INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO TOCANTINS
CAMPUS PALMAS

HARLY CARREIRO VARÃO

ANÁLISE DO DESENVOLVIMENTO DISTRIBUÍDO DE SOFTWARE, ARQUITETURA DE SOFTWARE E PROCESSO DE CORREÇÃO DE BUGS DO SISTEMA INTEGRADO DE GESTÃO ACADÊMICA – SIGA-EDU

Palmas, TO
2011

HARLY CARREIRO VARÃO

ANÁLISE DO DESENVOLVIMENTO DISTRIBUÍDO DE SOFTWARE, ARQUITETURA DE SOFTWARE E PROCESSO DE CORREÇÃO DE BUGS DO SISTEMA INTEGRADO DE GESTÃO ACADÊMICA – SIGA-EDU

Trabalho de Conclusão de Curso apresentado à Coordenação da Área de Informática como requisito parcial para obtenção da conclusão do Curso Superior de Tecnologia em Sistemas para Internet no Instituto Federal de Educação, Ciência e Tecnologia do Tocantins – Campus Palmas.

Orientador: Prof. Msc. Francisco Willians Makoto Plácido Hirano.

Palmas, TO

2011

V287a VARÃO, Harly Carreiro

Análise do desenvolvimento distribuído de software, arquitetura de software e processo de correção de bugs do Sistema Integrado de Gestão Acadêmica – SIGA-EDU / Harly Carreiro Varão – – Palmas, 2011.

62f. : il.

Trabalho de Conclusão de Curso (Graduação Tecnológica em Sistemas para Internet) - Instituto Federal de Educação, Ciência e Tecnologia do Tocantins – Campus Palmas, 2011.

Orientador: Prof. Msc. Francisco Willians Makoto Plácido Hirano.

1. Arquitetura em camadas. 2. Desenvolvimento Distribuído de Software. 3. Processo de correção de Bugs. I. Instituto Federal de Educação, Ciência e Tecnologia do Tocantins – Campus Palmas. II. Harly Carreiro Varão. III. Título.

CDD 004

HARLY CARREIRO VARÃO

ANÁLISE DO DESENVOLVIMENTO DISTRIBUÍDO DE SOFTWARE, ARQUITETURA DE SOFTWARE E PROCESSO DE CORREÇÃO DE BUGS DO SISTEMA INTEGRADO DE GESTÃO ACADÊMICA – SIGA-EDU

Este Trabalho de Conclusão de Curso foi julgado e aprovado como cumprimento às exigências legais do currículo do Curso Superior de Tecnologia em Sistemas para Internet pela Coordenação da Área de Informática do Instituto Federal de Educação, Ciência e Tecnologia do Tocantins – Campus Palmas.

Aprovado em _____/_____/_____

BANCA EXAMINADORA

_____

Prof. Msc. Francisco Willians Makoto Plácido Hirano
Orientador

_____

Prof. Esp. Marinaldo Oliveira Santos
Membro da Banca

_____

Prof. Esp. Vinícius Istofel Oliveira
Membro da Banca

Em memória de Maria das Mercedes Carreiro Varão e
Gustavo Carreiro Varão.

AGRADECIMENTOS

A gratidão é uma das mais importantes virtudes que o ser humano pode ter. Como disse certa vez o pensador Muslah-Al-Din Saadi, o coração ingrato assemelha-se ao deserto que sorve com avidez a água do céu e não produz coisa alguma. Como este trabalho é fruto de alguma produção, consequentemente tenho a quem agradecer.

Agradeço primeiramente ao Deus que sirvo e que tem me colocado a cada dia em lugar de honra, me dado prosperidade, mansidão e que tem me acolhido nos momentos difíceis e alegres. Ele é meu refúgio e fonte de esperança nos momentos de tristeza e alvo de gratidão em momentos de alegria.

Ao meu pai nesta terra que me deu forças das quais sempre necessitei e que acordava às quatro da manhã todos os dias pra preparar o café da manhã e a marmita para o almoço além de fazer a janta à noite, na maioria das vezes após a meia noite. Agradeço pela preocupação, por demonstrar tanto amor ao não dormir enquanto eu não chegasse em casa e pela paciência nos momentos em que não estava bem. Foi um presente de Deus na minha vida e sei que este sonho de estar concluindo este curso superior também é um sonho dele.

Agradeço à minha irmã Maria que conviveu comigo durante todos estes anos durante este curso e me ajudou bastante.

Aos meus amigos Wellington, Edgar, Horiano, Ulisses e Fernando que sempre foram grandes influenciadores no que sou e parceiros nos momentos de comunhão.

Aos meus colegas e chefes do Tribunal de Justiça que foram compreensivos em relação às ausências devido à conclusão deste trabalho.

E finalmente aos professores que fizeram isso possível: Francisco Willians (orientador), Marinaldo, Vinicius Istofel, Fernando Ebrahim, Simone Dutra e Gedalias. E a todos que contribuíram para que este trabalho e objetivo fossem realizados.

RESUMO

Este trabalho tem a finalidade de analisar o Sistema Integrado de Gestão Acadêmica – SIGA-EDU, no que se refere a aplicação da metodologia de desenvolvimento distribuído de software, arquitetura de software e processo de correção de bugs. Para isso, utiliza-se de conceitos da literatura sobre o assunto e realiza-se um comparativo entre estes conceitos e sua aplicação prática no SIGA-EDU considerando os desafios da gestão colaborativa.

Palavras-Chave: Desenvolvimento Distribuído de Software. Arquitetura em Camadas. Processo de Correção de Bugs.

ABSTRACT

This paper aims to analyze the *Sistema Integrado de Gestão Acadêmica – SIGA-EDU*, as regards the application of the methodology for developing distributed software, development architecture and bugs correction process. For this, we use concepts from the literature on the subject and carried out a comparison between these concepts and their practical application in SIGA-EDU considering the challenges of collaborative management.

Keywords: Distributed Software Development. Layered Architecture. Bugs Correction Process.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API - *Application Programming Interface* (Interface de Programação de Aplicações)

CDU – Caso de Uso

DAO – *Data Access Object* (Objeto de Acesso a Dados)

DDS – Desenvolvimento Distribuído de Software

DGS – Desenvolvimento Global de Software

EJB – *Enterprise JavaBeans*

EPCT – Educação Profissional, Científica e Tecnológica

IDE – Integrated Development Environment (Ambiente Integrado de Desenvolvimento)

JDK – *Java Development Kit* (Kit de Desenvolvimento Java)

JPA – *Java Persistece API* (API de Persistência para Java)

JRE – *Java Runtime Environment* (Ambiente de Tempo de Execução Java)

JSF – *JavaServer Faces*

JVM – *Java Virtual Machine* (Máquina Virtual Java)

PT – Projeto de Tela

RUP – *Rational Unified Process* (Processo Unificado)

SIGA – Sistema Integrado de Gestão Acadêmica

UML – *Unified Modeling Language* (Linguagem de Modelagem Unificada)

SVN – *Subversion*

WEB – *World Wide Web*

APRESENTAÇÃO

Tema do trabalho

Análise do desenvolvimento distribuído de software, arquitetura de software e processo de correção de bugs do Sistema Integrado de Gestão Acadêmica – SIGA-EDU.

Aluno

Harly Carreiro Varão.

Orientador

Prof. Msc. Francisco Willians Makoto Plácido Hirano.

Área do Conhecimento

Do Curso

O Curso Superior de Tecnologia em Sistemas Para Internet do IFTO Campus Palmas é uma graduação tecnológica pertencente ao eixo tecnológico "Informação e Comunicação".

Do CNPq

Ciências Exatas e da Terra.

Área

Ciência da Computação.

Subárea

Sistemas de Informação.

Tipo de Pesquisa

Acadêmica e bibliográfica.

Local da Pesquisa

Instituto Federal de Educação Ciência e Tecnologia – Laboratório do projeto SIGA-EPCT - núcleo TO.

SUMÁRIO

1 INTRODUÇÃO

O projeto SIGA-EPCT (Sistema Integrado de Gestão Acadêmica – Educação Profissional, Científica e Tecnológica), é um projeto desenvolvido com tecnologias livres e de maneira distribuída – ou colaborativa – por vários núcleos de instituições federais do Brasil. O SIGA-EPCT, ou apenas SIGA, é patrocinado pela Rede Nacional de Pesquisa e Inovação em Tecnologias Digitais (RENAPI), com o apoio do Ministério da Educação do Brasil (MEC), por meio da Secretaria da Educação Profissional e Tecnológica.

Com o objetivo de beneficiar as instituições federais de ensino em relação ao apoio à sua gestão e dispor o MEC da prestação de informações, o SIGA visa automatizar a gestão dos processos institucionais acadêmicos por meio do SIGA-EDU (Ensino, Pesquisa e Extensão) e administrativos (Protocolo, Recursos Humanos, Almoxarifado, Compras, Patrimônio etc.) por meio do SIGA-ADM.

Neste trabalho foi feita uma análise do uso da metodologia de desenvolvimento distribuído de software, da arquitetura de software e do processo de correção de bugs do SIGA-EDU, sendo feito um levantamento preliminar da documentação das tecnologias utilizadas, tais como: a linguagem de programação Java, os frameworks JSF e *Eclipselink*, a biblioteca de componentes *RichFaces*, o banco de dados *PostgreSQL*, o ambiente de desenvolvimento integrado Eclipse, o gerenciador de projetos *Redmine*, entre outras.

Foram utilizadas ainda, formas de comparação das metodologias e arquiteturas utilizadas do SIGA-EDU com os conceitos encontrados nas bibliografias. Todas as análises consideram os desafios impostos pela metodologia de desenvolvimento distribuído visando um melhor entendimento de que alguns aspectos necessitam ser considerados e estudados de forma diferente das metodologias de desenvolvimento tradicional – ou co-localizada.

O objetivo geral deste trabalho é demonstrar e analisar a metodologia de desenvolvimento distribuído e suas adaptações, a arquitetura de software e o processo de correção de bugs do Sistema Integrado de Gestão Acadêmica – SIGA-EDU de tal forma que seja realizado um comparativo da metodologia utilizada no

SIGA-EDU com os conceitos bibliográficos existentes sobre os temas abordados. Além de levar em conta as dificuldades da gestão distribuída e o fato de que existem empecilhos neste processo.

Os objetivos específicos podem ser enumerados da seguinte forma: detalhar as tecnologias utilizadas em cada camada da arquitetura de software, analisar a metodologia de desenvolvimento distribuído de software e analisar de demonstrar os resultados do processo de correção de bugs do projeto SIGA-EDU.

De acordo com Prikladnicki & Audy (2008, p. 65), o desenvolvimento distribuído de software é recente e ainda não tem sido abordado em universidades e demais instituições de ensino do país e do mundo, e isso acaba contribuindo para que as empresas se tornem as responsáveis pela formação complementar do profissional nesta área. Tal análise justifica-se pela relevância do tema abordado, pela sua aplicabilidade em um ambiente institucional e de grande porte; pela contribuição técnica relacionada ao assunto e contribui como difusor do conhecimento relacionado ao desenvolvimento colaborativo de software no meio acadêmico.

Nesse contexto podem-se levantar alguns questionamentos em relação à prática das metodologias e tecnologias diante do desenvolvimento distribuído: Como devem ser definidas as estratégias de gerência e controle? O que deve ser feito para que a comunicação não seja escassa entre os envolvidos no projeto? Quais as tecnologias utilizadas no SIGA-EDU e como elas contribuem para que o ambiente colaborativo seja eficaz? Quais as adaptações foram realizadas no SIGA-EDU para que o processo ocorra continuamente? De fato são vários questionamentos e eles são abordados durante este trabalho.

Portanto, o intuito é demonstrar como ocorre todo o processo para a construção de um sistema de grande porte face às tecnologias livres e a metodologia distribuída de software, levando em conta as dificuldades e causas que levam à aparição de bugs no sistema e à eventual correção dos mesmos, além de demostrar resultados das correções de bugs e classifica-los de acordo com a sua natureza de origem.

2 ARQUITETURA DE SOFTWARE

O SIGA está sendo desenvolvido totalmente com tecnologias e soluções livres, ou seja, sem custo para sua aquisição e manutenção no que se refere a software. Tais tecnologias são fornecidas aos seus usuários – no caso os colaboradores do SIGA-EDU – com a liberdade de executar, estudar, modificar e repassar (com ou sem alterações) sem que, para isso, os usuários tenham que pedir permissão ao autor do programa ou tecnologia.

Na computação, o conceito de software livre é de grande importância. De certo que, um software é considerado livre quando seu código-fonte está para que todos possam alterá-lo e/ou adequá-lo às suas necessidades, sem ter que usar de finanças para isso.

Neste capítulo foram relatadas quais são e como são utilizadas estas tecnologias dentro do projeto SIGA-EDU, de forma que foram abordadas as vantagens em utilizar destas tecnologias.

