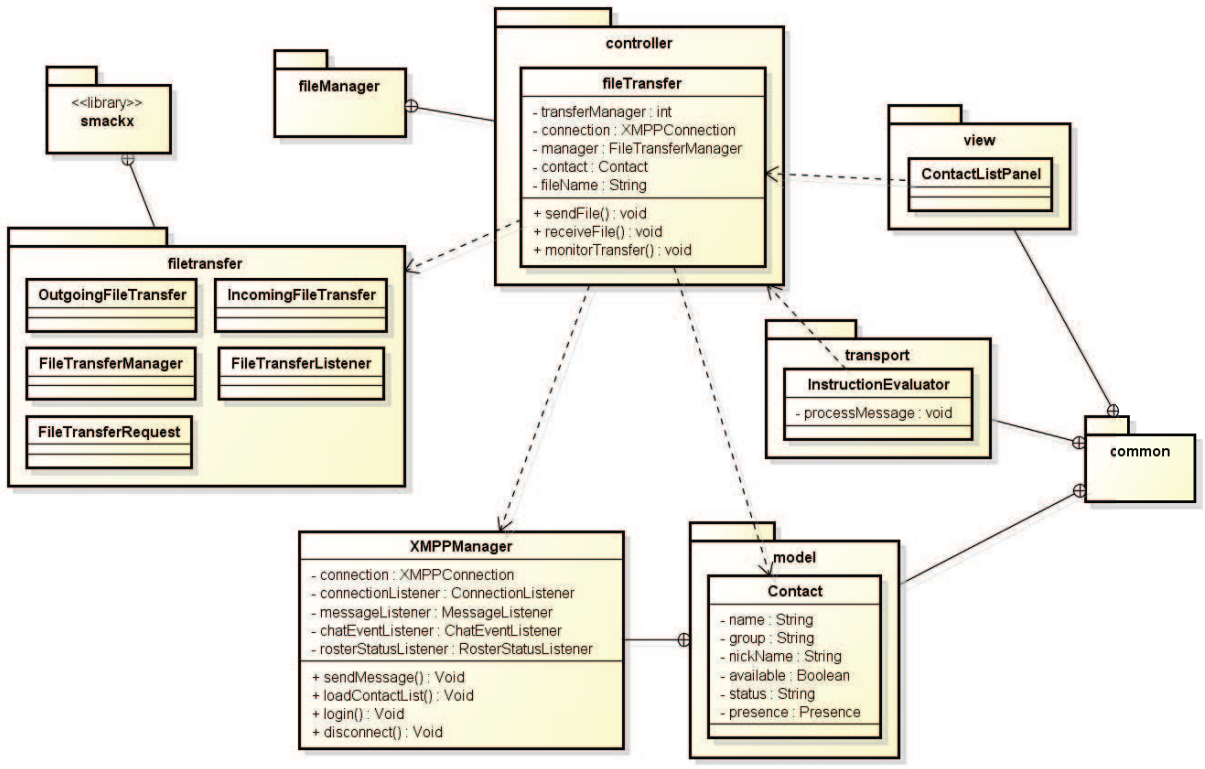


Figura 65: Diagrama de classes da ferramenta de transferência de arquivos

4.13 Análise Comparativa com Trabalhos Relacionados

Nesta seção será realizada uma análise comparativa entre o trabalho desenvolvido nesta dissertação e os trabalhos apresentados no Capítulo 3. Para a realização da comparação, foram estabelecidos 18 critérios com base nos trabalhos apresentados na revisão bibliográfica, além de considerar questões tecnológicas.

A Tabela 12 abaixo apresenta os critérios utilizados na comparação e uma descrição detalhada de cada um deles.

Tabela 12: Critérios de comparação entre os trabalhos

Critério	Descrição
C1	A ferramenta permite a comunicação utilizando mensagens instantâneas ou outra forma de comunicação via texto.
C2	Implementa comunicação via áudio;
C3	A comunicação via áudio suporta a comunicação com dispositivos da PSTN;
C4	Possibilita a comunicação utilizando vídeo;

Critério	Descrição
C5	A ferramenta implementa a funcionalidade de edição colaborativa de códigos fonte;
C6	A edição colaborativa suporta mais de 2 usuários simultaneamente na mesma sessão;
C7	A ferramenta de edição marca e identifica os autores das alterações no código fonte;
C8	A ferramenta de edição marca erros de sintaxe e efetua *code-highlight* da linguagem de programação no código fonte;
C9	A ferramenta de edição possibilita compilar, depurar e executar o programa;
C10	O protocolo de comunicação é aberto e/ou documentado;
C11	O protocolo de comunicação utiliza XML;
C12	A ferramenta implementa controle de tarefas;
C13	Permite o controle de compromissos/agenda;
C14	As mensagens são enviadas diretamente para o cliente, sem a necessidade da existência de um servidor intermediário para análise e distribuição das mensagens do protocolo;
C15	A ferramenta implementa a funcionalidade de quadro branco, permitindo que os usuários compartilhem diagramas e/ou desenhos;
C16	Suporta o envio/troca de arquivos entre os participantes;
C17	A ferramenta pode ser adaptada para ser utilizada em conjunto com outra IDE, a exemplo de Eclipse, Netbeans, JEdit;
C18	Sistemas Operacionais nos quais a ferramenta pode ser utilizada. Valores válidos são: Windows, Linux ou Ambos. Considera-se que esse critério está atendido se a ferramenta puder ser utilizada em ambos os Sistemas Operacionais.

Com base nos critérios estabelecidos, foi efetuado um comparativo entre os trabalhos, o qual é apresentado na Tabela 13 abaixo. Quadros com a informação “Sim” indicam que o critério é atendido pela ferramenta, enquanto que quadros com o valor “Não” assinalam que a ferramenta não implementa a funcionalidade. A informação “Parcial” indica que a ferramenta atende parcialmente o critério.

Tabela 13: Comparação com trabalhos relacionados

	VIMEE	RemotePP	CVW	CollabEd	IdDE
C1	Sim	Sim	Sim	Sim	Sim
C2	Não	Sim	Sim	Não	Sim
C3	Não	Não	Não	Não	Sim
C4	Não	Sim	Sim	Não	Não
C5	Não	Sim	Não	Sim	Sim
C6	Não	Não	Não	Sim	Sim
C7	Não	Não	Não	Não	Sim
C8	Não	Não	Não	Parcial	Sim
C9	Não	Parcial	Não	Não	Sim
C10	Não	Não	Sim	Sim	Sim
C11	Não	Não	Não	Não	Sim
C12	Não	Não	Não	Não	Sim
C13	Sim	Não	Não	Não	Sim
C14	Não	Sim	Não	Não	Sim
C15	Não	Sim	Sim	Não	Não
C16	Sim	Sim	Sim	Não	Sim
C17	Não	Não	Não	Sim	Sim
C18	Ambos	Windows	Ambos	Ambos	Ambos

Analisando os resultados da avaliação comparativa entre os trabalhos, percebe-se que, em termos quantitativos, o IdDE é a ferramenta que implementa o maior número de funcionalidades estabelecidos nos critérios de avaliação, atendendo um total de 16 itens. Enquanto isso, a ferramenta RemotePP atende 7,5 critérios, a CVW 7 critérios, a CollabEd 6,5 e a VIMEE 4 critérios.

5 ESTUDOS DE CASO

Neste capítulo serão apresentados os estudos de caso realizados com o objetivo de validar a aplicação desenvolvida. No total, foram realizados quatro estudos de casos em grupos heterogêneos, abrangendo profissionais que trabalham no desenvolvimento de *software* e professores e acadêmicos de cursos de graduação da área de tecnologia da informação. Dessa forma, é possível ter uma visão ampla sobre a aceitação do aplicativo por pessoas de diferentes perfis profissionais, bem como compreender as exigências de cada grupo em relação às ferramentas propostas no IdDE.

Em função do estágio de desenvolvimento do ambiente por ocasião da realização dos testes, aliado ao recesso de final de ano e período de férias, os testes foram concentrados nas ferramentas de edição colaborativa, comunicação através de *chat* e de áudio.

Para garantir que os resultados obtidos retratassem a opinião baseada na experiência de usabilidade dos participantes, os testes não seguiram um roteiro único, previamente combinado, e que definia as etapas e ações a serem executadas de forma a alcançar um resultado esperado. Ao contrário, cada grupo foi orientado a utilizar as ferramentas do ambiente para se comunicar, compartilhar e editar códigos com os demais participantes.

Após a realização dos testes, os usuários foram convidados a avaliar a ferramenta. A avaliação ocorreu através da realização de um questionário, cujas perguntas são apresentadas na Tabela 14 abaixo. As questões foram criadas com o objetivo de avaliar desde aspectos importantes relacionados ao DDS, até características como praticidade e usabilidade do *software*.

