CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA
MESTRADO EM TECNOLOGIA

RUBENS GONÇALVES JUNIOR

PROPOSTA DE ESTRUTURA DO PROCESSO DE ENSINO-APRENDIZAGEM DE DESENVOLVIMENTO DE SOFTWARE BASEADO EM PROCESSOS DE SOFTWARE E AMBIENTES DE APRENDIZAGEM COLABORATIVOS

SÃO PAULO

DEZEMBRO – 2007

RUBENS GONÇALVES JUNIOR

PROPOSTA DE ESTRUTURA DO PROCESSO DE ENSINO-APRENDIZAGEM DE DESENVOLVIMENTO DE SOFTWARE BASEADO EM PROCESSOS DE SOFTWARE E AMBIENTES DE APRENDIZAGEM

DISSERTAÇÃO APRESENTADA COMO EXIGÊNCIA PARCIAL PARA OBTENÇÃO DO TÍTULO DE MESTRE EM TECNOLOGIA NO CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA, NO PROGRAMA DE MESTRADO EM TECNOLOGIA: GESTÃO, DESENVOLVIMENTO E FORMAÇÃO, SOB ORIENTAÇÃO DO PROF. DR. ARISTIDES NOVELLI FILHO.

SÃO PAULO

DEZEMBRO, 2007

G635p Gonçalves Junior, Rubens

Proposta de estrutura do processo de ensino-aprendizagem de desenvolvimento de software baseado em processos de software e ambientes de aprendizagem / Rubens Gonçalves Junior. – São Paulo : CEETPS, 2007.
97 f.

Dissertação (Mestrado) – Centro Estadual de Educação Tecnológica Paula Souza, 2007.

1. Software – desenvolvimento. 2. Ensino Superior. 3. Tecnologia da informação e da comunicação. Título.

CDU 378:681.3.01

RUBENS GONÇALVES JUNIOR

PROPOSTA DE ESTRUTURA DO PROCESSO DE ENSINO-APRENDIZAGEM DE DESENVOLVIMENTO DE SOFTWARE BASEADO EM PROCESSOS DE SOFTWARE E AMBIENTES DE APRENDIZAGEM COLABORATIVOS

PROF. DR. ARISTIDES NOVELLI FILHO

PROF. DR. GETULIO DE SOUZA NUNES

PROF. DR. MAURICIO AMARAL DE ALMEIDA

SÃO PAULO, 13 DE DEZEMBRO DE 2007.

Dedicatória

A minha mãe Izaura e ao meu pai Rubens, pela luta, esforço e dedicação aos meus enriquecimentos físicos, morais e intelectuais.

A minha esposa Andréa pela paciência, cumplicidade, compreensão e carinho durante este percurso.

Aos meus avós pelos ensinamentos, e em especial a minha avó Cida por todo o carinho e compreensão por toda esta vida, e que infelizmente não pode estar presente em mais um obstáculo superado.

Aos meus irmãos Rodrigo e Natália, e a minha tia Maria Luiza que sempre estiveram na torcida para eu vencer mais esta empreitada.
Aos amigos pelo apoio, carinho e confiança durante a caminhada.

Agradecimentos

Agradeço a Deus pela força, saúde e perseverança para suplantar os desafios e alcançar o objetivo traçado.

A todos os funcionários do Programa de Pós-Graduação do Centro Paula Souza, a todos os docentes do curso, e em especial ao meu orientador, o Prof. Dr. Aristides Novelli Filho que norteou de forma magnífica o trabalho para que fossem alcançados os objetivos traçados.

A todos os colegas que do curso que contribuíram de algum modo, e em especial ao Adilson Antonio Barbosa.

Finalmente, agradeço aos colegas da Telefônica pela colaboração e compreensão prestada no decorrer do curso.

"COMEÇA FAZENDO O QUE É NECESSÁRIO, DEPOIS O QUE É POSSÍVEL, E, EM BREVE, ESTARÁS FAZENDO O IMPOSSÍVEL."

SÃO FRANCISCO DE ASSIS

RESUMO

O processo de globalização e o advento da internet tornaram o mercado de trabalho mais dinâmico e acirrado e, conseqüentemente, sistemas de software, cada vez mais sofisticados, tornam-se recursos indispensáveis para as organizações. Outra conseqüência deste fato é o aumento da procura por profissionais que possuam as competências necessárias para atuarem no desenvolvimento de software. Contudo, observa-se que as instituições de ensino superior estão encontrando dificuldades para formar profissionais adequadamente qualificados. Este trabalho tem como objetivo propor aperfeiçoamentos no processo de ensino-aprendizagem de cursos de nível superior que tem a formação de desenvolvedores de software como seu foco principal. Com a finalidade de atingir este objetivo, o trabalho: revê os principais processos de software, destacando-os como atividades tipicamente colaborativas; resume as principais disciplinas que formam o núcleo dos cursos de desenvolvimento de software; apresenta as tecnologias da informação e da comunicação que ajudam a promover atividades colaborativas; relaciona as competências necessárias ao profissional de desenvolvimento de software, divididas em genéricas e específicas; analisa quantitativamente alguns problemas dos cursos de graduação em nível superior; e, apresenta sugestões de como aprimorar a qualificação dos egressos, por meio de uma proposta de que sejam usados processos de software para nortear cursos de ensino-aprendizagem de desenvolvimento de software.

Palavras-Chave: Desenvolvimento de Software, Competências, Ensino-Aprendizagem, Tecnologias da informação e da Comunicação, Aprendizagem Colaborativa.

ABSTRACT

The globalization process and the internet advent turned the marketing more dynamic and incited and, hence, more sophisticated software systems became event more required to the organizations. Another consequence of this fact is the growing for professionals with the needed competences to develop software. However, colleges are facing some problems in order to prepare professionals accordingly. The goal is to propose improvements in the process of software developer's education and learning. To achieve this goal, this essay: review the main software development processes, as collaborative activities; summarize the main disciplines that encompasses the kernel of software development courses; present the information and communication technologies that help promoting collaborative activities; relates the required competences for professionals to develop software, both generic and specialized; analyze the issues of some college courses; and present suggestions to improve the qualifications of egresses, based on a proposal of being used in software processes to guide software development education and learning courses.

Keywords: Software Development, Competences, Education and Learning, Information and Communication Technologies, Collaborative Learning.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE GRÁFICOS

LISTA DE ABREVIATURAS E SIGLAS

CISC – Complex Instruction Set Computer

CMM – Capability Maturity Model

CRC – Class-Responsability-Colaborator

CSCL – Computer Support Cooperative Learning

DAS – Desenvolvimento Adaptativo de Software

DSDM – Método de Desenvolvimento Dinâmico de Sistemas

ERS – Especificação de Requisitos de Software

FDD – Desenvolvimento Guiado por Características

FTP – File Transfer Protocol

HD – Hardware

IBM – International Business Machines Corporation

ICQ – Programa para “bate-papo” da ICQ Corporation

IES – Instituições de Ensino Superior

IRC – Internet Relay Chat

LMS – Learning Management Systems

MIRC – Programa para “bate-papo” da mIRC CO Ltd

MSN – Messenger

NIED – Núcleo de Informática Aplicada à Educação

OSI – Open Systems Interconnection

PUC – Pontifícia Universidade Católica

RISC – Reduced Instruction Set Computer

RUP – Rational Unified Process

SGBD – Sistemas de gerenciamento de banco de dados

SQL – Structured Query Language

SW – Software

TCP/IP – Transmission Control Protocol/Internet Protocol

TIC – Tecnologia da Informação e da Comunicação

UML – Unified Modeling Language

UNICAMP – Universidade de Campinas

XP – Extreme Programming

WWW – World Wide Web

SUMÁRIO

INTRODUÇÃO

O mundo globalizado tem aumentado a competição entre as empresas. A necessidade de melhorar informações, processos internos e relacionamento com clientes e parceiros têm exigido das organizações altos investimentos em Tecnologia da Informação. Em particular, investimentos em sistemas de software de alta qualidade, desenvolvidos em prazos menores e a custos considerados razoáveis.

As empresas desenvolvedoras de software não estão à margem desta competição globalizada e investem para melhorar seus processos de desenvolvimento a fim de atender as exigências cada vez mais sofisticadas de seus clientes. Neste contexto, aumenta a demanda por profissionais com uma maior gama de competências em desenvolvimento de sistemas de software.

De acordo com pesquisa realizada pela Softex (2007), o setor de software no Brasil possui atualmente uma carência de aproximadamente 20 mil profissionais na área de desenvolvimento de software e dentro de 5 anos haverá cerca de 200 mil vagas em aberto na área. Segundo Antonioni (Softex, 2007), "a falta de profissionais qualificados será um dos entraves ao desenvolvimento da indústria brasileira de software, contudo ainda há tempo hábil para reverter esse quadro".

As instituições de ensino superior (IES) que atuam na formação de desenvolvedores de software devem alinhar-se às novas exigências do mercado e apresentar egressos com competências técnicas, gerenciais e de comunicação, entre outras, que possibilitem a eles adaptar-se rapidamente às organizações e evoluir em uma profissão em que novas tecnologias surgem com uma freqüência cada vez maior. Infelizmente, como menciona Wolynec (2007), as IES não têm alcançado plenamente os seus objetivos e, como conseqüência, não só na área de desenvolvimento de software, como em outras áreas de uso intenso de tecnologia, várias indústrias têm necessitado investir na formação de recém-formados porque eles não saem adequadamente preparados dos cursos

de graduação. Complementa, afirmando que algumas organizações até mesmo já cogitam recrutar pessoal no exterior para áreas estratégicas de tecnologia.

Este trabalho concentra-se na dificuldade que instituições de ensino superior têm para formar egressos que possam assumir em curto espaço de tempo responsabilidades no processo de desenvolvimento de software. Propõe para minimizar estas dificuldades e melhorar a qualificação dos egressos que os cursos de graduação possibilitem o envolvimento dos alunos, desde seu início, em processos de software desenvolvidos em ambientes de ensino-aprendizagem colaborativos que façam uso intenso de Tecnologias da Informação e da Comunicação.

Além do objetivo principal determinado para o trabalho, outros intermediários foram estabelecidos: realizar uma revisão dos principais conceitos de processo de software que são determinantes para a produção de software de qualidade; arrolar as principais disciplinas ministradas em cursos superiores que são determinantes para a formação de desenvolvedores de software; apresentar conceitos sobre aprendizagem colaborativa e realizar um breve resumo das tecnologias da informação e da comunicação que são utilizadas atualmente para impulsionar a aprendizagem e atividades colaborativas; apresentar as competências que são determinantes para o bom desempenho de um profissional de desenvolvimento de software; e, pesquisar sobre os principais problemas que afetam o processo ensino-aprendizagem de desenvolvimento de software em cursos superiores.

Este estudo justifica-se pela necessidade das instituições de ensino superior de formarem desenvolvedores de sistemas de software que atendam às atuais exigências da indústria de software brasileira. Entre suas principais contribuições este estudo apresenta sugestões que podem auxiliar as instituições a adaptarem os seus processos de ensino-aprendizagem para formar egressos com as competências exigidas pelo mercado de trabalho.

A principal motivação para a realização deste trabalho foi à dificuldade em trabalhar com recém-formados no decorrer das atividades desenvolvidas para a criação de um sistema de gestão parametrizável para algumas operadoras de

telefonia fixas da América Latina.

