EDENILSON JOSÉ DA SILVA

WORKSPACE SEMÂNTICO: INTEGRANDO ARTEFATOS E DESENVOLVEDORES NO AMBIENTE DE DESENVOLVIMENTO DE SOFTWARE

CURITIBA

2014

EDENILSON JOSÉ DA SILVA

WORKSPACE SEMÂNTICO: INTEGRANDO ARTEFATOS E DESENVOLVEDORES NO AMBIENTE DE DESENVOLVIMENTO DE SOFTWARE

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná, como requisito parcial para obtenção do título de Doutor em Informática.

Área de Concentração: *Ciência da Computação.*

Orientador: Prof. Dr. Emerson Cabrera Paraiso.

CURITIBA

2014

Dados da Catalogação na Publicação
Pontifícia Universidade Católica do Paraná
Sistema Integrado de Bibliotecas – SIBI/PUCPR
Biblioteca Central

Silva, Edenilson José da
S586w
2014
Workspace semântico : integrando artefatos e desenvolvedores no ambiente de desenvolvimento de software / Edenilson José da Silva ; orientador, Emerson Cabrera Paraiso. – 2014.
xx, 185 f. : il. ; 30 cm

Tese (doutorado) – Pontifícia Universidade Católica do Paraná, Curitiba, 2014
Bibliografia: f. 135–147

1. Software – Desenvolvimento. 2. Programas de computador – Verificação. 3. Informática. I. Paraiso, Emerson Cabrera. II. Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática. III. Título.

CDD 20. ed. – 004

Pontifícia Universidade Católica do Paraná
Escola Politécnica
Programa de Pós-Graduação em Informática

PUCPR
GRUPO MARISTA

ATA DE DEFESA DE TESE DE DOUTORADO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

ÁREA DE CONCENTRAÇÃO: CIÊNCIA DA COMPUTAÇÃO

DEFESA DE TESE DE DOUTORADO Nº 026/2014

Aos 19 dias de Agosto de 2014 realizou-se a sessão pública de Defesa da Tese de Doutorado intitulada **"Workspace Semântico: Integrando Artefatos e Desenvolvedores no Desenvolvimento de Software"** apresentada pelo aluno **Edenilson José da Silva** como requisito parcial para a obtenção do título de Doutor em Informática, perante uma Banca Examinadora composta pelos seguintes membros:

Prof. Dr. Emerson Cabrera Paraiso
PUCPR (Orientador) _____ APROV
(assinatura) *(aprov/reprov.)*

Profª. Drª. Andreia Malucelli
PUCPR _____ APROV.

Prof. Dr. Julio César Nievola
PUCPR _____ APROV.

Profª. Drª. Silvia Regina Vergilio
UFPR _____ aprov.

Prof. Dr. Milton Pires Ramos
TECPAR _____ aprovado

Conforme as normas regimentais do PPGIa e da PUCPR, o trabalho apresentado foi considerado APROVADO (*aprovado/reprovado*), segundo avaliação da maioria dos membros desta Banca Examinadora. Este resultado está condicionado ao cumprimento integral das solicitações da Banca Examinadora registradas no Livro de Defesas do programa.

Profª. Drª. Andreia Malucelli
Coordenadora do Programa de Pós-Graduação em Informática

Programa de Pós-Graduação em Informática PPGIa PUCPR

Rua Imaculada Conceição, 1155 Prado Velho CEP 80215 901 Curitiba Paraná Brasil
Tel.: (41) 3271 1669 Fax.: (41) 3271 2121 www.ppgia.pucpr.br

Resumo

Para desenvolver software é necessária a integração de várias ferramentas e o envolvimento de profissionais que desempenham diversos papéis para executar esta atividade. Neste escopo, este trabalho propõe um ambiente de apoio que permite que a visibilidade do que está sendo desenvolvido seja ampliada, principalmente do ponto de vista dos desenvolvedores. Este ambiente foi denominado como Workspace Semântico e foi concebido para pequenas equipes que desenvolvem software. Seu principal objetivo é ligar semanticamente os artefatos produzidos durante o processo de desenvolvimento de software, especialmente por meio da monitoração do código-fonte produzido durante a fase de programação. Um levantamento do estado da arte foi realizado, bem como uma pesquisa com desenvolvedores, gestores de projeto de software e profissionais da área da informática para levantar informações referentes à forma, métodos e ferramentas utilizadas por estes usuários. Dois experimentos foram conduzidos para avaliar o protótipo: um experimento piloto, conduzido com alunos de uma universidade e outro conduzido em um instituto de tecnologia. Os resultados obtidos auxiliaram a expor as funcionalidades do ambiente e atestar sua efetividade para criar relacionamentos semânticos entre os artefatos e aumentar a visibilidade do software sendo desenvolvido.

Palavras-Chave: Verificação de Software, Desktop Semântico, Desenvolvimento Colaborativo de Software, Workspace Semântico.

Abstract

In software development process, it is necessary the integration of different tools and the involvement of professionals who perform various functions. In this context, this research presents an environment for improvement of visualization, mainly from the point of view of software developers. This environment was named “Workspace Semântico”, and is designed for small teams of software development. Its main goal is to semantically interconnect the artifacts produced during software development, in particular monitoring the source code produced during the phase of programming of source code. A study about the state of the art was done, and a survey with developers, project managers and information technology professionals to collect information about how, methods and tools they used for this type of work. Two experiments were conducted to evaluate the prototype: a pilot experiment conducted with under-graduate students and another conducted in an institute of technology. The results helped to expose the functionality of the environment and prove the effectiveness in creating semantic relationships among the artifacts involved and improved the visibility of software being developed.

Keywords: Software Verification, Semantic Desktop, Collaborative Software Development, Semantic Workspace

Dedicatória

A toda minha família,
principalmente aos meus filhos,
que considero uma extensão da minha vida nas suas vidas.

Agradecimentos

São tantas pessoas que tenho que agradecer, que pensei em acrescentar mais um capítulo na tese. Isto não é possível, então irei me estender aqui, e espero não esquecer ninguém, pois este trabalho teve a auxílio de inúmeras pessoas, mesmo porque, sozinho, pouco poderia ser realizado. Foram muitos amigos, colegas de trabalho, colegas de estudo, professores, profissionais com quais mantive contato e pude solucionar dúvidas e trocar experiências para conduzir a pesquisa aqui realizada.

Inicialmente, meus agradecimentos aos colegas e amigos que fiz durante o tempo do doutorado no PPGIA. Alguns já conhecia como ex-alunos e colegas de trabalho, e outros tive o privilégio de partilhar experiências, tanto nas disciplinas como na convivência nos laboratórios de pesquisa. A todas estas pessoas minha gratidão pelo tempo dispendido na boa convivência: Mariza Miola Dosciatti, Andreia Marini, Irapuru Haruo Flórido, Franciele Beal, Glauco Carlos Silva, Viviane Dal Molin de Souza, Lohann Coutinho Paterno, Gregory Moro Puppi Wanderley, Daniel Sampaio, Alef Turatti, Elias Cesar Araujo de Carvalho, Rodolfo Botto Garcia, Ronan Assunção Silva, Joselaine Valaski, Fernando Schutz, Rosana Lachowski, Geraldo Ranthum, Leila Vriesmann, Ederson Marcos Sgarbi, Simone Bello Caminski Aires, Rafael Abud Menezes, Carla Machado da Trindade, José Adilson Lopes da Silva, André Pinz Borges, André Luiz Brun, Osmar Betazzi Dordal, Helyane Bronoski Borges, Luiz Melo Romão, Mauri Ferrandin, Regina Fabia Albuquerque, Marcus Vinícius Mazega Figueredo, Richardson Ribeiro, João Coelho Neto, Tânia Lúcia Monteiro, Jean-Paul Barddal, Alonso de Carli, Priscila Louise Leyser Santim, Patrícia Antoniolli, Patricia Rucker de Bassi, Denise Maria Veccino Sato, Kelly Bettio, Aline Maria Malachini Miotto Amaral, Marcelo Perez, Júlio César Zanoni, Geraldo Boz Junior, Lucas Galete, Laercio Martins Carpes, Heitor Murilo Gomes, Elieser Botelho Manhas Junior, Marcelo Zanetti, Marco Antonio Paludo, Bruno Miguel Souza, João Eugenio Marynowski, Rodrigo Siega, Ayslan Trevizan Possebom e tantos mais que partilharam comigo seu tempo e dedicação às atividades que compartilhamos durante o tempo do meu doutoramento. Em especial,

um agradecimento ao André Luís Andrade Menolli, que se mostrou um ótimo companheiro de estudos, grande pesquisador e amigo nesta caminhada.

Aos professores, colegas de trabalho e amigos da UTFPR – Campus Pato Branco, que forneceram suporte durante o tempo que estive afastado para cursar o doutorado, absorvendo minhas atividades e aulas para que eu pudesse qualificar-me profissionalmente: Ademir Roberto Freddo, Omero Francisco Bertol, Edilson Pontarolo, Mariza Miola Dosciatti, Eden Ricardo Dosciatti, Eliane Maria de Bortolli Fávero, Fábio Favarim, Geri Natalino Dutra, Kathya Silvia Collazos Linares, Robison Cris Brito, Adriana Santos Auzani, Jorge Roberto Grobe, Henrique Oliveira da Silva, Rubia Eliza de Oliveira Schultz, Soelaine Rodrigues, Vinicius Pegorini, Andreia Scariot Beulke, Luis Carlos Ferreira Bueno e demais professores da Coordenação de Informática (agora DAINF), por todo apoio, compreensão e incentivo, a todos agradeço de todo o coração a oportunidade concedida. De forma muito especial, agradeço a professora Beatriz Borsoi, que desde o rascunho do primeiro projeto até a versão final da tese, opinou, deu sugestões, torceu, incentivou, enfim, participou de forma muita ativa nesta empreitada.

A todos os alunos da PUCPR e também aos servidores do TECPAR, que participaram dos estudos e experimentos, contribuindo de forma significativa para a realização deste trabalho. Agradeço de forma especial a Emerson Torquato, que teve a coragem de aceitar o desafio de fazer parte de uma pesquisa ampla e com diversos obstáculos à frente e que extrapolariam, em muito, suas obrigações acadêmicas para a finalização de seu curso de graduação. Com muita competência e dedicação, Emerson Torquato desenvolveu o cerne do módulo de acompanhamento da programação que permitiu levar à frente importantes testes e experimentos da pesquisa de doutorado.

Agradeço aos diretores do PPGIA, Mauro Sérgio Pereira Fonseca e Fabrício Enembreck, à secretária Cheila Cristina Farias, ao suporte fornecido pelo Jhonatan Jeremias e Anderson Bertling e aos professores do Programa de Pós-Graduação em Informática da PUCPR: Júlio Cesar Nievola, Bráulio Coelho Ávila, Edson Emílio Scalabrim, Alessandro Lameira Koerich, Sheila Reinher, Edgard Jamhour, Jacques Facon, Manoel Camillo de Oliveira Penna Neto, Alcides Calssavara, Altair Olivo Santin, Luiz Augusto de Paulo Lima Jr., Alceu de Souza

Britto Jr., Cinthia Obladen de Alamendra Freitas, Edson José Rodrigues Justino. Recebi importantes contribuições, nas mais diversas formas no decorrer do tempo que estive vinculado ao curso, então meu muitíssimo obrigado por mostrarem os caminhos e partilhar o conhecimento anteriormente adquirido.

Aos membros da Banca Examinadora da qualificação e da tese de doutorado: Silva Regina Vergílio, Edson Emilio Scalabrin, Milton Pires Ramos, Andreia Malucelli, Julio Cesar Nievola e meu orientador, Emerson Cabrera Paraiso, pelas sugestões para a melhoria da tese defendida.

Ao meu pai Evaldo e à minha mãe Odete, pela criação e formação que foi possibilitada, e todo o apoio e torcida, durante não só o tempo do doutorado, mas desde os anos iniciais, quando aos sete anos tive o primeiro contato com a escola. Terei uma gratidão sem fim por vocês terem propiciado as condições de seguir por este caminho.

Agradeço também, de forma especial a todos os familiares que durante os anos de estudo torceram e acompanharam minha caminhada, que estiveram presentes, incentivando e rezando pelo meu sucesso acadêmico: Muito, muito obrigado a todos!

Minha família merece agradecimentos eternos pela compreensão da dedicação que tive que realizar ao curso, e que implicaram em muitas ausências do convívio com todos. Thatiana, minha companheira, e Edenilson, Arthur e Erica; meus filhos e presentes de Deus para minha vida: devo a vocês minha gratidão pela compreensão das horas ausente e do tempo que tive que dedicar a esta qualificação profissional. Vocês são os meus maiores orgulhos nesta vida!

Minha estadia e permanência em Curitiba por um tempo só foi possível pela generosidade de duas famílias que forneceram, além da logística, um porto seguro de convivência e tranquilidade: meus agradecimentos a família Kurata: Albert, Rita, Kaory, Maryna e Maryana, e também a família Kummer: Ivo, Nilda e Thais. O apoio durante a fase de viagens que vivi por foi essencial para poder chegar até o final.

Meus agradecimentos ao orientador desta tese, professor Emerson Cabrera Paraíso, por ter acreditado que um aluno com graduação em tecnologia, portanto não engenheiro, pudesse levar a termo um tema de pesquisa de doutorado

até o final. Foram várias dificuldades, que procurei compensar com dedicação e esforço durante todo o período que estivesse sob sua orientação, e acredito que tenha honrado sua escolha, atingindo os objetivos que foram traçados. Novamente meus agradecimentos pela oportunidade e distinção de ter sido seu orientado.

Agradeço também a CAPES e Fundação Araucária, pela bolsa de estudo concedida durante parte do período do curso, e à PUCPR pela oportunidade de fazer parte desta instituição de ensino. Meus agradecimentos também a UTFPR, pela concessão do afastamento das atividades docentes, o qual foi inteiramente dedicado à conclusão dos estudos no Doutorado.

A todos que me auxiliaram de alguma maneira, direta ou indiretamente, para que eu chegasse aqui, meu muito obrigado! Fica aqui o meu agradecimento, com a certeza de que o tempo permitirá retribuir, de forma exponencial, todo auxílio e ajuda prestada.

Lista de Figuras

Lista de Quadros

Lista de Gráficos

Lista de Abreviaturas e Siglas

3C	Comunicação, Coordenação, Cooperação
3D	Três Dimensões
ACM	*Association for Computing Machinery*
ADS	Ambientes de Desenvolvimento de Software
API	*Application Programming Interface*
CCWM	*CoSpaces Collaborative Working Model*
CE-SC	Comissão Especial de Sistemas Colaborativos
CMMI	*Capability Maturity Model Integration*
CQMM	*Code Quality Monitoring Method*
CSCL	*Computer-supported Collaborative Learning*
CSCW	*Computer-supported Cooperative Work*
CVS	*Subversion*
DeSS	Desktop Semântico Social
DS	Desktop Semântico
ES	Engenharia de Software
FOAF	*Friend-of-a-Friend*
IDE	*Integrated Development Environment*
IE	Informática na Educação
IEEE	*Institute of Electrical and Electronics Engineers*
IHC	Interação Humano Computador
IMAP	*Internet Message Access Protocol*
ISO	*International Organization for Standardization*
KAP	*Knowledge-Added Process*
KDE	*K Desktop Environment*
LDAP	*Lightweight Directory Access Protocol*
MIT	*Massachusetts Institute of Technology*
NEPOMUK	*Networked Environment for Personalized, Ontology-based Management of Unified Knowledge*
O.C.D.E.	Organização para a Cooperação e Desenvolvimento Econômico
OWL	*Web Ontology Language*

P2P	*Peer-to-Peer*
PCES	Plataforma Colaborativa de Engenharia de Software
PIM	*Personal Information Management*
PKM	*Personal Knowledge Management*
PSP	*Personal Software Process*
RDF	*Resource Description Framework*
RSL	Revisões Sistemáticas Da Literatura
RUP	*Rational Unified Process*
SBC	Sociedade Brasileira de Computação
SOA	*Service-Oriented Architecture*
SVN	Subversion
SWEBOK	*Software Engineering Body of Knowledge .*
TCP/IP	*Transmission Control Protocol/Internet Protocol*
UML	*Unified Modeling Language*
V&V	Verificação e Validação de Software
W3C	*World Wide Web Consortium*
WS	*Workspace* Semântico
WWW	*World Wide Web*

SUMÁRIO

Capítulo 1

Introdução

O desenvolvimento de software é uma atividade essencialmente colaborativa, dependente de tecnologia e realizada por grupos de pessoas. A tecnologia de software envolvida é um fator importante, à medida que ela fornece um conjunto de ferramentas para o desenvolvimento do trabalho e é determinante para que se obtenham mais e melhores formas de como produzir software. Esta atividade geralmente envolve pessoas com diferentes papéis: gerentes, arquitetos, desenvolvedores, engenheiros de requisitos, testadores e os clientes também são considerados. Estes profissionais trabalham com diferentes tipos de artefatos em diferentes tarefas, muitas das quais interdependentes. Para fins de definição, nesta pesquisa "artefato" é todo documento produzido durante o processo de criação de software, tais como: documentos de texto, planilhas, diagramas, requisitos do software, modelos, código-fonte, conjuntos de teste, dentre outros.

Estas atividades e pessoas envolvidas para desenvolver software, seguem os preceitos da engenharia de software (ES), que buscam melhores formas de produzir software com qualidade. Uma das primeiras definições para a ES encontrada na literatura foi dada por Friedrich Ludwig Bauer (1969, *apud* (SOMMERVILLE, 2010, pg. 5)) na Conferência *NATO Science Committee*, na qual Bauer define a ES como sendo "*o estabelecimento e uso de sólidos princípios de engenharia para que se possa obter economicamente um software que seja confiável e que funcione eficientemente em máquinas reais"* (SOMMERVILLE, 2010 pg. 6). Sommerville, neste seu livro sobre engenharia de software, também fornece uma definição

contemporânea sobre o tema, considerando a ES como uma disciplina da engenharia que se preocupa com todos os aspectos da produção de software desde as fases iniciais de especificação do sistema até a manutenção do sistema após a sua entrada em uso.

1.1. Motivação

A ES é posta em prática por meio de pessoas, consideradas como um importante componente dentro da ES. Pessoas criativas e com conhecimento trabalham colaborando entre si, e a efetiva colaboração requer que os intervenientes (atores do processo) coordenem suas tarefas. Por exemplo, em uma equipe, os desenvolvedores devem decidir antecipadamente quem trabalhará em uma solicitação de mudança de um sistema. Além disso, a equipe tem que prever como as mudanças resultantes influenciam no trabalho dos outros para garantir a consistência do sistema. Finalmente, artefatos de software ou tarefas de desenvolvimento são raramente independentes e necessitam de uma coordenação para serem executados (WHITEHEAD, 2007). Neste contexto, o número de participantes do projeto colaborativo impõe a necessidade de recursos especialmente projetados para permitir que o grau de colaboração seja maximizado. Grandes equipes (dezenas de participantes) têm sido beneficiadas com recursos provenientes, por exemplo, das pesquisas realizadas em Trabalho Colaborativo Suportado por Computador (CSCW - *Computer Supported Cooperative Work* do inglês) ((COOK e CHURCHER, 2005), (STOREY *et al.,* 2006), (JIANG *et al.,* 2006), (SARMA *et al.,* 2003)).

Equipes de pequeno porte, porém, têm sido negligenciadas pelos pesquisadores (CAMPAGNOLO *et al.,* 2009). Tais equipes possuem requisitos específicos que não têm sido levados em consideração em pesquisas recentes. Por exemplo, equipes de pequeno porte enfrentam problemas para utilizar e manter informações de projetos em andamento ou anteriores. Além disto, em geral, pequena equipe é sinônimo de pequeno orçamento, o que dificulta ainda mais o acesso a ferramentas adequadas. O número de componentes de uma pequena

equipe pode ser controverso. Para Pollice e colegas (POLLICE *et al.,* 2004) pequenas equipes possuem até 10 participantes. Em pequenas equipes, como será visto em detalhes no Capítulo 2, é comum que um participante assuma diferentes papéis, muitas vezes de forma simultânea, durante um mesmo projeto. Além disto, pequenas equipes têm dificuldades em gerir os artefatos criados ao longo do desenvolvimento do software. Normalmente os desenvolvedores são maioria, o que os coloca como elementos chave dentro do projeto. Por isto, neste trabalho, será dado enfoque neste papel.

Desta forma, esta pesquisa visa a propor um ambiente de apoio para pequenas equipes que desenvolvem software. Tal ambiente foi nomeado como Workspace Semântico (WS), que tem como principal meta ligar semanticamente os artefatos produzidos durante o projeto, bem como aos participantes deste projeto para que a visibilidade do que está sendo manipulado seja aprimorada. A ligação semântica entre os artefatos (código-fonte, diagrama de modelagem, planilhas de cálculo, documentos de textos e outros) é um meio potencial de apoiar a integração de informações e, por conseguinte, a execução de projetos de software de forma otimizada. Um exemplo de ligação semântica que o WS é capaz de criar surge a partir da programação do código-fonte escrita por um desenvolvedor em sua IDE (*Integrated Development Environment*). Quando este, por exemplo, criar uma determinada classe definida em um Diagrama de Classes da UML (*Unified Modeling Language)*, o WS fará a ligação desta com o desenvolvedor. A partir desta ligação é possível listar outras informações de contexto, como data, hora e local da criação. Com os artefatos de um projeto disponibilizados em um ambiente apropriado, acredita-se que a visibilidade dos mesmos (e do que está sendo desenvolvido) tende a aumentar, levando, consequentemente, a facilitação da colaboração.

Neste sentido, para aumentar a visibilidade do que está ocorrendo em tempo real no desenvolvimento de software, o WS provê as condições para que dois conceitos importantes da ES aconteçam: o *awareness* e a verificação de software. A*wareness* pode ser definido como sendo o estado ou habilidade de perceber o entorno durante o trabalho para obter conhecimento e estar ciente daquilo que pode impactar no desenvolvimento das suas tarefas em relação ao restante do grupo (SOHLENKAMP, 1998). Já a Verificação de Software (parte da Verificação e

Validação - (V&V)) tem o objetivo de verificar se o que foi planejado realmente foi realizado, permitindo avaliar se os requisitos definidos estão sendo corretamente criados, atendendo às especificações determinadas, e ainda prever falhas ou inconsistências entre requisitos (WALLACE e FUJII, 1989). Geralmente a verificação está atrelada a questão "*o software está sendo construído corretamente?*". Estes dois conceitos (*awareness* e verificação de software) são úteis no WS para atingir o compromisso organizacional com a qualidade, conforme apontado em (SOMMERVILLE, 2011, pg. 41 e 109). Como consequência da utilização do WS, espera-se que ocorra diminuição da disparidade entre a especificação realizada na modelagem e a efetiva implantação do software. Espera-se que este processo seja aprimorado, pois o WS permite realizar o acompanhamento da modelagem com a codificação, principalmente por meio das ligações semânticas dos programas sendo desenvolvidos (código-fonte), com a modelagem UML (diagrama de classes do sistema), que será realizada por meio do conteúdo destes dois artefatos.

A importância desta ligação entre o que foi modelado e o que está de fato sendo codificado pode contribuir para que problemas potenciais deixem de ocorrer, como por exemplo: os participantes do projeto responsáveis pela programação optarem por não seguir os modelos definidos ou até mesmo não fornecer o feedback necessário aos responsáveis pela codificação. Isto gera inconsistências ou retrabalho durante a fase de desenvolvimento do sistema sendo criado, e não raro tem levado projetos ao fracasso. Podem também existir dificuldades em verificar a exatidão do que foi modelado, ou então a sincronização dos modelos com o código-fonte pode ser negligenciada pela equipe de desenvolvimento e gerar erros de construção e de arquitetura, que podem ser descobertos somente em fases posteriores do projeto (YATAKE e KATAYAMA, 2008). O trabalho de (CHARFI *et al.*, 2012, pg. 171), que busca melhores formas de produzir software por meio da geração de código por intermédio de um modelo de compilação mais eficiente dos modelos da UML, também mostra esta dificuldade e a lacuna que existe entre o que é modelado e o que é efetivamente codificado: "*Apesar deste esforço de padronização e do grande uso da UML na indústria, os desenvolvedores ainda devem ajustar a mão o código gerado a partir de modelos para corrigir, melhorar ou otimizá-lo. Isso resulta em uma lacuna entre o modelo e o código gerado. O ajuste*

manual do código, com a exceção de ser propenso a erros, pode invalidar toda a análise já realizada no modelo".

Todos estes problemas podem afetar diretamente a qualidade do software sendo desenvolvido, trazendo diversos prejuízos, mas que podem ser minimizados, desde que detectados e retificados antes mesmo que tomem proporções que sejam difíceis de serem corrigidas. Em (YATAKE e KATAYAMA, 2010, pg. 515), encontra-se mais uma análise do tema, que trata da importância da verificação de software na fase de análise do sistema: "*Como a nossa sociedade tem se tornado mais dependente dos sistemas de informação, tornou-se mais importante garantir a exatidão e a validade desses sistemas. Especificamente, existe uma necessidade crescente para a verificação no nível de análise do desenvolvimento desde que a escala de sistemas torna-se grande e os erros encontrados na fase de codificação têm levado a uma perda fatal para os desenvolvedores. A verificação nos níveis de análise permitem a detecção precoce de erros, e como consequência, reduz o custo total de desenvolvimento"*.

Manter os modelos UML atualizados e os códigos sincronizados durante todo processo de desenvolvimento de software é uma atividade que merece atenção e pode ser propensa a muitas dificuldades, conforme denotado na pesquisa de (BURDEN *et al.,* 2011), onde uma extensão da UML chamada de xtUML foi testada em 50 grupos formados por alunos de um curso de graduação em Computação, que por um período de dois anos realizaram tarefas envolvendo o teste do modelo proposto pelo autores. Mesmo no meio acadêmico, as dificuldades enfrentadas por equipes formadas por profissionais reproduziram-se.

As adversidades relatadas servem como motivação para aprimorar a área do desenvolvimento de software. Assim, o WS como um ambiente que favoreça a visibilidade dos artefatos e beneficie o *awareness* e verificação de software, pretende auxiliar a diminuição da disparidade entre o que é modelado e o que realmente é executado pelas equipes que desenvolvem software.

Além destes trabalhos, outro fator motivacional para a proposição desta pesquisa é o fato de que, até onde alcança o conhecimento, este ambiente é um novo conceito que não encontra semelhantes na averiguação realizada para a

condução desta pesquisa, e que tem potencial para ser uma aplicação prática e de utilidade para a comunidade acadêmica e profissional.

A literatura apresenta trabalhos que se propõem a facilitar a utilização do computador em ambientes profissionais. Um destes trabalhos, desenvolvido por (GEROSA, 2006), apresenta o estudo e a forma como foi implementado um sistema multi-agentes para o auxílio à formação de grupos para o ambiente AulaNet[1] (LUCENA *et al.,* 1998). Seu trabalho visa a instrumentar o desenvolvedor de *groupware* com componentes concebidos baseados no modelo 3C[2] de colaboração (ELLIS *et al.,* 1991), no qual parte-se do levantamento de requisitos e da análise do domínio e selecionam-se os componentes necessários para oferecer suporte computacional para a colaboração que se pretende obter por meio do AulaNet.

Em trabalho apresentado por Sarmiento e Colazos (SARMIENTO e COLLAZOS*,* 2012), um framework geral para projetar e desenvolver sistemas de CSCW em ambientes virtuais 3D é discutido, ambientes estes que integram novas metodologias e fornecem técnicas formais para desenvolver tais sistemas.

Outro trabalho é o apresentado em (YUYAN *et al.,* 2011), onde os autores constroem uma rede de comunicação baseada em salas de aula para que os usuários comuniquem-se uns com os outros e possam editar documentos colaborativamente, baseado nos conceitos do CSCW. Trata-se do *Online Collaborative Editor System*, e cria um grupo de aprendizagem colaborativa, facilitando a comunicação, discussão e troca de experiências entre os participantes do processo (alunos, professores e tutores). Apesar da proposta dos autores não atuar na esfera de um ambiente profissional, as características colaborativas do trabalho assemelham-se as do CSCW, e por conseguinte, assemelham-se as características do desenvolvimento de software colaborativo.

---

[1] **AulaNet é um ambiente de aprendizado baseado na Web para administração, criação, manutenção e** assistência de cursos a distância. O ambiente foi desenvolvido no Laboratório de Engenharia de Software (LES) do Departamento de Informática da Pontifícia Universidade Católica do Rio de Janeiro (PUC/RJ).

[2] O modelo 3C de colaboração é oriundo do artigo seminal de (ELLIS *et al.,* 1991), e analisa a colaboração em três dimensões: 1.) Comunicação (troca de mensagens, pela argumentação e pela negociação entre os indivíduos); 2.) Coordenação (gerenciamento dos indivíduos, atividades e recursos); 3.) Cooperação (atuação conjunta no espaço compartilhado para a produção de objetos ou informações).

Outra área de estudo que tem fornecido meios para aprimorar a colaboração de pessoas que trabalham em grupo para desenvolver software é a área da Interação Humano Computador, e um dos seus componentes, objeto de avaliação do trabalho aqui apresentado, é o Desktop Semântico (DS). Pequenas ou grandes equipes de desenvolvimento de software produzem diversos artefatos. O número de artefatos está diretamente relacionado à complexidade do software produzido. Ferramentas específicas de gestão de documentos (os artefatos) são necessárias para geri-los. O Desktop Semântico (DS) é um ambiente de suporte à gestão de documentos armazenados em um computador (DECKER e FRANK, 2004a) que auxilia na centralização de todos estes artefatos. Os DSs são uma proposição fundamentada nos padrões da Web Semântica, para ajudar a solucionar o problema cada vez maior de sobrecarga de informação na área de trabalho dos usuários, fornecendo as bases necessárias para integrar e gerenciar informações pessoais que normalmente encontramos em nossos computadores que é comumente conhecida como “PIM” (*Personal Information Management* - (TEEVAN *et al.,* 2006)). O objetivo do DS é permitir ao usuário realizar seu trabalho, deixando os dados e informações do desktop do seu computador disponíveis, ligando-os semanticamente. Como será mostrado no Capítulo 2, o DS tem algumas limitações quando aplicado em um ambiente de desenvolvimento de software, sendo a principal delas não gerir semanticamente partes do código-fonte produzido na programação. Apesar das limitações, o DS pode ser utilizado como ferramenta de apoio à gestão de artefatos produzidos ao longo de um projeto de desenvolvimento de software.

Um efeito colateral positivo, que é visto como uma motivação adicional para a criação do WS é a possibilidade de diminuir a disparidade entre a especificação realizada na modelagem e a efetiva implantação do software. Esta disparidade tem levado muitos sistemas a sofrerem manutenções constantes, o que acarreta maiores custos ao projeto: mais tempo despendido, mais recursos alocados e mesmo custos financeiros mais elevados (AL-BADAREEN *et al.,* 2011). Com o acompanhamento da escrita do código-fonte, comparando-o com a modelagem realizada anteriormente em tempo real, existe uma tendência em diminuir o número de manutenções futuras. Isto permitiria também auxiliar na diminuição de custos e de

utilização de recursos exigidos com a manutenção de software. Segundo pesquisadores como AL-Badareen e colegas (AL-BADAREEN *et al.,* 2011) as manutenções de software alcançam taxas de 90% do custo total do ciclo de vida do desenvolvimento de sistemas. Assim, o WS pode contribuir para diminuir esta taxa, justamente por tornar o software que está sendo criado mais visível aos participantes do processo de desenvolvimento, permitindo melhorar a interação entre os componentes da equipe. Esta possibilidade poderia ser verificada em avaliações de longo termo, que incorporassem todo o ciclo de vida de desenvolvimento de software, e assim agregassem a fase de manutenções. Devido ao tempo relativamente longo que esta atividade levaria para ser conduzida, esta possibilidade não será incorporada nesta pesquisa, mas estuda-se sua viabilidade em trabalhos futuros.

1.2. Objetivo da Pesquisa

O principal objetivo desta pesquisa é desenvolver um ambiente de apoio ao desenvolvimento de software, capaz de integrar semanticamente os artefatos produzidos ao longo do projeto entre si e aos seus participantes por meio das ligações semânticas entre estes componentes.

Como objetivos específicos, pode-se citar:

- Estudar a viabilidade da utilização dos Desktops Semânticos, no contexto do desenvolvimento colaborativo de software por pequenas equipes;
- Desenvolver um protótipo do ambiente de desenvolvimento de software, denominado Workspace Semântico;
- Demonstrar a criação das ligações semânticas entre os artefatos e os desenvolvedores de software, a fim de propiciar condições para apoiar a atividade de verificação de software em tempo real de codificação;
- Avaliar o conceito e o protótipo do Workspace Semântico.

Com estes objetivos fixados, espera-se contribuir com a colaboração entre os participantes de projetos em pequenas equipes, principalmente por meio da melhora

da visibilidade dos artefatos manipulados e também da ocorrência *awareness* e da verificação de software.

1.3. Hipóteses de Trabalho

A principal hipótese de trabalho pode ser formulada como: "*A utilização do Workspace Semântico contribuirá na integração entre os desenvolvedores e os artefatos de projetos desenvolvidos por pequenas equipes de desenvolvimento de software*". Acredita-se que com esta integração dos artefatos, a colaboração entre as pessoas envolvidas no processo pode ser beneficiada positivamente. Outra hipótese que se deseja comprovar é definida como: "*O acompanhamento da escrita do código-fonte em tempo real e de forma automática, sendo confrontada com a modelagem realizada anteriormente, amplia a visibilidade do que está sendo realizado*". A comprovação desta hipótese impactará na gestão dos artefatos que são manipulados durante o desenvolvimento de software, propiciando condições de aprimorar. Com estas duas hipóteses definidas, pretende-se confirmá-las por meio dos experimentos de verificação que foram conduzidos para avaliar a efetividade do WS e sua utilidade no desenvolvimento de software.

1.4. Contribuições Científicas e Aplicativas

As contribuições deste trabalho de pesquisa estão nos eixos científicos e aplicativos e podem ser descritas da seguinte forma:

- No eixo científico, a contribuição deste trabalho é a proposta de um novo conceito: o Workspace Semântico. Por meio de um estudo teórico e da especificação, projeto e construção de um protótipo computacional, pretende-se demonstrar a viabilidade técnica desta solução como elemento de contribuição ao melhoramento da visibilidade dos artefatos, aprimorando o *awareness e* a verificação de software em pequenas equipes de desenvolvimento de software.

- A contribuição aplicativa se dará pela criação do protótipo do ambiente do Workspace Semântico e posterior disponibilização do mesmo para utilização pela comunidade interessada.

1.5. Caracterização e Escopo da Pesquisa

Caracteriza-se a pesquisa proposta neste documento como sendo de desenvolvimento e exploratória, pois se pretende melhorar um procedimento já existente (desenvolvimento de software) utilizando o WS, e também explorar formas de aperfeiçoar a visibilidade do artefatos, aprimorando o *awareness* e a verificação de software nas equipes de pequeno porte que desenvolvem software, por meio do novo ambiente proposto. Segundo a O.C.D.E. (1980), *apud* (CONTANDRIOPOULOS *et al.* 1999, pg. 41), uma pesquisa de desenvolvimento é definida da seguinte forma: "*A pesquisa de desenvolvimento é a estratégia de pesquisa que visa, utilizando de maneira sistemática os conhecimentos existentes, elaborar uma nova intervenção ou melhorar consideravelmente uma intervenção existente ou, ainda, elaborar um instrumento, um dispositivo ou método de medição".* Esta pesquisa também possui características exploratórias, pois pretende também proporcionar familiaridade com o problema, explicitando-o. Segundo (GIL, 2002), uma pesquisa exploratória "*têm como objetivo proporcionar maior familiaridade com o problema, com vistas a torná-lo mais explícito ou a constituir hipóteses. Pode-se dizer que estas pesquisas têm como objetivo principal o aprimoramento de ideias ou a descoberta de intuições. Seu planejamento é, portanto, bastante flexível, de modo que possibilite a consideração dos mais variados aspectos relativos ao fato estudado*". Assim estas duas vertentes caracterizam primordialmente a pesquisa aqui realizada.

Quanto ao escopo deste trabalho, identificam-se três pontos que definem a esfera de abrangência desta pesquisa. O primeiro diz respeito ao tamanho da equipe: utilizar pequenas equipes de desenvolvimento de software como objeto de estudo para aprimorar a forma como estas equipes desenvolvem software. Esta delimitação facilitará também o processo de testes que o WS necessitará para

comprovar sua viabilidade, tanto conceitual como tecnicamente. Mesmo com a definição deste escopo, mais experimentos poderão ser conduzidos em trabalhos futuros a fim de verificar se o WS será também extensível para grandes equipes ou mesmo equipes distribuídas geograficamente que desenvolvem software de forma colaborativa.

Um segundo ponto importante do escopo diz respeito ao enfoque no papel do desenvolvedor. Existem outros papéis que os participantes de um projeto colaborativo de software podem realizar, porém o papel do desenvolvedor é o que concentra os artefatos de maior interesse ao desenvolvimento de software, por produzirem dados necessários e essenciais durante a codificação do projeto. Como o enfoque será dado neste papel, e apesar de por definição, artefato englobar todos os documentos produzidos durante o desenvolvimento de um software, nesta pesquisa será dado ênfase ao código-fonte produzido pelas equipes de programação.

O terceiro ponto é de característica técnica, referenciando-se principalmente a linguagem de programação, ao ambiente integrado para desenvolvimento de software e a ferramenta de análise e projeto de software que foram utilizados nesta pesquisa. Esta limitação é essencial, devido à impossibilidade de alocação de tempo e recursos que seriam necessários para criar um ambiente que aceitasse múltiplos mecanismos. Assim, a linguagem JAVA, a IDE ECLIPSE a ferramenta ASTAH COMMUNITY foram definidas como sendo os componentes que serão utilizados nesta pesquisa, por serem livres de licença e fornecerem os recursos que serão exigidos para a execução do trabalho.

1.6. Estrutura da Tese

O trabalho está organizado em seis capítulos, sendo este o primeiro. Ele contém as considerações iniciais, os objetivos, as hipóteses que se deseja comprovar, as contribuições esperadas e o escopo onde a pesquisa irá ser conduzida. O restante do documento está organizado da seguinte forma:

O capítulo 2 apresenta a fundamentação teórica do trabalho, que inclui: a distinção entre colaboração e cooperação, as pequenas equipes de

desenvolvimento de software, um estudo sobre colaboração no desenvolvimento de software, *awareness*, V&V, formas de gerenciamento do PIM, o Desktop Semântico e o Desktop Semântico Social. Todos estes temas foram tratados e possuem um detalhamento no capítulo apresentado.

O capítulo 3 apresenta o estado da arte envolvendo as pesquisas com os ambientes e/ou processos de desenvolvimento de software, bem como as ferramentas de apoio e suporte a esta atividade e também a definição tipo da pesquisa. Algumas áreas são investigadas: a gestão dos artefatos, as aplicações do DS, a análise de código-fonte e ferramentas de apoio ao *awareness* e a V&V são apresentadas. Optou-se por separar a fundamentação teórica (Capítulo 2) do estado arte (Capítulo 3), por considerar que no segundo capítulo encontra-se a teoria dos conceitos utilizados, e no terceiro capítulo as aplicações que envolvem estes conceitos.

No capítulo 4, a proposta do WS é apresentada. Um levantamento de dados, realizado por meio do emprego de um questionário com pessoas envolvidas no desenvolvimento de software e também é mostrado, e uma análise das respostas foi realizada e é também apresentada. As características e a visão geral do conceito do WS, seu funcionamento, as melhorias esperadas com a conclusão da pesquisa também são apresentadas. Como forma de comparativo, algumas ferramentas que apresentam correlação com o conceito proposto são expostas.

No capítulo 5, explana-se sobre os experimentos conduzidos para avaliar o WS. Um experimento piloto, conduzido com acadêmicos da PUCPR foi executado nos laboratórios da instituição e serviu para validar os conceitos almejados para o WS. Outro experimento, agora nas dependências do TECPAR também foi executado e permitiu avaliar as hipóteses levantadas nesta pesquisa. Ambos os experimentos são descritos e analisados neste capítulo.

Por fim, o capítulo 6 apresenta as conclusões obtidas com a pesquisa, e também as contribuições e relevância do trabalho executado. Apresenta algumas limitações e trabalhos futuros, que podem contribuir para aprimorar a pesquisa aqui relatada.

O trabalho também contém apêndices, num total de quatro. O Apêndice 1 mostra o questionário realizado na pesquisa de campo e o Apêndice 2 mostra o

resultado desta pesquisa, com o quantitativo das respostas, os percentuais e gráficos comparativos. O apêndice 3 traz a descrição do módulo OPERAM do WS e um possível cenário de utilização do mesmo. Finalizando este documento, o Apêndice 4 detalha os testes de avaliação do DeSS NEPOMUK.

Capítulo 2

Fundamentação Teórica

Neste capítulo são apresentados os fundamentos teóricos necessários à realização desta pesquisa.

Uma análise da área que versa sobre projetos colaborativos de desenvolvimento de software é realizada.

Estes projetos são desenvolvidos por equipes que necessitam colaborar para produzirem software com qualidade. Neste sentido, o interesse desta pesquisa é notadamente em pequenas equipes que realizam este trabalho, justamente por considerar que as mesmas carecem de um estudo específico, pois a maioria dos trabalhos é direcionada para grandes equipes distribuídas de desenvolvimento de software.

Estas equipes (grandes ou pequenas) utilizam ferramentas e processos definidos para desenvolverem software. Desta forma, a seção 2.3 deste documento realiza uma análise da colaboração no desenvolvimento de software, analisando-se o CSCW, área que vem contribuindo nas últimas décadas com a proposição de ferramentas e processos para a engenharia de software.

Dois conceitos importantes para a pesquisa aqui proposta também são analisados: o *awareness* e a atividade de verificação e validação de software. Ambos são detalhados nas seções 2.2 e 2.3, respectivamente.

Também considerando que as informações geradas pela manipulação de arquivos digitais de todo tipo, comumente conhecida como P.I.M. (*Personal Information Management)* está crescendo, e que novas formas de manipular toda esta informação também estão sendo objetos de estudo, por exemplo, por meio da criação dos Desktops Semânticos (DS) e dos Desktops Semânticos Sociais (DeSS), os dois temas são discutidos neste capítulo.

2.1. Pequenas Equipes de Desenvolvimento de Software

Pequenas equipes têm recebido menor atenção quando da disponibilização de ferramentas que dão suporte ao desenvolvimento de software (CAMPAGNOLO *et al.*, 2009). Tais equipes possuem requisitos específicos que não têm sido levados em consideração em pesquisas recentes. Consideram-se pequenas equipes aquelas com até 10 participantes, como definido por Pollice e colegas (POLLICE *et al.,* 2004). Além de quantificar o número de pessoas para uma pequena equipe, os autores também listam as características que consideram chave para um pequeno projeto:

- O nível de formalidade, que geralmente se correlaciona com o número de pessoas envolvidas no projeto: em um pequeno projeto, a formalidade é reduzida em virtude da equipe também ser reduzida;
- A complexidade do novo código, o qual parece correlacionar-se com o número de pessoas envolvidas: com mais pessoas envolvidas, o código tende a aumentar;
- O período de tempo que se deve trabalhar no projeto: pequenos projetos demandam menor tempo de execução.

Estas três características levantadas pelos autores puderam ser comprovadas na pesquisa realizada com desenvolvedores de software em pequenas equipes para levantamento de dados e coleta de subsídios para o trabalho proposto neste documento (Apêndice 1). Uma síntese desta análise dos dados coletados é apresentada no Capítulo 4.

Tais características, porém, podem não ser regras gerais: alguns projetos com poucas pessoas requerem um alto grau de formalidade (por exemplo, o

desenvolvimento de um sistema de suporte a vida (*home care*), que atende requisitos de órgãos reguladores governamentais e da indústria). Por outro lado também, alguns projetos podem ser executados com mais informalidade (POLLICE *et al.,* 2004). Mesmo com esta informalidade, as pequenas equipes de desenvolvimento de software enfrentam problemas para manter, rastrear e usar informações geradas em projetos que estão em desenvolvimento ou já desenvolvidos (informações estas dispersas em atas de reunião, relatórios de projeto, modelos, código fonte, dentre outros). Tais equipes possuem características particulares que devem ser consideradas no desenvolvimento de ferramentas computacionais que dão suporte à colaboração de seus participantes. As principais características destas equipes, segundo (CAMPAGNOLO *et al.,* 2009) são:

- Trabalho colocado: os participantes destas equipes geralmente trabalham no mesmo local físico e se comunicam preferencialmente face-a-face ao invés de utilizarem ferramentas computacionais: ou seja, a troca de mensagens eletrônicas envolvendo temas do projeto é reduzida e a comunicação face-a-face é mais presente;
- Acúmulo de papéis: os participantes possuem funções específicas assim como nas equipes grandes (ex. programadores, testadores, analistas), porém participam de forma simultânea de várias atividades do projeto, realizando desta forma, atividades que fogem do escopo original de suas funções (em momentos distintos do ciclo de vida do desenvolvimento do software);
- Sobrecarga de trabalho: como os participantes realizam muitas atividades e desempenham funções diversas, estão frequentemente sobrecarregados de trabalho, deixando em segundo plano atividades importantes, como por exemplo, a documentação do processo decisório;
- Gerenciamento de código fonte: a gestão do código fonte é importante e deve ser privilegiada, tanto no nível de desenvolvimento como no nível da gestão, possibilitando o acompanhamento do que foi modelado em relação ao que está sendo codificado;
- Reuso e manutenção: o reuso e manutenção de software é fundamental, sendo, portanto, de vital importância dar suporte à tarefa de documentação

do código produzido, e em especial para pequenas equipes de desenvolvimento de software, onde o papel desempenhado pelos participantes pode variar muito;

- Integração de ferramentas: os participantes utilizam diversas ferramentas disponíveis na Web, na maioria das vezes *open source* ou gratuitas, e gastam tempo em gerenciar este uso, por exemplo, para encontrar documentação ou mesmo executar várias operações no aplicativo até efetuar a operação desejada. Um desafio para uma pequena equipe é montar um conjunto adequado de ferramentas para realizar seu trabalho.

O trabalho de Campagnolo e colegas citado anteriormente, também define uma arquitetura para suporte de pequenas equipes colocadas que desenvolvem software de forma colaborativa. Pode-se definir como equipes colocadas, aquelas que trabalham no mesmo ambiente, ao mesmo tempo e próximas fisicamente. Nas considerações do trabalho, os autores observam que o uso intensivo de tecnologia e trabalho de grupo cria um ambiente propício para a aplicação do CSCW.

No entanto, a pesquisa em CSCW é mais focada em equipes distribuídas, que têm um número considerável de membros. Equipes pequenas e colocadas são geralmente negligenciadas nas pesquisas sobre o tema, apesar da área de CSCW ser caracterizada pela multidisciplinaridade. Esta abordagem multidisciplinar envolve tanto o aspecto social quanto tecnológico do trabalho em grupo. No aspecto tecnológico tem-se a elaboração, implementação e avaliação de *groupware*, que normalmente é referenciada como a tecnologia para o CSCW (GRUDIN, 1994), (RAMA e BISHOP, 2006), (WAINER e BARSOTTINI, 2007).

As ferramentas de *groupware* podem ser classificadas usando uma taxonomia baseada em duas dimensões: tempo e espaço. Usando esta taxonomia, ferramentas de *groupware* podem ser classificadas por atividades de apoio que são realizadas no mesmo local e ao mesmo tempo; no mesmo local e em tempos diferentes; em locais diferentes e ao mesmo tempo e em locais diferentes com tempos diferentes ((ELLIS *et al.*, 1991), (GRUDIN, 1994)). Embora útil, esta taxonomia induz à ideia de que um tipo específico de grupo está ligado a um tipo específico de circunstância de tempo/espaço e, portanto, a um conjunto específico de ferramentas (GRUDIN, 1994). Isto é especialmente verdadeiro para pequenas

equipes. Por exemplo, as atividades realizadas por uma pequena equipe, em que todos os participantes trabalham no mesmo local e ao mesmo tempo, podem ser entendidas como atividades síncronas e colocadas.

No contexto geral do *groupware*, os membros das equipes podem realizar algumas atividades de forma assíncrona, como por exemplo, a elaboração de um documento. Neste tipo de redação colaborativa de documento, cada membro contribui para a sua elaboração. Os membros da equipe não precisam se reunir para redigi-lo. Como algumas ferramentas são desenvolvidas para apoiar as atividades em uma circunstância de tempo/espaço específica, como por exemplo, estarem em cidades diferentes, estas ferramentas não se enquadram em todos os tipos de atividades que podem ser realizadas em contextos diferenciados. Por exemplo, algumas ferramentas de escrita colaborativa (tais como Equitext[3] - (ALONSO *et al.,* 2003)),ou mesmo o GOOGLEDOCS[4] supõem que seus usuários estão distribuídos geograficamente, e por isso são menos apropriadas para os membros de equipes colocadas. Normalmente, essas ferramentas fornecem meios de comunicação (por exemplo, bate-papo ou RSS) que não são tão úteis para os membros colocados como elas são para os membros distribuídos geograficamente.

Em (POLLICE *et al*., 2004), os autores descrevem os artefatos classificados como absolutamente necessários para o desenvolvimento de um projeto de software por uma equipe pequena. No exemplo citado pelos autores, esta pequena equipe utiliza o RUP[5] (*Rational Unified Process)* para desenvolver o projeto. Os documentos considerados essenciais pelos autores são:

- Documento de visão (auxilia a entender o projeto que se está construindo e posteriormente o que foi construído);
- Lista de riscos (riscos baseados em pessoas, processos e ferramentas);
- Caso de desenvolvimento (descreve como adaptar o RUP as necessidades da equipe);
- Casos de uso (descreve as interações entre os atores e o sistema);

---

[3] disponível em `http://equitext.pgie.ufrgs.br`
[4] `http://docs.google.com`
[5] disponível em `www.ibm.com/software/awdtools/rup/`

- Casos de teste (geração dos casos de testes mais completos possíveis para os casos de uso definidos na etapa posterior);
- Arquitetura (pode ser bastante informal e até mesmo gerada durante o desenvolvimento do projeto);
- Plano de projeto (deve indicar as iterações da equipe e a agenda de desenvolvimento);
- Glossário (deve conter as definições para manter a linguagem da equipe consistente).

Durante o processo de desenvolvimento, a equipe pode sentir a necessidade de criar artefatos essenciais informalmente. Por exemplo, um caso de teste pode ser simplesmente uma nota autocolante presa à parede. Isto pode ser considerado como “suficientemente bom" para as necessidades de uma equipe pequena. Com estas características mínimas e com documentos considerados essenciais por (POLLICE *et al*., 2004) para desenvolver software, as pequenas equipes tem ao seu dispor formas de ajustarem-se a um meio que é direcionado para grandes equipes que trabalham de forma distribuída. Estas pequenas equipes podem assim criar softwares de forma colaborativa e com qualidade necessária para satisfazerem as exigências de seus usuários, sem alocarem grandes recursos, tanto financeiros quanto de tecnologia ou logística. Neste sentido, o WS pode ser um componente importante para auxiliar estas equipes na produção de software, pois se comprovando a sua efetividade em aprimorar o acompanhamento do desenvolvimento, elas podem obter uma vantagem competitiva.

2.2. Awareness

O *awareness* é um dos conceitos importantes para ser observado quando se trabalha em grupo, e uma das traduções aceitas em português para o termo é percepção. Assim, *awareness* é a qualidade de estar perceptivo, vigilante ou atento, ou seja, estar percebendo tudo que está acontecendo dentro do contexto do trabalho sendo desenvolvido. Um dos problemas existentes no desenvolvimento de software é justamente a falta de contexto entre os participantes destes grupos, que ocorre

quando os membros não conhecem o que seus colegas estão produzindo, ou não têm noção onde suas atividades se encaixam no trabalho global, ou ainda, qual é a situação atual do andamento deste trabalho. Esse conhecimento do contexto é importante em trabalhos colaborativos, ultrapassando o conteúdo das contribuições individuais, mas extrapolando para o que este conhecimento significa para os participantes do grupo.

Desta forma, o fornecimento deste contexto aos participantes de um grupo é então chamado de *awareness*, cuja principal característica é fornecer o conhecimento das atividades que o grupo realiza, englobando além do estado atual do processo, as atividades já realizadas e opções futuras para todos os participantes do processo (SOHLENKAMP, 1998). Os pesquisadores Dourish e Bellotti definem *awareness* como sendo "*uma compreensão das atividades dos outros, que fornece um contexto para sua própria atividade*" (DOURISH e BELLOTTI, 1992, pg. 108) e (GUTWIN e GREENBER, 1996) complementam afirmando que o *awareness* depende da atividade em que um grupo de pessoas está participando, e geralmente isto inclui informações como quem faz parte do grupo, quais as tarefas que desempenham, quão ativos eles estão, quais mudanças eles fizeram e quais objetos foram manipulados.

Pesquisadores como (SARMA, *et al., 2003),* argumentam que o *awareness* entre desenvolvedores de software que estão envolvidos em atividades de programação é importante porque a desconexão dos espaços de trabalho entre eles tem o potencial de criar um isolamento prejudicial que pode ser desfeito com a utilização do *awareness* para resolver determinados conflitos, tais como dois programadores modificarem o mesmo artefato ao mesmo tempo.

Pesquisas como a realizada por (TREUDE e STOREY, 2011), citam que o *awareness* em projetos de desenvolvimento de software é composto de muitos aspectos diferentes: os desenvolvedores necessitam estar cientes da situação global dos seus projetos e prazos críticos. Eles precisam entender as atuais prioridades e gargalos no processo e precisam saber de dependências entre os componentes da equipe, como também precisam ser informados das alterações nas tarefas que estão trabalhando em tempo real. A pesquisa realizada em (TREUDE e STOREY, 2011) também buscou responder a dois questionamentos: como e porque as

ferramentas de *awareness* são usadas, e também o impacto delas nas práticas do desenvolvimento de software. Para atingir estes objetivos, os autores realizaram um estudo empírico de várias equipes de desenvolvimento que utilizaram *dashboards* e *feeds* em atividades do dia-a-dia em seu trabalho.

Ainda, outro trabalho de revisão de literatura sobre o conceito do *awareness* foi proposto por Belkadi e colegas (BELKADI *et al.,* 2012), no qual é apresentado um modelo genérico para apoiar o *awareness* em *design* colaborativo. A originalidade do modelo vem principalmente das quatro visões complementares (operacional, colaborativa, estrutural, gerencial) que organizam a classificação do *awareness* proposto pelos autores. Ao final deste trabalho, os autores explicitam que por meio do modelo proposto, será desenvolvido um sistema de CSCW focado em *awareness*.

Por fim, em (TALAEI-KHOEI *et al.,* 2012) encontra-se uma revisão sistemática da literatura da área de *awareness* que analisou artigos de 1970 até o ano de 2010. Além desta revisão, os autores também propõem um *framework* de classificação para *awareness*, categorizando-o em cinco diferentes grupos, que auxiliam a situar em cada caso qual tipo de *awareness* está sendo utilizado. Os grupos listados pelos autores são os seguintes: *Workspace Awareness, Common-sense Awareness, Group Awareness, Social Awareness e Context Awareness*. Esta categorização é útil quando se desenvolve software, pois é importante manter o conhecimento e compreensão do trabalho de cada um dos participantes e das atividades dos componentes da equipe, além do estado geral do andamento do projeto e tarefas que atualmente estão sendo desenvolvidas. Isto é um desafio, pois não se limita à utilização de métricas baseadas em código fonte ou a execução de um único item do projeto. Levando em consideração projetos complexos de desenvolvimento de software o *awareness* deve incluir, além dos aspectos técnicos, os aspectos sociais do desenvolvimento, o trabalho de articulação atual e também os trabalhos futuros que a equipe pode desenvolver.

2.3. Verificação e Validação de Software

A atividade de Verificação e validação (V&V) está inserida dentro do escopo da engenharia de software, e o seu objetivo é auxiliar na melhora da qualidade do desenvolvimento de sistemas, durante o ciclo de vida dos que permeia esta atividade. Processos de V&V fornecem uma avaliação objetiva dos produtos e processos ao longo do ciclo de vida do software, e esta avaliação demonstra se os requisitos estão corretos, completos, precisos, consistentes e testáveis (SARGENT, 2013).

Barry Boehm, pioneiro da engenharia de software, escreveu em seu artigo (Boehm, 1979), a definição da V&V utilizada até nos dias atuais para diferenciar estes dois conceitos. “Verificação: Estamos construindo certo o produto? Validação: Estamos construindo o produto certo?”. Em seu livro, (Sommerville, 2011) afirma que “Processos de verificação e validação estão preocupados com a verificação de que o software que está sendo desenvolvido cumpre a sua especificação e entrega a funcionalidade esperada pelas pessoas que pagam pelo software. Estes processos de verificação de começam assim que os requisitos tornam-se disponíveis e continuar através de todas as fases do processo de desenvolvimento.” Segundo o mesmo autor, a definição de V&V engloba muitas das atividades que são abrangidas pela Garantia da Qualidade de Software (QOS), como por exemplo: revisões técnicas formais, auditoria de qualidade e configuração, monitoramento de desempenho, estudo de viabilidade, revisão da documentação, entre outras. Os processos de V&V definidos em (Sommerville, 2011) também auxiliam a determinar se os produtos desenvolvidos em uma determinada atividade estão em conformidade com os requisitos dessa atividade, e se o produto satisfaz as necessidades de uso dos usuários. Este processo inclui a avaliação, análise, revisão, inspeção e teste de produtos e processos. Neste sentido, a V&V auxilia na melhora da qualidade de construção, desenvolvimento e organização do sistema em desenvolvimento, fornecendo uma avaliação objetiva dos produtos e processos ao longo do ciclo de vida. Esta avaliação demonstra se os requisitos estão corretos, completos, precisos, consistentes e testáveis.

A V&V emprega métodos para tentar identificar possíveis problemas e assim fornecer *feedback* sobre a qualidade e o desempenho do que está sendo testado.

Esse *feedback* é composto por resoluções de problemas, melhorias de desempenho e também de qualidade, não só para as condições de operação previstas, mas também em todo o espectro do sistema sendo avaliado. Os resultados deste *feedback* permitem modificar os produtos em tempo hábil, caso exista detecção de problemas e, assim, reduzir o impacto no projeto e cronograma.

Pelo exposto, os processos de V&V são divididos em dois processos distintos, mas complementares entre si: a verificação e a validação. O processo de verificação fornece condições para saber se o processo de desenvolvimento está em conformidade com os requisitos (por exemplo: integridade, consistência e precisão) para todas as atividades durante cada processo de ciclo de vida do software. Também realiza a inspeção das normas, práticas e convenções durante os processos de ciclo de vida estão cumprindo todos os critérios para iniciar sucessivas atividades do ciclo de vida (ou seja, construindo o produto corretamente). Já o processo de validação fornece evidências para saber se os produtos estão satisfazendo os requisitos do sistema definidos para os produtos até o final de cada atividade do ciclo de vida do sistema. Da mesma forma, constata se a resolução do problema está correta, por exemplo, modelar corretamente as restrições legais, implementar regras de negócio e utilizar os pressupostos do sistema apropriados.

Desta forma, a atividade de Verificação e Validação (V&V) são utilizadas para determinar se os produtos de uma determinada atividade estão em conformidade com os requisitos dessa atividade (Verificação), e se o produto satisfaz as necessidades e uso pretendido pelos usuários (Validação). Requisitos do processo do ciclo de vida de V&V são especificados para os diferentes níveis de integridade e o escopo dos processos de V&V abrange sistemas, software e hardware, e também inclui suas interfaces.

O IEEE, em seu mais recente documento sobre o assunto, especifica a V&V da seguinte forma: "*Processos de V&V determinam se os produtos de desenvolvimento de software de certa atividade estão em conformidade com os requisitos dessa atividade, e se os produtos satisfazem seu uso pretendido e as necessidades do usuário. Esta determinação pode incluir a análise, avaliação, revisão, inspeção, avaliação e testes de produtos e processos*" (IEEE, 2012, pg. 10).

As atividades de V&V também estão previstas no modelo CMMI (*Capability Maturity Model Integration*), no processo MPS-BR (Melhoria do Processo de Software Brasileiro) e também no guia SWEBOK *(Software Engineering Body of Knowledge)*.

O CMMI[6] é dividido em cinco níveis, os quais atestam o grau de evolução em que uma determinada organização se encontra. Tem por objetivo principal funcionar como um guia para a melhoria dos processos da organização, considerando diversas atividades, tais como: o gerenciamento do desenvolvimento de software, prazos e custos, todos previamente estabelecidos. O objetivo principal do CMMI é auxiliar na produção de software com melhor qualidade e menos propenso a erros. A V&V está ligada neste no modelo CMMI no item Qualidade de Software, subárea Processos de Gerência de Qualidade. Seu objetivo também é garantir que os requisitos de software atendam aos usuários, tentando assegurar que o produto é construído corretamente.

No MPS-BR[7] existem sete níveis, e a V&V está localizada no nível D (quarto nível de maturidade - Largamente Definido), e também é chamado Verificação, que objetiva "*Confirmar que cada serviço e/ou produto de trabalho do processo ou do projeto reflete apropriadamente os requisitos específicos*." Neste sentido, a verificação tem o objetivo de avaliar se o que foi planejado realmente foi realizado, isto é, se os requisitos e funcionalidades documentadas foram efetivamente implantados seguindo os requisitos especificados anteriormente.

No SWEBOK[8], em sua mais recente versão, que conta com 15 áreas de conhecimento, a V&V encontra-se descrita em Qualidade de Software, na subárea Processos de Gerência de Qualidade (SWEBOK, 2014) e mantém os mesmos objetivos do CMMI e da MPS-BR.

A importância da V&V é percebida nos itens citados anteriormente, que convergem os objetivos entre si. A importância da sua citação é justificada na pesquisa aqui proposta, como forma de obter melhorias no processo de

---

[6] Detalhes são encontrados em http://whatis.cmmiinstitute.com/

[7] O guia que contém a descrição geral do Modelo MPS para Software está disponível em http://www.softex.br/wp-content/uploads/2013/07/MPS.BR_Guia_Geral_Software_2012.pdf

[8] A versão revisada e atualiza está disponível em http://www.computer.org/portal/web/swebok/swebokv3

desenvolvimento de software, principalmente permitindo que a verificação de software seja ampliada, sob o ponto de vista dos desenvolvedores, justamente por mostrar o que está acontecendo com o código no momento em que ele está sendo desenvolvido, trazendo benefícios também para o ***awareness***.