Tabela 14: Questionário para avaliação da ferramenta

Nº	Descrição da pergunta	Respostas possíveis	Permite Comentários
1	O ambiente auxiliou no processo de comunicação via chat?	- Sim - Não	Não
2	O ambiente auxiliou no processo de edição compartilhada e colaborativa de *software*?	- Sim - Não	Sim
3	A ferramenta de áudio mostrou-se útil e funcional?	- Sim - Não - Não testado	Não

Nº	Descrição da pergunta	Respostas possíveis	Permite Comentários
4	Você acredita que a comunicação via áudio seja importante, principalmente quando os programadores estão separados geograficamente?	- Sim - Não	Obrigatório
5	Em sua opinião, as ferramentas disponibilizadas pelo ambiente contribuem para o:		
5.1	Processo de ensino/aprendizagem presencial?	- Sim - Não	Sim
5.2	Processo de ensino/aprendizagem à distância?	- Sim - Não	Sim
5.3	Desenvolvimento de *software* por equipes que estão num mesmo ambiente?	- Sim - Não	Sim
5.4	Desenvolvimento distribuído de *software*, independente da localização física das pessoas, mesmo que em países e idiomas diferentes?	- Sim - Não	Obrigatório
6	Você acredita que, pelo fato de o ambiente ter sido desenvolvido como um módulo do Netbeans, tenha contribuído no sentido de enriquecer a qualidade e praticidade funcional do ambiente?	- Sim - Não	Sim
7	Em relação à Interface Homem-Computador (IHC), a disposição das janelas dentro do ambiente do Netbeans está adequada?	- Sim - Não	Sim
8	Em relação à performance do Netbeans, você considera que tenha continuado igual?	- Sim, continuou igual - Não, ficou mais lento	Não
9	No seu ponto de vista, qual o principal benefício proporcionado por essa ferramenta?	Descritiva	Obrigatório
10	O que poderia ser melhorado no ambiente, em sua opinião? Que outras características poderiam ser incorporadas para melhorar a experiência do	Descritiva	Sim

Nº	Descrição da pergunta	Respostas possíveis	Permite Comentários
	usuário?		
11	Você utilizaria esta ferramenta no seu dia-a-dia, no processo de desenvolvimento de *software*?	- Sim - Não	Sim
12	Em sua opinião, o ambiente contribui para um aumento da produtividade?	- Sim - Não	Sim
13	Comentários sobre outros aspectos relevantes, observados durante a realização dos testes.	Descritiva	Sim

Além das respostas objetivas, em algumas questões foi solicitado que o usuário comentasse a sua escolha, para que fosse possível compreender melhor a sua resposta. A coluna "Permite Comentários" da tabela indica as perguntas que permitiram a adição de observações. Como é possível perceber, em algumas questões a justificativa era de caráter obrigatório.

Cada cenário a seguir retrata, detalhadamente, as condições e metodologias aplicadas nos testes, bem como os resultados obtidos.

5.1 Cenário 1 – Estudantes Universitários

O primeiro estudo de caso foi realizado com alunos de uma turma de graduação do Centro Universitário UNIVATES, na cidade de Lajeado/RS. A turma era composta por alunos dos cursos de Sistemas de Informação, Engenharia da Computação e Análise de Sistemas, todos cursando a disciplina de Programação Orientada a Objetos. Os testes foram realizados num total de duas aulas, no mês de Dezembro de 2010.

Alguns alunos participantes já atuavam na área de desenvolvimento de sistemas, enquanto que outros somente estudavam ou eram profissionais de outras áreas. As ferramentas testadas foram edição colaborativa e comunicação via *chat*.

5.1.1 Descrição do cenário e metodologia

Os testes deste cenário foram realizados num dos laboratórios da instituição, com um total de 23 participantes. Antes do início dos testes, foi feita uma apresentação da ferramenta, descrevendo suas características e seus objetivos. Em seguida, os próprios alunos executaram

a instalação do *plugin* no Netbeans, que já é utilizado normalmente pelos alunos nas disciplinas de programação. O sistema operacional utilizado para esses testes foi o Windows XP. Para a realização dos testes, foi utilizado um servidor XMPP interno (OpenFire), já que a estrutura da rede da instituição não permitia a utilização de servidores externos. As ferramentas testadas neste cenário foram edição simultânea e comunicação através de mensagens instantâneas. A comunicação via áudio não foi possível em virtude do bloqueio das portas na rede da instituição.

Para a realização dos testes e análise do comportamento da ferramenta, os alunos foram instruídos a compartilharem arquivos com o professor, o qual acompanhava, a partir de sua mesa, as modificações feitas nos documentos. Além disso, o professor também colaborava com os alunos através da escrita de código diretamente no arquivo compartilhado.

A comunicação entre os alunos e professor, sempre que possível, ocorreu através da ferramenta de chat do ambiente. Todavia, foi possível perceber que, apesar da praticidade, os alunos com menos conhecimento de programação se sentiam mais confortáveis com a presença do professor ao seu lado.

Durante os testes realizados neste cenário, a ferramenta não apresentou nenhum tipo de problema. À medida que os alunos alteravam os códigos dos programas, eram feitas comparações com a versão existente na máquina do professor, para verificar a ocorrência de alguma discrepância entre as duas versões.

Apesar do nível de dispersão neste estudo ser do tipo “Mesma localização física”, segundo a classificação de Audy e Prikladnicki (2008), os testes permitiram observar e avaliar o funcionamento da ferramenta, bem como a forma de interação dos usuários com a mesma, e as dificuldades encontradas. Caso os usuários estivessem dispersos em outras localizações geográficas, a análise de eventuais problemas na ferramenta estaria inviabilizada.

Aliado a isso, segundo Audy e Prikladnicki (2008), o DDS ainda é pouco abordado pelas Universidades do Brasil. Para que as empresas consigam desenvolver *softwares* utilizando esse novo modelo, e para amenizar os problemas ocasionados pela mudança de paradigma, as universidades deveriam trazer essa preocupação para dentro das salas de aula, incentivando projetos entre instituições. Alunos com esse conhecimento e experiência, certamente seriam beneficiados e teriam vantagem, pois atualmente, para se manter competitivo, o profissional de engenharia de *software* precisa de uma formação que considere os desafios do DDS.

Nesse sentido, este cenário de estudo de caso procurou avaliar, também, o comportamento dos alunos, observando e compreendendo as necessidades e limitações desse perfil de usuário.

5.1.2 Análise de Resultados

Os resultados das avaliações feitas junto aos participantes deste estudo de caso estão apresentados no gráfico da Figura 66. Apesar da participação de 23 pessoas, somente 14 responderam ao questionário proposto para a avaliação da ferramenta.

Na avaliação dos respondentes, a ferramenta de *chat* se mostrou útil para todos eles (Pergunta 1). Alguns consideraram importante a possibilidade de poder efetuar a tradução das mensagens. Igualmente, para todos os respondentes a ferramenta auxiliou no processo de edição compartilhada e colaborativa (Pergunta 2).

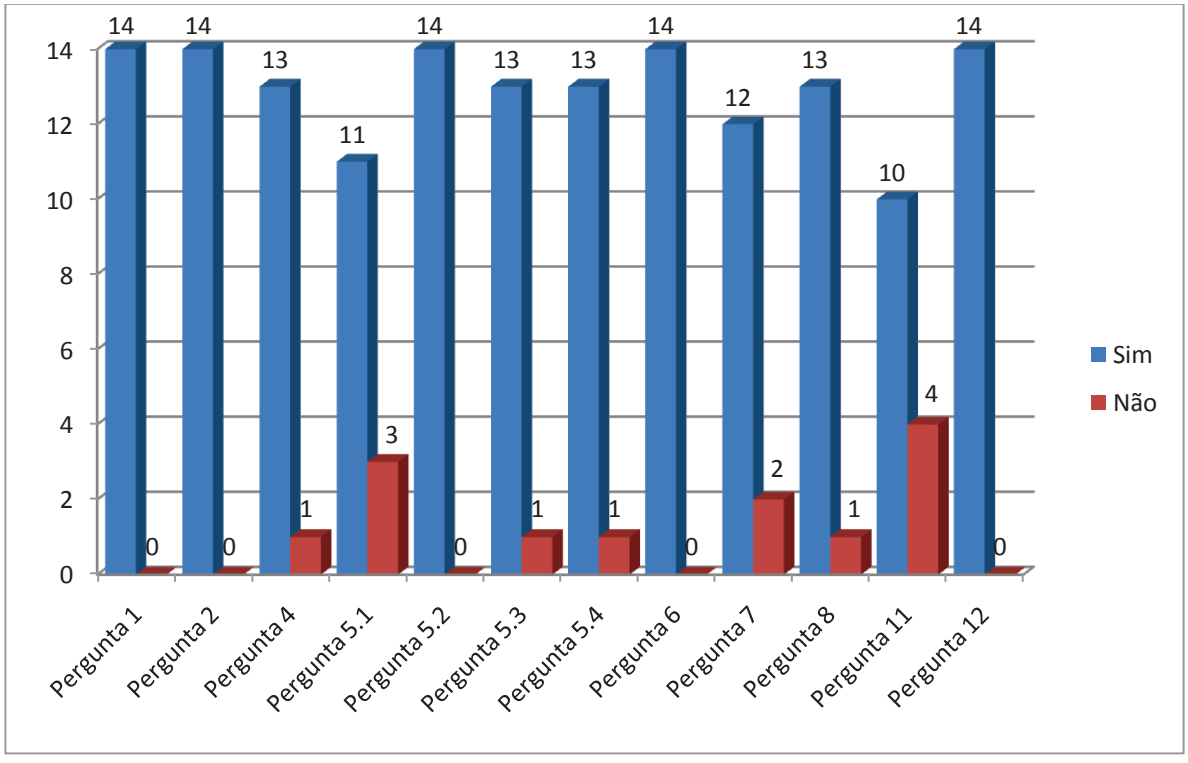


Figura 66: Resultado das avaliações do Estudo de Caso, Cenário 1.

Em relação à comunicação via áudio (Pergunta 3), apesar de não ter ocorrido nenhum teste neste cenário, 93% dos respondentes (13 pessoas) consideram importante essa forma de comunicação quando os programadores estão separados geograficamente. Alguns consideram que o áudio facilita a compreensão, enquanto outros alegam que pode ser mais rápido e

prático conversar com o usuário remoto. Outros, ainda, alegam que o texto digitado via chat pode não ficar claro o suficiente, gerando dúvidas, principalmente quando o texto for muito extenso. Por fim, alguns participantes alegaram que o tempo gasto por um programador para a explicação de um código seria muito menor quando feito com o uso de áudio.