Para atingir os objetivos propostos, foram realizadas revisões bibliográficas sobre processos de software, sobre o ensino de desenvolvimento de software nos cursos de graduação, sobre educação colaborativa e sobre as competências necessárias aos desenvolvedores de software. Realizou-se também uma pesquisa junto a discentes de ensino superior para conhecer suas principais dificuldades e seus conhecimentos de tecnologias da informação e comunicação.

Este trabalho é constituído por seis capítulos, acrescidos da introdução e das considerações finais.

O capítulo 1 apresenta os fundamentos sobre o processo de software, e contextualiza desenvolvimento de software como atividade colaborativa.

O capítulo 2 disserta sobre o ensino de desenvolvimento de software nos cursos de graduação, em particular sobre as disciplinas que determinam as competências necessárias aos desenvolvedores de software.

O terceiro capítulo discute a aprendizagem colaborativa e as TIC's (Tecnologias da Informação e Comunicação) necessárias à sua implementação.

O capítulo 4 abrange o tema competências, desde a definição até as principais competências genéricas e específicas que estudantes de desenvolvimento de software devem assimilar.

O capítulo 5 apresenta os resultados da pesquisa realizada junto a discentes para verificar sua opinião sobre o processo de ensino-aprendizagem de que participam e para verificar seu conhecimento de tecnologias da informação e da comunicação.

O capítulo 6 apresenta a proposta de como estruturar o processo ensino-aprendizagem para incorporar nos cursos de graduação de desenvolvimento de software, desde seu início, processos de software desenvolvidos em ambientes de ensino-aprendizagem colaborativos que façam uso intenso de Tecnologias da Informação e da Comunicação.

Nas considerações finais discute-se se os objetivos foram atingidos e apresentam-se propostas de futuros trabalhos que podem complementar os

estudos realizados.

1 PROCESSOS DE SOFTWARE

Este capítulo tem como objetivo expor os principais conhecimentos e atividades que um profissional de desenvolvimento deve possuir e dominar para participar produtivamente de um projeto de sistema de software. Serão discutidos os aspectos mais importantes e que sejam comuns a diferentes processos de software; serão usados como base para discussão o Ciclo de Vida Clássico, o Processo Unificado e a metodologia “Extreme Programming”.

Finalizando o capítulo, e com a finalidade de harmonizá-lo ao contexto do trabalho, discute-se o desenvolvimento de software como uma atividade colaborativa.

1.1 Conceituação de Processos de Software

Os processos de software estão intimamente relacionados com tarefas destinadas a criação de sistemas com qualidade. Entre as várias definições para processo de software pode-se citar:

> O processo de software pode então ser definido como o conjunto de atividades, métodos e práticas que guiam a produção de software. Um processo efetivo de considerar a relação das tarefas necessárias, as ferramentas e métodos, e as habilidades, treinamento e motivação das pessoas envolvidas (HUMPHREY, 1998 apud Prikladnicki; Audy, 2008).

Ou:

> Um processo de software é como um arcabouço para as tarefas que são necessárias para construir software de alta qualidade (Pressman, 2002, p.17).

Sommerville (2003) relaciona as seguintes atividades, como essenciais e comuns a todos os processos de software:

- especificação de software: estabelecer as funcionalidades do software, bem como as restrições a que deve atender;
- projeto e implementação de software: produzir o software, de forma a atender as especificações estabelecidas;

● validação de software: assegurar que realize as funções conforme as solicitações do cliente;

● evolução de software: alterar o software para atender novas exigências e necessidades do cliente.

1.2 Modelos de Processo de Software.

Sommerville (2003) define um modelo de processo de software como uma representação abstrata de um processo de software. Peters e Pedrycz (2001) complementam: um modelo de processo de software estabelece uma estrutura para as principais atividades, entradas, saídas e restrições de um projeto.

As principais razões, segundo Pflleger (2004, p. 38), para se modelar um processo são:

> a) quando um grupo registra a descrição do processo de desenvolvimento e descreve seu entendimento sobre as atividades, os recursos e as restrições envolvidos no desenvolvimento do software;
> b) a criação de um modelo de processo ajuda a equipe de desenvolvimento a encontrar inconsistências, redundâncias e omissões no processo e em suas partes constituintes. À medida que esses problemas são percebidos e corrigidos, o processo se torna mais efetivo e concentrado na construção do produto final;
> c) o modelo deveria refletir os objetivos do desenvolvimento, tais como: construir software de alta qualidade, encontrar antecipadamente falhas no desenvolvimento e seguir as restrições do orçamento e do cronograma. À medida que o modelo é construído, a equipe de desenvolvimento avalia as atividades propostas mais adequadas a esses objetivos. Por exemplo, a equipe pode adicionar revisões nos requisitos, de modo que problemas possam ser encontrados e corrigidos antes do início da fase de projeto;
> d) todo processo deveria ser definido especialmente para a situação na qual ele será utilizado. A construção de um modelo de processo ajuda a equipe de desenvolvimento a entender quando essa adequação deve ocorrer.

Podem-se utilizar vários processos de software no desenvolvimento de um sistema de grande porte ou de um sistema complexo, pois deste modo, os vários processos de software seriam direcionados para as distintas partes do sistema, conforme a necessidade e perfil.

Entre os principais modelos de processo de software destacam-se:

● Ciclo de Vida Clássico;

- Processo Unificado.
- Modelo em espiral;
- Desenvolvimento Iterativo;
- Desenvolvimento Incremental;
- Prototipação;
- Metodologias Ágeis.

Serão detalhados a seguir os modelos que interessam para este trabalho: o Ciclo de Vida Clássico, o Processo Unificado e a metodologia ágil conhecida como “Extreme Programming”.

1.2.1 Ciclo de Vida Clássico

Sommerville (2003, p.37) cita Royce para dizer que “o primeiro modelo publicado de processo de desenvolvimento de software originou-se de outros processos de engenharia”. O ciclo de vida clássico é conhecido, também, com o nome de modelo em cascata, em razão da seqüência entre as fases. As fases e atividades do ciclo clássico variam muito de autor para autor, por exemplo, Braude (2000) sugere as seguintes atividades: análise de requisitos, projeto, implementação, integração e testes.

“O modelo em cascata ou modelo seqüencial linear sugere uma abordagem sistemática seqüencial para o desenvolvimento de software, que começa no nível de sistema e progride através da análise, projeto, codificação, teste e manutenção” (PRESSMAN, 2002, p.26).

A figura 1 ilustra o ciclo de vida clássico, agrupando atividades em 3 fases principais: Definição, Implementação e Manutenção.

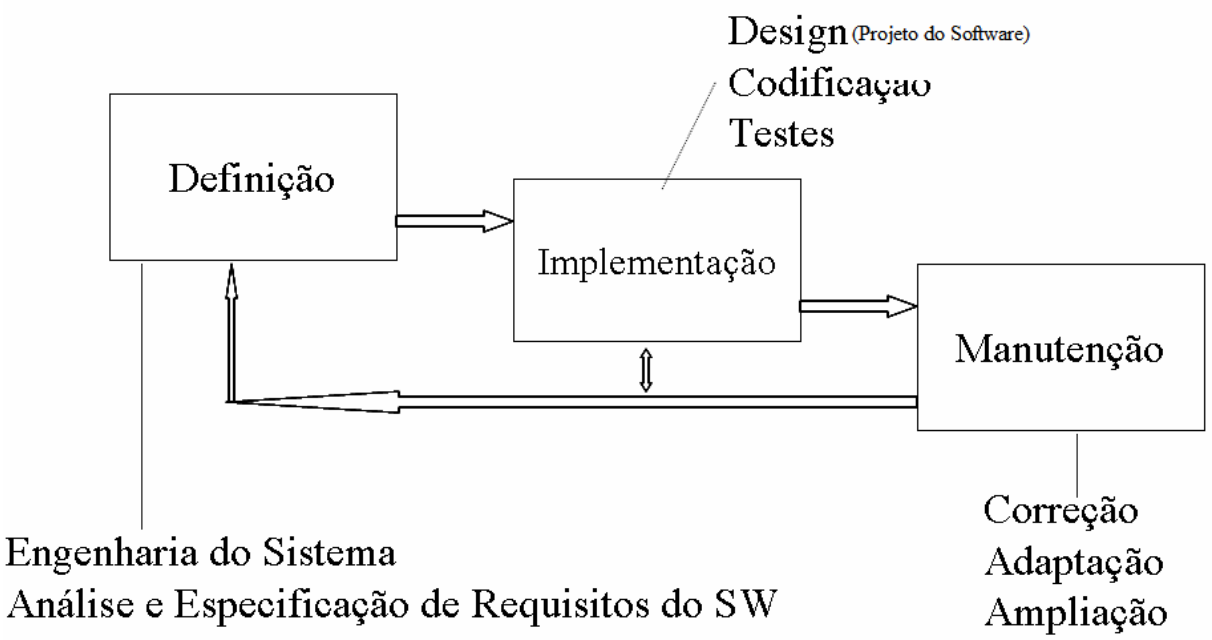


Figura 1- Ciclo de Vida Clássico
Fonte: o autor, baseado em Pressman (2002)

A fase de definição envolve as atividades de Engenharia do Sistema e a Análise e Especificação de Requisitos do Software que são detalhadas a seguir:

- Engenharia do Sistema. É responsável por estabelecer os requisitos para todos os elementos do sistema e posteriormente pela alocação de algum subconjunto desses requisitos para o software.
- Análise e Especificação de Requisitos do Software. Fase em que intensifica-se a definição dos requisitos e desenvolvem-se modelos que permitem avaliar funções necessárias, comportamentos desejados, desempenho esperado e interfaces exigidas.

A fase de implementação é compreendida pelas seguintes atividades:

- Design (Projeto) do Software. O projeto de software enfatiza a estrutura de dados, arquitetura do software, as interfaces com usuários e outros subsistemas, e os detalhes procedimentais. A finalidade é traduzir os requisitos para uma representação do software.
- Codificação. A codificação possui o objetivo de traduzir o projeto para uma linguagem de programação.
- Testes. Após a codificação, o código criado deve ser testado com a

finalidade de localizar possíveis erros e falhas, e assegurar que os resultados alcançados estejam em sintonia com os requisitos propostos.

A fase de manutenção é constituída das seguintes atividades:

- Correção. Caso sejam encontrados erros não localizados anteriormente, estes devem ser corrigidos para que o software produza os resultados esperados.
- Adaptação. Ao verificar pontos a serem aperfeiçoados com relação a desempenho ou mudanças por diferentes razões, como por exemplo, de mercado ou judicial, o software deve ser adaptado para acomodá-las.
- Ampliação. Ao constatar que novos requisitos são necessários, o software deve ser ampliado para incorporá-los e disponibilizar as novas funcionalidades aos usuários.

O modelo em cascata pode ser muito útil para auxiliar desenvolvedores a descrever que atividades devem ser realizadas. A sua simplicidade o torna fácil de explicar aos clientes não familiarizados com o desenvolvimento de software. Ele explica quais são os produtos produzidos em cada fase para que a fase posterior possa dar seqüência ao processo.

Neste modelo, a fase seguinte não deve ser iniciada até que a fase preliminar esteja finalizada. Na prática, estas fases se sobrepõem e trocam informações entre si. Em várias fases são encontrados problemas referente a fase anterior. O processo de software não é um modelo linear simples, envolve uma seqüência de interações das atividades de desenvolvimento.

Na utilização deste modelo o cliente precisa ser paciente, pois uma versão executável do programa não vai ficar disponível até o projeto terminar.