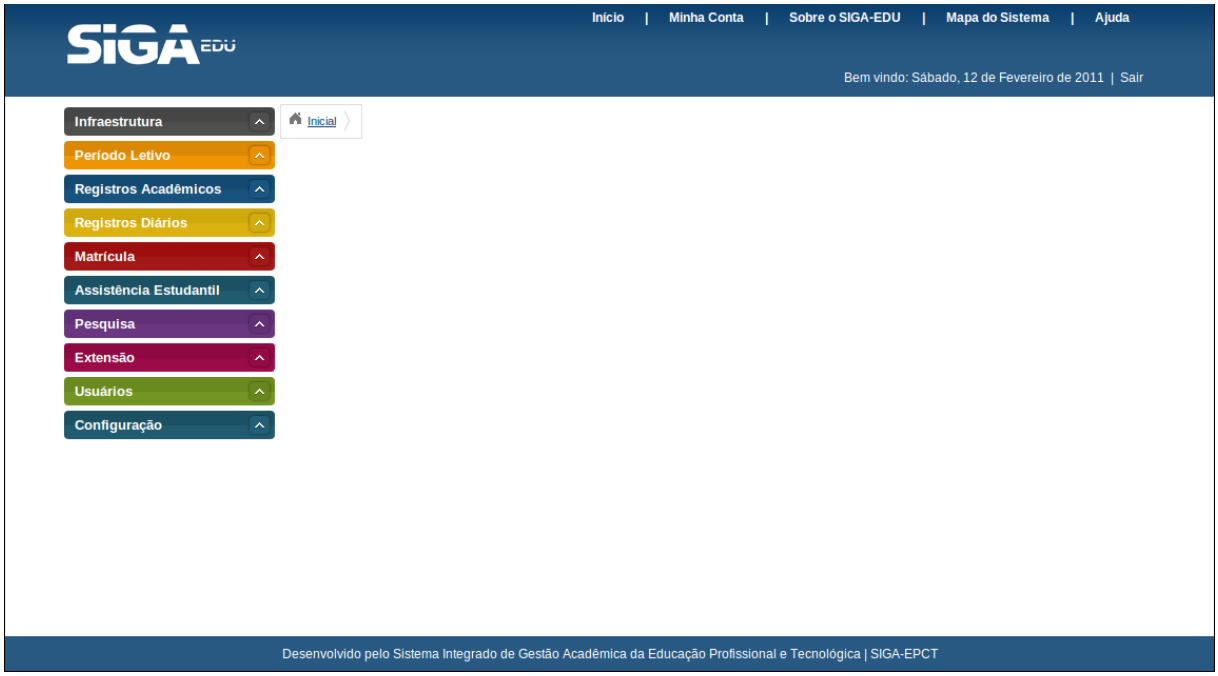


Figura 1: Página inicial do SIGA-EDU.
Fonte: SIGA-EDU, 2011.

2.1 Abordagem inicial

No projeto SIGA-EDU estas tecnologias são aplicadas por meio do padrão arquitetural com uso de camadas, que tem a função de separar as responsabilidades e encapsular as funcionalidades de um conjunto de classes que têm comportamento similar, acarretando em baixo acoplamento, que se refere à interconexão de módulos em uma estrutura de software; e alto índice de coesão em um sistema, que diz respeito ao fato de um módulo realizar uma (e apenas uma) tarefa funcional bem definida.

No SIGA-EDU as camadas que se destacam são: a de apresentação, onde se encontra a interface com o usuário, ou seja, a parte visual do sistema do ponto de vista do usuário final; a camada de negócio onde estão as classes que fazem parte do domínio; e a camada de persistência, que se refere ao armazenamento e recuperação das informações em um meio específico. Essas camadas são detalhadas de acordo com a sua estrutura no projeto levando em consideração os padrões de desenvolvimento utilizados no SIGA-EDU apresentados nos próximos capítulos.

2.2 Camada de Apresentação

Uma das boas práticas de engenharia de software do RUP (*Rational Unified Process* ou Processo Unificado), no qual o SIGA-EDU se baseia para ter seu processo de desenvolvimento e herda algumas características, é a arquitetura baseada em componentes, onde são aplicados os conceitos de reusabilidade, independência, substituição, encapsulamento de componentes, entre outros. A camada de apresentação ou interface agrupa as classes que têm por propósito implementar interação como o usuário e sua grande vantagem encontra-se no fato de que se alterada não afeta a camada de negócios.

O JSF (*JavaServer Faces*) é o framework para desenvolvimento WEB utilizado no SIGA-EDU, é baseado em componentes e visa tornar o desenvolvimento da aplicação mais fácil e produtivo.

No projeto também é utilizada a *RichFaces*, que é uma biblioteca de componentes com grande suporte a Ajax que é usada em conjunto com o JSF.

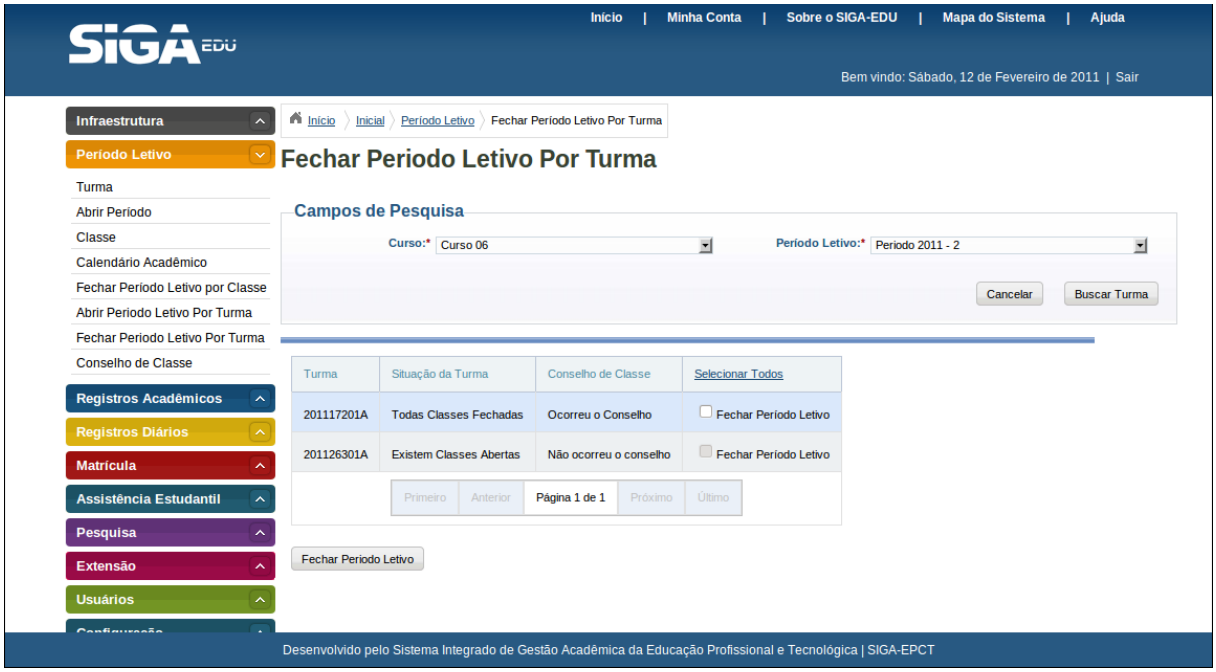


Figura 2: Exemplo de tela com *dataTable*, um dos componentes da *Richfaces*.
Fonte: SIGA-EDU, 2011.

De acordo com Mann (2005, p. 09, tradução nossa), "*JavaServer Faces* (ou simplesmente JSF) torna o desenvolvimento web mais fácil por fornecer suporte a componentes de interface com usuário e lidar com uma série de tarefas comuns ao desenvolvimento WEB".

No caso do SIGA-EDU a codificação visual é feita em um arquivo "*xhtml*" que deve seguir os *templates* pré-definidos além das padronizações de nomes e variáveis adotados para facilitar o entendimento do código-fonte.

2.3 Camada de Negócios

Nesta camada estão localizadas todas as classes que fazem parte do domínio, ou seja, as classes correspondentes aos objetos do artefato. Nela são definidas e localizadas as regras de negócio que ditam o comportamento, restrições e validação do sistema.

Esta camada é responsável pela lógica de negócios onde é seguido o padrão EJB3 (*Enterprise JavaBeans 3.0*) que tem o objetivo de facilitar o processo de desenvolvimento agrupando as regras de negócio.

No SIGA-EDU são usados EJBs do tipo *Session Beans*, que têm a função de executar uma determinada tarefa para a entidade. Esses *Session Beans* são a implementação das interfaces locais. Cada entidade possui seu EJB e os acessa por meio da interface local.

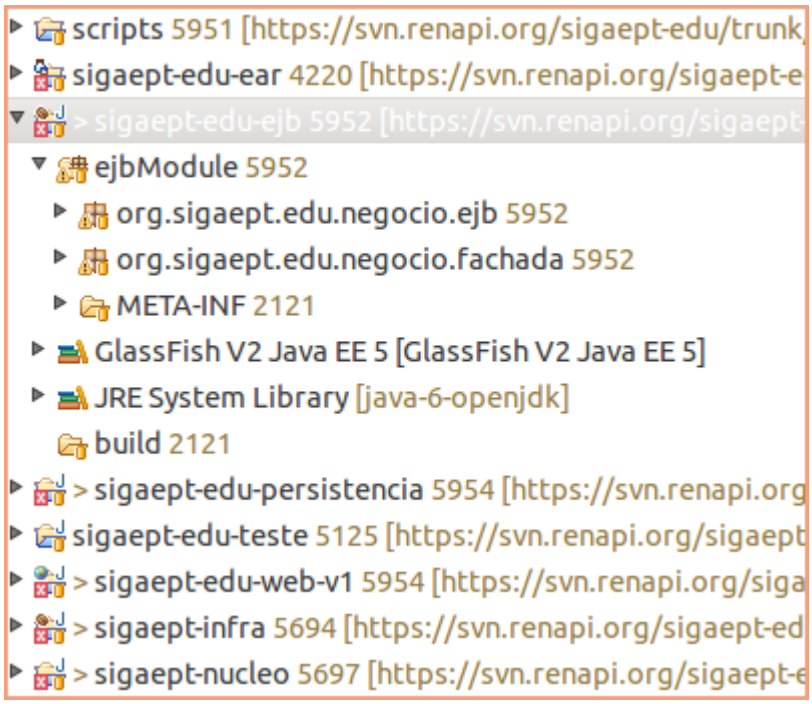


Figura 3: Camada de negócios com detalhes do pacote EJB e Fachada, este último contém as interfaces.
Fonte: SIGA-EDU, 2011.

A linguagem de programação Java é a adotada pelo SIGA-EDU, esta necessita do kit de desenvolvimento para aplicações Java, o JDK (*Java Development Kit*), o qual inclui o JRE (*Java Runtime Environment*) responsável por executar as aplicações desenvolvidas em Java. O JRE conta ainda com a JVM (*Java Virtual Machine*) que garante à linguagem Java seu principal atrativo: ser multiplataforma.

> O Java é uma poderosa linguagem de programação de computador [...] adequada para programadores experientes utilizarem na construção de sistemas de informações importantes. [...] O Java, desenvolvido pela Sun Microsystems, é hoje uma das linguagens de desenvolvimento de software mais populares. (DEITEL, 2006, p. 02)

Toda a codificação é feita visando à padronização de código-fonte, com a finalidade de facilitar a compreensão por parte dos colaboradores do projeto e consequentemente a fácil manutenção do sistema.

2.4 Camada de Persistência

Esta camada responde pelo acesso a dados e tem a função de fornecer serviços capazes de armazenar os dados e recuperá-los a qualquer momento. Ela responde às requisições da camada de negócios e as executa em um banco de dados, normalmente por meio de um framework de persistência.

O sistema gerenciador de banco de dados do projeto SIGA-EDU é o *PostgreSQL* que é um banco de dados de nível corporativo e que suporta grandes demandas. De acordo com o site oficial da ferramenta, o *PostgreSQL* é considerado o banco de dados de código aberto mais avançado do mundo.

Figura 4: Logomarca do *PostgreSQL*.
Fonte: Site oficial *PostgreSQL*, 2011.

Para acessar o banco de dados o SIGA-EDU utiliza o framework de persistência *Eclipselink* juntamente com a JPA (*Java Persistece API*) - que tem a função de criar, remover e consultar objetos Java simples - para o mapeamento objeto-relacional das entidades, pois provê alto desempenho e produtividade ao desenvolvedor.

No SIGA-EDU, a persistência primeira camada implementada durante a codificação e utiliza de anotações – ou metadados – para agrupar todos os métodos de acesso a dados definidos pelos DAOs (*Data Access Object*). Na maioria dos casos cada entidade tem um DAO correspondente, por exemplo. A entidade/classe *PeriodoLetivo* tem o acesso a dados definido pelo *PeriodoLetivoDAO.* Mas podem ocorrer casos que não necessite da criação de um DAO específico, como por exemplo, na classe *FecharPeriodoLetivoPorTurma.*

2.5 Outras Tecnologias

Servidor de aplicação é um software que tem a função de disponibilizar um ambiente propício à execução do sistema e oferecer recursos ao desenvolvedor, como: pool de conexão, segurança, controle de transações, entre outros. O projeto tem como servidor de aplicação o *Glassfish*, um programa de nível corporativo e que garante desempenho, facilidade, confiabilidade e produtividade.

O sistema operacional padrão do projeto é o GNU/Linux e a distribuição adotada é a Ubuntu. Porém, qualquer versão do Linux baseada em Debian Linux – um sistema operacional classificado como livre, monolítico, multitarefa e multiusuário – pode ser utilizada no desenvolvimento do SIGA-EDU.

O SIGA-EDU é desenvolvido no ambiente Eclipse que é um IDE (*Integrated Development Environment* ou Ambiente Integrado de Desenvolvimento) de código aberto para o desenvolvimento de softwares.

O gerenciamento do projeto é colaborativo feito por meio do *Redmine* que dispõe de uma série de funcionalidades, tais como: fóruns, controle de tarefas, acompanhamento de atividade, relatórios, alertas via e-mail, *wiki*, entre outras servindo então de repositório de informações. O SIGA-EDU tem seu próprio modelo de gestão colaborativa onde existem os seguintes papéis: um gerente de projeto, cinco coordenadores de área, grupos que executam atividades específicas e os núcleos que são compostos pelo gestor, orientadores e desenvolvedores. Mais detalhes sobre os papéis serão dados no capítulo 3.

O *Redmine* foi personalizado especialmente para que atendesse às necessidades desse projeto e que pudesse gerenciar todas estas equipes que formam hoje os cerca de 130 membros do projeto.