Em relação ao processo de ensino/aprendizagem presencial (Pergunta 5.1), 79% dos respondentes (11 pessoas) concordam que a ferramenta contribui nesse sentido. Entre os que concordam, as justificativas mencionam o fato de que o professor não precisa se deslocar até o computador do aluno, outras alegam que as ferramentas do ambiente fornecem maior interatividade entre os usuários e o consideram como um facilitador. O posicionamento de um dos respondentes que não concordam, é de que o *plugin* não seria necessário pelo fato da aula ser presencial.

Por outro lado, no que se refere ao processo de ensino/aprendizagem à distância (Pergunta 5.2), 100% dos respondentes acreditam que o IdDE contribua neste contexto. Entre as justificativas, está a transposição de obstáculos como a distância, graças ao uso das ferramentas disponibilizadas pelo ambiente. Nesse sentido, utilizando-se de *chat*, edição compartilhada e áudio, o professor conseguiria auxiliar o aluno.

Para 93% dos respondentes (13 pessoas), o IdDE contribui para o desenvolvimento de *software* por equipes que estão num mesmo ambiente (Pergunta 5.3). Entre as justificativas, está o fato de os profissionais não precisarem se deslocar até outras estações para a troca de idéias ou para discutir a respeito de alguma implementação de código.

Em relação à contribuição para o desenvolvimento distribuído (Pergunta 5.4), 93% (13 pessoas) consideram que o IdDE possui papel importante. O único respondente que assinalou que o ambiente não contribuiria para este contexto, alegou que não havia entendido "qual era o ponto", se referindo, possivelmente, a não compreensão da pergunta ou ainda em relação ao próprio tema DDS. Para os demais, a ferramenta contribui, definitivamente para este modelo de desenvolvimento de *software*. Entre as alegações, são feitas referências aos benefícios proporcionados pela edição colaborativa, pela ferramenta de áudio e *chat*. Outros salientam a existência do tradutor na comunicação em tempo real, importante quando os participantes falam diferentes idiomas. Um ponto bastante citado é a possibilidade de usar a ferramenta para trocar idéias e colaborar na edição com pessoas de outros lugares, com outras experiências e formas de programar.

Todos os participantes que responderam ao questionário acreditam que a adoção do Netbeans, como base para a construção do IdDE (Pergunta 6), tenha contribuído para tornar o

ambiente prático e funcional. Entre as alegações, alguns ressaltam o fato de ser um ótimo IDE para o desenvolvimento de *software*. Outros, afirmam a importância de ser desenvolvido na linguagem Java.

Em relação à interface humano-computador, 86% das pessoas acreditam que a disposição das janelas dentro do ambiente do Netbeans está adequada (Pergunta 7). Os dois usuários que responderam negativamente, alegam que a janela principal do IdDE é muito grande. Nestes casos, acredita-se que os usuários desconhecem as funcionalidades do Netbeans que permite que as janelas do ambiente sejam minimizadas, diminuindo o espaço ocupado.

Em relação à performance do Netbeans (Pergunta 8), 93% (13 pessoas) consideraram que a instalação do módulo IdDE não trouxe prejuízos à velocidade do ambiente, enquanto que uma pessoa acredita ter ficado mais lento.

A questão discursiva de número 9 permitiu que os participantes indicassem, sob seu ponto de vista, os principais benefícios proporcionados pelo ambiente. Entre as respostas estão: maior interatividade, ganho de tempo no desenvolvimento, utilização do ambiente no ensino a distância e mesmo presencial, comodidade, compartilhamento de ideias, ajuda entre os usuários, possibilidade de detectar erros de programação à distância, facilitar o desenvolvimento em equipes que estão no mesmo ambiente, chat e, a contribuição mais citada: edição compartilhada.

Com o objetivo de melhorar o ambiente, através da incorporação de novas características, a questão 10 foi adicionada ao questionário para receber sugestões dos usuários. Entre as ideias apresentadas, está a possibilidade de compartilhar todo um projeto com usuários remotos, melhorias no aspecto visual da tela de chat e na lista de usuários, além da adição de suporte a videoconferência.

Entre os usuários que responderam o questionário, 71% (10 pessoas) utilizariam a ferramenta no seu dia-a-dia, enquanto que 29% (4) não utilizariam (Pergunta 11). Entre os que a adotariam, as justificativas foram: comunicação rápida, a não necessidade de deslocamento dentro da empresa, o compartilhamento e edição colaborativa, visualização em tempo real das alterações que estão sendo feitas por outros usuários, conjunto de ferramentas melhores do que aquelas que o usuário utiliza atualmente. No caso dos usuários que não adotariam a ferramenta, apenas um respondente justificou, ressaltando que não trabalha na área de desenvolvimento.

Em relação ao aumento de produtividade obtida pelo uso da ferramenta (Pergunta 12), 100% das respostas foram positivas. Entre as justificativas apresentadas, estão: atuação como um facilitador, o que pode gerar um ganho de tempo e, consequentemente, da produtividade; a praticidade obtida na comunicação com a equipe através da ferramenta de *chat* e áudio; o compartilhamento de informações tornando possível o auxílio a outros usuários; possibilidade de duas (ou mais) pessoas poderem trabalhar conjuntamente na resolução de um problema de código, cada qual alterando partes do código.

Na questão de comentários livres (Pergunta 13), alguns participantes manifestaram-se afirmando que a ideia do *plugin* é muito interessante, enquanto que outros utilizaram o campo para parabenizar pelo resultado final obtido com o projeto.

5.2 Cenário 2 – Professores Universitários

Este estudo de caso foi realizado com quatro professores de cursos de graduação da área de tecnologia da informação, do Centro Universitário UNIVATES, na cidade de Lajeado/RS. Entre os professores, estavam os coordenadores dos cursos de Sistemas de Informação e Engenharia de Computação, além do Diretor do Centro de Ciências Exatas e Tecnológicas. As ferramentas testadas neste cenário foram: edição colaborativa e comunicação via *chat*.

5.2.1 Descrição do cenário e metodologia

Durante o desenvolvimento do projeto, percebeu-se que o aplicativo poderia auxiliar tanto no ensino presencial quanto no ensino a distância. Nesse sentido, foram analisados trabalhos que relatam experiências no âmbito acadêmico e cujo estudo está voltado para uma realidade de desenvolvimento de *software* entre equipes separadas geograficamente. GOTEL *et al*. (2006) relatam em seu trabalho a experiência prática realizada por três instituições de ensino: uma situada nos Estados Unidos, outra na Índia e outra no Camboja. Trata-se de um trabalho importante, pois demonstra o contato dos alunos com experiências e demandas cada vez mais crescentes no mercado. Nesse trabalho é possível identificar diversos aspectos e necessidades relacionados ao desenvolvimento distribuído e, o principal deles, é a falta de uma ferramenta integrada e pensada para suportar o DDS. Na realização dos trabalhos, os estudantes utilizaram duas formas de comunicação assíncrona: *e-mail* e blog, e três formas de comunicação síncrona: *chat*, telefone e reuniões presenciais (quando possível).

Dessa forma, a realização de testes junto a um grupo de professores também teve por objetivo validar o potencial da ferramenta em ambientes de ensino presencial e à distância, além de colher ideias e sugestões para a melhoria do ambiente. Os professores envolvidos nos testes realizam atividades de EAD nas disciplinas que ministram. Os testes deste estudo de caso ocorreram no mês de Dezembro de 2010, no mesmo laboratório de informática utilizado anteriormente para os testes com os estudantes de graduação, descrito no Cenário 1.

Antes de efetuar os testes no laboratório de informática, houve um contato com os professores através de e-mail e *chat*, para que eles pudessem fazer a instalação o *plugin* em seus computadores pessoais. Neste momento, além da instalação, foram feitos pequenos testes de edição colaborativa e utilização do *chat*, o que já permitiu que os mesmos tivessem uma ideia sobre o funcionamento do aplicativo.

No laboratório, após breve apresentação do ambiente e suas funcionalidades, os professores instalaram o *plugin* no Netbeans e foram instruídos a se comunicarem utilizando o *chat* do IdDE e efetuassem a edição colaborativa de códigos fonte de programas.

Inicialmente a edição ocorreu aos pares, mas posteriormente todos os participantes passaram a editar o mesmo arquivo. Percebeu-se que não houve nenhuma perda de desempenho nesta circunstância e as modificações feitas por um professor eram enviadas aos demais quase que instantaneamente. Acredita-se que o fato de o servidor XMPP utilizado estar na rede própria instituição tenha contribuído para esse resultado.

Em contrapartida, um dos objetivos dos testes com este grupo também foi o de conhecer e avaliar, empiricamente, as limitações da ferramenta. Nesse sentido, procurou-se gerar problemas de consistência entre os conteúdos dos arquivos de cada usuário, o que foi alcançado neste cenário. O problema surgiu quando um professor executou a formatação automática do código (alinhamento de chaves, identação e organização do código), o que gerou um grande número de instruções para os demais usuários. Como os outros participantes continuavam a fazer modificações, antes que todas as mensagens de formatação chegassem ao destino, instruções de outros usuários eram executadas, ocasionando, assim, problemas na formatação e, por conseqüência, causando discrepância nos conteúdos dos arquivos.

5.2.2 Análise de Resultados

Além de avaliar a solução com base nas perguntas do questionário, este grupo teve especial importância para a discussão do trabalho sob a ótica acadêmica, avaliando a proposta, solução final obtida e sugerindo características e melhorias a serem incorporadas em trabalhos

futuros. Nesse sentido, foram apontadas melhorias em relação a aspectos ergonômicos e de ordem funcional. Aliado a isso, o grupo considerou que uma das incorporações a serem feitas, em trabalhos futuros, seria a implementação de um algoritmo que garanta a unicidade de código para todos os participantes de uma sessão de edição colaborativa. Todavia, o grupo reconheceu que o erro foi gerado com certa dificuldade e somente após várias tentativas.

Os resultados das avaliações feitas junto aos professores deste estudo de caso estão apresentados no gráfico da Figura 67.