A maior dificuldade com o ciclo de vida clássico é a sua inflexível divisibilidade entre as fases. Torna-se muito complicado responder às novas necessidades do cliente, pois os acordos são selados na fase inicial do projeto. Portanto, o modelo ciclo de vida clássico deve ser implementado somente quando os requisitos forem bem delineados.

1.2.2 Processo Unificado.

Segundo Booch; Rumbaugh; Jacobson (2000), o Processo Unificado consiste em uma abordagem do ciclo de vida de um processo, especialmente adequada à UML *(Unified Modeling Language*). O objetivo do Processo Unificado é possibilitar a produção de software com alto índice de qualidade que atenda às necessidades dos usuários, conforme o planejamento prévio.

O Processo Unificado é um processo de engenharia de software que fornece uma abordagem disciplinada para atribuir atividades e responsabilidades dentro de uma organização de desenvolvimento, com a finalidade de garantir o desenvolvimento de softwares de alta qualidade que atendem às necessidades dos usuários finais dentro de orçamentos e prazos previsíveis (KRUTCHEN, 2003)

O Processo Unificado utiliza a UML para especificar, modelar e documentar o produto que está sendo elaborado. De acordo com Kroll e Krutchten em Javafree (2006) há três definições para o Processo Unificado:

> a) o Processo Unificado é uma maneira de desenvolvimento de software que é iterativa, centrada na arquitetura e guiada por casos de uso. É descrita em vários livros e artigos. Uma das maiores fontes de informações é o próprio produto IBM RUP, que contém guias detalhados, exemplos, modelos cobrindo todo o ciclo de vida do software;
> b) o Processo Unificado é um processo de engenharia de software bem definido e bem estruturado que define claramente quem é responsável pelo que, como as coisas devem ser feitas e quando fazê-las. O Processo Unificado também provê uma estrutura bem definida para o ciclo de vida de um projeto, articulando claramente os marcos essenciais e pontos de decisão;
> c) o Processo Unificado é também um produto de processo que oferece uma estrutura de processo customizável para a engenharia de software. O produto IBM RUP suporta a customização e autoria de processos, e uma vasta variedade de processos, ou configuração de processos, podem ser montadas nele. Essas configurações do RUP podem ser criadas para suportar equipes grandes e pequenas, e técnicas de desenvolvimento disciplinadas ou menos formais. O produto IBM RUP contém várias configurações e visões de processos prontas que guiam analistas, desenvolvedores, testadores, gerentes de projeto, gerentes de configuração, analista de dados, e outros membros da equipe de desenvolvimento em como desenvolver o software. Ele tem sido utilizado por muitas companhias em diferentes setores da indústria.

1.2.2.1 Fases do Processo Unificado.

> Uma fase é o período de tempo entre dois importantes marcos de progresso do processo, em que um conjunto bem-definido de objetivos é alcançado, artefatos são concluídos e decisões são tomadas em relação à passagem para a fase seguinte. (BOOCH; RUMBAUGH; JACOBSON, 2000, p.444).

Há quatro fases no Processo Unificado:

● concepção, que estabelece o caso de negócio para o projeto;

● elaboração, que estabelece um plano de projeto e uma arquitetura sólida;

● construção, que desenvolve o sistema;

● transição, que fornece o sistema a seus usuários finais.

Visualizam-se essas fases na figura 2:

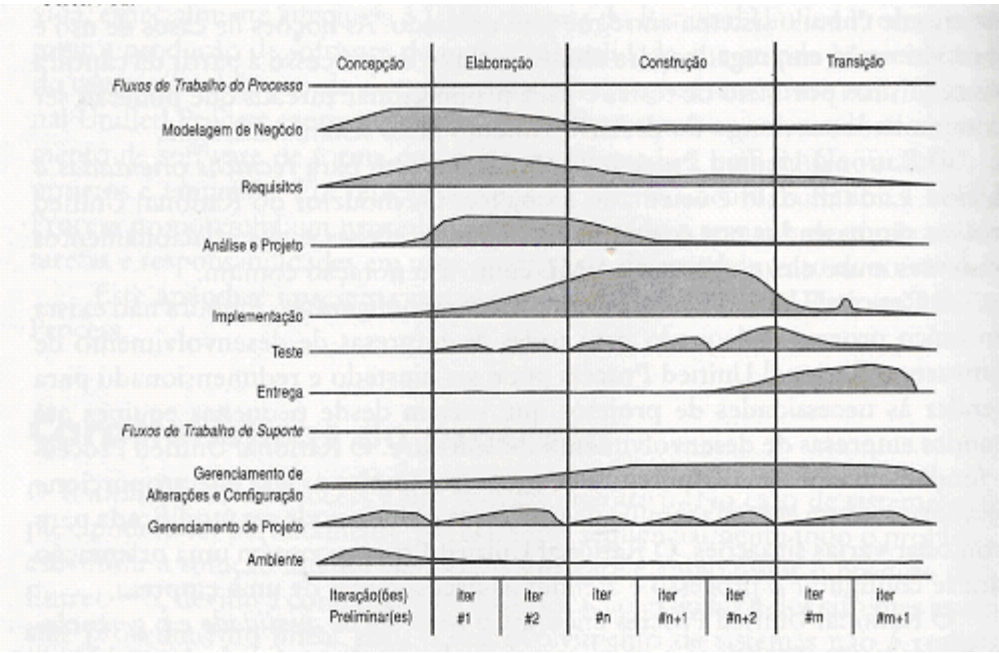


Figura 2 - Ciclo de vida de desenvolvimento de um software.
Fonte: BOOCH; RUMBAUGH; JACOBSON, 2000, p.444

No decorrer de cada fase, ocorrem inúmeras iterações. Uma iteração representa um ciclo completo de desenvolvimento, desde a captação de requisitos na análise até a implementação e a realização de testes, resultando em uma versão de projeto executável.

A fase de concepção estabelece o caso de negócio para o sistema e delimita o escopo do projeto. O caso de negócio insere critérios de sucesso, avaliação de riscos, viabilidade do projeto, os recursos necessários e um plano para cada fase, mostrando a programação dos principais marcos de progresso, além de constatar que o valor investido obterá retorno. Normalmente, nesta fase é criado um protótipo executável como teste para a concepção.

Ao encerrar a fase de concepção, são examinados os objetivos do projeto e os resultados obtidos e decide-se pela continuação ou não do desenvolvimento.

A fase seguinte é conhecida como elaboração, tendo como metas a revisão do escopo, a compreensão dos requisitos, o estabelecimento de uma arquitetura sólida, o desenvolvimento do plano de projeto e a eliminação dos riscos mais altos. No final da fase, analisam-se os resultados obtidos (escopo e objetivos do sistema minuciosamente detalhados, arquitetura selecionada, solução propostas para os riscos mais agudos etc.) e decide-se pelo prosseguimento ou não do projeto para a fase de construção.

Durante a fase de construção, é gerado um produto completo, de modo iterativo e incremental, sendo disponibilizado aos usuários. Isso implica na complementação dos requisitos, principalmente requisitos não funcionais e no estabelecimento de critérios de aceitação, dando corpo ao projeto e encerrando a implementação e o teste do produto.

No final da fase de construção, é decidido, se o software está apto para se tornar operacional.

A última fase é a transição, na qual o software é disponibilizado aos usuários. Comumente, após os testes realizados pelos usuários, surge a necessidade de fazer algum desenvolvimento adicional ou sanar alguns problemas. Normalmente, esta fase é iniciada com uma versão beta do produto, que depois é substituída pela versão de produção.

No final da fase de transição, é verificado se foram atingidas as metas estabelecidas para o ciclo do projeto e determina-se se outro ciclo de desenvolvimento deverá começar. As lições aprendidas no projeto deverão ser

avaliadas para aperfeiçoar o processo de desenvolvimento e serem utilizadas no próximo projeto.

1.3 Metodologias Ágeis

O termo metodologias ágeis tornou-se conhecido em 2001 quando Kent Beck e outros 16 especialistas em processos de desenvolvimento de software criaram o Manifesto Ágil.

De acordo com a Universidade de Brasília (2004), os conceitos chave do Manifesto Ágil são os seguintes:

- Indivíduos e Interações ao invés de ferramentas e processos;
- Software executável ao invés de documentação;
- Colaboração do cliente em vez de negociação de contratos;
- Respostas rápidas as alterações em vez de seguirem planos.

De acordo com Pressman (2006), o desenvolvimento ágil combina uma filosofia e um conjunto de diretrizes de desenvolvimento. A filosofia encoraja a satisfação do cliente e a entrega de software logo no princípio, a formação de equipes pequenas, métodos informais, artefatos de trabalho de engenharia de software mínimos e simplicidade global do desenvolvimento. Já as diretrizes estão baseadas na entrega do produto, e não na análise e projeto (porém estas atividades não são desencorajadas) e na comunicação permanente entre os desenvolvedores e clientes.

Atualmente, há um enorme interesse pelos modernos mecanismos de desenvolvimento de software, conhecidos como metodologias ágeis. Essas metodologias são uma resposta às chamadas metodologias tradicionais, e não rejeitam processos, ferramentas, documentação, negociação de contratos ou planejamento, somente dão a eles importância secundária e direcionam o foco às pessoas e às interações entre elas.

Há vários modelos ágeis de processo, dentre os quais vale destacar:

- Extreme Programming (XP)
- DAS – Desenvolvimento Adaptativo de Software
- DSDM – Método de Desenvolvimento Dinâmico de Sistemas

- Scrum
- Crystal
- FDD – Desenvolvimento Guiado por Características
- Modelagem Ágil

Na próxima seção será apresentada à metodologia *Extreme Programmming*, por se tratar da mais popular entre as metodologias ágeis.

1.3.1 Metodologia *Extreme Programming*

A *Extreme Programming* (XP) foi desenvolvido por Kent Beck e Ward Cunningham, sendo que é indicado para o trabalho de grupos pequenos ou médios, constituídos por até 10 integrantes.

De acordo com Pressman (2006), a XP é formada por um conjunto de regras e práticas que ocorrem em quatro atividades, a saber: planejamento, projeto, codificação e teste.

A figura 3 ilustra a metodologia XP. As atividades-chave possuem as seguintes características:

- Planejamento – A atividade de planejamento inicia com a criação de um conjunto de histórias que descrevem as características e funcionalidades necessárias para o software a ser gerado. Cada história é escrita pelo cliente e recebe um valor, ou seja, uma prioridade conforme o valor de negócio global para a instituição. Os membros da equipe avaliam cada história e atribuem um custo medido em semanas, sendo que caso a história ultrapasse três semanas de desenvolvimento, a mesma deverá ser fragmentada em histórias menores. Os clientes e a equipe XP trabalham em conjunto para decidir como agrupar histórias na próxima versão e como serão desenvolvidas em uma das seguintes formas: (a) todas as histórias serão implementadas imediatamente em poucas semanas, (b) as histórias com maior prioridade serão antecipadas e implementadas primeiro ou (c) as histórias de maior risco serão implementadas primeiro. Após a entrega da primeira versão do projeto, a equipe XP calcula a velocidade do projeto, ou seja, verifica a quantidade

de histórias do cliente que foram implementadas no decorrer da primeira versão. A velocidade do projeto pode ser utilizada para auxiliar na estimativa das datas de entrega e cronograma das versões posteriores, além de determinar se um comprometimento excessivo foi realizado para todas as histórias ao longo de todo o projeto de desenvolvimento.