2.4. Apoio a Gestão de Documentos Pessoais

Com a atual disseminação do acesso a informação digital, que alcança praticamente todas as esferas do conhecimento humano, tem-se notado também um acúmulo de dados nas máquinas dos usuários, em geral sob a forma de arquivos. Estes arquivos são armazenados em estruturas hierárquicas de diretórios (pastas), muitas vezes criados sem nenhum padrão de organização ou baseados em um modelo mental que o usuário concebe para armazenar em seus repositórios pessoais de informação.

O aprimoramento dos métodos de tratamento, utilização e recuperação da informação tem sido um tema de pesquisa de várias comunidades científicas, principalmente após a massificação da utilização do computador, que pesquisam o tratamento e recuperação eficiente da informação. Um tema inerente a estas comunidades é o que se conhece como P.I.M. (TEEVAN *et al.,* 2006), que se refere à prática e estudo das atividades que as pessoas desempenham, a fim de adquirir, organizar, manter, recuperar e utilizar itens de informação que normalmente usuários de computador manipulam no seu dia-a-dia, tais como documentos digitais, *e-mails, web pages*, contatos de redes sociais entre outros. No trabalho de Ramezani e Burns (RAMEZANI e BURNS, 2009), os autores sintetizam algumas definições do P.I.M., principalmente as que foram colhidas dos trabalhos de outros pesquisadores, como (BELLOTTI *et al.,* 2002), (BELOTTI *et al.,* 2004); (LANSDALE, 1988) e (BARREAU, 1995). Com os estudos dos autores citados, pesquisas têm sido conduzidas para aprimorar a forma como a informação digital é organizada, recuperada e utilizada. São alguns exemplos recentes que podem ser citados: a Web Semântica (BERNES-LEE *et al.,* 2001), o *Peer-to-Peer* (P2P) (CRUZ *et al.,* 2005), ou mais recentemente, o Desktop Semântico ((SAUERMANN *et al.,* 2005) e (SAUERMANN, 2009)). Este último, descrito em detalhes nos próximos parágrafos,

possibilita que os usuários sejam capazes de identificar relações semânticas implícitas entre diferentes fontes de documentos, e então abstrair conhecimento e informações relevantes.

2.4.1. O Desktop Semântico

A ideia precursora do Desktop Semântico (DS) foi a máquina imaginada por Vannevar Bush, o MEMEX (BUSH, 1945). Em seu trabalho, Bush descreve como a mente humana opera por associação (modelos mentais), e o "extensor de memória" que ele acabou nominando de MEMEX. O MEMEX poderia armazenar todos os documentos e livros que uma pessoa se depararia em sua vida, juntamente com as várias "trilhas associativas" (hoje conhecidas como *link* ou *hiperlinks*), que estes documentos poderiam ter. Seu artigo é um marco histórico e descreve, em maneiras semelhantes, a *World Wide Web* (WWW). Anos mais tarde, outro trabalho que também está anos a frente de seu tempo foi o realizado por Doug Engelbart no final dos anos sessenta sobre hipertextos, demonstrando o que poderia ser feito com organização da informação para "*aumentar a inteligência humana*" (ENGELBART, 1963, pg. 35). Estas ideias não dispunham da tecnologia necessária para serem implantadas quando seus autores a imaginaram décadas atrás, mas elas também serviram de base para o surgimento dos DSs. A primeira publicação que tratava sobre os DSs foi a dissertação de mestrado de Leo Sauermann (SAUERMANN, 2003). O termo "desktop semântico" cria um conceito mútuo para ideias que têm um entendimento semelhante.

Encontram-se na literatura diversas definições para o Desktop Semântico (DS). Como este conceito evoluiu por meio da junção de vários outros conceitos, é natural obter uma grande variedade de definições na literatura. Uma definição do DS, encontrada em (SAUERMANN *et al.*, 2005), fornece uma descrição abrangente do termo: *"Um Desktop Semântico é um dispositivo em que um indivíduo armazena toda sua informação digital como documentos, conteúdo multimídia e mensagens. Estes são interpretados como recursos Web-Semânticos, cada um sendo identificado por um Identificador de Recurso Universal (IRU) e todos os dados são acessíveis e consultáveis através de grafos RDF. O Identificador de Recurso*

*Universal pode ser armazenado e o conteúdo anotado pode ser compartilhado com o outros. Ontologias permitem que o usuário expresse modelos mentais pessoais e formem a cola semântica que interconecta a informação e os sistemas. As aplicações respeitam isto e armazenam, lêem e se comunicam através das ontologias e dos protocolos da Web Semântica. O Desktop Semântico é um suplemento ampliado à memória do usuário." (*SAUERMANN *et al.,* 2005, pg. 4).

Em (BREITMAN *et al.*, 2007, pg. 230), encontra-se outra definição para DS: trata-se do "*conjunto de métodos, estruturas de dados e ferramentas que ampliam a metáfora do desktop tradicional, dando aos dados um significado bem definido*". Na prática, os DSs podem ser entendidos como aplicações que combinam ontologias, taxonomias e metadados em geral, para melhorar a gestão de informações pessoais e o uso de aplicativos de software (OREN, 2006). Pesquisadores como Xiao e Cruz (XIAO e CRUZ, 2006) consideram que os DSs devem suportar pelo menos três características importantes: a organização dos dados de forma semântica, a manipulação flexível de dados e a visualização customizável e rica destes. Nesta definição, surge um eixo importante na conceitualização do DS: a Interação Humano-Computador, preocupada com a forma como os usuários podem interagir e organizar seus documentos. Estas características englobam uma série de fatores relevantes aos DSs, tais como: organização dos dados, semântica explícita, associações significativas e uma representação uniforme dos dados.

Como exemplo de utilização do DS, pode-se citar um documento anexado a um *e-mail*: os DSs podem armazenar as informações da pessoa que enviou o documento, a data do envio ou mesmo permitir ao usuário fazer anotações semânticas vinculando esta imagem ao assunto do *e-mail*, gerando metadados que irão facilitar posteriormente a indexação e procura deste documento por meio do DS. Outro exemplo de utilização dos DSs, é o citado no trabalho de Sauermanm e Schwarz (SAUERMANN e SCHWARZ, 2004), quando os autores apresentam sua visão de como uma aplicação web-semântica pode melhorar o ambiente desktop. Segundo os autores, isto é possível por meio do GNOWSIS[9], um DS desenvolvido pelo primeiro autor em sua dissertação de mestrado (SAUERMANN, 2003),

[9] http://www.gnowsis.com/about/

elaborada na Universidade de Tecnologia de Viena. Na exemplificação citada pelos autores, o GNOWSIS fornece informações para um caso típico de uso: um usuário deseja saber mais informações sobre um arquivo de música. No caso em questão, o pedido é enviado para o servidor GNOWSIS local. Nele, uma central de banco de dados RDF é consultada para identificar recursos relacionados ou anotações adicionais (tais como "*Eu recebi este arquivo de Peter - uma foaf:Person - do meu catálogo de endereços*"). O resultado dessa consulta é integrado e permanece acessível por meio de um modelo *Jena*[10]. Finalmente, o servidor inicia um módulo de interface com o usuário, (o navegador semântico), para visualizar o resultado. No navegador, o usuário pode visualizar informações sobre o recurso e manipulá-lo. Clicando no recurso, o servidor irá abrir o arquivo usando o aplicativo padrão para a música. Um segundo recurso pode ser selecionado em outro aplicativo (como um contato em um gerenciador de *e-mail*) e ligado ao recurso em uso no momento. (SAUERMANN e SCHWARZ, 2004).

O GNOWSIS (SAUERMANN, 2005) é um *spin-off* do projeto NEPOMUK (BERNARDI *et al.,* 2011). Foi a primeira ferramenta a ser disponibilizada pelo consórcio NEPOMUK que conjugava todas as facilidades de um DS. O GNOWSIS acrescenta itens da Web Semântica em aplicações desktop comuns (SAUERMANN e SCHWARZ, 2004), permitindo aos usuários utilizar computadores desktop como um site pessoal. A vinculação de documentos entre aplicativos e navegação por meio das informações é uma possibilidade da ferramenta. *E-mails*, documentos, endereços, fotos, compromissos que foram distribuídos aleatoriamente podem ser interligados, criando uma rede pessoal semântica. As estruturas de dados não são alteradas e as aplicações existentes são estendidas para utilizar a ferramenta e não são substituídas, permanecendo inalteradas durante o processo. Na Figura 1 é apresentada a imagem do GNOWSIS em utilização.

[10] Jena é um framework Java para construir aplicações semânticas. Este framework fornece um ambiente de programação para RDF, RDFS, OWL e SPARQL

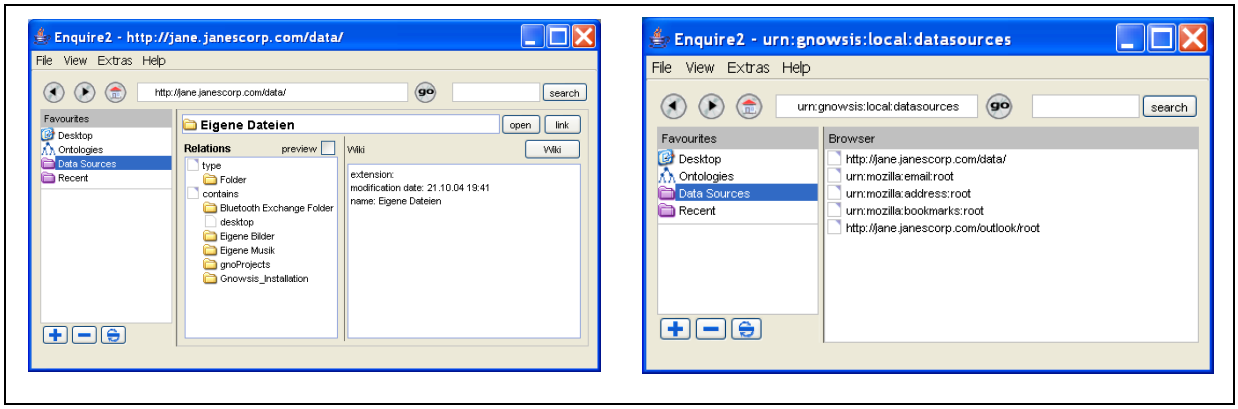


Figura 1 – Uma pasta de usuário e fonte de dados do GNOWSIS.

Fonte: `http://www.gnowsis.org/About/Screenshots`

Como mais um exemplo de utilização, cita-se o relatório técnico do NEPOMUK, realizado por Reif e colegas (REIF *et al.*, 2007), no qual é mostrado como uma ação no Desktop Semântico Social (DeSS) pode integrar e iniciar outras ações, interligando *e-mails*, calendários, agendamento de reuniões e outras funções disponíveis. Neste exemplo citado, um componente importante é adicionado: a função Social dos Desktops Semânticos. Esta função tem o objetivo de interligar as áreas de trabalho dos computadores dos usuários, criando um meio de intercambiar informações dentro de um contexto específico de trabalho. A Seção 2.6.2. deste documento trata sobre os DeSS.

Com estes exemplos e considerando que os usuários geralmente manipulam uma quantidade de informação dispersa nas diversas formas de manter informações digitais, o DS tem como função facilitar a manipulação e a recuperação das informações dos usuários, de uma forma não convencional ao que se utiliza atualmente. Cabe ressaltar que a ideia do DS em si não é nova, e outras ferramentas têm funcionalidades similares a um DS, e até alguns pacotes de aplicativos de escritório já realizam parte deste trabalho. Pode-se citar como exemplo a empresa *Google* (`www.google.com`), que disponibiliza dois softwares para utilização dos usuários: Google Desktop e o Google Docs. Os dois aplicativos citados têm funções semelhantes, mas ressalva-se que ambos trabalham em esferas diferentes de utilização: enquanto o primeiro permite pesquisar textos em documentos ou arquivos no desktop, o segundo permite que as atividades

colaborativas sejam realizadas sem a necessidade de instalação de software exclusivo para realizar o trabalho dos usuários.

Apesar desta facilidade proporcionada pelos DSs, estes não foram projetados para dar suporte para equipes de desenvolvimento de software, sejam elas de pequeno, médio ou grande porte, pois o seu principal foco é o gerenciamento do P.I.M. dos usuários.

Algumas ferramentas que utilizam os conceitos do Desktop Semântico encontram-se disponíveis para utilização, tais como o GNOWSIS (SAUERMANN *et al.,* 2006); *seMOUSE* (ITURRIOZ *et al*., 2006); HAYSTACK*,* (QUAN *et al.,* 2003); SEMANTIC MARKUP TOOL, (KETLER *et al.,* 2005) entre outras. Existe uma grande variedade de outros projetos implementados que estão à disposição da comunidade em geral. Contudo, foi graças aos esforços do projeto NEPOMUK (*Networked Environment for Personalized Ontology-based Management of Unified Knowledge*)[11]*,* que o Desktop Semântico tornou-se uma realidade concreta. Este consórcio de pesquisadores, representantes da academia e desenvolvedores de software tem dirigido esforços no desenvolvimento do Desktop Semântico Social (*Social Semantic Desktop*), que visa a dar significado semântico as informações do espaço de trabalho dos computadores dos usuários (seus “desktops”), para que eles possam interconectar e trocar informações com os desktops de outros usuários. O trabalho de (JOO, 2011) cita alguns destes desktops semânticos dentro da perspectiva da tecnologia da inovação.

Existem dois objetivos principais no desenvolvimento dos DS/DeSS. Em primeiro lugar, eles devem fornecer dados e interoperabilidade de aplicações em desktops pessoais, que é na verdade o principal objetivo do paradigma do desktop semântico. Num segundo momento, ele visa a conectar os desktops pessoais em um espaço de informação unificado de comunidades sociais, levando ao Desktop Semântico Social (DeSS). O DeSS é uma infraestrutura de colaboração em grupo, ou seja, um ambiente de colaboração que permite a criação, compartilhamento e desenvolvimento de dados e metadados por pessoas que irão interconectar-se. O DeSS capacita pessoas e comunidades para colaborarem diretamente com seus

[11] Disponível no endereço eletrônico `http://nepomuk.semanticdesktop.org/`

colegas, ocasionando numa diminuição na quantidade de tempo que eles utilizam para filtrar e arquivar a informação, muito semelhante às redes sociais. O DeSS fornece a conexão para os usuários da sua comunidade, permitindo compartilhar informações e metadados de seus desktops. A seção a seguir aborda os DeSS, apresentando-os em maiores detalhes.

2.4.2. O Desktop Semântico Social

A interligação dos diversos desktops semânticos dos usuários é a evolução natural do DS e é conhecida como Desktop Semântico Social (DeSS). Em (DECKER e FRANK, 2004), é apresentado um esboço inicial do que hoje é conhecido como DeSS. Os autores reconheceram que as diversas tecnologias que estavam emergindo na época (como a Web Semântica, computação P2P e Redes Sociais Semânticas), se integradas em uma aplicação, poderiam causar um impacto positivo na forma como as pessoas iriam interagir e realizar seus trabalhos no computador. As previsões descritas em (DECKER e FRANK, 2004) estão sendo constatadas passados poucos anos de sua publicação, e as ideias iniciais dos autores foram transformando-se em diversos aplicativos que se encontram disponíveis para utilização. Estes aplicativos podem variar, desde um *plug-in* para um navegador web até sistemas mais complexos que gerenciam toda a informação do usuário.

A Figura 2 mostra a evolução do DS para o DeSS, passando por três fases distintas. Na primeira fase, as tecnologias convergiram para o aparecimento do DS, e entre elas a Web Semântica e o P2P como principais componentes. A segunda fase é a consolidação do DS, principalmente pelo uso das ontologias como catalisador do conhecimento e para realizar a inferência sobre os objetos do domínio, agregando a semântica como principal fator para definir esta fase. A fase três é o próprio surgimento do DeSS e a sua interligação com outros DeSS, formando uma rede de colaboração especializada, sendo uma consequência das fases anteriores.

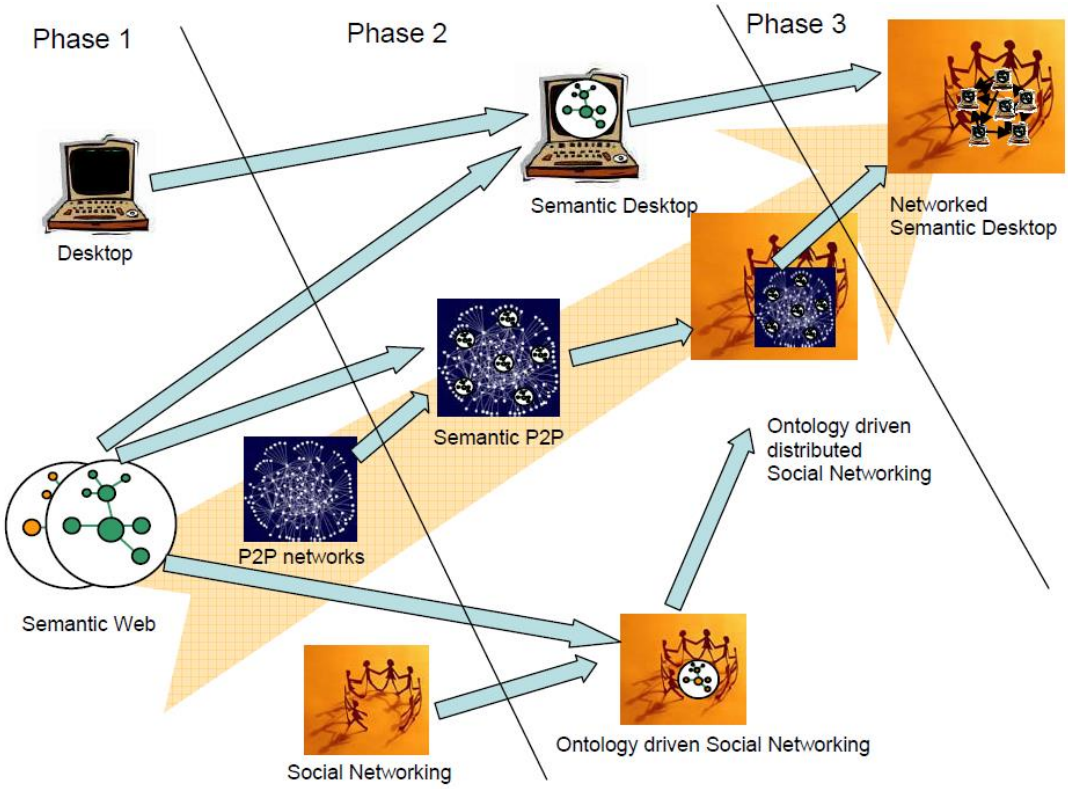


Figura 2 – Fases em direção ao Desktop Semântico Social.

Fonte: (DECKER e FRANK, 2004a).

A definição do DeSS é dada por Groza e colegas (GROZA *et al.,* 2007). Nesta definição, fica clara a utilização dos conceitos da web semântica para a elaboração do DeSS, assim como a transformação que o ambiente de trabalho do usuário poderá sofrer com a adoção desta tecnologia como ambiente de trabalho. Esta transformação irá ocorrer principalmente pelo fato das aplicações individuais terem seus dados tratados pelo DeSS para buscar informações relevantes: ”*O paradigma Desktop Semântico Social (DeSS) adota as ideias da Web Semântica, que oferece uma solução para a web. Ontologias formais capturam tanto a conceituação compartilhada de dados e modelos mentais pessoais. O RDF serve como representação comum de dados. Web Services - aplicações na Web - descrevem as suas capacidades e interfaces de forma padronizada e, portanto, tornam-se serviços da Web Semântica. No desktop, as aplicações (ou melhor: suas interfaces), serão modeladas de forma similar. Juntas, essas tecnologias oferecem um meio de construir as pontes semânticas necessárias para troca de dados e integração de aplicações. O DeSS irá transformar o ambiente de trabalho convencional em*

ambiente de rede de trabalho consistente, liberando as fronteiras entre aplicações individuais e o espaço de trabalho físico de diferentes usuários" (GROZA *et al.,* 2007. pg. 204).

As funcionalidades do DeSS são vistas no Quadro 1 e elas são necessárias para fornecer suporte às atividades que o DeSS realizam.

Quadro 1 – Funções e Funcionalidades do DeSS NEPOMUK. Adaptado de (REIF *et al.,* 2007).

Função	Funcionalidades
Desktop	Anotação Acesso *off-line*, Compartilhamento de Desktop, Gerenciamento de Recursos, Integração de Aplicações, Gerenciamento de Notificações.
Procura	Procura, Encontrar Itens Relacionados.
Social	Interação Social, Compartilhamento de Recursos, Gerenciamento de Direitos de Acesso, Publicação/Subscrição, Gerenciamento de Grupos de Usuários.
Perfis	Treinamento, Adaptação, Confiança Entre Usuários, Registro.
Análise de Dados	Inferências, Extração de Palavras-Chave, Classificação e Agrupamento.

Estas funcionalidades foram descritas no contexto do projeto NEPOMUK, e são divididas em cinco grupos, considerados como os diferentes aspectos que o DeSS deve possuir. As funcionalidades de cada aspecto foram exemplificadas em (BERNADI *et al*., 2008), (BERNARDI *et al.,* 2011) e em (REIF *et al.,* 2007a). Outros DeSS possuem também as mesmas funções e funcionalidades, sendo, portanto extensível a estes sistemas. Todos estes itens do DeSS visam a auxiliar o usuário a realizar seus trabalhos do dia-a-dia com vistas a cumprir suas tarefas, e também colaborar com demais colegas. O trabalho de (REIF *et al.,* 2007a), além de detalhar todos estes componentes citados, mostra como estas funcionalidades auxiliam a melhorar a colaboração por meio do *awareness* e da verificação de software, que é um dos objetivos que pretende-se aprimorar no desenvolvimento da pesquisa exposta neste documento.

2.5. Considerações Finais

No presente capítulo foi apresentada a fundamentação que rege este trabalho, em relação aos assuntos inerentes à pesquisa aqui proposta. Pequenas equipes que desenvolvem software também são tratadas neste capítulo, mostrando sua importância e os trabalhos que possuem este foco na literatura.

Dois outros conceitos também foram tratados neste capítulo: o *awareness* e V&V. Por intermédio deles, espera-se aprimorar a colaboração em equipes que desenvolvem software.

Outros temas de interesse, como o PIM, os DSs e os DeSS e a utilização destes últimos em tarefas específicas propostas por pesquisadores da área também são discutidos e ajudaram a formar a fundamentação do tema que se pretende trabalhar nesta pesquisa.

O WS, proposto neste trabalho, é citado como componente para auxílio às equipes que desenvolvem software e todos os itens apresentados forneceram subsídios para a condução desta pesquisa. Desta forma, ferramentas que implementam os conceitos discutidos nesta fundamentação teórica, são descritas no próximo capítulo.

Capítulo 3

Estado da Arte

Considerado o escopo deste estudo e os itens apresentados no capítulo 2, pesquisas relacionadas tais como: ferramentas de apoio ao desenvolvimento de software colaborativo, ao *awareness*, a V&V e apoio a gestão de artefatos, aplicações do DS, e análise de código-fonte, foram definidas para uma avaliação, com objetivo de situar a investigação científica em relação aos trabalhos já desenvolvidos que envolvem estes temas.

Neste levantamento, foram considerados os trabalhos que pudessem contribuir para a criação do WS, analisando critérios como a atualidade, relevância e o alinhamento com os objetivos do WS. Estes itens foram levados em consideração e são tratados na sequência. Mais especificamente, as pesquisas que apresentam modelos, arquiteturas ou protótipos, foram selecionadas para serem explanadas, e os trabalhos que atenderam aos requisitos descritos são apresentados.

Porém, antes de analisar estes ambientes e ferramentas, e visando a conhecer o estado atual dos trabalhos foram buscados trabalhos que realizaram revisões sistemáticas sobre estes temas, por considerar-se que estas revisões disponibilizam muitas informações relevantes e que podem contribuir para o entendimento mais acurado sobre o tema.

Assim, um estudo empírico que culminou na elaboração de uma revisão sistemática da literatura sobre a colaboração no desenvolvimento de software é apresentada por Treude e colegas (TREUDE *et al.,* 2009). Sua pesquisa estendeu os estudos realizados anteriormente por não tratar apenas o desenvolvimento de software global ou distribuído, mas incluindo pesquisas que tivessem o foco na colaboração entre desenvolvedores.

Outro trabalho que tem como principal objetivo investigar empiricamente a adoção e uso de Revisões Sistemáticas da Literatura (RSL) na engenharia de software (ES) a partir de diferentes perspectivas, é o trabalho de Zhang e Babar (ZHANG e BABAR, 2011). A pesquisa aponta que a adoção das RSL como metodologia de pesquisa tem se tornado popular. Há uma percepção geral positiva sobre esta metodologia, e os autores afirmam que os resultados fornecem motivos para introspecção interessantes em diferentes aspectos da RSL para a ES.

Mais um trabalho teórico que analisa os fatores de produtividade e estratégias de desenvolvimento de software, fornecendo uma visão consolidada dos principais fatores que têm afetado a produtividade ao longo dos anos e as estratégias utilizadas para lidar com tais fatores atualmente é o trabalho realizado por (SAMPAIO *et al.,* 2010). Através da sua revisão da literatura, os autores têm como objetivo apoiar o desenvolvimento de software na seleção de suas estratégias para melhorar a produtividade, maximizando os fatores positivos e minimizando ou evitando o impacto dos negativos. São estudados fatores que são citados desde a década de 70 até estudos contemporâneos, fornecendo assim uma ampla revisão destes fatores. Outros trabalhos que pesquisaram a forma de se trabalhar colaborativamente em desenvolvimento de software são encontrados em (HILDENBRAND *et al.,* 2008), (TERUEL *et al.,* 2012) e (DUQUE *et al.,* 2012). Na pesquisa realizada por (CANFORA *et al.,* 2003), os autores definem as principais funções de uma Plataforma Colaborativa de Engenharia de Software (PCES) e também definem que a mesma pode ser um conjunto de soluções integradas que fornecem suporte para as tarefas específicas do desenvolvimento de software. As principais funções de uma PCES apontadas pelos autores são rotuladas em uma taxonomia, para auxiliar a classificá-las. Esta taxonomia apresentada pelos autores foi adaptada de Carmel (1999 *apud* CANFORA *et al.,* 2003) e contém oito itens:

gerenciamento de configuração; rastreamento de *bugs* e de mudanças; modelagem de produtos e processos; ferramentas de programação; estado do projeto; agendamento de tarefas; centro de conhecimentos e notificação de serviços/eventos. Destes itens, deverão fazer parte do protótipo do WS, o rastreamento de mudanças (necessário para acompanhar as modificações que serão realizadas no código-fonte), o estado do projeto (para verificação e acompanhamento da codificação em relação ao que modelado por meio da UML para o sistema), o centro de conhecimentos (para concentrar as informações do andamento do projeto sendo desenvolvido) e a notificação de eventos (eventos relacionados com a programação e acompanhamento em tempo real do sistema em desenvolvimento). Na visão dos autores, as PCESs também incluem algumas funções genéricas de colaboração, tais como e-mail (principalmente para serviços de notificação) ou fóruns de discussão baseados na Web para apoiar a comunicação entre os membros da equipe. Esta classificação auxilia aos interessados em catalogar as ferramentas colaborativas que podem ser úteis no desenvolvimento de software.

Neste sentido, para fornecer suporte ao desenvolvimento de software colaborativo, pesquisadores da área do CSCW estão entre os que mais contribuem com ferramentas de apoio ao desenvolvimento de software. Vários trabalhos já foram desenvolvidos anteriormente ((COOK e CHURCHER, 2005), (STOREY *et al.,* 2006), (JIANG *et al.,* 2006), (SARMA *et al.,* 2003)) e outros estão sendo realizados e divulgados recentemente ((YUYAN *et al.,* 2011), (LINXIA, 2010), (SARMIENTO e COLLAZOS, 2012)). Estes trabalhos, em sua maioria, privilegiam dois aspectos: melhorar a infraestrutura de apoio ao desenvolvimento distribuído (por meio de ambientes integrados ou *groupwares*) e motivar a comunicação dos participantes do projeto colaborativo e o foco é fornecer suporte para grandes equipes distribuídas de desenvolvimento de software.

Os trabalhos citados anteriormente contribuíram para listar as ferramentas de apoio e métodos para desenvolvimento de software que possuíssem características semelhantes ao que se pretende com a criação do WS. Estes trabalhos representam o estado da arte levantado para esta pesquisa. Além disto, é apresentada a forma

como os artigos foram buscados nas bases de dados eletrônicas disponíveis, além da caracterização atribuída para pesquisa aqui desenvolvida.

3.1 Obtenção de Artigos para Fundamentação da Pesquisa

A fim de encontrar os artigos que dão suporte para elaboração desta pesquisa, algumas bases de conhecimento foram consultadas. Estas bases são apresentadas nos trabalhos de Kitchenham e Charters (KITCHENHAM e CHARTES, 2007) e de Brereton e colegas (BRERETON *et al.,* 2007). O primeiro trabalho identificou nove bases eletrônicas de pesquisa, e o segundo trabalho citado, catalogou oito bases.

Portanto, adotou-se a utilização dos trabalhos de (KITCHENHAM e CHARTERS, 2007) e (BRERETON *et al.,* 2007), por serem os trabalhos adotados pela comunidade de engenharia de software como referências para realização de revisões sistemáticas. Ressalta-se que a utilização destas fontes serviu para determinar quais bases de conhecimento eletrônicas seriam utilizadas para a busca de artigos relevantes para os objetivos desta pesquisa. Desta forma, o Quadro 2 mostra as bases de conhecimentos eletrônicas citadas nos trabalhos mencionados e que foram utilizadas para busca de artigos de referência para elaboração desta pesquisa:

Quadro 2 – Bases de conhecimentos da área de informática.

Fonte	Bases de conhecimento
BRERETON *et al.,* 2007.	1 - IEEExplore (http://ieeexplore.ieee.org/Xplore/guesthome.jsp/) 2 - ACM Digital Library (http://dl.acm.org/) 3 - Google scholar (scholar.google.com/) 4 - Citeseer Library (citeseer.ist.psu.edu/) 5 - Keele University's Electronic Library (www.keele.ac.uk/library/) 6 - Inspec (www.iee.org/publish/inspec/) 7 - ScienceDirect (www.sciencedirect.com/) 8 - EI Compendex (http://www.engineeringvillage.com/home.url?acw=)
	1 - IEEExplore (http://ieeexplore.ieee.org/Xplore/guesthome.jsp) 2 - ACM Digital Library (http://dl.acm.org/) 3 - Google scholar (scholar.google.com/) 4 - Citeseer Library (citeseer.ist.psu.edu) 5 - Inspec (www.iee.org/publish/inspec/) 6 - ScienceDirect (www.sciencedirect.com/) 7 - EI Compendex (http://www.engineeringvillage.com/home.url?acw=) 8 - SpringerLink (http://www.springerlink.com/) 9 - Scopus (http://www.scopus.com/home.url/)

Utilizando estas bases encontradas nos dois trabalhos apresentados, foram pesquisados artigos de trabalhos que tratassem de temas como colaboração, gestão

de artefatos de software e análise de código-fonte. Os DS/DeSS também tiveram as pesquisas relevantes localizadas nas bases indicadas pelos pesquisadores.

Com a definição das bases, o próximo passo foi a elaboração dos termos que seriam pesquisados de quais termos seriam pesquisados para a busca de áreas/trabalhos que pudessem fornecer os subsídios necessários para o levantamento. O Quadro 3 apresenta alguns destes termos e *as* consultas de pesquisa que foram utilizados. Como todas as bases usadas são internacionais, a linguagem utilizada para efetivar a busca foi o inglês.

Quadro 3 – Alguns termos e Consulta da pesquisa realizada nas bases de conhecimento.

Termos de pesquisa	Consultas
• Semantic Desktop • Social Semantic Desktop • Small Team • Software Development Team • Semantic Software Development • Collaborative Software Development • Collaboration • Cooperation • Collaborative Writing • Collaborative Applications • Small Teams • Collaboration in Software Development • Co-location • Software Development Process • Semantic Software Engineering Environments • Awareness • Verification and Validation • V&V	• Collaboration AND “software development” AND study • Semantics AND “software development” • “Semantic Desktop” • “Social Semantic Desktop” • Collaboration OR Cooperation AND “software development” • “small teams” AND “software development” • Awareness • Awareness AND "software engineering" • "verification and validation" OR V&V OR "software verification"

Como a pesquisa dos artigos nas bases de conhecimento foi realizada totalmente em inglês, para suprir a falta de uma base de artigos em português, a Scielo[12] foi utilizada para garantir que pesquisas relevantes na língua-mãe pudessem ser levantadas e também porque sua exclusão poderia deixar de fora trabalhos relevantes de autoria de pesquisadores brasileiros.

O próximo passo após a identificação preliminar dos artigos de interesse foi a eliminação dos títulos duplicados e também a exclusão dos artigos que não

[12] http://www.scielo.org/php/index.php

tratassem dos temas de interesse deste trabalho, como por exemplo, "*Semantic Web*" ou "*Desktop Search*", que apesar de serem temas interessantes, tem pouca utilidade para os objetivos deste trabalho.

A totalização dos artigos levantados por este método é apresentada no Quadro 4.

Quadro 4 – Totais de artigos levantados pela pesquisa realizada.

Termo agrupador	Total de Artigos
Colaboração	44
Desktop semântico	123
Desenvolvimento de Software	45
Awareness	43
Verificação e Validação	27

Os artigos levantados estão disponíveis na plataforma Mendeley[13], sendo disponibilizados aos interessados, pois além de ser uma ferramenta de gerenciamento de referências, permite o compartilhamento de artigos científicos. A utilização destes artigos foi realizada nos capítulos desde documento e estes estão agrupados nas categorias citadas por meio do termo agrupador, indicado no Quadro 4. As análises nas seções seguintes são realizadas por meio das referências obtidas neste processo, e o Quadro 5 mostra os links que contêm os artigos de cada termo agrupador que foi levantado da pesquisa. Todos os artigos estão disponíveis aos interessados nestes links, na forma de livre acesso.

Quadro 5 – Links das bases de artigos levantados.

Termo Agrupador	Links para os artigos levantados
Colaboração	www.mendeley.com/groups/2138193/artigos-colaboracao/
Desktop semântico	www.mendeley.com/groups/2121063/artigos-desktop-semantico/
Desenvolvimento de software	www.mendeley.com/groups/2138183/artigos-desenvolvimento-de-software/
Awareness	www.mendeley.com/groups/4497621/artigos-awareness/
Verificação e Validação	www.mendeley.com/groups/4497741/artigos-verificacao/

[13] http://www.mendeley.com/

3.2. Ferramentas de Apoio ao Desenvolvimento Colaborativo de Software

O processo utilizado para se desenvolver software na atualidade é altamente baseado em equipes que trabalham em conjunto, e estas equipes necessitam colaborar entre si com vistas a alcançar objetivos comuns em um menor tempo e de forma eficiente. Assim, o projeto colaborativo de desenvolvimento de software é um tema onde vários pesquisadores têm contribuído para o aprimoramento da pesquisa.

As Plataformas Colaborativas de Engenharia de Software (PCES), descritas no início deste capítulo, tomam forma em diversos trabalhos de pesquisadores que contribuem para o aprimoramento da colaboração no desenvolvimento de software. Um destes PCES é o resultado do trabalho produzido por (STOREY *et al.,* 2006), que apresenta o projeto de uma ferramenta colaborativa de apoio ao desenvolvimento de software assíncrono, chamada TagSEA. O objetivo deste trabalho foi desenvolver uma ferramenta de anotação de código que melhore a coordenação, navegação e captura de conhecimento considerado relevante em uma equipe de desenvolvimento de software. A ferramenta é desenvolvida baseada nos conceitos de *Waypoints* e *Tags Sociais*. *Waypoints* são usados por sistemas de posicionamento geográfico para salvar locais de interesse, e no trabalho em questão é utilizado com o mesmo propósito, no entanto aplicado no ambiente de desenvolvimento de software. As *tags* sociais, também conhecidas como *bookmarking* social, permitem aos usuários criar marcadores compartilhados de recursos *on-line* com metadados adicionais. Baseado nesses dois conceitos foi criada uma ferramenta integrada ao ambiente de desenvolvimento *Eclipse* (CLAYBERG e RUBEL, 2004), na qual adicionam-se *waypoints* e cria-se um conjunto de *tags* associadas a esta IDE, com o intuito de criar um mecanismo de documentação que capture itens relevantes sobre o código fonte e que possa ser compartilhado com a equipe de desenvolvimento. Alguns destes itens são:

- Marcação (ou Tagueamento) de locais de interesse com palavras-chave;
- Adição de dados e metadados do autor do processo;
- Filtragem das *tags* criadas;
- Retorno a lugares anteriormente tagueados;

- Criação de *tags* compartilhadas armazenadas no código ou *tags* armazenadas no espaço de trabalho privado;
- Criação de apresentações que combinam o código-fonte, slides, etc.

Esta ferramenta está disponível no SOURCEFORGE, no endereço `http://tagsea.sourceforge.net/`, podendo desta forma ser utilizada por qualquer interessado sem necessitar de licenças especiais ou autorizações de uso.

Os trabalhos de Cubranic e colegas ((CUBRANIC *e* MURPHY*,* 2003), (CUBRANIC *et al.,* 2005)) apresentam uma ferramenta denominada HIPIKAT, que fornece aos desenvolvedores de software acesso à memória do grupo, para projetos de desenvolvimento de software, que é implicitamente formado pelos artefatos produzidos durante este desenvolvimento. Esta memória do projeto é criada automaticamente com pouca alteração às atuais práticas de trabalho, já que integrada ao ambiente de desenvolvimento. A abordagem proposta é divida em duas partes. A primeira é a memória dos projetos, construída a partir de artefatos e comunicações criados durante a história de um projeto de desenvolvimento de software. Na segunda fase são recomendados artefatos aos desenvolvedores que são selecionados a partir da memória do projeto e que podem ser relevantes para a tarefa a ser executada. Para validar a ferramenta criada, dois estudos são conduzidos. O primeiro avaliou a utilidade das recomendações HIPIKAT e no segundo estudo é avaliado como os novos desenvolvedores que se juntam ao projeto podem ser beneficiados pelas recomendações do HIPIKAT. Através destes dois estudos qualitativos de avaliação, os autores mostram que a abordagem proposta se mostra promissora para ajudar um novato executar uma tarefa de forma eficaz em um sistema desconhecido. A ferramenta está disponível para ser avaliada no endereço `http://www.cs.ubc.ca/labs/spl/projects/hipikat/`.

Outro trabalho (LIMA *et al.,* 2010) apresenta a ferramenta colaborativa baseada em agentes chamada ARARA (do acrônimo em inglês - *ARtifacts And Requirements Awareness Reinforcement Agents*). Seu desenvolvimento foi motivado pela necessidade de prover a percepção (*awareness*) em ambientes de desenvolvimento de software, onde os requisitos mudam com frequência e existe falta de conhecimento dessas mudanças por parte da equipe. A ARARA atualiza os artefatos de forma automática em relação às mudanças ocorridas nestes, e também

em outros artefatos que devem ser propagados. As conclusões dos autores apontam para um bom desempenho da ARARA, tanto em termos de precisão e renovação no processo de rotular os artefatos. Mostrou-se capaz também de relacionar os artefatos que tratam dos mesmos conceitos, proporcionando a redução nos esforços necessários para coordenar tarefas e manter conhecimento do que está ocorrendo na área de trabalho da equipe que está desenvolvendo software.

Mais um trabalho interessante que apoia o desenvolvimento de software de forma colaborativa é o proposto por Falbo e colegas (FALBO *et al.,* 1998). O trabalho aborda a integração do conhecimento no que os autores denominam de Ambientes de Desenvolvimento de Software (ADS). Falbo et. al. defendem que com o auxílio de um ADS, o processo de desenvolvimento pode ser conduzido de modo uniforme e consistente. Além disso, dada a complexidade das tarefas realizadas no desenvolvimento de software, é importante que o ADS ofereça algum tipo de suporte baseado em conhecimento para o desenvolvedor. Idealmente, o conhecimento não deve ficar embutido em uma ferramenta, mas, ao contrário, deve estar integrado ao ambiente para que possa ser compartilhado e reutilizado por diversas ferramentas. Dessa forma, Falbo e colegas propõem um modelo de integração de conhecimento para ADSs. A abordagem proposta utiliza o conceito de servidores de conhecimento, o qual segundo os autores, não é apenas uma ferramenta projetada para atuar como um meio para resolver problemas ou para ser utilizado pelo engenheiro de software em um processo de desenvolvimento. Sua funcionalidade consiste em prover uma infraestrutura comum para o desenvolvimento de um conjunto de agentes dessa natureza em um domínio de interesse como agentes de integração de conhecimento em ADSs. Neste trabalho também é apresentada uma ontologia de processo de desenvolvimento de software desenvolvida para o servidor de conhecimento de processo.

Os ADSs propostos por Falbo e colegas tiveram o acréscimo do componente semântico em trabalhos posteriores ((FALBO *et al.,* 2002), (FALBO *et al.,* 2004) e (FALBO *et al.,* 2005)). Este componente semântico é implementado principalmente pela utilização de ontologias na criação de um ADS, que Falbo denominou de ODE (*Ontology-based software Development Environment*), cuja principal característica é estar centrado em processos e tem a sua fundamentação totalmente baseada em

ontologias. Segundo os autores, se uma ferramenta para ADS é construída baseada em ontologias, a integração destas ferramentas pode ser facilitada, pois os conceitos envolvidos são bem definidos pela própria ontologia, o que facilita a sua utilização e integração no desenvolvimento de software.

O Quadro 6 resume as principais características das ferramentas analisadas anteriormente.

Quadro 6 – Ferramentas analisadas e suas principais características.

Fonte	Nome da Ferramenta	Principais Características
STOREY *et al.*, 2006.	TagSEA.	- Permitir anotação no código com captura de conhecimento relevante, tagueamento de locais, adição de metadados em uma ferramenta assíncrona;
CUBRANIC *et al.,* 2005, CUBRANIC *e* MURPHY*,* 2003.	HIPIKAT	- Acesso a memória do grupo de desenvolvimento de software construído por meio dos artefatos criados;
LIMA *et al.,* 2010.	ARARA	- *Awareness* para o grupo de desenvolvimento onde os requisitos de software sofrem mudanças constantes; - Atualização automática das mudanças nos artefatos;
STOREY *et al.,* 2005.	-	- Analise de doze ferramentas de *awareness* e a criação de um framework de auxílio para comparar e compreender instrumentos de *awareness*, utilizando cinco dimensões principais.
BRAUN *et al.,* 2007.	-	- Análise de processos de trabalhos científicos (*e-science*), dentro do domínio de prototipagem rápida.
FALBO *et al.*, 1998.	ADS	- Integração do conhecimento em Ambiente de trabalho de Software.
FALBO *et al.,* 2002; FALBO *et al.,* 2004; FALBO *et al.*, 2005.	ODE	- Adição do componente semântico por meio da inserção de ontologias que provêm uma infraestrutura comum para o desenvolvimento de um conjunto de agentes.

3.3. Apoio a Gestão de Artefatos

O apoio à gestão de artefatos (documentos de texto, planilhas, diagramas, código-fonte, modelos, conjuntos de teste, entre outros) é realizado por diversas ferramentas. Os DSs são uma destas formas que auxiliam os usuários na tarefa de manipular suas informações e são um dos componentes que foram utilizados como fontes de inspiração para a implementação desta pesquisa, e assim trabalhos relacionados ao DS são apresentadas nos próximos itens.

O trabalho de Schandl (SCHANDL, 2009) apresenta uma arquitetura para o desenvolvimento de aplicações “desktop-semânticas”, mostrando uma alternativa para o modelo organizacional de gerenciamento dos dados do usuário, com intuito de armazenar e recuperar informação, utilizando os conceitos da web semântica. A discussão de um modelo de infraestrutura semântica e como os atuais artefatos podem ser utilizados por desenvolvedores de aplicações é apresentado pelo autor, mostrando os três cenários de aplicação para realizar isto: aprimoramento da comunicação eletrônica, gerenciamento de informações pessoais dirigidas por meio de *wikis* e, por último, mas de capital interesse para o desenvolvimento da pesquisa proposta neste documento, o desenvolvimento de software semântico. A criação deste modelo (chamado de SILE MODEL) é a principal contribuição do trabalho de Schandl, sendo este criado para permitir sua utilização por desenvolvedores de aplicativos em seus artefatos reais. Para atingir seu objetivo, o autor realizou uma análise extensiva dos sistemas de arquivo disponíveis para implementação e também um estudo comparativo de oito desktops semânticos. Para a crítica destes DSs, foram utilizados alguns critérios e dimensões, pelos quais foram analisados e classificados. Ao final de seu trabalho, Schandl apresenta uma nova tabela, incluindo o SILE MODEL, realizando a comparação de sua proposta com as demais apresentadas, utilizando o mesmo conjunto de critérios e dimensões utilizados anteriormente para classificar os DSs. Suas conclusões apontam que o novo modelo de gerenciamento de dados proposto oferece um alto nível de flexibilidade capaz de representar os diferentes tipos de dados tipicamente encontrados em um desktop. Para atingir este objetivo, três tipos de implementações de repositórios foram criados e testados: um baseado na web semântica, outro em mensagens armazenados em servidores IMAP (*Internet Message Access Protocol*) e um protótipo para busca, listagem e manipulação dos dados no SILE MODEL, com uma interface gráfica chamada SEMPLORER. O trabalho foi delineado com sistemas que continham dados no contexto do gerenciamento das informações pessoais, notadamente um servidor de correio eletrônico. Esta delimitação do escopo de atuação do SILE MODEL é tratada como sendo apenas o passo inicial e como prova preliminar da hipótese levantada. Nos trabalhos futuros, dois itens ficaram em aberto para serem tratados, que são: a integração das diversas aplicações desktop e a integração de

diversas fontes de dados que existem em um computador. A implantação destes dois itens pode fornecer subsídios para gerenciar os sistemas que um usuário normalmente utiliza no seu dia-a-dia, como editores de texto, planilhas eletrônicas, gerenciamento de dados e outros itens de seu cotidiano.

Outro trabalho que trata sobre documentos semânticos é a tese de doutorado de Sasa Nesic (NESIC, 2010). Ele apresenta um modelo semântico de documentos chamado SDArch (*Semantic Document Architecture*) que fornece soluções para o repositório de documentos e serviços que suportam os documentos semânticos, além de ferramentas que permitem aos usuários interagir com os artefatos geridos por esta arquitetura. Segundo o autor, as principais contribuições da sua tese são o próprio desenvolvimento e a concepção do SDArch. A tese é validada pelos dois estudos de avaliação: a avaliação experimental da recuperação da informação em coleções de documentos integrados semanticamente, e a avaliação de usabilidade da eficácia do usuário, eficiência e satisfação em usar os serviços da SDArch e suas ferramentas. Os resultados destes dois estudos de avaliação mostraram que os documentos semânticos têm potencial para integração semântica e apresenta melhora na interoperabilidade dos dados da área de trabalho, aprimorando assim a eficácia e a eficiência dos usuários de desktop, quando estes realizam suas tarefas diárias utilizando o modelo semântico criado.

O desenvolvimento de uma ferramenta semântica é apresentado na dissertação de mestrado proposta por Muhammed Faheen (FAHEEN, 2010). Para a elaboração de seu trabalho, Faheen formalizou primeiramente uma metodologia para extração do modelo conceitual (ou ontologia) dos dados do desktop, bem como a criação de uma base de dados. Uma API do sistema *Spotlight* foi utilizada como framework para auxiliar a obter a tecnologia de busca para o desktop semântico que foi desenvolvido. Esta API foi utilizada para extrair as meta-informações do sistema, para que em uma terceira fase, a população da ontologia e da base de dados da ferramenta proposta, fosse efetivada. Para testar sua proposta, o autor realizou a população da ontologia e da base de dados com informações extraídas do sistema, por meio de esquema conceitual e utilizando como ferramenta a OWL API, disponível em `www.owlapi.sourceforge.net/`. Na sequência do trabalho, alguns casos de uso foram utilizados para comparar o vocabulário da ontologia com

a base de dados para obter os resultados da avaliação. A contribuição de Faheen ao desenvolver a ferramenta, é apresentar a metodologia utilizada para elaboração da mesma, incluindo benefícios para várias áreas de pesquisa, tais como: projeto de banco de dados, integração de dados e mapamento de ontologias, entre as principais.

O objetivo principal da tese proposta por Ali Reza Etezadi (ETEZADI, 2008), é a coleta de informações que existem, por exemplo, em um escritório, especificamente em documentos relacionados com processamento de texto como MICROSOFT WORD ou OPEN OFFICE WRITER. O autor sugere o que ele denomina de método de “colheita” de entidades semânticas, que são criadas ao longo da manipulação destes documentos, de acordo com o padrão criado pelo próprio usuário sobre a estrutura do documento e as palavras-chave do mesmo. O autor define como “colheita” o ato de importar dados externos para as estruturas ontológicas criadas, e esta colheita é realizada utilizando como ambiente de execução, o framework chamado *Iris Semantic Desktop* (`www.openiris.org`), o qual permite aos usuários criar um mapa pessoal por meio de seus objetos de escritório relacionados com as informações. Um *plug-in* foi desenvolvido e agregado ao IRIS para interpretar se os documentos foram adicionados ou modificados pelo usuário dentro do aplicativo. Para a validação de seu trabalho, Etezadi realizou testes dentro do contexto do comando militar, que possui um fluxo de processo de documentos altamente estruturado. Os resultados do trabalho auxiliam o usuário final na extração de dados utilizando seus próprios padrões da atividade de planejamento operacional, especialmente aqueles pertencentes ao domínio militar, objeto de estudos do autor.

O tema central da tese de Danish Nadeem (NADEEM, 2007) centra-se nas considerações cognitivas da Gestão de Informação Pessoal. O trabalho baseia-se em uma avaliação do protótipo do DS GNOWSIS, observando a utilidade do desktop DS no comportamento do usuário ao utilizar suas informações. Para conduzir sua pesquisa, Nadeem realizou uma revisão da literatura sobre aplicações do DS, levando em consideração principalmente aspectos advindos da Ciência Cognitiva e também da Filosofia, com a finalidade de avaliar sua utilização em P.I.M.. A revisão da literatura realizada foi detalhada em ambas as áreas, para explorar as ideias que

são relevantes no contexto da aplicação do desktop semântico. A teoria dos modelos mentais (CRAIK, 1943) também é explorada e suas características são utilizadas como uma ferramenta analítica para compreender o comportamento das pessoas para organizar suas informações pessoais. O autor afirma que os modelos mentais têm sido pesquisados para entender como as pessoas representam suas imagens mentais, premissas e conceitos do mundo real, e isto é de interesse para a condução do trabalho que foi desenvolvido. Suas principais contribuições podem ser resumidas em dois itens:

- A realização da revisão da literatura detalhada em teorias da filosofia e da ciência cognitiva, para explorar os conceitos que são relevantes no contexto do DS, criando uma abordagem para fornecer suporte eficiente para o gerenciamento de informações pessoais (PIM).
- A utilização dos conceitos da teoria dos modelos mentais para a incorporação da mesma no projeto e implantação de ferramentas que consideram esta teoria em seus projetos.

A proposta de Simon Scerri (SCERRI, 2010) identifica o *e-mail* como um instrumento fundamental de comunicação, que poderia sofrer uma transformação por meio da utilização dos conceitos do DeSS. Com esta noção, o autor trata o envio de mensagens eletrônicas mais do que apenas um meio de comunicação, muitas vezes servindo como uma extensão virtual para o ambiente de trabalho do usuário, em que eles colaboram; geram e partilham de novas informações durante o processo. Nesta visão, a inclusão de componentes semânticos ao *e-mail* é a principal proposta da tese de Scerri, criando a plataforma denominada SEMANTA. A plataforma SEMANTA foi incluída, na forma de um *add-on* para dois correios de mensagens eletrônicas disponíveis, o MICROSOFT OUTLOOK e o MOZILLA THUNDERBIRD, agregando as seguintes funções semânticas:

- Agendamento de reuniões ou eventos dentro do próprio corpo do e-mail;
- Delegação de tarefas por meio do texto do e-mail, analisando palavras-chaves;
- Troca de informações por meio de arquivos;
- Inclusão de funções que executam tarefas de forma automática, tais como lembrar ao usuário tarefas não cumpridas.

Os resultados obtidos pelo autor apontam que existe um potencial para sistemas de suporte à comunicação semântica para melhorar o processo de colaboração, por meio de uma aplicação de gestão de *workflow* que suporte a colaboração por meio de *e-mail*. Como benefícios adicionais, a plataforma SEMANTA pode reduzir a sobrecarga de informações, como também ser um meio de comunicação apropriado para o DeSS, permitindo o intercâmbio e a integração do conhecimento semântico por meio dela.

A tese de doutorado de Leo Sauermann, intitulada "*The Gnowsis Semantic Desktop approach to Personal Information Management: Weaving the Personal Semantic Web*" (SAUERMANN, 2009), é um dos principais projetos advindos do consórcio NEPOMUK. Em sua tese, Sauermann consolida o GNOWSIS, mostrando que a implementação da arquitetura é válida e funcional para situações reais. Os protótipos avaliados mostraram seus benefícios e diversas pessoas fizeram uma utilização produtiva de seu sistema, conforme se pode verificar em suas publicações: (SAUERMANN e SCHWARZ, 2004), (SAUERMANN *et al.,* 2007). Uma avaliação de longo prazo também foi feita por Sauermann, onde a abordagem realizada em 2006, um estudo de caso com duração de dois meses, com oito participantes foi continuado (SAUERMAN *et al.,* 2006), mas agora com dois participantes da equipe original utilizando o GNOWSIS por um período maior, (mais dois anos). O objetivo foi analisar seus padrões por este tempo mais longo de utilização. Esta continuação do estudo por um tempo maior de utilização permitiu ao autor avaliar como o sistema foi utilizado para o gerenciamento do P.I.M. neste tempo dilatado. Segundo as conclusões do estudo, descobriu-se que no ambiente pessoal, as simples relações (semelhantes ao que neste trabalho será denominado de Ligação Semântica) *has-Part* e *is-Related* (oriundos da Lógica de Inferência e utilizados pelo GNOWSIS) são suficientes para os usuários encontrarem as informações procuradas, e que o *wiki* semântico pessoal do GNOWSIS foi utilizado criativamente para anotar informações (SAUERMANN e HEIM, 2008). Ressalta-se que o sistema GNOWSIS está disponível em código aberto, e pesquisadores têm utilizado este código como base para realizar seus trabalhos de pesquisa, tais como (SAUERMANN e SCHWARZ, 2004), (SAUERMANN *et al.,* 2006) e (WOERNDL e WOEHRL, 2008). O Quadro 7 resume as ferramentas apresentadas.

Quadro 7 – Características das ferramentas analisadas de apoio a gestão de artefatos.

Fonte	Nome da Ferramenta	Principais Características
SCHANDL, 2009.	SILE MODEL	- Novo modelo de gerenciamento de dados; - Criação de repositórios para diferentes tipos de dados do desktop;
NESIC, 2010.	SDArch	- Novo modelo semântico de documentos; - Melhora a interoperabilidade dos dados dos usuários;
FAHEEN, 2010.	DS	- Criação de uma metodologia para extração da ontologia por meio dos dados do desktop;
ETEZADI, 2008.	-	- Realizar a coleta de informações de entidades semânticas do desktop dos usuários; - *Plug-in* agregado ao DS *Iris;*
NADEEM, 2007.	-	- Avaliação do DS Gnowsis e sua utilização relacionada com a teoria dos Modelos Mentais;
SCERRI, 2010.	SEMANTA	- *E-mail* - extensão virtual do ambiente de trabalho por meio da inclusão de componentes semânticos;
SAUERMANN, 2009.	GNOWSIS	- DS disponível que serviu de base para muitos outros trabalhos; - Gerenciamento do P.I.M. melhorado; - Código fonte aberto.

3.4. Aplicações do Desktop Semântico

Os DSs podem ser utilizados para auxiliar a resolver diversos problemas do dia-a-dia das pessoas. Com relação à aplicação do DS em ambiente profissional, alguns estudos discutem sua viabilidade. No estudo desenvolvido por Leifler e Eriksson (LEIFLER e ERIKSSON, 2008), os autores estudaram o DS dentro do contexto do comando militar, que possui um fluxo de processo de documentos altamente estruturado. Os autores concluíram que o DS tem o potencial para melhorar os processos de documentos facilitando a gestão do conhecimento. Uma das principais vantagens apontadas pelos autores, é que o DS suporta os usuários em suas atividades diárias sem a introdução de novos sistemas, que exigiriam fluxos separados de tratamento da informação. Para maximizar os benefícios desta abordagem, porém, é necessário adaptar o DS para o ambiente de gestão do conhecimento dos usuários, levando em consideração as tarefas que estão sendo realizadas.

Outra forma de utilização dos DSs é encontrar nas meta-informações que os DSs possuem, pessoas que têm afinidades profissionais entre si e que podem colaborar para resolver um problema ou um trabalho em conjunto em itens específicos que necessitem de colaboração. Estas possibilidades são exploradas

nos trabalhos de Demartini e Niederée (DEMARTINI e NIEDERÉE, 2008) e de Braun e Colegas (BRAUN *et al.,* 2007).

No trabalho (DEMARTINI e NIEDERÉE, 2008), os autores desenvolveram um sistema que possibilita encontrar especialistas em determinados assuntos ou questões analisando o conteúdo indexado por um DS. No trabalho desenvolvido pelos autores, o sistema chamado BEAGLE++ é utilizado como base. Os autores afirmam que encontrar itens na área de trabalho não é a mesma tarefa de encontrar documentos na web. Mostram-se também que vários sistemas comerciais têm sido propostos e realizam este tipo de tarefa (por exemplo, Google, Yahoo!, Microsoft), e justificam a escolha do BEAGLE++ (MINACK *et al.,* 2010), por ser um motor de busca semântica da área de trabalho. Para os testes do sistema que foi desenvolvido, os autores utilizaram duas bases de teste específicas para DSs disponibilizadas pelo Consórcio NEPOMUK em sua página web (disponíveis em `http://dev.nepomuk.semanticdesktop.org/wiki/Dirk` e também em `http://dev.nepomuk.semanticdesktop.org/wiki/Claudia`). Estas bases simulam situações reais de uso de um DS (no caso, o DS NEPOMUK) e estes dois conjuntos de dados também podem ser utilizados para testar os diferentes desktops semânticos com a mesma base. Isto facilitaria o trabalho de comparação, pois o desempenho de vários DSs tem condições de ser avaliado com um mesmo conjunto de dados, além de fornecer todos os relacionamentos semânticos de uma situação real de utilização. Além desta simulação de situações reais e de auxiliar a padronizar um conjunto de testes para os DSs, estas bases auxiliam a diminuir muito o tempo de preparação de uma base que contenha todos os itens necessários para realizar os experimentos. Os resultados obtidos com o trabalho de Demartini e Niederée (DEMARTINI e NIEDERÉE, 2008) mostram que o sistema desenvolvido utilizando um DS como base para encontrar especialistas são bons, e que o mesmo não ocorre quando utilizam dados diretamente do desktop do usuário, sem uso de um DS.

Outro trabalho que tem o objetivo de produzir um sistema que permite ao usuário criar tarefas colaborativas e também facilita que estes possam encontrar possíveis colaboradores de acordo com a tarefa necessária, foi realizado pelos pesquisadores Debasttita e Abela (DEBATTISTA e ABELA, 2011). Os autores

consideram que pesquisadores empenham uma enorme quantidade de tempo e esforço em tarefas colaborativas, como a escrita de propostas de pesquisa ou artigos. Geralmente para estas tarefas, utilizam-se uma combinação de diferentes ferramentas: softwares de controle de versões, editores de texto, mensagens instantâneas e *e-mails*, tudo para se manter a par do estado das tarefas a serem desempenhadas. A proposta dos autores para gerenciar este processo é a criação do ambiente denominado BIZZILLA. Como espinha dorsal para o protótipo desenvolvido pelos autores, foi utilizado o DS NEPOMUK, e os resultados obtidos com o módulo localizador de especialistas foi positivo. Como a pesquisa conduzida é muito recente e utilizaram apenas de dados de testes fictícios para comprovar a validade e utilidade do sistema, outros testes ainda deverão ser conduzidos para comprovar sua eficácia em situações reais de utilização. O Quadro 8 resume os principais estudos levantados sobre o DS:

Quadro 8 – Pesquisas com aplicações para o DS.

Fonte	Nome da Aplicação ou estudo	Principais características
LEIFLER e ERIKSSON, 2008.	-	- Estudo do DS dentro do contexto do comando militar;
DEMARTINI e NIEDERÉE, 2008.	-	- Sistema que permite encontrar especialistas analisando o conteúdo indexado por um DS chamado BEAGLE;
DEBATTISTA e ABELA, 2011.	BIZZILA	- Sistema que possibilita criar tarefas colaborativas e facilita a encontrar possíveis colaboradores de acordo com a tarefa necessária.

3.5. Desktops Semânticos Analisados

Para a criação do WS, foram analisados alguns DSs que poderiam ser utilizados na pesquisa aqui proposta. A ideia é utilizar o DS escolhido para gerenciar os artefatos manipulados referentes à troca de arquivos, *e-mails*, agendamento de reuniões e demais tarefas inerentes ao gerenciamento do P.I.M.. Foram analisados três DSs que reúnem as condições para servir aos propósitos e objetivos da pesquisa aqui desenvolvida, sendo suas particularidades explicitadas nos itens a

seguir, bem como a definição do NEPOMUK para realizar o gerenciamento do P.I.M. dos usuários na concepção do WS.

3.5.1. HAYSTACK

Trata-se de um projeto do *Massachusetts Institute of Technology* (MIT)[14] que é baseado nos conceitos da Web Semântica para criar um "pesquisador" de informações pessoais. O HAYSTACK (KARGER *et al.,* 2005) unifica os diferentes tipos de manuseio de informações semiestruturadas, de modo que a representação destes dados apresentada ao usuário seja de uma forma que facilite o processo de interação humana. As primeiras publicações que tratam sobre o HAYSTACK datam de 1999 ((ADAR *et al.*, 1999) e (LISANSKIY, 1999)).