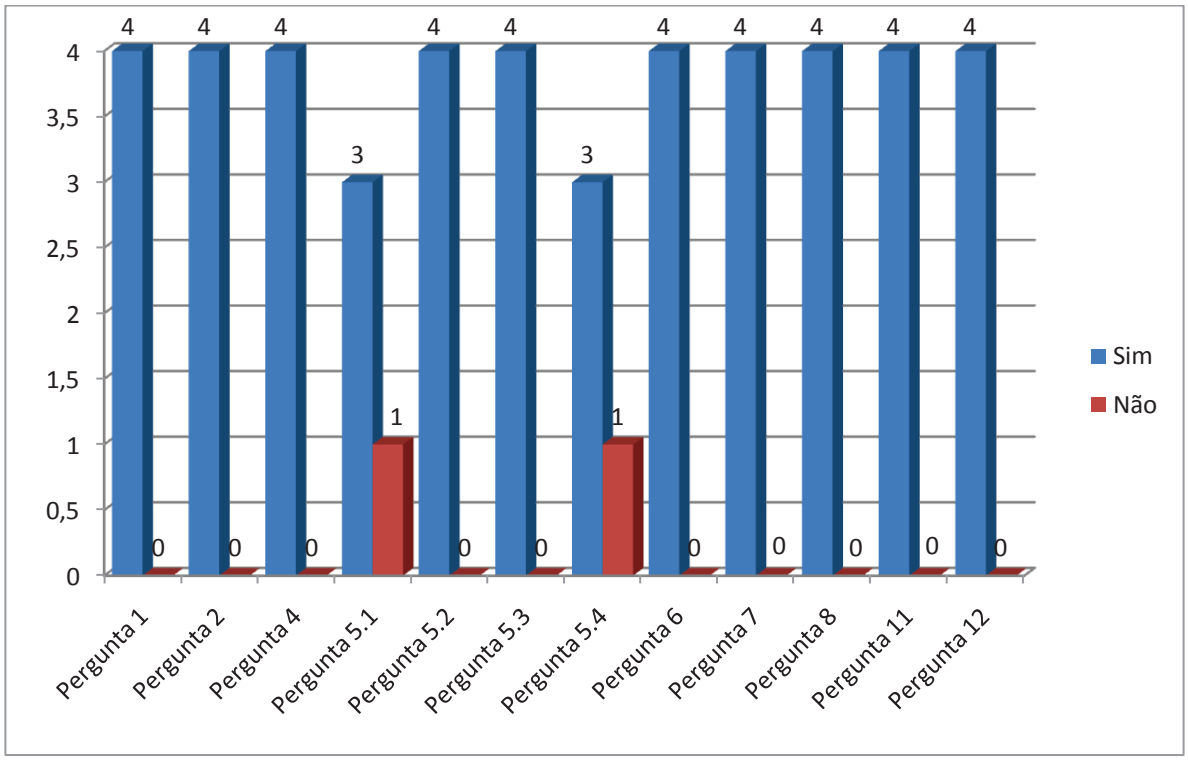


Figura 67: Resultado das avaliações do Estudo de Caso, Cenário 2.

Como é possível observar no gráfico, todos os participantes concordam que o ambiente auxilia no processo de comunicação através de mensagens instantâneas (Pergunta 1). Da mesma forma, todos os participantes concordaram que a ferramenta é útil na edição colaborativa (Pergunta 2), afirmando que o sistema funcionou de forma satisfatória para alterações em um mesmo trecho de código, arbitrando na intervenção de diversos colaboradores. Todavia, houve a ressalva que não ocorram edições nas mesmas linhas, para que não ocorra o problema de sincronia. Igualmente, consideraram que a ferramenta é extremamente útil em qualquer situação que demande desenvolvimento compartilhado: entre desenvolvedores de *software*, entre professor e aluno, na sala de aula ou auxílio à distância.

Apesar de não ter sido testada neste estudo de caso, todos os participantes acreditam que a comunicação via áudio seja importante num cenário onde os desenvolvedores estão separados geograficamente (Pergunta 4). Entre as justificativas, está a agilidade proporcionada para a resolução de dúvidas, principalmente nas situações onde a troca de mensagens de texto seria muito excessiva. Além disso, essa ferramenta permite que se possa editar o texto e conversar ao mesmo tempo, o que torna o processo mais prático e funcional e não seria possível no caso da digitação de mensagens de *chat*. Por outro lado, também houve considerações que afirmam que, como os desenvolvedores estão mais acostumados a utilizar mensagens de texto no seu dia-a-dia, talvez o áudio seja mais relevante somente nos casos em que a explicação via texto seja muito complicado.

Em relação ao processo de ensino aprendizagem presencial (Pergunta 5.1), apenas um dos participantes considera que esse modelo não seria beneficiado pelo uso do IdDE, afirmando que nesse caso a troca de ideias com os alunos ocorre através da demonstração do conteúdo através da utilização de projeção multimídia. Todavia, o professor acredita que o ambiente até possa auxiliar, mas bem menos que no processo de aprendizagem à distância. Em contrapartida, os demais participantes consideram que o *software* seja interessante também nesse modelo, pois o professor pode acompanhar de forma mais efetiva o que os alunos estão fazendo. Avaliam que a ferramenta permitiria que o docente auxiliasse mais de um aluno simultaneamente, sem a necessidade de se deslocar na sala de aula, multiplicando o número de alunos atendidos no período de aula. Da mesma forma, acreditam que, apesar da proximidade física, a ferramenta seja útil pelo fato de permitir a intervenção direta do docente na área de trabalho de um aluno, facilitando a inspeção e intervenção no código em desenvolvimento.

Os participantes foram unânimes ao afirmar que a ferramenta possui uma grande contribuição para a aprendizagem à distância (Pergunta 5.2). Entre as justificativas, está a contribuição para o auto-aprendizado, capacidade inerente aos profissionais da informática. Aliado a isso, consideram que um recurso como esse seja extremamente útil para apoiar o ensino de programação à distância, afirmando que outros meios tradicionais como *e-mail* e *chat* são bem menos adequados, além de serem mais trabalhosos tanto para o professor quanto para o aluno. Com o uso do ambiente, o professor poderia explicar o funcionamento de determinado código para o aluno de forma *online*.

O processo de desenvolvimento de *software* por equipes que estão num mesmo ambiente (Pergunta 5.3) também seria beneficiado, na opinião de todos os participantes, com

o uso da ferramenta. Nesse sentido, consideram que ela pode auxiliar na resolução de dúvidas de forma mais direta que o deslocamento físico. Da mesma forma, o programador líder consegue acompanhar o trabalho dos demais e auxiliar na resolução de problemas de codificação diretamente do seu computador. Além disso, dois ou mais programadores poderiam codificar o mesmo *software*, com a comodidade de cada um estar em frente a sua máquina.

No que se refere ao desenvolvimento feito por equipes distribuídas (Pergunta 5.4), um dos participantes considera que deveria haver uma alternativa intermediária. O mesmo acredita que a ferramenta possa ser utilizada nesse modelo de desenvolvimento de *software*, porém, ainda não atende plenamente a este requisito em função de não haver um mecanismo de resolução de conflitos. Por outro lado, os demais professores acreditam que a ferramenta atenda este requisito, justificando que ela auxilia a encurtar as distâncias. Consideram que a facilidade de uso e a instalação simplificada seja um ponto importante a considerar. Afirmam também que ela permite muito mais agilidade na inspeção de código e intervenção durante o processo de desenvolvimento. Por fim, acreditam que o fato de ser possível visualizar as modificações feitas pelos demais participantes colabora muito em situações de desenvolvimento compartilhado.

Na avaliação de todos participantes desses testes, o fato de o ambiente ter sido desenvolvido como um módulo do Netbeans (Pergunta 6) contribuiu positivamente para enriquecer a qualidade e praticidade funcional do ambiente. Argumentam que o ambiente é amplamente utilizado hoje em dia, e o *plugin* valoriza ainda mais o seu uso. Da mesma forma, acreditam que é importante que se procure adicionar ferramentas a ambientes já consolidados e que permitem boa produtividade, ao invés de desenvolver uma solução isolada. Por fim, houve a sugestão de desenvolver o *plugin* para outros ambientes de desenvolvimento.

Em relação à IHC (Pergunta 7), todos os professores acreditam que a disposição das janelas do IdDE dentro do Netbeans estão adequadas, avaliando que não houve nenhuma dificuldade e ressaltando, de forma positiva, a maneira como o ambiente adiciona as informações visuais quando ocorre alguma modificação de código.

Ao efetuar a avaliação da performance do Netbeans (Pergunta 8), todos consideraram que não houve mudanças após a instalação do *plugin*.

No item 9, que solicita que cada participante avalie, sob seu ponto de vista, o principal benefício proporcionado pela ferramenta, a principal vantagem apontada foi a edição colaborativa de código. Afirmam que a ferramenta apresenta o resultado imediato das

modificações, permitindo o acompanhamento e intervenção remota em tempo real e dentro do próprio ambiente de desenvolvimento. Além disso, consideram que o IdDE agiliza a atividade de desenvolvimento e o ensino e treinamento em programação, principalmente quando os participantes estão separados geograficamente, permitindo o auxílio e suporte a alunos e programadores menos experientes.

Ao sugerir melhorias para o ambiente, solicitado na pergunta 10, a principal manifestação foi em relação à implementação do tratamento de situações de conflito, que levam a diferentes versões de códigos. Todavia, ainda assim, esta foi considerada uma situação pontual que poderia ser facilmente controlada através de uma coordenação das modificações, pela utilização de um ou um "*token* virtual". Além disso, foi sugerida a criação de um histórico que mostre as alterações feitas num arquivo e que auxilie na resolução de problemas. Entretanto, é importante observar que essa funcionalidade já está implementada no IdDE, mas no momento destes testes ainda não estava concluída.