● Projeto – O projeto XP segue o princípio da simplicidade, pois é preferível um projeto simples a um projeto complexo. A XP encoraja o uso de cartões CRC (Class-Responsability-Colaborator) como um mecanismo efetivo para raciocinar sobre o software no contexto orientado a objetos. Os cartões CRC identificam e organizam as classes orientadas a objetos que são fundamentais para o software atual. Os cartões CRC são os únicos produtos de trabalho do projeto que é feito como parte do processo XP. Caso um problema de projeto difícil é localizado como parte do projeto de uma história, a XP indica a geração rápida de um protótipo desta parte do projeto. O objetivo é reduzir o risco quando ocorrer a implementação e validar as estimativas originais relacionadas a história que contém o problema do projeto. A XP encoraja a refabricação, que é uma técnica de construção e projeto. Segundo Fowler apud Pressman (2006, p.65), “Refabricação é o processo de modificar um sistema de software de tal modo que ele não altere o comportamento externo do código, mas aperfeiçoe a estrutura interna. É um modo disciplinado de limpar o código (e modificar/simplificar o projeto interno), fato que minimiza as chances de introdução de defeitos. Em essência, quando você refabrica está aperfeiçoando o projeto do código depois que ele foi escrito”. Uma idéia fundamental no XP é de que o projeto acontece antes e após o início da codificação.

● Codificação – A XP recomenda que após as histórias terem sido geradas e o trabalho preliminar de projeto for finalizado, a equipe não desenvolva o código, e sim crie inúmeros testes unitários para exercitar cada história que deve ser incluída na versão atual. Após a geração dos testes unitários, o desenvolvedor está mais familiarizado e possui mais

conhecimentos para nortear o desenvolvimento e obter sucesso na execução dos testes unitários. Um conceito-chave no decorrer da atividade de codificação é a programação em pares. A XP recomenda que dois indivíduos trabalhem em conjunto em um computador na geração do código de cada história, revezando-se nas atividades. Quando os pares de programadores finalizam a atividade, o código gerado é integrado aos demais já existentes.

● Teste – Os testes unitários que são desenvolvidos devem ser implementados utilizando uma estrutura que possibilite a automatização, pois deste modo, encoraja os testes de regressão sempre que houver alterações no código. À medida que os testes unitários são finalizados com sucesso, podem-se realizar os testes de integração e validação com a finalidade de sinalizar a equipe de desenvolvimento como está o progresso do projeto ou indicar sinais de alerta caso haja divergências. Os testes de aceitação são criados pelo cliente e norteiam as características e funcionalidades do sistema que são visíveis e passíveis de revisão pelo cliente.

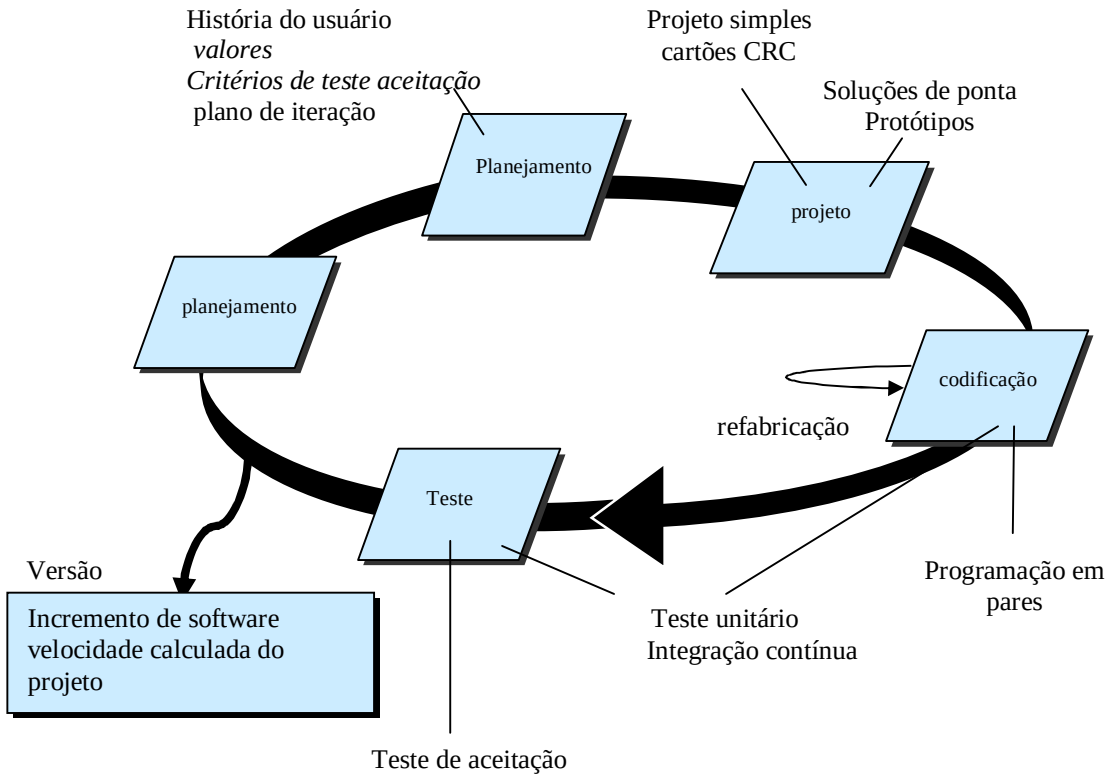


Figura 3 – O Processo Extreme Programming

Fonte: Pressman (2006)

As principais práticas que norteiam a XP são:

● Planejamento – estipula rapidamente o escopo das próximas versões, combinando as prioridades dos negócios e as estimativas técnicas;

● Pequenas Versões – a equipe deve disponibilizar rapidamente uma versão simples em produção, e posteriormente entregar novas versões em um curto espaço de tempo, ou seja, em dias ou semanas;

● Testes – os programadores escrevem testes de unidade continuamente, sendo criados antes da codificação e devem ser executados perfeitamente para que o desenvolvimento prossiga.

● Refabricação – os programadores aperfeiçoam o código e a arquitetura no decorrer de todo o desenvolvimento, porém não alteram seu comportamento externo;

● Programação em Pares – o código criado é realizado em pares, ou seja, dois desenvolvedores trabalham em conjunto no mesmo computador;

● Padronização do código – os programadores geram o código seguindo

as regras comuns com enfoque na comunicação por intermédio do código.

1.4 O Desenvolvimento de Software como uma Atividade Colaborativa.

De acordo com Souza (2007), a atividade de desenvolvimento de software é uma tarefa inerentemente colaborativa, já que apresenta as seguintes características:

- existe a necessidade de compreensão entre todos os envolvidos no processo, desde a elicitação de requisitos junto aos usuários, a modelagem, o desenvolvimento dos artefatos, os testes e a manutenção;
- há reuniões para a coordenação das atividades;
- há interação com outros sistemas;
- há comunicação formal e informal em grande parte do tempo entre os *stakeholders*[1].

Atualmente, o desenvolvimento de software não é mais centralizado em apenas um local, devido principalmente ao advento da internet e dos novos mecanismos de comunicação (e-mail, chat, videoconferência), e assim, espalha-se por inúmeros lugares, sendo que pode ser classificado como: desenvolvimento distribuído de software ou desenvolvimento global de software.

No desenvolvimento distribuído de software, as atividades são realizadas em vários lugares diferentes, localizados em cidades distintas, sendo que um fator primordial é a coordenação das atividades, como é o caso do Serpro (Serviço Federal de Processamento de Dados), onde os clientes se encontram em Brasília e o desenvolvimento é feito em outras unidades localizadas em outros centros.

O desenvolvimento global de software envolve a realização de tarefas dispersas em lugares diferentes, localizados em países distintos e até mesmo em continentes diferentes. Há grandes organizações que já utilizam deste

[1] Refere-se a todos os envolvidos em um processo, como por exemplo, clientes, colaboradores, investidores, fornecedores e comunidade.

artifício, como por exemplo: a Lucent que possui desenvolvimento nos EUA, Alemanha, Inglaterra, Índia e Brasil; a IBM que realiza o desenvolvimento nos EUA, Irlanda e Canadá, sendo que os testes são feitos na China; a Motorola que desenvolve nos EUA e testa no Brasil (Recife); a Dell que desenvolve nos EUA e no Brasil (Poá); a Siemens que desenvolve na Alemanha, Brasil, EUA, Polônia, e outros.

Há fatores relevantes no desenvolvimento global de software, dentre os quais vale destacar a distribuição, que pode levar ao desenvolvimento de software em tempo integral, ou seja, quando um analista para de desenvolver em uma localidade, outro indivíduo começa a desenvolver em outra localidade, procedimento conhecido como siga o sol.

Segundo Souza (2007), com o surgimento do desenvolvimento distribuído de software e do desenvolvimento global de software, ficou ainda mais evidente a necessidade de se trabalhar de forma colaborativa, visto que há problemas a serem sanados, tais como diferenças em fuso horário, feriados são distintos, diferenças de idioma, dificuldade de localizar o responsável por um determinado componente e culturas e comportamentos divergentes.

O desenvolvimento do trabalho colaborativo está se expandindo a cada dia por todos os continentes, principalmente com a expansão da internet, e podem ser observados vários casos clássicos públicos de sucesso, como o Linux e o Apache.

1.5 Considerações Finais do Capítulo.

Profissionais que dominem o processo de desenvolvimento de software e estejam treinados para trabalhar cooperativamente são os mais procurados pelas organizações desenvolvedoras. Portanto o processo de software, os modelos de processo de software e o desenvolvimento de software como atividade colaborativa devem integrar o processo formativo definido pelas IES nos cursos direcionados ao desenvolvimento de software.

2 O ENSINO DE DESENVOLVIMENTO DE SOFTWARE EM CURSOS DE GRADUAÇÃO

Este capítulo apresenta os principais referenciais estabelecidos para cursos direcionados ao desenvolvimento de software, enfatizando as disciplinas que mais contribuem para a aquisição de competências necessárias aos profissionais que atuam em projetos de sistemas de software. O capítulo é concluído com considerações de como práticas colaborativas podem contribuir com o ensino de desenvolvimento de software.

2.1 Regulamentação dos Cursos de Computação e Informática

Os cursos direcionados à computação e à informática são regulamentos pelo Ministério da Educação através das diretrizes curriculares que são aprovadas pelo Conselho Nacional da Educação.

As diretrizes curriculares são criadas após discussões e troca de idéias realizadas por meio de workshops de computação que são gerenciados pela Sociedade Brasileira de Computação. Estes workshops contam com a presença de inúmeros professores renomados que contribuem com suas opiniões, conhecimentos e experiências com o intuito de aperfeiçoar o processo de ensino-aprendizagem.

Os cursos da área de computação e informática possuem como objetivos a formação de profissionais altamente capacitados para o desenvolvimento tecnológico da computação (software e hardware) com o intuito de atender as necessidades da sociedade, para a aplicação das tecnologias da computação no interesse da sociedade e para a constituição de docentes para o ensino médio e profissional. Para que os objetivos dos cursos sejam alcançados, os profissionais devem adquirir inúmeras competências, e conseqüentemente necessitam de uma grade curricular que efetivamente possa prepará-los.