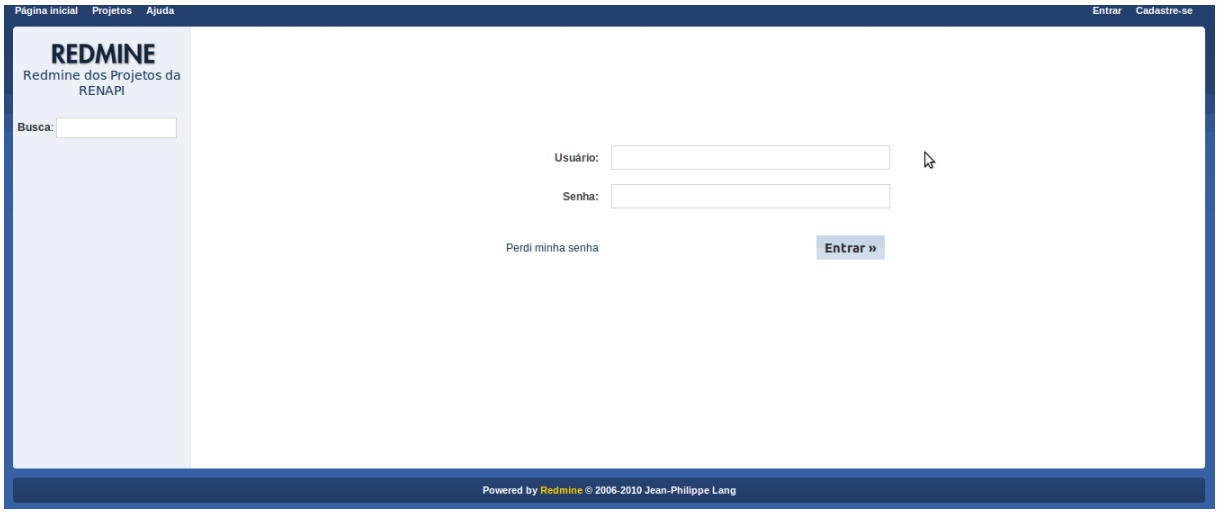


Figura 5: Página inicial do *Redmine*.
Fonte: *Redmine* do SIGA-EDU, 2011.

Para que todos esses membros acessem o projeto de forma organizada e sem que haja perda de trabalho ou problemas ao acessar um documento, o SIGA-EDU tem como padrão para controle de versões o *Subversion*, ou simplesmente SVN. A principal função do SVN é guardar todo o histórico de desenvolvimento dos artefatos (documentos, códigos-fonte, diagramas, entre outros) desde a sua criação até sua última versão. Isso garante que seja possível recuperar uma determinada versão de qualquer data passada, evitando perda de tempo no desenvolvimento para desfazer alterações quando se toma algum rumo equivocado. Os artefatos ficam armazenados em repositórios e são acessados por meio do SVN (figura 6).

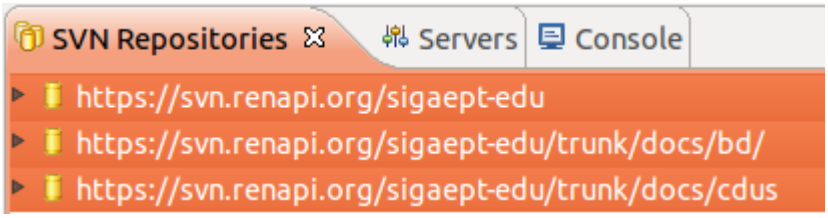


Figura 6: Repositórios do *Subversion* no SIGA-EDU.
Fonte: SIGA-EDU, 2011.

3 METODOLOGIA

3.1 Abordagem Inicial

Sabe-se que o software, com o passar do tempo, se tornou um componente vital nos negócios. A cada dia as organizações estão mais dependentes da utilização deste elemento como diferencial competitivo. Nesta crescente, tanto clientes como fornecedores desejam que os softwares produzidos atendam às suas necessidades de tal forma que venha realizar aquilo que foi solicitado dentro do prazo determinado e sem exceder os custos previstos.

Visando reduzir os problemas no processo de desenvolvimento de software em relação ao custo-benefício, foi instituída e Engenharia de Software (ES). Segundo Prikladnicki & Audy (2008, p. 09), pode-se entender a engenharia de software como um conjunto de disciplinas que incluem a especificação, o desenvolvimento, o gerenciamento e a evolução de sistemas de software. Outra definição para o termo seria:

> A engenharia de software é uma disciplina da engenharia que se ocupa de todos os aspectos da produção de software, desde os estágios iniciais de especificação do sistema até a manutenção desse sistema, depois que ele entrou em operação. (SOMMERVILLE, 2007)

Existem várias definições diferentes na literatura, mas todos têm um ponto em comum: a necessidade existente para que haja um maior rigor no desenvolvimento de software.

Pressman (2004) define a engenharia de software em três camadas compostas de três elementos primordiais: ferramentas, métodos e processo. Cada elemento representa uma camada e todas se baseiam no foco na qualidade. Isso quer dizer que os elementos primordiais devem contribuir para que o gerente controle o desenvolvimento de software e os desenvolvedores tenham uma base para produzirem um produto de qualidade.

Figura 7: As camadas da Engenharia de Software.
Fonte: Pressman, 2004.

Abaixo uma descrição dos elementos/camadas representados na figura 7:

- **Métodos:** são tópicos estruturados para o desenvolvimento de software que têm a função de definir "como fazer" para desenvolver um software de boa qualidade. Os métodos envolvem: planejamento, análise de requisitos, projeto, arquitetura, codificação, teste, entre outros;

- **Ferramentas:** exercem uma função de apoio automatizado ou semi-automatizado aos métodos. São formas de facilitar todo o processo de desenvolvimento;

- **Processo:** trata-se da camada mais importante, por justamente constituir o elo entre as ferramentas e os métodos. É no processo que se encontram os pontos mais importantes da engenharia de software. De acordo com Prikladnicki & Audy (2008, p. 11), um processo define a sequência em que os métodos serão aplicados, como os produtos serão entregues, os controles que ajudam a assegurar a qualidade e a coordenar as mudanças, e os pontos de referência que possibilitam aos gerentes de software avaliar o progresso do desenvolvimento.

Em relação ao processo, Sommerville (2007) destaca que há quatro atividades de processos fundamentais, que são encontradas em todos os processos de software: especificação, desenvolvimento, validação e evolução de software.

O processo de desenvolvimento de software é identificado por um modelo e esse modelo é gerenciado por meio de uma metodologia. O modelo de desenvolvimento define de que forma será a execução do processo, este por sua vez segue uma metodologia que basicamente tem a função de sequenciar as atividades e definir como elas relacionam entre si identificando quando os métodos e ferramentas serão utilizados.

O SIGA-EDU é um projeto de grande porte, e como tal, utiliza dos conceitos relacionados acima. O conceito de colaboração/distribuição, tanto na gestão como no processo de desenvolvimento demanda atenção redobrada e escolha de métodos, metodologias, processos, tecnologias e ferramentas que se adaptem melhor à situação. A seguir serão abordados os tópicos referentes ao modelo, metodologia e processo adotados no SIGA-EDU.

3.2 Modelo de Ciclo de Vida

O modelo de clico de vida tem um papel essencial no processo de desenvolvimento de software. Tanto na gerência, possibilitando que o gerente de projeto controle o todo o processo; como no desenvolvimento, permitindo que a equipe tenha uma base sólida para produzir de forma eficiente e eficaz um software que atenda aos requisitos elencados.

O SIGA-EDU trabalha com o modelo incremental – também chamado de prototipação – que abrange algumas atividades: coleta de requisitos, que da mesma forma que acontece em todos os ciclos de vida de software, define o que deve ser feito; elaboração de um projeto, que deve ser rápido e conter os aspectos visíveis ao cliente e deve levar à construção de um protótipo – no SIGA-EDU são representados pelo projeto de tela (PT) e pelo “demo” (protótipo) respectivamente; e avaliação do protótipo, na qual servirá de refinamento dos requisitos de software em desenvolvimento. No SIGA-EDU já foram lançadas 6 versões que são identificadas por número e nome: 1.0 - Fênix, 2.0 - Daros, 3.0 – Pegasus, 4.0 – Khronos, 5.0 – Piatã e 6.0 – Prope. Durante cada versão podem ser lançadas subversões incrementais devido às mudanças de requisitos ou correções de bug. Normalmente as versões são lançadas a cada três meses e as subversões não possuem um

cronograma e são lançadas quando necessário. Na figura 8 é demostrado o modelo desse ciclo de vida.

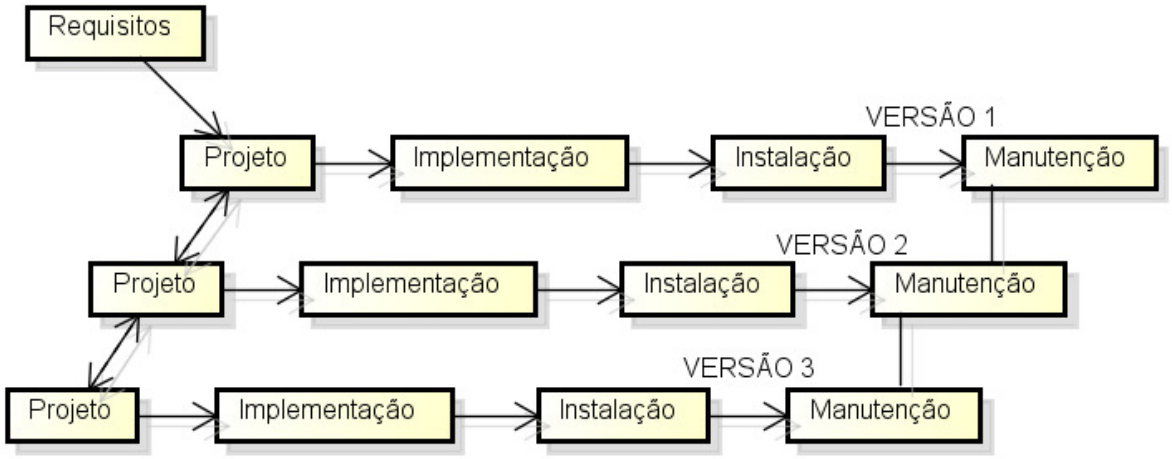


Figura 8: Modelo de Ciclo de Vida Incremental.
Fonte: Prikladnicki & Audy, 2008.

3.3 Processo de Desenvolvimento de Software

Prikladnicki & Audy (2008, p. 19) afirmam que a essência da utilização de processos é a resolução de problemas através da criação e instanciação de descrições de processo. Em relação ao processo de software, que tem a função de definir o conjunto de tarefas, métodos e práticas que dirigem o desenvolvimento do software, Pressman (2004) o define como o elo entre as camadas tecnológicas e causador do desenvolvimento racional e em tempo de software.

O SIGA-EDU, utiliza um processo de desenvolvimento que herda características inerentes ao RUP, uma metodologia iterativa que basicamente gerencia o projeto de software usando UML para especificação dos artefatos. Em tempo com o modelo de ciclo de vida incremental o RUP é iterativo e organiza os projetos em termos de disciplinas e fases, onde cada uma consiste em uma ou mais iterações.

Um processo pode ser descrito da seguinte forma: “quem” está fazendo “o quê”, “como” e “quando”. Esses pronomes são representados no RUP como os elementos primários:

- **Papéis:** é a definição da situação de um membro do projeto, ou seja, *quem* atua em uma determinada função;

- **Atividades:** determina *como* deve ser feito algum processo. É uma unidade de trabalho atuando com um determinado papel;

- **Artefatos:** é *o que* deve ser feito. Uma informação que é produzida, modificada ou utilizada no processo;

- ***Workflows*:** é uma das etapas do processo de desenvolvimento. Define *quando* acontece cada processo dentro do ciclo de desenvolvimento.

O processo de desenvolvimento do SIGA-EDU sofreu algumas adaptações em relação ao RUP, porém, os *workflows* mais importantes foram mantidos e aperfeiçoados para o modelo de desenvolvimento distribuído de software.

O RUP tem nove *workflows* que dividem as atividades em grupos: modelagem de negócios, requisitos, análise e projeto, implementação, teste, implantação, gerência de configuração e alteração, gerência de projeto e ambiente. Na Tabela 1, pode ser observado um comparativo entre o RUP e o modelo descendente do RUP utilizado no SIGA-EDU, os *workflows* não estão necessariamente na ordem que se sucedem, todo o fluxo de desenvolvimento pode ser observado no Anexo 1.

RUP	Metodologia SIGA-EDU
MODELAGEM DE NEGÓCIOS	PLANEJAMENTO
REQUISITOS	REQUISITOS
ANÁLISE E PROJETO	ESPECIFICAÇÃO
	MODELAGEM E BD
—	INTERFACE GRÁFICA
IMPLEMENTAÇÃO	CODIFICAÇÃO
TESTE	TESTES
IMPLANTAÇÃO	PRODUTO
GERÊNCIA DE CONFIGURAÇÃO E ALTERAÇÃO	—[1]
GERÊNCIA DE PROJETO	—[2]
—	QUALI-EPT
AMBIENTE	INFRAESTRUTURA[3]

Tabela 1: Comparativo entre o RUP e a metodologia de desenvolvimento do SIGA-EDU.

[1] Não consta no fluxo do processo de desenvolvimento do SIGA-EDU (Anexo1), mas é observado no processo de codificação.

[2] Apesar de não constar explicitamente no fluxo principal (Anexo 1), este *workflow* é presente no processo de desenvolvimento do SIGA-EDU e suas atividades são realizadas pelo gerente do projeto.

[3] Também não consta no fluxo principal (Anexo 1), mas corresponde em termos com o *workflow* AMBIENTE.