Todos os professores participantes consideraram a possibilidade de utilizar a ferramenta em seu dia-a-dia (Pergunta 11), pois ela seria bastante útil para prestar e receber auxílio durante o desenvolvimento. Da mesma forma, consideram que ela poderia ser vista como um controle de versionamento em tempo real.

Em relação à produtividade (Pergunta 12), todos os professores concordam que a ferramenta contribui para o seu aumento, principalmente em situações onde existem dúvidas de programação ou quando um usuário necessita de ajuda na codificação. Todavia, foi ressaltado que um aumento de produtividade também depende do modelo de desenvolvimento adotado, além da experiência da equipe.

Na avaliação final dos professores, o ambiente possui um grande potencial pedagógico, inclusive houve grande interesse em utilizá-lo nas disciplinas ministradas pelos participantes.

5.3 Cenário 3 – Empresa de Desenvolvimento de *Software*

Os testes deste cenário foram realizados junto ao setor de desenvolvimento da empresa Interact Solutions, de Lajeado/RS, especializada em *softwares* para a gestão da qualidade.

A empresa possui 49 colaboradores, dos quais 15 fazem parte do setor de desenvolvimento de sistemas. Desse total, 5 participaram dos testes com o IdDE, que foram realizados no período de Dezembro de 2010 e Fevereiro de 2011. Entre os participantes estavam analistas, programadores e o diretor de desenvolvimento

5.3.1 Descrição do cenário e metodologia

O objetivo principal da realização dos testes junto a essa empresa foi o de avaliar a aceitação da aplicação num ambiente real de desenvolvimento. A empresa já adota o Netbeans como IDE para o desenvolvimento e, da mesma forma, já utiliza o XMPP para a comunicação através de mensagens instantâneas, mantendo, para esse fim, um servidor na própria empresa. O Sistema Operacional instalado nas estações de trabalho dos participantes é o Linux OpenSUSE.

Antes da realização dos testes com a equipe, houve um contato com o diretor de desenvolvimento, para o qual foi apresentado o IdDE, além de terem sido feitos pequenos testes de edição colaborativa e troca de mensagens para a demonstração das funcionalidades. Posteriormente, o *plugin* a foi instalado nos equipamentos dos participantes e as instruções de uso foram disseminadas internamente pela própria equipe.

Durante o período de testes, a equipe utilizou a ferramenta de *chat* e a edição colaborativa. A comunicação através de mensagens de texto foi utilizada para a troca de ideias e solução de dúvidas, enquanto que a ferramenta de edição colaborativa foi utilizada em situações nas quais um programador necessitava de auxílio na criação do seu programa. Dessa forma, nos casos onde a comunicação através de *chat* não solucionava a dúvida, os programadores mais experientes puderam alterar diretamente o código, sem que houvesse a necessidade de deslocamento dentro da empresa. Aliado a isso, a ferramenta de edição colaborativa foi utilizada para o acompanhamento e inspeção da qualidade de código.

Apesar do nível de dispersão ser do tipo “Mesma localização física” na classificação de Audy e Prikladnicki (2008), considera-se que a realização dos testes neste cenário seja fundamental, pois além de serem realizados num ambiente de desenvolvimento real, acredita-se que o nível de exigência em relação do funcionamento do *software* num ambiente de produção também seja superior aos experimentos realizados nos cenários 1 e 2. Nesse sentido, qualquer problema apresentado pelo ambiente, trará impactos no resultado final dos testes e, certamente, influenciará na do IdDE. Aliado a isso, pelo fato de os profissionais já estarem habituados ao uso do Netbeans em seu dia-a-dia, também será mais fácil avaliar a resistência em relação à aceitação de uma nova ferramenta. Por fim, acredita-se que pelo fato de não ter havido um contato com toda a equipe participante, seja mais fácil avaliar a dificuldade que os usuários tiverem na utilização do *software*.

5.3.2 Análise de Resultados

Os resultados obtidos na avaliação dos testes neste cenário estão apresentados na Figura 68.

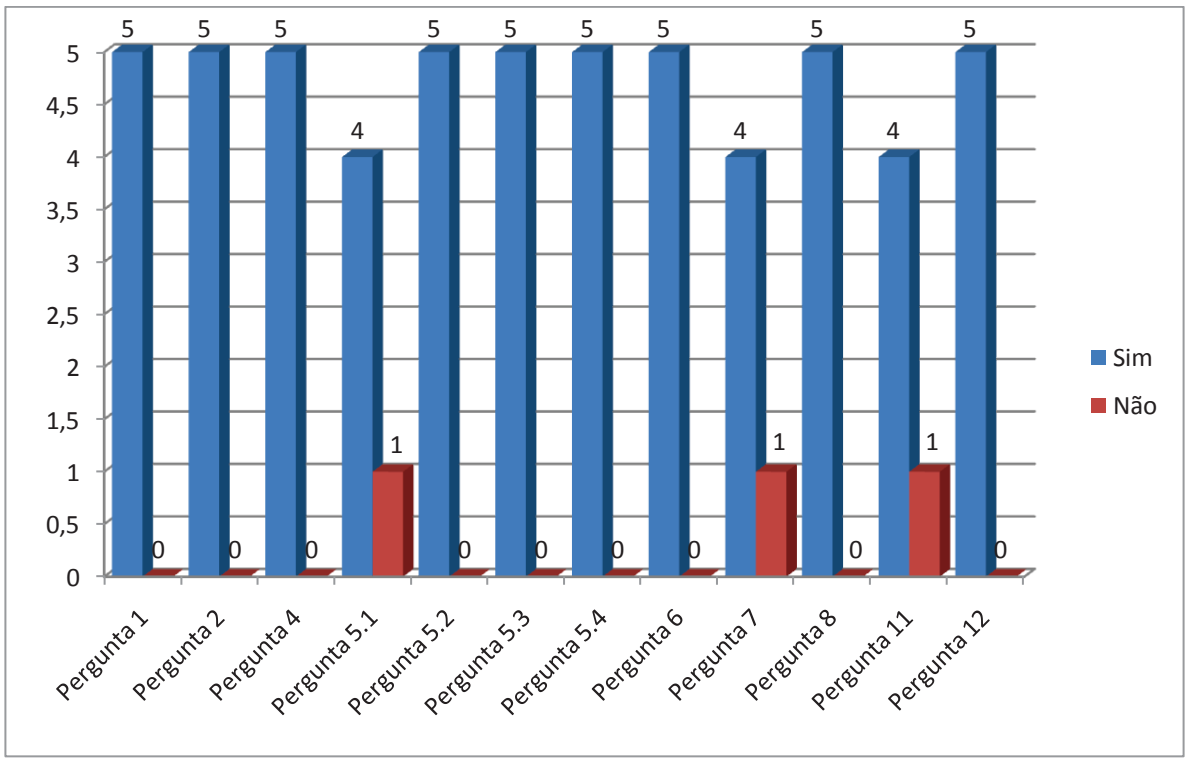


Figura 68: Resultado das avaliações do Estudo de Caso, Cenário 3.

Analisando o gráfico, percebe-se que todos os participantes consideram que o IdDE auxiliou na comunicação através de mensagens instantâneas (Pergunta 1). Da mesma forma, na opinião de todos os participantes o ambiente auxiliou no processo de edição compartilhada e colaborativa de código (Pergunta 2). Entre as justificativas, está o fato de a ferramenta ter se mostrado muito eficiente na edição compartilhada, sem apresentar problemas quando duas ou mais pessoas trabalharam simultaneamente no mesmo arquivo. Igualmente, afirmam que a ferramenta auxiliou nas situações em que foi necessário um trabalho conjunto num mesmo programa. Na avaliação do diretor de desenvolvimento, a ferramenta tornou mais prática e fácil a troca de experiências em situações que demandaram a orientação e auxilio aos desenvolvedores, tarefa que desempenha ordinariamente. Além disso, avalia que contribuiu também o fato de não ser necessário que todo o código fonte tivesse que estar num repositório SVN na máquina local.

A ferramenta de comunicação através de áudio não foi testada neste cenário, todavia, todos os participantes consideram que esse tipo de comunicação seja importante, quando os programadores estão separados geograficamente (Pergunta 4). Entre os motivos mencionados, está o fato de facilitar o entendimento entre os envolvidos sendo, geralmente, mais eficaz que outra forma de comunicação escrita, principalmente em situações que necessitam a tomada de decisões. Também houve a justificativa que o áudio agiliza o processo de comunicação e ajuda a evitar mal-entendidos. Da mesma forma, foi considerado que essa ferramenta complementa ideias e pensamentos, além de deixar a comunicação mais prática e dinâmica em comparação com o *chat*.

Em relação ao processo de ensino/aprendizagem presencial (Pergunta 5.1), apenas um participante considerou que a ferramenta não auxilia nesse contexto, justificando que a ferramenta seria dispensável. Contudo, entre aqueles que acreditam que o uso da ferramenta seja interessante também nesta situação, houve justificativas como o fato de não obrigar que dois ou mais programadores estejam sentados lado a lado utilizando o mesmo computador, mas sim, cada um podendo estar em seu próprio computador.

Todos os participantes consideraram que o IdDE contribui no processo de ensino/aprendizagem à distância (Pergunta 5.2). Entre as justificativas, está o fato de a ferramenta possibilitar que as pessoas possam trabalhar no mesmo código, mesmo estando longe. Aliado a isso, a ferramenta possibilita que se demonstre e explique um código simultaneamente para diversas pessoas, inclusive utilizando áudio.

A pergunta 5.3, que questiona sobre a contribuição da ferramenta para o desenvolvimento de *software* realizado por equipes que estão num mesmo ambiente, retrata a situação física deste grupo de estudo. Para todos os participantes, a contribuição da ferramenta é inquestionável nesta situação, pois possibilita que a equipe visualize o código e acompanhe uma demonstração mesmo que não haja uma sala de reuniões com projetor disponível. Da mesma forma, foi considerado que os programadores podem interagir num mesmo código fonte, cada um estando em seu computador.