Os programas das disciplinas relacionadas neste capítulo foram baseados nas diretrizes da Sociedade Brasileira de Computação (2006) e de projetos de

cursos da Universidade Tecnológica Federal do Paraná (2006) e da Universidade Federal de Uberlândia (2007). Serão apresentadas divididas em três grupos: as essenciais para a formação de desenvolvedores de software, as que formam a base tecnológica do desenvolvedor e as que complementam a formação, tanto sob o aspecto humanístico como organizacional.

2.2 Disciplinas Essenciais a Formação do Desenvolvedor

Este tópico apresenta as disciplinas que são essenciais à formação de desenvolvedores de software, enfatizando a contribuição de cada uma delas para a formação de profissionais aptos a participarem de projetos de sistemas de software.

2.2.1 Programação

A programação, normalmente conhecida como programação de computadores, é uma atividade direcionada a resolução de problemas através de software com qualidade.

No estudo das linguagens de programação deve-se dar ênfase aos temas funcionais e estruturais das linguagens de programação, em detrimento aos detalhes de sintaxe. Devem ser destacados temas como: associação, atribuição, chamada de procedimento, envio de mensagens, passagem de parâmetros, herança, agregação, abstração, composição, polimorfismo, encapsulamento, comandos de entrada e saída de informação, compilação, tipos de dados, variáveis locais e globais, constantes, operadores aritméticos, operadores relacionais, estruturas de decisão, estruturas de repetição, comandos de desvio, matriz, vetor, registro, acesso a banco de dados, ferramentas de desenvolvimento de imagem, recursos de desenvolvimento de áudio.

No desenvolvimento de algoritmos e análise de estrutura de dados deve-se dar atenção especial ao estudo de técnicas de especificação, projeto e validação de programas.

2.2.2 Computação e Algoritmos

Os programas de computador estão sedimentados em três conceitos teóricos importantes: algoritmos, modelos de computação e linguagens formais. Um algoritmo é um método abstrato, mas bem estabelecido para a solução de um problema em tempo finito, sendo que o mesmo é executado em algum tipo de máquina. Os modelos de computação são as distintas máquinas onde são processados os algoritmos. O meio pelo qual se unem estes dois conceitos são as linguagens formais, que possibilitam a expressão de um determinado algoritmo para um determinado modelo de computação, sendo esta expressão intitulada de programa.

O estudo dos itens sintáticos e semânticos das linguagens formais é primordial para a atividade de programação.

2.2.3 Banco de Dados

As inovações tecnológicas têm facilitado às tarefas de armazenamento e gerenciamento dos dados. A área de banco de dados possui o intuito de disponibilizar soluções para este problema. Atualmente, qualquer organização, seja pequena, média ou grande, necessita de banco de dados para armazenar as suas informações, e posteriormente, manuseá-los.

O ensino em banco de dados deve considerar dois fatores principais: o conteúdo do curso e a possibilidade de associá-lo com outras disciplinas. Dentre os assuntos a serem discutidos vale destacar: organização, modelagem, integridade, armazenamento, distribuição, empacotamento e os sistemas de gerenciamento de banco de dados (arquitetura, interfaces, linguagens de interação, processamento de consultas, controle de concorrência, recuperação, segurança, indexação, gerenciamento de buffers e arquivos), distinguir os sistemas de armazenamento de dados, diferenciar as formas de modelagem de banco de dados, modelo entidade relacionamento (MER), normalização,

recuperação de transações concorrentes, regras de integridade, deadlocks, gatilhos, segurança, visões, privilégios de acesso, criptografia de acesso e replicação de dados. Também vale mencionar as linguagens de consultas, que realizam manipulações na base de dados, sendo que a mais comumente utilizada é a SQL (*Structured Query Language*), ou seja, Linguagem de Consulta Estruturada.

Atualmente, há vários bancos de dados que podem ser utilizados no processo de ensino, dentre os quais se podem mencionar: SQL Server, Oracle e DB2.

2.2.4 Engenharia de Software

> A Engenharia de Software compreende um conjunto de disciplinas técnicas, sociais e gerenciais que sistematizam a produção, a manutenção, a evolução e a recuperação de produtos intensivos em software. Estas atividades ocorrem dentro de prazos estabelecidos e gastos estimados, com progresso controlado e usando princípios, métodos, tecnologias e processos em contínuo aperfeiçoamento. Os produtos criados e mantidos de acordo com os conceitos da Engenharia de Software garantem qualidade satisfatória, operam de forma positiva e podem evoluir constantemente, adaptando-se as inúmeras alterações que surgem no cotidiano.

O ensino de Engenharia de Software em cursos de graduação pode dar origem a várias disciplinas com distintas ênfases. A ênfase pode ser dada em diferentes fases do processo de desenvolvimento e manutenção de software, como: engenharia de requisitos, análise, arquitetura e projeto, programação, testes, manutenção, garantia de qualidade e gestão do processo de software.

Há inúmeros assuntos de grande valia a serem tratados pela Engenharia de Software como: técnicas de análise de requisitos, métricas de software, análise de risco, CMM, modelos de processo de software e UML. É importante destacar que esses temas devem estar relacionados com outras disciplinas, como por exemplo: banco de dados, interface homem-máquina, redes, dentre outras.

2.2.5 Sistemas Multimídia

A capacitação de profissionais capazes de gerar programas de ação multimídia e que verdadeiramente se adaptem aos meios computacionais atualmente disponíveis precisa de um grupo mínimo de disciplinas de graduação. A computação multimídia é o resultado da junção de matérias que lidam com técnicas e conceitos referentes ao mundo visual e auditivo, como a computação gráfica, a computação sônica e a criação de peças multimídia.

A computação gráfica deve ser apresentada ao egresso em sua forma canônica, de forma que possa abranger as transformações geométricas, a visualização em 3D, a modelagem de objetos, os sistemas de cores, a iluminação, a textura e os elementos da animação.

A computação sônica, que é classificada como a contrapartida auditiva da computação gráfica, trata a natureza da forma sônica, os algoritmos fundamentais para a construção de formas sônicas, as técnicas de processamento de sons digitais, as linguagens para síntese de áudio e para a manipulação de sons, além de algumas noções básicas sobre sistemas musicais e linguagens auditivas em geral.

A aplicação da matéria para a geração de peças multimídia deverá interligar os conhecimentos citados acima à tecnologia disponível (Java, Midi, JavaSound) para definir as bases da elaboração criteriosa e fundamentada de programas que tragam soluções em níveis perceptivos superiores no que se refere a uma lógica de senso comum das percepções visual e auditiva.

2.2.6 Interface Homem-Máquina

Os profissionais da área de Computação criam produtos destinados a públicos específicos com as mais variadas habilidades técnicas e perfis sócio-culturais. Os produtos devem ser inseridos de modo natural nas atividades de trabalho de seus usuários. Para que isto possa ocorrer, o profissional deve conhecer profundamente a estrutura do trabalho realizado pelos usuários e, analisar os pontos de inserção de tecnologia com base no perfil do usuário, avaliar as suas implicações bem como re-projetar as formas corrente de executar o trabalho. Nesse sentido, tem surgido cada vez mais a preocupação

dos profissionais em fazer uma união perfeita das ferramentas e ambientes computacionais aos usuários, às suas atividades e às suas aspirações sociais.

No decorrer de todo o processo de desenvolvimento de uma interface de usuário, a maior preocupação é com a usabilidade do sistema interativo. Os usuários são os responsáveis pela determinação se um sistema interativo será ou não bem sucedido. Normalmente, os usuários preferem os sistemas fáceis de aprender e utilizar, mesmo que tenha uma diminuição das funcionalidades, a usar sistemas com funcionalidade poderosa, porém com uma interface pobre e de difícil uso.

Dentre os tópicos que podem ser apresentados aos estudantes pode-se citar: a engenharia cognitiva, as diretrizes para os projetos de interfaces e a engenharia semiótica.

2.3 Disciplinas de Formação Tecnológica

As disciplinas de tecnologia básica são de grande importância, pois formam a base tecnológica do desenvolvedor e possui uma interação com as disciplinas essenciais na realização de tarefas direcionadas ao desenvolvimento de software.

2.3.1 Arquitetura de Computadores

O termo arquitetura de computadores refere-se às características existentes em um projeto de hardware destinado a executar as tarefas definidas em um programa escrito em alguma linguagem de programação. O conhecimento dos princípios básicos de funcionamento dos computadores e da tecnologia embutida neles possibilita uma utilização mais eficiente dos recursos e a determinação das classes de problemas que podem ser solucionadas com a tecnologia disponível.

Há vários temas de grande relevância a serem abordados, dentre os quais merecem destaque: bases de numeração, unidade de controle e processamento, arquitetura física de sistemas computacionais, modo de endereçamento, tipo de dados, conjunto de instruções, chamada de subrotinas, tratamento de interrupções, organização de memória, RISC, CISC, arquiteturas paralelas, multiprocessamento e análise de desempenho.

2.3.2 Sistemas Operacionais

Os Sistemas Operacionais tem a finalidade de gerenciar a operação de computadores de forma a disponibilizar a seus usuários flexibilidade, eficiência, segurança, transparência e compartilhamento de recursos.

Os recursos computacionais disponibilizados pelos Sistemas Operacionais são agrupados basicamente em quatro classes diferentes: processo, memória, armazenamento, entrada e saída. O gerenciamento de processos envolve conceitos de comunicação, sincronização, escalonamento, solução de conflitos e troca de contexto. O gerenciamento de memória envolve temas sobre endereçamento, hierarquias de memória e memória virtual. O gerenciamento de arquivos trata de assuntos sobre diretórios, estrutura de endereçamento e acesso, segurança, concorrência e proteção. O gerenciamento de entrada e saída abrange conceitos sobre interrupções, dispositivos, interfaces e controladores de acesso.

Com o avanço dos sistemas computacionais, e consequentemente uma expansão das redes de computadores, outros temas a serem abordados são: interconexão de computadores, protocolos de comunicação, chamada de procedimentos remotos, comunicação em grupo, arquivos distribuídos, resolução de nomes, coordenação distribuída, paginação de memória, tolerância a falhas, escalamento de processos.

O conhecimento destes recursos é essencial para determinar o ambiente em que deverá ser processado um determinado sistema de software.

2.3.3 Redes de Computadores

As Redes de Computadores são formadas por computadores agrupados através de sistemas de comunicação, sendo que assim torna-se possível compartilhar recursos de hardware e de software, possibilitando a troca de informações entre seus usuários.

Os conhecimentos que devem ser apresentados por intermédio desta disciplina são os seguintes: meios de transmissão e suas características, arquitetura e topologia de redes de computadores, protocolos de comunicação, padrões, modelo OSI, interligação de redes locais, intranets, protocolo TCP/IP, gerenciamento e segurança e implementação de serviços em sistemas operacionais.

Assim como para a disciplina sistemas operacionais, o conhecimento de redes de computadores é essencial para determinar o ambiente em que deverá ser processado um determinado sistema de software.

2.3.4 Sistemas Distribuídos

Os Sistemas Distribuídos são sistemas constituídos por computadores fracamente acoplados, interconectados por rede que disponibilizam serviços e permitem acesso e manejarem dados e recursos compartilhados.

Os principais temas a serem tratados na área de sistemas distribuídos são algoritmos distribuídos, sistemas operacionais e kernels, ambientes de programação e linguagens, confiabilidade (tolerância a falhas e segurança de dados), base de dados, sistemas multimídia, sistemas de tempo real.

Considerando-se que atualmente a maioria dos sistemas de software atua em ambientes distribuídos, esta disciplina tem se tornado de vital importância para a definição das arquiteturas desses sistemas.