Apesar de ter sido desenvolvido antes do surgimento do conceito do DS, ele pode ser considerado como tal. Um dos objetivos do HAYSTACK é tornar-se uma aplicação que trata do gerenciamento de PIM. Semelhante a outro PIM, chamado *Chandler* (O.S.A.F., 2011), o HAYSTACK unifica o manuseio de diferentes tipos de informações não estruturadas. Estas informações são representadas em RDF e apresentadas aos usuários de forma configurável e legível. Sua arquitetura em camadas estratifica as funções do sistema, colocando no primeiro nível os servidores que tratarão das mensagens do sistema de arquivos. No nível imediatamente superior, são armazenadas as transformações que estes arquivos passam para serem interpretados. Antes de estes arquivos serem apresentados aos usuários na última camada (*User Interface*), a camada de agentes é a responsável pelo tratamento das requisições dos usuários.

3.5.2. IRIS

IRIS é uma aplicação que também utiliza os conceitos do DS, ao permitir que os usuários criem um mapeamento por meio de seus objetos do desktop (CHEYER *et al.*, 2005). IRIS inclui aprendizagem de máquina para ajudar a automatizar esse

[14] http://groups.csail.mit.edu/haystack/

processo. Ela fornece a possibilidade de navegação contextual, com relacionamentos baseados na estrutura por meio de um conjunto extensível de aplicações de escritório, incluindo um calendário, navegador de arquivos, cliente de *e-mail* e também um cliente de mensagens instantâneas, conforme pode ser visto na Figura 3.

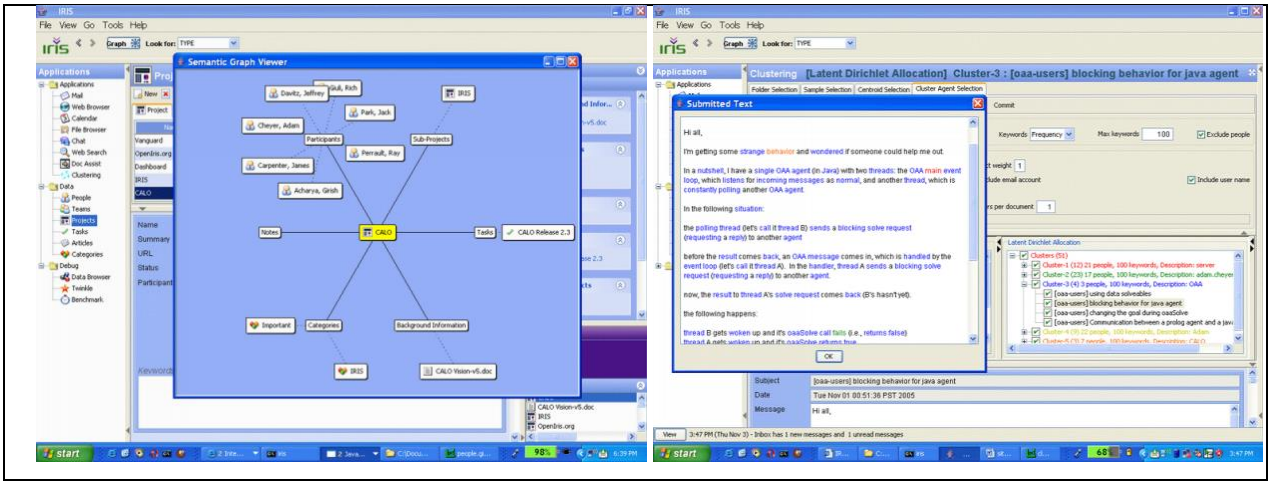


Figura 3 – Um grafo semântico e análise de um *e-mail* no IRIS

Fonte: `http://www.openiris.org/screenshots`

Os autores definem esta estrutura como sendo um *shell* que encapsula todas as aplicações que serão utilizadas no IRIS (*e-mail*, *web browser*, calendário, chat, gerenciador de arquivos e um editor de dados) e assim o IRIS integra todos os aplicativos necessários para sua utilização em um único local.

3.5.3. NEPOMUK

O projeto NEPOMUK (*Networked Environment for Personalized, Ontology-based Management of Unified Knowledge*) ((GROZA *et al.,* 2007) e (BERNADI *et al.,*2008)), é uma das ferramentas mais completas e que deu origem a diversas outras e embasou vários trabalhos sobre o assunto. Desenvolvido por um consórcio de pesquisadores, representantes da academia e desenvolvedores de software, que tem dirigido esforços no desenvolvimento do *Social Semantic Desktop*, que visa a dar significado semântico as informações do espaço de trabalho dos computadores dos usuários (seus “desktops”), para que estes possam interconectar e trocar

informações com os desktops de outros usuários (DECKER e FRANK, 2004). Existem versões para instalação em diversos sistemas operacionais e também uma versão integrada ao ambiente gráfico KDE, conforme se pode notar na Figura 4.

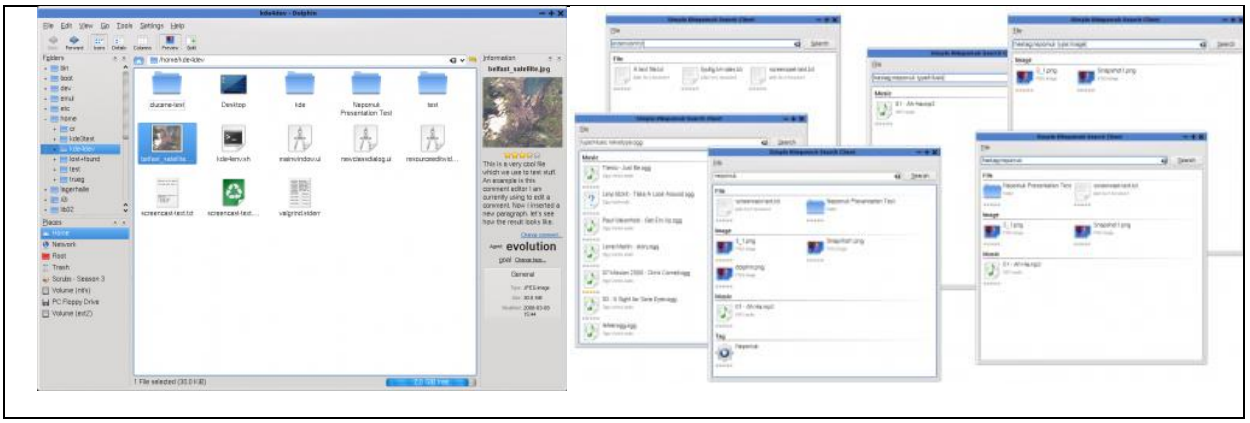


Figura 4 – Gerenciador de arquivos Dolphin, do KDE-4, integrado com o Nepomuk.

Fonte: `http://nepomuk.kde.org/discover/user`

Da mesma forma que os demais DSs, o NEPOMUK interliga dados de diferentes aplicações do desktop, utilizando metadados semânticos armazenados em formato RDF. A Figura 5 apresenta uma visão geral da arquitetura em camadas do NEPOMUK. Para apoiar a comunicação entre os usuários, a camada mais baixa é a camada Comunicação de Rede, que fornece um sistema que é responsável pela distribuição dos eventos que ocorrem no NEPOMUK. Acima desta camada, o NEPOMUK disponibiliza os serviços centrais do aplicativo.

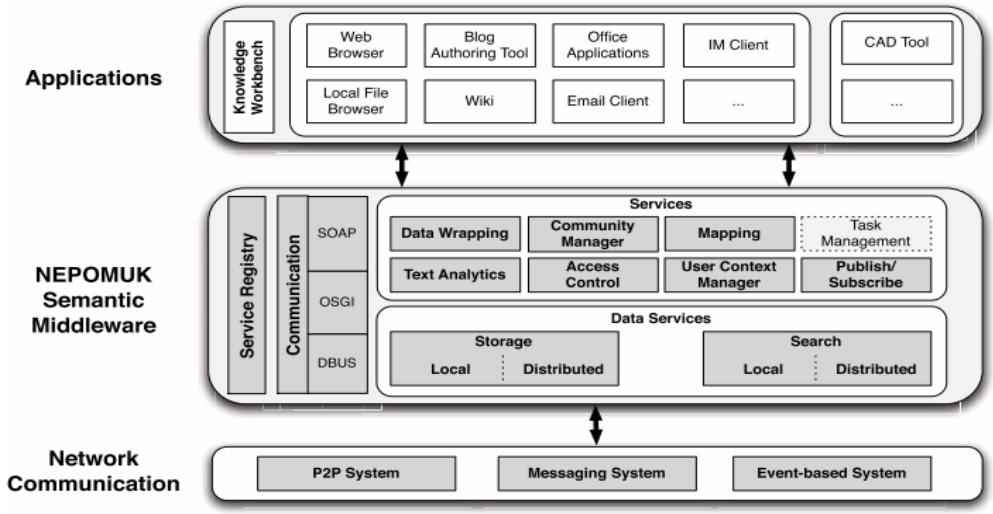


Figura 5 – Arquitetura em Camadas do NEPOMUK.

Fonte: (GROZA *et al.*, 2007)

3.5.3.1 DEFINIÇÃO DO NEPOMUK PARA GERENCIMANENTO DO P.I.M.

Dos desktops semânticos analisados, o NEPOMUK foi escolhido para realizar o gerenciamento do P.I.M. dos usuários que trabalham em projetos de desenvolvimento de software. Sua escolha é justificada para gerir este tipo de artefato pelos seguintes motivos:

- É uma ferramenta *open-source* e que está à disposição da comunidade que desenvolve software com toda a documentação disponível (em JavaDoc) e também o conjunto de materiais necessários para realizar melhorias acessíveis no endereço eletrônico (http://dev.nepomuk.semanticdesktop.org/repos/);
- Disponibilização de versões de avaliação e para desenvolvimento do DS NEPOMUK. Estas versões estão disponíveis em ambientes Windows, MacOs e também para Linux. Estas versões podem ser encontradas em (http://dev.nepomuk.semanticdesktop.org/download/); além disto, o NEPOMUK está integrado ao ambiente KDE para o Linux, fazendo parte dos itens que compõem o sistema operacional. O endereço eletrônico (http://nepomuk.kde.org/) mantém todas as informações sobre o NEPOMUK na versão para o KDE;
- O NEPOMUK foi desenvolvido utilizando as ferramentas Eclipse e Java, mesmas ferramentas que foram definidas para a criação do módulo de acompanhamento da codificação do WS. Na página do consórcio NEPOMUK encontra-se um suporte para estas ferramentas, (http://dev.nepomuk.semanticdesktop.org/wiki/EclipseDevelopment).
- É possível interligar as diferentes áreas de trabalho dos usuários por meio do componente denominado SOCIAL do NEPOMUK. Isto permite que os usuários realizem conexões entre seus DeSS para intercâmbio de informações, formando uma rede segura entre os participantes.

Estas características do DS NEPOMUK foram determinantes para sua escolha como ferramenta de suporte para a criação do WS em comparação as demais analisadas, e os experimentos de avaliação do DS NEPOMUK, bem como

os resultados e um tutorial para sua configuração e instalação para os interessados em reproduzir estes experimentos estão disponíveis no Apêndice 4 deste documento.

A principal função do NEPOMUK é a de realizar o rastreamento semântico dos artefatos, tais como planilhas de cálculo, e-mails, documentos de textos, imagens, figuras e outros. Este tipo de rastreamento é possível pela própria estrutura do NEPOMUK, que é composto por diversas ontologias interligadas, conforme apresenta a Figura 6. Estas ontologias monitoram todas as atividades no *desktop* do usuário que o utilizará como ferramenta de comunicação e intercâmbio de informações. A fundação OSCAF, resultado do projeto de NEPOMUK, é a instituição responsável por manter essas ontologias, e ela disponibiliza um projeto no SOURCEFORGE (`http://sourceforge.net/projects/oscaf/)`, que tem o propósito de manter e desenvolver colaborativamente estas ontologias, disponibilizando aos interessados todas as informações necessárias para tal fim.

Na Figura 6, pode-se atentar para as cinco ontologias que a integram. A NIE (*NEPOMUK Information Element)* é a ontologia agregadora do conjunto de ontologias que fornecem o vocabulário para descrever os elementos de informação que são comumente presentes nos desktops dos usuários. Um destes componentes é o NFO (*NEPOMUK File Ontology)*, ontologia que fornece o vocabulário para expressar informações extraídas de várias fontes. Estas fontes incluem arquivos, partes de software e hosts remotos, onde os arquivos e outros recursos de desktop são entendidos como sequências de bytes armazenados em um sistema de arquivos ou em uma rede. A NMO (*NEPOMUK Messaging Ontology)* estende a NIE para o domínio das mensagens, que incluem e-mails e mensagens instantâneas, e é o componente responsável por agregar este tipo de conteúdo no NEPOMUK. Já a NCAL (*NEPOMUK Calendar Ontology)* fornece o vocabulário para descrever dados de calendário, tais como eventos, tarefas e datas, que são uma parte importante do conjunto de informações referentes às atividades que os usuários geralmente armazenam em um desktop. Por fim, a NCO (*NEPOMUK Contact Ontology)* descreve as informações dos contatos dos usuários e também é integrada a NIE.

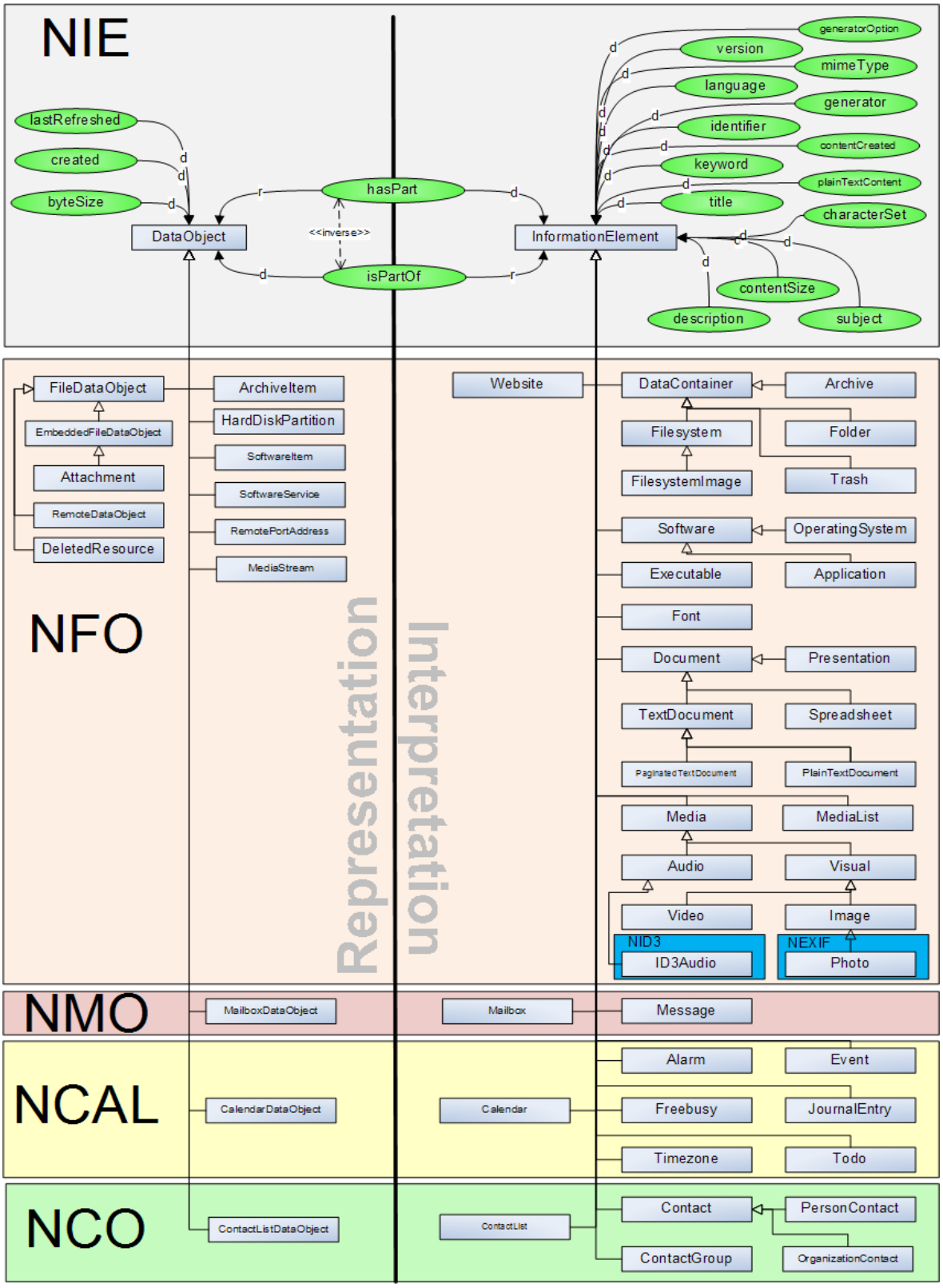


Figura 6 – Ontologias do NEPOMUK.

Fonte: (http://oscaf.sourceforge.net/)

Estas ontologias são o principal componente do NEPOMUK, e por meio delas é possível representar o conhecimento deste domínio e realizar as inferências necessárias para obter as meta-informações que o NEPOMUK disponibiliza, formando a base para o modelo de dados que representa os conceitos e os relacionamentos das informações que os usuários usualmente manipulam em seus desktops.

O Quadro 9 exibe, no lado esquerdo, parte da especificação da NFO, que trata da criação e último acesso a arquivos no NEPOMUK. Se for necessário o acesso aos dados de forma direta (por meio de codificação), o lado direito do Quadro 9 mostra como isto pode ser realizado para um arquivo específico por intermédio desta ontologia, utilizando-se do domínio *FileDataObject* da Ontologia, com o objetivo de buscar um arquivo (IMG 0012.JPG) em um determinado local dentro do KDE em que o usuário marcou uma *tag* (Summer, 2008).

Quadro 9 – Detalhe da NEPOMUK ONTOLOGY NFO.

<table>
<tr>
<td>NEPOMUK Ontology

nfo:fileCreated

<table>
<tr><td>Label</td><td>fileCreated</td></tr>
<tr><td>Description</td><td>File creation date</td></tr>
<tr><td>Domain</td><td>nfo:FileDataObject</td></tr>
<tr><td>Range</td><td>xsd:dateTime</td></tr>
<tr><td>Maximum Cardinality</td><td>1</td></tr>
<tr><td>Super-properties</td><td>nao:created, nao:modified, nie:modified, nie:created (direct), nao:annotation</td></tr>
<tr><td>Sub-properties</td><td></td></tr>
</table>

nfo:fileLastAccessed

<table>
<tr><td>Label</td><td>fileLastAccessed</td></tr>
<tr><td>Description</td><td>Time when the file was last accessed.</td></tr>
<tr><td>Domain</td><td>nfo:FileDataObject</td></tr>
<tr><td>Range</td><td>xsd:dateTime</td></tr>
<tr><td>Maximum Cardinality</td><td>1</td></tr>
<tr><td>Super-properties</td><td></td></tr>
<tr><td>Sub-properties</td><td></td></tr>
</table>
</td>
<td>

```
<file:/Pictures/IMG_0012.jpg>
  a nfo:FileDataObject ;
  nfo:fileUrl <file:/Pictures/IMG_0012.jpg> ;
    nao:hasTag <nepomuk:/tags/Summer08> ;
    nao:relatedTo <nepomuk:/KDE> .

<nepomuk:/KDE>
  a pimo:Project ;
  nao:prefLabel "KDE" .

<nepomuk:/tags/Summer08>
  a nao:Tag ;
  nao:prefLabel "Summer 08" .
```

</td>
</tr>
</table>

As ontologias, porém, não se limitam as apresentadas na Figura 6. Existem outras que fornecem, por exemplo, os modelos mentais dos próprios usuários, linguagens de representação, conteúdo multimídia, todas também disponíveis no SOURCEFORGE. Para a pesquisa aqui conduzida, estas ontologias são utilizadas dentro do escopo do DS NEPOMUK, para gerir e acompanhar todos os processos referentes à utilização dos artefatos não ligados diretamente na programação do código-fonte pelas equipes de desenvolvimento.

3.6. O Framework Hackystat

Dentre as opções existentes para coletar os dados advindos da programação do código-fonte, foi escolhido o framework HACKYSTAT. Este framework, disponível em (http://code.google.com/p/hackystat/), é uma ferramenta *Open Source* para automatizar a coleta e análise de dados do processo de desenvolvimento de software e de dados de produto ((JOHNSON *et al.*, 2003), (JOHNSON *et al.*, 2004)). Usuários do HACKYSTAT podem anexar sensores em ferramentas de desenvolvimento que, de forma não intrusiva, podem recolher e enviar dados sobre o processo de desenvolvimento para um *web service* chamado *Hackystat SensorBase for Storage*. Este componente pode ser consultado por outro *web service* para formar abstrações de nível mais elevado dos dados armazenados, ou gerar visualizações dos dados brutos. O HACKYSTAT tem sido usado por pesquisadores e organizações industriais para compreender melhor o desenvolvimento de software. Hochstein e colegas (HOCHSTEIN *et al.*, 2005), realizaram várias experiências sobre a produtividade de programadores realizado pelo grupo de trabalho DTWG (*Development Time Working Group)*, que faz parte do HPCS (*High Productivity Computing Systems*). Os profissionais do DTWG usaram o HACKYSTAT para registrar as atividades de desenvolvimento dentro dos IDEs e também em linha de comando, integrado o HACKYSTAT em seu conjunto de ferramentas chamado UMDInst (UMD *Instrumentation Package*). Morisio e seus colegas também utilizaram o HACKYSTAT para monitorar uma sala de aula de um curso de programação, a fim de obter dados quantitativos sobre o comportamento de dados de testes e sua relação com a qualidade de software. Os resultados deste trabalho estão disponibilizados em (MORISIO *et al.,* 2004).

Além destes, existem outros trabalhos que referenciam ou utilizaram o HACKYSTAT. Em (JOHNSON, 2007), o autor apresenta outros cinco *frameworks* semelhantes à sua pesquisa e levanta 12 requisitos e as implicações que estes têm para o desenvolvimento de novos frameworks, além de apontar aos pesquisadores que tenham interesse em aprimorar este tipo de pesquisa, com seis direções para trabalhos futuros que podem ser investigados. Mais um trabalho que, apresenta uma pesquisa onde alunos de uma instituição de ensino podem aprender sobre nove diferentes projetos empíricos de tratamento de saúde intensiva é o apresentando em

(JOHNSON e ZHANG, 2009). Nessa pesquisa, os sinais vitais de pacientes foram monitorados e avaliados. Este nove projetos foram colocados no que foi denominado como uma unidade virtual de cuidados intensivos, onde o software denominado ICU (*Intensive Care Units),* forneceria informações com os dados sendo coletados pelo *framework* HACKYSTAT. Por fim, em (JOHNSON *et al.,* 2009), os autores adaptaram uma nova versão do HACKYSTAT, direcionando para uma arquitetura orientada a serviços chamada de SOA (*Service-Oriented Architecture*), permitindo assim que outras linguagens além do JAVA pudessem ser utilizadas, além de permitir que plataformas sociais (como FACEBOOK e canais de comunicação como TWITTER) também possam ser monitorados. Todos os trabalhos citados auxiliaram na decisão de utilizar o HACKYSTAT nesta pesquisa. Outros fatores que contribuíram para a escolha deste framework para monitorar e coletar os dados são listados na sequência:

- É um *framework* de código aberto (sob licença GNU GPL), e que possui suporte pela comunidade de desenvolvedores JAVA;
- É mantido pela empresa GOOGLE, e patrocinado por diversas instituições e empresas, entre elas a SUN MICROSYSTEMS, *The National Science Foundation*, NASA e a IBM;

Com a definição da utilização do HACKYSTAT, testes de validação foram conduzidos, e os resultados destes testes são detalhados no capítulo 05, onde experimentos utilizando o HACKYSTAT foram executados e são descritos.

3.7. Ferramentas de Apoio ao *Awareness*

O trabalho de Storey e colegas (STOREY *et al.,* 2005) apresenta um *survey* que analisa doze ferramentas de *awareness*. Também mostra a criação de um *framework* para auxiliar os demais pesquisadores a comparar e compreender instrumentos de *awareness* para apoiar as tarefas de desenvolvimento de software em cinco dimensões que os autores consideraram principais: intenção, interação informação, apresentação e eficácia.

Braun e colegas (BRAUN *et al.,* 2007a) criaram o modelo KAP (*Knowledge-Added Process*), utilizando-se dos conceitos do DS criados em (SAUERMANN,

2005a) para gerenciamento pessoal de conhecimento (*Personal Knowledge Management – PKM)* e aprimoraram a forma da análise de processos de trabalhos científicos no que eles chamaram de *e-science* (pesquisa em meios de conhecimento intensivo), adicionando o a*wareness* para auxiliarem as pessoas a tomarem ciência do trabalho de outros e que possuam afinidade entre os trabalhos realizados por ambos. Como resultado, os autores descobriram que as abordagens tradicionais não podem suportar eficientemente os processos de conhecimento devido ao seu alto grau de variabilidade e imprevisibilidade. Com base no *awareness*, os autores elaboraram um novo paradigma de processos de apoio que auxilia a encontrar pesquisas e especialistas no domínio desejado. Ainda, na análise realizada é introduzido o conceito do "*Socially-Aware Deskotp",* para ajudar a encontrar pesquisadores que possuem trabalhos e interesses em comum, a fim de auxiliar de forma eficaz os processos de colaboração científica.

Outra ferramenta disponível é conhecida como PALANTÍR (SARMA*,* 2008). Ferramentas como a PALANTÍR auxiliam aos desenvolvedores a ficaram cientes de quem está trabalhando em um artefato específico, fornecendo a eles detalhes das atividades atuais de outros desenvolvedores. A principal funcionalidade da ferramenta é fornecer meios para que o *awareness* seja efetivo em equipes de desenvolvimento distribuído de software. O principal objetivo é permitir aos participantes da equipe evitar o que a autora define como "conflitos de codificação". Como a PALANTÍR extrai informações diretamente dos Controladores de Versões (CVS, do inglês *Control Version System)* mais populares disponíveis, a ferramenta permite aos participantes estarem cientes de possíveis conflitos na codificação e assim resolvê-los antes de finalizar seu trabalho, possibilitando um ganho de tempo e de qualidade do software produzido.

Em (BULEJ *et al.,* 2012) os pesquisadores fornecem um *framework* para teste de desempenho do *awareness* no domínio de sistemas baseados em componentes. Nesse trabalho, o framework apresentado testa as condições para aliviar a carga de coletar dados de desempenho e proporciona uma forma unificada de capturar estes dados com base no desempenho da Lógica e Simulação Estocástica. São apresentados múltiplos casos de uso para avaliação do framework, que permitem não só personalizar a forma como os dados de desempenho são coletados, mas

também a definição de métricas. A principal contribuição, segundo os autores é a forma de expressar e avaliar as declarações de desempenho para efeitos de adaptação dinâmica - mesmo em casos de uso bastante complexos através de simples combinação de fórmulas e fontes de dados associados.

Mais uma pesquisa que trata sobre *awareness* é a desenvolvida por (DUQUE *et al.,* 2012), onde os autores propõe uma arquitetura para ser utilizada por desenvolvedores de *groupware* como um guia para a integração dos subsistemas de análise. Esta mesma arquitetura permitiu projetar o sistema de *groupware* chamado COLLECE, que apoia práticas de programação. São diversos subsistemas que apoiam diversas atividades, tais como: Identificação, Meta-informação, *Awareness*, Análise, Observação e Aplicação. Estes subsistemas interagem uns com os outros para apoiar as funções básicas de qualquer sistema de groupware e analisar o trabalho colaborativo realizado pelos usuários.

O trabalho de (TERUEL *et al.,* 2014) define um *design pattern* que pode ser aplicado no projeto de sistemas para CSCW e que fornece suporte para a área de *Workspace Awareness (WA),* conforme definido em (GUTWIN e GREENBER, 1996)

A aplicação do trabalho dos autores pode ser como ponto de partida para o desenvolvimento de sistemas complexos de CSCW. Como prova de conceito, um modelo foi implementado para suporte nas linguagens C# e VISUAL STUDIO.

Uma pesquisa que verifica a utilização de componentes da Web 2.0 foi realizada por (TELLIOGLU e DIESENREITER, 2013). Neste trabalho o objetivo foi entender esses mecanismos utilizando conceitos relevantes para empresas como o *awareness*. O autor realizou-se uma diferenciação entre os mecanismos de *awareness* do sistema e o entendimento dos usuários para estes, alinhando os conceitos para componentes da *Enterprise 2.0.*

Um estudo que desenvolveu um modelo de promoção do awareness baseado em eventos para atividades colaborativas distribuídos em sistemas pervasivos foi realizado por (CAI e YU, 2014). Neste trabalho, sugeriram-se como sistemas computacionais em ambientes pervasivos poderiam desempenhar um papel na utilização do awareness nas apresentações formais de atividades e dependências de trabalho cooperativo. Trabalhos futuros ainda deverão ser conduzidos para validar o modelo proposto pelos autores.

O trabalho de (HAJIZADEH et al., 2014) considera a retenção das atividades que geram ações de *awareness* dos colaboradores fundamental durante o trabalho colaborativo, inclusive durante as atividades de visualização de colaboração, e articuladamente quando os colaboradores estão localizados remotamente, sendo importante conhecer no que todos estão trabalhando, a fim de evitar a duplicação de esforços, compartilhar resultados relevantes em tempo hábil e construir sobre os resultados sobre o trabalho individual de cada um. Neste sentido, o trabalho levado a termo pelos autores realizou um estudo para responder a estas questões no contexto da visualização colaborativa distribuída de dados tabulares. Os resultados apontam que a seleção persistente permitiu alguma *awareness* nas atividades dos colaboradores, causando o mínimo de interferência com o trabalho independente que cada colaborador realiza. O Quadro 10 sintetiza as ferramentas de *awareness* analisadas e as principais características de cada uma.

Quadro 10 – Ferramentas de *awareness* analisadas e suas principais características.

Fonte	Nome da Ferramenta	Principais Características
BRAUN *et al.,* 2007a.	KAP	- Pesquisa em meios de conhecimento intensivo utilizando conceitos dos DSs e *awareness* para auxiliar os usuários a tomarem ciência dos trabalhos de seus pares nos meios de conhecimento de colaboração científica;
SARMA*,* 2008.	PALANTÍR	- *Awareness* para o grupo de desenvolvimento com a finalidade evitar conflitos de codificação;
BULEJ *et al.*, 2012*.*	-	- Framework para avaliar a captura de dados de desempenho baseados em lógica estocástica;
DUQUE et al., 2012.	COLLECE	- Sistema que inclui o gerenciamento de usuários e fornece suporte para tarefas colaborativas dentro de espaços de trabalho compartilhados.
TERUEL *et al*., 2014.		- Define um *design* pattern para reutilização na refatoração de código, reduzindo o tempo gasto neste processo, em equipes que utilizam o CSCW nos seus processos, principalmente para sistemas complexos de CSCW.
TELLIOGLU e DIESENREITER, 2013.	-	- Trabalho que realiza uma investigação conceitual e empírica em empresas de desenvolvimento de software para verificar componentes da Web 2.0 que utilizam awareness.
CAI e YU, 2014.	D.A.C.E	- Proposta de modelo de promoção do awareness com base em eventos na computação pervasiva.
HAJIZADEH et al., 2014.	-	- Comparação de três métodos de visualização de awareness em multi-visões (*brushing & linking, selection, and persistent selection*) como forma de fornecer informação sobre as atividades dos colaboradores.

3.8. Ferramentas de Verificação e Validação de Software

Um estudo empírico sobre o ensino de V&V em um curso de engenharia de software é apresentado no trabalho de (LI e BOEHM, 2011), cujo objetivo é auxiliar os alunos a obter as habilidades na prática da V&V por meio de inspeções, comitês de revisão de arquitetura, critérios de classificação, acompanhamento dos seus planos de gestão de qualidade de experiências de projetos e avaliações de clientes analisados. Os resultados, segundo os autores, auxiliam os alunos a obter a prática do processo V&V integrando as várias ferramentas utilizadas.

O trabalho de (WANG, 2011) descreve um framework para o desenvolvimento de um modelo de classificação e também uma aplicação piloto do sistema para medir e controlar o processo de V&V. As investigações indicam que a abordagem proposta é capaz de fornecer diagnósticos de processos que refletem os problemas da tecnologia, além de identificar os potenciais diagnósticos de melhoria. Ao aplicar o framework em um caso concreto que utilize o processo V&V, pode-se determinar um esquema específico do aplicativo que está em análise.

Mais um trabalho que propõe um framework chamado de MAFSE (Model-bAsed Framework for Software vErification), que realiza checagem de modelos por meio da verificação de software é proposto por (QUAN *et al.*, 2011). O framework foi testado com alguns dados em escala de laboratório e sua aplicação é voltada à engenharia de software na área industrial, aproveitando-se da vantagem de que a abordagem da checagem de modelos possui de produzir contraexemplos quando detecta problemas indesejáveis. O framework utiliza estas informações para melhorar os resultados obtidos com aplicação de métodos apropriados nos programas de software testados.

A EMFtoCSP é uma ferramenta para a verificação automática de modelos MDE (*Model Driven Engineering*) e a mesma é apresentada em (GONZALES *et al.,* 2012). Como o desenvolvimento de software baseado em MDE tem o foco na concepção e criação de modelos semiautomáticos e também na geração de código-fonte baseado nestes modelos, a ferramenta verifica propriedades que são transformadas em um problema de satisfação de restrições (CSP, do acrônimo em inglês de *Constraint Satisfation Problem*), que checa se existe uma solução ou não em CSP. Se uma solução é encontrada, a ferramenta realiza a validação da

instância do modelo encontrado para certificá-lo em algumas propriedades pré-definidas. A ferramenta é integrada a IDE Eclipse, e, portanto disponível aos interessados na forma de livre acesso.

A ferramenta PINCETTE (CHOCKLER *et al.,* 2013), integra a análise dinâmica e estática de software conjuntamente para analisar sistemas de controle de redes, tais como de distribuição de água, eletricidade ou telecomunicações que necessitam passar por atualizações de software. Estas atualizações, se não forem bem executadas, podem inserir novos erros ou mesmo interromper funcionalidades já existentes e funcionais, e a correção destes problemas pode ser altamente custosa. A combinação das duas técnicas (dinâmica e estática) gera um novo campo de estudo, pois originalmente, as duas técnicas foram desenvolvidos como domínios separados. A PINCETTE auxilia a aumentar a confiança nas atualizações de software nas redes, justamente por integrar vários métodos de V&V em uma única ferramenta com objetivos específicos de utilização.

O trabalho de (OLAMIDE e KABA, 2013) apresenta uma técnica de verificação baseada em um modelo construído sobre a utilização seletiva e pragmática de métodos formais, utilizando para isto modelo simplificado ferramentas de verificação, que se concentram na detecção de erros. Os autores enfatizam a importância da V&V em Modelagem e Simulação, pois entendem que é importante integrar as atividades de V&V com o ciclo de vida de desenvolvimento do modelo, de forma organizada e eficiente e bem documentado.

O artigo de (WANG, 2013) apresenta uma abordagem de caracterização para o desenvolvimento de um catálogo de técnicas de V & V que encapsula as técnicas disponíveis com as informações sobre as suas condições de aplicação. Além disto, inclui um planejamento e estratégia de adaptação para a seleção de técnicas de V&V apropriadas aos seus propósitos.

Em (HU, 2014) é proposto um framework que é composto por mecanismos de *feedback* interno e externo, em que os padrões industriais são atualizados e os requisitos do cliente são incorporadas, permitindo que o teste a ser realizado seja realizado em conformidade com o que se espera do mesmo para aplicações em larga escala. O Quadro 11 mostra as ferramentas analisadas e suas principais características.

Quadro 11 – Características das Ferramentas Analisadas.

Fonte	Nome da Ferramenta ou Modelo	Principais Características
LI e BOEHM, 2011.	-	- Estudo empírico que utiliza diversas técnicas no ensino da V&V que auxilia os alunos a obter prática no processo de verificação e validação de software;
WANG, 2011.	-	- Framework de classificação que define quatro atributos para verificação de software; - Realiza o controle do processo de verificação e validação de software;
QUAN *et al.*, 2011.	MAPFSE	- Framework voltado para a engenharia de software na área industrial; - Voltada para aproveitar os contraexemplos na detecção de problemas na verificação de software;
GONZALES *et al.,* 2012.	EMFTtoCSP	- Utiliza a CSP para verificar se existe uma solução para um determinado problema encontrado; - Integrada a IDE de desenvolvimento ECLIPSE;
CHOCKLER *et al.,* 2013.	PINCETTE	- Atua em sistemas de redes que necessitam de atualização nos sistemas de software; - Integra vários métodos da V&V; - Utiliza simultaneamente as técnicas estáticas e dinâmicas para avaliar um software.
OLAMIDE e KABA, 2013.	-	- Framework que constitui um modelo de refinamento para o processo iterativo que ajuda a melhorar simulações, corrigir erros ou adaptar itens aos requisitos de mudança de contexto.
WANG, 2013.	-	- Catálogo de técnicas com informações sobre a sua aplicabilidade e os custos necessários para efetiva implantação.
HU, 2014.	-	- Modelo de negócio para *cloud computing* para sistemas de larga escala utilizando técnicas de V&V.

3.9. Acompanhamento da Codificação em SVN

Pesquisas realizadas em repositórios de artigos e bases científicas até o momento em que este documento foi redigido, não mostraram a existência na literatura de uma ferramenta, método ou conceito que apresente as características propostas no WS. Algumas ferramentas apresentam funções que auxiliam a monitorar, e até mesmo criar código fonte, mas nenhuma com as características semelhantes aos objetivos propostos nesta pesquisa. Por exemplo, existem ferramentas que auxiliam a criar, a partir da análise realizada, parte da codificação dos programas, gerando o código fonte diretamente dos diagramas da UML. É comum este tipo de software também gerar as classes da modelagem, analisando o

código fonte criado. Estas funcionalidades são possíveis de serem utilizadas por meio da técnica da engenharia reversa, que fornece os dados necessários para esta tarefa.

Outro tipo de ferramenta que realiza uma forma de controle de código fonte são os softwares de controle de versão, do tipo SVN (*Subversion*). Os SVNs têm sido a principal abordagem para gerenciar os artefatos de projetos criados com ferramentas que não suportam múltiplos usuários. Eles também armazenam documentos, planos, projetos e outros artefatos em repositórios da equipe, de modo a disponibilizá-los de forma consistente a outros membros, o que permite acesso controlado aos materiais disponibilizados, além de apresentar uma visão do projeto para todos os participantes. Para manter as versões dos softwares, o SVN centraliza todo o código em um repositório, que é basicamente um banco de dados com todo código da aplicação em desenvolvimento. Alterações nunca são feitas diretamente no repositório, mas em cópias locais de trabalho (chamadas de *working copy*), e a partir destas cópias, as mudanças de código são enviadas para os repositórios, por meio de comandos especiais chamados *commits*.

Para trabalhar na programação de um determinado código, deve-se solicitar ao repositório uma cópia do mesmo, realizando um c*heckout*. Esta cópia fornecida é a *working copy* de trabalho. Ao término, as modificações devem ser enviadas para o repositório por meio de um *commit*, e todo *commit* gera uma nova revisão do código. Como cada *commit* gera uma revisão e as revisões não são enviadas automaticamente para todas as cópias de trabalho, se algo foi alterado enquanto outra pessoa estiver trabalhando, é necessário que os envolvidos atualizem suas cópias de trabalho com o código existente no controle de versões. Como o repositório é responsável por manter a integridade do código, não são permitidas revisões anteriores a que está atualmente sendo utilizada por meio de um *commit*, pois isto pode gerar conflitos. Caso duas pessoas alterem o mesmo arquivo, o sistema de SVN irá gerar provavelmente um *diff* (diferenças encontradas na cópia de trabalho) entre os arquivos em questão. Se as diferenças não estiverem nas mesmas linhas do código, é provável que o próprio SVN resolva o conflito; caso contrário, uma intervenção humana será necessária para corrigir o problema.

A principal diferença do que os SVNs realizam com o WS reside no fato de que os primeiros controlam as versões finais de cada usuário que são produzidas durante o seu trabalho. O SVN pode indicar, por exemplo, qual programador alterou determinada parte do código fonte e quando isto foi feito, permitindo assim rastrear todas as alterações realizadas pelos usuários. Como analogia, pode-se dizer que a *working copy* do usuário é como uma fotografia do estado do código no repositório no momento que ele inicia seu trabalho, e o *commit* é o retorno desta fotografia ao repositório com as alterações efetuadas. Tudo o que foi realizado durante este processo não é monitorado pelo SVN, pois sua função é guardar as diferenças do que existia anteriormente no código-fonte, com as alterações realizadas até a finalização do trabalho por parte de cada usuário, e assim poder atribuir a cada um sua suas responsabilidades nas alterações do código-fonte. O WS atua enquanto o código está sendo desenvolvido, monitorando as ações dos programadores durante a codificação do programa, tais como a criação/alteração de classes, métodos ou atributos, sendo esta a principal diferença do trabalho com um SVN do trabalho do WS e uma integração dos dois sistemas é considerada em um trabalho futuro.

Por fim, existem trabalhos que tratam sobre a visualização de software, que é a área que utiliza símbolos para representar diversas propriedades dos sistemas de software com o objetivo de revelar padrões e comportamentos que de outra forma poderiam permanecer ocultos (PETRE, 2010). Um trabalho representante deste grupo é o desenvolvido por (NOVAIS *et al.,* 2011), que apresenta a SOURCEMINER, um ambiente extensível de visualização de software baseado em múltiplas visões coordenadas entre si para a execução de atividades de compreensão de software. Este ambiente foi implementado como um *plug-in* do ECLIPSE para representar sistemas de software em Java por meio de visões coordenadas entre si. Como o foco do trabalho é a visualização estática, o SOURCEMINER não analisa a execução do software, mas a sua estrutura estática e o relacionamento entre as entidades que o compõem.

3.10. Considerações Finais

Buscou-se mostrar neste capítulo, o estado da arte sobre diversas pesquisas referentes aos temas que abrangem este trabalho. Os trabalhos levantados foram principalmente aqueles sobre os objetos de interesse para a pesquisa aqui apresentada, a saber: desenvolvimento de software e ferramentas colaborativas, apoio a gestão de artefatos, *awareness*, verificação de software, além de algumas aplicações do DS. Estes temas foram pesquisados e apresentados nos diversos tópicos que compõem este capítulo, sendo mostrados trabalhos atuais de cada área relacionada.

No levantamento realizado, verificou-se a ausência de ferramentas similares ao WS. Até onde foi possível buscar, não foi encontrada nenhuma ferramenta ou método que apresenta as características que se pretende agregar ao WS, que é descrito em maiores detalhes no próximo capítulo.

Capítulo 4

O Workspace Semântico

Para desenvolver software, são utilizadas ferramentas que exigem dos participantes coordenação para desenvolver as tarefas necessárias, além de colaborarem de forma constante para centrar esforços e construir software que atenda aos requisitos dos usuários. Algumas ferramentas dão suporte a esta atividade, contribuindo para que o desenvolvimento seja realizado da forma mais produtiva possível. Dentre estas ferramentas podem-se citar: linguagens de programação, softwares de controle de versão, de gerência de configuração, de análise de projeto, de banco de dados, entre outras. Com esta variedade de ferramentas disponíveis, também se convive com vários problemas relacionados à questão tecnológica, como má arquitetura do sistema, códigos duplicados, mau uso das habilidades das pessoas envolvidas, entre outros fatores que podem ser citados. Estes problemas podem ser potencializados nos papéis dos desenvolvedores e dos gerentes de projeto, por serem os papéis que acumulam muitas atividades e que podem sofrer com maior intensidade a falta de conhecimento geral do andamento do projeto por parte de todos os envolvidos.

A concepção do Workspace Semântico (WS) surgiu da observação de algumas destas questões de processo e tecnológicas, mais especificamente de ferramentas para o desenvolvimento de software. Além disto, conforme apresentado

no Capítulo 2, pequenas equipes de desenvolvimento têm seu trabalho dificultado pela falta de ferramentas especialmente desenvolvidas para elas. No âmbito desta pesquisa, uma pequena equipe também possui até 10 participantes, como igualmente definido por Campagnolo e colegas em (CAMPAGNOLO *et al.,* 2009).

Destas observações, surgiu a percepção de desenvolver um ambiente integrador, cujo principal objetivo é centralizar os artefatos produzidos ao longo do projeto. Devido à dificuldade de monitorar estes artefatos, decidiu-se propor a concentração nas atividades realizadas por desenvolvedores, por exemplo, monitorando o código fonte para buscar informações que pudessem auxiliar os envolvidos no processo de criação de software a colaborarem de forma mais efetiva. Através deste monitoramento será possível acompanhar a execução do projeto que está sendo desenvolvido, por meio do acompanhamento das ações realizadas na programação, confrontando esta programação com o que foi definido na fase de projeto (especificamente em diagramas de classe UML representativos do sistema), e assim poder diminuir os problemas relatados no início deste capítulo. O monitoramento das ações realizadas pelo WS tem o potencial de prover as condições para que o acompanhamento do código ocorra de forma efetiva.

Desta forma, o objetivo deste capítulo é definir o WS, apresentando a visão geral para o seu funcionamento, a sua arquitetura e a forma como este conceito será implantado para realizar os testes iniciais de adequação e ações necessárias para a criação do protótipo.

Para obter dados e realizar uma sondagem da área de inserção em que o WS pode ser utilizado, uma pesquisa de campo foi realizada com desenvolvedores de software e também com gestores de equipes que desenvolvem software, com intuito de levantar dados para a execução da pesquisa aqui conduzida.

4.1. Pesquisa de Campo com Profissionais da Área de Informática

Esta pesquisa de campo (realizada por meio de um questionário - Apêndice 1) foi realizada com o objetivo principal de obter uma visão geral da área onde o WS irá atuar, envolvendo a forma como software é desenvolvido, especialmente o que é desenvolvido por pequenas equipes, conforme definido por (POLLICE *et al.,* 2004).

Este questionário foi disponibilizado de forma eletrônica, por meio do GOOGLE DOCS e ficou disponível pelo período de trinta dias para receber as respostas e considerações dos pesquisados.

Foram contatadas seis empresas, escolhidas dentre aquelas que possuem desenvolvimento de software e que trabalham com equipes de programação. Todas deram retorno positivo para a solicitação de realizar a pesquisa de campo.

O contato com as empresas que foram selecionadas para receber o convite para que seus funcionários respondessem os questionamentos foi feito na figura do proprietário ou diretor responsável, para que ele tomasse ciência do conteúdo do questionário, aprovasse a solicitação e só então pudesse distribuir aos seus colaboradores que desempenhassem a função de programador/gestor de projetos.

Projetou-se como meta satisfatória de retorno de respostas, a quantidade de 30 formulários. Para determinar este número, foi levado em consideração a quantidade total de pessoas disponíveis para responder ao questionário, estimada em torno de 120 (cento e vinte) pessoas. Este número foi obtido com os responsáveis de cada empresa consultada para verificar a possibilidade do pessoal da área técnica em responder ao questionário. No total, foram obtidas 54 (cinquenta e quatro) respostas. Ressalta-se que as empresas escolhidas para receber o convite e responder ao questionário eram na sua maioria dos estados do Paraná, Santa Catarina e São Paulo, algumas destas empresas com filiais em outros estados do país.

O questionário totaliza 21 perguntas. As pessoas tinham a possibilidade de verificar todas antes de iniciar as respostas. Além disto, elas eram informadas do tempo aproximado para responder (em torno de 5 minutos), dos objetivos do questionário e que o mesmo seria anônimo. Cada pergunta estava ligada a pelo menos um fundamento importante a ser conhecido para a criação do WS, como por exemplo, sobre a forma de desenvolvimento de software adotado no local de trabalho, o tamanho de equipe de desenvolvimento, os softwares utilizados, quais papéis eram desempenhados pelo respondente durante o trabalho e o seu conhecimento prévio de ferramentas similares que poderiam realizar o tipo de trabalho que o WS está se propondo a realizar. Sobre estes fundamentos, os

seguintes objetivos foram traçados e incluídos nas questões apresentadas para responder:

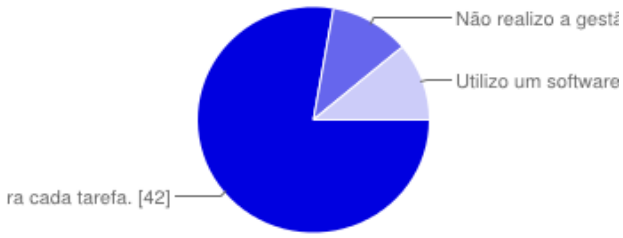
- Conhecer a forma e ferramentas de gestão dos artefatos produzidos durante o desenvolvimento de software nas empresas pesquisadas;
- Qual o papel desempenhado pelo profissional durante o desenvolvimento do sistema e se este papel pode variar durante a execução do projeto;
- Saber se a gestão do código-fonte é realizada por alguma ferramenta de apoio ou metodologia de desenvolvimento de software, e em caso positivo, qual seria;
- Conhecer o tamanho médio da equipe de desenvolvimento na qual as pessoas estão inseridas;
- Conhecer qual método/ferramenta é a mais utilizada; para troca de informação entre a equipe. Obter o tempo médio que os projetos de software levam para serem executados pela equipe;
- Saber se a equipe utiliza algum tipo de controle ou ferramenta que mostre se o código-fonte produzido está refletindo os diagramas projetados para o sistema;
- Verificar se alguma ferramenta de controle de projetos é utilizada, e em caso afirmativo, qual ferramenta;
- Saber a opinião sobre uma ferramenta que monitorasse o código-fonte, como por exemplo: a criação de classes, alteração de métodos e outras funcionalidades e que mostrasse o progresso do código-fonte desenvolvido, confrontando com um diagrama de classes (UML) do sistema, compartilhando estas informações com os outros membros da equipe.
- Classificar esta ferramenta descrita em opções que vão de “extremamente útil” até “intrusiva”, e conhecer o interesse em utilizá-la;
- Conhecer a opinião acerca das características que a ferramenta descrita anteriormente deveria possuir para ser considerada útil.

A pesquisa auxiliou no balizamento de algumas ações e ajudou a comprovar algumas suposições levantadas. Uma análise dos principais pontos é apresentada na sequência (o Apêndice 2 mostra todas as respostas e os gráficos da pesquisa gerados pelo Google Docs). Para a apresentação destes pontos principais e dos resultados mais relevantes da pesquisa, optou-se por mostrar as informações completas sobre cada questão, em um gráfico com as seguintes informações: a pergunta realizada, as opções disponíveis (quando aplicado), os resultados totais e os percentuais obtidos e também o gráfico das respostas obtidas. Adotou-se este formato por entender-se que condensaria as informações sobre cada questão e facilitaria o entendimento.

Uma questão que se buscava responder foi acerca da gestão dos artefatos (Gráfico 1). O objetivo foi verificar como os artefatos produzidos durante o desenvolvimento de um sistema eram geridos na empresa onde a pessoa que respondeu ao questionário trabalhava. A maioria das respostas aponta para o uso de ferramentas específicas para cada situação, mostrando que a integração não é priorizada. Apenas 11% utilizam uma ferramenta integrada. Outra questão (aberta) solicitava a indicação do nome da ferramenta que utilizava. As respostas variam entre CONTOUR, REQPROJ, GOOGLE DOCS, TORTOISESVN, CLEARCASE, IBM RATIONAL e SISTEMAS PRÓPRIOS.

Gráfico 1 - Forma da Gestão de Artefatos

01 - Como é realizada a gestão dos artefatos (documentos de texto, planilhas, código-fonte, mensagens e outros) que você produz durante o desenvolvimento do seu trabalho?

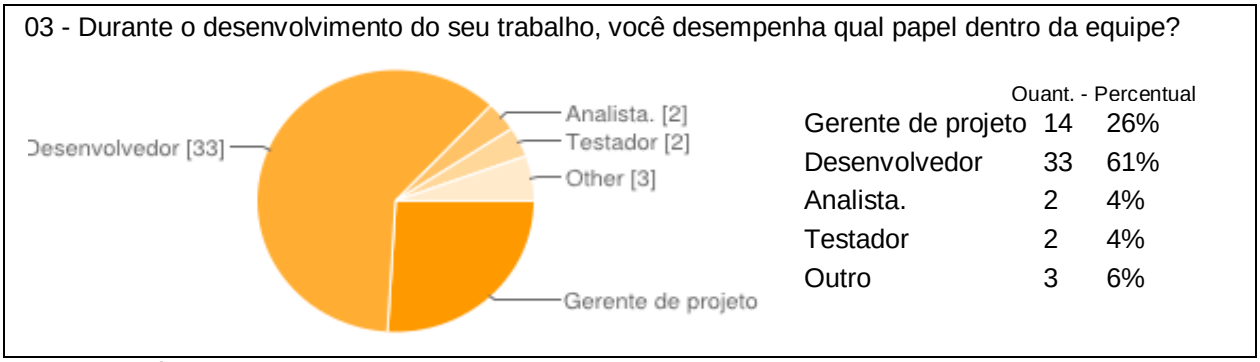


	Quant.	Percentual
Utilizo ferramentas específicas e independentes para cada tarefa.	42	78%
Não realizo a gestão de documentos com nenhuma ferramenta.	6	11%
Utilizo um software integrado para gestão destes artefatos.	6	11%

Fonte: Própria

Outra questão importante para a pesquisa aqui proposta era verificar o papel que a pessoa desempenhava durante seu trabalho, e se este papel poderia variar. A grande maioria dos pesquisados (61%) respondeu que atua como desenvolvedor (Gráfico 2).

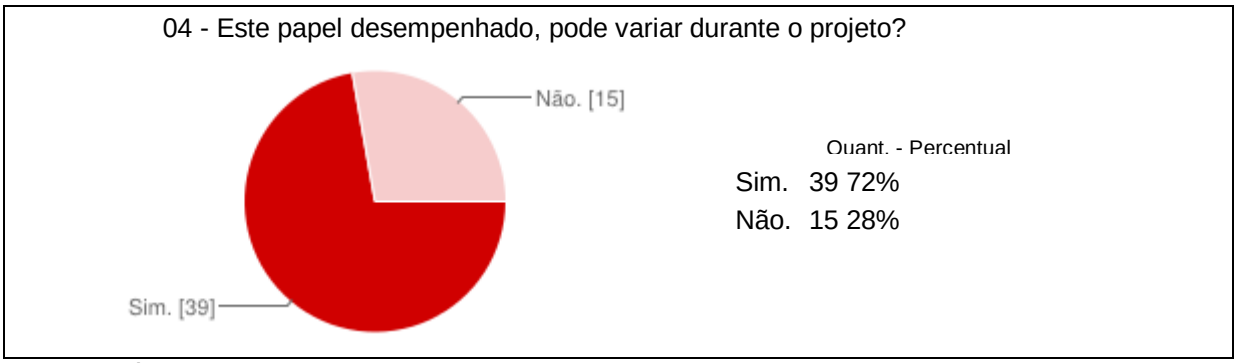
Gráfico 2 - Papel desempenhado na equipe.



Fonte: Própria

Quando questionados da variação do papel desempenhado durante o decorrer do projeto, 72% dos pesquisados respondeu que esta variação existe em grande número (Gráfico 3).

Gráfico 3 - Percentual de variação do papel desempenhado.

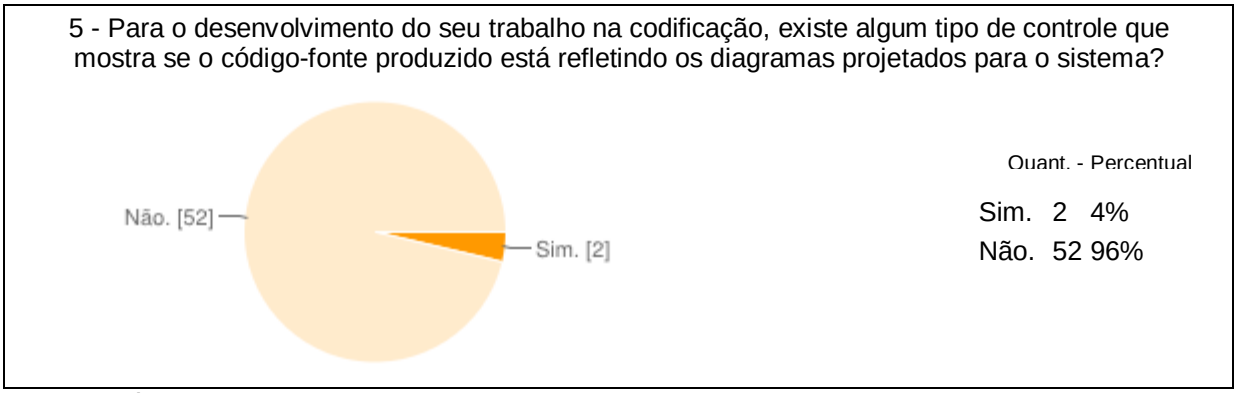


Fonte: Própria

Outra questão importante realizada na pesquisa buscava conhecer dos profissionais se estes já utilizavam ferramentas que acompanhassem o código-fonte

sendo produzido em confrontação com os diagramas projetados em fases anteriores para o sistema. O objetivo era verificar se outra ferramenta já realiza este tipo de trabalho, que é um dos itens importantes concebidos para o desenvolvimento do WS. A grande maioria, quase a totalidade das pessoas responderam negativamente. Apenas 4% dos questionados respondeu sim, que existia este tipo de controle do código com a modelagem, conforme pode ser visto no Gráfico 4. Também foi solicitado indicar o nome da ferramenta em uma questão posterior, mas nenhum nome foi indicado. Este fato leva a acreditar que talvez as repostas para SIM tenham sido equivocadas ou mal interpretadas pelas pessoas que responderam.

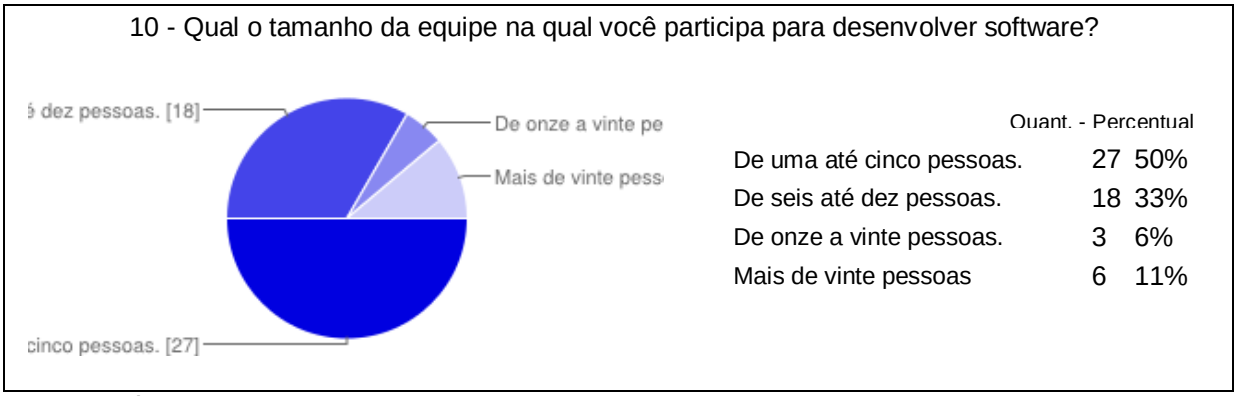
Gráfico 4 - Utilização de controle de código-fonte com os diagramas do sistema.



Fonte: Própria

Também foi questionado o número de componentes da equipe de trabalho que a pessoa participava no desenvolvimento de software. O Gráfico 5 apresenta os resultados, que mostram que as equipes com até dez pessoas ocorrem com maior frequência. Das 54 pessoas que respondem ao questionário, 9 participam de equipes com mais de onze pessoas.

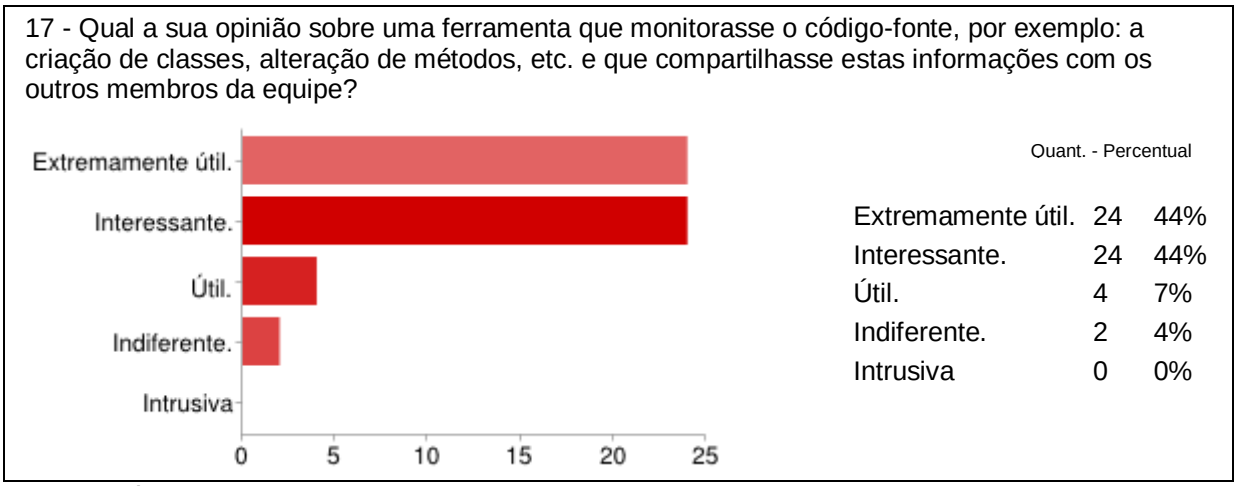
Gráfico 5 - Tamanho da equipe de desenvolvimento



Fonte: Própria

Com o objetivo de conhecer a opinião sobre uma ferramenta que monitorasse o código-fonte e que compartilhasse estas informações com a equipe de desenvolvimento, o Gráfico 6 apresenta os resultados, que indicam a boa aceitação de uma ferramenta com estas características.

Gráfico 6 - Opiniões sobre ferramenta de monitoração.