Os *workflows* da metodologia de desenvolvimento do SIGA-EDU constantes na tabela 1 podem ser definidos da seguinte forma:

- **PLANEJAMENTO:** fase comissionada aos gestores do projeto, onde é feito o planejamento geral, sua validação, definição de cronograma, prazos, metas, entre outros. É onde se deve procurar entender a estrutura, dinâmica e os problemas da organização onde o sistema será desenvolvido, visando sempre às melhorias e a produção de um software de qualidade;

- **REQUISITOS:** trata-se de um processo no SIGA-EDU. Tal processo visa identificar os requisitos do produto a partir da necessidade do cliente além de decidir os requisitos que terão prioridade e serão implementados, para então definir o cronograma. A figura 9 demonstra as principais atividades do grupo de requisitos: elicitar e validar requisitos, definir diagrama geral de casos de uso, planejar especificação e criar tarefas;

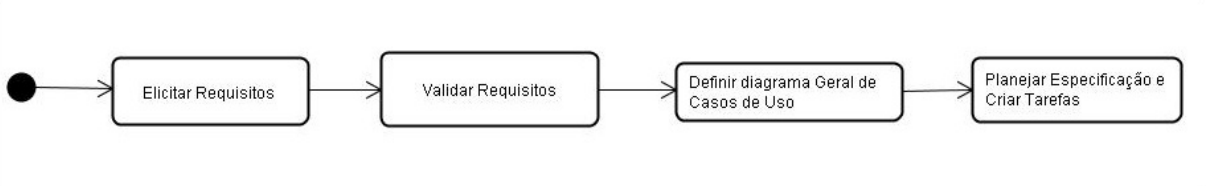


Figura 9: Processo de requisitos do SIGA-EDU.
Fonte: *Redmine* do SIGA-EDU, 2011.

- **ESPECIFICAÇÃO & MODELAGEM E BD:** estão separados no fluxo geral (Anexo 1), mas fazem parte do mesmo processo – especificação e projeto. Diz respeito ao processo que produz as especificações dos casos de uso (CDUs) a partir dos requisitos. Tais casos de uso descrevem como implementar o sistema. As atividades da equipe de especificação e projeto são: descrever CDUs, revisar descrição de CDUs, projetar tela, criar diagrama de classes, revisar especificação de CDUs, definir casos de testes, atualizar diagrama de classe geral, e produzir o projeto de BD para os CDUs desenvolvidos;

- **INTERFACE GRÁFICA:** faz parte do processo de especificação e projeto e serve como apoio para o desenvolvedor no processo de codificação. Na interface gráfica os atuantes são o designer e o especificador do CDU e as atividades são divididas em: projetar e validar tela, implementar e novamente validar a tela. Esse fluxo pode ser observado na figura abaixo;

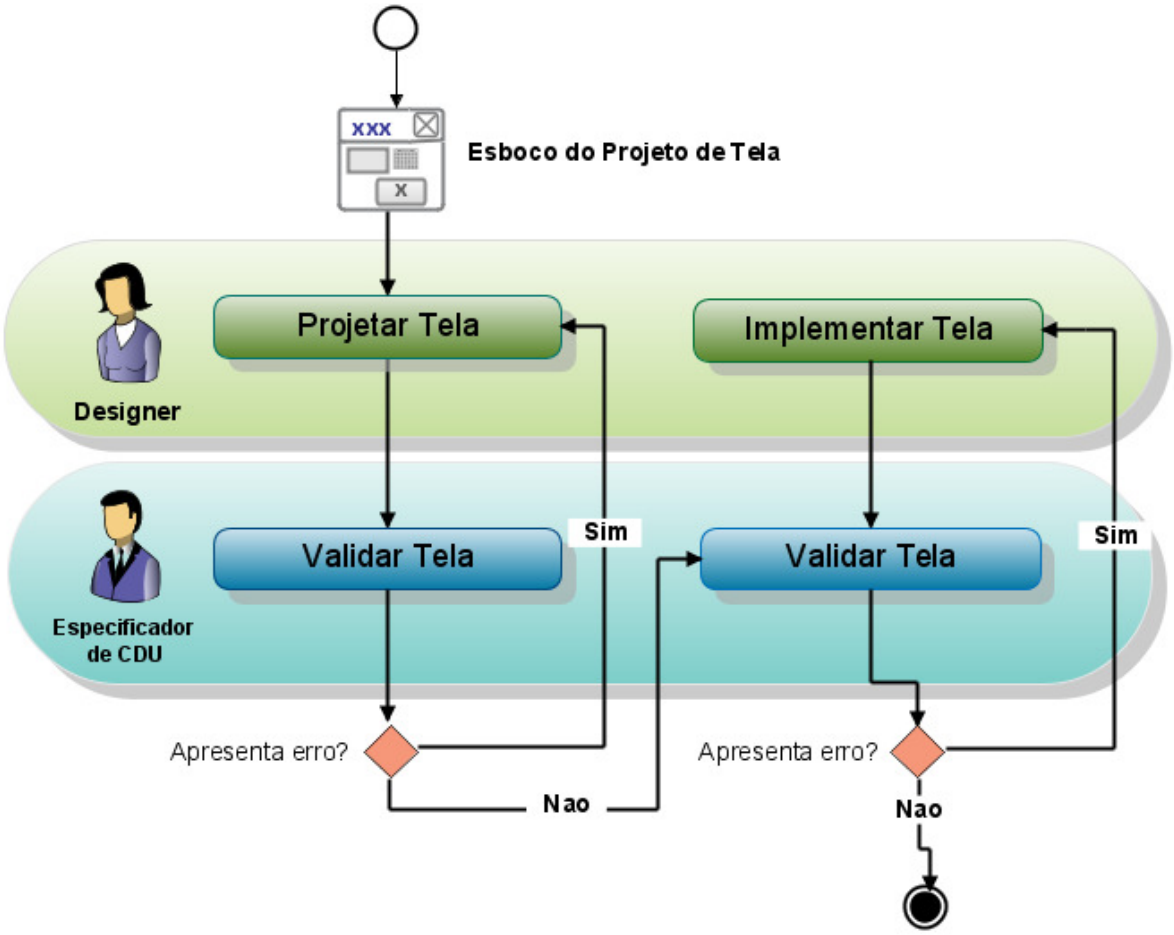


Figura 10: Fluxo das atividades do processo de criação de tela do SIGA-EDU.
Fonte: *Redmine* do SIGA-EDU, 2011.

- **CODIFICAÇÃO:** Nesta etapa é implementado o CORE (persistência, negócio e interface), a tela, são definidos os *inserts* do banco de dados e são realizados testes de unidade e testes de integração. A codificação é a aplicação prática dos resultados da especificação e projeto. Visa organizar o código em termos de subsistemas de componentes (código-fonte, arquivos de configuração, telas e outros);

- **TESTES:** é a fase onde são realizados os testes de aceitação. Nos testes são considerados os requisitos originais e as necessidades atuais dos usuários visando garantir o que foi solicitado pelos *stakeholders* (as partes envolvidas no desenvolvimento do sistema: gestores, fornecedores, clientes, desenvolvedores e etc.) e são realizados antes da disponibilização do sistema;

- **PRODUTO:** tem como finalidade disponibilizar para o usuário um *release* de uma versão do software. As principais atividades são: elaborar manual de uso e instalação do sistema, criar mecanismos de disponibilização do produto, sincronização da versão da documentação com a versão do código. Atua em conjunto com a infraestrutura no que diz respeito às necessidades do usuário em relação à instalação do sistema;

- **QUALI-EPT:** tem como objetivo realizar revisão de especificação de CDUs, os quais são selecionados por amostragem; definir matriz de rastreabilidade; e realizar engenharia reversa;

- **INFRAESTRUTURA:** é realizado o planejamento, manutenção e disponibilização de infraestrutura de servidores e serviço visando o trabalho eficiente e eficaz de forma distribuída por parte dos integrantes do SIGA-EDU. Alguns aspectos tecnológicos de infraestrutura são fornecidos pela coordenação de codificação.

Podem-se observar algumas diferenças no processo de desenvolvimento do SIGA-EDU em relação ao RUP, essas diferenças são justificáveis pela atuação em um modelo distribuído de desenvolvimento. Mas as características mais importantes do RUP foram herdadas para o processo de desenvolvimento do SIGA-EDU, pois o primeiro tem aspectos e características relevantes em um modelo colaborativo que levam vantagem sobre o modelo cascata, por exemplo.

Independente da metodologia de desenvolvimento deve-se considerar a que se trata de uma atividade complexa e que, portanto, trás uma série de desafios. Estes desafios (figura 11) podem ser classificados em três categorias principais (Prikladnicki & Audy 2008, p. 31): pessoas, que está relacionado aos fatores de recursos humanos envolvidos no desenvolvimento de software; processo, que trata da forma como o projeto será definido para ser desenvolvido; e tecnologia, que relacionam as ferramentas e aspectos tecnológicos utilizados para a produção no desenvolvimento de software.

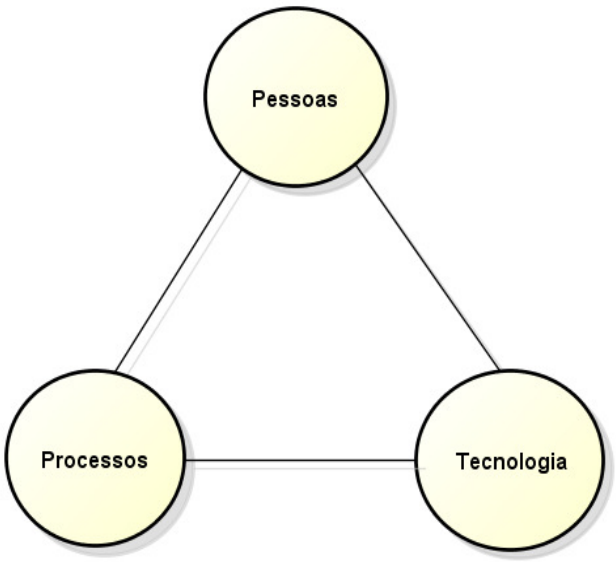


Figura 11: Desafios no desenvolvimento de software.
Fonte: Prikladnicki & Audy, 2008.

3.4 Desenvolvimento Distribuído de Software (DDS)

De forma proporcional em que a engenharia de software evolui, existe cada vez mais a necessidade de se produzir software com mais qualidade. A globalização dos negócios tem sido um dos fatores que contribuem para essa evolução e necessidade. Como foi mencionado anteriormente, o software é um diferencial competitivo e atua em diversas áreas de negócio.

Não é de hoje que o mercado de software tem ganhado proporções globais. Isso acarreta numa forma de competitividade e cooperação que pode ir além das fronteiras dos países. No que se refere à ambientes coorporativos, o custo-benefício do desenvolvimento de software em um mesmo espaço físico, organização ou até no mesmo país não tem sido tão interessante.

Fatores como o crescimento da economia, avanços nos meios de comunicação e a pressão por custos têm colaborado para o grande investimento em Desenvolvimento Distribuído de Software (DDS). A bibliografia sobre o tema é escassa no Brasil. Tanto em nível nacional como internacional existem vários termos sinônimos. A bibliografia utilizada nessa pesquisa considera o termo DDS com

abrangência geral e o desenvolvimento global de software (DGS) como o DDS em nível global.

Em um nível global, os fatores preponderantes para o avanço do DDS são:

- A necessidade de os recursos serem utilizados a qualquer hora, independentemente de localização;
- A vantagem de estar próximo do mercado local, o que leva ao conhecimento dos clientes e as condições locais de oportunidades no mercado;
- O desenvolvimento *time-to-market*, ou seja, sem afetar o nível de demanda e oferta do mercado utilizando-se do paradigma *follow-the-sun*, que trata do desenvolvimento 24 horas continuamente aproveitando-se da diferença de fuso horário. Isso se deve à demanda por serviços de software superar historicamente a disponibilidade de mão de obra que os realizam (figura 12).

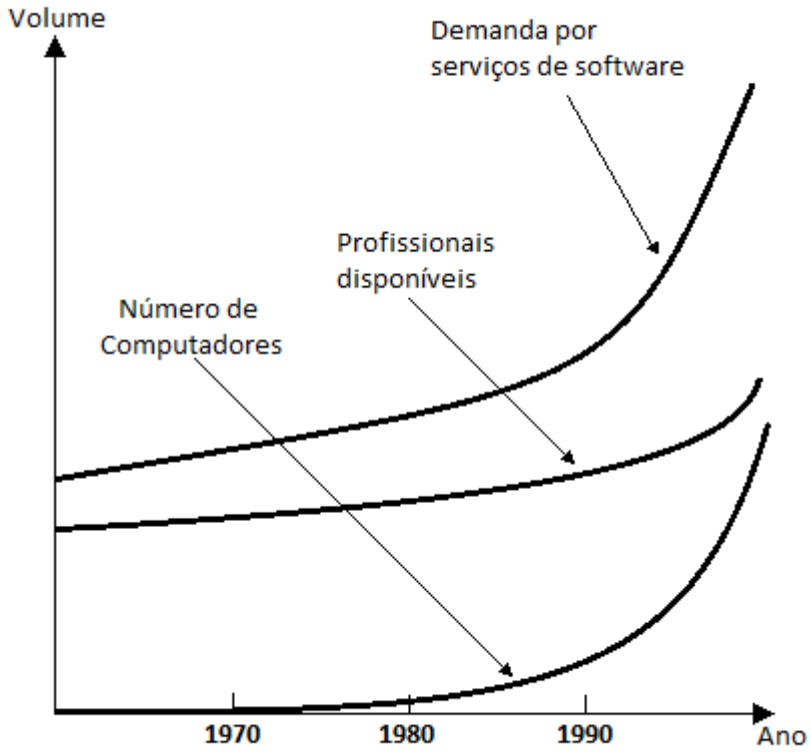


Figura 12: Demanda por software em relação ao número de computadores.
Fonte: Prikladnicki & Audy, 2008.

É fato que tanto o desenvolvimento de software tradicional ou co-localizado como o desenvolvimento distribuído de software têm suas dificuldades. As grandes diferenças entre essas metodologias de desenvolvimento são: distância física (dispersão geográfica), fuso horário e diferenças socioculturais.

Segundo (Prikladnicki & Audy, 2008, p. 44), o DDS tem sido caracterizado pela colaboração e cooperação entre departamentos de organizações e pela criação de grupos de desenvolvedores que trabalham em conjunto, localizados em cidades ou países diferentes. Sendo bem planejado, o desenvolvimento distribuído trás grandes benefícios para as tarefas de desenvolvimento de software.