Em relação ao desenvolvimento distribuído de *software* (Pergunta 5.4), a ferramenta foi considerada importante para 100% dos participantes. Entre as razões apontadas está a possibilidade de tradução das mensagens do *chat*, o que facilita a comunicação com pessoas de outros idiomas. Aliado a isso, a ferramenta de edição colaborativa foi considerada de extrema importância por possibilitar que os usuários trabalhem simultaneamente no mesmo código fonte.

No que se refere à experiência de uso e praticidade funcional (Pergunta 6), os participantes foram unânimes em considerar que a integração da ferramenta ao Netbeans tenha contribuído positivamente. Nesse sentido, afirmaram que, quanto mais integrada for a ferramenta de desenvolvimento, mais prática e ágil será a sua utilização. Da mesma forma, houve afirmações que consideram o Netbeans como uma ferramenta muito boa para o desenvolvimento e a integração de novas funcionalidades enriquece o ambiente, pois todas as funcionalidades e necessidades estão atendidas num mesmo IDE.

A disposição das janelas dentro do Netbeans e a Interface Homem-Computador (Pergunta 7) foram consideradas inadequadas por apenas um usuário. Como justificativa, está o fato de as janelas de *chat* não estarem integradas totalmente ao Netbeans, não possibilitando o seu acoplamento, como as demais janelas.

Em relação à performance do Netbeans, todos os participantes consideraram que não houve mudança após a instalação do *plugin* (Pergunta 8).

Na pergunta 9, que solicita que cada participante descreva o principal benefício proporcionado pela ferramenta, as principais considerações fizeram referência à edição simultânea por várias pessoas, tornando o desenvolvimento mais ágil, rápido e confortável, mesmo que não se esteja fisicamente próximo e sem a ocorrência de problemas. A possibilidade de poder revisar e inspecionar o código também foram apontados como benefícios da ferramenta. Da mesma forma, a facilidade de comunicação e o fato da ferramenta não exigir que cada usuário tenha que manter uma cópia local de todo o repositório SVN do projeto também foram apontados como benefícios.

Ao considerarem melhorias ou novas funcionalidades que pudessem ser incorporadas ao ambiente (Pergunta 10), os usuários fizeram referência à integração das janelas de chat ao ambiente do Netbeans, permitindo que elas fiquem acopladas como janelas nativas. Aliado a isso, foram sugeridas mudanças na lista de usuários, solicitando que o nome do usuário fosse substituído pelo nome completo do pessoa. Por fim, também foram sugeridas mudanças nos painéis de configuração do *plugin*.

Entre os participantes dos testes, apenas um considerou que não utilizaria o *plugin* no seu processo diário de desenvolvimento (Pergunta 11), sem apresentar, no entanto, as razões para essa escolha. Todos os demais utilizariam a ferramenta, alegando que ela é de grande auxílio no desenvolvimento compartilhado, possibilitando que cada usuário permaneça em seu computador e, eventuais problemas de código possam ser encontrados mais rapidamente. Além disso, também foi considerado importante o fato de a ferramenta integrar o *chat*

diretamente no Netbeans, sem haver a necessidade de utilizar uma ferramenta externa, como ocorre atualmente.

Em relação à produtividade (Pergunta 12), todos os participantes consideraram que a ferramenta contribui para o seu aumento. Nesse sentido, afirmaram que os problemas podem ser diagnosticados mais rapidamente, além de reduzir a probabilidade de conflitos no desenvolvimento, tornando mais rápido o aprendizado. Também foi considerado que o ambiente reduziu o tempo de configuração em termos de facilidade de acesso e compartilhamento de código. Até então, a empresa utilizava aplicativos de compartilhamento de área de trabalho e acesso remoto, mas estes se mostraram mais pesados e mais complicados de configurar, além de apresentar problemas como, por exemplo, as diferentes características de vídeo entre as estações de trabalho.

No item 13, que solicitava que os usuários adicionassem sugestões, comentários e outros aspectos relevantes observados durante os testes, foi sugerido que o ambiente proteja os dados transmitidos através de algum mecanismo de criptografia. Além disso, houve manifestações que consideraram o IdDE uma ótima ferramenta e de muita utilidade.

5.4 Cenário 4 – Profissionais de Instituições de Ensino

Neste estudo de caso, foram realizados experimentos com profissionais de desenvolvimento de *software* de três instituições de ensino superior do país: UFJF (Universidade Federal de Juiz de Fora/MG), UCS (Universidade de Caxias do Sul/RS) e FAI (Centro de Ensino Superior em Gestão, Tecnologia e Educação, de Santa Rita do Sapucaí/MG).

Além da edição colaborativa e a comunicação utilizando mensagens de texto, neste estudo de caso foram feitos testes utilizando a comunicação através de áudio.

5.4.1 Descrição do cenário e metodologia

Neste cenário, cujo nível de dispersão é do tipo “Distância Nacional”, conforme a classificação de Audy e Prikladnicki (2008), não houve nenhum contato pessoal para apresentação ou explicações detalhada das ferramentas do IdDE. Todos os contatos e instruções sobre a instalação e funcionamento do *plugin* foram mantidos através de e-mail e *chat*. Os testes ocorreram no mês de Fevereiro de 2011 e nenhum dos participantes utilizava o Netbeans em seu dia-a-dia.

Com a UCS, os testes foram realizados com a Coordenadora do Setor de Sistemas e as ferramentas testadas foram: edição colaborativa e *chat*. Já no caso da FAI, os testes ocorreram com a Coordenadora de TI e as ferramentas utilizadas foram: edição colaborativa, *chat* e comunicação via áudio.

Na UFJF, os testes aconteceram com um Analista de Sistemas e as funcionalidades testadas foram edição colaborativa, *chat*, áudio, ferramenta de tradução, gerenciamento de tarefas e agenda. Além disso, ocorreram testes internos entre dois analistas da UFJF.

Para a realização desses testes, o servidor XMPP utilizado foi o GMail e para a comunicação através de áudio foi utilizado um servidor Asterisk.

Além da execução dos testes individuais com os representantes de cada instituição, ocorreu um teste conjunto com o analista de sistemas da UFJF e a Coordenadora de TI da FAI, momento no qual foram testadas as ferramentas de edição colaborativa, *chat* e áudio. Para a realização destes testes, foi efetuada a instalação do servidor XMPP OpenFire ao invés do uso do servidor do GMail.

Os testes realizados nestes cenários seguiram a mesma metodologia: primeiro eram feitas modificações nos códigos fontes e, posteriormente, eram confrontadas as versões finais dos documentos em cada ambiente para a verificação da consistência e integridade dos códigos.

5.4.2 Análise de Resultados

A avaliação dos testes neste cenário foi realizada por um total de quatro pessoas: uma da UCS, duas da UFJF e uma da FAI. Os resultados das avaliações deste cenário são apresentados na Figura 69.

Como é possível observar no gráfico, todos os participantes consideram que a ferramenta auxiliou no processo de comunicação através de *chat* (Pergunta 1).

Da mesma forma todos os participantes avaliaram que o ambiente auxilia no processo de edição colaborativa de *software* (Pergunta 2). Entre as justificativas apresentadas está a simplicidade e a funcionalidade do ambiente e o fato de se poder acompanhar a posição onde o usuário remoto está efetuando as modificações. Além disso, também foi citada a rapidez da atualização do arquivo remoto, muito superior se comparada ao uso os tradicionais aplicativos de acesso remoto. Por fim, foi mencionado que a edição compartilhada permite a prática de programação em pares do XP de uma forma muito mais efetiva.

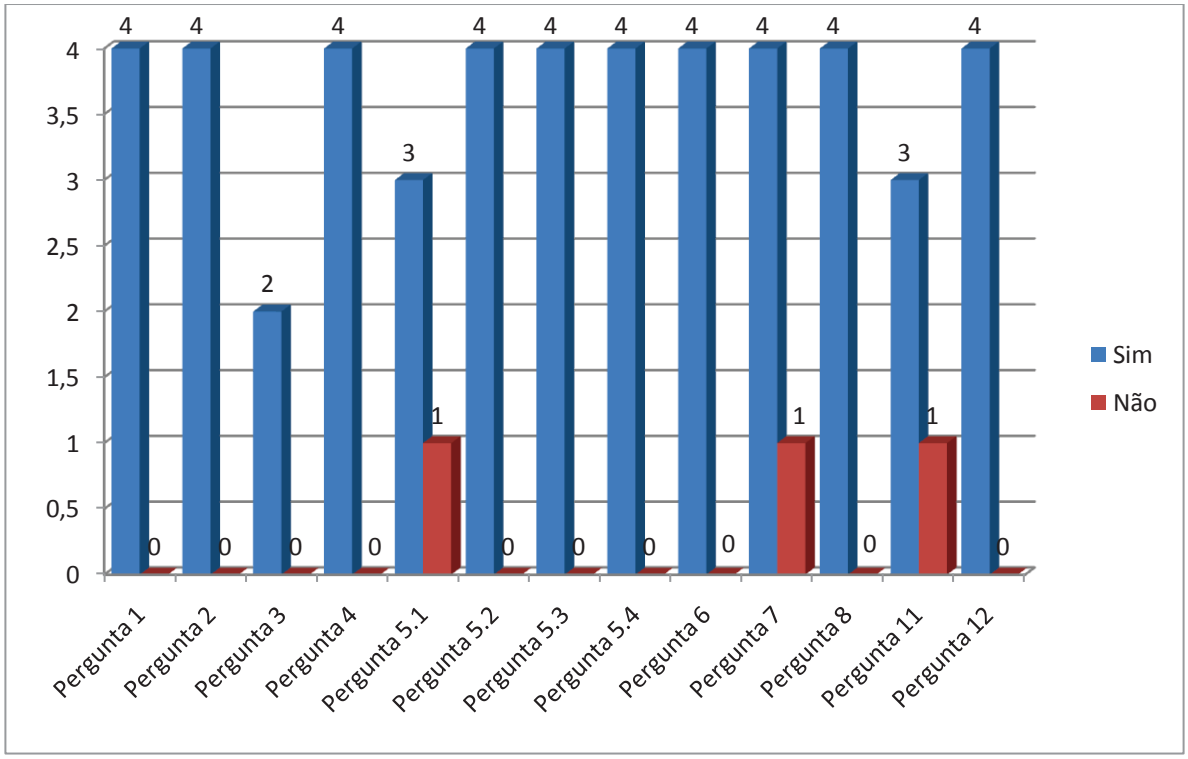


Figura 69: Resultado das avaliações do Estudo de Caso, Cenário 4.