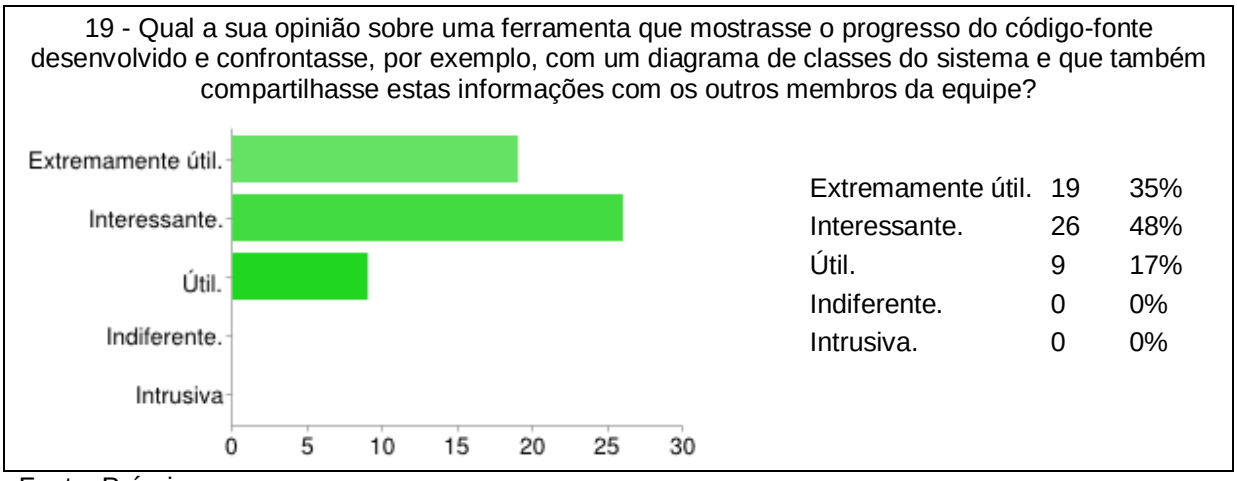


Fonte: Própria

A questão 19 perguntava sobre a utilidade de uma ferramenta que, sobre o código-fonte produzido, confrontasse com um diagrama de classes do sistema e compartilhasse esta informação com o grupo de pessoas que estaria trabalhando em conjunto. Os resultados mostrados no Gráfico 7 mostram que nenhum dos questionados escolheu responder que a ferramenta seria indiferente ou mesmo

intrusiva (duas últimas opções da questão), pelo contrário, 17% consideraram que ela seria útil, 48% acharam que seria interessante e 35% avaliaram este tipo de ferramenta como extremamente útil.

Gráfico 7 - Opiniões sobre ferramenta que confrontasse código-fonte com a modelagem.



Fonte: Própria

Foram apresentadas dezenove sugestões, que podem ser encontradas no Apêndice 2 deste documento. Estas sugestões são importantes, pois representam a opinião de profissionais que trabalham diretamente no desenvolvimento de software, em situações do mundo real, onde vários outros conceitos e ferramentas diferentes são utilizados e fazem parte do dia-a-dia das pessoas envolvidas em produzir software. Com estas questões apresentadas, na sequencia é mostrada uma visão do WS, suas funcionalidades e características.

4.2. O conceito do Workspace Semântico

A proposição do conceito do WS objetiva agregar características e funcionalidades consideradas importantes para apoiar a monitoração do código, a gestão dos artefatos e confrontação do código-fonte com a modelagem UML. Estas características são:

- Facilitar a visibilidade dos artefatos manipulados no desenvolvimento de software: As formas de facilitar sua efetivação foram levadas em consideração durante todo o ciclo de desenvolvimento e utilização do WS;
- Contribuir para que o *awareness* seja efetivo no WS: O *awareness* também é um conceito importante relacionado à pesquisa, e o WS contribuirá para que a falta de contexto ou desconhecimento do que ocorre no grupo de trabalho seja minimizada, justamente por oferecer aos participantes do projeto condições de acompanhar o desenvolvimento realizado pelos membros da equipe. Isto é possível em razão de ser mostrado o código-fonte aos participantes no momento em que ele está sendo produzido, desta forma alterações podem ser acompanhadas de forma *on-line* pelos programadores e outros profissionais envolvidos na atividade.
- Apoiar a verificação de software: Conforme definição da IEEE em (IEEE, 2012), a verificação de software tem o objetivo de avaliar se o planejamento realizado foi realmente executado, por exemplo: se os requisitos e funcionalidades do sistema modelado anteriormente foram efetivamente implementados. Com o WS existe a possibilidade de analisar se o programa sob desenvolvimento está seguindo o modelo previamente elaborado. Esta verificação ocorre em tempo real de codificação, permitindo aos participantes estarem cientes do que está sendo realizado pelos componentes da equipe no momento que o fato está ocorrendo. Um exemplo disto é a criação de uma Classe não existente no Diagrama de Classes do sistema sendo desenvolvido: neste caso os participantes do projeto podem ser avisados desta ocorrência.
- Integrar semântica os artefatos: Um exemplo desta ligação semântica é a integração dos programas sendo desenvolvidos (código-fonte), com a modelagem UML (diagrama de classes do sistema), que será realizada por meio do conteúdo destes dois artefatos, que podem ser interligados por meio do mapeamento que o HACKYSTAT (JOHNSON *et al.*, 2003) realiza

no código-fonte com a documentação fornecida pela ferramenta ASTAH COMMUNITY[15];

As funcionalidades do WS:

- Visibilidade dos documentos: o WS irá manter os usuários informados sobre o que está acontecendo, fornecendo um *feedback* aos participantes das equipes. Por exemplo, e-mails serão enviados aos envolvidos na execução do projeto em andamento, quando ocorrerem mudanças em artefatos do projeto. Outra forma de *feedback* imediato será a visualização das mudanças ocorridas no código fonte, quando alterações realizadas pelos programadores forem efetuadas, mostrando estas alterações para todos os participantes do projeto em tempo real de codificação;
- Envio de e-mails ao responsável do projeto nas modificações executadas pelos envolvidos na codificação do projeto em andamento. Este envio é configurado para ser realizado quando ocorrerem mudanças nos itens referentes à codificação do sistema que não estavam previstas na fase de análise e projeto do sistema.
- Gestão do projeto de desenvolvimento: Informações referentes ao desenvolvimento do projeto de software são apresentadas ao usuário responsável por esta atividade, tais como mensurar a criação por classes, métodos e atributos manipulados, tempo total gasto na codificação dos programas e outras informações gerenciais que auxiliam na tomada de decisão no modo como o projeto está sendo conduzido;
- Apresentação gráfica dos artefatos manipulados pelo WS: A apresentação gráfica do WS visa a melhorar a forma como os artefatos são monitorados e manipulados, facilitando a interpretação dos resultados obtidos. O KANBAN (IKONEN *et al.,* 2011), foi uma fonte de inspiração para a proposição deste tipo de composição gráfica facilitando a interpretação dos

[15] Ferramenta gratuita que auxilia nas atividades de engenharia de software, especificamente na modelagem de sistemas e está disponível em `http://astah.net/editions/community`.

resultados obtidos. Esta monitoração registra diversos dados, tais como: quais classes e quanto tempo um determinado programador está trabalhando nela, quais métodos foram criados, excluídos ou alterados, entre outros dados. A apresentação destas informações é realizada de forma visual e em tempo real de programação

- Criação de Ligações Semânticas: São coletados dados relacionados com colaboração entre os envolvidos no processo de desenvolvimento, principalmente por meio dos relacionamentos semânticos criados entre o projeto UML e a codificação JAVA dos programas sendo codificados, que foi denominada de Ligação Semântica. Alguns exemplos destes relacionamentos são mostrados no Quadro 13. O trabalho (SILVA *et al.,* 2013), também apresenta alguns destes relacionamentos semânticos criados em um projeto de software;
- Visualização de dados históricos mantidos pelo WS: Pode-se verificar, na forma de uma linha do tempo, os artefatos que foram criados/modificados/manipulados no WS permitindo o gerenciamento dos dados e verificação de resultados, principalmente dos dados oriundos da codificação realizada pela equipe de desenvolvimento.

4.3. Workspace Semântico

O WS é composto por três camadas, baseado nos conceitos do padrão MVC (*Model-View-Controller* – SENGUPTA *et al.,* 2007). Dessa forma, as camadas ficaram distintas (Figura 7) promovendo fácil manutenção e baixo acoplamento entre componentes, o que proporciona a reutilização dos mesmos como integração em outros possíveis módulos.

A primeira camada (Camada de Dados) contém os artefatos envolvidos no desenvolvimento do projeto. Um destes artefatos é o código-fonte que cada programador estiver desenvolvendo. Outros componentes desta camada são os diagramas de classe da modelagem UML e os artefatos produzidos pelo P.I.M., que será gerido pelo DS NEPOMUK, principalmente documentos de texto, planilhas de cálculo e e-mails.

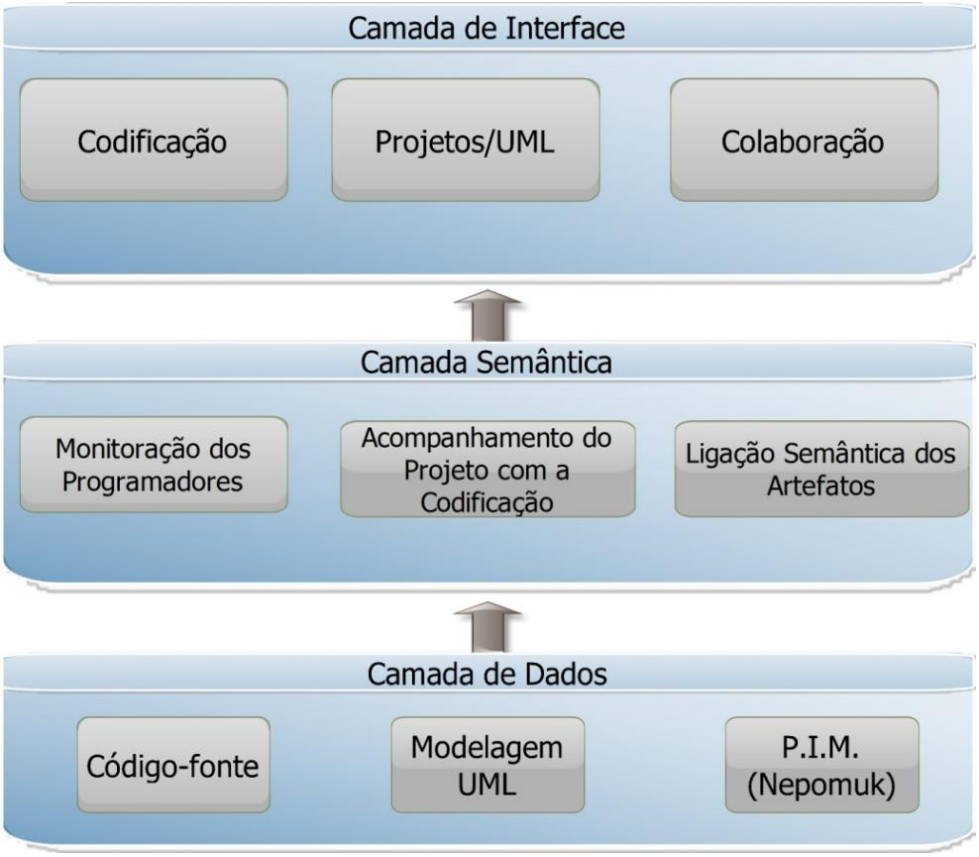


Figura 7 – Visão Geral do Workspace Semântico.

Fonte: (própria)

Na segunda camada (Camada Semântica), a monitoração do trabalho dos programadores é realizada com o auxílio do *framework* HACKYSTAT. Como ele é o principal componente responsável por esta tarefa, uma descrição de maiores detalhes é realizada na seção 3.6. O segundo componente da camada irá realizar o acompanhamento da codificação do projeto por meio dos diagramas da UML, onde o WS verifica o diagrama de classes cadastrado para o projeto e gera, com base nas informações coletadas, a arquitetura do diagrama de classes que é utilizado posteriormente no projeto de software sendo monitorado pelo WS, bem como a geração dos relatórios de acompanhamento. O terceiro componente desta camada é a ligação semântica dos documentos (artefatos). A parte semântica será realizada por meio do conteúdo destes artefatos (código-fonte e diagramas UML), sem a necessidade de utilizar outras fontes, como por exemplo, ontologias. Estes três componentes levam em consideração os dados e informações que foram gerados anteriormente na primeira camada.

A terceira camada (Camada de Interface) utiliza as técnicas da IHC (Interação Humano Computador) para gerar as interfaces que serão necessárias para auxiliar os gestores a acompanharem o desenvolvimento do trabalho.

4.4. O Funcionamento do Workspace Semântico

O funcionamento do WS foi inspirado na metodologia conhecida como KANBAN (IKONEN *et al.,* 2011), vinculado à metodologia ágil de desenvolvimento de software SCRUM[16]. O KANBAN é um conceito relacionado com a utilização de cartões (na informática, geralmente são utilizados papéis autocolantes coloridos) para indicar o andamento da produção de determinados itens. Nesses cartões são colocadas indicações sobre uma tarefa, por exemplo, “para executar”, “em andamento” ou “finalizado”. A Figura 8 apresenta esta técnica sendo utilizada de forma manual, permitindo um controle da produção com informações sobre quando, quanto e o que produzir.

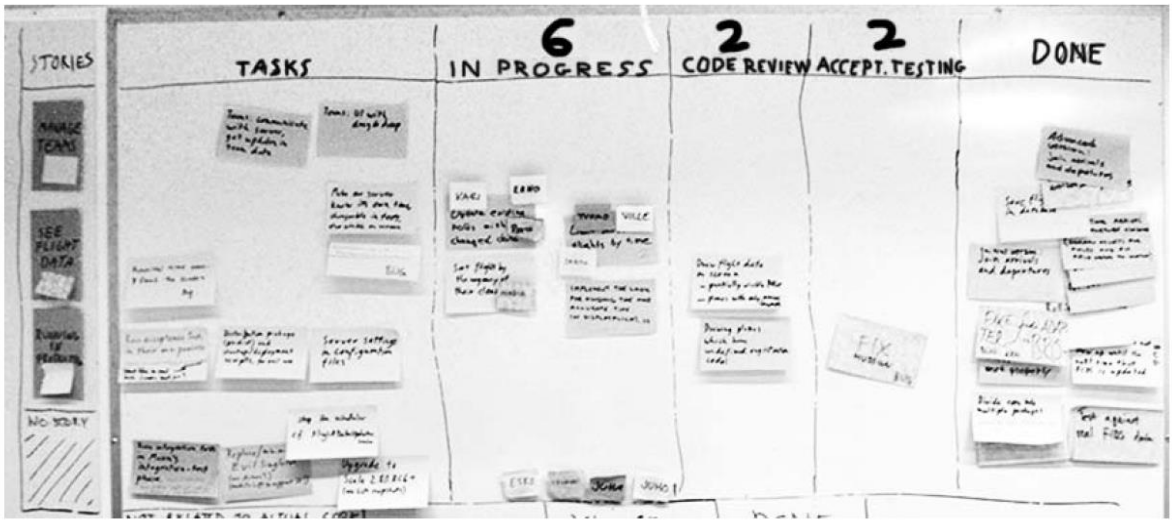


Figura 8 – Utilização de um quadro KANBAN em um projeto de desenvolvimento de software.

Fonte: (IKONEN *et al.,* 2011)

[16]O SCRUM é um *framework* de desenvolvimento iterativo e incremental utilizado no gerenciamento de projetos e desenvolvimento ágil de software. A palavra não é um acrônimo, mas é muito utilizada desta forma. O SCRUM contém grupos de práticas e papéis pré-definidos, entre eles o do SCRUMMASTER, que mantém os processos, o PROPRIETÁRIO DO PRODUTO, que representa os *stakeholders* e o negócio e a EQUIPE, um grupo multifuncional que realizam a análise, projeto, implementação e testes do sistema.

Existem versões eletrônicas que permitem um controle mais detalhado e com possibilidades de integrar o acompanhamento contínuo do *workflow* das tarefas em cada fase do KANBAN. Esta facilidade dispensa a utilização de um único local para o quadro do KANBAN, que na versão manual é único e em um local pré-determinado. Várias empresas oferecem softwares que utilizam o conceito do KANBAN, variando desde versões livres ou de demonstração e até versões comerciais. A Figura 9 mostra um exemplo de software que realiza o controle eletrônico das tarefas por meio do KANBAN.

Este conceito foi adotado para o acompanhamento visual do desenvolvimento e dos diagramas de classe da UML. As cores auxiliam a determinar em que fase da codificação cada parte do projeto está inserido.

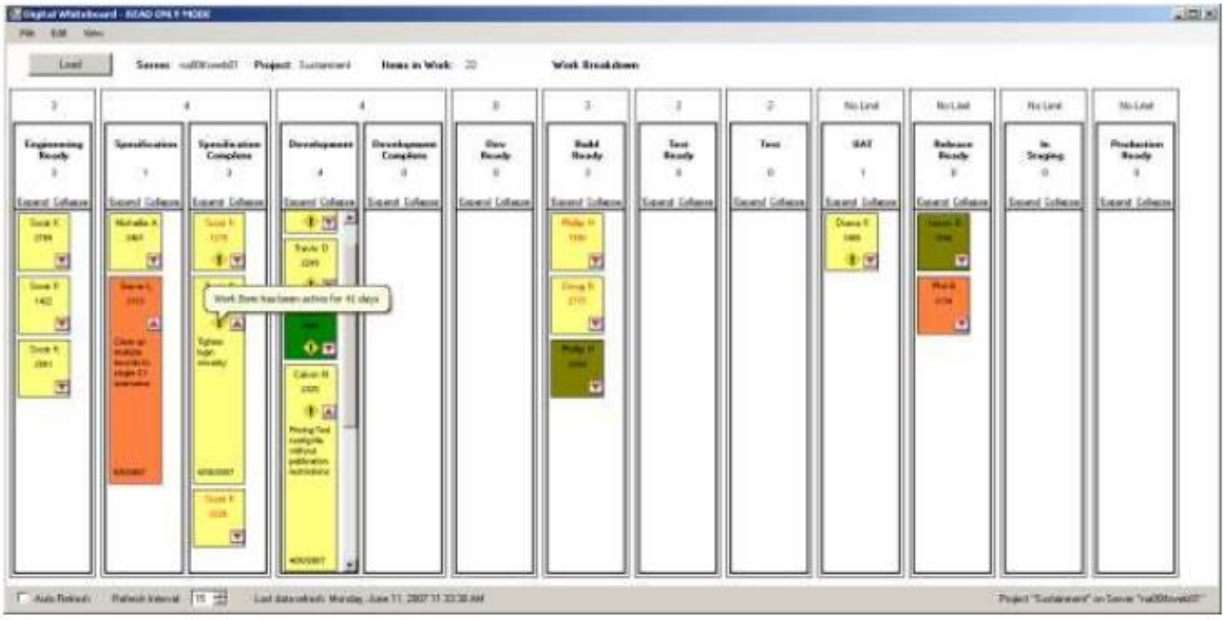


Figura 9 – Utilização do KANBAN em um software aplicativo.

Fonte: extraído de (ANDERSON, 2010, pg. 73).

4.5. Acompanhamento da Codificação com o Diagrama de Classes

A partir da coleta dos dados provenientes do código-fonte, um diagrama de classes é criado e atualizado a cada interação dos usuários (desenvolvedores) com o sistema.

Os dados colhidos são utilizados também para a emissão de relatórios. Estes relatórios são de dois tipos: visuais e em tempo de execução e também relatórios gerenciais, com dados estatísticos sobre a codificação realizada e que mostram os dados históricos do projeto. Nestes relatórios, são apresenta as informações tais como a criação, modificação e exclusão de classes, métodos e atributos de um projeto em particular. Com estes podem-se gerar estatísticas do projeto, tais como: o tempo gasto em cada atividade, número de compilações de um programa, erros de execução e uma gama de outras medidas. O Quadro 12 mostra a lista dos eventos e uma descrição de seu funcionamento dos relatórios gerenciais que pode ser obtida pelo OPERAM (ferramenta que faz parte do WS e é descrita em maiores detalhesno item 4.7.1 deste documento .

Quadro 12 – Lista de eventos.

EVENTO	DESCRIÇÃO
Classes Criadas	Disparado sempre que o desenvolvedor cria uma classe Java em seu código fonte;
Classes Renomeadas	Gerado quando o desenvolvedor modifica o nome de uma classe;
Classes Excluídas	Ocorre quando uma classe é excluída;
Atributos Criados	Quando um atributo é adicionado a uma classe;
Atributos Renomeados	Atualizado quando o desenvolvedor modifica o nome de um atributo de uma classe;
Atributos Excluídos	Ocorre quanto um atributo é excluído de uma classe;
Métodos Criados	Disparado quando um novo método é criado em uma classe;
Métodos Renomeados	Gerado quando o desenvolvedor modifica o nome de um método;
Métodos Excluídos	Ocorre quando um método é excluído de uma classe;
Compilações Executadas	Ocorre quando uma compilação é executada por um desenvolvedor;
Erros de Execução	Gerado quando um erro de execução ocorre;
BREAK POINTS inseridos	Contabiliza a quantidade de *BREAK POINTS* incorporados no código-fonte;
IMPORTS realizados	Ocorre quando um *IMPORT* é feito para agregar itens ao projeto em desenvolvimento;
Tempo gasto em itens do projeto	Contabilizado sempre que o desenvolvedor inicia o trabalho em atividades nos itens do projeto, como classes, atributos ou métodos e finalizado quando cessa a interação com o item.

A Figura 10 mostra o metamodelo do diagrama de classes que permite realizar o acompanhamento da codificação e na qual os eventos do Quadro 12 estão relacionados. A função de um diagrama de classes da UML é proporcionar uma visão das classes que irão compor determinado sistema, com seus atributos, métodos e também do relacionamento entre as classes e como as mesmas se complementam e se relacionam entre si. Como este diagrama de classes é específico para modelar o processo de desenvolver software, é natural que siga procedimentos na sua modelagem. Por este motivo, são encontradas componentes

que dizem respeito ao processo de desenvolver um software que utiliza estes conceitos, tais como classes, atributos métodos, parâmetros, além de outros como, por exemplo, usuários e suas atividades.

Nesta figura observa-se a visualização das classes para gerenciar os projetos, com seus respectivos atributos e métodos, e também a forma como as classes se relacionam e transmitem informações, colaborando entre si para a execução de processos específicos. São elas:

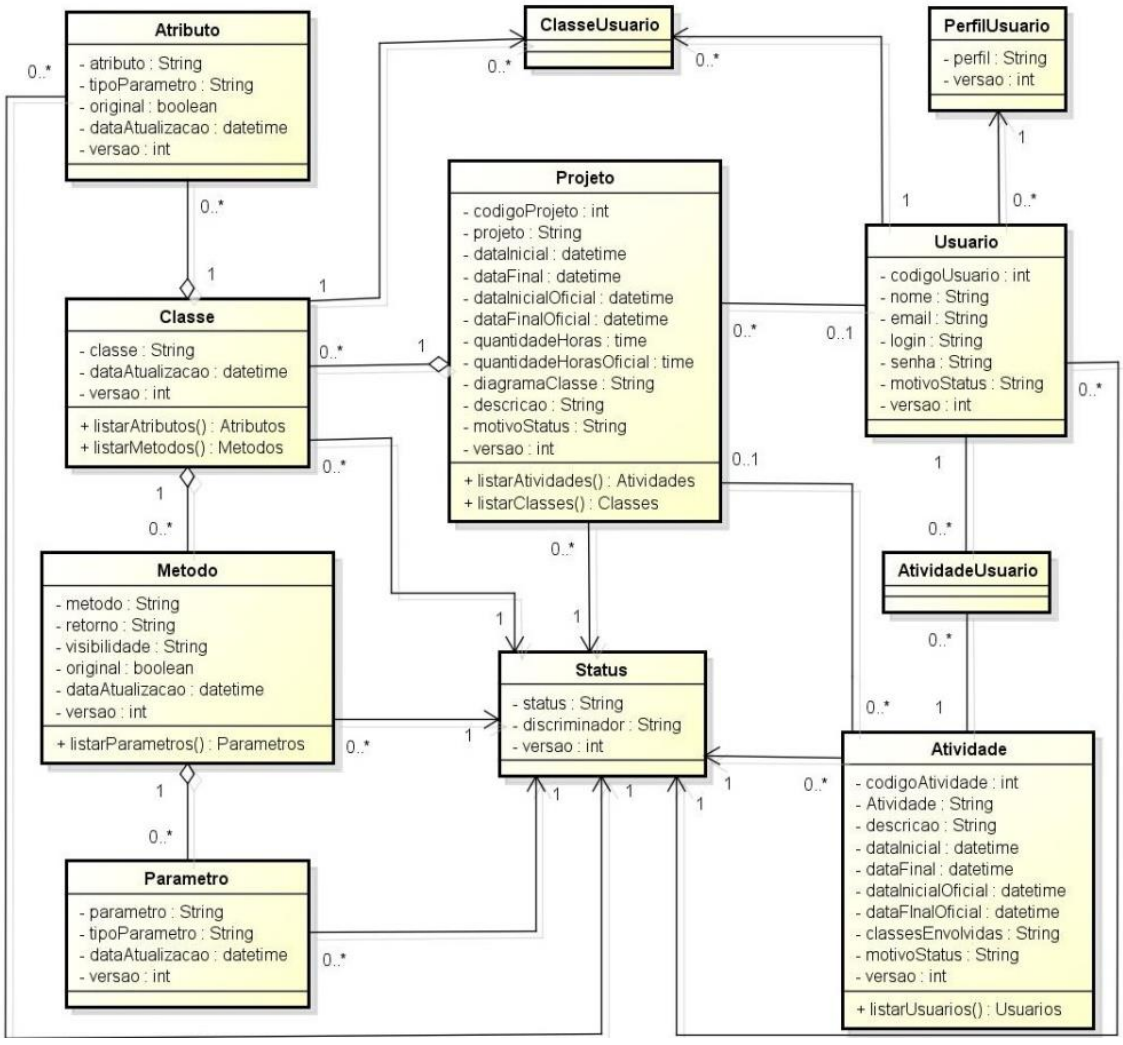


Figura 10 – Metamodelo do Diagrama de Classes para Acompanhamento da Codificação no OPERAM.
Fonte: (Própria)

- Classe PROJETO: O objetivo desta classe é possibilitar o cadastro dos sistemas que serão monitorados pelo módulo. O controle do período de tempo do projeto (estimado e realizado), a quantidade de horas despendidas (horas estimadas e efetivamente realizadas para a execução do projeto), e a ligação com diagrama do ASTAH COMMUNITY são alguns dos componentes desta classe. Esta classe é ligada diretamente a classe STATUS, cuja função é permitir verificar a condições em determinado momento do andamento do projeto;
- Classe STATUS: É uma classe que contém informações da situação (e.g., em andamento, parado, em desenvolvimento) das seguintes classes: PROJETO, CLASSE, METODO, ATRIBUTO, PARAMETRO e ATIVIDADE. Sua função e permitir que seja verificado em que condição do desenvolvimento cada uma das classes vinculadas se encontra;
- Classe CLASSE: Permite o cadastro das classes do sistema inseridas no OPERAM;
- Classe ATRIBUTO: são os atributos das classes a serem controladas;
- Classe METODO: refere-se aos métodos de uma classe. Contém atributos do tipo nome do método, retorno e visibilidade. Nesta classe destacam-se os atributos “original” e “dataAtualizacao” que representam se um método é original, ou não, e a data da última atualização, respectivamente. Os parâmetros de um método são representados pela classe PARAMETRO;
- Classe USUARIO: Esta classe agrega os usuários que podem participar do projeto, e conta com o alguns atributos identificadores, tais como: nome, e-mail, login e senha.
- Classe PERFILUSUARIO: O usuário estará vinculado a um determinado perfil (gerente, programador, etc.);
- Classe ATIVIDADE descreve as atividades que os usuários podem desenvolver dentro do módulo, é composta por atributos tais como: atividade, descrição da atividade, data inicial e final da atividade, classes envolvidas dentro outros atributos.

Desta forma e com esta estruturação, o OPERAM pode contribuir também para a verificação de software, com o principal objetivo de subsidiar a avaliação do

planejamento realizado e se este está sendo executado, bem como se os requisitos e funcionalidades também estão sendo efetivamente implementados.

4.6. Ligações Semânticas no OPERAM

As ligações semânticas criadas no OPERAM são a principal forma de realizar a integração entre os desenvolvedores e os artefatos oriundos da programação. A função de uma ligação semântica é proporcionar meios de obter informações de contexto entre estes elementos. Define-se uma ligação semântica como sendo a ligação criada entre o conteúdo dos artefatos manipulados durante a codificação de programas e as pessoas envolvidas neste processo.

O OPERAM identifica ligações semânticas entre artefatos e seus componentes, com os participantes do projeto, recolhendo dados relativos à interação entre eles. Este relacionamento fornece informações de contexto sobre autor de uma determinada parte do código-fonte e um item específico da modelagem. O Quadro 13 mostra ligações semânticas que podem ser criadas durante o desenvolvimento de um projeto de software.

Quadro 13 – Ligações Semânticas do OPERAM.

LIGAÇÕES SEMÂNTICAS	SIGNIFICADO
é-responsável-por	Identificação do usuário responsável pelo item;
item-eliminado-por	Identificação do usuário responsável pela eliminação do item;
última-atualização-por	Identificação do usuário responsável pela última atualização do item;
foi-alteração-por	Identificação do usuário que realizou alteração de um item;
é-relacionado-a	Ligação entre itens distintos do projeto;
item-criado-por	Identificação do usuário que criou o item;
é-coautor-de	Ligação entre um usuário e um item, onde o item foi criado por um terceiro;
última-modificação-por	Data e responsável pela última modificação realizada em um item.

Estas ligações semânticas criadas pelo OPERAM podem ser visualizadas em tempo real. Uma ligação semântica ocorre, por exemplo, quando existe um relacionamento entre o autor de uma determinada parte do código-fonte e um item específico da modelagem. O Quadro 14 mostra a notação BNF (BLAUTH, 2005) como meio de representação para descrever a forma como as ligações semânticas são criadas.

Quadro 14 – Notação BNF das Ligações Semânticas.

```
<operador> ::= ,

<diagrama-classes> ::=  <item-classe>

<item-classe> ::= (<classe> | <atributo> | <metodo> )

<programa> ::= (<item-programa>

<item-programa> ::= (<programa-classe> | <programa-atributo> | <programa-metodo>)

<operacao> ::= (<criar> | <alterar> | <eliminar> | <data>)

<identificacao> ::= <desenvolvedor>

<ligacao> ::= (<é-responsável-por> | <item-eliminado-por> | <ultima-atualização-
          por> | foi-alterado-por | <é-relacionado-a> | <item-criado-por> | <é-
          coautor-de> | <ultima-modificacao-por>)

<expressao> ::= (<ligacao> | <identificação> | <operador> | <operação> | <item-
classe> | <item-programa>)

<é-responsável-por> ::= (<desenvolvedor> , <criar> , <item-classe>)

<item-eliminado-por> ::= (<desenvolvedor> , <eliminar> , <item-classe>)

<ultima-atualização-por ::= (<desenvolvedor , <item-classe> , <data>)

<foi-alterado-por> ::= (<desenvolvedor> , <item-classe> , <data>)

<é-relacionado-a> ::= (<desenvolvedor> , <item-programa>   )

<item-criado-por> ::= (<desenvolvedor> , <item-classe> , <item-programa>)

<é-coautor-de> ::= (<desenvolvedor> , <item-classe> , <item-programa> ,
            <desenvolvedor>)

<ultima-modificacao-por> ::= (<desenvolvedor> , <data> , <item-programa> , <item-
            classe>)
```

Como exemplo de expressão neste formalismo, pode-se citar a ligação semântica `<item-criado-por>`, que derivada na notação seria representada por `classe-criada-por = x.java` e `desenvolvedor1`.

Como referência, a Figura 11, apresenta a visão do sistema para acompanhamento da codificação, por meio do diagrama de casos de uso.

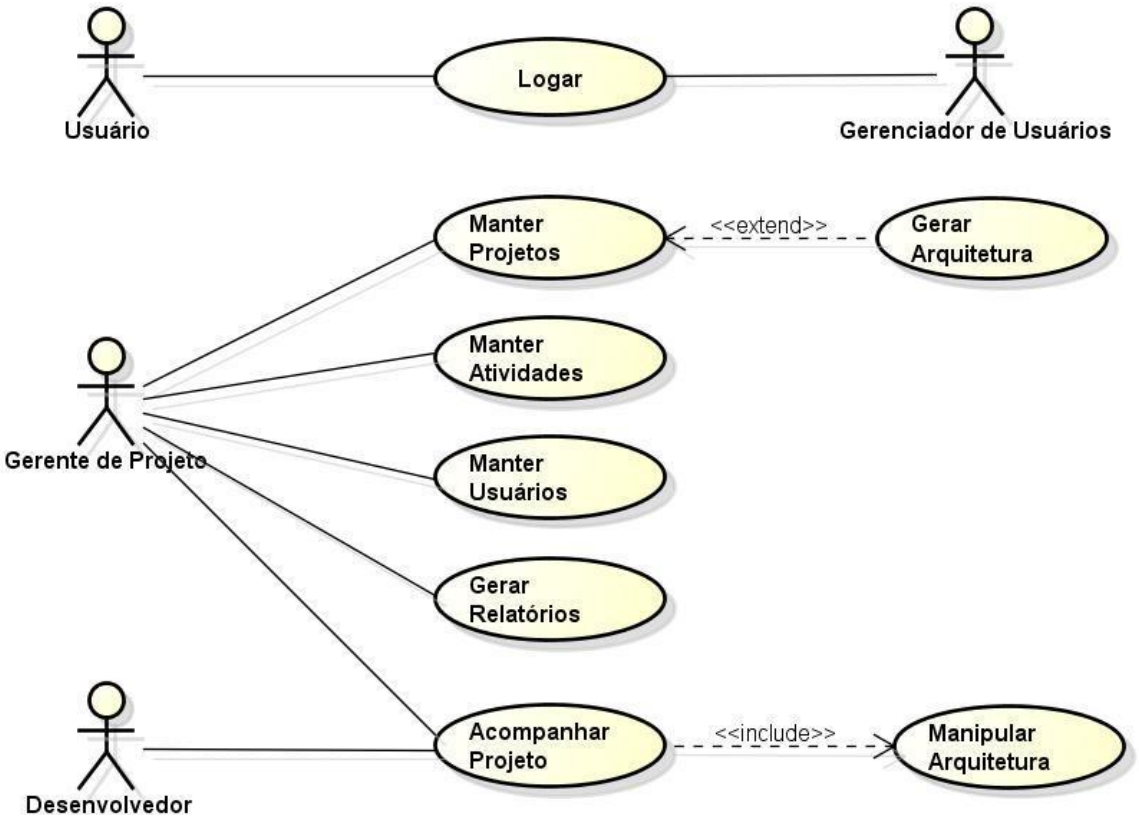


Figura 11 – Diagrama de casos de uso geral do OPERAM.

Fonte: (Própria)

Neste diagrama, é mostrado o cenário que ilustra as principais funcionalidades do OPERAM, do ponto de vista da sua utilização:

- USUÁRIO: Ao acessar o OPERAM, será exibida a tela de *login*, onde deverão ser informados o usuário e a senha. A consistência destes dados será realizada pelo GERENCIADOR DE USUÁRIOS, e caso seja permitido o acesso, será redirecionada para a tela inicial da aplicação, onde o usuário poderá escolher as possíveis opções do Menu. Em caso de não haver consistência, uma mensagem de erro é exibida na própria tela de *login*, habilitando novamente os campos de usuário e senha para uma entrada válida.
- GERENTE DE PROJETOS: Usuário que lidera um projeto, sendo responsável pela inserção das informações básicas e gerenciamento das atividades do mesmo. Geralmente é o líder técnico mais próximo da equipe de desenvolvimento e análise. Acessando o OPERAM com o perfil de

Gerente de Projetos o usuário poderá incluir, excluir, editar, pesquisar ou consultar projetos, podendo também vincular atividades ao mesmo. Ao final de qualquer um destes itens, o caso de uso GERAR ARQUITETURA será executado, criando a arquitetura principal do diagrama de classes do projeto manipulado. Este tipo de usuário também poderá realizar as operações citadas anteriormente para outros dois casos de uso: MANTER ATIVIDADES e MANTER USUÁRIOS. Ainda disto, o usuário pode realizar o acompanhamento dos projetos cadastrados no WS.

- PROGRAMADOR: Usuário operacional que executa a codificação do projeto na IDE de trabalho, gerando os dados necessários para o acompanhamento dos projetos cadastrados no módulo.
- MANIPULAR ARQUITETURA: Tanto os usuários com o perfil de Gerente de Projetos como de Desenvolvedor realizam alterações nos projetos cadastrados por meio deste caso de uso. No nível operacional, o módulo busca as informações do projeto, tais como atividades, usuários relacionados e arquitetura principal das classes, busca as informações de codificação coletadas pelo HACKYSTAT, faz a inferência dessas informações e atualiza a arquitetura principal das classes.

4.7. Arquitetura do Workspace Semântico

O WS é formado por dois componentes principais: o Desktop Semântico Social NEPOMUK, descrito no Capítulo 3 e o OPERAM. Nas próximas seções, apresenta-se um detalhamento do OPERAM.

4.7.1. A FERRAMENTA DE MONITORAÇÃO OPERAM

O NEPOMUK, descrito anteriormente, tem algumas limitações para ser utilizado em domínios específicos, como por exemplo, no desenvolvimento de software. A principal delas está no fato do NEPOMUK, bem como nenhum DS, ser

capaz de analisar o desenvolvimento da codificação, criando os relacionamentos semânticos apropriados. Conforme visto em anteriormente, estes relacionamentos são feitos pelo NEPOMUK quando envolvem e-mails ou imagens por exemplo. Além disto, existem outras limitações apresentadas a seguir:

- A interface com o usuário: A interface com o usuário do NEPOMUK é centrada na perspectiva do gerenciamento do P.I.M. do usuário. No processo de desenvolvimento de software, a perspectiva do grupo que desenvolve software precisa ser levada em consideração;
- A segurança no acesso a informação: Os componentes de uma equipe devem ter acesso a partes diferenciadas, e também em diferentes níveis, dos artefatos que estão sendo produzidos, fato que não ocorre no NEPOMUK;
- Endereçamento de partes dos documentos: O NEPOMUK utiliza uma forma de endereçamento dos documentos em que eles são considerados como um todo, e não é possível organizá-los em outras unidades (exemplo: um documento composto de seções, parágrafos, tabelas e figuras). O trabalho de (SCHANDL, 2009) já havia percebido esta limitação.

Para solucionar algumas destas limitações, um módulo de acompanhamento do desenvolvimento de software foi criado e recebeu o nome de OPERAM, que em latim tem o significado de “trabalhar em conjunto”. A Figura 12 mostra como os participantes relacionam-se no tocante ao início do processo de acompanhamento da codificação. Inicialmente, um novo projeto deve ser adicionado. O gerente de projeto é o responsável pelo registro dos modelos UML e também dos participantes (programadores) envolvidos com o projeto.

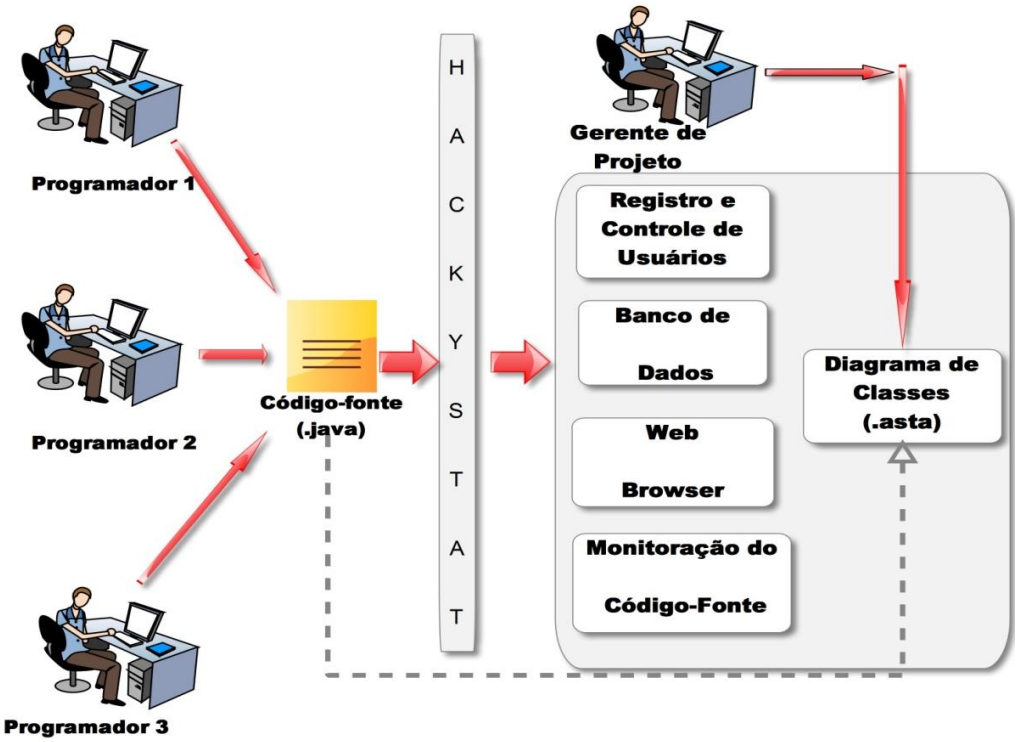


Figura 12 – Acompanhamento da Codificação no WS por meio do módulo OPERAM.

Fonte: (Própria)

Os programadores, por sua vez, recebem cópia do diagrama de classes inserido no OPERAM e podem iniciar a codificação do projeto. Suas estações de trabalho devem estar preparadas com a IDE ECLIPSE e corretamente configuradas com o HACKYSTAT, que é detalhado na seção 4.5 deste documento. Com estes passos instituídos, é iniciado o acompanhamento do desenvolvimento do projeto. A ideia é que os desenvolvedores iniciem a codificação e, em tempo real, tenham o código em desenvolvimento confrontado com o modelo (Diagrama do ASTAH COMMUNITY) previamente inserido pelo gerente do projeto. Nesta fase, o OPERAM utiliza-se do sensor que coleta dados de cada código fonte em desenvolvimento pelos programadores. Este sensor foi desenvolvido, e de forma não intrusiva, coleta e envia dados brutos sobre todo o processo de desenvolvimento para um *web service* para armazenamento. Esta coleta registra diversos dados, tais como: qual classe e quanto tempo um determinado programador está trabalhando nela, quais métodos foram criados ou alterados, entre outros.

O repositório que armazena estes dados pode ser consultado por outros serviços da web para formar abstrações de mais alto nível dos dados brutos armazenados e/ou integrá-lo com outro tipo de comunicação, via internet ou mecanismos de coordenação, para gerar visualizações ou anotações. Este sensor foi desenvolvido para coletar informações de usuários da IDE ECLIPSE e vários dados são produzidos e armazenados durante a interação entre o usuário e o OPERAM. Estes dados estão relacionados com um determinado evento e o Quadro 15 apresenta um trecho da lista de eventos gerados durante a construção do código-fonte.

Quadro 15 – Trecho da lista de eventos.

Nome do projeto em construção
Nome do arquivo em construção (arquivo .java)
Criação de um método
Supressão de um método
Alteração de um método
Criação de uma classe
Erros de compilação e sucedidos
Erros de execução e sucedidos
Quando e quem fez a modificação em um arquivo

Monitorar estes eventos fornece várias possibilidades de melhorar o gerenciamento de projetos de desenvolvimento de software. O gerente de projeto pode acompanhar a evolução do mesmo, avaliando diferentes medidas, tais como: o tempo de cada programador gasto trabalhando em cada classe, o número de itens codificados pelos desenvolvedores, inferir qual desenvolvedor é mais produtivo, entre outros exemplos.

Estes eventos e a monitoração são realizados em projetos cadastrados no OPERAM, que possuam desenvolvedores e um diagrama de classes corretamente inserido. A partir deste ponto, os projetos podem ser modificados, alterados e excluídos. Para cada interação que for realizada, o algoritmo lerá o diagrama de classes e buscará nele as classes do diagrama, bem como os atributos da classe e métodos, unificando as informações dentro do objeto Classe, que será persistido no banco de dados. Com isto, os objetos do tipo Atributo e Método serão persistidos também. No caso de alteração de um projeto, como por exemplo, serem adicionados atributos as classes do diagrama, o algoritmo fará o mesmo processo de cadastrar um projeto, porém executará um processo de inferência onde será verificado o

projeto de classes, atributos e métodos da situação anterior. Com isto, o projeto, será atualizado para a nova situação. No caso de exclusão de um projeto, todas as classes, atributos, métodos e parâmetros serão excluídos, junto com o próprio projeto.

O pseudocódigo do Quadro 16, baseado na linguagem Java, demonstra a lógica dos processos de cadastrar, alterar e excluir itens estes processos no OPERAM.

Quadro 16 – Pseudocódigo das Interações no OPERAM.

```
1  gerarArquitetura(Projeto)
2      DiagramaDeClasses = obterDiagramaDeClasses
3          if(acao.igual("cadastrar"))
4              for (diagrama.obterClasse)
5                      Classe novaClasse = nova Classe
6                      for (diagrama.obterAtributo)
7                          novo = novoAtributo
8                       for (diagrama.obterMetodo)
9                           novo = novoMetodo
10                     persistir(novaClasse)
11
12 else if(acao.igual("alterar"))
13     listaDeClasses = buscarClassesDoProjeto;
14     for (diagrama.obterClasses)
15         for (diagrama.obterAtributos)
16             (novo = novoAtributo)
17         for (diagrama.obterMetodos)
18              novo = novo Metodo)
19         for (diagrama.obterClasses)
20             (novo = novaClasse)
21     persistir(ClasseAlterada)
22
23     else if(acao.igual("excluir"))
24         listaDeClasses = buscarClassesDoProjeto();
25             for (diagrama.obterAtributos)
26                 delete(atributo);
27             for (diagrama.obterMetodos)
28                 delete(metodo);
29             for (diagrama.obterClasse)
30                 delete(classe);
31     endif
32 endif
```

A Figura 13 mostra um exemplo de utilização do OPERAM no WS para acompanhar de forma visual um projeto. Cada cor refere-se a um status diferente, a saber: Vermelho – remoção de um item; Amarelo – depuração, por meio da inserção ou remoção de um *break point* dentro do código; Verde – criação de um item e Azul – renomear um item do projeto. Define-se como “item” um componente do projeto, como por exemplo: uma classe, um método ou um atributo. Neste exemplo, composto de cinco classes, a cor verde identifica que as classes foram criadas no

sistema (Itens LOCADORA, CLIENTE, CARRO e CLIENTEESPECIAL), e a cor vermelha que a classe foi removida (Classe ALUGUEL).

Esta forma de acompanhamento visual por meio dos diagramas da UML tem como objetivo instigar a colaboração da equipe, mostrando em tempo real o que está sendo desenvolvido (*awareness*), ampliando o grau de conhecimento do projeto sendo trabalhado pela equipe de desenvolvimento de software.

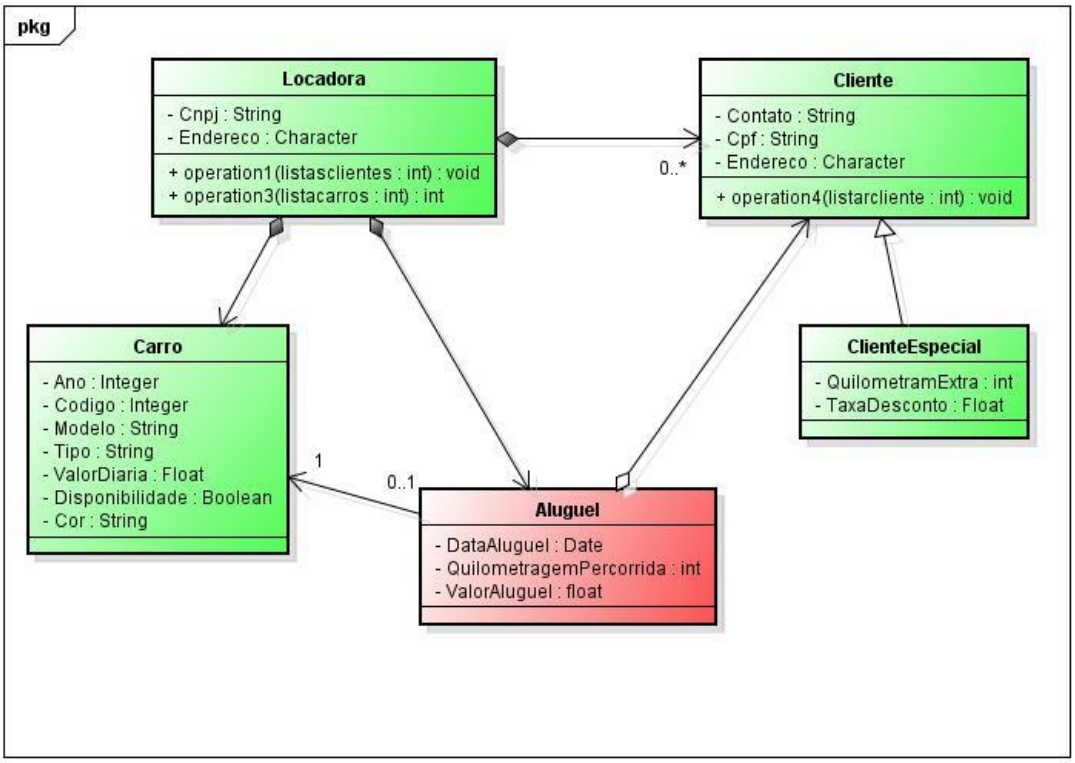


Figura 13 – Exemplo de utilização da técnica KANBAN para acompanhar projetos no OPERAM.

Fonte: (Própria)

A Figura 14 mostra as ligações semânticas que são apresentadas de forma visual e em tempo real, mostrando nesta visualização as Classes que compõem um determinado projeto e que estão sendo desenvolvidas e manipuladas pelos programadores. As cores do gráfico auxiliam a visualizar quais Classes estão sendo modificadas pelos usuários. A legenda, que é mostrada na parte inferior da figura, auxilia a identificar estas classes no gráfico. Cada classe possui uma cor distinta, e cada interação que um usuário realiza é atualizada neste gráfico em tempo real, o que produz alterações que são percebidas também pelos outros usuários que estiverem utilizando o OPERAM. Usuários são os desenvolvedores cadastrados no OPERAM e que realizam atividades de programação no projeto sendo monitorado.

Estes usuários são identificados por seu e-mail, e neste protótipo do OPERAM, são mostradas as interações que os usuários realizam nas Classes, Atributos e Métodos do diagrama de classes.

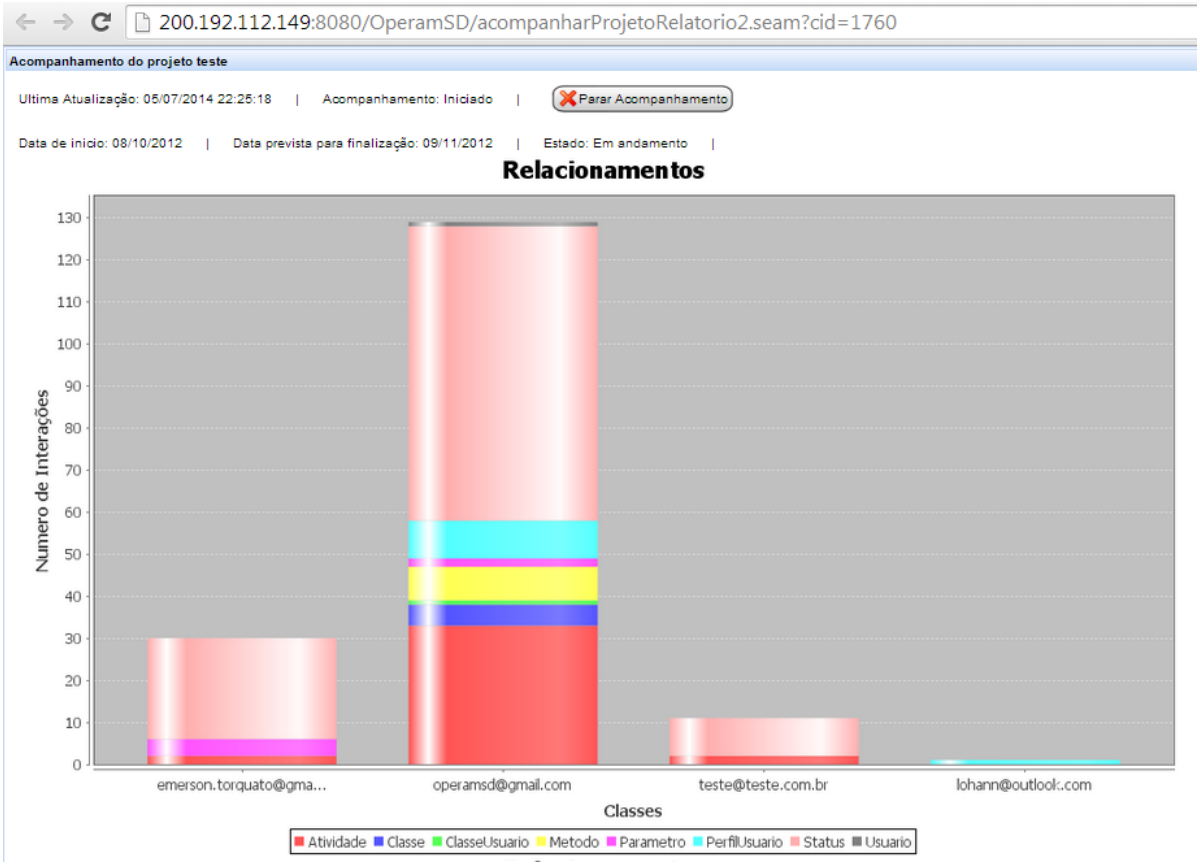


Figura 14 – Exemplo de acompanhamento das ligações semânticas on-line no OPERAM.

Fonte: (Própria)

Nota-se neste gráfico em particular a existência de quatro desenvolvedores atuando na programação do sistema, que é composto de oito classes. Destes desenvolvedores, o segundo desenvolvedor tem maior atividade que os demais (*operamsd@gmail.com*). O último desenvolvedor (*lohann@outlook.com*) foi o que menos interagiu, manipulando apenas uma classe (PerfilUsuario). Também nota-se que a classe que possui mais interações por parte dos desenvolvedores é a classe Status, onde três usuários interagiram, produzindo modificações na sua estrutura e composição. Ainda sobre esta classe, percebe-se no gráfico mostrado que o usuário que mais produziu interações foi *operamsd@gmail.com*, e que a classe em que o mesmo interagiu mais foi a classe Status. Esta mesma classe

também teve interações de outros dois usuários, e por meio da cor que identifica esta classe, percebe-se as ligações destes três usuários com esta classe específica. Nesta forma de visualização, não é possível mostrar o tipo de ligação semântica entre os artefatos e os desenvolvedores, pois o objetivo é mostrar os relacionamentos e as interações entre os usuários de forma gráfica e sintetizada, sem pormenorizar os detalhes e assim sintetizar o máximo de informações a serem apresentadas.

Outra forma de visualizar o projeto sendo desenvolvido de forma gráfica, é por meio do acompanhamento do trabalho sendo realizado por um determinado desenvolvedor, como mostra a Figura 15, onde os itens manipulados estão ligados a este usuário. Pode-se observar também as ligações semânticas deste desenvolvedor e assim obter informações que auxiliam na atividade de verificação de software deste participante em particular.

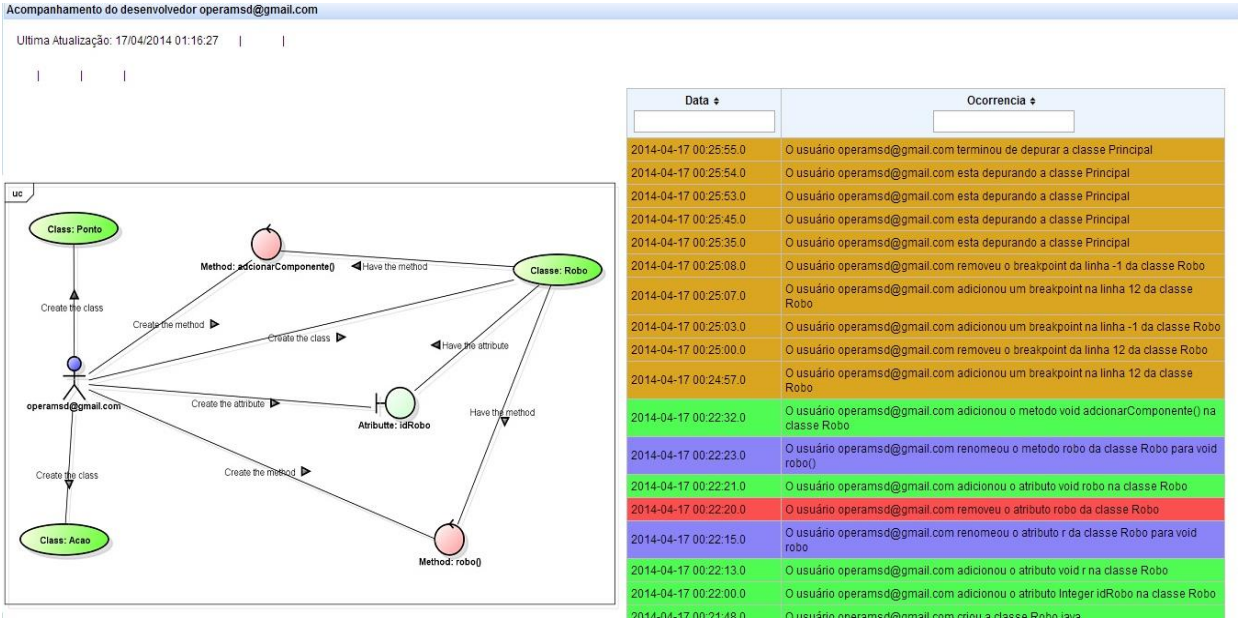


Figura 15 – Ligações Semânticas de um usuário no OPERAM.

Fonte: (Própria)

No exemplo mostrado nesta figura, o desenvolvedor, o código-fonte e o item da modelagem que ele está manipulando possuem uma ligação semântica que envolve os rótulos de conceito associado a ele, que especifica o tipo de relacionamento entre os objetos para a criação de dois métodos (adicionarComponente() e robo()), três classes (Robo, Acao e Ponto) e um atributo

(IdRobo). A classe Robo possui os métodos e atributos citados e que foram criados pelo usuário *operamsd@gmail.com*. Estes relacionamentos são úteis para fornecer aos participantes do projeto (usuários) além das informações inerentes à codificação, outros dados adicionais, como por exemplo, itens manipulados por mais de um desenvolvedor, quem realizou a criação do item e quem alterou sua composição. O lado direito da figura, mostra os dados históricos deste desenvolvedor em particular para cada ação que ele desenvolve durante a programação.

Além deste acompanhamento mostrado na Figura 15, outra forma de acompanhamento disponível no OPERAM para análise do desenvolvimento por meio das ligações semânticas pode ser vista na Figura 16, que mostra um único usuário, com atualização em tempo real dos relacionamentos deste usuário com as classes do sistema. O exemplo desta figura mostra apenas um desenvolvedor com os itens manipulados por ele e que estão ligados ao mesmo. O número mostrado nesta ligação indica a quantidade de interações que este usuário realizou naquele item. Uma interação envolve qualquer forma de acesso ao item, seja criação, troca do nome, eliminação ou toda e qualquer conexão que acontecer com o item.

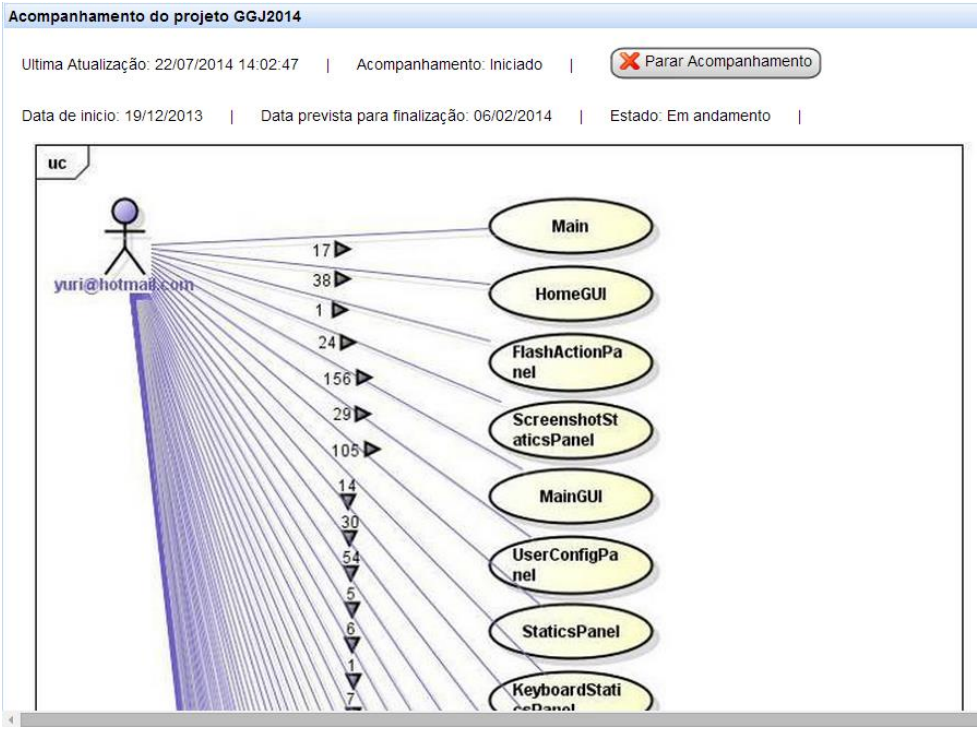


Figura 16 – Ligações Semânticas Entre um Usuário e Classes do Diagrama de Classes.

Esta forma de visualização por meio de gráficos, mostrada nas Figuras 15, 16 e 17, apresenta dados que são manipulados advindos da interação dos desenvolvedores em suas *IDEs* de trabalho, e que são monitoradas pelo HACKYSTAT, e possuem ligações com o diagrama de classes que foi anteriormente produzido no ASTAH COMMUNITY. Estas informações necessitam ser armazenadas em um banco de dados (no caso do OPERAM é o MySQL). Este gráfico mostra as ligações semânticas por meio do relacionamento entre os desenvolvedores e as classes que estes manipulam, em tempo real de codificação.