O paradigma do desenvolvimento colaborativo é recente. Surgiu no desenvolvimento de software no final do século passado, nos anos 90 e só alcançou grandes proporções nos últimos 12 anos, após o crescimento da internet para que atingisse tamanha grandeza e eficácia. Hoje existem grandes projetos mantidos com essa metodologia, a saber, GNU/Linux e Wikipédia; e grandes empresas apoiando a utilização dessa metodologia, tais como: Google, Microsoft, Oracle, IBM e HP.

> O desenvolvimento distribuído de software caracteriza-se pela distância física e/ou temporal entre alguns elementos (cliente, usuário e desenvolvedores, por exemplo) envolvidos no processo de desenvolvimento. (PRIKLADNICKI & AUDY, 2008, p. 54)

> Entende-se como desenvolvimento colaborativo o ato de vários programadores que podem estar separados geograficamente ou não, serem voluntários ou pagos, com interesses em comum e que dedicam parte do seu tempo para programar, manter e melhorar algum projeto de software livre. (ANDERSON, 2009)

Em um modelo de gestão e execução distribuído onde os participantes ou colaboradores podem estar dispersos em localidades geográficas diferentes, uma metodologia de desenvolvimento distribuído de software abraça as necessidades como um todo. Nestas circunstâncias, o fluxo de informações se dá por meios eletrônicos com poucas reuniões presenciais e são necessárias ferramentas eficientes e dinâmicas para o sucesso do projeto.

3.5 Desenvolvimento Distribuído no SIGA-EDU

O projeto SIGA-EDU segue essa premissa de colaboração em seu desenvolvimento e gestão. O projeto é classificado em relação ao modelo de negócio de desenvolvimento distribuído como sendo *onshore*, isto é, seu nível de dispersão física é nacional (figura 13) e é caracterizado por ter os atores localizados dentro de um mesmo país e que podem reunir-se em um intervalo de tempo.

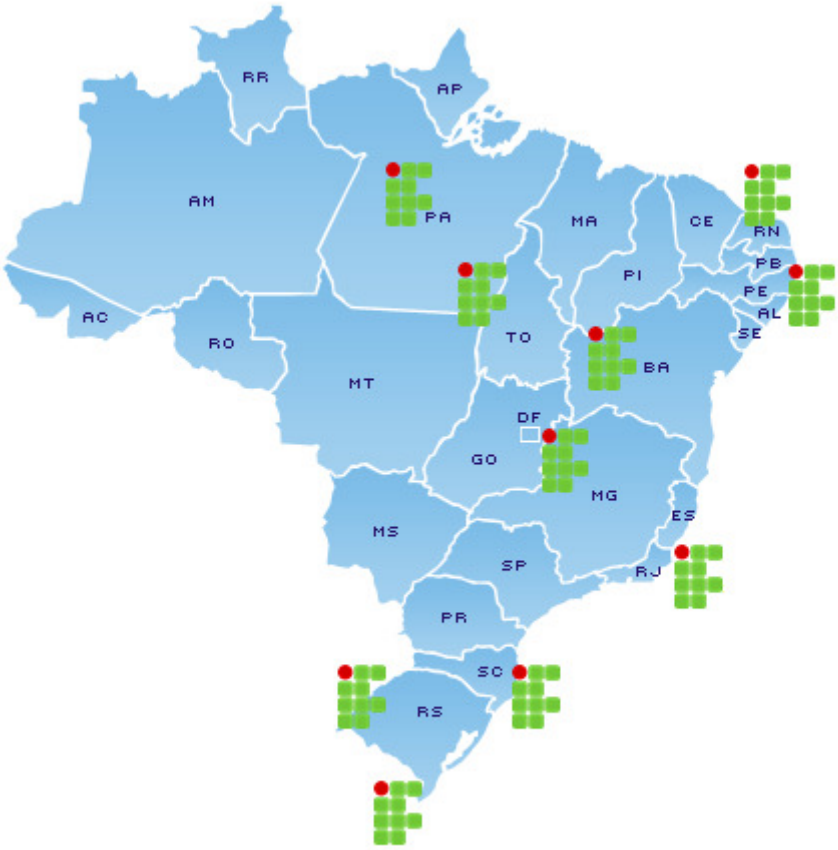


Figura 13: Dispersão física nacional do SIGA-EDU.

Os desafios no desenvolvimento de software abordados no subitem 3.3 e que pode ser observado na figura 11 levam a conclusão que o desenvolvimento de software se apresenta de forma complexa. O DDS, com suas peculiaridades no que se refere à dispersão física, distância temporal e diferenças culturais – os dois últimos não acontecem no caso do SIGA-EDU –, adicionou alguns outros desafios ao processo.

Além dos desafios já existentes (pessoas, processos e tecnologia), o DDS atribui ainda duas novas categorias: gestão e comunicação (figura 14).

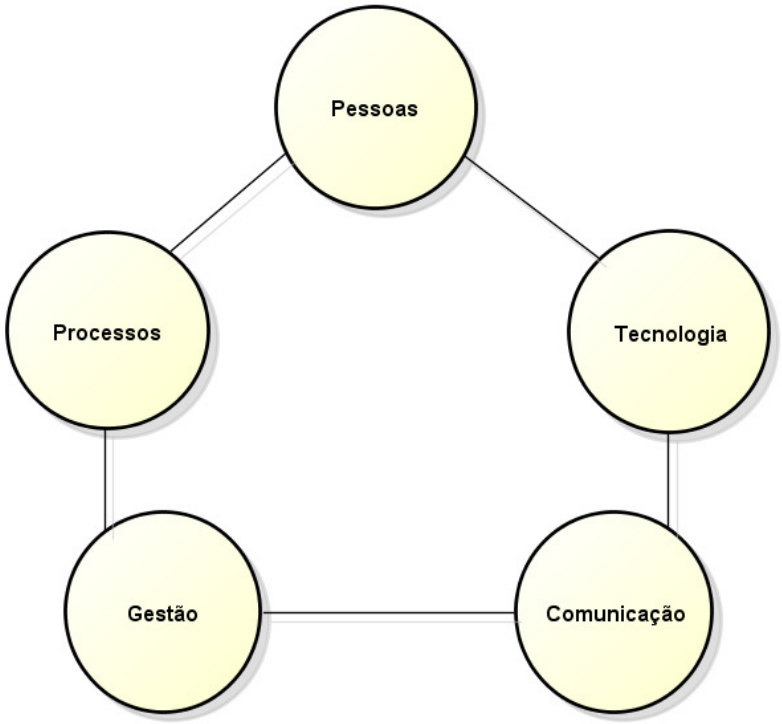


Figura 14: Desafios no desenvolvimento distribuído de software.
Fonte: Prikladnicki & Audy, 2008.

Em relação à gestão, os desafios estão relacionados aos três níveis de decisão organizacional do desenvolvimento distribuído de software: gestão estratégica, gestão tática e gestão operacional. A comunicação é um desafio que interfere no relacionamento e gestão da infraestrutura dos membros do projeto distribuído. Tanto os desafios do desenvolvimento de software co-localizado, quanto os desafios do desenvolvimento de software distribuído são elencados na Tabela 2.

CATEGORIAS	DESAFIOS
Pessoas	Confiança Conflitos Diferenças culturais Ensino de DDS Trabalho em equipe Formação de equipes e grupos Tamanho da equipe
Processo	Arquitetura do software Engenharia de requisitos Gerência de configuração Processo de desenvolvimento
Tecnologia	Tecnologia de colaboração Telecomunicações
Gestão	Coordenação e controle Modelos de negócio
Comunicação	*Awareness* Dispersão geográfica Formas de comunicação

Tabela 2: Principais desafios do desenvolvimento distribuído de software (adaptado).
Fonte: Prikladnicki & Audy, 2008.

Os desafios de DDS relacionados a pessoas são classificados tendo como base os aspectos de recursos humanos: convivência, relacionamento, conhecimento, trabalho em equipe, liderança entre outros. A seguir são abordados alguns destes desafios na ambiente do SIGA-EDU:

- **Confiança:** é o fator crucial na interação entre os membros da equipe. Reuniões presenciais acontecem constantemente no projeto SIGA-EDU, com uma frequência quase que mensal e ajudam a nutrir o sentimento de confiança do grupo apesar de todos os desafios;

- **Conflitos:** acontecem com frequência em ambientes de desenvolvimento distribuído. Algumas vezes os conflitos acontecem de forma saudável visando à evolução do processo. Tanto os conflitos técnicos como não-técnicos, caso aconteçam e não sejam resolvidos, cabe a autoridade central do projeto resolver, no caso de SIGA-EDU, o gerente do projeto;

- **Diferenças culturais:** apesar de ser desenvolvido em um país de dimensões continentais e deter uma série de etnias, culturas e comportamentos, o SIGA-EDU não é tão afetado por este desafio;

- **Ensino de DDS:** o conhecimento dessa área é fundamental no projeto, mas não é tão abordado em instituições de ensino da área de TI. Ao entrar no SIGA-EDU, o novo membro tem um auxílio do gestor do núcleo que pertence em relação a todos os aspectos tecnológicos utilizados no projeto e o conhecimento de DDS é um deles. Como foi dito anteriormente, DDS é uma área nova que está ganhando força no âmbito corporativo e acredita-se que brevemente serão incluídos tópicos relativos a desenvolvimento distribuído em disciplinas de cursos de graduação (Watson-Manheim & Audy, 2005);

- **Trabalho em equipe:** o SIGA-EDU tem esse conceito e é aplicado nos núcleos de desenvolvimento e grupos de trabalho que realização atividades específicas. Esses conceitos serão abordados mais a frente ao tratar dos papéis dos membros do SIGA-EDU;

- **Tamanho da Equipe:** o SIGA-EDU tem cerca de 130 membros envolvidos distribuídos em 10 núcleos de desenvolvimento (figura 13). É uma equipe muito grande e cabe à gestão do projeto controlar e gerir os problemas causados por este desafio. A SIGA-EDU tem uma divisão organizacional que colabora com a descentralização das tarefas e facilita na gestão, isso pode ser observado no organograma do projeto (figura 15).

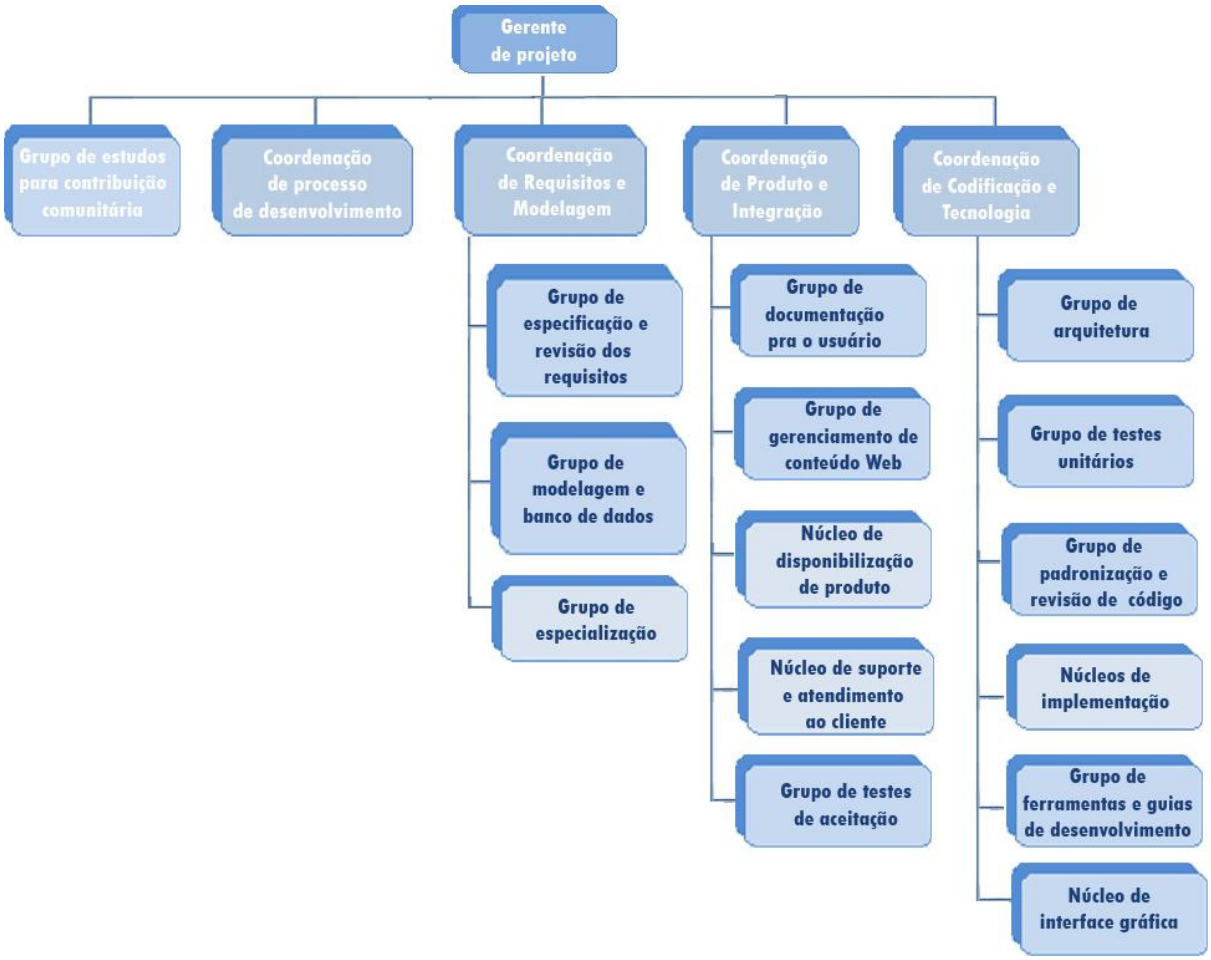


Figura 15: Organograma do SIGA-EDU.
Fonte: *Redmine* do SIGA-EDU, 2011.