2.4 Disciplinas Complementares à Formação do Desenvolvedor

Os profissionais devem adquirir uma formação complementar, ou seja, adquirir conhecimentos sob o enfoque humanístico e organizacional, tendo uma preocupação com valores como ética, empreendedorismo, preservação do meio ambiente e qualidade de vida.

Sob o aspecto organizacional, conhecimentos sobre a estrutura, transações e processos típicos das organizações, podem tornar-se essenciais para a formação de profissionais que atuam no desenvolvimento de sistemas de software tipicamente administrativos. Esses conhecimentos são ensinados em disciplinas típicas de cursos de administração.

Como as disciplinas Empreendedorismo e Ética são importantes para a formação de desenvolvedores de software que atuam em qualquer área, e não apenas em sistemas organizacionais, serão comentadas a seguir.

2.4.1 Empreendedorismo

A formação de empreendedores é um processo de prover profissionais de áreas técnicas ou administrativas com os conceitos e habilidades para reconhecer e aproveitar as oportunidades de negócio, criando e gerenciando empreendimentos de sucesso, seja através do estabelecimento de uma empresa ou da atuação empreendedora em departamentos. Este processo inclui treinamento em reconhecimento de oportunidades, gerenciamento de recursos, análise e gerenciamento de risco, planejamento de negócio, marketing, técnicas de fluxo de caixa e conhecimento sobre normas e legislação para a definição de um empreendimento. São também desenvolvidas habilidades como: liderança, criatividade, trabalho em grupo e facilidade de comunicação.

2.4.2 Ética

Os computadores estão presentes na nossa sociedade, sendo que a sua

importância se tornou inquestionável. O estudo da ética na área de computação é o estudo das questões éticas que surgem como conseqüência do desenvolvimento e uso dos computadores e das tecnologias de computação.

Os tópicos a serem estudados devem evoluir à medida que a tecnologia avança e afeta o comportamento da sociedade. Os temas atuais que podem ser citados são: acesso não autorizado a recursos computacionais (hackers, vírus), direitos de propriedade de software (pirataria, a lei que regulamenta a propriedade do software), confidencialidade e privacidade dos dados e segurança.

2.5 Práticas Colaborativas no Ensino de Desenvolvimento de Software.

O ensino de Desenvolvimento de Software também deve ser trabalhado como uma atividade colaborativa, semelhante ao modo que é tratado pelo mercado de trabalho.

Há várias práticas colaborativas que podem ser introduzidas nos cursos de graduação direcionados ao desenvolvimento de software, como por exemplo, a inspeção de software, a decomposição modular e separação entre especificação das interfaces e sua implementação e a programação em pares.

A inspeção de software é a atividade na qual um indivíduo valida o software construído por outra pessoa, sendo que verifica se o software está em conformidade com a especificação técnica e dentro dos padrões de qualidade utilizados pela organização.

A decomposição modular e a separação entre especificação das interfaces e sua implementação é introduzida com a finalidade de diminuir a complexidade do problema, dividir funcionalidades por membros da equipe ou por equipes, e fazer com que haja uma comunicação, coordenação e colaboração entre as áreas de desenvolvimento e implementação.

A programação em pares sugere que o código gerado no projeto seja sempre implementado por dois indivíduos juntos, utilizando o mesmo computador e alternando na digitação.

É importante realçar que um dos tópicos que aumenta a valorização da atividade educacional colaborativa é o apoio à interação entre os estudantes de um curso, permitindo que os alunos menos qualificados possam aprender com os mais qualificados, que por sua vez, progridem ainda mais em suas habilidades.

3 APRENDIZAGEM COLABORATIVA E TECNOLOGIAS DA INFORMAÇÃO E DA COMUNICAÇÃO

O desenvolvimento de software como atividade inerentemente colaborativa exige profissionais que tenham aprendido a agir colaborativamente. Este capítulo discute a aprendizagem colaborativa e está dividido em três seções. A primeira seção define aprendizagem colaborativa, suas principais características, os principais obstáculos e os benefícios mais relevantes. A segunda seção trata dos sistemas CSCL, apresentando a definição e os principais mecanismos que oferecem para trabalho cooperativo. A terceira seção resume as principais tecnologias da informação e comunicação que possuem potencial para facilitar e impulsionar a aprendizagem colaborativa no ensino de desenvolvimento de software.

3.1 Aprendizagem Colaborativa

De acordo com Borges, Campos, Santoro e Santos (2003), a aprendizagem colaborativa é uma técnica ou proposta pedagógica na qual os estudantes ajudam-se no processo de aprendizagem, atuando como parceiros entre si e com o professor, com a finalidade de adquirir conhecimento sobre um dado objeto.

Corroborando com a idéia acima, pode-se definir aprendizagem colaborativa como um conjunto de métodos ou técnicas de aprendizagem para uso em grupos, assim como de estratégias de desenvolvimento de competências, onde cada membro do grupo é responsável, por sua aprendizagem e pela aprendizagem dos colegas.

A aprendizagem colaborativa enfatiza a importância do meio físico, das trocas sociais e da interação, conforme os estudos feitos por pesquisadores como Piaget e Vygotsky. Isto pode ser constatado nas palavras de Piaget apud Assis (2007, pág 131):

> (...) as relações entre o sujeito e seu meio consistem numa interação radical, de modo tal que a consciência não começa pelo conhecimento dos objetos nem pelo da atividade do sujeito, mas por um estado indiferenciado; e é desse estado que derivam dois movimentos complementares, um de incorporação das coisas ao sujeito, o outro de acomodação às próprias coisas.

Segundo Castorina (2000), Vygotsky surge com um pensamento histórico-social do desenvolvimento que, pela primeira vez, propõe uma visão da formação das funções psíquicas superiores como “internalização” mediada da cultura, e deste modo, postula um indivíduo social que não é apenas ativo mas sobretudo interativo.

O quadro 1 apresenta as principais características referente a aprendizagem tradicional e a aprendizagem colaborativa.

Aprendizagem Tradicional	**Aprendizagem Colaborativa**
Sala de aula	Ambiente de aprendizagem
Professor – autoridade	Professor – facilitador
Centrada no Docente	Centrada no Discente
Aluno – “Uma garrafa a encher”	Aluno – “Uma lâmpada a iluminar”
Reativa, passiva	Proativa, investigativa
Foco no produto	Foco no processo
Aprendizagem individual	Aprendizagem coletiva
Memorização	Transformação

Quadro 1
Fonte: Adaptado a partir de Kenski (2003).

Apesar de vários estudos estarem comprovando a eficácia da aprendizagem colaborativa, ainda ocorre grande resistência por intermédio de professores e estudantes, que segundo Panitz e Panitz apud Komosinski (2000) utilizam argumentos específicos para rejeitá-la.

Professores:

- Perda do controle em sala de aula. Alguns professores sentem que estão perdendo o controle ao designar mais responsabilidades aos egressos.
- Falta de autoconfiança. Compartilhar a responsabilidade da aprendizagem com os estudantes pode criar situações onde eles façam perguntas não antecipadas. Tais questionamentos podem deixar o professor numa situação incômoda, já que ele é conhecido como o "detentor da sabedoria".
- Medo de não conseguir cobrir todo o conteúdo. As interações em grupo, por definição, tomam mais tempo do que as aulas expositivas tradicionais.
- Falta de material preparado para uso em sala de aula. Quase todo material bibliográfico (livros, textos, apostilas, etc) estão dirigidos para a aprendizagem individual.
- Ego do professor. Vários professores superestimam sua importância e gostam de ser o centro das atenções.
- Falta de familiaridade com técnicas alternativas de avaliação. O medo está em não ser capaz de avaliar corretamente cada estudante, pois não haveria como saber exatamente qual é a participação efetiva de cada membro do grupo na execução das tarefas.
- Falta de familiaridade com metodologias de ensino-aprendizagem colaborativas. A adoção do ensino-aprendizagem colaborativo implica no domínio de técnicas de gerenciamento do relacionamento dos grupos de trabalho, uma vez que surgem dificuldades específicas como a influência exagerada de um membro sobre o grupo, ritmos de aprendizagem muito distintos entre os grupos, desentendimentos dentro do grupo e entre grupos.

Alunos:

- Falta de treinamento para adaptar-se a aprendizagem colaborativa.
- Falta de familiaridade com técnicas alternativas de avaliação. A transferência da nota do grupo para o indivíduo, pode não ser aceita por

alunos que valorizam ser avaliados individualmente.

● Resistência às técnicas de aprendizagem colaborativa. O método tradicional centrado no professor é compreendido pelos estudantes como sendo mais fácil, pois o esforço se concentra no docente, e o aluno assume uma atitude passiva e, portanto, mais confortável.

● Falta de familiaridade com técnicas colaborativas. Os estudantes possuem dificuldades em compreender a filosofia da aprendizagem colaborativa. Imersos em um sistema que encoraja a competitividade, lhes parece um contra-senso terem que compartilhar com os colegas suas estratégias de aprendizagem;

● Medo da perda de conteúdo e habilidade para alcançar notas altas. Os estudantes sentem-se inseguros nos seus processos de construção de conhecimento, pois não existe uma referência claramente determinada a ser seguida.

Além das resistências oferecidas por professores e alunos, administradores também podem ser fonte de resistência à implementação do ensino-aprendizagem colaborativo. Como, em geral, os administradores são professores que estão distantes do cotidiano do ensino, é natural que ofereçam resistência a modificações que podem não apresentar resultados satisfatórios de imediato.

Ainda, segundo Panitz e Panitz apud Komosinski (2000), baseando-se em estudos feitos por outros pesquisadores, apontam que há vários benefícios que podem ser alcançados por meio da aprendizagem colaborativa. Dentre estes, vale mencionar os que são relevantes para este trabalho:

● aumenta a retenção de conteúdos por parte do estudante

● desenvolve habilidades de comunicação

● desenvolve habilidades de interação social

● cria um ambiente de aprendizagem ativo, envolvente e exploratório

● estimula a formação de equipe e uma abordagem baseada em equipe para a solução de problemas

● os estudantes exploram soluções alternativas para os problemas em um

ambiente seguro, pois as opiniões são do grupo e não dos indivíduos

● estimula o pensamento crítico e ajuda os estudantes a esclarecer suas idéias através da discussão e do debate

● incrementa as habilidades de auto-gerenciamento

● estabelece uma atmosfera de cooperação e ajuda entre todos os estudantes

● encoraja técnicas alternativas para avaliação dos estudantes

● incentiva e desenvolve relacionamentos interpessoais

● incentiva o compartilhamento de técnicas de resolução de problemas desenvolvidas pelos colegas

● estudantes aprendem a criticar idéias e não pessoas

● maior habilidade dos estudantes em observar situações a partir da perspectiva de outros

● cria uma atitude mais positiva em relação às pessoas envolvidas (professores, estudantes e administradores), pois todos passam a manter relacionamentos mais próximos

● atende às diferenças de estilo de aprendizagem entre os estudantes

● a sala de aula se parece com as situações reais de vida social e de emprego

● os estudantes exercitam papéis típicos existentes no mundo social e de trabalho.

A aprendizagem colaborativa tem ganho expansão e repercussão no mercado acadêmico, principalmente após o surgimento de tecnologias que podem contribuir no processo de ensino-aprendizagem, como os ambientes CSCL.