4.8. Projetos sendo acompanhados no OPERAM

O endereço web `http://200.192.112.149:8080/OperamSD/`, mostrado na Figura 17, permite aos participantes de projetos de desenvolvimento acompanhar a codificação sendo realizada.

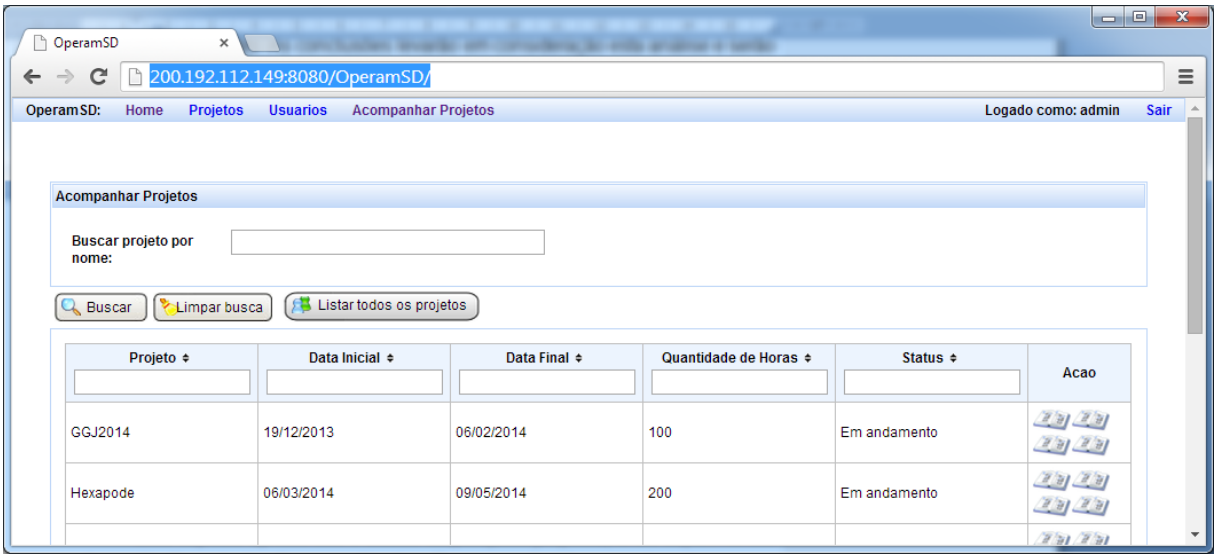


Figura 17 – Acesso para acompanhamento de projetos no servidor OPERAM.

Fonte: própria

O Apêndice 3 fornece uma descrição do OPERAM, bem como apresenta um cenário de utilização do módulo para exemplificar sua forma de funcionamento e melhorar o entendimento desta ferramenta que apoia o desenvolvimento de software de forma colaborativa e um detalhamento do OPERAM, com sua forma de

funcionamento em um projeto que envolva todos os conceitos explanados aqui, é mostrado em um cenário de utilização que é apresentado em detalhes no Apêndice 4 deste documento.

4.9. Considerações Finais

Neste capítulo foi apresentada uma visão geral do WS, sua arquitetura e composição, principais características e funcionalidades. Apresentou-se também uma pesquisa realizada com desenvolvedores e gestores de software para colher subsídios sobre itens de interesse para o desenvolvimento do WS, tais como o tamanho da equipe que estas pessoas trabalham, se as ferramentas de auxílio para a gestão do projeto e outras questões que pudessem auxiliar a formar um entendimento sobre o assunto.

O módulo desenvolvido para monitorar a programação, chamado OPERAM, também foi descrito. Ele provê meios de realizar a verificação de software e o acompanhamento da codificação com a modelagem, mostrando em tempo real as ligações semânticas criadas durante o processo.

A forma de funcionamento do WS está baseada em uma arquitetura em três camadas, que foi mostrada na visão geral do WS e que visa a facilitar a organização, seu uso e entendimento. Assim, a abordagem apresentada tem condições de auxiliar tanto os programadores como os gestores dos projetos de software na execução de suas tarefas, objetivando melhorar a forma como o software é desenvolvido.

O HACKYSTAT, uma ferramenta de código aberto para coleta e análise de dados referentes ao processo de desenvolvimento de software também foi explicitada. Mostrou-se também a forma do acompanhamento da codificação com o diagrama de classes no OPERAM, bem como as ligações semânticas que o mesmo produz. A visualização destas ligações semânticas em tempo real de codificação também é detalhada.

O próximo capítulo relata os experimentos realizados para obter dados para análise do módulo OPERAM, que foram conduzidos com alunos de uma universidade e em também em um instituto de tecnologia.

Capítulo 5

Experimentos Realizados

O trabalho proposto para esta pesquisa envolve a aplicação de diversas tecnologias, experimentos que foram conduzidos no módulo de acompanhamento da codificação, e testes de avaliação do DeSS NEPOMUK. Os objetivos deste capítulo são expor a avaliação e forma de condução dos experimentos realizados com a monitoração da programação e a gestão de projeto no módulo desenvolvido para acompanhar o desenvolvimento do código-fonte, e principalmente, obter os resultados para a comprovação das hipóteses definidas para esta pesquisa. Os experimentos foram definidos segundo uma metodologia descrita no item 5.1. Os experimentos realizados não possuem similares na literatura, por este motivo não puderam ser comparados.

5.1. Metodologia para Avaliação dos Experimentos

Inicialmente, define-se a metodologia para avaliação dos experimentos propostos. Neste sentido, o trabalho (WOHLIN *et al.,* 2000) identifica quatro métodos de pesquisa considerados como relevantes para a condução de experimentos na engenharia de software:

- Método Científico: o mundo é observado e um modelo é construído baseado na observação (ex: modelos de simulação para avaliação de desempenho);
- Método da Engenharia: as soluções existentes são estudadas e mudanças são propostas;
- Método Empírico: um modelo é proposto e avaliado por meio de estudos empíricos, como por exemplo, experimentos ou estudos de caso;
- Método Analítico: uma teoria formal é proposta e então é comparada com observações empíricas, sendo este o método mais utilizado nas áreas da computação e engenharia elétrica.

Definiu-se o método empírico como forma para conduzir os experimentos de avaliação, por ser aquele que atende os pressupostos e atividades que foram idealizadas e formalizadas para a análise do OPERAM.

Para este tipo de análise, foram definidos os procedimentos para executar os experimentos, e os procedimentos referentes à engenharia de software foram utilizados, conforme descrito em (WOHLIN *et al.,* 2000) e também no trabalho de (KITCHENHAM *et al.,* 2002), que são: elaboração do experimento, elaboração do objeto de estudo, coleta dos dados, análise dos dados coletados e apresentação dos resultados e das conclusões. Nos itens a seguir, um detalhamento é apresentado para execução de cada um destes procedimentos:

- Elaboração do experimento: inclui a apresentação do experimento, o contexto de realização, a definição dos objetivos, variáveis e instrumentos de coleta de dados que podem ser utilizados.
- Elaboração do objeto de estudo: Define exatamente o que será avaliado ou observado com a realização do experimento.
- Coleta dos dados: A coleta de dados pode ser realizada antes, durante ou após a realização do experimento. No caso do OPERAM, a coleta foi realizada durante todo o tempo estimado para a realização do experimento. Ela foi realizada por meio dos instrumentos de coleta definidos e armazenaram-se os dados para serem analisados na próxima fase do procedimento.

- Análise dos dados coletados: Para esta tarefa em específico, a forma mais usual de obter as respostas necessárias são as entrevistas e os questionários. Assim, foi definida a utilização de questionário para obter a coleta dos dados e para proceder com uma avaliação qualitativa do OPERAM. Os questionários possuem um grau de generalização e podem ser descritos como uma forma de obter dados ou informações sobre características, ações ou opiniões de um determinado grupo de pessoas. O principal intuito do questionário nesta pesquisa é verificar se o objetivo de criar um ambiente de apoio ao desenvolvimento colaborativo de software capaz de integrar semanticamente os artefatos produzidos na programação e contribuir no aumento da visibilidade dos artefatos entre os participantes da equipe de desenvolvimento foi atingido. Assim, o questionário foi o instrumento aplicado no experimento de avaliação para fornecer subsídios e assim comprovar a hipótese de trabalho formulada como "*A utilização do Workspace Semântico contribuirá na integração entre os desenvolvedores e os artefatos de projetos desenvolvidos por pequenas equipes de desenvolvimento de software*" Outra hipótese que se deseja comprovar com a aplicação do questionário no experimento de avaliação é: "*O acompanhamento da escrita do código-fonte em tempo real e de forma automática, sendo confrontada com a modelagem realizada anteriormente, amplia a visibilidade do que está sendo realizado*".
- Apresentação dos resultados e conclusões: Os resultados referem-se aos dados obtidos no contexto das respostas em relação aos objetivos do experimento. As conclusões levaram em consideração esta análise e foram utilizadas como forma de comprovar ou refutar hipóteses levantadas, como também sua viabilidade técnica como ambiente para favorecimento do *awareness* e da verificação de software no desenvolvimento de software.

Com estas considerações sobre a metodologia para avaliação dos experimentos, dois estudos foram conduzidos: um experimento piloto, que objetivava realizar uma avaliação da forma de utilização e efetividade do OPERAM como ambiente para monitorar o desenvolvimento do código-fonte em pequenas equipes. Após a conclusão deste experimento e comprovação de sua efetividade, e com a

correção de eventuais problemas ou ajustes necessários, outro experimento foi conduzido, agora com o objetivo de comprovar as hipóteses levantadas nesta pesquisa. Para este segundo experimento, um projeto de desenvolvimento de software foi elaborado e conduzido por uma equipe real de desenvolvimento em uma instituição pública que atua em pesquisa, desenvolvimento tecnológico e inovação no estado do Paraná (Instituto de Tecnologia do Paraná - TECPAR).

5.2. Experimento Piloto - Apresentação e Contexto da Realização

Para a execução do experimento piloto, uma pequena equipe composta por três desenvolvedores e um gerente de projeto foi convidada para desenvolver um sistema chamado IMC. O principal objetivo do sistema IMC é controlar o peso de um indivíduo, além da gestão das calorias consumidas, avaliando assim o índice de massa corporal (IMC) do mesmo. Para este experimento, alunos da Pontifícia Universidade Católica do Paraná (PUCPR), oriundos dos cursos de graduação em Engenharia de Computação e Sistemas de Informação, que desempenharam o papel de programadores, alternaram-se no desenvolvimento. A este grupo foi fornecido o diagrama de classes deste sistema, que foi criado na ferramenta ASTAH COMMUNITY, visto na Figura 18. A partir deste diagrama, os participantes poderiam iniciar a codificação do mesmo de forma colaborativa. A equipe tinha o prazo de 10 dias para desenvolver a aplicação.

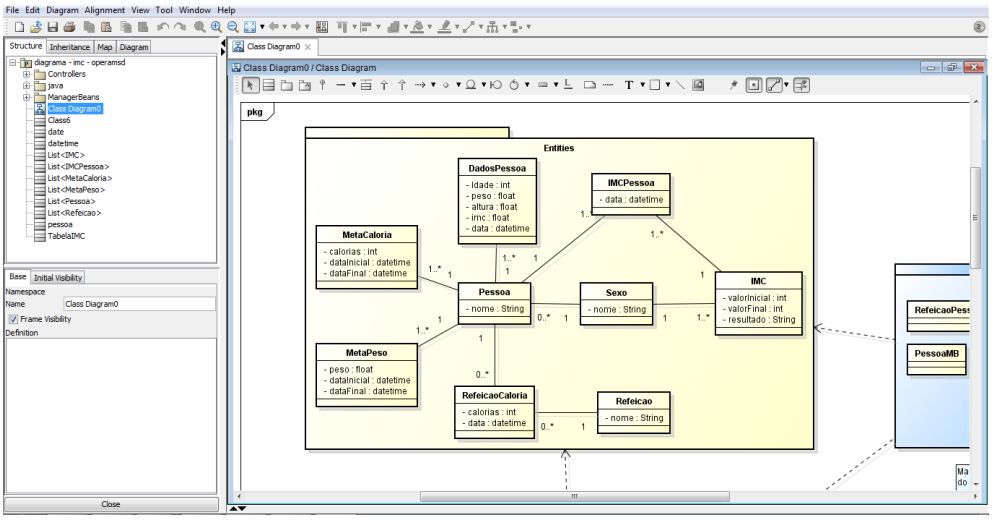


Figura 18 – Visão Parcial do Diagrama de Classes do projeto IMC criado no ASTAH.
Fonte: própria

Para auxiliar na execução do experimento piloto, um cenário foi elaborado para servir de apoio ao sistema que seria desenvolvido. O Quadro 17 apresenta o cenário utilizado para o experimento piloto de avaliação do OPERAM.

Quadro 17 – Cenário criado para o experimento de avaliação IMC.

Aplicativo para cálculo do IMC e da gestão diária de calorias
O sistema deve permitir que o usuário informe seus dados pessoais (idade, peso, sexo, altura e nome), calcule e apresente o seu IMC (peso /(altura * 2)). Deve-se atentar para o fato que o IMC é diferente para homens e mulheres. O sistema também deve registrar as refeições diárias e a quantidade de calorias ingeridas destas refeições, com a finalidade de seguir uma dieta pré-determinada de ingestão de calorias diárias e controlar o peso do usuário. Um dos objetivos é controlar as calorias que são ingeridas diariamente em cada uma das refeições pré-determinadas: café da manhã, lanche da manhã, almoço, lanche da tarde e jantar. Para realizar estes lançamentos, o usuário deve determinar anteriormente uma quantidade máxima de calorias diárias que podem ser ingeridas diariamente (chamada de calorias meta, como por exemplo, 2000 calorias). Os lançamentos das calorias devem ser realizados por refeição, sem a necessidade de discriminar os alimentos e os seus valores calóricos. O sistema deve apresentar a diferença entre as calorias efetivamente ingeridas e as calorias meta, apresentando esta diferença como saldo, para que o usuário possa acompanhar a dieta definida e assim saber se está próximo ao limite (meta) ou se já ultrapassou a meta. A função do sistema é ajudar o usuário a fazer esse controle, e o objetivo do mesmo será não ultrapassar a meta estabelecida. Outro objetivo do sistema é realizar o acompanhamento do peso do usuário. O sistema deve permitir cadastrar um objetivo meta para perda de peso e o acompanhamento deste processo. Toda vez que o peso for alterado pelo usuário, seu IMC e peso meta devem ser atualizados. O sistema deve apresentar a todo o momento a quantidade de calorias ingeridas, as calorias meta, o peso atual da pessoa, a meta desejada e seu IMC. O sistema também deve realizar um fechamento diário das calorias ingeridas e apresentar um histórico entre datas definidas pelo usuário dos totais de calorias ingeridas.

Aos participantes também foi apresentado o objetivo, contexto e forma de realização do experimento, bem como o que seria observado e avaliado na coleta de dados.

5.2.1. Coleta de Dados Descrição e Análise do Experimento Piloto IMC

Após a definição da equipe, distribuição do diagrama de classes e do cenário de uso do sistema IMC e do cadastro do mesmo no OPERAM, iniciou-se a programação por parte dos desenvolvedores. A Figura 19 mostra a tela inicial do OPERAM para acompanhamento do projeto IMC, sem que nenhum desenvolvedor tenha iniciado ainda a codificação e, portanto, sem atividades monitoradas até aquele momento.

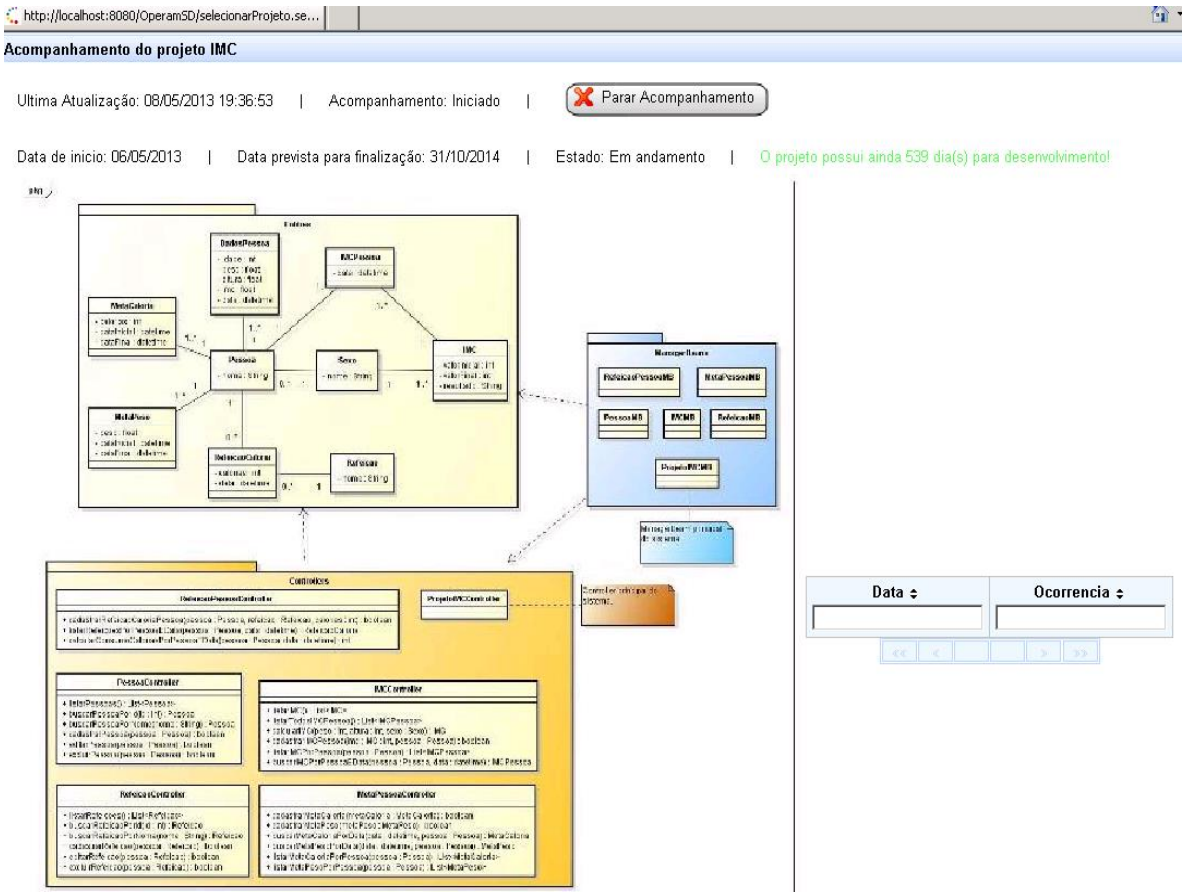


Figura 19 – Início do Experimento IMC no OPERAM.

Fonte: própria

A Figura 20 apresenta as atividades realizadas após seis dias do início do experimento, onde se pode observar o progresso no desenvolvimento do sistema obtido até aquele momento. Percebe-se que todas as classes do diagrama que foi entregue aos desenvolvedores foram criadas, conforme nota-se no lado esquerdo da figura. Conforme explicitado anteriormente, a cor verde nas classes utiliza o conceito

do KANBAN, e nos casos apontados, indicam a criação desta classe no código-fonte JAVA.

No lado direito da mesma figura, observa-se o histórico deste desenvolvimento, com as seguintes informações sendo apresentadas: data da ocorrência, usuário responsável e itens do código sendo tratado. Estas informações do histórico também utilizam o conceito do KANBAN para facilitar a visualização dos processos mostrados.

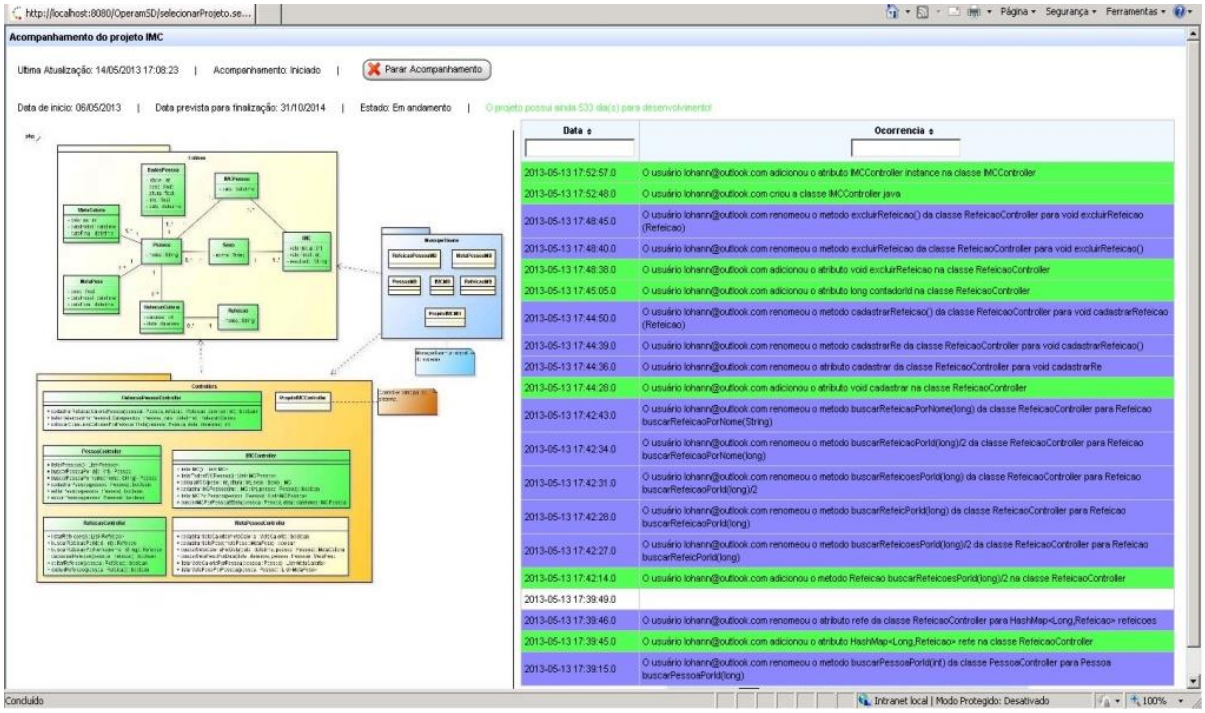


Figura 20 – Experimento IMC no OPERAM após seis dias de desenvolvimento.

Fonte: própria

A Figura 21 mostra em detalhe o histórico de desenvolvimento do sistema em um dado momento do tempo. Esta figura apresenta a criação (adição) de cinco atributos, uma classe e nove novos métodos. Também mostra que três itens foram renomeados (trocaram de nome) e todas estas alterações foram realizadas por um único usuário. As cores auxiliam a identificar o tipo de modificação que foi realizada.

Data	Ocorrencia
2013-05-10 16:14:43.0	O usuário lohann@outlook.com adicionou o metodo Date getData() na classe DadosPessoa
2013-05-10 16:14:43.0	O usuário lohann@outlook.com adicionou o metodo void setImc(float) na classe DadosPessoa
2013-05-10 16:14:43.0	O usuário lohann@outlook.com adicionou o metodo float getImc() na classe DadosPessoa
2013-05-10 16:14:43.0	O usuário lohann@outlook.com adicionou o metodo void setAltura(float) na classe DadosPessoa
2013-05-10 16:14:43.0	O usuário lohann@outlook.com adicionou o metodo float getAltura() na classe DadosPessoa
2013-05-10 16:14:43.0	O usuário lohann@outlook.com adicionou o metodo void setPeso(float) na classe DadosPessoa
2013-05-10 16:14:43.0	O usuário lohann@outlook.com adicionou o metodo float getPeso() na classe DadosPessoa
2013-05-10 16:14:43.0	O usuário lohann@outlook.com adicionou o metodo void setIdade(int) na classe DadosPessoa
2013-05-10 16:14:43.0	O usuário lohann@outlook.com adicionou o metodo int getIdade() na classe DadosPessoa
2013-05-10 16:13:54.0	O usuário lohann@outlook.com renomeou o metodo DadosPessoa() da classe DadosPessoa para DadosPessoa(Pessoa)
2013-05-10 16:13:45.0	O usuário lohann@outlook.com criou a classe Pessoa.java
2013-05-10 16:13:39.0	O usuário lohann@outlook.com renomeou o atributo p da classe DadosPessoa para Pessoa pessoa
2013-05-10 16:13:37.0	O usuário lohann@outlook.com adicionou o atributo Pessoa p na classe DadosPessoa
2013-05-10 16:12:37.0	O usuário lohann@outlook.com adicionou o metodo DadosPessoa() na classe DadosPessoa
2013-05-10 16:12:19.0	
2013-05-10 16:12:15.0	O usuário lohann@outlook.com adicionou o atributo Date data na classe DadosPessoa
2013-05-10 16:12:07.0	O usuário lohann@outlook.com adicionou o atributo float imc na classe DadosPessoa
2013-05-10 16:11:58.0	O usuário lohann@outlook.com adicionou o atributo float altura na classe DadosPessoa
2013-05-10 16:11:54.0	O usuário lohann@outlook.com adicionou o atributo float peso na classe DadosPessoa
2013-05-10 16:11:40.0	O usuário lohann@outlook.com renomeou o atributo i da classe DadosPessoa para int idade

Primeira « 17 18 19 20 21 22 23 24 25 26 » Ultima

Figura 21 – Detalhes do Histórico de um Usuário no Projeto IMC.

Fonte: própria

O módulo OPERAM também disponibiliza dados gerenciais, que podem ser utilizados pelo responsável do projeto para tomar decisões sobre a forma como o projeto está sendo conduzido, como por exemplo, deslocar pessoas para ajudar outros com dificuldades em uma dada tarefa. Estes dados estão disponíveis inspecionando o banco de dados onde o OPERAM armazena os eventos gerados durante a codificação do projeto, e assim podem ser úteis ao responsável para tomar decisões baseadas em dados obtidos do processo de interação com o OPERAM. Tais decisões podem envolver informações tais como: usuário que mais criou itens, quanto tempo foi gasto neste processo, quais itens foram mais acessados ou modificados, a quantidade de pessoas que trabalharam em um determinado componente, se itens não previstos na modelagem inicial foram criados, *break-points* inseridos e depurações realizadas. Estes são exemplos de informações disponíveis, mas outros dados podem ser disponibilizados para a tomada de decisões referentes ao projeto em desenvolvimento.

Ao verificar os eventos gerados durante a codificação do experimento IMC, pode-se concluir, por exemplo, que o Desenvolvedor 1 trabalhou durante 10 horas. Também é possível nomear o desenvolvedor que escreveu o código-fonte com mais diferenças do modelo originalmente apresentado (por exemplo, aquele desenvolvedor que criou novos métodos e atributos). Além disso, o experimento

também pode auxiliar a efetividade do OPERAM em possibilitar a verificação da programação de dados históricos realizados pela equipe, permitindo assim obter dados de interesse sobre o desenvolvimento do projeto, pois o OPERAM registra todas as atividades que os programadores realizam.

Este registro envolve desde a criação, modificação e exclusão de classes, métodos e atributos de um projeto em particular, além de outras medidas, tais como o tempo gasto em cada atividade, número de compilações de um programa, erros de execução, e outras medidas. O Quadro 18 sintetiza algumas destas estatísticas sobre o experimento IMC.

Quadro 18 – Alguns Dados Gerenciais do Experimento IMC.

Ação	Desenvolvedor1	Desenvolvedor2	Desenvolvedor3
Classes Criadas	20	05	01
Métodos Criados	92	02	-
Métodos Removidos	07	02	-
Inserção de *Break Points*	07	02	02
Execução de *Debug*	20	05	03

O experimento também possibilitou verificar a eficácia do OPERAM como um espaço de trabalho para fornecer suporte à verificação de software e ao *awareness*. Ele permite que a codificação realizada em tempo real seja confrontada com a modelagem UML executada anteriormente. O OPERAM também recolhe dados relativos à interação entre os envolvidos no processo de desenvolvimento. O Quadro 19 mostra algumas destas relações (ligações semânticas) criadas durante o desenvolvimento do experimento IMC, comprovando sua efetividade como ferramenta que fornece apoio a equipes que desenvolvem software de forma colaborativa.

Quadro 19 – Algumas Ligações Semânticas do Experimento IMC.

Ligação Semântica	Descrição
é-responsável-por	Projeto e Gerente do Projeto
última-atualização-por	CaloriaMeta.java e Desenvolvedor 1
última-atualização-por	Pessoa.java e Desenvolvedor 2
é-relacionado-a	CaloriaMeta.java e Item Diagrama de Classe
é-relacionado-a	DadosPessoa.java e Item Diagrama de Classe
item-criado-por	IMC.java e Desenvolvedor 3
é-coautor-de	MetaCaloria.java e Desenvolvedor 1
é-coautor-de	MetaCaloria.java e Desenvolvedor 3

Outro componente testado neste experimento piloto foi a criação das ligações semânticas entre o código-fonte e a modelagem UML. A Figura 22 mostra o relacionamento entre o framework HACKYSTAT, a IDE ECLIPSE e o diagrama de classes do ASTAH do projeto IMC para criar a ligação semântica "é-relacionado-a" capturada no HACKYSTAT por meio do sensor chamado SENSORBASE 8.4.127 e de seu relacionamento com parte do diagrama de classes do projeto IMC.

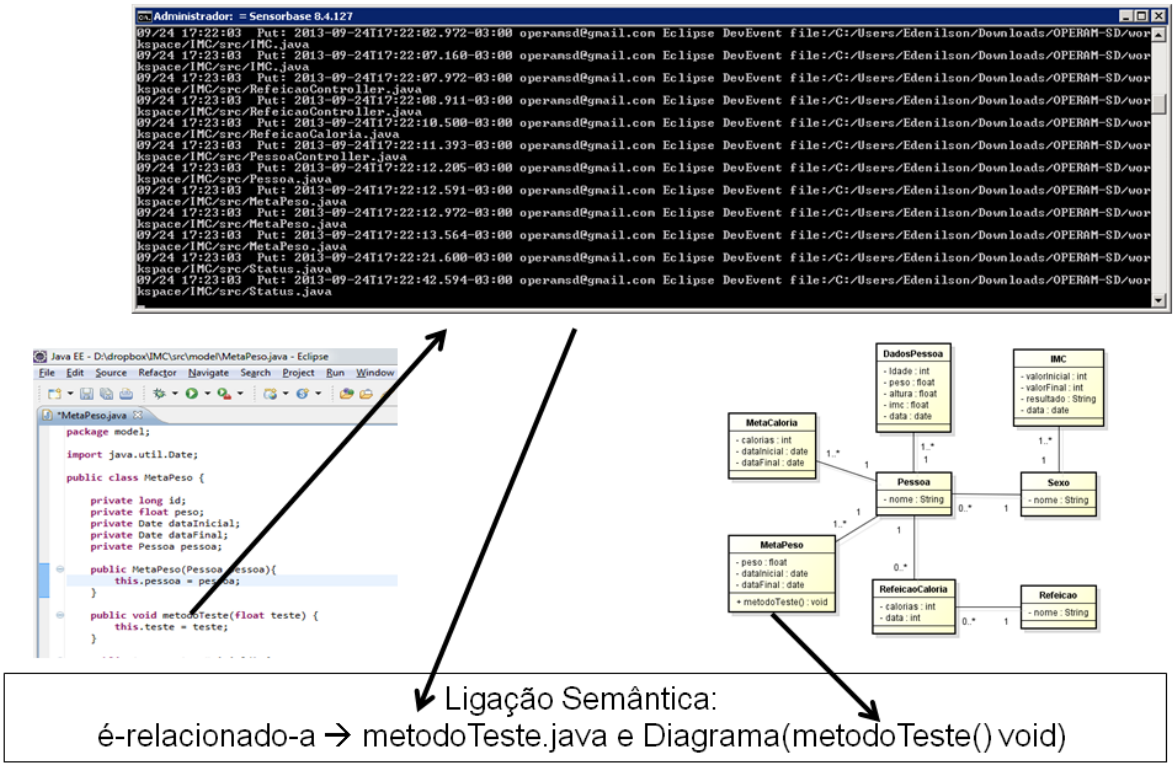


Figura 22 – Ligação Semântica entre o Código-fonte e Diagrama UML do Experimento IMC.

Fonte: própria

Estas ligações semânticas podem ser acompanhadas em tempo real de codificação por meio de relatórios on-line disponibilizados no OPERAM. Um exemplo desta forma de acompanhamento é apresentado na Figura 23, onde as interações entre as classes e os usuários que realizaram a programação são mostradas. Neste relatório, a forma gráfica foi escolhida para mostrar os relacionamentos entre todas as Classes que foram manipuladas por todos os usuários que foram adicionados como programadores no projeto. Esta forma de visualização permite verificar todos

os itens que fazem parte do projeto de software sendo manipulados em um único local, de forma a facilitar sua análise e interpretação dos dados. Os Métodos e os Atributos dos projetos monitorados, também podem ser visualizados da mesma forma. Nesta figura algumas informações gerenciais podem ser visualizadas: o segundo desenvolvedor foi aquele que mais interações realizou com as classes do sistema IMC e também foi o que mais tempo trabalhou em quase todas estas classes; em contrapartida, o último desenvolvedor foi o que menos contribuiu, tendo acessado apenas uma única classe. Sobre as ligações semânticas, algumas considerações podem ser observadas, como por exemplo, as mesmas cores das classes nos diferentes desenvolvedores: isto indica que eles tiveram interações nas mesmas classes, e, portanto trabalharam no mesmo item do diagrama de classes. Neste caso são coautores da mesma classe.

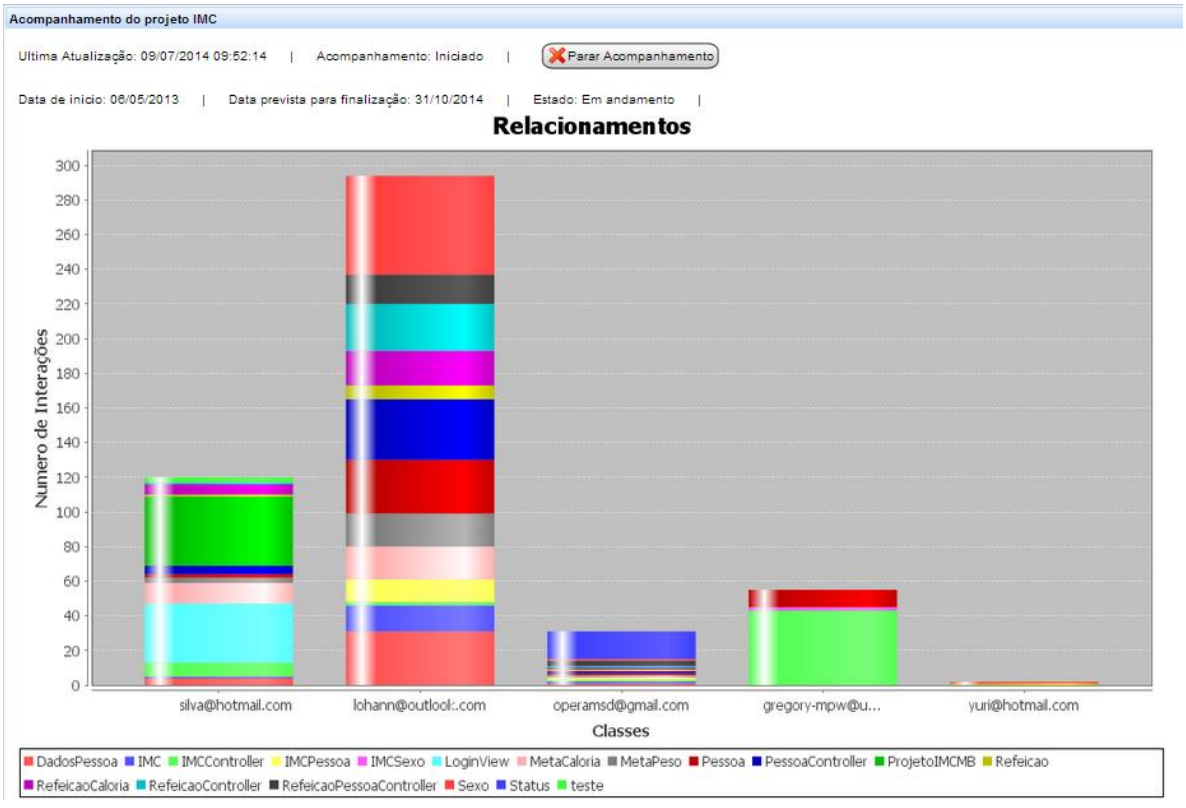


Figura 23 – Ligações Semânticas do Experimento IMC Mostradas no OPERAM.

Fonte: própria

5.2.2. Resultados e Conclusões do Experimento Piloto

Os resultados do experimento piloto mostraram a efetividade do módulo desenvolvido para acompanhamento da programação. A visualização da codificação mostrou-se eficaz, monitorando os itens que foram determinados para serem observados, a saber: as classes, métodos e os atributos do projeto de software sob monitoração. O histórico da codificação também se apresentou consistente e a linha do tempo da programação é funcional, apresentando os itens corretamente. As ligações semânticas entre o código e a modelagem foram criadas e são apresentadas aos usuários.

O experimento piloto também levantou uma preocupação no que diz respeito ao monitoramento de um projeto maior, onde, por exemplo, o número de classes seja elevado. Esta preocupação é em relação à forma da apresentação gráfica concebida para mostrar os resultados em tempo real, que pode ser comprometida, já que muitos itens podem "poluir" a tela de exibição, e assim dificultar a visualização destes dados.

Os resultados parciais obtidos com este experimento de avaliação são detalhados em (SILVA *et al.,* 2013), e permitiram conduzir outro experimento, agora para avaliação do OPERAM em um ambiente real de desenvolvimento de software. Mesmo com os resultados parciais obtidos com a monitoração, histórico de programação, visualização da codificação e criação das ligações semânticas executadas neste experimento, existem indícios positivos de que as hipóteses levantadas nesta pesquisa podem ser verificadas. Na próxima seção, o experimento de validação é descrito e as conclusões com sua finalização são apresentadas.

5.3. Experimento de Avaliação - Apresentação e Contexto da Realização

O experimento de avaliação do OPERAM foi realizado nos laboratórios do TECPAR (Instituto de Tecnologia do Paraná), uma empresa pública e vinculada à Secretaria de Estado da Ciência, Tecnologia e Ensino Superior do Paraná. O projeto

desenvolvido foi denominado HEXAPODE, o qual consiste no controle de uma plataforma robótica móvel, composta por um robô hexápode (seis patas). Para o controle deste robô, foi utilizado o ARDUINO MEGA (ATmega2560), um sensor ultrassônico para detecção de obstáculos e um módulo XBee (módulos de rádio frequência que fazem comunicações no padrão ZigBee IEEE 802.15.4) para comunicação serial sem fio.

O sistema modelado consiste no controle externo deste robô, onde uma trajetória específica pode ser descrita e repassada ao robô para ser executada. O acompanhamento da sua execução, por meio da trajetória realizada e comparação com a rota planejada anteriormente, são alguns dos objetivos que se pretende obter com o desenvolvimento do projeto.

O acompanhamento pelos usuários vinculados ao projeto HEXAPODE é realizado por meio de uma conexão HTTP, disponível no endereço web `http://200.192.112.149:8080/OperamSD`. Interessados em acompanhar projetos gerenciados pelo OPERAM podem utilizar o mesmo endereço, e por meio do usuário convidado, pode-se visualizar o andamento dos todos os projetos liberados como sendo de domínio público. O projeto HEXAPODE encontra-se disponível para visualização e acompanhamento.

Para início do projeto, foi definido, juntamente com a equipe de desenvolvimento do TECPAR, o Diagrama de Classes do experimento HEXAPODE. Este diagrama foi elaborado de forma completa, prevendo-se itens que atualmente não fazem parte do projeto, mas podem ser agregados futuramente sem causar prejuízos à modelagem elaborada, como por exemplo, um sensor de calor. Para a criação deste diagrama de classes, um cenário de utilização foi elaborado pelos profissionais do TECPAR (Quadro 20) e serviu de base para a elaboração do diagrama que seria inserido no OPERAM.

Quadro 20 – Cenário do experimento de avaliação HEXAPODE.

TECPAR - Instituto de Tecnologia do Paraná
ESI - Centro de Engenharia de Sistemas Inteligentes

Projeto:Planejamento e monitoramento de rotas para um robô móvel.

Cenário:
Dispomos no laboratório de uma plataforma robótica móvel composta por um robô hexápode (6 patas), controlado por um Arduino Mega (ATmega2560), um sensor ultra sônico para detecção de obstáculos e um módulo XBee para comunicação serial sem fio com um computador externo (figura 1).
Está em implantação a inclusão de outros sensores na plataforma como acelerômetro, bússola, etc.
Gostaríamos de desenvolver um módulo externo onde uma trajetória específica pudesse ser descrita ou desenhada, transferindo a mesma para o Robô, e acompanhando a sua execução, recebendo a trajetória realizada efetivamente pelo robô e comparando-a com a rota planejada.

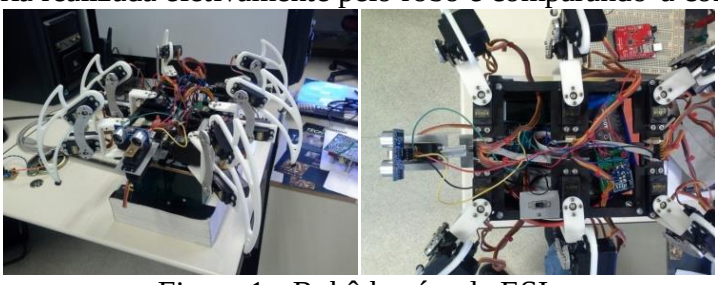


Figura 1 - Robô hexápode ESI.

A partir deste acréscimo de sensores à plataforma, se poderia pensar em novas tarefas para o robô como o desvio automático de um obstáculo que lhe impede a realização de determinada rota, retomando a rota original após o desvio.
Este módulo será desenvolvido em JAVA.

Facilidades:
1. Interface gráfica para desenho da rota a ser executada pelo Robô - arquivo;
2. Cadastro e configuração dos sensores disponíveis na plataforma - arquivo;
3. Tradução da rota e sequência de passos para o Arduino;
4. Comunicação bidirecional serial entre o módulo externo e o Robô - XBee;
5. Retorno da posição atual do Robô;
6. Recálculo da rota, quando necessário realizar um desvio.

Após a definição do cenário, o diagrama de classes mostrado na Figura 24 foi definido como sendo o diagrama representativo do sistema que a equipe de desenvolvimento iria desenvolver em seus laboratórios.

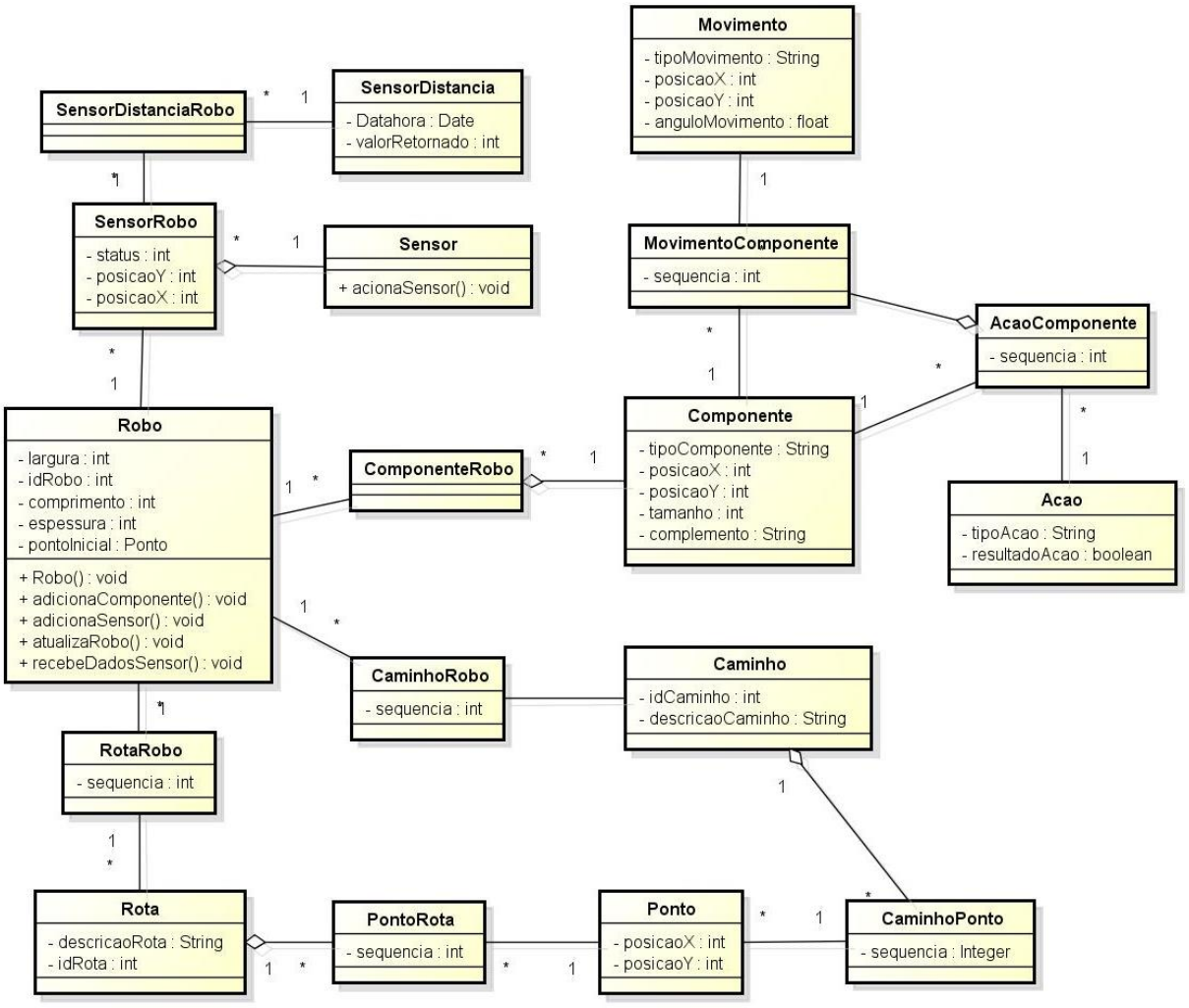


Figura 24 – Diagrama de Classes do Projeto HEXAPODE.

Fonte: própria

Este diagrama foi então adicionado ao OPERAM, os programadores cadastrados e vinculados ao projeto. Neste caso, como se trata de um projeto real de desenvolvimento em uma instituição pública, os programadores foram identificados com DEVELOPER932, DEVELOPER933, DEVELOPER934 e DEVELOPER935, e ligados aos usuários reais, a fim de não identificar estes usuários por seus verdadeiros nomes e preservar também sua identificação na divulgação dos resultados do experimento. Destes quatro usuários, três participaram diretamente da programação e um ficou a cargo da modelagem do sistema HEXAPODE e da gestão do projeto no módulo. O tempo estimado para implementação foi de um total de 100 horas. Para a implementação foi utilizada a linguagem de programação JAVA e a IDE ECLIPSE com o sensor HACKYSTAT

acoplado. Neste experimento, toda a distribuição do trabalho a ser realizado pela equipe ficou a cargo do próprio responsável no TECPAR. Este posicionamento foi adotado para não gerar distorções ou mesmo perturbações e aproximar o experimento do que é conduzido na realidade quando um projeto é posto em prática. Desta forma, todas as decisões do projeto, sobre a divisão do trabalho, tempo de dedicação ao projeto, quais desenvolvedores trabalhariam em partes específicas da modelagem e da codificação ficou a cargo da equipe do TECPAR.

5.3.1. Coleta de Dados, Descrição e Análise do Experimento de Avaliação HEXAPODE

A coleta dos dados foi realizada durante o tempo estipulado para desenvolvimento do projeto HEXAPODE nas dependências do TECPAR (aproximadamente dois meses). A Figura 25 mostra as informações referentes à codificação realizada em um determinado momento, onde se observa a criação de todas as classes (lado esquerdo da figura, com todas as classes na cor verde, o que indica que todas foram criadas na codificação do sistema).

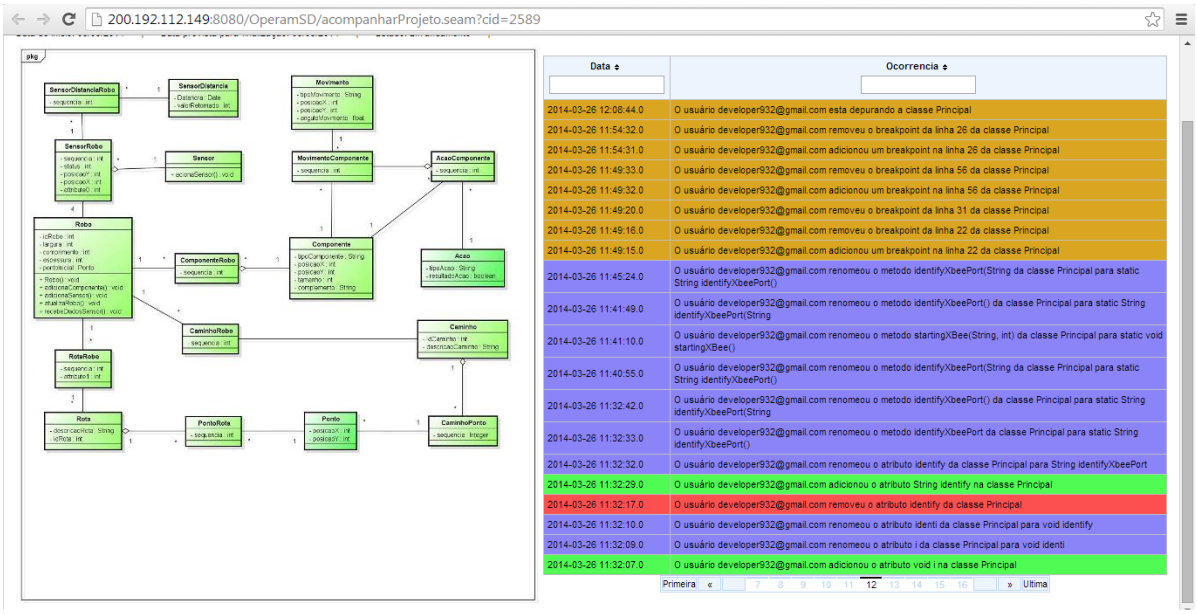


Figura 25 – Acompanhamento da Execução do Projeto HEXAPODE.

Fonte: própria

O histórico da programação mostra também o trabalho desenvolvido naquele período do tempo, indicado no nome do desenvolvedor, o item no qual o mesmo

está realizando a codificação e as informações de data e hora em que o processo foi realizado (lado direito da figura).

Além do acompanhamento mostrado na Figura 25, a análise do desenvolvimento pode ser realizada por meio de outros relatórios *on-line* e que fornecem informações a respeito das ligações semânticas que ocorrem durante a programação do código-fonte. A Figura 26 mostra, com atualização em tempo real, os relacionamentos entre todos os participantes do projeto e as classes que estes estão manipulando. A representação de cada desenvolvedor é ligada também a cada Classe que o mesmo manipulou, e o número mostrado nesta ligação, mostra a quantidade de interações que este usuário realizou naquele item.

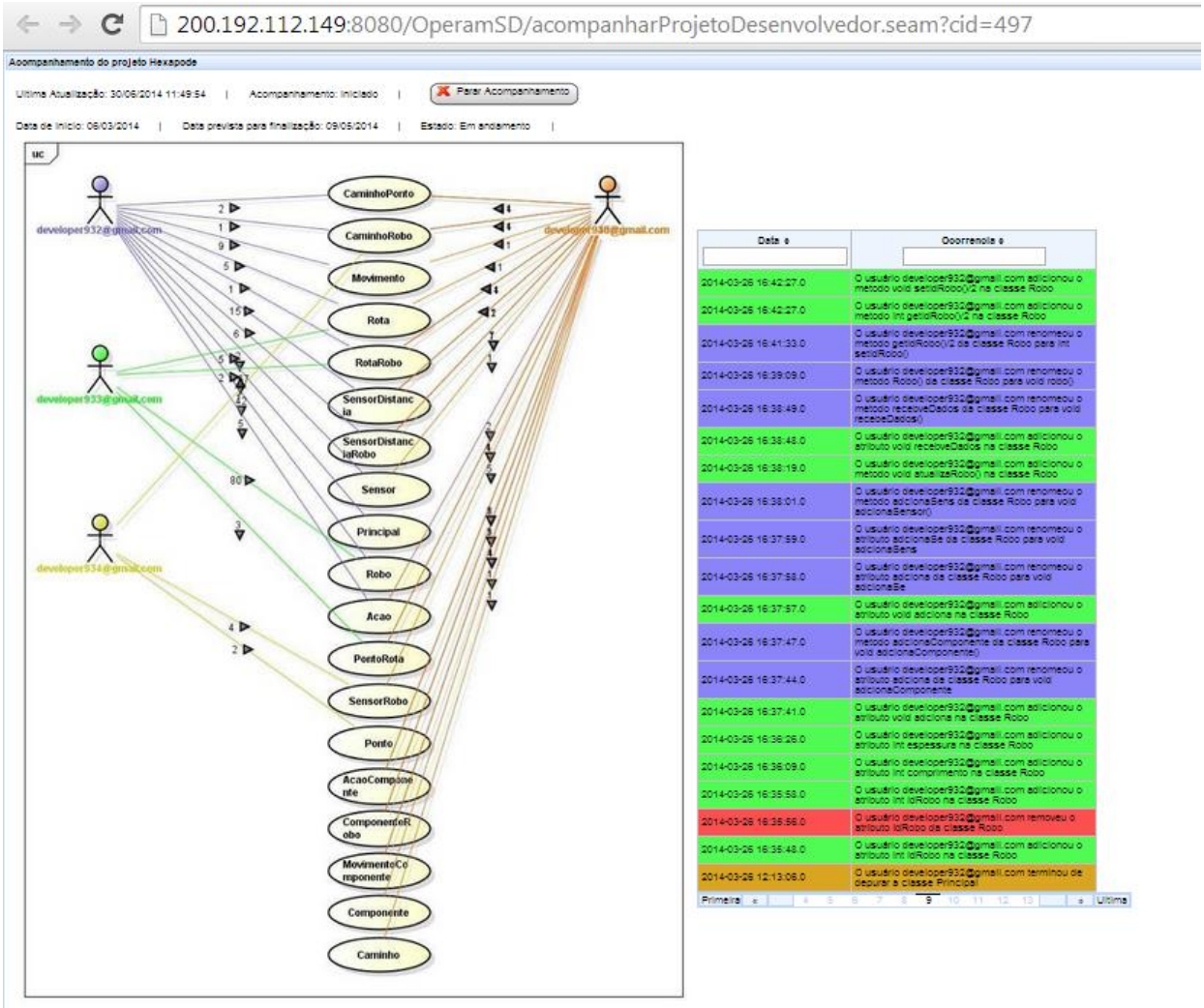


Figura 26 – Apresentação em Tempo Real do Relacionamento Entre os Desenvolvedores em Itens (Classes) do Projeto HEXAPODE.

Fonte: própria

Outros dois modos de visualizar os relacionamentos são por meio da visualização de gráficos disponibilizados pelo OPERAM. O primeiro modo é mostrado na Figura 27, onde visualização das ligações semânticas entre os desenvolvedores as classes do sistema HEXAPODE são mostradas.

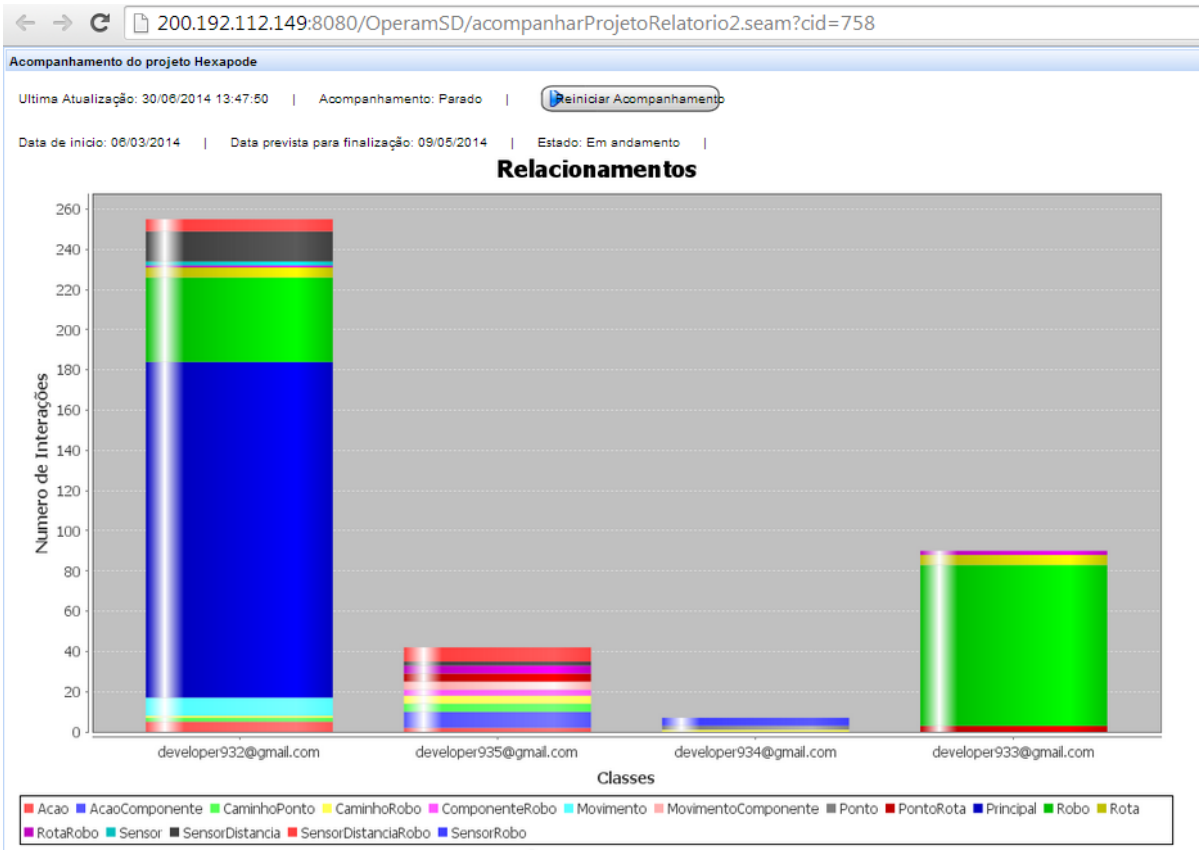


Figura 27 – Gráfico em tempo real entre os desenvolvedores e Classes do Projeto HEXAPODE.

Fonte: própria.

Outro exemplo de visualizar as ligações semânticas pode ser observado na Figura 28 onde os atributos manipulados por dois desenvolvedores são mostrados graficamente.

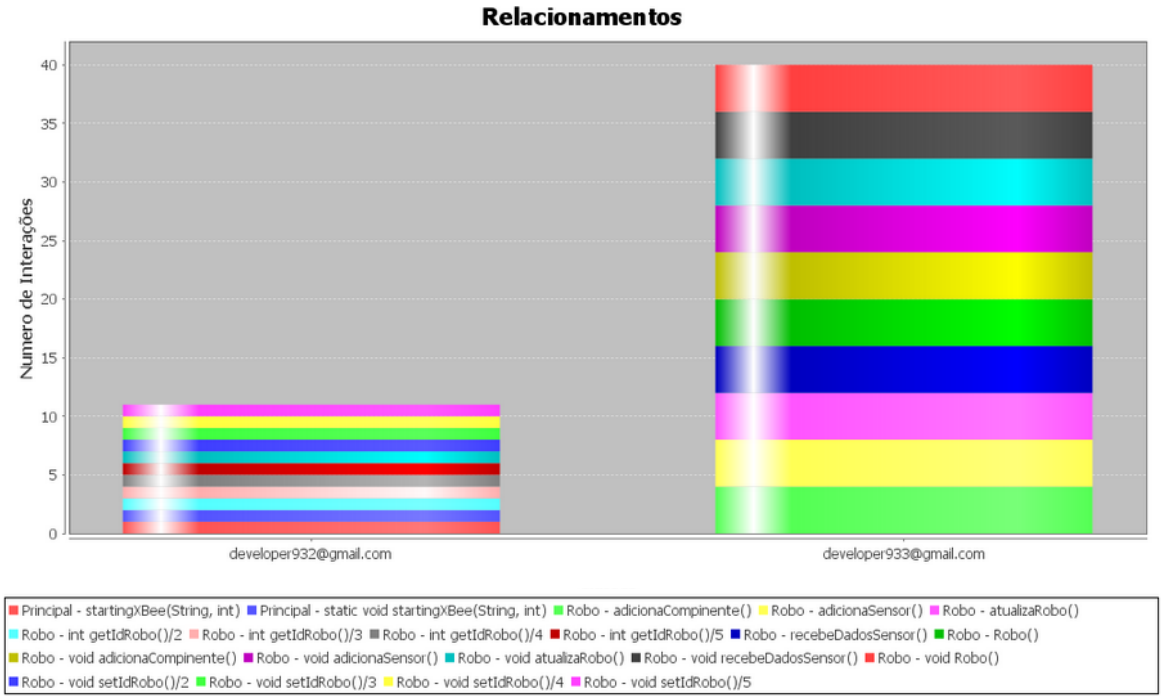


Figura 28 – Gráfico em tempo real - Desenvolvedores e Atributos do Projeto.

Fonte: própria.

O segundo modo de visualizar as ligações semânticas no OPERAM é mostrada na Figura 29.