O processo de desenvolvimento de software é uma peça chave para a produção de software de qualidade. É no processo que estão definidas as bases para gerenciar o desenvolvimento e direcionar o uso de métodos e tecnologias no intuito de padronizar o desenvolvimento. O SIGA-EDU adota alguns padrões no desenvolvimento e sua aplicação requer atenção para alguns desafios:

- **Arquitetura do software:** como foi abordado no capítulo 2 sobre Arquitetura em Camadas, o SIGA-EDU garante um alto índice de coesão e o mínimo de acoplamento. Está é a única maneira de resolver e alocar tarefas complexas de forma distribuída (Prikladnicki & Audy, 2008, p. 68);

- **Engenharia de requisitos:** no desenvolvimento distribuído, a engenharia de requisitos é um fundamento que necessita de um alto índice de comunicação e coordenação. Isso é necessário, porque a maior dificuldade na construção de um software é decidir precisamente o que construir, pois nenhuma teoria relacionada ao processo de desenvolvimento é tão difícil quanto estabelecer detalhes em requisitos técnicos. O SIGA-EDU trata o processo de requisitos com a devida importância pelo fato de a maioria das reuniões presencias serem para discutir este processo;

- **Gerência de configuração:** esse desafio não é tão problemático. Em relação ao SIGA-EDU, foi adotado a ferramenta de controle de versão *Subversion* que trata as modificações dos artefatos e gera um histórico de alterações;

- **Processo de desenvolvimento:** um grande problema no processo de desenvolvimento, diz respeito à falta de sincronia das atividades. A ferramenta de gerenciamento de projeto do SIGA-EDU – o *Redmine* – é um facilitador nesse contexto. Mas os gestores e coordenadores têm que ficar atentos ao que acontece nas atividades/tarefas para que o processo seja contínuo e síncrono;

As tecnologias avançam a cada dia e tornam o processo de desenvolvimento de software cada vez mais produtivo. As tecnologias de colaboração e telecomunicações estão presentes no SIGA-EDU:

- **Tecnologias de colaboração:** a comunicação eletrônica entre os locais distribuídos deve ser o atenuante para os desafios impostos pela distância, coordenação e falta de comunicação presencial. Algumas dessas tecnologias utilizadas no SIGA-EDU são: telefone, videoconferência, chat eletrônico, *email* e listas de discussão. O *Redmine* atua na colaboração para suporte nas atividades de engenharia de software e serve como repositório de informações para a equipe atuando como redutor de retrabalho, fornecedor de suporte à coordenação das tarefas e controle de qualidade;

- **Telecomunicações:** no SIGA-EDU existe a coordenação de infraestrutura que visa garantir o ambiente propício para o desenvolvimento. Pode envolver disponibilidade dos sistemas utilizados, integridade da informação, manutenção dos serviços entre outros;

A gestão colaborativa é uma tarefa bem complexa que envolve aspectos estratégicos e operacionais. Uma das características no desenvolvimento de software livre, face à gestão colaborativa, é que se deve haver uma maior flexibilidade dos papéis dos membros do projeto. Isso se dá pela grande rotatividade desses membros durante o ciclo de vida do software. O modelo de estrutura organizacional do SIGA foi definido levando em consideração os pontos de melhoria apontados pelos gestores ao longo do tempo. Existe também a necessidade de planejamentos contínuos e controle dos processos visando promover o acompanhamento do projeto e o fornecimento do *feedback* por meio de informações conclusivas de controle de cada etapa do processo de desenvolvimento.

Esse modelo tem como objetivo a descentralização da gerência do projeto, o que acarreta da distribuição da carga de trabalho diminuindo a dependência de outrem e aumentando a agilidade nas respostas.

Todas estas características do modelo de estrutura organizacional do SIGA-EDU (figura 15) demostram como o desenvolvimento pode ser ainda mais colaborativo. Essa estrutura está da seguinte forma: coordenação, que atua na definição de estratégias, planejamento, atribuição de tarefas e acompanhamento de procedimentos nas suas respectivas áreas de atuação; núcleos, que visam realizar atividades específicas, ficando todos ou parte de seus membros dedicados às atividades específicas do núcleo. Os núcleos trabalham em conjunto com os gerentes de área. Tais núcleos estão dispersos pelo país e podem ser observados na figura 13; grupos, que são formados por gestores, professores orientadores, analistas de sistemas e/ou desenvolvedores experientes, podendo estes pertencer ao mesmo núcleo ou não. Cada grupo possui um líder que representará o grupo. Os membros de cada grupo são responsáveis por propor soluções, atuar na elaboração de atividades, possuindo uma equipe de desenvolvedores para executar algumas destas atividades. Os grupos trabalham orientados por um coordenador de área;

gestor de núcleo, que tem a função de representar o núcleo e possui atividades como acompanhamento das atividades dos membros do núcleo e realização de atividades técnicas e administrativas; líder de grupo, que atua na representação do grupo e possui atividades como acompanhamento e execução de atividades do grupo; pesquisador orientador, que atua na orientação aos desenvolvedores e realização de atividades técnicas dentro do domínio do projeto; e por fim, o desenvolvedor que realiza atividades técnicas.

Nesse modelo, o SIGA-EDU segue as premissas de gestão colaborativa levando em consideração a coordenação e controle e modelo de negócio:

- **Coordenação e controle:** a coordenação tem como objetivo integrar as unidades organizacionais, no caso do SIGA-EDU, os núcleos de desenvolvimento no intuito de contribuir para o objetivo geral do desenvolvimento do sistema. Já o controle, atua no que se referem a metas, políticas e padrões. Exemplos de controle no SIGA-EDU são: padrão de banco de dados, de codificação, metas a serem cumpridas em cada versão, guias, entre outros;

- **Modelos de negócio:** o SIGA-EDU é classificado como *onshore insourcing* que significa demanda doméstica interna. Ou seja, o governo federal utiliza de uma de suas áreas, a educação federal (universidades e institutos federais), para o desenvolvimento de um sistema. Na visão conceitual pode-se dizer que o *onshore insourcing* trata da existência de um departamento dentro da própria empresa ou uma subsidiária da empresa dentro do país de origem.

A comunicação é um dos fatores essenciais no desenvolvimento distribuído de software pelo fato da dispersão física impedir o contato frequente entre os membros do projeto. O SIGA-EDU enfrenta esses desafios com uma série de artifícios para que o processo não seja tão afetado. A seguir alguns dos desafios dessa área:

- ***Awareness* (conscientização):** se refere ao conhecimento ou consciência das atividades desenvolvidas e das alterações em padrões de codificação, BD, especificação, documentação, entre outros. O SIGA-EDU conta com as tarefas no *Redmine* que têm a funcionalidade de observação. Tal funcionalidade permite adicionar observadores nas tarefas, ou seja, a parte interessada naquela fase do desenvolvimento e envia um *email* a cada alteração na tarefa. Outras formas de manter o *awareness* são: wiki do projeto, fórum e lista de discussão.

- **Dispersão geográfica:** fator característico do DDS, a distância física literalmente impõe barreiras no processo de desenvolvimento. Esses impedimentos são atenuados com as técnicas de comunicação utilizados no SIGA-EDU.

- **Formas de comunicação:** O projeto SIGA-EDU detém várias formas de comunicação, seja ela formal ou informal. Apesar disso as dificuldades são muitas e deve-se ter uma comunicação clara e efetiva para o sucesso no processo de desenvolvimento. Um problema de interpretação na leitura de um caso de uso pode afetar bastante o produto final. Por isso a padronização na escrita da especificação é utilizada no SIGA-EDU. Mas as formas de comunicação são diversas e abrangem tanto a linguagem formal como a informal:

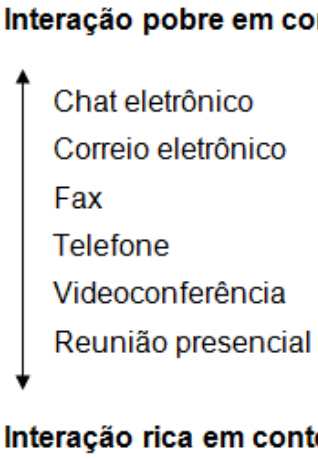


Figura 16: Níveis de interação contextual (adaptado).
Fonte: Prikladnicki & Audy, 2008.

O Fluxo de comunicação no SIGA-EDU acontece conforme a Figura 17 de forma hierárquica desde a gerência do projeto até os membros dos núcleos, por meio de fóruns de discussão, lista de e-mail do projeto, comunicadores instantâneos e poucas reuniões presenciais.

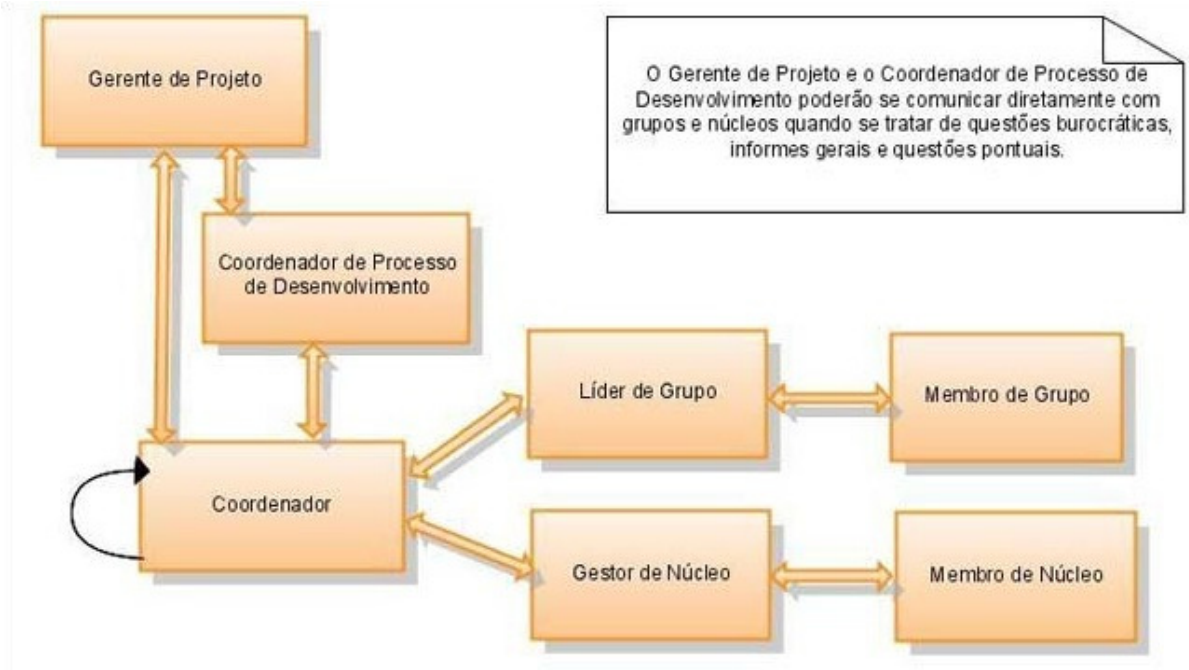


Figura 17: Fluxo de comunicação do SIGA-EDU.
Fonte: *Redmine* do SIGA-EDU, 2011.

Os desafios na metodologia de desenvolvimento distribuído de software são muitos e exigem atenção redobrada, pois os empecilhos existentes são vários e carecem de uma gerência atenta em todas as fases do processo para que os desafios não se tornem problemas e venham atrapalhar o desenvolvimento do sistema.

4 PROCESSO DE CORREÇÃO DE BUGS

4.1 Abordagem Inicial

Como foi dito anteriormente, o processo de desenvolvimento de software visa organizar, documentar, definir fluxos e responsáveis relacionados às tarefas que serão realizadas durante o projeto. Esse processo é um dos principais responsáveis para se adquirir um software de qualidade. O SIGA-EDU tem seu próprio processo de desenvolvimento que herda várias características do RUP, a saber, os seguintes workflows (SOMMERVILLE, 2007): requisitos, análise e projeto, implementação, teste e implantação, além de ser orientado a casos de uso. O SIGA-EDU conta ainda com o planejamento e testes.

Esse processo acarreta em um desenvolvimento iterativo e incremental onde em cada ciclo uma parte do produto será entregue gerando uma nova versão. Em cada versão são disponibilizadas novas funcionalidades para o cliente.

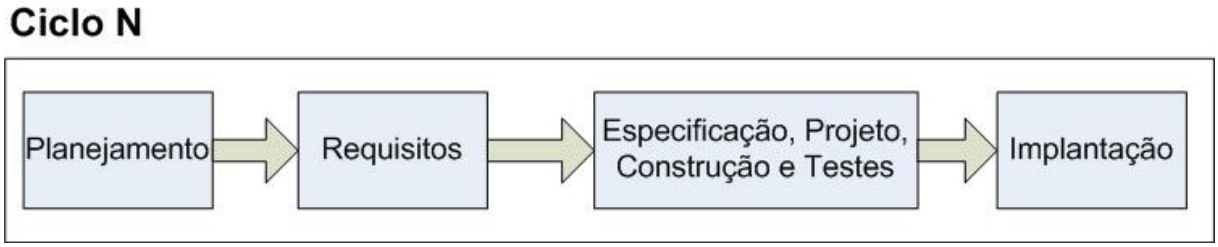


Figura 18: Fases do Clico de Vida do SIGA-EDU.
Fonte: *Redmine* do SIGA-EDU, 2011.

No SIGA-EDU, em cada ciclo o sistema é submetido a testes durante e após a codificação. Os testes são divididos em testes de aceitação e testes de integração e ambos visam à detecção de problemas durante o processo de desenvolvimento.

A atuação na detecção, análise e resolução de bugs é de grande importância no desenvolvimento distribuído de software por contribuir amenizando os impactos no processo e evitando que chegue às mãos do usuário um produto não funcional.

4.2 Fluxo de Correção de Bugs

No decorrer de todas as etapas de desenvolvimento pode ocorrer de uma funcionalidade não ter sido desenvolvida de acordo com a documentação, uma das etapas do ciclo não ser executada corretamente, um componente do layout não estar correto, uma codificação errada acarretando na aparição de erros no sistema, os chamados bugs.