A ferramenta de áudio se mostrou útil e funcional para ambos os participantes que realizaram este teste (Pergunta 3). Igualmente, todos os participantes dos testes deste cenário consideram a comunicação via áudio importante num cenário onde os programadores estão separados geograficamente (Pergunta 4). Entre os motivos alegados, está o fato de que, se um programador precisar explicar as modificações feitas num código, é muito mais rápido e fácil que o *chat* textual. Por outro lado, também foi considerado que o uso do áudio seja importante, porém não essencial, avaliando que os programadores estão mais acostumados a utilizar o *chat* tradicional.

Em relação à contribuição da ferramenta para o processo de ensino/aprendizagem presencial (Pergunta 5.1), apenas um participante considerou que não tinha condições de avaliar a ferramenta desse ângulo por não ter utilizado ela para esse fim. Os demais consideraram que ela seja útil por auxiliar em diversas situações como, por exemplo, permitir que um professore corrija os programas dos alunos num laboratório, sem a necessidade de ficar se deslocando de computador em computador.

Todos os participantes consideraram a ferramenta importante num processo de ensino/aprendizagem à distância (Pergunta 5.2), avaliando que a ferramenta permite que um

professor possa interagir com os alunos à distância e *online*. Da mesma forma, possibilita que o professor corrija e acompanhe os programas dos alunos.

Na avaliação dos participantes do estudo de caso, o ambiente contribui no desenvolvimento de *software* realizado por equipes que estão num mesmo ambiente (Pergunta 5.3). Consideram que o compartilhamento de código *online* permite que os membros da equipe se auxiliem na correção de *bugs* e na troca de idéias sobre determinada implementação. Além disso, argumentaram que a ferramenta pode servir muito bem para equipes que utilizam a metodologia de programação em pares.

Em relação ao desenvolvimento distribuído de *software* (Pergunta 5.4), todos os participantes consideram que o IdDE contribua para esse modelo de desenvolvimento. Entre as justificativas, está o fato de a ferramenta possibilitar o acesso instantâneo às alterações do código, proporcionando uma melhor forma de interação e aprendizagem, contribuindo para o desenvolvimento do *software*. Da mesma forma, consideram que a ferramenta quebra a barreira das distâncias, possibilitando que os usuários consigam trabalhar da mesma forma como se estivessem utilizando a ferramenta estando num mesmo ambiente. Além disso, também contribui para essa avaliação a possibilidade de traduzir as mensagens de *chat*.

Na opinião de todos os participantes, o fato do ambiente ter sido desenvolvido como módulo do Netbeans tenha contribuído para enriquecer sua a qualidade e praticidade funcional (Pergunta 6). Entre as razões apontadas, está o fato de o Netbeans ser uma ferramenta bastante completa para programação em diversas linguagens de programação e já possui diversas facilidades para o desenvolvimento de projetos, e o IdDE enrique o ambiente fazendo uso de funcionalidades já existentes.

Para os participantes, a disposição das janelas dentro do Netbeans está adequada (Pergunta 7). Entretanto, foram sugeridas mudanças de forma que as janelas de *chat* sejam agrupadas e acopladas ao ambiente ao invés de ficarem "soltas".

No que se refere à performance do Netbeans, nenhum dos participantes acredita que tenha havido qualquer tipo de mudança após a instalação do *plugin* (Pergunta 8).

Entre os principais benefícios proporcionados pela ferramenta (Pergunta 9) está a possibilidade de trabalho conjunto para a correção de *bugs* ou implementação de soluções pensadas colaborativamente de maneira muito mais rápida e eficiente. Aliado a isso, consideraram interessante também a possibilidade de duas ou mais pessoas poderem se comunicar e realizar alterações instantâneas no código e simultaneamente.

No item 10, que solicitou que fossem citadas características ou melhorias que pudessem ser implementadas no ambiente, foi sugerida a possibilidade de compartilhar com o ambiente remoto todos os arquivos abertos no Netbeans, de uma só vez. Além disso, foi sugerida a criação de um agenda telefônica, que pudesse ser utilizada com o SIP. Por fim, também foi mencionada a possibilidade de guardar as configurações mais usadas, para que não seja necessária sua digitação no caso da utilização de outros servidores XMPP e SIP.

Em relação à utilização da ferramenta no dia-a-dia (Pergunta 11), apenas um participante não considerou esta possibilidade pelo fato de ainda não ter usado muito a ferramenta. Os demais consideraram que utilizariam a ferramenta em seu trabalho cotidiano.

Todos os participantes deste cenário acreditam que a ferramenta contribua para um aumento da produtividade (Pergunta 12). Entre os principais motivos apontados está o fato de ser possível a comunicação entre os desenvolvedores, além de possibilitar que todos os usuários consigam visualizar as alterações nos códigos imediatamente, diminuindo o tempo de compreensão e resposta.

6 CONCLUSÕES

O desenvolvimento de *software* geograficamente distribuído tem aumentado significativamente nos últimos anos e tornou-se uma necessidade de negócio para muitas corporações.

Desde que surgiu, o DDS mudou grande parte da tradição do desenvolvimento de *software* e criou uma nova classe de problemas a serem resolvidos pelos pesquisadores da área de desenvolvimento de *software*. Entre os problemas e desafios desse modelo estão: questões estratégicas, questões culturais, comunicação inadequada, gestão do conhecimento, alocação de tarefas, confiança, questões técnicas, entre outros.

Nesse sentido, a maior contribuição deste trabalho está no fato de propor um ambiente computacional integrado a uma ferramenta de mercado já existente, possibilitando a definição de mecanismos de Coordenação, Comunicação e Controle para o Desenvolvimento Colaborativo e Distribuído de *Software*.

Os resultados das avaliações dos estudos de caso comprovam que os objetivos traçados no planejamento deste trabalho foram plenamente atingidos. Na opinião de todos os participantes, as ferramentas de chat e edição colaborativa contribuem decisivamente para o desenvolvimento de *software*.

Da mesma forma, os participantes dos estudos de caso consideraram que as ferramentas do ambiente tornam possível sua utilização no processo de ensino/aprendizagem, tanto presencial quando à distância, bem como validam seu uso no desenvolvimento de *software* tanto por equipes locais, quando distribuídas.

Em relação à produtividade, 100% dos usuários consideraram que as ferramentas do ambiente contribuem para o seu aumento. No que se refere à utilização do Netbeans como base para a criação do aplicativo, todos os participantes avaliaram que tenha sido uma escolha apropriada, salientando que aplicativo desenvolvido agrega funcionalidades importantes para este IDE, tornando-o ainda mais completo.

Os resultados demonstram ainda que as escolhas dos protocolos XMPP e SIP se mostraram acertadas. Além de possibilitarem a interoperabilidade do ambiente com outros aplicativos e dispositivos, diversas avaliações fizeram referência à velocidade e rapidez das ferramentas do ambiente, principalmente se comparadas a ferramentas normalmente utilizadas no desenvolvimento de *software*, como compartilhamento de área de trabalho e acesso remoto aos computadores.

6.1 Trabalhos Futuros

Durante o desenvolvimento do aplicativo percebeu-se que outros recursos interessantes poderiam ser incorporados ao ambiente. Da mesma forma, com a realização dos estudos de caso foram sugeridas várias funcionalidades a serem desenvolvidas em trabalhos futuros. Entre as funcionalidades, pode-se destacar:

- Implementar um algoritmo de controle e ordenação das mensagens, que garanta a unicidade de código para todos os participantes de uma sessão de edição colaborativa;
- Melhorar a apresentação das mensagens exibidas ao passar com o mouse sobre o ícone verde, que informam as alterações efetuadas num documento, agrupando as modificações por autor e possibilitando um acesso direto à janela do histórico das alterações;
- Adicionar suporte a vídeo;
- Desenvolver uma ferramenta de quadro branco, que permita que os usuários compartilhem imagens e desenhos, à exemplo do que ocorre com a edição colaborativa;
- Possibilitar a criptografia das mensagens que são enviadas de um ambiente para outro;
- Agrupar e integrar as janelas de chat ao ambiente do Netbeans;
- Portar o *plugin* para outros IDEs;
- Possibilitar a seleção do Codec de áudio a ser utilizado nas comunicações;
- Implementar uma ferramenta que possibilite a execução passo-a-passo todo o histórico de alterações de uma sessão de edição colaborativa, simulando todo o processo ocorrido anteriormente;

REFERÊNCIAS

[Audy e Prikladnicki, 2008] AUDY, J., PRIKLADNICKI R. Desenvolvimento Distribuído de Software. 1. ed. Rio de Janeiro: Elsevier, 2008. ISBN 978-85-352-2720-8

[Baheti *et al*, 2002] BAHETI, P., GEHRINGER, E., STOTTS D. Exploring the efficacy of distributed pair programming. Extreme Programming and Agile Methods – X/P Agile Universe 2002, August, 2002.