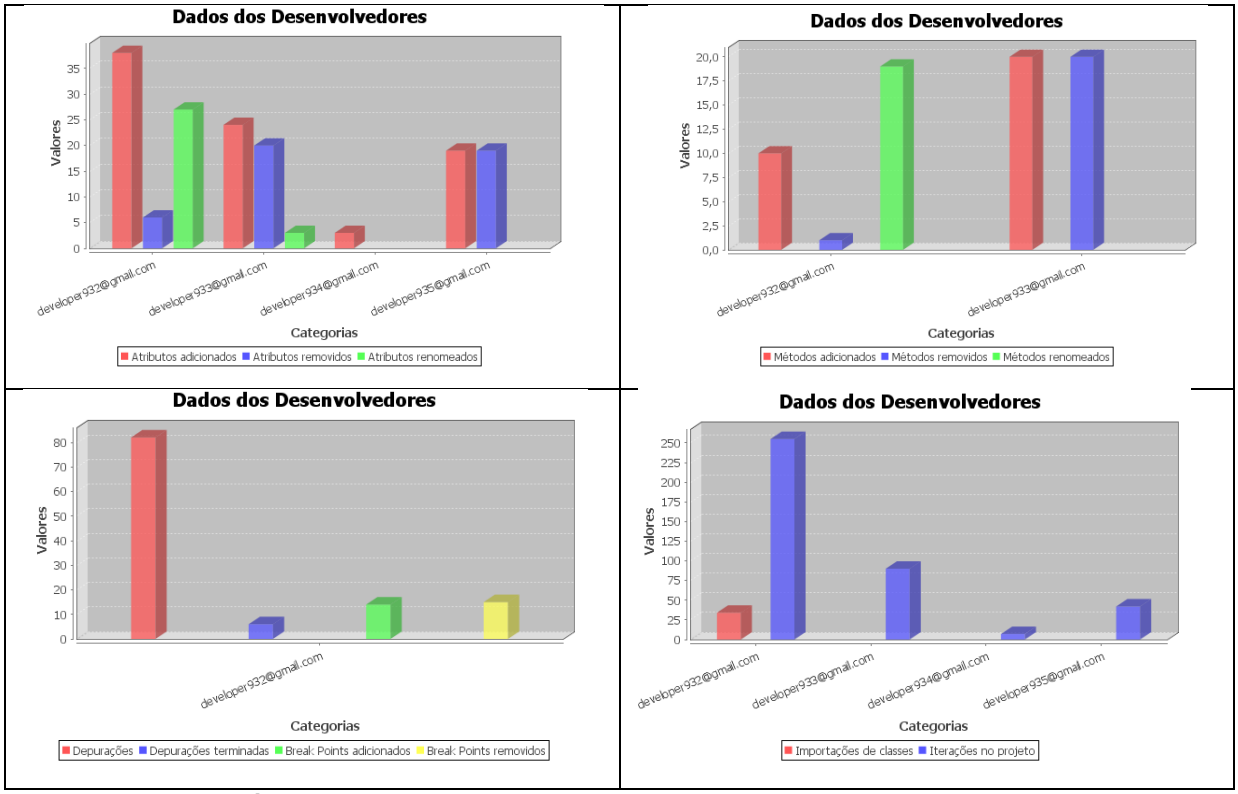


Figura 29 – Gráficos entre os desenvolvedores e diversos itens do projeto HEXAPODE.

Fonte: própria

Nesta figura, apresentam-se outros relacionamentos entre os usuários e itens do projeto que também são disponibilizados de forma on-line pelo OPERAM. Na parte superior estão sintetizados os dados referentes à manipulação dos atributos e dos métodos do projeto HEXAPODE. A parte inferior concentra os dados referentes às depurações realizadas no projeto e também as importações de classes que foram executadas pelos usuários.

Informações estatísticas também estão disponíveis na forma de relatórios do sistema, conforme Quadro 21, que mostra os principais itens manipulados na codificação: a criação, remoção e troca do nome de classes, métodos e atributos do projeto HEXAPODE, obtidos em um determinado momento da execução do projeto.

Quadro 21 – Itens Manipulados no Experimento HEXAPODE.

Ação	Classes	Métodos	Atributos
Criar	07	30	63
Renomear	-	17	30
Remover	01	21	26

O Quadro 22 mostra os dados estatísticos do desenvolvimento do projeto HEXAPODE, referentes aos totais dos itens manipulados e que foram apresentados no quadro anterior:

Quadro 22 – Estatísticas dos Desenvolvedores no Experimento HEXAPODE.

Ação	Desenvolvedor932	Desenvolvedor933	Desenvolvedor934
Classes Criadas	3	3	1
Classes Renomeadas	-	-	-
Classes Removidas	-	-	1
Métodos Criados	10	20	-
Métodos Renomeados	15	2	-
Métodos Removidos	1	16	-
Atributos Criados	38	22	3
Atributos Renomedos	30	-	-
Atributos Removidos	6	20	-
Percentual de Participação	80,61	8,16	11,23

5.3.2. Avaliação do Experimento HEXAPODE

Conforme definido na seção 5.1 deste documento, o meio escolhido para avaliação do experimento foi a aplicação de um questionário aos participantes do projeto de avaliação HEXAPODE, que foi desenvolvido nos laboratórios do TECPAR. Este questionário e os percentuais de respostas obtidos são mostrados no Quadro 23, onde se observa que ele é composto por dez itens (baseados na Escala de Likert), onde os pesquisados especificam seu nível de concordância com uma afirmação, que varia de CONCORDO FORTEMENTE a DISCORDO FORTEMENTE. O questionário era ainda composto por mais duas questões abertas, onde os pesquisados podem emitir sua opinião sobre a ferramenta ou indicar alguma sugestão de melhoria ou uma crítica sobre a mesma. O questionário foi respondido por três integrantes do projeto que participaram diretamente da codificação do experimento HEXAPODE.

Buscou-se nas primeiras cinco questões, validar o OPERAM sob a perspectiva de suas funcionalidades específicas e características como ferramenta para apoiar o desenvolvimento colaborativo de software. Na análise das respostas fornecidas, o tópico 01 mostra que a maioria concorda que a ferramenta está integrada a IDE ECLIPSE, e que de forma não intrusiva, recolhe dados dos participantes. Um participante discordou desta integração, fato que sugere que um aprimoramento pode ser feito, como por exemplo, apresentar as estatísticas do projeto dentro da própria IDE do ECLIPSE, por meio de uma aba na guia de programação, o que aumentaria a integração do OPERAM na ferramenta de programação.

Nos tópicos 02, 03 e 04 do questionário buscava-se conhecer a opinião a respeito da apresentação gráfica dos dados, da linha do tempo da programação e da visualização das modificações em tempo real de codificação dos programas, e as respostas ficaram entre CONCORDO TOTALMENTE e CONCORDO, atestando que os itens mostravam estes conceitos e existiam de forma satisfatória no módulo desenvolvido. Como estas questões estavam atreladas com a validação da ferramenta, pode-se considerar que o ambiente proposto suporta os processos de gestão dos artefatos que são manipulados no ambiente.

Quadro 23 – Questionário e Resultados de Avaliação do Experimento HEXAPODE.

OBJETIVO: Verificar a efetividade da ferramenta e sua utilidade no desenvolvimento de software. Para responder, escolha entre as opções de cada resposta, aquela que melhor representa sua opinião sobre o tópico apresentado.

VALIDAÇÃO E AVALIAÇÃO SOBRE O WORSPACE SEMÂNTICO

Itens sobre o Workspace Semântico	Concordo Fortemente	Concordo	Indiferente	Discordo	Discordo Fortemente
01 - A ferramenta funciona integrada a IDE Eclipse, de forma não intrusiva e recolhendo dados de todos os participantes do projeto de desenvolvimento de software.	66,66			33,34	
02 - A ferramenta apresenta a modelagem do sistema em desenvolvimento, de forma gráfica, por meio do diagrama de classes da UML.	33,34	66,66			
03 - Existe a apresentação das atividades desenvolvidas pelos programadores em uma linha de tempo, que possibilita escolher entre datas e visualizar o trabalho desenvolvido por cada componente da equipe.	66,66	33,34			
04 - A ferramenta mostra, de forma gráfica e em tempo real, a visualização das modificações realizadas durante a programação do código-fonte.	66,66	33,34			
05 - A ferramenta possibilita diferenciar situações como, por exemplo, a criação, eliminação e alteração de itens do projeto, por meio de cores e em tempo real de programação.	33,34	66,66			
06 - O acompanhamento do desenvolvimento do software de forma visual é útil para apoiar os participantes do projeto de software.		100,00			
07 - As ligações semânticas entre o diagrama de classes da UML e a codificação realizada em tempo real, são de utilidade para fornecer informações aos participantes do projeto de software.		100,00			
08 - A utilização de cores para diferenciar as alterações realizadas no código-fonte são importantes para fornecer suporte aos participantes do projeto de software.	33,34	66,66			
09 - A ferramenta é útil para os participantes de um desenvolvimento colaborativo de software.	33,34	66,66			
10 - A colaboração entre os participantes aumentou com a utilização da ferramenta.	66,66	33,34			

11 - Se você desejar e for possível, utilize o espaço abaixo ou o utilize o verso desta folha para escrever sua opinião sobre a ferramenta:

12 - Se você desejar expressar alguma sugestão de melhoria ou crítica sobre a ferramenta, escreva no espaço abaixo ou utilize o verso desta folha para escrever:

Já o tópico 05 visava obter a opinião com o trabalho relacionado à criação, eliminação e renomeação dos itens do projeto de software, tais como classes, métodos e atributos, por meio da utilização das cores para indicar as mudanças que aconteciam em tempo real de codificação. Para este tópico, 66,66% das respostas foram marcadas na opção CONCORDO e 33,34% na opção CONCORDO TOTALMENTE, validando desta forma o tópico como presente no módulo avaliado. Esta questão também auxilia indiretamente a corroborar a importância do *awareness* no acompanhamento da codificação. Esta diferenciação existente com a utilização de cores mostrando a evolução de itens do código e da modelagem fornece sinais da importância da utilização deste conceito na ferramenta na visão das pessoas que responderam ao questionário. Como o *awareness* diz respeito a estar ciente do que está acontecendo no entorno do trabalho realizado pelos componentes de uma equipe, a possibilidade de diferenciar estas situações no módulo foi apontada de forma positiva por todos que responderam ao questionário.

Os próximos cinco tópicos do questionário tratavam da avaliação da ferramenta, sob a ótica da utilidade da mesma para fornecer suporte à atividade de criação de software. Neste sentido, o tópico 06 investigava a utilidade do acompanhamento do desenvolvimento de software de forma visual para os participantes do projeto, e a totalidade das respostas foi marcada na opção CONCORDO. Esta questão auxilia a verificar a importância do auxílio visual no acompanhamento da codificação para apoiar o desenvolvimento de software.

O tópico 07 referia-se a utilidade das ligações semânticas criadas entre a modelagem e o código-fonte durante a produção do software, e 100% das respostas obtidas foram marcadas na opção CONCORDO, o que dá indícios da utilidade das ligações semânticas criadas no módulo desenvolvido. De forma indireta, as respostas também fornecem elementos para respaldar a importância da verificação de software que o módulo oferece com estas informações disponibilizadas para os participantes da equipe de desenvolvimento.

A utilização das cores para mostrar as alterações realizadas no código-fonte foi avaliada no tópico 08. As respostas variaram entre CONCORDO TOTALMENTE (33,34%) e CONCORDO (66,66%), demonstrando a aceitação deste tópico. Importante ressaltar que o tópico 05 deste questionário tratava da existência da

utilização de cores (técnica do KANBAN), e o tópico 08 abordava questões referentes ao benefício desta técnica ser utilizada no módulo desenvolvido.

O tópico 09 buscava comprovar a utilidade da ferramenta para o desenvolvimento colaborativo de software, e as indicações obtidas com o questionário foram de 33,34% para CONCORDO TOTALMENTE e de 66,66% para CONCORDO, o que também confirma a utilidade do módulo desenvolvido para acompanhar a codificação.

Já o tópico 10 questionava sobre o aumento da colaboração com a utilização da ferramenta, complementando assim o tópico 09 do questionário, e a maioria dos participantes optou por responder que concordavam totalmente (66,66%) com o aumento da colaboração em virtude da utilização da ferramenta, e 33,34% consideraram a concordância com o tópico. Estas respostas neste ponto do questionário tem uma importância ampla, pois o aumento da colaboração é o principal objetivo que se pretende dar suporte e fornecer condições para que ocorra com a utilização do ambiente proposto nesta pesquisa. Com a totalidade das respostas na parte positiva do questionário, fornecem uma forte indicação que a colaboração pode ser aprimorada com a utilização do ambiente desenvolvido, e este objetivo tem condições de ser alcançado.

Por fim, duas questões que buscavam obter opiniões sobre a ferramenta (tópico 10) ou sugestão de melhoria ou crítica sobre a mesma (tópico 11), foram apresentadas e o Quadro 24 exibe as respostas obtidas por dois participantes da pesquisa que fizeram suas considerações.

Estas considerações são importantes e podem ser efetivadas na ferramenta, pois como se trata de um protótipo, existem condições de serem implementadas em melhorias futuras. Estas sugestões indicam pontos que devem receber atenção, especialmente porque advêm de usuários que utilizaram por um determinado período de tempo a ferramenta, e em um projeto conduzido por uma equipe real de desenvolvimento de software, e desta forma têm possibilidade de fornecer contribuições para aprimorar suas funcionalidades e características.

Quadro 24 – Considerações dos desenvolvedores do Projeto HEXAPODE.

Questões	Opiniões e Sugestões
Tópico 11: Opinião sobre a ferramenta	A ferramenta não toma tempo do programador, tanto em configuração ou inicialização. Trabalha de forma imperceptível com baixo custo computacional, sem paradas ou atrasos. Quanto à sua funcionalidade, apesar de ser apenas um protótipo, traz benefícios quanto ao acompanhamento do projeto e a evolução de cada um dos desenvolvedores, além de gerar um histórico do desenvolvimento, mostrando o que foi gerado, apagado e modificado nos códigos criados.
	O 'Workspace Semântico' efetivamente cumpre com suas metas de não ser intrusivo, não 'atrapalha' o desenvolvedor de software na sua rotina de trabalho, e aporta informações importantes para gestão da equipe e para estabelecer possibilidades de colaboração entre seus membros. Foi uma boa experiência utilizá-la.
Tópico 12: Sugestão ou crítica sobre a ferramenta	Como sugestão, apesar de não ser relacionado à pesquisa, de melhoria modo a torná-la uma ferramenta mais eficiente no acompanhamento de projetos, seria: • possibilitar ao gerente de projeto definir classes/métodos tidos como importantes para o desenvolvimento do projeto; • apresentar gráficos destas classes/métodos isoladamente, de forma que o gerente de projeto possa ter uma visualização mais clara do seu desenvolvimento; • acrescentar uma linha de tempo e dependências para cada uma das classes, mais ou menos como um gráfico de Gantt.
	A questão da visualização do tratamento das informações coletadas pela ferramenta ainda poderia ser melhorada, por exemplo, como a visualização de apenas uma classe escolhida e o engajamento dos membros da equipe nela, ou ainda com a possibilidade de determinar as classes mais críticas e/ou importantes, formando uma linha de base (base-line) do projeto, e acompanhar apenas a evolução destas classes.

5.3.3. Resultados e Conclusões do Experimento de Avaliação

O experimento de avaliação conduzido buscava auxiliar validar as hipóteses da pesquisa aqui conduzida. A primeira hipótese foi definida como "*A utilização do Workspace Semântico contribuirá na integração entre os desenvolvedores e os artefatos de projetos desenvolvidos por pequenas equipes de desenvolvimento de software*". A segunda hipótese levantada foi definida como "*O acompanhamento da escrita do código-fonte em tempo real e de forma automática, sendo confrontada com a modelagem realizada anteriormente, amplia a visibilidade do que está sendo realizado*?".

Considerando o experimento de avaliação executado, e as indicações positivas obtidas com o questionamento realizado, fica evidenciado que o módulo desenvolvido para o ambiente proposto, pode contribuir em diversos aspectos aos membros de uma equipe de desenvolvimento de software na realização do seu trabalho, tais como: verificar o andamento da codificação com o projeto em

andamento, estar ciente da programação praticada por seus pares em tempo real, acompanhar o desenvolvimento do trabalho de toda a equipe de forma visual também por meio de relatórios gerenciais e beneficiar-se da utilidade das ligações semânticas produzidas no módulo OPERAM.

Portanto, considerando o experimento realizado, fica evidenciado que o uso de ambiente pode contribuir em diversas perspectivas aos membros de uma equipe de desenvolvimento de software no tocante ao aprimoramento do *awareness* e da verificação de software que estes usuários realizam. Da mesma forma, o experimento permitiu deduzir que o acompanhamento da codificação confrontando-a com a modelagem realizada anteriormente, tem condições de ampliar a visibilidade dos artefatos manipulados pelos desenvolvedores.

5.4. Considerações Finais

Este capítulo apresentou os resultados dos experimentos de avaliação que foram conduzidos para avaliação do ambiente. Dois experimentos (experimento piloto e experimento de avaliação) foram realizados considerando principalmente o módulo OPERAM, que visavam verificar a monitoração do código-fonte e a gestão do projeto a fim de comprovar a viabilidade do ambiente proposto nesta pesquisa e comprovar as hipóteses formuladas para este pesquisa.

Também se apresentou a metodologia para conduzir e avaliar os dois experimentos, bem como os resultados, que permitiram analisar e discutir as hipóteses levantadas nesta pesquisa.

Em conjunto com as comprovações, os resultados obtidos com o questionário aplicado também permitiram concluir os pontos referentes ao *awareness* e a verificação de software, ou seja, o ambiente também contribui para aprimorar o *awareness* das equipes de desenvolvimento de software, e o acompanhamento da escrita do código-fonte com a ampliação da visibilidade do que está sendo desenvolvido, contribui para aperfeiçoar a verificação de software. O próximo capítulo apresenta as conclusões obtidas com a execução deste trabalho de pesquisa.

Capítulo 6

Conclusões

Apresentam-se neste capítulo as conclusões da pesquisa desenvolvida, suas contribuições e relevância, as limitações e trabalhos futuros que podem ser conduzidos para aprimorar o ambiente proposto.

Com as considerações apresentadas, em relação às hipóteses definidas na seção 1.4, concluiu-se que a primeira hipótese não pôde ser totalmente comprovada no tocante a integração entre os desenvolvedores e os artefatos de projetos, mas apresenta fortes indícios de que a hipótese é factível de ser realizada, ou seja, o ambiente pode auxiliar na integração dos artefatos e no acompanhamento da codificação em tempo real, como também na ampliação da visibilidade dos artefatos manipulados. Além disso, o ambiente tem condições de apoiar a diminuição de problemas recorrentes em equipes que desenvolvem software de forma colaborativa. O experimento realizado no TECPAR, na forma de um único projeto, não foi suficiente para inferir quanto a integração dos artefatos impactou o projeto com a utilização do WS, pois apesar desta integração existir, ela não pode ser medida no experimento para fornecer subsídios para comprovação da hipótese. Em relação a gestão dos artefatos de projetos de software desenvolvidos por pequenas equipes, esta ficou demonstrada. Assim, considera-se que a primeira hipótese foi parcialmente comprovada como verdadeira. Experimentos de longo prazo, em

projetos que envolvam duas ou mais equipes de desenvolvimento devem ser desenvolvidos para gerar mais dados que possam auxiliar a comprovar esta hipótese de forma integral.

A segunda hipótese, com os dados levantados no experimento é verdadeira, ou seja, o acompanhamento da escrita do código-fonte em tempo real sendo confrontada com a modelagem, amplia a visibilidade do que está sendo criado pela equipe de desenvolvimento. Dois conceitos, o *awareness* e a verificação de software foram utilizados no módulo para acompanhamento da codificação, e serviram como fundamento para auxiliar a comprovar esta hipótese.

6.1. Contribuições e Relevância da Pesquisa

A principal contribuição deste trabalho está traduzida no objetivo essencial da pesquisa, o ambiente semântico que integra os artefatos e desenvolvedores no ambiente de desenvolvimento de software. Este ambiente é fruto da observação de ferramentas para o desenvolvimento de software, dos DSs e suas funcionalidades e da necessidade existente da criação e aprimoramento de ferramentas que sejam adequadas e úteis para o desenvolvimento de software, principalmente para as pequenas equipes que buscam melhorar a qualidade do que produzem. O protótipo desenvolvido contribui para o aprimoramento da visibilidade dos artefatos manipulados no desenvolvimento de software, integrando estes artefatos às pessoas que diretamente os manipulam, os desenvolvedores do software.

A condução dos experimentos realizados permitiu verificar a eficácia do ambiente, principalmente do módulo desenvolvido para realizar a monitoração e o acompanhamento da criação do código-fonte. Pode-se verificar que o *awareness* e a verificação de software são capazes de serem ampliados por meio da visibilidade dos artefatos monitorados. Apesar de não fazer parte da avaliação dos experimentos, outros problemas inerentes à atividade de desenvolver software podem ser reduzidos com a utilização do protótipo desenvolvido, tais como diminuir a disparidade entre a especificação realizada na modelagem e a efetiva implantação do software, manter e rastrear informações do projeto de software e mesmo evitar conflitos de codificação que podem ocorrer no trabalho em grupo quando se

desenvolve software. Esta possível redução da disparidade, também auxilia a realizar a integração dos artefatos e desenvolvedores. Em um ambiente em que o *awareness* é um elemento significativo por influenciar positivamente o processo de criar software, esta integração possui importância para melhorar a qualidade do software sendo desenvolvido.

A pesquisa aqui desenvolvida resultou em contribuições científicas que se traduzem em artigos publicados em conferências na área de computação, sendo elas (SILVA *et al.,* 2013) e (WANDERLEY *et al.,* 2012).

Outras contribuições podem ser agregadas as já elencadas, pois as potencialidades que o WS pode alcançar com seu desenvolvimento são amplas, e a relevância do tema é justificada pela inserção desta pesquisa numa área que necessita de meios para produzir software mais facilmente e com qualidade. Além disto, o levantamento realizado no estado da arte, o questionário realizado com os profissionais da área e os experimentos conduzidos, apontam que inexiste conceito ou até mesmo uma ferramenta semelhante a que foi apresentada nesta pesquisa, fato que auxilia a demonstrar a relevância do trabalho realizado.

6.2. Limitações de Escopo da Pesquisa

Três pontos podem ser enumerados como limitadores de escopo nesta pesquisa. O primeiro está relacionado com o tamanho da equipe, que foi definido no escopo deste trabalho com no máximo 10 participantes e de forma colocada. Um número maior de integrantes em uma equipe de desenvolvimento distribuída geograficamente pode levar a um número maior de pessoas com a função de desenvolvedor, e consequentemente, a um número maior de atribuições a serem realizadas por estes profissionais. Isto pode causar um crescimento dos itens que necessitam ser monitorados e apresentados em tempo real de codificação, e consequentemente, a um número maior de interações que estas pessoas necessitarão realizar para conduzir seu trabalho.

O segundo ponto limitante diz respeito às tecnologias monitoradas no módulo OPERAM. Neste trabalho, as equipes obrigatoriamente deveriam utilizar como linguagem de programação a linguagem JAVA, e como ferramenta de análise e

modelagem da arquitetura do software ASTAH COMMUNITY. O framework HACKYSTAT, utilizado intensamente nesta pesquisa, possibilita que sensores possam ser desenvolvidos para outras linguagens de programação, tais como o C e o Visual Studio, o que amplia o cenário de utilização e aumenta as perspectivas de expansão para outros campos, além do que aqui foi avaliado. As duas primeiras obrigatoriedades foram necessárias a fim de definir o escopo da pesquisa realizada neste trabalho. Apesar de serem softwares de ampla utilização, tanto no meio acadêmico como no profissional, estas obrigatoriedades restringiram a variedade de opções que poderiam ser também avaliadas entre as disponíveis, e assim trabalhos futuros que envolvem outras linguagens, IDEs de trabalho, ferramentas de modelagem podem ser conduzidos para atuar em outras esferas do domínio do desenvolvimento de software.

A terceira limitação é quanto ao experimento de avaliação conduzido, com uma equipe de quatro componentes, onde três atuaram como desenvolvedores. Um experimento com duas equipes com maior número de componentes, onde uma equipe utilizasse o OPERAM e outra não, em um projeto com maiores proporções e por um tempo maior de desenvolvimento, para efeitos de comparação, poderia ser conduzido para obter mais dados a respeito do processo da escrita do código fonte e confronto deste com a modelagem realizada. Com estas duas equipes, um número maior de pessoas com o papel de desenvolvedor, em um projeto de maior abrangência terão condições de trazer mais informações a respeito da utilização do OPERAM, e como consequência, um maior número de contribuições nas opiniões, sugestões e críticas que este tipo de experimento pode agregar.

6.3. Trabalhos Futuros

Alguns trabalhos de pesquisa são indicados como possíveis de serem realizados para aprimorar o WS. Primeiramente, pode ser interessante gerar um estudo de longo termo e um número maior de participantes com papéis diferenciados. Isto elevaria o número de interações e de testes a serem executados, e certamente forneceria mais subsídios para avaliar o ambiente proposto.

Outro trabalho futuro pode ser conduzido para gerar dados para o *Personal Software Process* (PSP), que, por meio de um completo conjunto de conhecimento voltado ao aperfeiçoamento das tarefas, visa a realizar no nível pessoal aquilo que se propõe o CMMI (*Capability Maturity Model Integration*) no nível organizacional. Como o WS já coleta uma grande parte das informações que são relevantes e utilizadas no PSP, um módulo que tratasse do assunto traria proveito nesta área do conhecimento.

Outros profissionais que atuam na produção colaborativa de software poderiam ser vinculados ao WS em futuras versões. Estes profissionais podem incluir gerentes, arquitetos, testadores, entre outros. Para isso, será necessário construir sensores de coleta de dados para acoplar às ferramentas utilizadas por esses participantes, e desta forma poder também acompanhar todo o seu trabalho.

Outro ponto interessante seria a avaliação do WS em grandes equipes de desenvolvimento que trabalham distribuídas. Os experimentos realizados para avaliação foram conduzidos em pequenas equipes que trabalhavam muito próximas entre si, inclusive no mesmo ambiente. A inserção de grandes equipes e distribuídas geograficamente são itens que podem ser considerados em um trabalho futuro a ser desenvolvido.

Um ponto de melhoria para o WS que pode ser tratado em um trabalho futuro é a criação de um *dashboard* que sintetize as informações em detrimento ao modelo apresentado nos experimentos para visualização e acompanhamento das informações oriundas do desenvolvimento de software. Um *dashboard* apresenta informações importantes em uma única tela, sintetizando na forma de números, textos, gráficos e outros elementos visuais de maneira simplificada, fornecendo rápida comunicação e monitoramento de diversas informações para o auxílio à tomada de decisão. Como a finalidade do *dashboard* é coletar, resumir e apresentar informações de várias fontes para que o usuário possa avaliar informações relevantes ao acompanhamento e monitoramento de seus interesses, é uma opção a ser considerada em um trabalho futuro.

Outro ponto a ser considerado em um trabalho futuro é a integração do WS em um sistema SVN (*Subversions*), de forma a interligar as informações produzidas por estes dois sistemas de acompanhamento. Desta forma, a junção do

acompanhando da codificação em tempo real do WS com gerenciamento das versões que os SVNs realizam, é uma opção que pode trazer benefícios para as equipes de desenvolvimento e também da documentação dos sistemas.

Por fim, a substituição do NEPOMUK para gerenciar os dados gerenciais dos usuários é uma pesquisa que pode gerar interesse em ser conduzida em um trabalho futuro. Assim, o WS poderia ter sua fonte nativa de gerenciamento dos artefatos que poderia ser obtida, por exemplo, por meio de ontologias e formas próprias de inferência das informações criadas especificamente para esta finalidade.

Referências Bibliográficas

(ADAR *et al.,* 1999) ADAR, E.; KARGER, D.; STEIN, L. A. *Haystack: Per-user information environments*. Proceedings of the eighth international conference on Information and knowledge management – CKIKM'99.pp .413–422.ACM Press. http://portal.acm.org/citation.cfm?id=319950.323231. 1999.

(AL-BADAREEN *et al.,* 2011) AL-BADAREEN A. B.; SELAMAT, M. H.; JABAR M. A.; DIN J.; TURAEV S.: *The Impact of Software Quality on Maintenance Process*. ACM International Journal Of Computers. Issue 2, Volume 5. 2011.

(ALONSO *et al.,* 2003) ALONSO, C. M. M. C.; RIZZI, C. B.; SEIXAS, L. M. J.: *Software EquiText - Uma Ferramenta para a escrita Colaborativa na Web*. In: VIII Taller Internacional de Software Educativo TISE 2003. Universidad de Chile - Facultad de Ciencias Físicas e Matematicas. 2003.

(ANDERSON, 2010) ANDERSON D. J.: *Kanban: Successful Evolutionary Change for Technology Organizations*. Blue Hole Press. ISBN 0-9845214-0-2. 208 pg. 2010

(BARREAU, 1995) BARREAU, D.: *Context as a Factor In Personal Information Management systems.* Journal of the American Society for Information Science, 46(5). pp. 327–339. 1995.

(BELLOTTI *et al.,* 2002) BELLOTTI, N.; DUCHENEAUT, M.; HOWARD, I.; SMITH, A.; NEUWIRTH C.: Innovation in extremis: evolving an application for the critical work of email and information management. Proc. of the Conference on Designing Interactive Systems, pp. 181–192. ACM Press. 2002.

(BELLOTTI *et al.,* 2004) BELLOTTI, V.; DALAL, B.; GOOD, N; FLYNN, P; BOBROW, D. G.; DUCHENEAUT N.: *What A To-Do: Studies Of Task Management Towards The Design Of A Personal Task List Manager*. CHI'04: Proc. of the SIGCHI Conference on Human Factors in Computing Systems, pp 735-742. 2004.

(BELKADI *et al.,* 2012) BELKADI, F.; BONJOUR, E.; CAMARGO, M.; TROUSSIER, N.; EYNARD, B.*A Situation Model To Support Awareness In Collaborative Design.* International Journal of Human-Computer Studies. 2012.

(BERNADI *et al.,* 2008) BERNARDI, A.; DECKER, S.; ELST, L. VAN; GRIMNES, G. A.; GROZA, T.; HANDSCHUH, S.; JAZAYERI, M.; MESNAGE, C.; MOLLER, K.; REIF, G. SINTEK, M.: *The Social Semantic Desktop: A New Paradigm Towards Deploying the Semantic Web on the Desktop*. Semantic Web Engineering in the Knowledge Society IGI Global ISBN 978-1-60566-112-4. 2008.

(BERNARDI *et al.,* 2011) BERNARDI, A.; GRIMNES, G; GROZA, T.; SCERRI, S.: *The NEPOMUK Semantic Desktop*. In Paul Warren, John Davies and Elena

Simperl (Ed.), Context and semantics for knowledge management: Technologies for personal productivity. pp. 255-273. Heidelberg, Germany: Springer ISBN: 9783642195099; 9783642195105. 2011

(BERNES-LEE *et al.,* 2001) BERNERS-LEE, T.; HENDLER, J.; LASSILA, O.: *The Semantic Web.* Scientific American. Acesso em 17-10-2011. Disp. em: http://www.scientificamerican.com/article.cfm?id=the-semantic-web. 2001.

(BLAUTH, 2005) BLAUTH M, P.: *Linguagens formais e autômatos*. ISBN : 85-241-0554-2. 215 p. 5º. ed. Porto Alegre : Sagra Luzzatto. 2005.

(BOEHM, 1979) BOEHM B. W.: *Software engineering; R & D Trends and defense needs.* Research Directions in Software Technology. Wegner, P. (ed.). Cambridge, Mass.: MIT Press. 1–9. 1979

(BRAUN *et al.,* 2007) BRAUN, S.; SCHMIDT, A.; HENSTCHEL, C.: *Semantic Desktop Systems for Context Awareness – Requirements and Architectural Implications*. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.80=pdf>. SemDesk07 - 2007.

(BRAUN *et al.,* 2007a) BRAUN, S.; SCHMIDT, A.; HEFKE, M*. A Socially-Aware Desktop for e-Science: Supporting Learning in Networked Scientific Processes*. Conference on Professional Knowledge Management, pp.47–54. 2007

(BREITMAN *et al.,* 2007) BREITMAN, K.; CASANOVA, M. A.; TRUSZKOWSKI, W.: *Semantic Web: Concepts, Technologies and Applications.* Springer. ISBN 1-84628-581-X. Springer-Verlag Heidelberg pp 313. 2007.

(BRERETON *et al.,* 2007) BRERETON, P.; KITCHENHAM, B. A.; BUDGEN, D. TURNER, M.; KHALIL, M.: *Lessons From Applying The Systematic Literature Review Process Within The Software Engineering Domain*. pp 571-583. Journal of Systems and Software – Elsevier. 2007.

(BULEJ, L. *et al.,* 2012) BULEJ, L.; BURES, T.; HORKY, V.; KEZNIKL, J.; Tuma, P.: *Performance Awareness In Component Systems: Vision Paper.* IEEE 36th Annual Computer Software and Applications Conference Workshops, p. 514–519. 2012.

(BURDEN *et al.,* 2011) BURDEN, H.; HELDAL, R.; SILJAMAKI, T. *Executable and Translatable UML--How Difficult Can it Be?*APSEC Conference, ISBN978-1-4577-2199-1 pp. 114–121,. 2011.

(BUSH, 1945) BUSH, V.: *As We May Think*. The Atlantic Monthly, 176(1):101–108, July 1945.

(CAI e YU, 2014) CAI G.; Yu B.: *Event-based Awareness Promotion For Distributed Collaborative Activities.* Conference on Collaboration Technologies and Systems. pp 302 – 309. 2014

(CAMPAGNOLO *et al.,* 2009) CAMPAGNOLO, B.; TACLA, C. A.; PARAISO, E. C.; SATO G.; RAMOS, M. P.: *An Architecture For Supporting Small Collocated Teams In Cooperative Software Development*. In: IEEE Computer Supported Cooperative Work in Design, 2009, Santiago. Proceedings of the 13th IEEE Computer Supported Cooperative Work in Design,. p. 264-269. 2009.

(CANFORA *et al.,* 2003) CANFORA, G.; LANUBILE F.; MALLARDO, T.: *Can Collaborative Software Development Benefit from Synchronous Groupware*

Functions? Proc. of the 2nd Workshop on Cooperative Supports for Distributed Software Engineering Processes. pp. 44-55. ISBN 88-464-4774-3. 2003.

(CHARFI *et al.*, 2012) CHARFI A.; MRAIDHA C.; BOULET P.: *An Optimized Compilation of UML State Machines.* Proceedings of the IEEE 15th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing. ISBN: 978-0-7695-4643-8. pp. 172-179. 2012.

(CHEYER *et al*., 2005) CHEYER, A.; PARK, J.; GIULI, R.: *Iris: Integrate, Relate. Infer. Share*. 1st Workshop on The Semantic Desktop. 4th International Semantic Web Conference. 2005.

(CHOCKLER *et al.,* 2013) CHOCKLER H.; DENARO G.; LING M.; FEDYUKOVICH G.; HYVRINEN A. E.J.; MARIANI L.; MUHAMMAD A.; ORIOL M.; RAJAN A.; SERY O.; SHARYGINA N.; TAUTSCHNIG M*.: Validating Changes and Upgrades in Networked Software.* 17th European Conference on Software Maintenance and Reengineering ISBN: 978-1-4673-5833-0 2013.

(CLAYBERG e RUBEL, 2004) CLAYBERG E.; RUBEL D.: *Eclipse: Building Commercial-Quality Plug-Ins*. Eclipse Series.: Addison-Wesley Professional. p. 800. ISBN 978-0321553461. 2004.

(CONTANDRIOPOULOS *et al.,*1999) CONTANDRIOPOULOS A. P., CHAMPAGNE F., POTVIN, L., DENIS JL., BOYLE P.: *Saber preparar uma pesquisa*: *Definição, Estrutura, Financiamento*. 3ª ed. São Paulo: Hucitec - Abrasco ISBN 85-271-0265-X; 215 p. 1999.

(COOK e CHURCHER, 2005) COOK C.; CHURCHER N.: *Modeling and Measuring Collaborative Software Engineering*, Proceedings of ACSC2005: Twenty-Eighth Australasian Computer Science Conference, volume 38 of Conferences in Research and Practice in Information Technology. 2005.

(CRAIK, 1943) CRAIK K.: *The Nature of Explanation*. Cambridge University Press, 134 pages. ISBN 978-0521047555. 1943.

(CRUZ *et al.,* 2005) CRUZ, I.F.; XIAO, H.; HSU, F.: *Peer-to-peer Semantic Integration Of Xml And Rdf Data Sources*. Third Internacional Workshop on Agents and Peer-to-Peer Computing, vol 3601, Lecture Notes in Computer Science, pp. 108–119. Springer, Heidelberg. 2005.

(CUBRANIC *e* MURPHY*,* 2003) CUBRANIC, D,; MURPHY G. C.: *Hipikat: Recommending Pertinent Software Development Artifacts*. ICSE, pp.408~418. 2003.

(CUBRANIC *et al.,* 2005) CUBRANIC, D.; MURPHY, G. C.; SINGER, J.; KELLOGG, S. B.: *Hipikat: A Project Memory For Software Development*. Transactions on Software Engineering.vpp 446–65. 2005.

(DEBATTISTA e ABELA, 2011) DEBATTISTA, J.; ABELA, C.: *Bizzilla: A Collaborative Task Management Environment with Expert Finding.* University of Malta <http://staff.um.edu.mt/cabe2/super/undergraduate/overview/bizzilla.pdf>. 2011.

(DECKER e FRANK, 2004). DECKER, S., FRANK M R.: *The Social Semantic Desktop*.WWW 2004 Workshop Application Design, Development and Implementation Issues in the Semantic Web. DERI Technical Report. 2004.

(DECKER e FRANK, 2004a). DECKER, S.; FRANK, M. R.: *The Networked Semantic Desktop*. Workshop on Application Design, Development and Implementation Issues in the Semantic Web. v. 105, P. 1613–0073. 2004.

(DEMARTINI e NIEDERÉE, 2008) DEMARTINI, G.; NIEDERÉE, C.: *Finding Experts On The Semantic Desktop.* The 7th International Semantic Web Conference. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.143.1736&rep=rep1&type=pdf#page=23>. 2008.

(DILLENBOURG *et al.,* 1995) DILLENBOURG, P.; BAKER, M.; BLAYE, A.; O'MALLEY, C.: *The Evolution of Research on Collaborative Learning* Learning in humans and machines. Towards an interdisciplinary learning science, 189-211. Ed. Pergamon. 1995.

(DOURISH e BELLOTTI, 1992) DOURISH, P.; BELLOTTI, V.: *Awareness and Coordination in Shared Workspaces*. CSCW '92: Proc. of the Conf. on Computer supported cooperative work, pages 107–114, New York. ACM. 1992.

(DUQUE *et al.,* 2012) DUQUE, R.; RODRÍGUEZ, M. L.; HURTADO, M. V.; BRAVO, C.; RODRÍGUEZ-DOMÍNGUEZ, C.: *Integration of Collaboration And Interaction Analysis Mechanisms In A Concern-Based Architecture For Groupware Systems*. Science of Computer Programming, v. 77, pp. 29–45. Editora Elsevier. 2012.

(ELLIS *et al.,* 1991) ELLIS, C.A.; GIBBS, S.J.; REIN, G.L: *Groupware - Some Issues and Experiences.* Communications of the ACM, Vol. 34, No. 1, pp. 39-58. 1991.

(ENGELBART, 1963) ENGELBART, D.C.: *A Conceptual Framework for the Augmentation of Man's Intellect*. Spartan Books, pp. 1–29. Republicado pela editora Greif - *Computer Supported Cooperative Work: A Book of Readings,* Morgan Kaufmann Publishers, Inc., 1988, pp. 35–65. Relatório técnico original disponível no endereço eletrônico http://www.dougengelbart.org/pubs/augment-3906.html. 1963.

(ETEZADI, 2008) ETEZADI, A. R: *Semantic Desktop - Focusing On Harvesting Domain Specific Information In Planning Aid Documents*. Final thesis. Linköping Institute of Technology http://www.essays.se/essay/517efa88db/. 2008.

(FAHEEN, 2010) FAHEEM, M.: *Implementation of Semantic Desktop Tool*. Free University of Bozen. Master Thesis. Disponível: http://www.emclstudy.eu/fileadmin/master_theses-/thesis_faheem.pdf. 2010.

(FALBO *et al.,* 1998) FALBO R.A.; MENEZES C.S.; ROCHA A.R.: *Integração de Conhecimento sobre Processos de Software em um Ambiente de Desenvolvimento*. Anais da IX Conferência Internacional de Tecnologia de Software , IX CITS, Curitiba-Paraná, Brasil. 1998.

(FALBO *et al.,* 2002) FALBO, R.A.; GUIZZARDI, G.; NATALI, A. C. C.; BERTOLLO, G., RUY F. B.; MIAN, P. G.: *Towards Semantic Software Engineering Environments*. Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering.SEKE'2002, pp. 477 – 478. 2002.

(FALBO *et al.,* 2004) FALBO, R.A.; RUY, F.B.; PEZZIN, J.; DAL MORO R.: *Ontologias e Ambientes de Desenvolvimento de Software Semânticos*. Actas de

las IV Jornadas Iberoamericanas de Ingeniería Del Software e Ingeniería del Conocimiento, JIISIC'2004, Volumen I, pp. 277-292. 2004.

(FALBO *et al.,* 2005) FALBO, R.A.; RUY, F.B.; MORO, R.D.: *Using Ontologies to Add Semantics to a Software Engineering Environment.*17th International Conference on Software Engineering and Knowledge Engineering, SEKE'2005, p. 151 – 156. July 2005.

(GEROSA, 2006) GEROSA, M. A.: *Desenvolvimento de Groupware Componetizado com Base no Modelo 3C de Colaboração*. Tese de Doutorado, Departamento de Informática da PUC-Rio. 2006.

(GIL, 2002) GIL, A. C. *Como elaborar projetos de pesquisa*. ISBN85-224-3169-8. 4. ed. São Paulo:Atlas,. 175 p. 2002.

(GONZALES *et al.,* 2012) GONZALES, C. A.; FABIAN, B.; CLARISÓ, R.; CABOT,: *EMFtoCSP : A Tool for the Lightweight Verification of EMF Models*. Software Engineering: Rigorous and Agile Approaches. ISBN 978-1-4673-1907-2 p. 44–50. 2012.

(GROZA *et al.,* 2007) GROZA, T.; HANDSCHUH, S.; MOELLER, K.; GRIMNES, G.; SAUERMANN, L.; MINACK, E.; MESNAGE, C.; JAZAYERI, M.; REIF, G.; GUDJONSDOTTIR, R.: *The Nepomuk Project-On The Way To The Social Semantic Desktop.* Proceedings of I-Semantics, pp. 201–211. 2007.

(GRUDIN, 1994) GRUDIN, J.: *Computer-Supported Cooperative Work: History and Focus*". Computer, 27(5): p. 19-26. 1994.

(GUJÓNSDÓTTIR e LINDQUIST, 2008) GUJÓNSDÓTTIR; R.: LINDQUIST; *Personas and scenarios: Design Tool Or A Communication Device?* Proceedings of the 8th International Conference on the Design of Cooperative Systems, pp. 165-176. 2008.

(GUTWIN e GREENBER, 1996) GUTWIN, C.; GREENBER S.: *Workspace Awareness for Groupware*. Proc. of the CHI'96 Conference Companion on Human Factors in Computing Systems. pp. 208-209. 1996.

(HAJIZADEH *et al.,* 2014) HAJIZADEH A. H.; TORY, M.; LEUNG R.: *Supporting Awareness through Collaborative Brushing and Linking of Tabular Data*. Transactions on Visualization and Computer Graphics. pp 2189 – 2197. 2014.

(HALLER, 2008) HALLER, H. *QuiKey – a Demo.*Semantic Search, p. 74. ISSN 1613-0073. SemSearch Workshop. Disponível em: online at CEUR-WS.org/Vol-334/. 2008.

(HILDENBRAND *et al., 2008*) HILDENBRAND, T.; ROTHLAUF, F.; GEISSER, M.; HEINZL, A.; KUDE, T.: *Approaches to Collaborative Software Development.* Conf. on Complex, Intelligent and Software Intensive Systems, pp. 523–528. 2008.

(HOCHSTEIN *et al*, 2005) HOCHSTEIN, L.; BASILI, V. R.; ZELKOWITZ, M. V.; HOLLINGSWORTH J. K.; CARVER, J.: *Combining Self-reported and Automatic Data to Improve Programming Effort Measurement*, Foundations of Software Engineering, 2005.

(HU, 2014) HU, Jhen-Jia: *The Verification and Validation of a Large-Scale System: Equipment TaaS as an Example.* International Symposium on Computer, Consumer and Control. 2014. 978-1-4799-5277-9/14. p. 13-18. 2014.

(IEEE, 2012) The Institute of Electrical and Electronics Engineers, Inc. *IEEE Standard for System and Software Verification and Validation*. ISBN 978-0-7381-7268-2. Disponível em http://standards.ieee.org/findstds/standard/1012-2012. Último acesso em março/2014.

(IKONEN *et al.,* 2011) IKONEN, M.; PIRINEN, E.; FAGERHOLM, F.; KETTUNEN, P.; ABRAHAMSSON, P.: *On the Impact of Kanban on Software Project Work: An Empirical Case Study Investigation*. 16th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS), pp: 305-314. ISBN: 978-1-61284-853-2. 2011.

(ITURRIOZ *et al.,* 2006) ITURRIOZ, J; ANZUOLA, S.F.; DÍAZ, O.: *Turning the Mouse into a Semantic Device: The seMouse Experience*. Proceedings of 3rd European Semantic Web Conf. (ESWC 06 LNCS 4011). 2006.

(JIANG *et al.,* 2006) JIANG, T.; YING, J.; WU, M.; FANG, M.: *An Architecture of Process-centered Context-aware Software Development Environment*. In: 10th Computer Supported Cooperative Work in Design. CSCW2006, Alberta, Canada. 2006.

(JOHNSON e ZHANG, 2009) JOHNSON, P.; ZHANG, S.: *We Need More Coverage, Stat! Classroom Experience with the Software ICU.*2009 3rd International Symposium on Empirical Software Engineering and Measurement, n. 1, pp. 168–178, 2009. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?ar=5315989>. 2009.

(JOHNSON *et al.*, 2003) JOHNSON P. M.; KOU, H.; AGUSTIN J.; CHAN, C.; MOORE C.; MIGLANI, J.; ZHEN, S.; DOANE, W. E. J.: *Beyond the Personal Software Process: Metrics Collection And Analysis For The Differently Disciplined.* In Proceedings of the 25th International Conference on Software Engineering (ICSE '03). IEEE Computer Society, Washington, DC, USA, pp. 641-646. 2003.

(JOHNSON *et al.*, 2004) JOHNSON, P. M.; KOU, H.; AGUSTIN, J. M.; ZHANG, Q.; KAGAWA, A.; YAMASHITA, T.: *Practical Automated Process and Product Metric Collection and Analysis in a Classroom Setting: Lessons Learned from Hackystat-UH*. International Symposium on Empirical Software Engineering (ISESE'04). ISBN: 0-7695-2165-7 2004.

(JOHNSON *et al.,* 2009) JOHNSON, P.; ZHANG, S.; SENIN, P.: *Experiences With Hackystat As A Service-Oriented Architecture.* University of Hawaii, Honolulu,. Disponível em: <http://www.imamu.edu.sa/dcontent/IT_Topics/java/09-07.pdf>. 2009.

(JOHNSON, 2007) JOHNSON, P. M.: *Requirement and Design Trade-offs in Hackystat: An In-Process Software Engineering Measurement and Analysis System*. International Symposium on Empirical Software Engineering and Measurement. pp. 81–90, 2007.

(JOO, 2011) JOO, J. *Adoption of Semantic Web from the perspective of technology innovation: A grounded theory approach*. International Journal of Human-Computer Studies. Volume 69 Issue 3. pp 139-154. 2011.

(KARGER *et al.,* 2005) KARGER, D. R.; BAKSHI, K.; HUYNH, D.; QUAN, D.; SINHA, V.: *Haystack: A Customizable General-Purpose Information Management Tool For End Users Of Semistructured Data*. CIDR Conference. 2005.

(KITCHENHAM e CHARTRES, 2007) KITCHENHAM, B.; CHARTERS, S: *Guidelines for performing Systematic Literature Reviews in Software Engineering*. Version 2.3 - EBSE-2007-01 http://www.dur.ac.uk/ebse/resources/guidelines/Systematic-reviews-5-8.pdf - Durham University. 2007.

(KITCHENHAM *et al.,* 2002) KITCHENHAM B.; PFLEEGER S.M.; PICKARD L.M.; JONES P.W,.; HOAGLIN D.C.; EL EMAN K.; ROSENBERG J.: *Preliminary Guidelines For Empirical Research In Software Engineering*. IEEE Trans Soft. Eng 28(8):721– 734. 2002.

(LANSDALE, 1988) LANSDALE, M.: *The Psychology Of Personal Information Management.* Applied Ergonomics, 19 (1) pp. 55–66, 1988.

(LEIFLER e ERIKSSON*,* 2008) LEIFLER O.; ERIKSSON H.: *A Model for Document Processing in Semantic Desktop Systems.* Proc. I-KNOW08 and I-MEDIA08. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.162.1424&rep=rep1&type=pdf. 2008.

(LI e BOEHM, 2011) LI, Q.; BOEHM, B. W. *Making winners for both education and research: Verification and validation process improvement practice in a software engineering course.*24th IEEE-CS Conference on Software Engineering Education and Training (CSEE&T), p. 304–313. 2011.

(LIMA *et al.,* 2010), LIMA, E. J. C. DE; NETO, J. A. R.; XEXÉO, G. B.; SOUZA, J. M. DE.: *ARARA - A Collaborative Tool To Requirement Change Awarenes.*14th International Conference on Computer Supported Cooperative Work in Design, pp. 134–139. <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5471987>. 2010.

(LINXIA, 2010) LINXIA Y.: *An Application of a Task-Based CSCW System in Enterprise Informatization.* Management and Service Science (MASS), 2010 Inst. of Comput. Sci. e Technol., Taiyuan Univ. of Sci. e Technol. ISBN: 978-1-4244-5325-2. 2010.

(LISANSKIY, 1999) LISANSKIY, I.: *A Data Model For The Haystack Document Management System*.http://citeseerx.ist.psu.edu/viewdoc/?doi=10.1.1.17type. MIT Dept. of Electrical Engineering and Computer Science. 1999.

(LUCENA *et al.,* 1998) LUCENA, C. J. P.; FUKS, H.; MILIDIU, R.; MACEDO, L.; SANTOS, N.; LAUFER, C.; RIBEIRO, M. B.; FONTOURA, M. F.; NOYA, R. C.; CRESPO, S.; TORRES, V.; DAFLON, L.; LUKOWIECKI, L.: *AulaNet - An Environment For The Development And Maintenance Of Courses On The Web.* Proceedings of the International Conference on Engineering in Education. Rio de Janeiro, RJ. 1998.

(MINACK *et al.,* 2010) MINACK, E.; PAIU, R.; COSTACHE, S.: *Leveraging Personal Metadata For Desktop Search: The Beagle++ System*. Web Semantics: Science, Services and Agents on the World Wide Web, v. 8, n. 1, p. 37-54, 2010. Disp. em: <http://linkinghub.elsevier.com/retrieve/pii/S1570826809000754>. 2010.

(MORISIO *et al*, 2004) MORISIO M.; TORCHIANO,M.; ARGENTIERI G.: *Assessing Quantitatively a Programming Course*. 10th International Symposium on Software Metrics, pp. 326-336. 2004.

(MYSQL, 2012) MySQL Website: http://www.mysql.com/. Acesso em 22/05/2012.

(NADEEM, 2007) NADEEM, D. *Cognitive Aspects of Semantic Desktop to Support Personal Information Management.* Thesis Institute of Cognitive Science - Univ. of Osnabrueck. http://citeseerx.ist.psu.edu/viewdoc/169.1648&amdf. 2007.

(NESIC, 2010) NEŠIC, S.: *Semantic Document Architecture for Desktop Data Integration and Management:* Doctoral Dissertation. Università della Svizzera Italiana. 2010.

(NOVAIS *et al.,* 2011)NOVAIS, R. L.; LIMA, C.A.N.; CARNEIRO, G F.; PAULO, R.M.S. ; MENDONCA, M.: *An Interactive Differential and Temporal Approach to Visually Analyze Software Evolution*. 6th IEEE International Workshop on Visualizing Software for Understanding and Analysis.p 1-4. 2011.

(OLAMIDE e KABA, 2013) OLAMIDE, S.E. KABA, T.M.: *Formal Verification and Validation of DEVS Simulation Models*. IEEE AFRICON Conference, pp 1–6 ISBN: 978-1-4673-5940-5 2013.

(O.S.A.F, 2011) O.S.A.F.: *What's Compelling About Chandler: A Current Perspective*. Acesso em 12-12-2011. http://www.osafoundation.org/Chandler_Compelling_Vision.htm. 2011.

(OREN, 2006) OREN E.; *An Overview of Information Management and Knowledge Work Studies: Lessons for the Semantic Desktop.* ISWC - Workshop on the Semantic Desktop. Knowledge Creation Diffusion Utilization. 2006.

(PETRE, 2010) PETRE, M. *Mental Imagery And Software Visualization In High-Performance Software Development Teams*. Journal of Visual Languages & Computing, p. 171-183. 2010.

(POLLICE *et al.,* 2004) POLLICE, G.; AUGUSTINE, L.; LOWE C.; MADHUR, J.: *Software Development for Small Teams: A RUP-Centric Approach*. Addison-Wesley, pp. 304. ISBN:0321199502. 2004.

(QUAN *et al.,* 2003) QUAN, D; HUYNH, D.; KARGER, D. R.: *Haystack: A Platform for Authoring End User Semantic Web Applications*. International Semantic Web Conference. http://www.springerlink.com/index/H74TVQB63.pdf. 2003.

(QUAN *et al.*, 2011) QUAN, T. T.; HOANG, D. L. N.;NGUYEN, B. T.; NGUYEN, A. N.; TRAN, Q.D.; NGUYEN, P. H.; BUI, T. H.; DO, A. T.;HUYNH, L. V.;DOAN, N. T.; HUYNH, N. T.; NGUYEN, T. D.; NGUYEN, T. T.; NGUYEN, V.H.:*MAFSE: A Model-Based Framework for Software Verification*. Fourth International Conference on Secure Software Integration and Reliability Improvement Companion, p. 150–156 . 2010.

(RAMA e BISHOP, 2006) RAMA J.; BISHOP J.: *A Survey and Comparison of CSCW Groupware Applications.* Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on IT Research in Developing Countries. South African Institute for Computer Scientists and Information Technologists. p. 198-205. 2006

(RAMEZANI e BURNS, 2009) RAMEZANI, M.; BURNS, A. T.: *Mapping the Semantic Desktop to a Personal Knowledge Management framework*. School Of Computing Res.Symposium. http://josquin.cs.depaul.edu/~mramezani/papers/PKM2.pdf. 2009.

(REIF *et al.,* 2007) REIF, G.; GROZA, T.; HANDSCHUH S.; JAZAYERI M.; MESNAGE C.: *Intermediate Nepomuk Architecture.* Technical Report. Deliverable D6.2.A. Version 1.0. 2007.

(REIF *et al.,* 2007a) REIF, G.; GROZA, T.; SIEGFRIED, H.; MESNAGE C.; GUDJONSDOTTIR, R: *Collaboration on The Social Semantic Desktop*. Workshop on Ubiquitous Mobile Information and Collaboration Systems (UMICS 2007). <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.140.4754&rep=rep1&type=pdf>. 2007.

(SAMPAIO *et al.,* 2010) SAMPAIO, S. C. D. B.; BARROS, E. A.; AQUINO JUNIOR, G. S. D.; SILVA; M. J. C. E.; MEIRA, S. R. D. L.: *A Review of Productivity Factors and Strategies on Software Development.* Fifth International Conference on Software Engineering Advances, pp. 196–204, 2010. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5615739>. 2010.

(SARGENT, 2013) SARGENT, R G.: V*erification And Validation Of Simulation Models.* Journal of Simulation. pp 12–24 ISBN 978-1-4799-3950-3/13. 2013.

(SARMA *et al.,* 2003) SARMA, A.; NOROOZI, Z.; VAN DER HOEK, A.: *Palantír: Raising Awareness Among Configuration Management Workspaces*. In: 25th International Conference on Software Engineering (ICSE). 2003.

(SARMA*,* 2008) SARMA, A*. Palantír: Enhancing Configuration Management Systems with Workspace Awareness to Detect and Resolve Emerging Conflicts.* Dissertation, University Of California, Irvine. 204 Páginas. 2008.

(SARMIENTO e COLLAZOS*,* 2012) SARMIENTO, W. J.: COLLAZOS, C. A.: *CSCW Systems in Virtual Environments: A General Development Framework*. Conference on Creating, Connecting and Collaborating through Computing (C5), 10th. ISBN: 978-1-4673-1009-3. 2012

(SAUERMANN e HEIM, 2008) SAUERMANN L.; HEIM D.: *Evaluating Long-Term Use of the Gnowsis Semantic Desktop for PIM*. Proceeding of ISWC08 - 7th International Conference on The Semantic Web. Pages 467 – 468. Springer-Verlag Berlin, Heidelberg. : ISBN 978-3-540-88563-4. 2008.

(SAUERMANN e SCHWARZ, 2004) SAUERMANN, L.; SCHWARZ, S. *Introducing the Gnowsis Semantic Desktop*. Proceedings of the International Semantic Web Conference. pp 3-4. 2004.

(SAUERMANN *et al.,* 2005) SAUERMANN, L.; BERNARDI, A.; DENGEL, A.: *Overview And Outlook On The Semantic Desktop*. Proceedings of the 1st Workshop on The Semantic Desktop at the ISWC 2005 Conference. 175, pp. 1–19. http://citeseerx.ist.psu.edu/viewdoc/download?rep1&type=pdf. 2005.

(SAUERMANN *et al.,* 2006) SAUERMANN, L. AASTR, G.; KIESEL, M.: MAUS, H.; HEIM, D.; NADEEM, D.; HORAK, B.; DENGEL, A.: *Semantic desktop 2.0: The Gnowsis Experience.* International Semantic Web Conference.Vol.4273 of Lecture Notes in Computer Science. pp. 887—900.Springer. 2006.

(SAUERMANN *et al.,* 2007) SAUERMANN, L.; ELST, L. VAN; DENGEL, A*. Pimo - A Framework For Representing Personal Information Models.* I-Semantics, v. 7, 270–277. http://citeseerx.ist.psu.edu/ownload?doi=10.1.1.162.843&&type=pdf, 2007.

(SAUERMANN, 2003) SAUERMANN, L: *The Gnowsis: Using Semantic Web Technologies To Build A Semantic Desktop*. Diploma thesis, Technical University of Vienna. http://www.mendeley.com/research/the-gnowsis-using-semantic-web-technologies-to-build-a-semantic-desktop-1/. 2003.

(SAUERMANN, 2005) SAUERMANN, L. *The Gnowsis Semantic Desktop for Information Integration*. Proceedings of the IOA 2005 Workshop.p.39-42. http://scholar.google.com/scholar?cluster=1727570589043382828. 2005.

(SAUERMANN*,* 2005a) SAUERMANN, L. The *Semantic Desktop - A Basis For Personal Knowledge Management*. Proceedings of the I-KNOW. pp 294–301. 2005.

(SAUERMANN, 2009) SAUERMANN, L.: *The Gnowsis Semantic Desktop Approach to Personal Information Management - Weaving the Personal Semantic Web*. ISBN 9783866244498 Doctoral Thesis - Kaiserslautern University of Technology 2009.

(SCERRI *et al.,* 2009)SCERRI, S.; DAVIS, B.; HANDSCHUH, S.; HAUSWIRTH, M.: *Semanta – Semantic Email Made Easy*. The Semantic Web: Research and Applications, pp. 36–50. Springer. 2009.

(SCERRI, 2010) SCERRI, S.: *Supporting Email-based Collaborative Work across the Social Semantic Desktop Techniques*. Doctoral Thesis. National University of Ireland - Galway. http://resources.smile.deri.ie/semanta/THESIS.PDF. 2010.

(SCHANDL, 2009) SCHANDL, B.: *An Infrastructure for the Development of Semantic Desktop Applications*. Doctoral Dissertation. University of Vieen. http://eprints.cs.univie.ac.at/132/1/schandl.pdf. 2009.

(SENGUPTA *et al.,* 2007) SENGUPTA, S.; , SENGUPTA, A.; BHATTACHARYA S.: *Requirements to Components: A Model-View-Controller architecture*., Proceedings of 14th Monterey Workshop, pp 167--184, Monterey, CA, USA. 2007.

(SILVA *et al.,* 2013) SILVA E. J., TORQUATO E., RAMOS, M. P., PARAISO, E. C.: *OPERAM: A Collaborative Semantic Workspace for Software Verification*". IEEE International Conference on Systems, Man and Cybernetics, Manchester. pp. 1026-1031. 2013.

(SOMMERVILLE, 2011) SOMMERVILLE, I: *Software Engineering*. Harlow, England. Editora Addison-Wesley. 9ª Ed. 773 pg. ISBN 978-0-13-703515-1. 2011

(SOHLENKAMP, 1998) SOHLENKAMP, M.: *Supporting Group Awareness In Multi-User Enviroment Through Perceptualization.* Dissertation. Mathematik-Informatik der Universität - Gesamthochschule. ISBN 9783884573556. 1998.

(STOREY *et al.,* 2005) STOREY, M. A.; CUBRANIC, D.; GERMAN, D. M.: *On The Use Of Visualization To Support Awareness Of Human Activities In Software Development: A Survey And A Framework.* n. 212, pp. 193–203. <http://dl.acm.org/citation.cfm?id=1056045>. 2005.

(STOREY *et al.,* 2006) STOREY M. A.; CHENG, L.T.; BULL I.: RIGBY, P.: *Shared Waypoints and Social Tagging to Support Collaboration in Software Development*. Proceedings of the CSCW - 20th Conference on Computer Supported Cooperative Work: Canada, pp. 195-198. 2006.

(SWEBOK, 2014) P. BOURQUE and R.E. FAIRLEY, eds., *Guide to the Software Engineering Body of Knowledge, Version 3.0*, IEEE Computer Society, 2014. Disponível em http://www.computer.org/portal/web/swebok/swebokv3. Ultimo acesso em abril/2014.

(TALAEI-KHOEI *et al.,* 2012) TALAEI-KHOEI, A.; RAY, P.; PARAMESHWARAN, N.; LEWIS, L.: *A Framework For Awareness Maintenance.*Journal of Network and Computer Applications, v. 35, n. 1, p. 199–210,.Elsevier. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S1084804511001238>. 2012.