Ou ainda, no caso do SIGA-EDU, devido ao número alto de rotatividade de membros do projeto, algumas tarefas acabam sendo atribuídas a desenvolvedores inexperientes (a maior parte da mão de obra do SIGA-EDU é formada por acadêmicos de cursos superiores de institutos federais e universidades). Isso provoca a aparição de bugs no sistema e consequentemente pode causar refatoramento do CDU, o que gera retrabalho.

O SIGA-EDU conta com um fluxo de correção de bugs que dá origem ao processo de correção de bugs do SIGA-EDU e é composto de várias atividades, são elas: descrever novo bug, analisar bug, atribuir bug, resolver bug, verificar correção de bug e fechar bug.

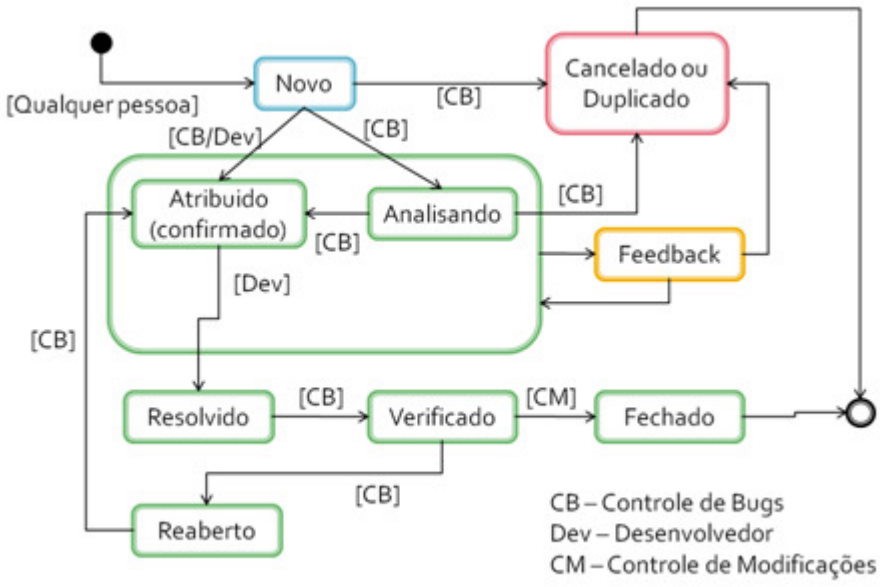


Figura 19: Fluxo de correção de bugs do SIGA-EDU.
Fonte: *Redmine* do SIGA-EDU, 2011.

O processo de correção de bugs descreve todas as etapas detalhadamente e demonstra como deve ser feito para que o problema seja reparado e o ciclo continue. Abaixo segue os detalhes do processo:

Atividade: descrever novo bug

Descrição	Após identificar um novo Bug no sistema, qualquer pessoa cria tarefa descrevendo novo Bug.
Critérios de Entrada	Novo Bug descoberto
Critérios de Saída	Bug cadastrado no sistema
Responsáveis	Qualquer pessoa, desde desenvolvedor e pesquisador, até usuário final.
Participantes	N/A
Artefatos Requeridos	N/A
Artefatos Produzidos	Tarefa com descrição do bug em seu conteúdo e/ou anexos. Entre outras coisas, deve conter: gravidade, reprodutibilidade, versão alvo;
Modelo comunicação	Criar tarefa dentro da hierarquia de tarefas do CDU em tarefa específica para o processo de bugs. Padrão de nome: Analisar bug SIGA-EDU-CDU-XXXX-NNN: <descrição resumida> [<origem>]

Tabela 3: Atividade: descrever novo bug.
Fonte: *Redmine* do SIGA-EDU, 2011.

Atividade: analisar bug

Descrição	Analisar o novo bug descrito, resultando em indicação para correção do bug, ou mesmo cancelamento (duplicado). Durante o processo de análise, desenvolvedores, pesquisadores ou consultores podem ser convocados para ajudar no entendimento do bug (*feedback*).
Critérios de Entrada	Novo Bug cadastrado no sistema
Critérios de Saída	Cancelamento do bug ou atribuição para correção
Responsáveis	Líder do grupo de correção de bugs
Participantes	Qualquer desenvolvedor, pesquisador ou consultor que possa ajudar na compreensão do bug.
Artefatos Requeridos	Artefatos relacionados ao CDU
Artefatos Produzidos	Atualização da tarefa, caso possível, com informações sobre o parecer.
Modelo comunicação	Emitir parecer sobre a análise do bug na tarefa “Analisar bug SIGA-EDU-CDU-XXXX-NNN: <descrição resumida> [<origem>]”. Caso o parecer resulte em cancelamento ou duplicação, colocar essa informação no título da tarefa.

Tabela 4: Atividade: analisar bug.
Fonte: *Redmine* do SIGA-EDU, 2011.

Atividade: atribuir bug

Descrição	Atribuir a tarefa ao responsável pela correção do bug após análise prévia, ou após verificar que correção de bug não foi realizada com sucesso.
Critérios de Entrada	Bug pendente de correção
Critérios de Saída	Atividade de correção atribuída a pesquisador ou desenvolvedor
Responsáveis	Desenvolvedor ou pesquisador
Participantes	N/A
Artefatos Requeridos	Lista de desenvolvedores/pesquisadores
Artefatos Produzidos	Tarefa para correção de bug.
Modelo comunicação	Realizar cópia da tarefa “Analisar bug SIGA-EDU-CDU-XXXX-NNN: <descrição resumida> [<origem>]” com o nome “Atualizar YYY SIGA-EDU-CDU-XXXX-NNN” onde YYY identifica o que deve ser atualizado (Descrição do CDU, Projeto de Tela, etc.) e atribuir ao responsável pela correção.

Tabela 5: Atividade: atribuir bug.
Fonte: *Redmine* do SIGA-EDU, 2011.

Atividade: resolver bug

Descrição	Responsável pela correção do bug deve realizar conserto
Critérios de Entrada	Tarefa com descrição de bug para ser consertado
Critérios de Saída	Bug corrigido
Responsáveis	Desenvolvedor ou pesquisador
Participantes	N/A
Artefatos Requeridos	Artefatos originais
Artefatos Produzidos	Artefatos atualizados. Descrição da correção (Resumo não técnico do bug; Causa do bug; Descrição de como foi resolvido; o Número do commit).
Modelo comunicação	Atualizar a tarefa “Atualizar YYY SIGA-EDU-CDU-XXXX-NNN” com resultados da correção do bugs.

Tabela 6: Atividade: resolver bug.
Fonte: *Redmine* do SIGA-EDU, 2011.

Atividade: verificar correção bug

Descrição	Verificar se o bug foi realmente corrigido. Caso o bug não tenha sido corrigido a contento, deve retornar para tarefa atribuir bug.
Critérios de Entrada	Conclusão da atividade “Resolver bug (Atualizar YYY SIGA-EDU-CDU-XXXX-NNN)”
Critérios de Saída	Parecer positivo sobre a correção do bug
Responsáveis	Líder do grupo de correção de bugs
Participantes	N/A
Artefatos Requeridos	Artefatos originais (antes da correção do bug) e Artefatos atualizados (após correção)
Artefatos Produzidos	Parecer informando se o bug foi corrigido ou não; caso corrigido, acrescentar na descrição a versão na qual a correção será disponibilizada.
Modelo comunicação	Atualizar a tarefa “Atualizar YYY SIGA-EDU-CDU-XXXX-NNN” com resultados da verificação de correção do bug.

Tabela 7: Atividade: verificar correção bug.
Fonte: *Redmine* do SIGA-EDU, 2011.

Atividade: fechar tarefa

Descrição	Líder do grupo de correção de bugs fecha tarefa após bug corrigido e parecer positivo sobre a conclusão da tarefa
Critérios de Entrada	Parecer positivo sobre a conclusão da tarefa “Resolver Bug”.
Critérios de Saída	Tarefa fechada
Responsáveis	Líder do grupo de correção de bugs
Participantes	N/A
Artefatos Requeridos	N/A
Artefatos Produzidos	N/A
Modelo comunicação	Fechar a tarefa “Atualizar YYY SIGA-EDU-CDU-XXXX-NNN”

Tabela 8: Atividade: fechar tarefa.
Fonte: *Redmine* do SIGA-EDU, 2011.

4.3 Resultados

O SIGA-EDU tem pouco mais de três anos de existência e já foram realizadas aproximadamente 1600 tarefas de bug, o que leva a uma média de um bug e meio por dia. É fato que os bugs aumentam proporcionalmente com o tamanho do projeto e consequentemente a equipe para atender essa demanda também aumenta.

Ao atuar no processo de desenvolvimento do SIGA-EDU como desenvolvedor, há uma necessidade de mapeamento dos bugs para melhor organização do conteúdo e resolução das tarefas. Os bugs se apresentam de diversas formas e intensidade. Em véspera de lançamento de versão os testes são realizados e consequentemente os problemas aparecem e carecem de ser resolvidos imediatamente. Em sua maioria são de fácil resolução e não exigem muito tempo por parte do desenvolvedor. Mas não significa que não existam os bugs que requerem um pouco mais de tempo e atenção. Com base nessas conclusões e nas tarefas realizadas, é possível dividir os bugs em três categorias: bugs de layout, bugs de codificação e bugs de modelagem e/ou especificação.

Os bugs de layout geralmente são de fácil resolução e não consomem muito tempo, fazem referência à parte visual da funcionalidade e não afetam o banco de dados ou a camada de negócio. Abaixo segue alguns resultados de bugs desta categoria:

Título	SIGA-EDU-CDU-MATRI-065: TMVL - Manter Aproveitamento, Aceleração e Dispensa: Layout.		
Prioridade	Normal	**Categoria**	Bug
Núcleo	IFTO	**Coordenação SIGA**	Codificação e Tecnologia
Início	24/02/2011	**Data prevista**	01/03/2011
Tempo gasto	0.1 horas		
Descrição	Na tela inicial – O *fildset* “Elemento Curricular a Cursar” juntamente com o botão “cancelar” tem que surge após a consulta.		

Tabela 9: Tarefa de bug de layout do SIGA-EDU.
Fonte: *Redmine* do SIGA-EDU, 2011.

Título	Atualizar CT03/014 SIGA-EDU-CDU-EXTEN-028: Campos sem mascara. [TM]		
Prioridade	Normal	**Categoria**	Bug
Núcleo	IFTO	**Coordenação SIGA**	Codificação e Tecnologia
Início	01/11/2010	**Data prevista**	05/11/2010
Tempo gasto	0.2 horas		
Descrição	CT03-Os campos data de início e data de termino estão sem mascara		

Tabela 10: Tarefa de bug de layout do SIGA-EDU.
Fonte: *Redmine* do SIGA-EDU, 2011.

Título	Atualizar CT04/014 SIGA-EDU-CDU-EXTEN-028: Campo desalinhado. [TM]		
Prioridade	Normal	**Categoria**	Bug
Núcleo	IFTO	**Coordenação SIGA**	Codificação e Tecnologia
Início	01/11/2010	**Data prevista**	10/11/2010
Tempo gasto	0.1 horas		
Descrição	CT04-O campo Possui Bolsa está desalinhado.		

Tabela 11: Tarefa de bug de layout do SIGA-EDU.
Fonte: *Redmine* do SIGA-EDU, 2011.

Título	CT005/13 SIGA-EDU-CDU-MATRI-012: erro na exibição de dados [TM]		
Prioridade	Normal	**Categoria**	Bug
Núcleo	IFTO	**Coordenação SIGA**	Codificação e Tecnologia
Início	04/10/2010	**Data prevista**	08/10/2010
Tempo gasto	0.2 horas		
Descrição	Ao realizar uma matrícula, o PDF gerado exibe o CPF apenas com os números em sequência. Sugiro que seja usada uma máscara para exibir o CPF com um ponto a cada 3 dígitos e o hífen antes dos dois últimos números. Outro detalhe do PDF é a data que apresenta o mês em inglês.		

Tabela 12: Tarefa de bug de layout do SIGA-EDU.
Fonte: *Redmine* do SIGA-EDU, 2011.

Por não contar com o aspecto visual por parte do usuário e por ser submetida a fatores suscetíveis a causa de bugs (desenvolvedor inexperiente, especificação errada, diagrama de classes errado entre outros), a codificação é a etapa do processo de desenvolvimento onde ocorre a maioria dos bugs. Geralmente demandam mais tempo e atenção por parte do desenvolvedor. Segue alguns resultados:

Título	TCEX018Servidor/5.1RC04 [TM] Erro ao excluir servidor		
Prioridade	Urgente	**Categoria**	Bug
Núcleo	IFTO	**Coordenação SIGA**	Codificação e Tecnologia
Início	23/12/2010	**Data prevista**	28/12/2010
Tempo gasto	4.40 horas		
Descrição	Ao tentar excluir servidor "Monteiro Lobato" a seguinte mensagem é mostrada: "*Erro ao realizar a exclusão: Ocorreu um erro de Banco de Dados. Detalhe: ERROR: update or delete on table "ato_autorizativo" violates foreign key constraint "ato_autorizativo_fk" on table "elemento_organizacional" Detail: Key (id)=(1) is still referenced from table "elemento_organizacional". STATUS:[23503].*"		

Tabela 13: Tarefa de bug de codificação do SIGA-EDU.
Fonte: *Redmine* do SIGA-EDU, 2011.

Título	Atualizar CT01/014 SIGA-EDU-CDU-EXTEN-028: Mensagem de erro sem informação [TM]		
Prioridade	Normal	**Categoria**	Bug
Núcleo	IFTO	**Coordenação SIGA**	Codificação e Tecnologia
Início	01/11/2010	**Data prevista**	09/11/2010
Tempo gasto	6.00 horas		
Descrição	CT01-Ao se escolher um novo participante e clicar em incluir participação sem preencher os dados requisitados, aparece a mensagem de erro "*formInserirAlterar:papelParticipante: An error occurred when processing your submitted information.*"		

Tabela 14: Tarefa de bug de codificação do SIGA-EDU.
Fonte: *Redmine* do SIGA-EDU, 2011.