3.2 Sistemas CSCL

Segundo Levy (1999), sistemas CSCL (Computer Support Cooperative Learning) possuem dispositivos que permitem à discussão coletiva, a divisão de

conhecimentos, as trocas de saberes entre indivíduos, o acesso a tutores on-line aptos a guiar as pessoas em sua aprendizagem e o acesso a base de dados, hiperdocumentos e simulações.

De acordo com Borges, Campos, Santoro e Santos (2003), o CSCL é uma área de estudos que trata de formas pelas quais a tecnologia pode ajudar os processos de aprendizagem provenientes de esforços colaborativos entre estudantes trabalhando numa tarefa.

Ainda, segundo Borges, Campos, Santoro e Santos (2003), sistemas CSCL disponibilizam vários mecanismos de trabalho cooperativo. Entre os mecanismos mais comuns vale destacar:

- Envio de mensagens

  A colaboração pode ocorrer por intermédio da troca de mensagens entre os membros de um grupo, como por exemplo na criação de um algoritmo para solucionar um determinado problema.

- Coordenação de atividades

  Na aprendizagem colaborativa direcionada a um objetivo educacional, como desenvolvimento de projetos, a coordenação se refere à definição dos modos de trabalho e do acesso ao conhecimento compartilhado. A coordenação também está relacionada à implementação de guias que possam nortear o trabalho dos alunos e auxiliá-los a percorrer os caminhos necessários para a aprendizagem sobre as questões referentes aos projetos.

- Negociação e tomada de decisões

  Com relação à aprendizagem, a negociação deve ser analisada como um mecanismo auxiliar aos egressos para que tomem decisões sobre o planejamento e a execução das tarefas que levarão à criação da solução para o problema apresentado.

- Representação dos conhecimentos do grupo

  De acordo com a atividade colaborativa elaborada pelo grupo pode ser primordial que os indivíduos possuam métodos formais, estruturados, para representar um conhecimento, ou debater uma colocação, de

forma que todos os participantes tenham a oportunidade de compreender o que se está querendo comunicar.

- Compartilhamento de uma base de dados

  O compartilhamento de uma base de dados deve armazenar todas as informações geradas pela atividade colaborativa que está sendo desenvolvida em grupo, além de registrar o próprio processo da atividade em si. O conhecimento informal, ou seja, o registro das idéias, os fatos, as questões levantadas, os pontos de vista, as conversas, as discussões e as decisões também devem ser preservadas, já que estas experiências podem vir a serem aproveitadas na solução de novos problemas.

- Percepção da presença e das ações dos demais participantes

  A percepção é uma contextualização sobre as atividades e sobre o grupo como um todo: envolve saber quem é o grupo, quem são seus membros, onde estão localizados, quais tarefas estão executando, qual seu intuito, o conhecimento sobre o que aconteceu, o que vem acontecendo e o que está se passando no presente momento das atividades do grupo.

- Designação de Papéis

  Sistemas CSCL devem propiciar flexibilidade na definição de papéis e na designação destes aos membros do grupo. Os principais papéis que podem ser estabelecidos em um ambiente de aprendizagem colaborativa apoiada em computadores são os seguintes: coordenador, membro do grupo responsável por coordenar a execução de uma determinada tarefa; facilitador, membro do grupo que fornece meios para auxiliar os demais, pode ser o próprio professor ou instrutor; aprendiz, que aprende o conteúdo educacional; emissor e receptor, aprendizes que alternam-se como emissores e receptores de um conteúdo, sendo que o receptor deve tomar notas sobre o que o emissor está falando e então deve ser capaz de explicar este conteúdo; executor, aprendizes que possuem

> mais aptidão para executar atividades assumem o papel de executores, ao passo que os mais tímidos assumem papel de observadores e produtores da memória do grupo; analisador, crítico, argumentador, revisor, redator, papéis desempenhados por membros do grupo, conforme a fase em que se localiza a tarefa cooperativa.

Corroborando com o pensamento de Levy (1999), percebe-se que a gama de possibilidades educacionais que as novas tecnologias oferecem são inúmeras, porém ainda não exploradas em todas as suas potencialidades.

3.3 Ferramentas Tecnológicas

Este tópico apresenta e comenta algumas das principais ferramentas tecnológicas existentes atualmente que podem contribuir em ambientes colaborativos. Em geral essas ferramentas tecnológicas são implementadas na internet, porém podem ser exploradas também em redes locais por meio de intranets.

3.3.1 Correio Eletrônico

O correio eletrônico é uma das formas de comunicação mais utilizada na Internet. Possibilita a troca de mensagens com qualquer usuário do mundo e o envio de arquivos anexados em diversos formatos (texto, imagem, áudio, filmes, etc). Por se tratar de uma forma de comunicação assíncrona, permite que as mensagens recebidas sejam analisadas com maior tempo e cuidado, antes de serem respondidas, proporcionando um tipo de interação mais ponderada com o tutor e com os demais estudantes, além de ter a vantagem de enviar e receber as mensagens conforme a sua disponibilidade.

3.3.2 Fórum.

É utilizado para debates ou reuniões com um mesmo objetivo, sendo classificado como assíncrono (*off-line*). É baseado no correio eletrônico, porém com a diferença de que os indivíduos inscritos no fórum são simultaneamente emissores e receptores, e a comunicação é coletiva. Os fóruns são montados por pessoas que desejam debater e trocar idéias sobre um tema em comum, normalmente formado por um grupo restrito.

O uso do fórum em sistemas colaborativos pode ser uma ferramenta eficaz para promover reuniões participativas, contando com a presença de docentes e discentes que estão interessados no assunto em pauta.

3.3.3 Telnet.

Trata-se de um sistema que possibilita a oportunidade de um usuário qualquer acessar um computador do outro lado do mundo, ou seja, a uma distância enorme e manuseá-lo como se fosse seu próprio computador.

Ao utilizar este recurso, estabelece-se uma comunicação de duas vias de modo online com o hospedeiro remoto, e desta forma, as digitações realizadas no micro será direcionado para o hospedeiro.

Através deste sistema pode realizar consultas ou pesquisas sobre temas distintos. É o caso dos sistemas que permite acesso às bibliotecas públicas ligadas à Internet, como as do Vaticano, do congresso americano e do museu do Louvre.

3.3.4 *Internet Relay Chat* – IRC (bate-papo).

É um serviço de comunicação síncrona, ou seja, os participantes encontram-se *online*. É muito popular, sendo conhecido como *chat*, e permite a troca de mensagens escritas, podendo ser implementado através de um programa específico ou ser integrado em páginas web.

Os programas mais conhecidos são: MIRC, ICQ e MSN (*messenger*).

Essas ferramentas promovem discussões interativas entre dois ou mais integrantes simultaneamente, disponibilizam um ou mais canais para discussão de diversos temas e permitem que se enviem mensagens para todos os usuários conectados num canal ou apenas para um usuário, privativamente.

3.3.5 FTP *(file transfer protocol*).

É o protocolo utilizado na Internet para a transferência de arquivos entre um servidor e o computador de um usuário. A finalidade de usar o protocolo é para que programas emissores e receptores possam verificar se as informações foram recebidas corretamente.

As transferências podem ser realizadas em dois sentidos: do servidor para o usuário (*download*) ou do computador do usuário para o servidor (*upload*).

Esse recurso possibilita aos estudantes o *download* dos grandes bancos de dados quando localizarem coleções de imagens, artigos, livros, apostilas, vídeos, músicas, etc., que podem contribuir no processo de aprendizagem.

3.3.6 Videoconferência.

É um tipo de comunicação síncrono, ou seja, *online*, que proporciona aos usuários uma interação em tempo real. Essa interação permite a utilização de áudio e vídeo. Desta forma, a videoconferência é a mídia utilizada em ensino a distância que mais se aproxima do ensino tradicional, pois possibilita aos participantes verem uns aos outros e ouvirem-se simultaneamente.

O custo da implantação da videoconferência é relativamente alto, devido aos equipamentos de ponta e dos circuitos dedicados que são utilizados, além de necessitar de conexão de rede de média a alta velocidade.

Atualmente, a Internet disponibiliza aos usuários vários software de videoconferência, dentre os quais destacam-se: *NetPhone*, *Netmeeting* e o *Messenger*. Outro software conhecido e de grande utilização é o *CU-SeeMee*

devido a sua facilidade de aquisição, pois pode ser utilizado gratuitamente e por um período de tempo indeterminado.

3.3.7 Video/Áudio sob Demanda.

Este recurso possibilita assistir a vídeos ou ouvir áudios previamente gravados e armazenados em um servidor. O uso desta ferramenta é similar a um videocassete, ou seja, pode-se avançar, pausar ou retroceder. Com a implantação do sistema streaming, ou seja, fluxo contínuo, o usuário não precisa carregar todo o arquivo de vídeo/áudio antes de iniciar o seu uso, otimizando o tempo de espera, principalmente com conexões muito lentas. A utilização desta ferramenta exige grande espaço de armazenamento de vídeo/áudio digitalizado no servidor.

3.3.8 *Whiteboard.*

É um tipo de serviço que compartilha documentos através da Internet. Possibilita que um grupo de usuários geograficamente distantes faça um trabalho cooperativo, em que um mesmo documento é apresentado na tela e pode ser editado. A inserção de um texto ou gráfico por um dos participantes é propagada instantaneamente aos demais.

Várias instituições têm utilizado esta ferramenta para a tutoria de seus alunos, em complemento às aulas presenciais. No entanto, é um sistema ainda com um custo elevado para que usuários comuns tenham acesso como ativos. Em geral, os *Whiteboards* são usados em sala de aula pelos docentes, o que permite que os alunos assistam às aulas em casa, em tempo real.

3.3.9 *World Wide Web.*

Este serviço, um dos principais responsáveis pela popularização da

Internet é conhecido como *homepages*, *sites* ou *web pages* e integra praticamente todos os demais serviços, utilizando uma interface gráfica amigável que combina páginas com hipertextos (palavras ligadas a outras páginas) com multimídia (hipermídia) e permite a visualização de páginas contendo texto formatado, imagens, animações, vídeos e sons, além de programas interativos (Java, Javascript).

3.3.10 Ambientes de Aprendizagem.

Os ambientes de aprendizagem conhecidos como LMS – Learning Management Systems (Sistemas de Gerenciamento de Aprendizagem) – incorporam a maioria dos recursos anteriormente citados para implementar sistemas CSCL.

Há vários destes ambientes virtuais de aprendizagem que podem ser utilizados pelas instituições de ensino superior, como o WebCT, o AulaNet, o TeleEduc e o Moodle.

A seguir, serão relacionados alguns dos recursos dos ambientes WebCT e TeleEduc para ilustrar os ambientes de aprendizagem.