[Böck, 2009] BÖCK, Heiko. The Definitive Guide to NetBeans™ Platform. 1. ed. New York, NY: Apress, 2009. ISBN 978-1-4302-2417-4

[Borges *et al*, 2007] BORGES, R. M.; PINTO, S. C. C. S.; BARBOSA J. L. V.; BARBOSA D. N. F., Usando o modelo 3C de colaboração e Vygotsky no ensino de programação distribuída em pares. XVIII Simpósio Brasileiro de Informática na Educação (SBIE), 2007.

[Canfora *et al*, 2003] CANFORA, G.; CIMITILE, A.; VISAGGIO, C.A., Lessons learned about distributed pair programming: what are the knowledge needs to address?. Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003. WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on , vol., no.pp. 314- 319, 9-11 June, 2003,

[Carmel, 1999] CARMEL, Erran. Global Software Teams – Collaborating Across Borders and Time Zones. Upper Saddle River, NJ: Prentice Hall, 1999. ISBN 0-13-924218-X.

[Carmel e Tjia, 2005] CARMEL, Erran, TJIA, Paul. Offshoring Information Technology – Sourcing and Outsourcing to a Global Workforce. Cambridge, UK: Cambridge University Press, 2005. ISBN 978-0-521-84355-3

[Carmel e Agarwal, 2001] CARMEL E., AGARWAL R., "Tactical Approaches for Alleviating Distance in Global Software Development," IEEE Software, pp. 22-29, March/April, 2001.

[Cheng *et al*, 2003] CHENG, L.; DE SOUZA, C.R.B.; HUPFER, S.; ROSS, S.; PATTERSON, J. "Building Collaboration into IDEs.", ACM Queue, Vol. 1, N. 9, pp:40-50, December / January 2003/2004

[Cockburn e Williams, 2001] COCKBURN, A., WILLIAMS, L. The costs and benefits of pair programming, Extreme Programming Examined, Succi, G., Marchesi, M. eds., pp. 223-248, Boston, MA: Addison Wesley, 2001. ISBN 0-201-71040-4

[Damian e Moitra, 2006] DAMIAN, Daniela; MOITRA, Deependra. "Global Software Development: How Far Have We Come?" IEEE Software, Vol. 23, No. 5, 2006.

[Davidson *et al*, 2006] DAVIDSON, Jonathan; PETERS, James; BHATIA, Manoj; KALIDINDI, Satish; MUKHERJEE, Sudipto. Voice Over IP Fundamentals. 2. ed. Indianápolis, IN: Cisco Press, 2006. ISBN 1-58705-257-1

[Deitel e Deitel, 2006] DEITEL, P.J.; Deitel, H.M. JAVA How to Program. 7. Ed. New Jersey: Prentice Hall, 2006. ISBN 9780136085676

[Ellis e Gibbs, 1989] ELLIS, C. A.; GIBBS, S.J. Concurrency control in groupware systems. In: SIGMOD '89, Proceedings of the 1989 ACM SIGMOD international conference on Management of data, New York, NY, 1989.

[German, 2003] GERMAN, D. GNOME, a case of open source global software development. International Workshop on Global Software Development (GSD 2003), Portland, 2003.

[German, 2004] GERMAN, D. The GNOME project: a case study of open source, global software development. Journal of Software Process: Improvement and Practice, 8(4):201–215, 2004.

[Gotel *et al*, 2006] GOTEL, Olly; SCHARFF, Christelle; SENG, Sopheap. Preparing Computer Science Students for Global Software Development. In 36th ASEE/IEEE Frontiers in Education Conference, San Diego, CA... October, 2006.

[Herbsleb, 2007] HERBSLEB, J. D. Global Software Engineering: The Future of Socio-Technical Coordination, Proceedings of the 29th International Conference on Software Engineering (ICSE), p. 188-198. Minneapolis, 2007.

[Herbsleb e Grinter, 1999] HERBSLEB, J. D., GRINTER, R. E. Splitting the Organization and Integrating the Code: Conway's Law Revisited. In: ICSE'99, Los Angeles, CA. Proceedings… 1999.

[Herbsleb e Moitra, 2001] HERBSLEB, James D.; MOITRA Deependra. Guest Editors' Introduction: Global Software Software Development. IEEE Software, 18(2), p. 16-20, March/April 2001.

[Herbsleb e Mockus, 2003] HERBSLEB, James D.; MOCKUS, Audris. An Empirical Study of Speed and Communication in Globally Distributed Software Development. IEEE Transactions on Software Engineering, 29(6), 2003.

[Hersent, 2010] HERSENT, Olivier. IP telephony: deploying VoIP protocols and IMS infrastructure. 2. ed. West Sussex, UK: John Wiley & Sons, 2010. ISBN 978-0-470-66584-8

[Karolak, 1998] KAROLAK, Dale Walter. Global Software Development: Managing Virtual Teams and Environments. 1. ed. Los Alamitos, California: IEEE Computer Society Press, 1998. ISBN 0-8186-8701-0.

[Kotonya e Sommerville, 1998] KOTONYA, Gerald; SOMMERVILLE, Ian. Requirements Engineering: Processes and Techniques. 1. ed. New York: John Wiley & Sons, 1998. ISBN 0-471-97208-8.

[Lamersdorf *et al,* 2009] LAMERSDORF, Ansgar; MUNCH, Jürgen; ROMBACH, Dieter. A Survey on the State of the Practice in Distributed Software Development: Criteria for Task Allocation. In Proceedings of the 2009 Fourth IEEE International Conference on Global Software Engineering (ICGSE), pp.41-50, 2009.

[Lawrence, 2009] LAWRENCE, E.. CollabEd: A Platform for Collaboratizing Existing Editors. International Conference on Mobile, Hybrid, and On-line Learning, 2009.

[Maybury, 2001] MAYBURY, M., Collaborative Virtual Environments for Analysis and Decision Support. Communications of the ACM, 2001, p. 51-54.

[Miller, 2001] MILLER, Jeremie. Jabber Conversation Technologies. In Peer-to-Peer: Harnessing the Power of Disruptive Technologies. 1. ed. Sebastopol/CA: O'Reilly & Associates, 2001. ISBN 0-596-00110-X.

[Netbeans, 2010] NETBEANS IDE. Disponível em <http://netbeans.org>. Acesso em: 30 de Maio de 2010.

[Paasivaara e Lassenius, 2004] PAASIVAARA M. e LASSENIUS C.. Collaboration practices in global inter-organizational software development projects. Software Process: Improvement and Practice, 8(4):183–199, 2004.

[Petri, 2010] PETRI, Jürgen. NetBeans Platform 6.9 Developer's Guide. 1. ed. Birmingham,UK: Packt Publishing, 2010. ISBN 978-1-849511-76-6

[Pressman, 2010] PRESSMAN, Roger S. Engenharia de Software. 6. ed. Porto Alegre, RS: AMGH, 2010. ISBN 978-85-63308-00-9

[Prikladnicki *et al*, 2004] PRIKLADNICKI, Rafael; LOPES, Leandro; AUDY, Jorge L. N.; EVARISTO, Roberto. Desenvolvimento Distribuído de Software: um Modelo de Classificação dos Níveis de Dispersão dos Stakeholders. Simpósio Brasileiro de Sistemas de Informação (SBSI). Porto Alegre, 2004.

[Sahay *et al*, 2007] SAHAY, Sundeep; NICHOLSON, Brian; KRISHNA, S. Global IT Outsourcing: Software Development Across Borders. 1. ed. New York, NY: Cambridge University Press, 2007. ISBN 0-521-81604-1

[Saint-Andre *et al*, 2009] SAINT-ANDRE, Peter; SMITH, Kevin; TRONÇON, Remko. XMPP: The Definitive Guide – Building Real-Time Applications with Jabber Technologies. 1. ed. Sebastopol, CA: O'Reilly Media, 2009. ISBN 978-0-596-52126-4

[Sangwan *et al*, 2006] SANGWAN, Raghvinder; BASS, Matthew; MULLICK, Neel; PAULISH, Daniel J.; KAZMEIER, Juergen. Global Software Development Handbook (Auerbach Series on Applied Software Engineering Series), Auerbach Publications, Boston, MA, 2006. ISBN 0-8493-9384-1

[Scacchi, 2002] SCACCHI, W. Understanding the requirements for developing open source software systems. IEEE Software, 149(1):24–39, Fevereiro 2002.

[Schümmer e Schümmer, 2001] Schümmer, T., Schümmer J. Support for distributed teams in Extreme Programming. Extreme programming examined, Succi, G., Marchesi, M. eds., p. 355–377, Boston, MA: Addison Wesley, 2001. ISBN 0-201-71040-4

[Sinnreich e Johnston, 2006] SINNREICH, Henry; JOHNSTON, Alan B.. Internet Communications Using SIP: Delivering VoIP and Multimedia Services with Session Initiation Protocol. 2. ed. Indianapolis, IN: Wiley Publishing, 2006. ISBN 0-471-77657-2

[Sommerville, 2003] SOMMERVILLE, Ian. Engenharia de Software. 6. ed. São Paulo, SP: Addison Wesley, 2003. ISBN 85-88639-07-6.

[Trindade *et al*, 2008] TRINDADE, Daniela de Freitas Guilhermino; TAIT, Tania Fatima Calvi; HUZITA, Elisa Hatsue Moriya. A tool for supporting the communication in distributed software development environment. Journal of Computer Science & Technology, 8 (2), pp: 118-124, 2008.

[Williams e Kessler, 2000] WILLIAMS, L. A., KESSLER, R. R. 2000. All I really need to know about pair programming I learned in kindergarten. Commun. ACM 43, 5 (May. 2000), 108-114.

[XMPP, 2010] XMPP: Extensible Messaging and Presence Protocol. Disponível em: <http://xmpp.org>. Acesso em: 19 de Março de 2010.