Título	CT008/13 SIGA-EDU-CDU-MATRI-012: erro na listagem do período letivo [TM]		
Prioridade	Urgente	**Categoria**	Bug
Núcleo	IFTO	**Coordenação SIGA**	Codificação e Tecnologia
Início	04/10/2010	**Data prevista**	08/10/2010
Tempo gasto	4.5 horas		
Descrição	Na tela de cadastro de matrícula, o sistema lista períodos letivos anteriores a data corrente e permite o cadastro de matrícula com períodos letivos que já passaram.		

Tabela 15: Tarefa de bug de codificação do SIGA-EDU.
Fonte: *Redmine* do SIGA-EDU, 2011.

Título	CT015/13 SIGA-EDU-CDU-MATRI-012: erro no cadastro de matrícula [TM]		
Prioridade	Normal	**Categoria**	Bug
Núcleo	IFTO	**Coordenação SIGA**	Codificação e Tecnologia
Início	04/10/2010	**Data prevista**	08/10/2010
Tempo gasto	4.00 horas		
Descrição	O sistema permitiu que o mesmo aluno fosse matriculado duas vezes no mesmo curso, para o mesmo período na mesma unidade de ensino e com a mesma matriz curricular.		

Tabela 16: Tarefa de bug de codificação do SIGA-EDU.
Fonte: *Redmine* do SIGA-EDU, 2011.

Alguns bugs são notados apenas depois de feito todo o processo por não terem sido previstos nos requisitos, especificação e modelagem. Na maioria das vezes sua resolução não depende apenas do desenvolvedor e é encaminhado ao alguém responsável pelos requisitos, modelagem ou projeto. Alguns desses bugs são demonstrados abaixo:

Título	TCVL008ManterEstruturaOrganizacional/6.0.RC02 [TM] Mensagem		
Prioridade	Normal	**Categoria**	Bug
Núcleo	IFTO	**Coordenação SIGA**	Codificação e Tecnologia
Início	22/02/2011	**Data prevista**	25/02/2011
Tempo gasto	0.1 horas		
Descrição	TCVL008- Ao informar menos de 8 dígitos no campo número do telefone o sistema informa: ' Campo NÚMERO DO TELEFONE deve conter 8 dígitos.' Porém esta mensagem não está presente na documentação do CDU.		
Observação	**Nesse caso, a correção do código-fonte não foi necessária, pois a validação de telefone é padrão em todo o sistema.**		

Tabela 17: Tarefa de bug de especificação do SIGA-EDU.
Fonte: *Redmine* do SIGA-EDU, 2011.

Título	TCVL001ManterEstruturaOrganizacional/6.0.RC02 [TM] Campo não declarado.		
Prioridade	Normal	**Categoria**	Bug
Núcleo	IFTO	**Coordenação SIGA**	Codificação e Tecnologia
Início	22/02/2011	**Data prevista**	25/02/2011
Tempo gasto	0.1 horas		
Descrição	TCVL001- No formulário deste CDU possui um campo 'Descrição', porém este não está sendo descrito no fluxo do caso de uso nem na tabela de especificação dos dados.		
Observação	**A descrição era necessária nesse cadastro. Foi sugerida a alteração da descrição do CDU.**		

Tabela 18: Tarefa de bug de especificação do SIGA-EDU.
Fonte: *Redmine* do SIGA-EDU, 2011.

Título	TCVL005ManterEstruturaOrganizacional/6.0.RC02 [TM] Valores não relacionados		
Prioridade	Normal	**Categoria**	Bug
Núcleo	IFTO	**Coordenação SIGA**	Codificação e Tecnologia
Início	22/02/2011	**Data prevista**	25/02/2011
Tempo gasto	0.1 horas		
Descrição	TCVL005- O sistema permite o cadastro e alteração deste CDU com valores que não estão relacionados com os campos município e CEP. EX: Município = Salvador e CEP = 44200-000.		
Observação	**Este bug foi criado sem necessidade, pois esta exceção não estava prevista na documentação.**		

Tabela 19: Tarefa de bug de especificação do SIGA-EDU.
Fonte: *Redmine* do SIGA-EDU, 2011.

5 CONSIDERAÇÕES FINAIS

Algumas considerações devem ser levantadas em relação à análise do uso de metodologia de desenvolvimento de sistemas distribuídos, arquitetura de software e processo de correção de bugs no SIGA-EDU.

Observa-se que os desafios inerentes à metodologia de desenvolvimento distribuído são os fatores mais influentes em todo o processo e necessitam de boas práticas de programação, modelagem e especificação para que ocorram de forma contínua. Todos os aspectos tecnológicos e metodológicos são utilizados visando atenuar o impacto dos desafios da área de DDS. O SIGA-EDU conta com reuniões presenciais com frequência mensal, tecnologia de colaboração, tecnologias de comunicação, ciclo de vida incremental, padrões adotados em todos os processos (codificação, especificação, projeto, requisitos, testes entre outros), arquitetura de software que visa baixo acoplamento e alto índice de coesão, trabalho em equipe, gestão colaborativa entre outros.

No SIGA-EDU abordagem colaborativa é um fator necessário pela distribuição dos núcleos de desenvolvimento ser a nível nacional e a arquitetura de software em camadas auxilia no desenvolvimento, pois suas características são o baixo acoplamento e a alto nível e coesão durante o processo de desenvolvimento. Durante as fases do desenvolvimento, os papéis são bem definidos mesmo em nível de núcleo (os desenvolvedores têm suas tarefas dentro do sistema focadas em uma determinada área), o que contribui para a coesão de diminui o risco de conflito de artefato durante o processo.

As características do RUP aliadas ao modelo de ciclo de vida incremental abraçam as necessidades dos ambientes distribuídos de desenvolvimento. Assim como nesses ambientes, a unidade de análise (SIGA-EDU) define bem os aspectos relacionados aos papéis, atividades, artefatos e *workflows* que fazem parte do processo de desenvolvimento.

Em relação ao processo de correção e bugs, nota-se que grande parte dos problemas poderiam ser evitados em fases anteriores do processo de desenvolvimento por meio de especificação mais clara, atenção na construção dos

diagramas de classe, auxílio aos desenvolvedores inexperientes entre outros. Observa-se ainda que algumas das tarefas relacionadas a esta categoria poderiam ser classificadas de outra forma, a exemplo dos bugs de layout que são e sua grande maioria simples.

Portanto, conclui-se que o “pulo do gato” em ambientes de desenvolvimento distribuídos de software (DDS) é fazer com que os desafios elencados a respeito desta metodologia não se transformem em problemas e venha prejudicar o andamento do processo. É uma área recente que cresce tanto quanto se aperfeiçoa assim como os envolvidos nela.

6 REFERÊNCIAS BIBLIOGRÁFICAS

ANDERSON, Christiano. **Desenvolvimento Colaborativo.** Disponível em: <http://www.superdownloads.com.br/materias/desenvolvimento-colaborativo.html> Acesso em: 04 mar. 2010.

SUBVERSION. **Apache subversion.** Disponível em <http://subversion.apache.org> Acesso em 15 out. 2010.

ASTAH. **Astah* - UML and Mind Mapping Integrated Modeling Tool.** Disponível em: <http://astah.change-vision.com/en/index.html> Acesso em: 30 jul. 2010.

DEITEL, Harvey M. DEITEL, Paul J. **JAVA: Como Programar**. 6ª edição. São Paulo: Pearson Prentice-Hall, 2005.

ECLIPSE. **Eclipse.org home.** Disponível em: <http://www.eclipse.org> Acesso em: 20 jun. 2010.

ECLIPSELINK. **EclipseLink.** Disponível em: <http://www.eclipse.org/eclipselink> Acesso em: 22 jul. 2010.

FIGUEIREDO, A; SANTOS, D; TOMIMORI, E; SILVA, F; MIRANDA, I; **Softwares Livres: Vantagens.** Disponível em: <http://www.maringamanagement.com.br/include/getdoc.php?id=111&article=44&mode=pdf> Acesso em 12 set 2010.

FOWLER, M. **UML Essencial: Um breve guia para a linguagem-padrão de modelagem de objetos**. - 3ª Ed. – Bookman : Porto Alegre:, 2005.

GLASSFISH ENTERPRISE SERVER. **O que é o Glassfish?** Disponível em: <http://br.sun.com/practice/software/glassfish/index.jsp> Acesso em: 05 abr. 2010.

JAVAFREE. **Tutorial Java: O que é Java?** Disponível em: <http://javafree.uol.com.br/artigo/871498> Acesso em: 30 abr. 2010.

JAVASERVER FACES TECHNOLOGY. **JavaServer Faces.** Disponível em: <http://java.sun.com/javaee/javaserverfaces> Acesso em: 15 mar. 2010.

LARMAN, Craig. **Utilizando UML e Padrões: uma introdução à análise e ao projeto orientados a objeto e ao desenvolvimento interativo**. 3 ed. Porto Alegre: Bookman, 2007.

MANN, Kito D. **JavaServer Faces in Action.** Greenwich: Manning, 2005.

POSTGRESQL. **PostgreSQL: About.** Disponível em: <http://www.postgresql.org/about/> Acesso em: 20 abr. 2010.

PRESSMAN, R. **Engenharia de Software.** 6ª edição. McGraw Hill, 2004.

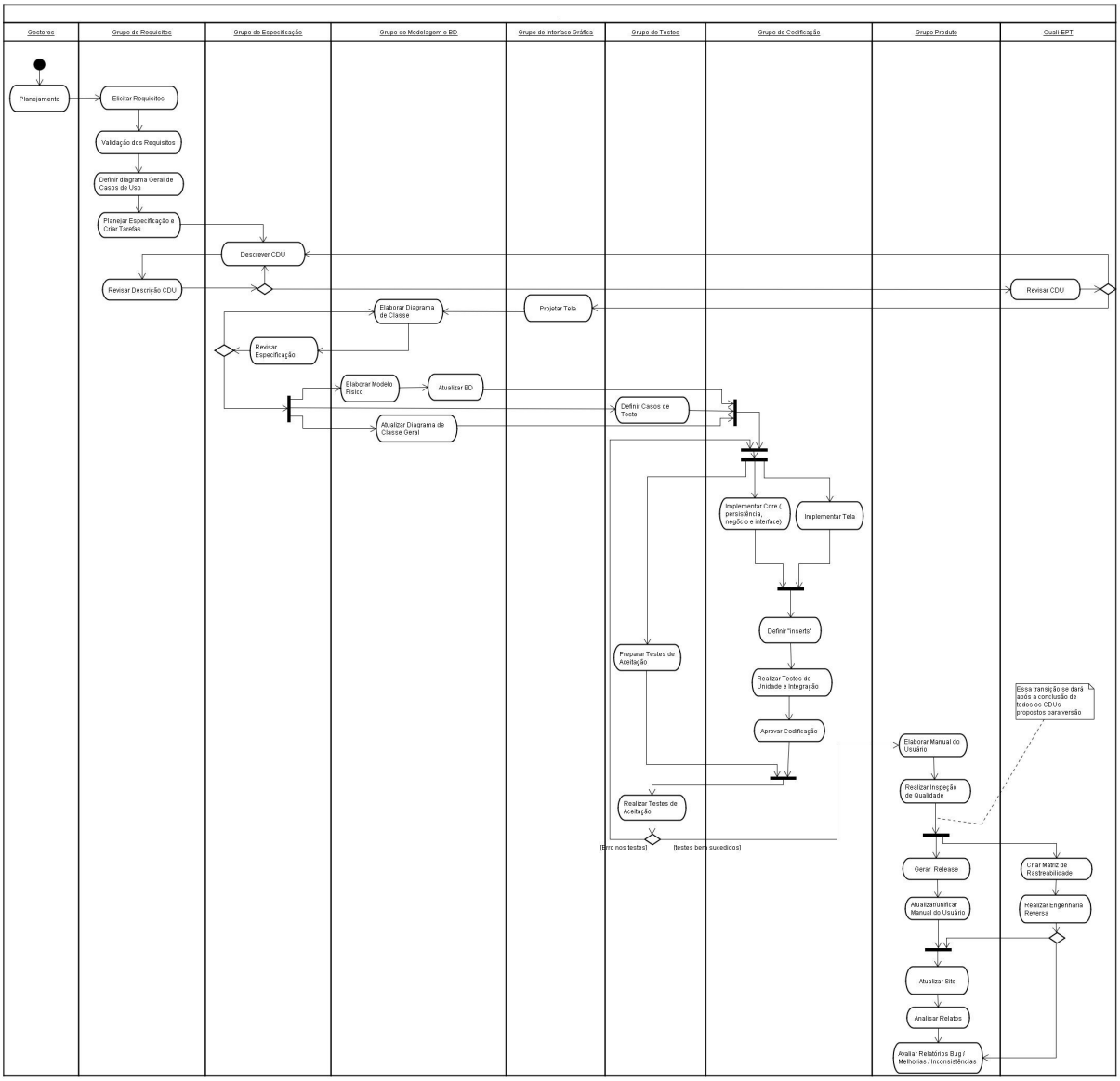
PRIKLADNICKI, R.; AUDY, J. **Desenvolvimento Distribuído de Software: Desenvolvimento de software com equipes distribuídas.** Rio de Janeiro: Elsevier, 2008.

REDE DE PESQUISA E INOVAÇÃO EM TECNOLOGIAS DIGITAIS. **Projeto SIGAEPT**. Disponível em: <http://www.renapi.gov.br/sigaept/o-projeto> Acesso em: 05 mar. 2010.

REDMINE DOS PROJETO DA RENAPI. **Redmine SIGA-EDU.** Disponível em: <https://redmine.renapi.gov.br> Acesso em: 13 mar. 2011.

SOMMERVILE, I. **Engenharia de Software**. 8ª Ed. São Paulo: Pearson, 2007.

7 ANEXOS



Anexo 1: Fluxo de desenvolvimento completo do SIGA-EDU.
Fonte: *Redmine* do SIGA-EDU, 2011.