(TELLIOGLU e DIESENREITER, 2013) TELLIOGLU, H.; DIESENREITER, S.: *Enterprise 2.0 in action: Potentials for improvement of awareness support in enterprises*. International Conference Conference on Collaborative Computing: Networking, Applications and Worksharing (Collaboratecom). Pp 485-494. 2013

(TEEVAN *et al.,* 2006) TEEVAN, J.; JONES, W.; BEDERSON, B.B.: *Personal Information Management*. Communications of the ACM, Vol. 49, No. 1, pp. 40-43. 2006.

(TERUEL *et al.,* 2012) TERUEL, M. A.; NAVARRO, E.; LÓPEZ-JAQUERO, V.; MONTERO, F.; JAEN, J.; GONZÁLEZ, P.: *Analyzing the understandability of Requirements Engineering languages for CSCW systems: A family of experiments*. Information and Software Technology, v. 54, n. 11, p. 1215–1228, 2012. Elsevier B.V. 2012.

(TERUEL *et al.,* 2014) TERUEL, M. A.; NAVARRO, E.; LÓPEZ-JAQUERO, V.; MONTERO, GONZÁLEZ, P.: *A design pattern for representing Workspace Awareness*. Proceedings of the 2014 IEEE 18th International Conference on Computer Supported Cooperative Work in Design. pp 678 – 683. 2014.

(TREUDE e STOREY, 2011) TREUDE, C.; STOREY, M.: *Awareness 2.0: Staying Aware Of Projects, Developers And Tasks Using Dashboards And Feeds.* Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1.pp.365–374,. ACM. Disponível em: <http://portal.acm.org/citation.cfm?id=1806854>. 2011.

(TREUDE *et al.,* 2009) TREUDE, C.; STOREY, M.; WEBER, J.: *Empirical Studies on Collaboration in Software Development: A Systematic Literature Review*. Technical Report - Department of Computer Science. University of Victoria. 2009.

(WAINER e BARSOTTINI, 2007) WAINER J.; BARSOTTINI C.: *Empirical research in CSCW - a Review of the ACM/CSCW*. Journal of the Brazilian Computer Society,. pp. 27-36. 2007.

(WANDERLEY *et al.,* 2012) WANDERLEY, G. M. P.; RAMOS, M. P.; TACLA, C. A.; SATO, G.; SILVA, E. J.; PARAISO, E. C.: *MODUS-SD: User Modeling in Collaborative Software Development*. In: IEEE International Conference on Computer Supported Cooperative Work in Design. pp. 372-377. 2012.

(WANDERLEY *et al.,* 2012) WANDERLEY, G. M. P.; RAMOS, M. P.; TACLA, C. A.; SATO, G.; SILVA, E. J.; PARAISO, E. C.: *MODUS-SD: User Modeling in Collaborative Software Development*. In: IEEE International Conference on Computer Supported Cooperative Work in Design. pp. 372-377. 2012.

(WALLACE e FUJII, 1989) WALLACE, R. D.; FUJJI, R, U,: *Software verification and validation: an overview.* IEEE Software, p. 10-17. 1989

(WANG, 2011) WANG, Z*. Towards a Measurement Tool for Verification and Validation of Simulation Models*. Proceedings of the Winter Simulation Conference ISBN 978-1-4577-2108-3 p. 1233–1244. 2011.

(WANG, 2013) WANG, Z*. Selecting Verification And Validation Techniques For Simulation Projects: A Planning And Tailoring Strategy*. Proceedings of the Winter Simulation Conference ISBN 978-1-4799-3950-3/13 p. 581–592. 2011.

(WHITEHEAD, 2007) WHITEHEAD, J.: *Collaboration in Software Engineering - A Roadmap.* Proceeding's of FOSE '07 - Future of Software Engineering. Pp 214-225. IEEE Computer. ISBN:0-7695-2829-5. 2007.

(WOERNDL e WOEHRL, 2008) WOERNDL, W.; WOEHRL, M. *SeMoDesk*: *Towards a Mobile Semantic Desktop.* Proc. Personal Information Management (PIM) Workshop, CHI 2008. pp.1–6. 2008.

(WOHLIN *et al.,* 2000) WOHLIN C.; RUNESON P.; HÖST M; OHLSSON M. C.; REGNELL B.; WESSLÉN A.: *Experimentation in software engineering: an introduction.* Kluwer Academic Publishers Norwell. ISBN:0-7923-8682-5. 2000.

(XIAO e CRUZ, 2006) XIAO, H.; CRUZ I. F.: *Application Design and Interoperability for Managing Personal Information in the Semantic Desktop***.** 2nd International Workshop on the Semantic Desktop. 2006.

(YATAKE e KATAYAMA, 2008) YATAKE, K.; KATAYAMA, T.: *An Executable Semantics of Object-oriented Models for Simulation and Theorem Proving*. MSVVEIS,isbn978-989-8111-43-2. pp. 71-80. 2008.

(YATAKE e KATAYAMA, 2010) YATAKE, K.; KATAYAMA, T*. An Executable Object-Oriented Semantics and its Application to Firewall Verification*. Software & Systems Modeling, v. 10, n. 4, p. 515–536. Ed. Springerlink. 2010.

(YUYAN *et al.,* 2011) YUYAN J.; ZHUTING Z.; ZHENG W.; HAO S.: *Analysis and Design of an Online Collaborative Editing Systems on CSCW.* Multimedia

Technology (ICMT-2011) - School. of Manage. Sci. e Eng., Anhui Univ. of Technol., Maanshan, China. P. 490-493 ISBN: 978-1-61284-771-9. 2011.

(ZHANG e BABAR, 2011) ZHANG, H.; BABAR, M. A.: *An Empirical Investigation of Systematic Reviews in Software Engineering.* International Symposium on Empirical Software Engineering and Measurement, pp. 87–96, ISBN: 978-0-7695-4604-9- <http://ieeexplore.ieee.org/lpdocs/epi3/wrapper.htm?arnr=6092557>. 2011.

APÊNDICE 1 – Questionário Aplicado a Programadores e Gestores de Software em Empresas de Desenvolvimento.

O questionário mostrado no quadro abaixo foi disponibilizado para desenvolvedores/gerentes de projetos de pequenas equipes e também para alguns profissionais da área de informática. Ele foi publicado eletronicamente, e ficou disponível durante um determinado período para as respostas, no endereço `https://docs.google.com/spreadsheet/viewform?formkey=dFR5clMtOGxUM3NpeGFBSHROdGhzMVE6MQ#gid=0`.

No âmbito do trabalho de doutorado que está sendo desenvolvido no Programa de Pós-Graduação em Informática (PPGIa) da Pontifícia Universidade Católica do Paraná - PUCPR, agradeço a sua colaboração no preenchimento deste questionário online. O tema do doutorado é "WORKSPACE SEMÂNTICO: INTEGRANDO ARTEFATOS E DESENVOLVEDORES NO AMBIENTE DE DESENVOLVIMENTO DE SOFTWARE". As informações solicitadas servirão de base este projeto, que busca subsídios para aprimorar a colaboração das pessoas que participam de equipes (preferencialmente pequenas) de desenvolvimento de software e que trabalham na programação ou na gestão de projetos. Os resultados, após análise e tabulação serão disponibilizados aos interessados O questionário é anônimo e o tempo de preenchimento é inferior a 5 minutos, mas de grande valia para o andamento da pesquisa. Sua participação é muito importante e desde já agradeço pelas respostas aos questionamentos.

Edenilson José da Silva - Doutorando PPGIa
edenilson@ppgia.pucpr.br

01 - Como é realizada a gestão dos artefatos (documentos de texto, planilhas, código-fonte, mensagens e outros) que você produz durante o desenvolvimento do seu trabalho? *

- ○ Utilizo ferramentas específicas e independentes para cada tarefa.
- ○ Não realizo a gestão de documentos com nenhuma ferramenta.
- ○ Utilizo um software integrado para gestão destes artefatos.

02 - Se a resposta a pergunta anterior for "Utilizo um software integrado para gestão destes artefatos", por favor, indique o nome do software: _____

03 - Durante o desenvolvimento do seu trabalho, você desempenha qual papel dentro da equipe? *

- ○ Gerente de projeto
- ○ Desenvolvedor
- ○ Analista.
- ○ Testador
- ○ Outro: _____

04 - Este papel desempenhado, pode variar durante o projeto? *

- ○ Sim.
- ○ Não.

05 - A gestão do código-fonte produzido por seu trabalho é realizada por alguma ferramenta de apoio? *

- ○ Não faço a gestão de código com nenhuma ferramenta.
- ○ Utilizo Visual Source Safe (VSS).
- ○ Utilizo Subversion (SVN).
- ○ Utilizo Concurrent Version System (CVS).
- ○ Utilizo outra ferramenta.

06 - Se a resposta a pergunta anterior for "Utilizo outra ferramenta", por favor, indique o nome da ferramenta: _____

07- Utiliza algum destes processos ou metodologia de desenvolvimento de software? *

- ○ XP.
- ○ SCRUM.
- ○ Modelo Clássico de desenvolvimento.
- ○ RUP.
- ○ CMMi.
- ○ Outro: _____

08 - Como é realizada a documentação do código-fonte produzido durante o desenvolvimento do seu trabalho? *

- ☐ Não realizo a documentação do código-fonte.
- ☐ Através de comentários manuais no código-fonte.

- [ ] Através de comentários manuais no código-fonte e de documento criado para este fim.
- [ ] Através de comentários automatizados no código-fonte.
- [ ] Outro:

09 - Se a resposta a pergunta anterior for "Através de comentários automatizados no código-fonte", por favor, indique o nome da ferramenta:

10 - Qual o tamanho da equipe na qual você participa para desenvolver software? *

- [ ] De uma até cinco pessoas.
- [ ] De seis até dez pessoas.
- [ ] De onze a vinte pessoas.
- [ ] Mais de vinte pessoas

**11 - Classifique as ferramentas abaixo conforme o nível de utilização da mesma em relação ao andamento de seu trabalho para a troca de informações ou para resolução problemas ***1 – muito utilizada 2 – moderadamente utilizada 3 – pouco utilizada 4 – não utilizo

	1	2	3	4
Email.	○	○	○	○
Chat.	○	○	○	○
Comunicação face-a-face.	○	○	○	○
Outra forma que não as listadas acima.	○	○	○	○

12 - Qual o tempo médio para execução dos projetos de software? *

- [ ] De um a três meses.
- [ ] De quatro a doze meses.
- [ ] Mais de doze meses.

13 - Você ou alguém de sua equipe utiliza alguma ferramenta de controle de projetos? * Sim

14 - Se a resposta a pergunta anterior for "Sim", por favor, indique o nome da ferramenta:

15 - Para o desenvolvimento do seu trabalho na codificação, existe algum tipo de controle que mostra se o código-fonte produzido está refletindo os diagramas projetados para o sistema? *

16 - Se a resposta a pergunta anterior for "Sim", por favor, indique o nome da ferramenta:

17 - Qual a sua opinião sobre uma ferramenta que monitorasse o código-fonte, por exemplo: a criação de classes, alteração de métodos, etc. e que compartilhasse estas informações com os outros membros da equipe? *

- Extremamente útil.
- Interessante.
- Útil.
- Indiferente.
- Intrusiva

18 - Você utilizaria esta ferramenta descrita na questão17? *

- Sim.
- Não.

19 - Qual a sua opinião sobre uma ferramenta que mostrasse o progresso do código-fonte desenvolvido e confrontasse, por exemplo, com um diagrama de classes do sistema e que também compartilhasse estas informações com os outros membros da equipe? *

- Extremamente útil.
- Interessante.
- Útil.
- Indiferente.
- Intrusiva

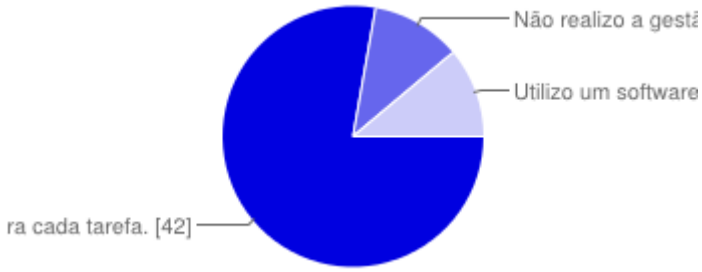
20 - Você utilizaria esta ferramenta descrita na questão 19? *

- Sim.
- Não.

**21 - Em sua opinião, que tipo de características as ferramentas da questões 17 e 19 deveriam possuir? ***As ferramentas realizariam: a monitoração do código-fonte e mostraria a evolução desta codificação na modelagem realizada para o sistema.

APÊNDICE 2 – Respostas ao Questionário Aplicado a Programadores e Gestores de Software em Empresas de Desenvolvimento

01 - Como é realizada a gestão dos artefatos (documentos de texto, planilhas, código-fonte, mensagens e outros) que você produz durante o desenvolvimento do seu trabalho?



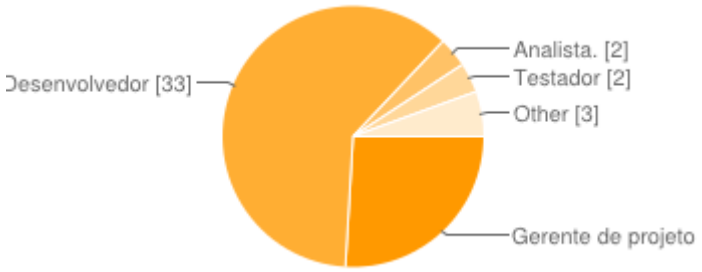
	Quant.	Percentual
Utilizo ferramentas específicas e independentes para cada tarefa.	42	78%
Não realizo a gestão de documentos com nenhuma ferramenta.	6	11%
Utilizo um software integrado para gestão destes artefatos.	6	11%

02 - Se a resposta a pergunta anterior for "Utilizo um software integrado para gestão destes artefatos", por favor, indique o nome do software:

"O software é próprio da organização, antes utilizávamos o DotProject, para o cronograma, agora associamos a função dele a um software organizacional. Também utilizamos para Analise o EA, o qual exportamos os diagramas em html para o SVN que pode ser acessado por todos."

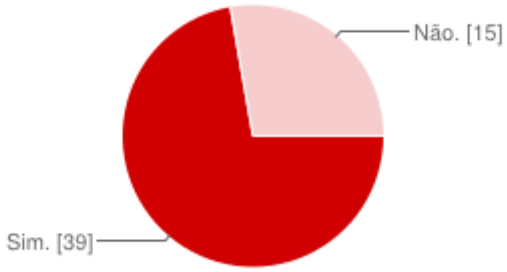
a) "Contour"
b) "Reqpro"
c) "google docs, open goo"
d) "Google"
e) "TortoiseSVN"
f) "ClearCase"
g) "Suíte da IBM Rational"
h) "Sistemas internos da Empresa."
i) "Proprio"

03 - Durante o desenvolvimento do seu trabalho, você desempenha qual papel dentro da equipe?



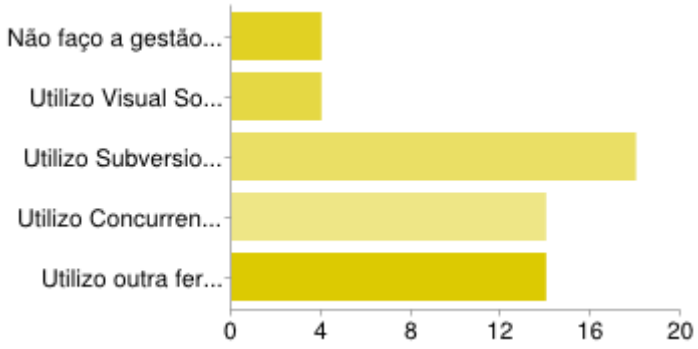
	Quant.	Percentual
Gerente de projeto	14	26%
Desenvolvedor	33	61%
Analista.	2	4%
Testador	2	4%
Outro	3	6%

04 - Este papel desempenhado, pode variar durante o projeto?



	Quant.	Percentual
Sim.	39	72%
Não.	15	28%

05 - A gestão do código-fonte produzido por seu trabalho é realizada por alguma ferramenta de apoio?

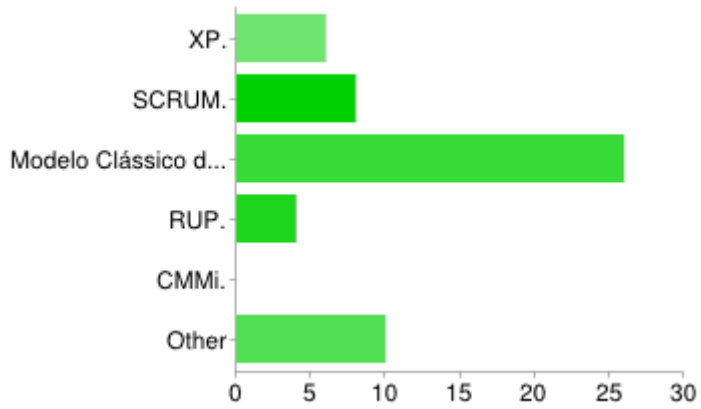


	Quant.	Percentual
Não faço a gestão de código com nenhuma ferramenta.	4	7%
Utilizo Visual Source Safe (VSS).	4	7%
Utilizo Subversion (SVN).	18	33%
Utilizo Concurrent Version System (CVS).	14	26%
Utilizo outra ferramenta.	14	26%

06 - Se a resposta a pergunta anterior for "Utilizo outra ferramenta", por favor, indique o nome da ferramenta:

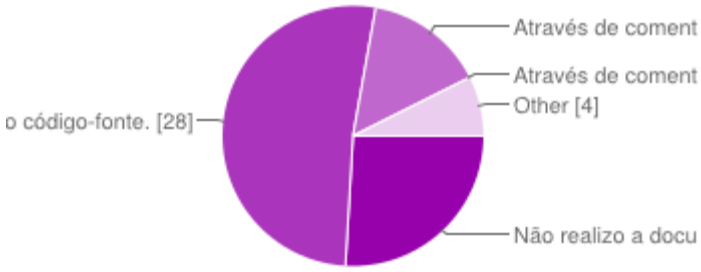
a) MKS
b) ClearQuest
c) ClearCase
d) VCS
e) JEDI VCS
f) JediVCS
g) Jedi VCS
h) Jedi VCS
i) Jedi VCS
j) JEDI
k) JEDI Version Control System
l) VCS
m) Free VCS
n) Jedi VCS

07- Utiliza algum destes processos ou metodologia de desenvolvimento de software?



XP.	6	11%
SCRUM.	8	15%
Modelo Clássico de desenvolvimento.	26	48%
RUP.	4	7%
CMMi.	0	0%
Other	10	19%

08 - Como é realizada a documentação do código-fonte produzido durante o desenvolvimento do seu trabalho?



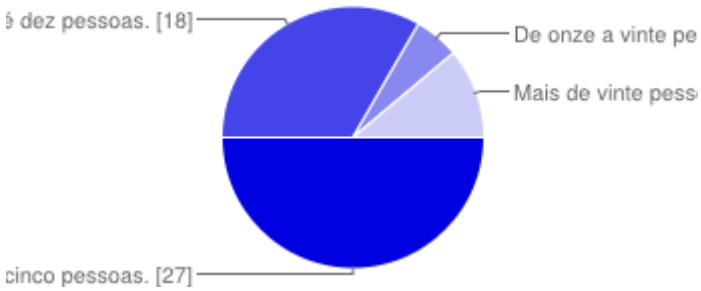
	Quant.	Percentual
Não realizo a documentação do código-fonte.	14	26%
Através de comentários manuais no código-fonte.	28	52%
Através de comentários manuais no código-fonte e de documento criado para este fim.	8	15%
Através de comentários automatizados no código-fonte.	0	0%
Other	4	7%

09 - Se a resposta a pergunta anterior for "Através de comentários automatizados no código-fonte", por favor, indique o nome da ferramenta:

“Também são realizados comentario automaticos quando carregado uma versão para o servidor pelo SVN.”

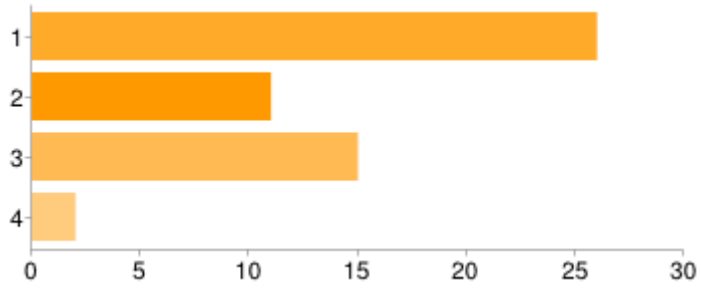
“ViaAction”

10 - Qual o tamanho da equipe na qual você participa para desenvolver software?



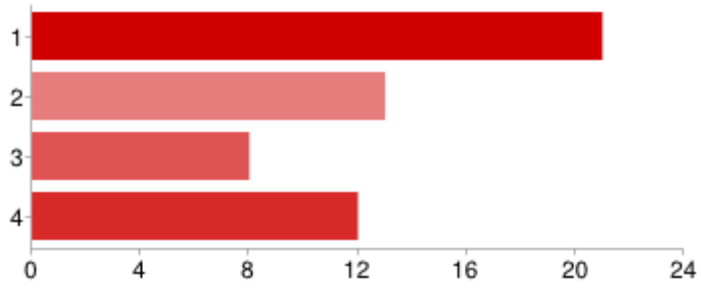
	Quant.	Percentual
De uma até cinco pessoas.	27	50%
De seis até dez pessoas.	18	33%
De onze a vinte pessoas.	3	6%
Mais de vinte pessoas	6	11%

11 - Classifique as ferramentas abaixo conforme o nível de utilização da mesma em relação ao andamento de seu trabalho para a troca de informações ou para resolução problemas - **Email**.



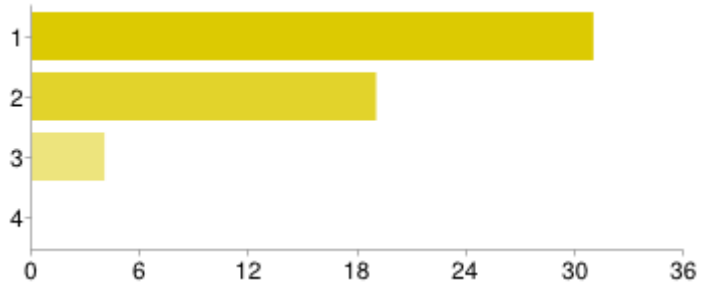
	Quant.	Percentual
1	26	48%
2	11	20%
3	15	28%
4	2	4%

11 - Classifique as ferramentas abaixo conforme o nível de utilização da mesma em relação ao andamento de seu trabalho para a troca de informações ou para resolução problemas - **Chat**.



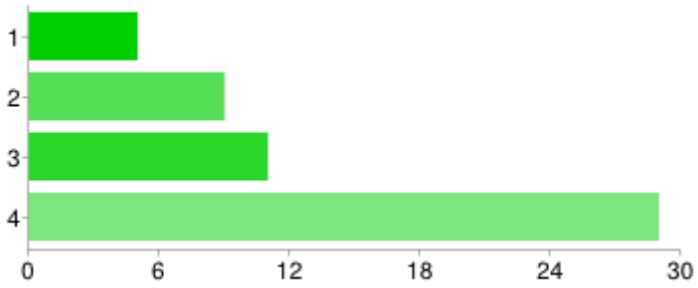
	Quant.	Percentual
1	21	39%
2	13	24%
3	8	15%
4	12	22%

11 - Classifique as ferramentas abaixo conforme o nível de utilização da mesma em relação ao andamento de seu trabalho para a troca de informações ou para resolução problemas - **Comunicação face-a-face.**



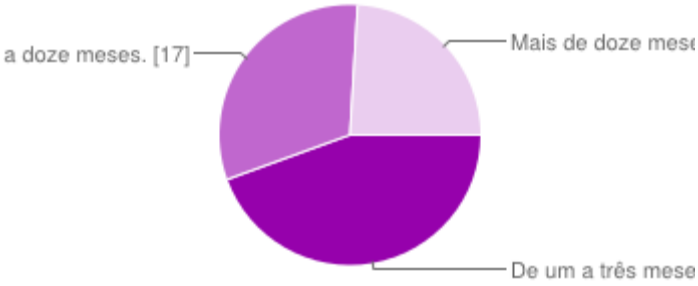
	Quant.	Percentual
1	31	57%
2	19	35%
3	4	7%
4	0	0%

11 - Classifique as ferramentas abaixo conforme o nível de utilização da mesma em relação ao andamento de seu trabalho para a troca de informações ou para resolução problemas - **Outra forma que não as listadas acima.**



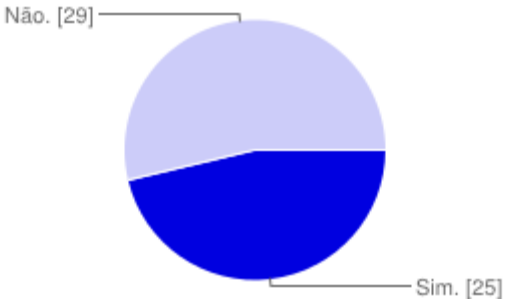
	Quant.	Percentual
1	5	9%
2	9	17%
3	11	20%
4	29	54%

12 - Qual o tempo médio para execução dos projetos de software?



	Quant.	Percentual
De um a três meses.	24	44%
De quatro a doze meses.	17	31%
Mais de doze meses.	13	24%

13 - Você ou alguém de sua equipe utiliza alguma ferramenta de controle de projetos?

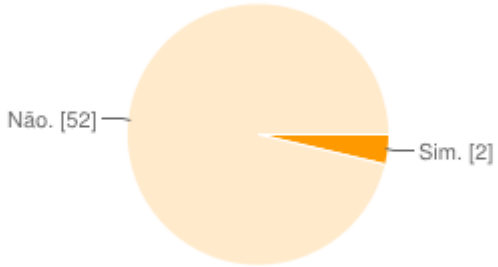


	Quant.	Percentual
Sim.	25	46%
Não.	29	54%

14 - Se a resposta a pergunta anterior for "Sim", por favor, indique o nome da ferramenta:

a. msproject
b. Própria da organização.
c. Microsoft Project
d. REQPROJ
e. Ms Project
f. reqproj
g. MS Project
h. Ms Project
i. ReqProj
j. redmine
k. Redmine
l. pivotal
m. Open Workbench
n. Microsoft Project
o. IBM Rational Project
p. MS Project
q. Ferramenta interna
r. project
s. ViaAction - Software desenvolvido internamente
t. VsProject
u. Sistema Interno.
v. VsProject
w. dot.Project / GPWeb

15 - Para o desenvolvimento do seu trabalho na codificação, existe algum tipo de controle que mostra se o código-fonte produzido está refletindo os diagramas projetados para o sistema?



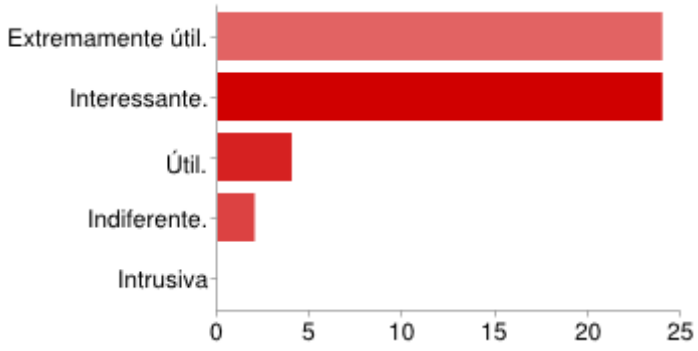
Quant. - Percentual

Sim. 2 4%
Não. 52 96%

16 - Se a resposta a pergunta anterior for "Sim", por favor, indique o nome da ferramenta:

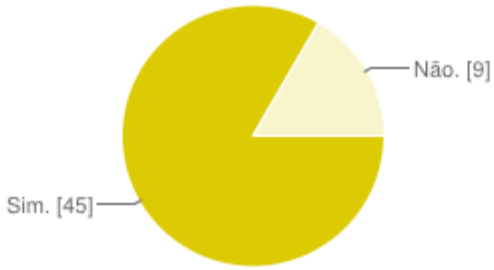
--- SEM RESPOSTAS PARA ESTA PERGUNTA-

17 - Qual a sua opinião sobre uma ferramenta que monitorasse o código-fonte, por exemplo: a criação de classes, alteração de métodos, etc. e que compartilhasse estas informações com os outros membros da equipe?



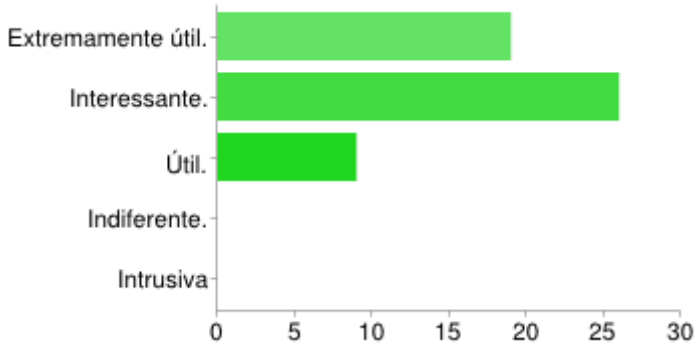
	Quant.	Percentual
Extremamente útil.	24	44%
Interessante.	24	44%
Útil.	4	7%
Indiferente.	2	4%
Intrusiva	0	0%

18 - Você utilizaria esta ferramenta descrita na questão17?



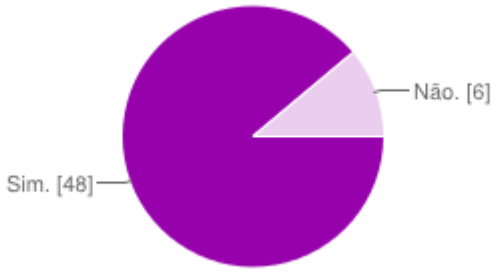
	Quant.	Percentual
Sim.	45	83%
Não.	9	17%

19 - Qual a sua opinião sobre uma ferramenta que mostrasse o progresso do código-fonte desenvolvido e confrontasse, por exemplo, com um diagrama de classes do sistema e que também compartilhasse estas informações com os outros membros da equipe?



	Quant.	Percentual
Extremamente útil.	19	35%
Interessante.	26	48%
Útil.	9	17%
Indiferente.	0	0%

20 - Você utilizaria esta ferramenta descrita na questão 19?



	Quant.	Percentual
Sim.	48	89%
Não.	6	11%

21 - Na sua opinião, que tipo de características as ferramentas da questões 17 e 19 deveriam possuir para realmente serem úteis durante o processo de programação/codificação?

a) “auxiliar os programadores a saberem o que os colegas estao programando.”

b) "Um dos maiores problemas hoje é conflitar a análise com o código fonte, principalmente nas linguagens predominantes na nossa região. Outro problema grave é a padronização de código e comentários, além da cultura fraca do assunto, tem o problema também de perder tempo, seria interessante associair a automatização ao processo de inserir comentário, também padrão de códigos, como identação."

c) “Estar integrada com as demais ferramentas já utilizadas no desenvolvimento.”

d) “Controle de tarefas com prazo determinado dentro das classes do projeto. “

e) “Fácil uso, não "engessar" o processo. “

f) "A ferramenta da 17, poderia ser integrado com alguma ferramenta de diagramação e efetuar o processo inverso, documentando as alterações em diagramas de classe a partir do código criado, tornando mais ágil, o papel do analista, que poderia já criar, além do diagrama, a implementação da classe. A ferramenta da 19 deveria permitir que, a partir das diferenças detectadas entre o código fonte e o diagrama, fosse possível corrigir tanto o código fonte (gerando declaração dos métodos que não foram implementados, propriedades que ainda não foram criadas, ou corrigindo os encapsulamentos das propriedades e métodos, por exemplo), quanto o diagrama (criando no diagrama, os métodos e propriedades que estão no código, mas não no diagrama), ajudando a manter tanto o documento quanto o código, sincronizados um com o outro. O processo deveria funcionar de forma arbitrária ao programador/analista, sendo da escolha dele, qual artefato será readequado ao outro (corrigir fonte de acordo com diagrama ou corrigir diagrama de acordo com fonte). Isto agilizaria o processo de implementação do código, assim como correção da documentação.”

g) “Envio de email diário com resumo, ou notificação no desktop.”

h) Auto-documentação, sincronismo com o projeto, Facilidade pra elaboração dos Manuais do Sistema, etc...

i) “Permitir a criação de dashboards personalisados para cada gerente de projetos, podendo exibir o progresso do projeto em sintese ou detalhadamente.”

j) “ Seria interessante se possível que fossem integradas com as IDEs de desenvolvimentos mais utilizadas, como eclipse por exemplo.”

k) “Integrassem os diagramas UML realizados com o código-fonte, e possíveis refatorações no código também refletissem no diagrama. E que integrasse também com casos de uso e requisitos, mostrando o que impacta a mudança de um código ou requisito em ambos e facilitando a visualização de uma regra de negócio implementada no código para não-desenvolvedores.”

l) “Devem ser muito práticas, sem limitação a linguagem de programação (em caso de projetos com mais de uma linguagem). Monitoramento para a criação de classes ser configurável para atingir o nível de customização que cada gerente de processo definir como aceitável para sua equipe de desenvolvedores.”

m) “A ferramenta de monitoração de código fonte deveria conseguir sintetizar de forma clara e de fácil entendimento as alterações, pois se as informações são apresentadas de forma complexa ou que exija muito tempo para entende-las, logo os desenvolvedores irão abandona-la, deixando de avalia-las. Quanto a ferramenta de confrontação com o modelo, é util, mas infelizmente muito se altera na fase de desenvolvimento da solução, e nem sempre isso é refletido no modelo lógico, portanto, se a empresa realmente possui uma boa gestão de documentação de software, a ferramenta realmente é útil, caso contrário, só atrapalhará o desenvolvimento e logo será abandonada."

n) "- Função ""compare"". - Gráficos (número de classes ou colunas criadas por exemplo)."

o) "Impedir que dois ou mais desenvolvedores estejam utilizando o mesmo código-fonte. Controlar versões do código para posterior consulta das alterações."

p) “Simples de manusear e que exibisse alguma informação referente a produtividade de cada membro da equipe.”

q) "Se a modelagem for feita pela ferramenta, penso que seria interessante algum tipo de geração automática de classes, tipo Getters e Setters para os membros da classe.”

r) "A característica fundamental é não ser mais uma carga. Se a ferramenta demandar muito trabalho nela, não será usada. Ela precisa prestar serviço importante, facilitar a vida do desenvolvedor e do gerente de projeto, sem se tornar um estorvo, uma tarefa chata. Boa sorte!”

s) “Extremamente útil."

APÊNDICE 3 – Descrição do OPERAM e Cenário de Utilização do Módulo

Uma visão geral do OPERAM é visto na Figura 30, que foi baseado nos conceitos do padrão MVC (*Model-View-Controller* – SENGUPTA *et al.,* 2007). Esse padrão divide em três camadas básicas a aplicação, sendo elas:

- Camada de negócio: Esta camada contém os objetos e serviços que possuem as informações do negócio, tais como, por exemplo, dados do banco de dados.
- Camada de Visualização: Contém as páginas web e recursos utilizados nas mesmas.
- Camada de controle: Contém os componentes que ligarão as telas com os objetos e serviços do negócio.

Dessa forma as camadas ficarão distintas, promovendo manutenção facilitada e baixo acoplamento entre componentes, o que proporcionará a reutilização dos mesmos como integração em outros módulos. O OPERAM foi elaborado no padrão Cliente–Servidor, para proporcionar que o sistema seja distribuído em um ou mais servidores, e estes servidores serem acessados através de uma aplicação cliente, que neste cenário de utilização apresentado, será uma página web acessando o OPERAM por meio do protocolo HTTP.

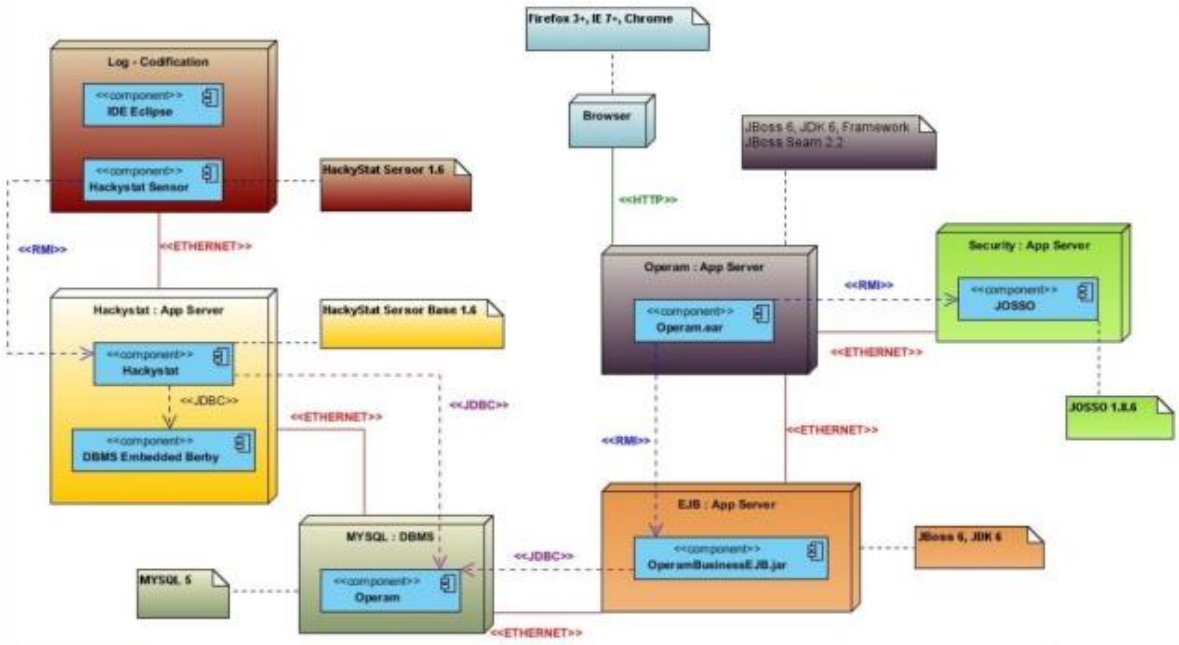


Figura 30 – Visão Geral do OPERAM.

Fonte: própria.

Descrição do OPERAM para a Gestão de Projetos

Para mostrar como o OPERAM funciona, alguns itens de sua operação são descritos, para situar de forma geral como foi idealizado para sua atuação nos experimentos para comprovar sua viabilidade técnica.

Na Figura 31 apresenta-se a tela de cadastro de um projeto no módulo desenvolvido. Este cadastro é necessário para que vários projetos pudessem ser inseridos e monitorados, simulando uma atividade da rotina normal de um gerente que acompanha mais de um projeto ao mesmo tempo.

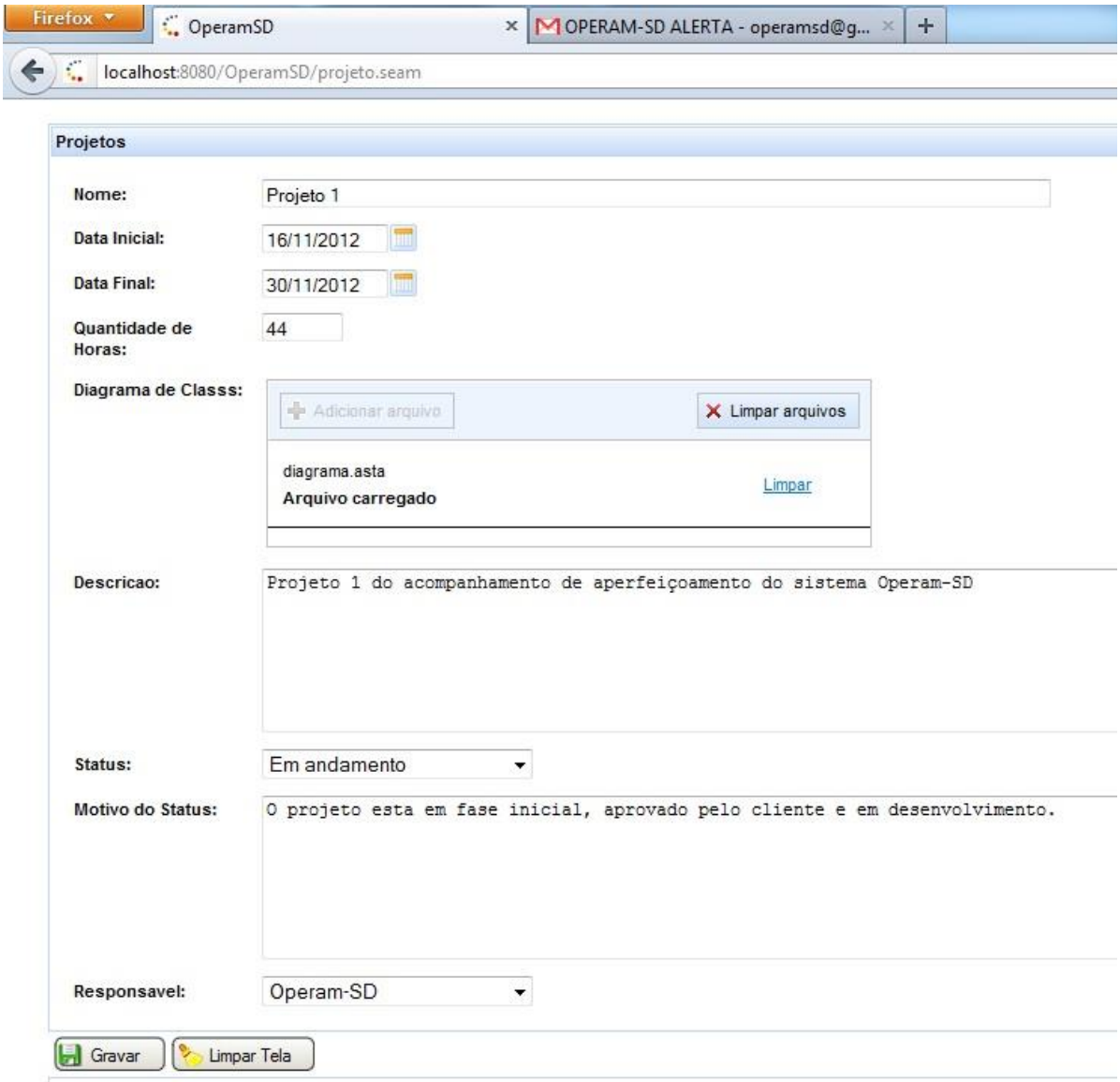


Figura 31 – Cadastro de um Projeto no OPERAM.

Fonte: própria.

Para listar os projetos cadastrados no sistema, a figura 32 mostra a tela onde podem ser consultados e manipulados.

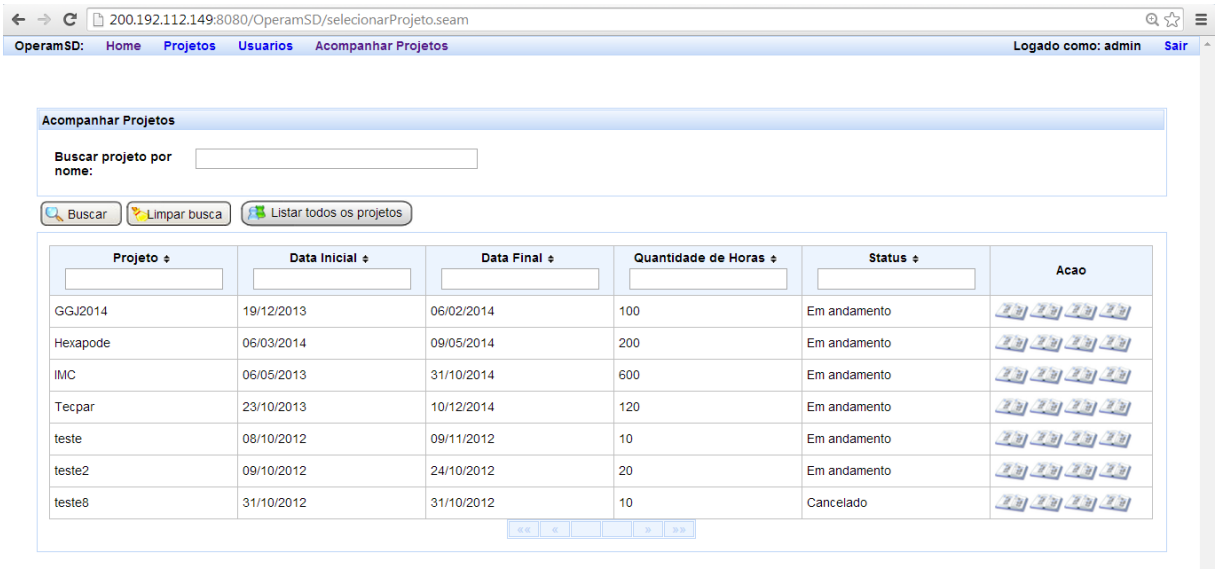


Figura 32 – Projetos cadastrados no OPERAM.

Fonte: própria.

Os usuários desempenham um papel importante na execução das tarefas e no levantamento dos dados necessários no OPERAM. A figura 33 apresenta a tela onde este cadastro e vinculação com os projetos onde o usuário irá atuar.

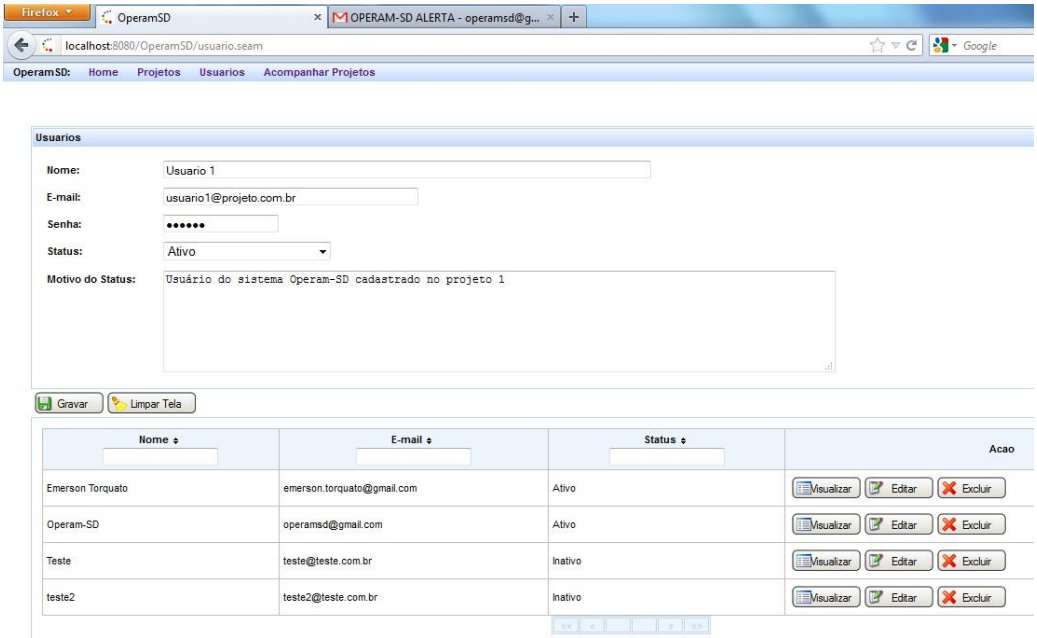


Figura 33 – Tela de cadastro de um Projeto no OPERAM.

Fonte: própria.

Algumas funcionalidades que auxiliam no gerenciamento dos projetos estão disponíveis no OPERAM, tais como o aviso ao responsável do sistema sobre o andamento do trabalho, por meio do envio de e-mails sobre a situação em que se encontra o projeto. A figura 34 mostra esta funcionalidade.

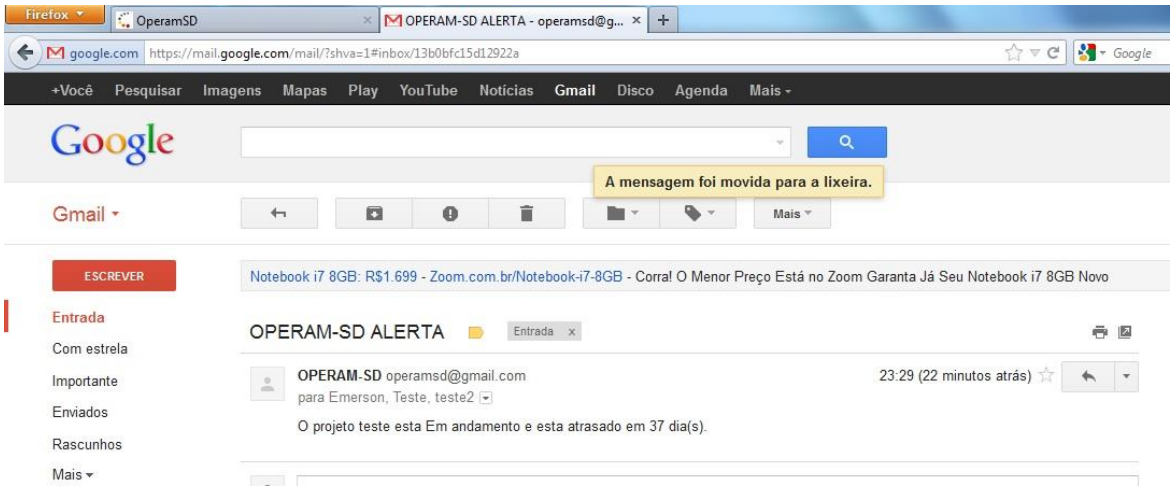


Figura 34 – Envio de e-mails.

Fonte: própria.

Um dos principais itens é a possibilidade do acompanhamento da codificação, sendo esta confrontada com a modelagem do sistema realizada em fases anteriores, principalmente utilizando diagramas da modelagem UML. O OPERAM busca as informações do projeto (atividades e usuários relacionados e o diagrama de classes do sistema a ser desenvolvido), como também busca as informações de codificação coletadas pelo HACKYSTAT, faz a inferência dessas informações e atualiza a estas informações. O sistema verifica o diagrama de classes cadastrado no projeto e gera, com base nas informações coletadas, a arquitetura da classe que será utilizada posteriormente para o acompanhamento do projeto, bem como também os atributos da classe, métodos e atributos do projeto, unificando todas estas informações. O usuário poderá acompanhar graficamente o projeto no estado atual de acordo com uma data selecionada, podendo assim fazer um acompanhamento cronológico do desenvolvimento do sistema.

No caso de alteração de dados durante a programação pelos usuários responsáveis pela codificação, o mesmo processo descrito acima será repetido. Porém executará um processo, onde verificará as alterações na arquitetura (classes, atributos, métodos do projeto) e refletirá estas alterações para a nova situação da arquitetura, atualizando, atribuindo, alterando ou excluindo os objetos de acordo com a nova situação do diagrama de classes que será gerado automaticamente pelo processo de programação do código-fonte por parte do programador. A figura 35 , nota-se a imagem gerada pelo próprio módulo de teste do OPERAM, mostra o processo descrito anteriormente, onde pode-se observar as cores que fornecem a indicação visual do andamento do projeto.

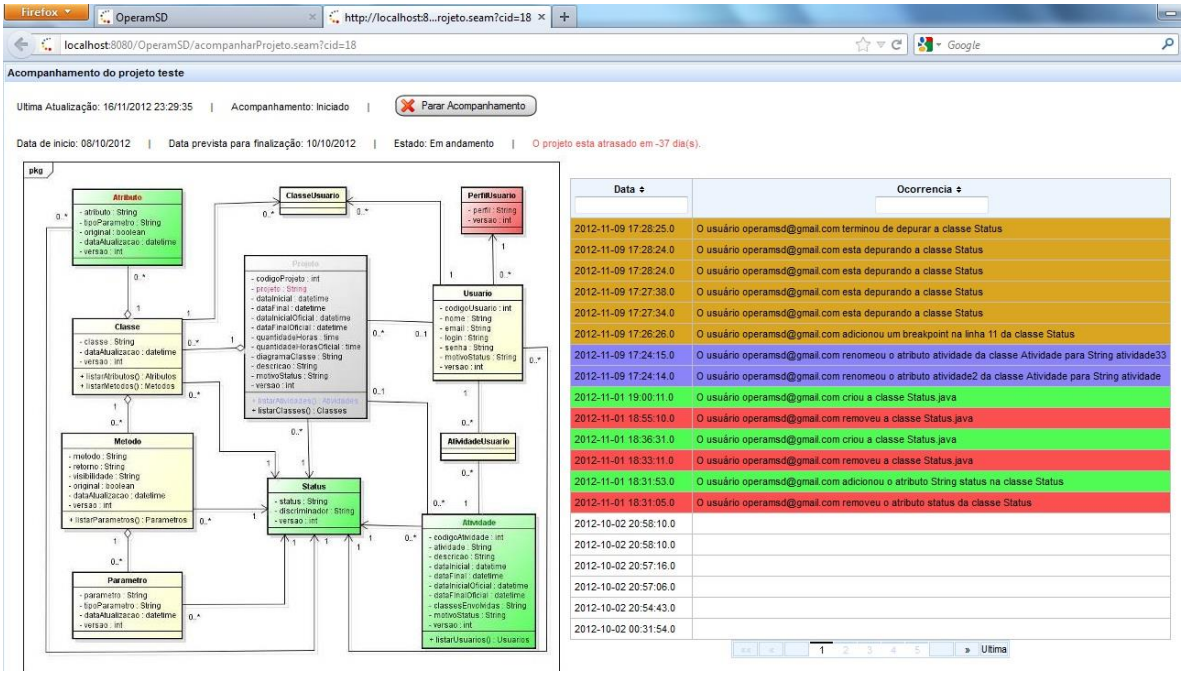


Figura 35 – Cadastro de um Projeto no OPERAM.

Fonte: própria.

Cenário de Uso do Módulo OPERAM

Pré-Condições:

1) Possuir os usuários/desenvolvedores **teste@teste.com.br** e **teste2@teste.com.br** criados e ativos no **Módulo**
2) Possuir os projetos **teste** e **teste2** criados corretamente no **Módulo**.
3) Possuir dentro dos diagramas de classe dos projetos teste e teste2 a classe Status.

4) Possuir os projetos JAVA e **teste2** criados no Workspace do Eclipse dos dois usuários citados acima.
5) Iniciar o acompanhamento dos dois projetos citados acima no **Módulo**

Um cenário de utilização:

1) Na tela de cadastro de usuários, desativar o usuário **teste@teste.com.br**. No eclipse do desenvolvedor, na aba de configuração do **Hackystat**, inserir o usuário **teste@teste.com.br**, a senha definida e clicar no botão OK. Deverá apresentar a mensagem de usuário inválido.

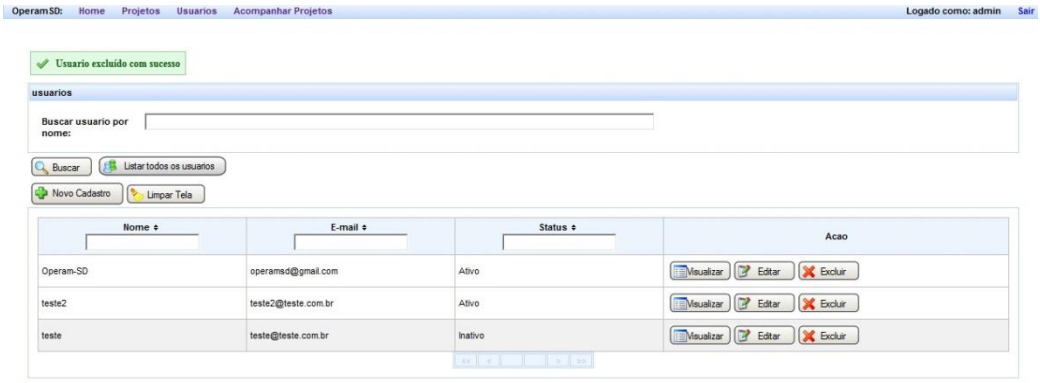


Figura 36 – Tela de cadastro de usuários.

Fonte: própria.

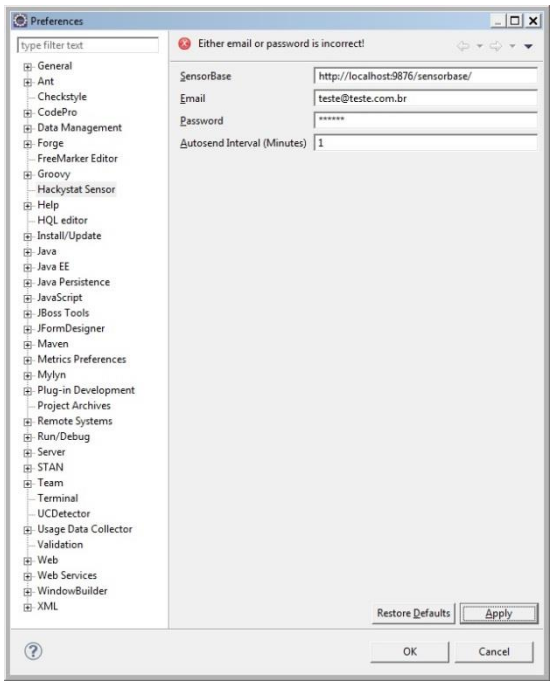


Figura 37 – Sensor no Eclipse.

Fonte: própria.

2) Na tela de cadastro de usuários, ative novamente o usuário **teste@teste.com.br**. No eclipse do desenvolvedor, na aba de configuração do **Hackystat**, inserir o usuário **teste@teste.com.br**, a senha definida e clicar no botão OK. Não deverá apresentar nenhuma mensagem e a aba de configuração será fechada, indicando que o usuário agora é válido.

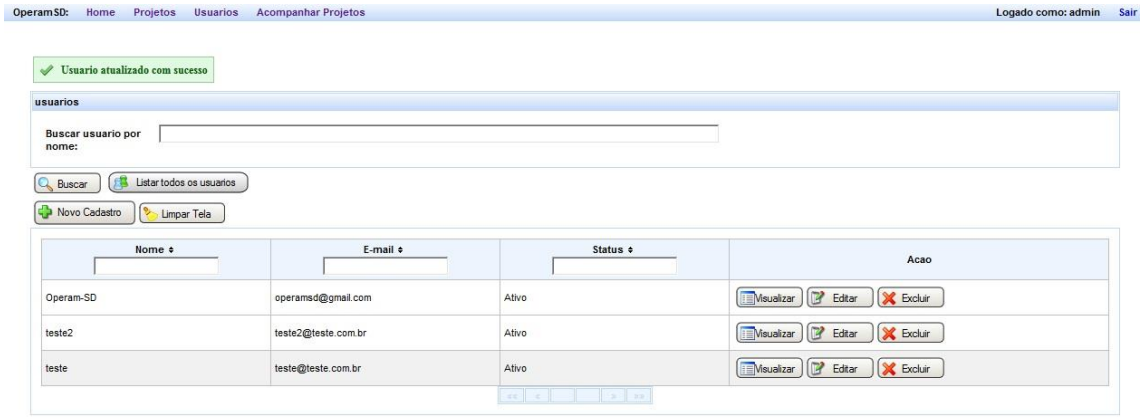


Figura 38 – Ativação de usuários.

Fonte: própria.

3) Dentro do projeto JAVA **teste**, na workspace do Eclipse, o desenvolvedor **teste@teste.com.br** criou uma classe chamada de **Status**. Na tela de acompanhamento do projeto **teste** do **Módulo**, deverá a classe **Status** do diagrama de classes, situado ao lado esquerdo da tela, ficar verde e aparecer log na cor verde, situado a lado direito da tela, indicando a criação da classe.

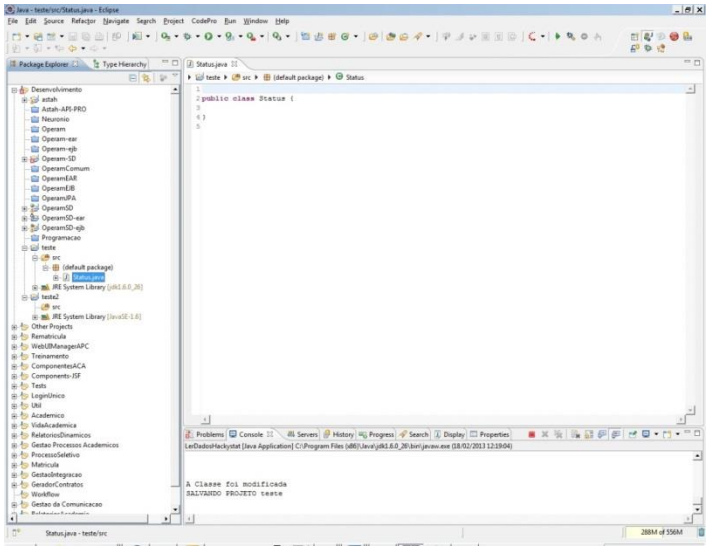


Figura 39 – Hackystat no Eclipse.

Fonte: própria.

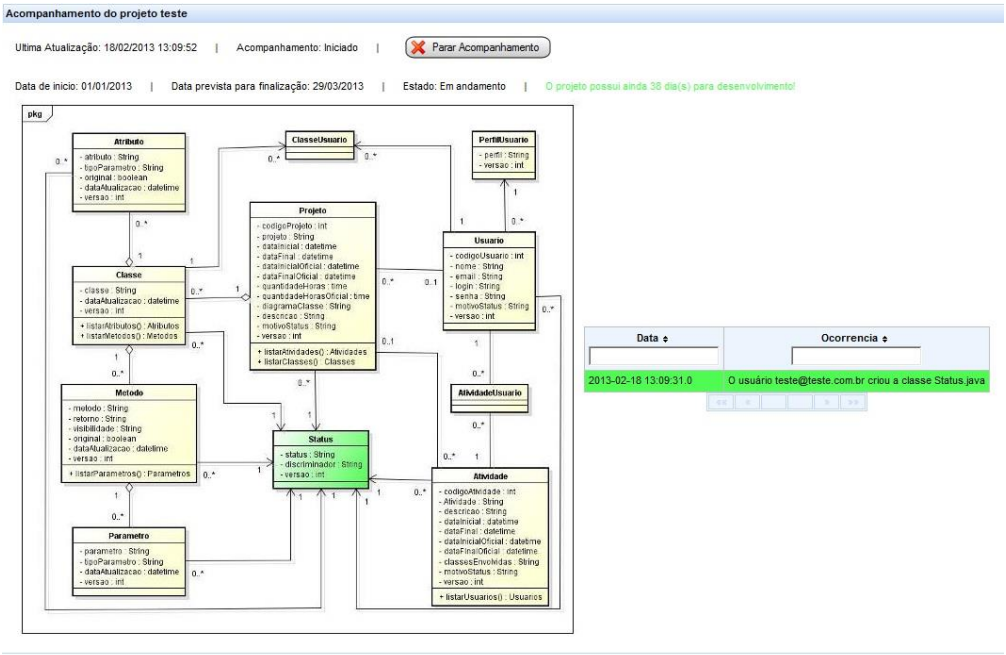


Figura 40 – Início do acompanhamento no OPERAM.

Fonte: própria.

4) Dentro do projeto JAVA **teste**, na workspace do Eclipse, o desenvolvedor **teste@teste.com.br** excluiu a classe chamada de **Status**. Na tela de acompanhamento do projeto **teste** do Módulo, deverá a classe **Status** do diagrama de classes, situado ao lado esquerdo da tela, ficar vermelha e aparecer um log na cor vermelha, situado a lado direito da tela, indicando a exclusão da classe.

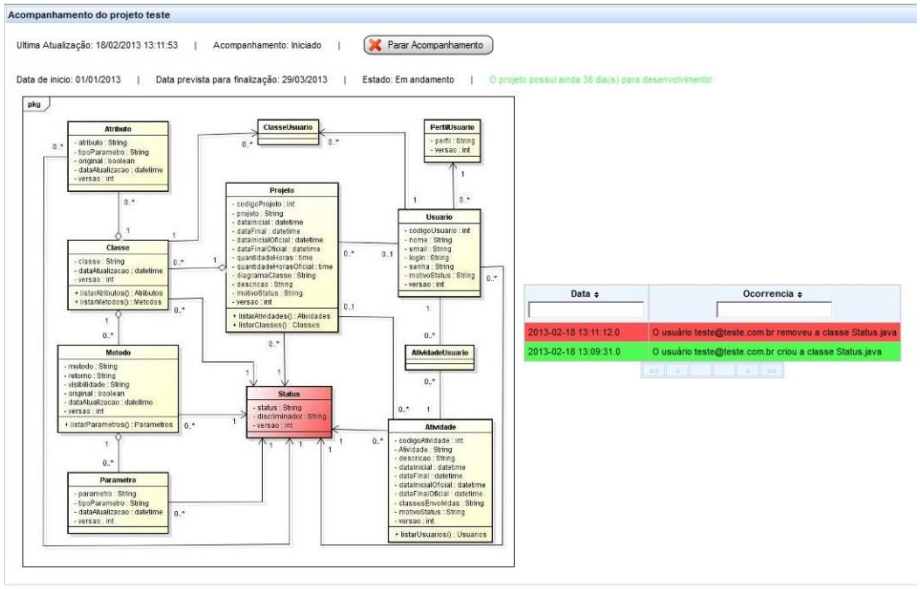


Figura 41 – Acompanhamento de exclusão no OPERAM.

Fonte: própria.

5) **Executar o teste número 3 novamente.**

6) Dentro do projeto JAVA **teste**, na workspace do Eclipse, o desenvolvedor **teste@teste.com.br** criou um atributo do tipo **String** chamado de **nome** dentro da classe **Status**. Na tela de acompanhamento do projeto **teste** do Módulo, deverá aparecer um log na cor verde, indicando a criação do atributo **nome** na classe **Status**.

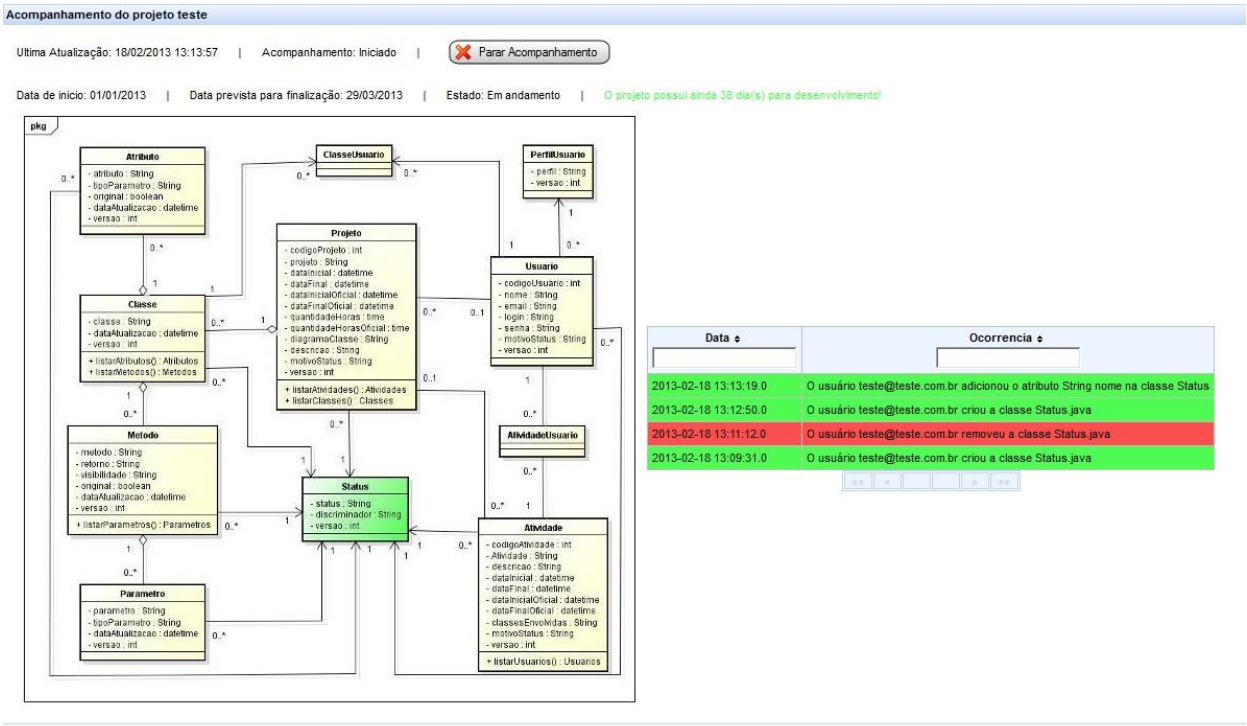


Figura 42 – Criação de itens e acompanhamento no OPERAM.

Fonte: própria.

7) Dentro do projeto JAVA **teste**, na workspace do Eclipse, o desenvolvedor **teste@teste.com.br** renomeou o atributo chamado **nome** para **nome2**, pertencente à classe **Status**. Na tela de acompanhamento do projeto **teste** do **Módulo**, deverá aparecer um log na cor azul, indicando a renomeação do atributo **nome** para **nome2** na classe **Status**.

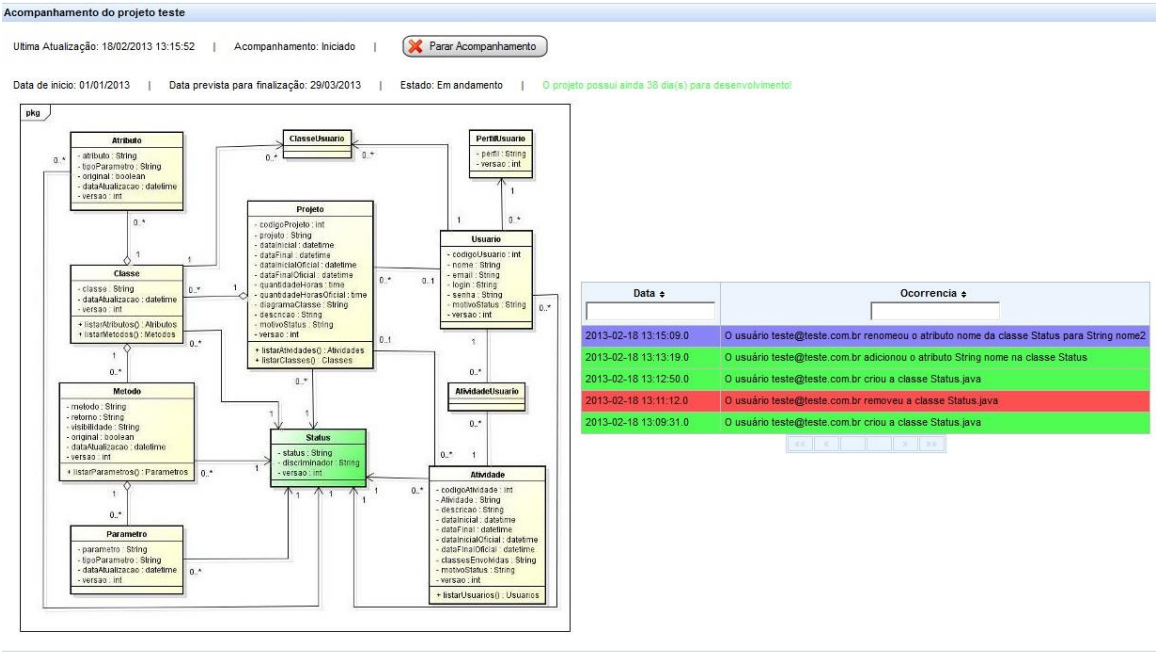


Figura 43 – Alteração de itens no OPERAM.

Fonte: própria.

8) Dentro do projeto JAVA **teste**, na workspace do Eclipse, o desenvolvedor **teste@teste.com.br** excluiu o atributo **nome2** pertencente à classe **Status**. Na tela de acompanhamento do projeto **teste** do **Módulo**, deverá aparecer um log na cor vermelha, indicando a exclusão do atributo **nome2** da classe **Status**.

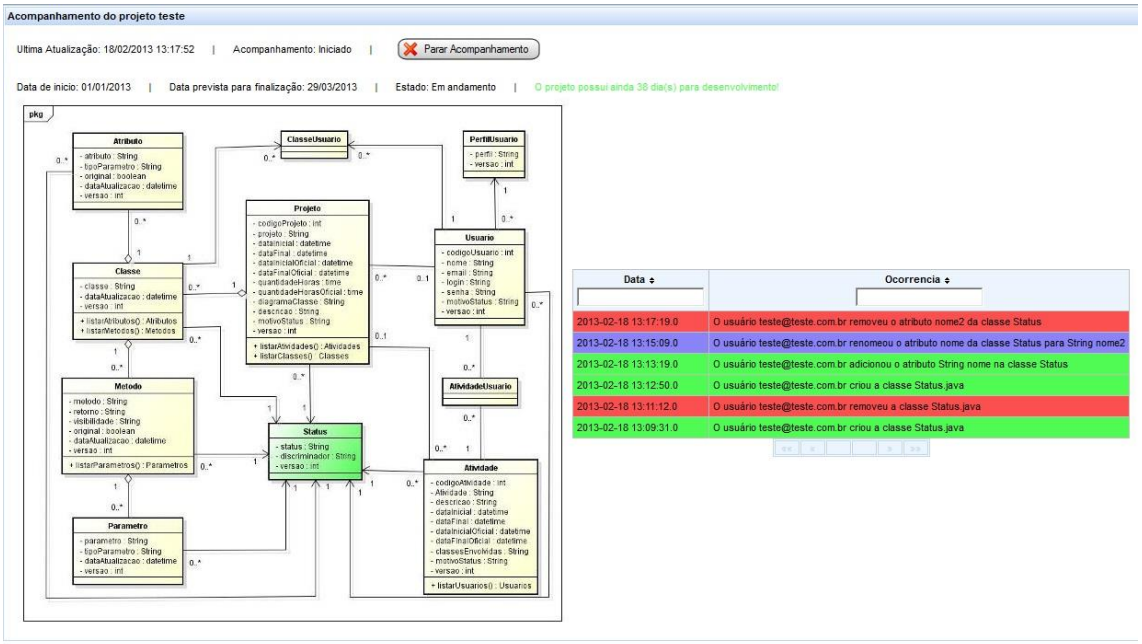


Figura 44 – Remoção de Itens no OPERAM.

Fonte: própria.

9) Dentro do projeto JAVA **teste**, na workspace do Eclipse, o desenvolvedor **teste@teste.com.br** criou um método chamado de **metodoTeste** na classe **Status**. Na tela de acompanhamento do projeto **teste** do **Módulo**, deverá aparecer um log na cor verde, indicando a criação do método na classe **Status**.

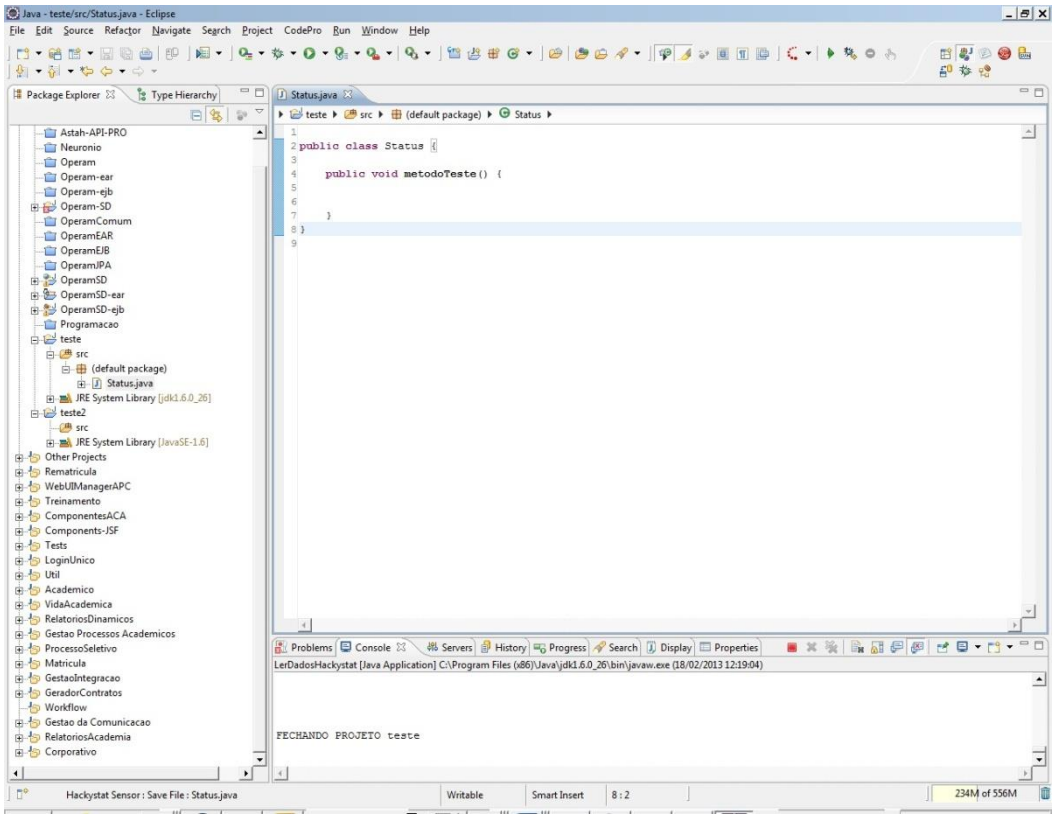


Figura 45 – Criação de itens no Eclipse.

Fonte: própria.

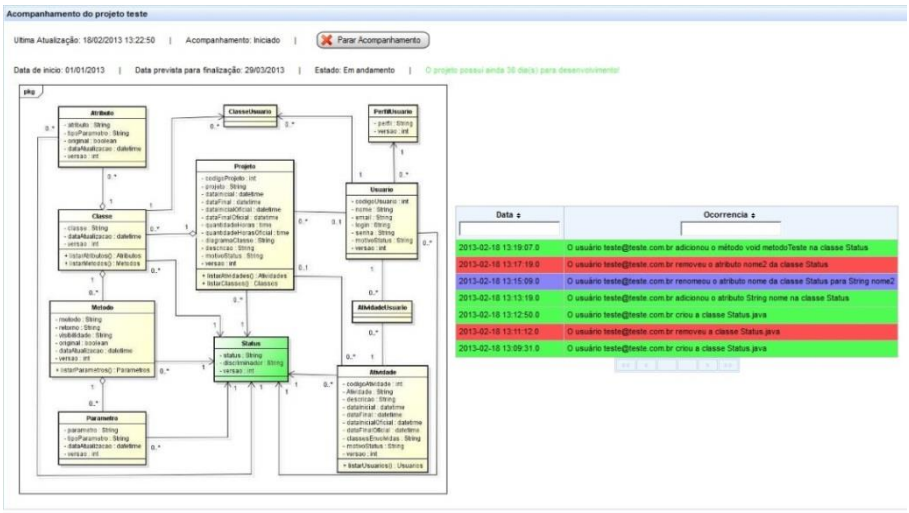


Figura 46 – Acompanhamento da criação de itens no OPERAM.

Fonte: própria.

10) Dentro do projeto JAVA **teste**, na workspace do Eclipse, o desenvolvedor **teste@teste.com.br** inseriu alguns códigos dentro do método **métodoTeste**, pertencente à classe **Status**. Após isso, inserir um **ponto de parada (BreakPoint)** na linha de algum desses códigos inseridos. Na tela de acompanhamento do projeto **teste** do **Módulo**, deverá aparecer um log na cor amarelo escuro, indicando a inserção de um **ponto de parada (BreakPoint)** na linha específica no método **metodoTeste** na classe **Status**.

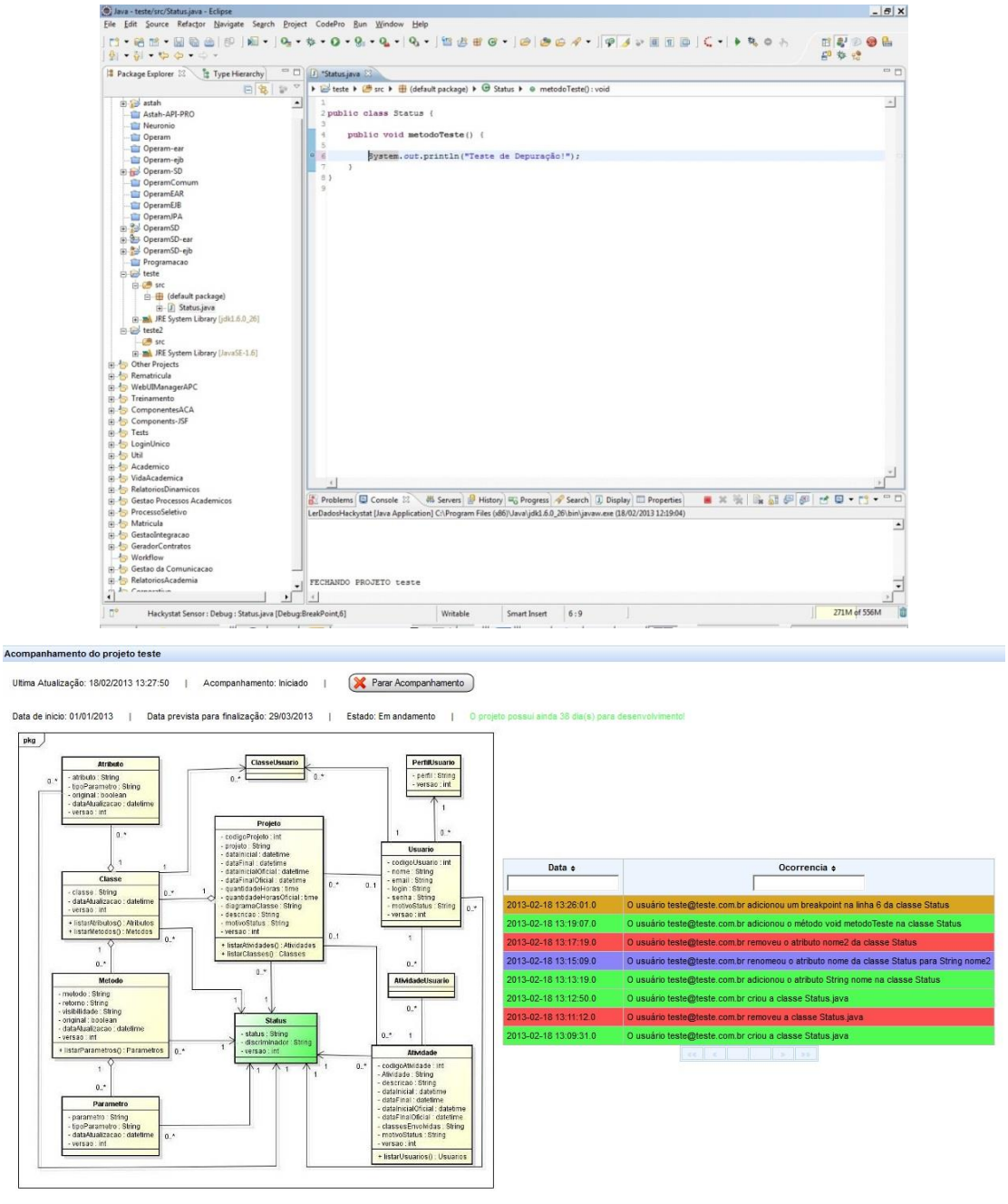


Figura 47 – Monitoração de breakpoints no OPERAM.

Fonte: própria.

11) Dentro do projeto JAVA **teste**, na workspace do Eclipse, o desenvolvedor **teste@teste.com.br** removeu o **ponto de parada (BreakPoint)** inserido no caso de teste anterior. Na tela de acompanhamento do projeto **teste** do **Módulo**, deverá aparecer um log na cor amarelo escuro, indicando a remoção do **ponto de parada (BreakPoint)** na linha específica no método **metodoTeste** na classe **Status**.

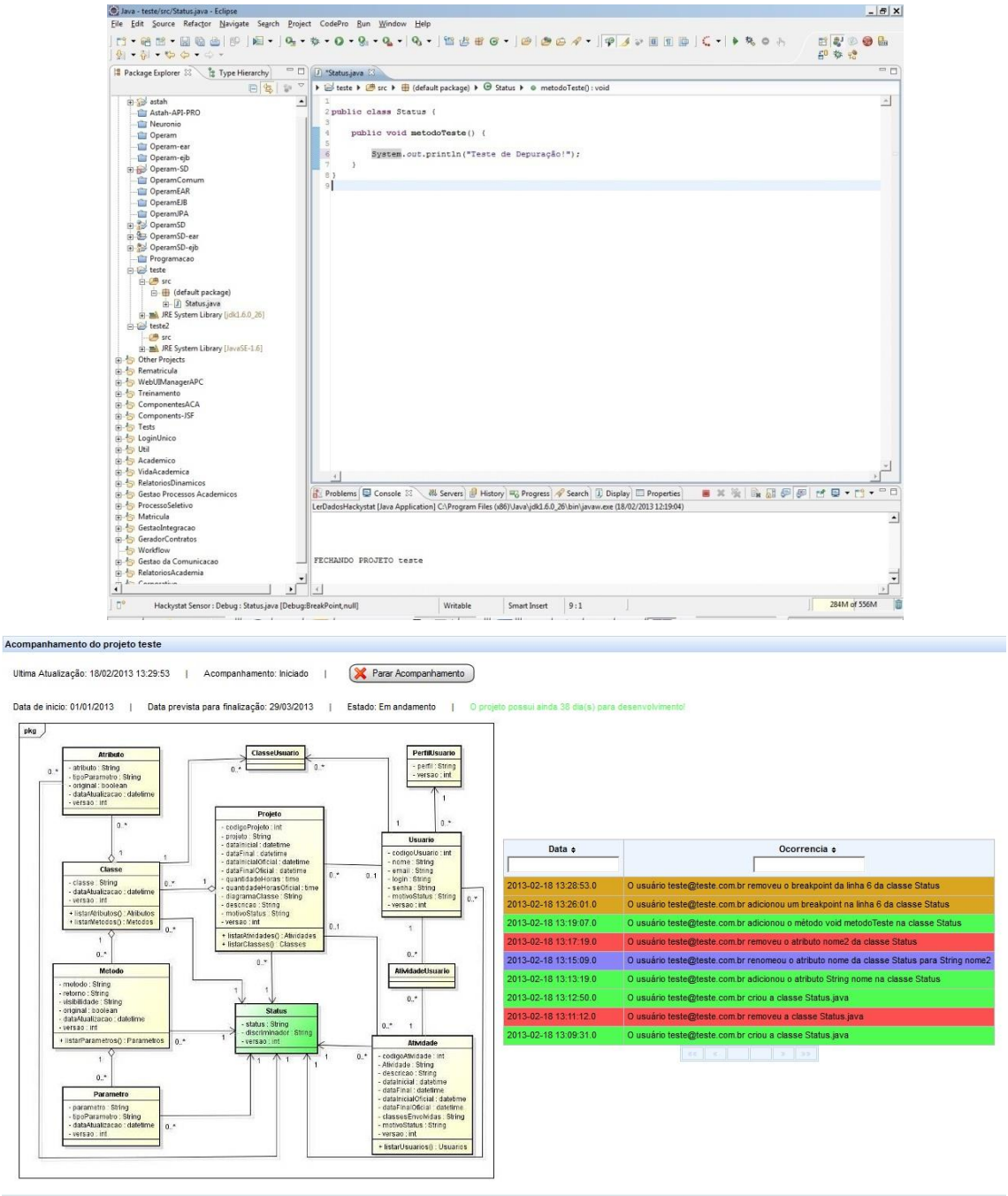


Figura 48 – Acompanhamento da remoção de um breakpoint no OPERAM.

Fonte: própria.

12) Executar o teste número 10 novamente.

13) Dentro do projeto JAVA **teste**, na workspace do Eclipse, o desenvolvedor **teste@teste.com.br** iniciou a **Depuraração (DEBUG)** da classe **Status**. Na tela de acompanhamento do projeto **teste** do **Módulo**, deverá a classe Status do diagrama de classe ficar na cor amarelo escuro e aparecer um log também na cor amarelo escuro, indicando que a classe **Status** está sobre um processo de **Depuração (DEBUG).**

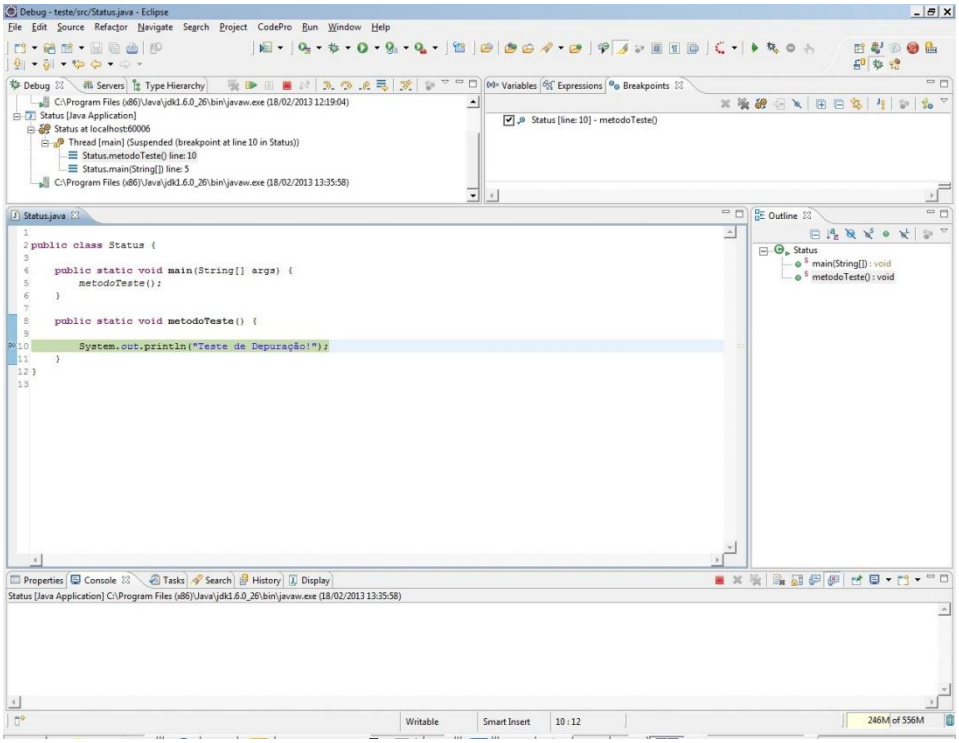


Figura 49 – IDE Eclipse na criação de itens sendo monitorada.

Fonte: própria.

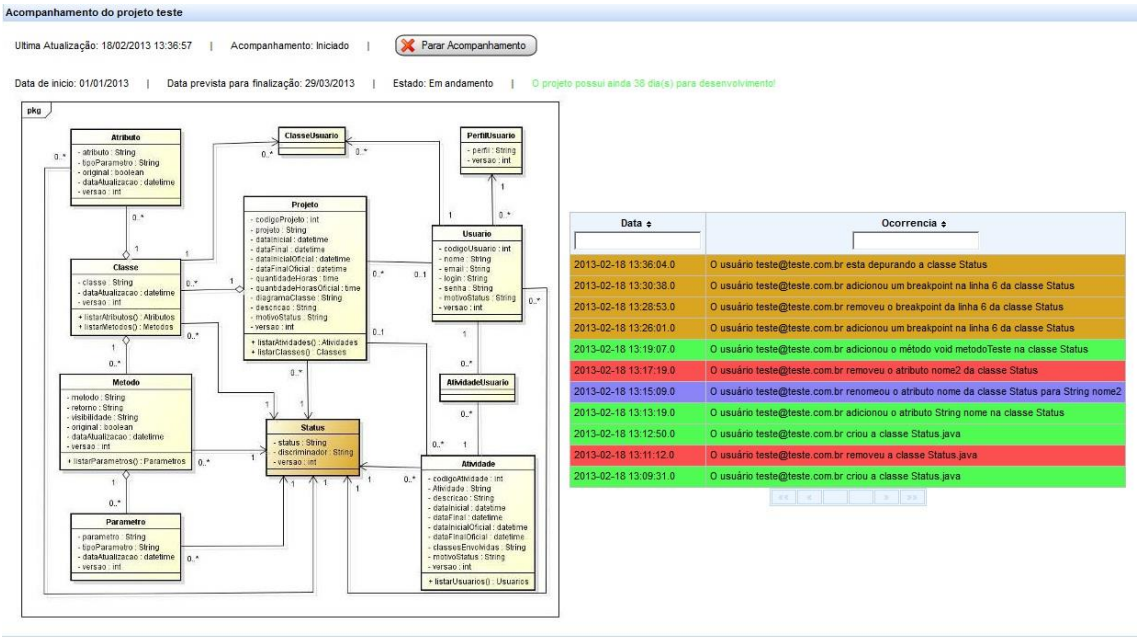


Figura 50 – Troca de cor de acompanhamento no OPERAM.

Fonte: própria.

14) Dentro do projeto JAVA **teste**, na workspace do Eclipse, o desenvolvedor **teste@teste.com.br** terminou a **Depuração (DEBUG)** da classe **Status**. Na tela de acompanhamento do projeto **teste** do **Módulo**, deverá a classe Status do diagrama de classe ficar na cor verde e aparecer um log na cor amarelo escuro, indicando que o processo de **Depuração (DEBUG)**.

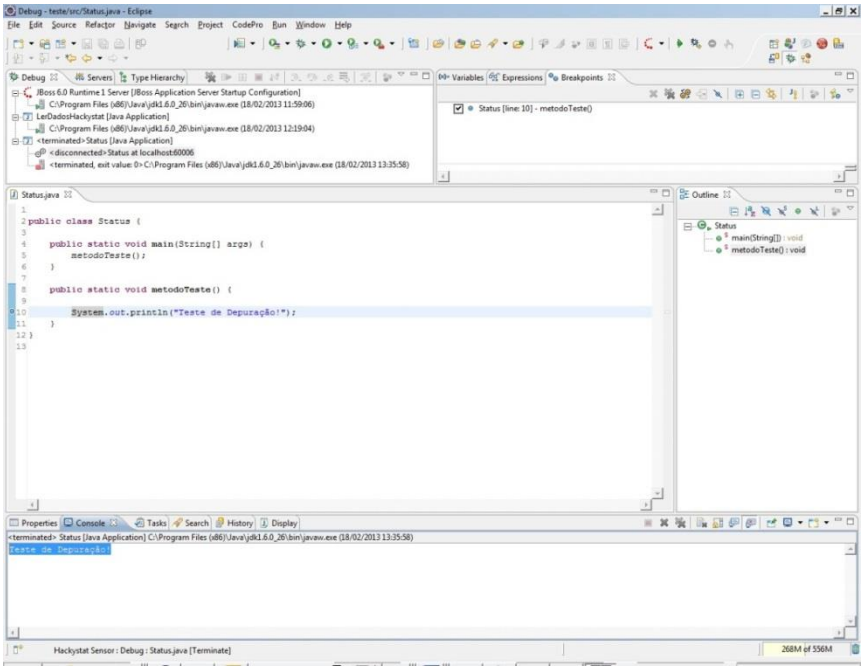


Figura 51 – Interface da IDE Eclipse na criação de uma classe.

Fonte: própria.

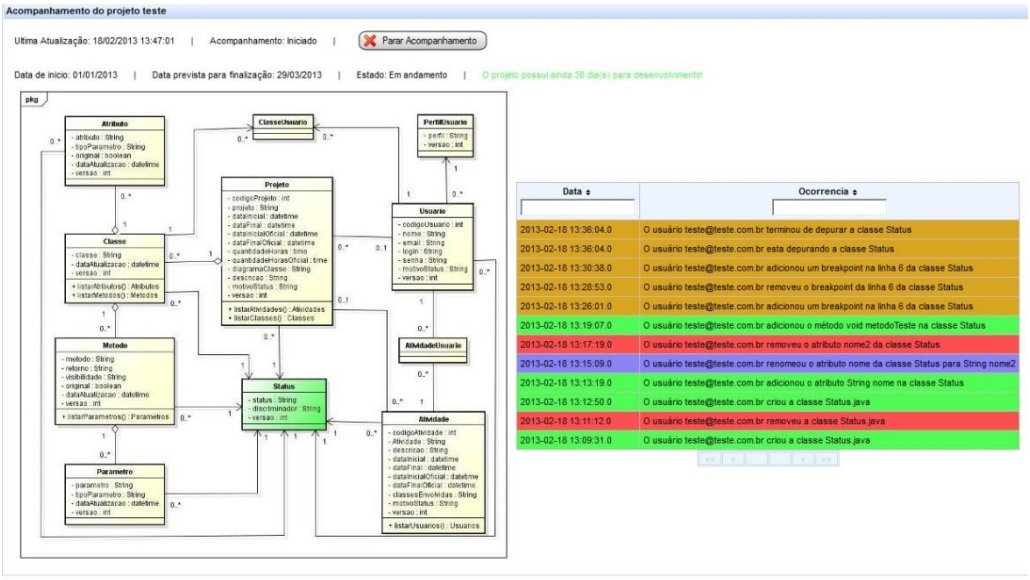
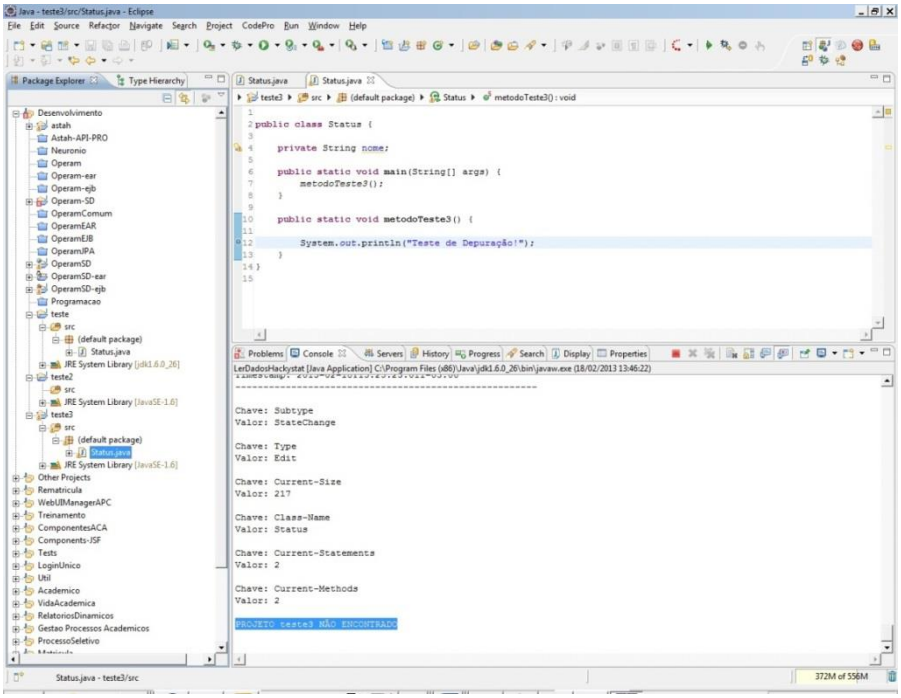


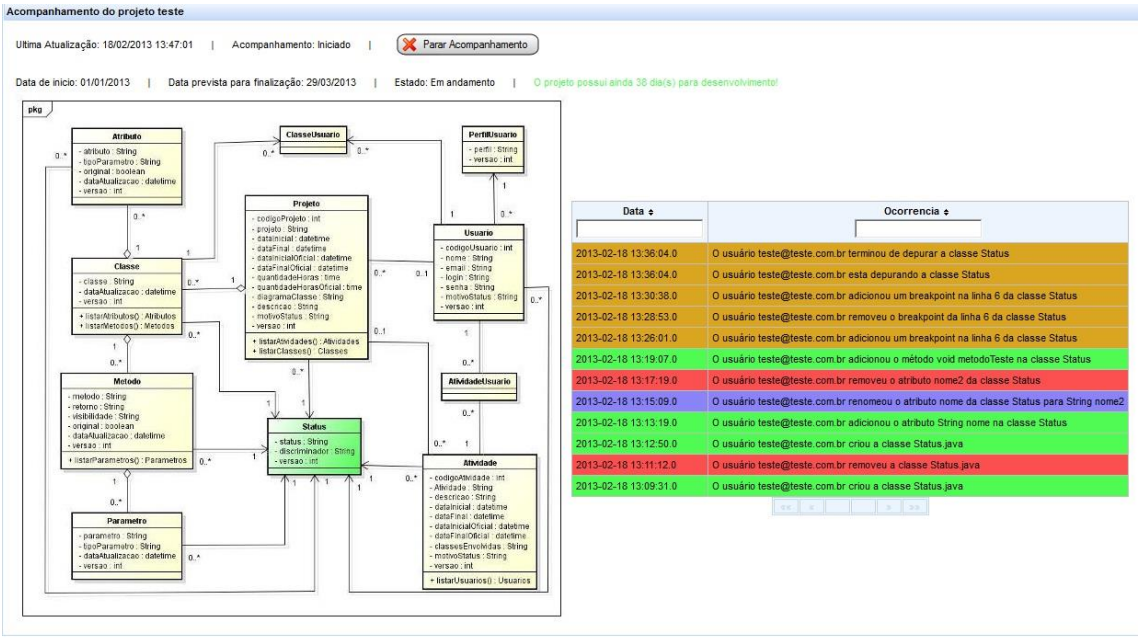
Figura 52 – Acompanhamento da criação da classe no OPERAM.

Fonte: própria.

15) O usuário **teste@teste.com.br** criou um novo projeto JAVA chamado de **teste3**dentro de sua workspace do Eclipse. Criou também uma classe chamada de **Status**, com atributos, métodos e executou algumas **depurações (DEBUG)**. Tanto na tela de acompanhamento do projeto **teste** e **teste2**, não deverá aparecer log algum, pois o projeto **teste3**não está cadastrado no **Módulo**.



16) Na tela de acompanhamento do projeto **teste**, todos os logs criados e situados ao lado direito da tela, deverão possuir a **data** em que cada ação ocorreu e o usuário **teste@teste.com.br** como usuário que efetuou as ações.



17) Na tela de acompanhamento do projeto **teste2**, a classe **Status** do diagrama de classes não deverá haver cor alguma e nenhum log deverá ser exibido, pois as ações ocorridas nos testes anteriores foram exclusivas do projeto **teste**.

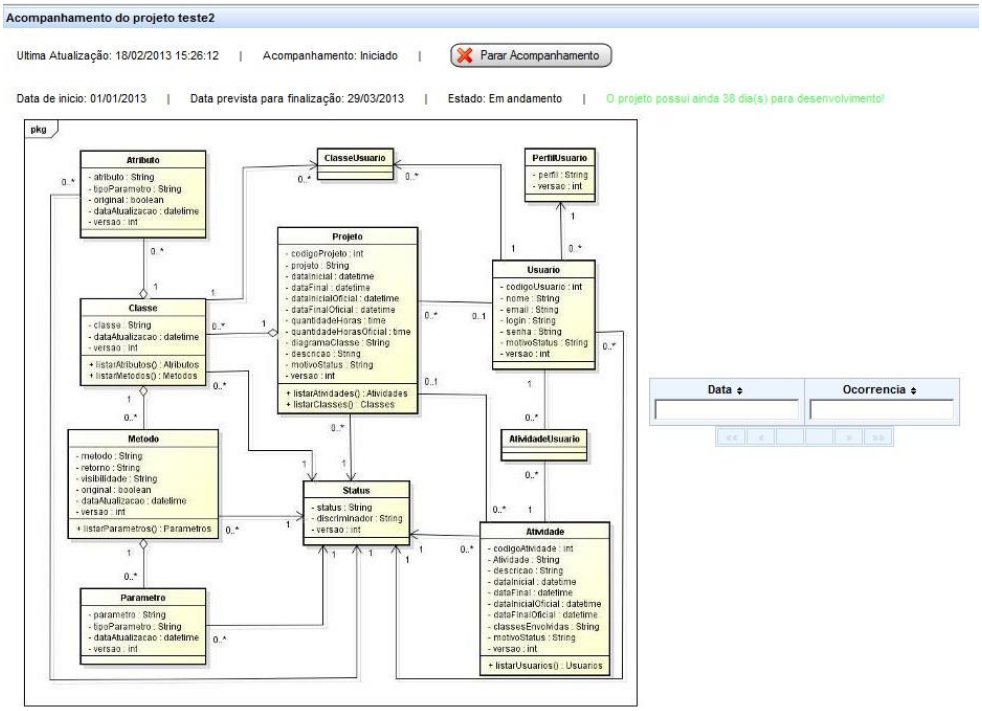


Figura 53 – Acompanhamento de teste no OPERAM.

Fonte: própria.

18) Dentro do projeto JAVA **teste**, na workspace do Eclipse, o desenvolvedor **teste2@teste.com.br** criou um método chamado de **metodoTeste2** na classe **Status**. Na tela de acompanhamento do projeto **teste** do **Módulo**, deverá aparecer um log na cor verde, indicando a criação do método na classe **Status**. Também deverá aparecer a **data** da ação ocorrida e o usuário **teste2@teste.com.br** como usuário que efetuou a ação.

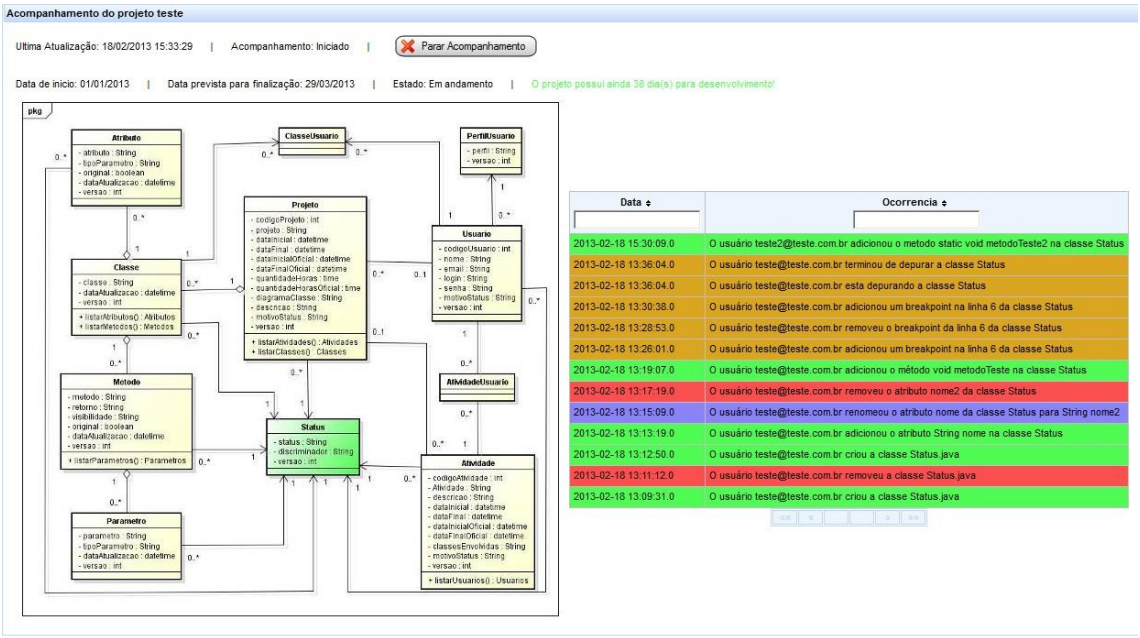


Figura 54 – Linha do tempo de um projeto.

Fonte: própria.

19) Executar o teste número 4 novamente

20) Dentro do projeto JAVA **teste2**, na workspace do Eclipse, o desenvolvedor **teste2@teste.com.br** criou uma classe chamada de **Status**. Na tela de acompanhamento do projeto **teste2**do **Módulo**, deverá a classe **Status** do diagrama de classes, situado ao lado esquerdo da tela, ficar verde e aparecer log na cor verde, situado a lado direito da tela, indicando a criação da classe. Também deverá aparecer a **data** da ação ocorrida e o usuário **teste2@teste.com.br** como usuário que efetuou a ação.

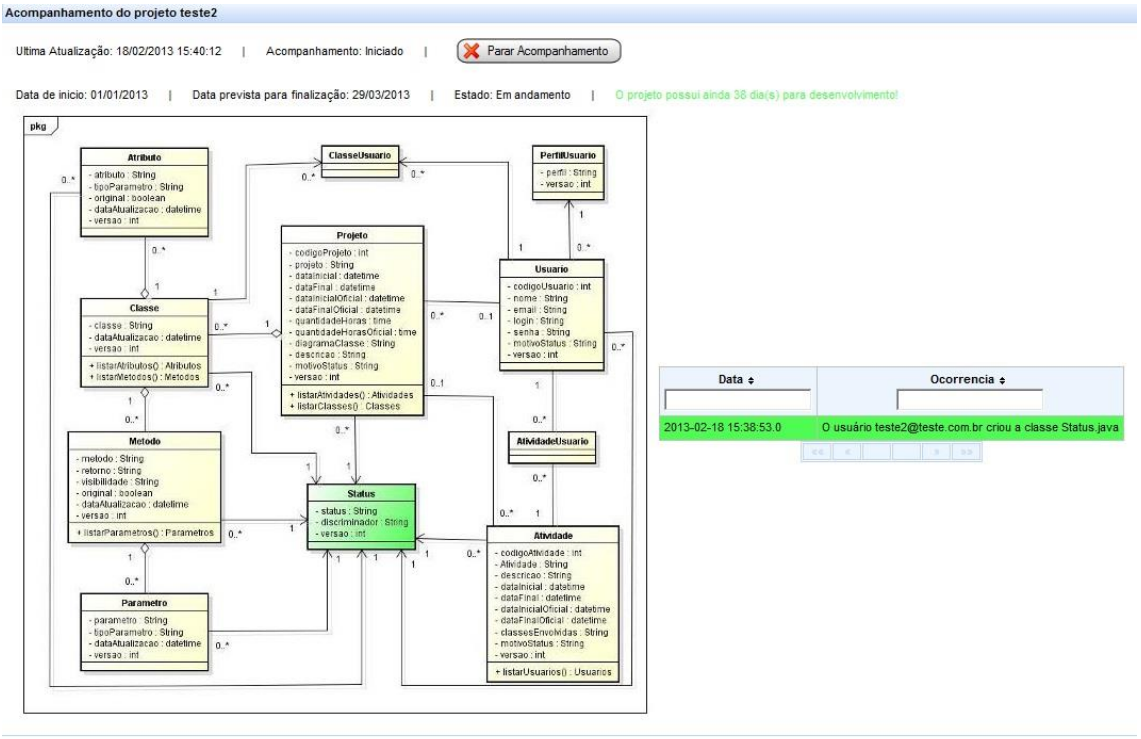


Figura 55 – Log na cor verde no OPERAM.

Fonte: própria.

21) Na tela de acompanhamento do projeto **teste**, a classe **Status** deverá permanecer na cor vermelha, pois a criação da criação da classe **Status** do teste **número 20**, ocorreu no projeto JAVA **teste2**.

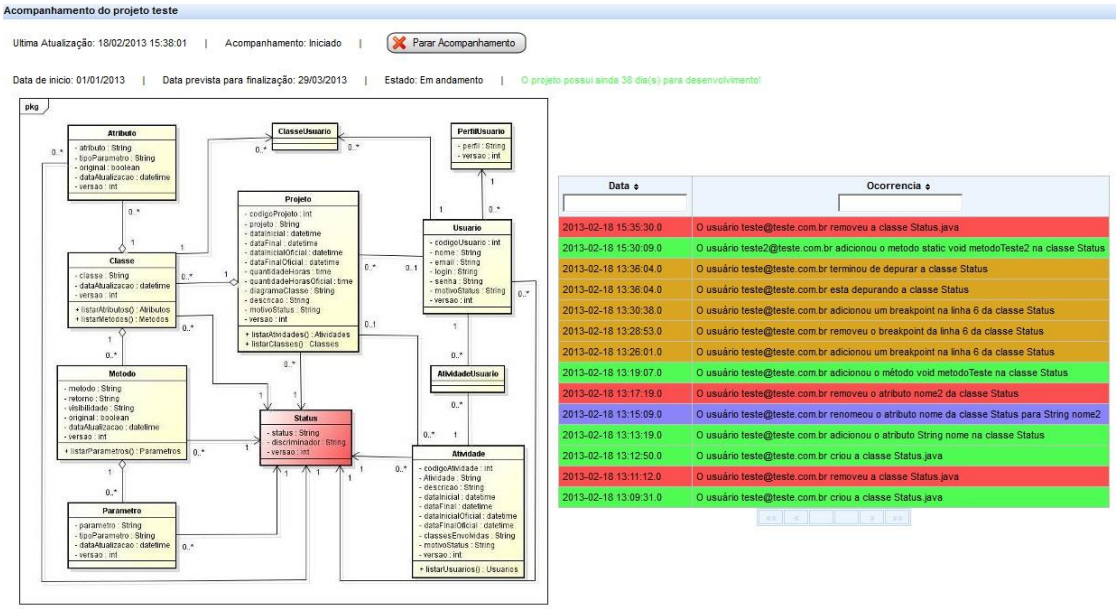


Figura 56 – Log de acompanhamento de usuário.

Fonte: própria.

APÊNDICE 4 – Testes de Avaliação do DeSS Nepomuk

Com a finalidade de verificar como o NEPOMUK poderia ser útil na utilização de um DS para gerenciamento de artefatos de equipes que desenvolvem software, alguns experimentos foram conduzidos e são descritos na sequência.

- Primeiro experimento: objetivo: verificar como o NEPOMUK funciona com os dados do usuário. Para isto, uma instalação do NEPOMUK foi providenciada e utilizada na execução de tarefas rotineiras que normalmente um usuário executa no seu dia-a-dia. Operações como ler arquivos de diversos formatos, acessar a internet, utilizar e-mail, copiar arquivos e outras operações foram realizadas.
- Segundo experimento: Acrescido ao experimento as funções de programação/análise, onde softwares de programação e de análise de sistemas foram instalados (JAVA, ASTAH COMMUNITY).
- Terceiro experimento: grupo de programadores utilizando o NEPOMUK em conjunto.
- Quarto experimento: utilizando a interligação dos DSs dos programadores com repositório de código fonte.

Para estes experimentos foram criadas também contas de *e-mail* em um serviço de webmail, para que a troca de mensagens, avisos de agendamento de tarefas e outras atividades dependentes de comunicação eletrônica pudessem ser efetivados em uma situação real de uso. Para auxiliar a execução destes testes, foi utilizada uma base com informações semânticas disponibilizada pelo consórcio NEPOMUK, que criou uma coleção de documentos da área de trabalho que contém vários itens: *e-mails*, contatos, itens de calendário e outros artefatos para quatro personagens fictícios que tem o objetivo se suprir a falta dados que pesquisadores enfrentam quando pretende realizar testes do DeSS NEPOMUK Com esta base, consegue-se obter dados para resultados experimentais reproduzíveis e comparáveis, justamente por utilizar um conjunto de teste comum e disponível à comunidade.

Dos quatro personagens disponibilizados, dois possuem um conjunto de dados mais completo e com relacionamentos semânticos entre os documentos de ambos. Eles personagens chamados de Claudia Stern e Dirk Hagemann, representando atores fictícios do DeSS NEPOMUK. O personagem representante de Claudia desempenha o papel de gerente de projeto e seus interesses são principalmente sobre ontologias, gestão do conhecimento e recuperação da informação. Sua área de trabalho contém 56 publicações sobre temas de interesse, 36 *e-mails*, 19 documentos do Word sobre reuniões e entregas do projeto, 12 apresentações de slides, 17 itens de calendário, 2 contatos, e um *log* de atividade recolhidos enquanto uma viagem estava sendo organizada (ou seja, reserva de vôo, reserva de hotéis, busca de locais de compras. Etc.). Este *log* contém um total de 122 ações. Já o personagem representando Dirk trabalha como subordinado no projeto que Claudia gerencia, e seus interesses são semelhantes aos dela. Sua área de trabalho contém 42 publicações, 9*e-mails*, 19 documentos do Word e 7 arquivos de texto, além do *log* de atividades compartilhado com Claudia.

O trabalho de (GUJÓNSDÓTTIR *e* LINDQUIST, 2008) apresenta a criação e utilização destes personagens, ajudando a entender o seu de uso e como elas são utilizadas dentro do projeto NEPOMUK*,* fornecendo ainda um detalhamento amplo a respeito do contexto onde estes personagens atuam, como também a forma que seus arquivos estão interligados. Outros trabalhos utilizaram estas mesmas bases para desenvolver suas pesquisas, tais como em (HALLER, 2008) e (SCERRI *et al.,* 2009).

Para o experimento proposto em nossa pesquisa, estas bases de informações semânticas disponibilizadas pelo consórcio NEPOMUK foi utilizada. Para isto, máquinas virtuais foram criadas utilizando o software livre *Oracle VM VirtualBox,* disponível em `http://www.oracle.com/technetwork/server-storage/virtualbox/downloads/index.html`. Este processo pode ser visto na figura 57. Uma instalação do *Nepomuk Server* e mais três instalações de máquinas virtuais simulando usuários foram criadas e configuradas com os seguintes softwares.

- Software de programação (Java, eclipse);
- Navegação internet;

- Pacote de aplicativos de escritório (editor de textos, editor de apresentações, planilha de cálculo);
- E-mail;
- ASTAH COMMUNITY.

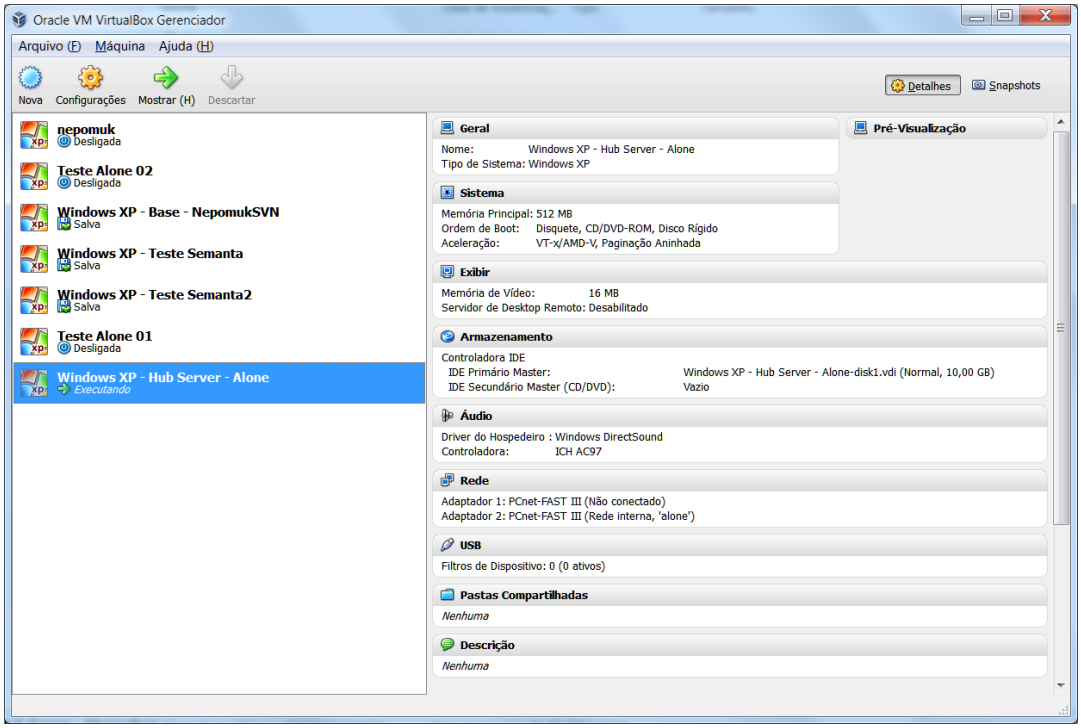


Figura 57 – A máquina virtual gerenciadora dos ambientes de teste.

Fonte: própria.

O funcionamento do NEPOMUK é sobre o P2P, e para que o intercâmbio das informações ocorresse de forma correta, um servidor do NEPOMUK foi instalado com as três outras máquinas virtuais acessando os serviços disponibilizados na máquina determinada a ser a servidora dos arquivos. Esta instalação é necessária para criar o ambiente para troca de mensagens entre as máquinas componentes da rede do NEPOMUK. Esta máquina central é chamada de NEPOMUKHUB é responsável pelo gerenciamento dos usuários e autenticação dos mesmos. O consórcio NEPOMUK disponibiliza no endereço eletrônico (http://dev.nepomuk.semanticdesktop.org/wiki/NepomukHub) as instruções para instalação do NEPOMUKHUB e inicialização de uma série de serviços necessários para o correto funcionamento do sistema. Os principais serviços que o

NEPOMUKHUB utiliza são para criar este ambiente são: o APACHE TOMCAT, o *framework* SESAME, o servidor de mensagens OPENFIRE e o protocolo de troca de mensagens JABBER.

Apesar de o consórcio NEPOMUK fornecer instruções para utilização destes serviços do NEPOMUKHUB, a instalação e configuração dos mesmos não é trivial, enfrentando uma série de problemas com versões de softwares e ajustes de componentes. Desta forma, foi criado um manual de instalação com todos os detalhes de configuração para que interessados em reproduzir o experimento, tenham condições de realizar isto de maneira funcional e semelhante ao que foi realizada para esta pesquisa. Este manual está disponível para download no formato *.docx* e *.pdf* no endereço eletrônico do programa PPGIa: (`http://www.ppgia.pucpr.br/~edenilson/ftp`).

RESULTADOS DO EXPERIMENTO DE AVALIAÇÃO DO DESS NEPOMUK

Para efeitos de comparação e análise das potencialidades e dos pontos de melhoria para utilização do DeSS NEPOMUK, foram consideradas as funcionalidades descritas em (REIF et al., 2007), e o abaixo apresenta um comparativo entre as funções que o DeSS NEPOMUK possui e aquelas que podem fazer parte do WS. Algumas destas funcionalidades são possíveis de utilização, mas podem necessitar de adaptações para serem completamente úteis no DS:

DeSS NEPOMUK e as possíveis utilizações no WS. Adaptado de (REIF et al., 2007)

Função	Funcionalidades	Descrição	DeSS	WS
Desktop	Anotação	Armazenar e recuperar relações semânticas sobre qualquer item da área de trabalho.	Presente	Possível utilização
	Acesso *offline*	Se DeSS de outra pessoa não estiver conectada à rede, o acesso *off-l*ine exportará as informações relevantes para outra área de trabalho.	Presente	Não aplicável
	Compartilhamento de Desktop	Capacidade na qual diferentes usuários do DeSS possam trabalhar nos mesmos recursos, até mesmo remotamente.	Presente	Possível utilização

	Gerenciamento de Recursos	Quando um novo conceito ou até mesmo um novo arquivo é criado ele é manipulado por este componente.	Presente	Não aplicável
	Integração de Aplicações,	Aplicações interagem com o DeSS por meio dele deste integrador.	Presente	Possível utilização
	Gerenciamento de Notificações	Centraliza as notificações aos usuários.	Presente	Não aplicável
Procura	Procura	Uso das relações semânticas bem como as relações sociais para recuperar itens relevantes.	Presente	Não aplicável
	Encontrar Itens Relacionados	Encontrar outros itens que possuem relacionamento entre si e descobrir informações correlatas e relevantes.	Presente	Possível utilização
Social	Interação Social,	Por meio de técnicas de comunicação síncrona e assíncrona.	Presente	Possível utilização
	Compartilhamento de Recursos	Modificar os direitos de acesso ou para partilhar recursos	Presente	Possível utilização
	Gerenciamento de Direitos de Acesso	Fornece maneiras de definir relações específicas de direitos entre os usuários, grupos e os recursos.	Presente	Possível utilização
	Publicação/ Subscrição	Permite a criação destas notificações de informações relevantes.	Presente	Possível utilização
	Gerenciamento de Grupos/Usuários	Facilita a criação de novos grupos a partir de uma lista de usuários.	Presente	Não aplicável
Perfis	Treinamento	Onde é previsto o comportamento do usuário	Presente	Possível utilização
	Adaptação	Contextos aprendidos podem ser irrelevantes, precisam ser readaptados ou até removidos.	Presente	Não aplicável
	Confiança Entre Usuários	Entre as pessoas ou fontes de informação	Presente	Possível utilização
	Registro	Registra a atividade do usuário, que irá ajudar a detectar o contexto em que o ele está trabalhando.	Presente	Possível utilização
Análise de Dados	Inferências	Inferências ou consultas nos tagueamentos.	Presente	Não aplicável
	Extração de Palavras Chave	Criar marcações automaticas ou sumarizar recursos.	Presente	Não aplicável
	Classificação e Agrupamento	Suporta aplicações que realizam pesquisa.	Presente	Possível utilização

Observou-se que todas as funções e funcionalidades descritas em (REIF *et al.,* 2007) estão presentes e fazem parte do DeSS NEPOMUK.