WebCT é um software desenvolvido pelo Departamento de Ciência da Computação da Universidade da Columbia Britânica que fica localizada no Canadá e possui recursos que possibilitam a geração de ambientes educacionais através da Internet. Há várias instituições no Brasil que utilizam este software, dentre as quais vale mencionar: Universidade de São Paulo, Universidade de Campinas, Universidade Presbiteriana Mackenzie e Pontifícia Universidade Católica de Campinas. Este ambiente possui inúmeros recursos, dos quais se destacam:

- conteúdo do curso: disponibiliza informações e material sobre o curso como glossários, sumários, perguntas freqüentes, e outras fontes de referência;
- estrutura do ambiente: disponibiliza informações sobre as ferramentas do ambiente;

● ferramentas de comunicação: além das ferramentas tradicionais como fórum e *chat*, também disponibiliza o *whiteboard;*

● exercícios: possibilita a criação, a edição e o gerenciamento de exercícios;

● diário de bordo: é um espaço reservado para as anotações dos alunos que podem ser lidas e comentadas pelos docentes;

● ferramentas de avaliação: as avaliações podem ser realizadas de três modos: auto-avaliação (possibilita que o aluno verifique seus conhecimentos tendo retorno imediato do resultado de sua performance); avaliação (o docente disponibiliza um arquivo em padrão texto e a correção é feita manualmente); testes *online* (funcionam como as auto-avaliações, porém nesta situação o docente deve validar a nota gerada pela correção automática das questões, sendo que se pode verificar o status do aluno com relação ao andamento do teste e ao final, constatar o tempo que o aluno gastou);

● trilha de progresso: permite a monitoração e o progresso dos alunos. Indica também a data do primeiro e último acesso, tempo gasto e porcentagem de páginas visitadas. Outra trilha do progresso disponível indica o número total de acessos para cada página do curso. Isso pode ser usado para inferência sobre o nível de interesse ou dificuldade da página;

● mural: consiste num espaço reservado para todos os participantes disponibilizarem informações, consideradas relevantes, no contexto do curso.

O Teleduc foi criado pelo NIED (Núcleo de Informática Aplicada à Educação) da Unicamp e é um ambiente norteado a criação, administração e participação de cursos na Web. Este ambiente está sedimentado em atividades teóricas-práticas on-line, comunicação entre os participantes e discussão de temas. Há várias ferramentas disponibilizadas pelo Teleduc, sendo que as mais relevantes são as seguintes:

● Agenda: é a página de entrada do curso com a programação do dia, ou seja, é o local onde informam as datas, as atividades a serem realizadas, novos conteúdos que estão disponíveis;

● Fórum de Discussão: possibilita a discussão sobre um determinado assunto, sendo possível acompanhar todo o processo, verificando as mensagens já postadas e inserindo novas;

● Bate-Papo: permite uma conversa em tempo real entre os discentes e os docentes, sendo que as datas e horários que os professores participarão são cadastrados na agenda;

● Correio: sistema de correio interno do sistema, sendo que todos os participantes podem utilizá-los.

● Diário de Bordo: permite que os discentes descrevam as suas percepções e questionem sobre o processo de aprendizagem, bem como insiram relatos de experiências profissionais e suas expectativas em relação ao curso. Os docentes podem comentar as informações inseridas.

● Portfólio: este recurso possibilita que arquivos, textos ou endereços da Web sejam armazenados. Esses dados podem ser particulares, compartilhados somente com os docentes ou com todos os participantes. Cada participante pode visualizar o portfólio dos colegas e fazer comentário sobre eles caso entenda ser relevante.

3.4 Considerações Finais do Capítulo

Considerando-se as características das TIC's apresentadas neste capítulo pode-se constatar o alto potencial que possuem para serem utilizadas em atividades colaborativas em ambientes de ensino-aprendizagem. Porém, considerando o grau de complexidade para organizar o uso desses recursos de modo produtivo por professores, alunos e administradores, pode-se inferir o grau de dificuldades a ser enfrentado para introduzi-los nas instituições de ensino superior; principalmente em cursos direcionados ao desenvolvimento de software que precisarão desses ambientes organizados tanto para a

colaboração entre os alunos na elaboração de projetos como para a colaboração entre os alunos e professores para o gerenciamento dos projetos e sua avaliação.

4 COMPETÊNCIAS EM DESENVOLVIMENTO DE SOFTWARE.

A necessidade de sistemas de software cada vez mais complexos implica em desenvolvedores com competências cada vez mais abrangentes. Este capítulo conceitualiza e caracteriza competência. Relaciona as competências necessárias ao profissional de desenvolvimento de software, dividindo-as em genéricas e específicas.

4.1 Competências: Definição, Características e Elementos Envolvidos.

Atualmente, as organizações interpretam como competências as qualificações do colaborador em executar determinadas tarefas com habilidade. Porém, competências é um tema muito mais complexo e desafiador, sendo que não há um consenso com relação a sua definição, devido as distintas origens e inúmeras abordagens, conforme será apresentado no decorrer deste capítulo.

Segundo Fleury e Fleury (2006), competência é um saber agir responsável e reconhecido, que implica mobilizar, integrar, transferir conhecimentos, recursos e habilidades, que agreguem valor econômico às organizações e valor social ao indivíduo.

Com a finalidade de facilitar o entendimento, é apresentada a figura 4.

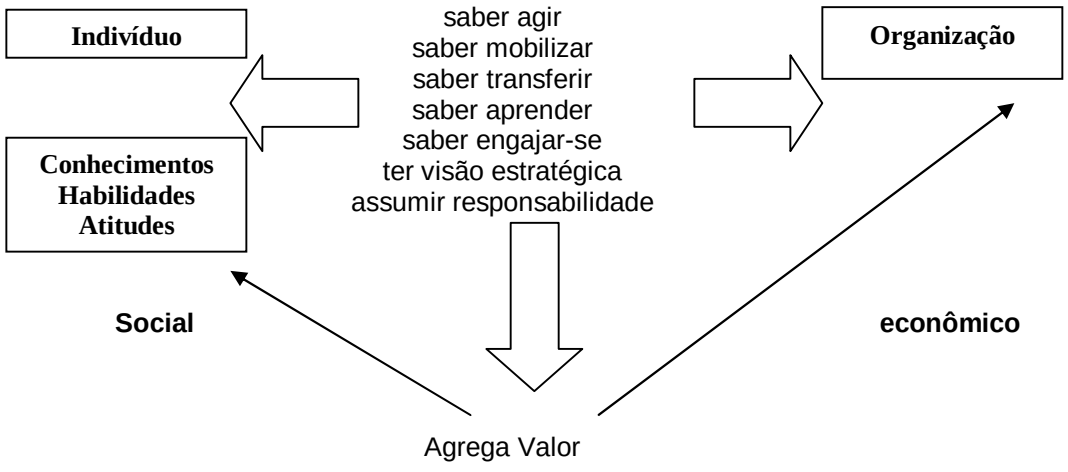


Figura 4 – Competências como fonte de valor para o indivíduo e para a organização

Fonte: Fleury e Fleury (2006)

De acordo com Sveiby (1998, p.42) ao descrever a competência de um indivíduo, podem-se considerar os seguintes elementos mutuamente dependentes:

> a) conhecimento explícito. Envolve o conhecimento dos fatos e é adquirido, principalmente pela informação, quase sempre pela educação formal;
> b) habilidade. É a arte de 'saber fazer', envolvendo proficiência prática – física e mental – e é adquirida por meio de treinamento e prática;
> c) experiência. Estimulada pela reflexão sobre os erros e sucessos passados;
> d) julgamentos de valor. São resultantes das percepções do que o indivíduo acredita estar certo, traduzidas pelas crenças e valores;
> e) rede social. É formada pelas relações do indivíduo com outros seres humanos dentro de um ambiente e uma cultura transmitidos pela tradição.

Diante dos elementos mencionados anteriormente, o termo competência deve ser tratado como individual, ou seja, característico de cada pessoa, segundo o ambiente no qual se insere. Assim, todas as pessoas desenvolvem sua própria competência, por meio de treinamento, de prática, de erros, de reflexão e de repetição.

De acordo com Gramigna (2002), o desenvolvimento de competências do ser humano pode está sedimentado em conhecimento, habilidade e atitudes. O Conhecimento corresponde aos saberes assimilado; quanto maior a quantidade de conhecimentos assimilados pelo indivíduo, mais a competência se fortalece e possibilita que o profissional enfrente os inúmeros desafios do seu cotidiano. A habilidade corresponde a agir com talento, com capacidade e técnica para solucionar os problemas e alcançar resultados satisfatórios; significa demonstrar que sabe. As atitudes são constituídas ao longo da vida através de um conjunto de valores, crenças e princípios; atitude está relacionada com "querer ser e querer agir", ou seja, está diretamente associada ao envolvimento e comprometimento do indivíduo.

Segundo Boyatzis (2004), um erro muito comum é pensar que, apenas adquirindo novos conhecimentos, um indivíduo irá se tornar um melhor profissional. Para ser eficiente é necessário usar este conhecimento para fazer

com que as coisas aconteçam.

A partir das abordagens apresentadas no decorrer deste capítulo, chega-se a definição de competência que será utilizada como referência no contexto deste trabalho. Assim, tem-se que: competência individual é a união de habilidades, conhecimentos e atitudes colocadas em prática para solucionar problemas, e conseqüentemente alcançar resultados satisfatórios.

A classificação das competências, assim como a definição de competências, não possui consenso. A partir dos estudos e análises realizados, adotou-se a seguinte classificação:

- competências técnicas: conhecimentos específicos sobre o trabalho que deve ser realizado, sendo que estão associadas às atividades de natureza técnica, normalmente pautadas no uso de tecnologia;
- competências intelectuais: capacidade de reconhecer e definir problemas, equacionar soluções, pensar estrategicamente, introduzir alterações no processo de trabalho, atuar preventivamente, transferir e generalizar conhecimentos;
- competências organizacionais: capacidade de auto planejamento, auto organização, estabelecimento de métodos próprios, gerenciamento do tempo e espaço de trabalho;
- competências comunicativas: capacidade de expressão e comunicação com seu grupo, superiores hierárquicos ou subordinados, de cooperação, trabalho em equipe, diálogo, exercício da negociação e de comunicação interpessoal;
- competências comportamentais: iniciativa, determinação, criatividade, vontade de aprender, abertura às mudanças, consciência da qualidade e das implicações éticas do seu trabalho;
- competências sociais: capacidade de utilizar todos os conhecimentos alcançados, por meio de inúmeras fontes e recursos diferenciados nas mais diversas situações encontradas no mundo do trabalho, ou seja, capacidade de transferir conhecimentos da vida cotidiana para o ambiente de trabalho e vice-versa;

● competências de liderança: são competências que reúnem habilidades pessoais e conhecimentos de técnicas de influenciar e conduzir indivíduos para diversos fins ou objetivos na vida profissional ou social;

● competências gerenciais: são competências que compreendem habilidades pessoais e conhecimentos de técnicas de administração ou gerenciamento, de aplicação em situações de direção, coordenação ou supervisão.

4.2 As Competências Direcionadas ao Desenvolvimento de Software.

As competências apresentadas ganharam destaque nos últimos anos e devem ser adquiridas pelos profissionais que desejam ser bem sucedidos, independentemente da sua área de atuação. Contudo, o foco deste trabalho está relacionado com o desenvolvimento de software, e a seguir, são apresentadas as competências genéricas e específicas inspiradas nas sugestões da Sociedade Brasileira de Computação (2006), e nos projetos de cursos da Universidade Tecnológica Federal do Paraná (2006) e da Universidade Federal de Uberlândia (2007).

4.2.1 Competências Genéricas.

As competências genéricas são aquelas que não estão associadas diretamente às atividades de desenvolvimento de software, contudo podem e muito colaborar junto às competências específicas para superarem os desafios no desenvolvimento de software, ou seja, são complementares, e são classificadas da seguinte forma:

● Intelectuais

- compreensão das diferentes atividades envolvidas no desenvolvimento de um software;
- integrar recursos de tecnologia da informação aos negócios das organizações;

- auxiliar os profissionais das outras áreas a compreenderem de que modo os sistemas de informação podem contribuir para as áreas de negócio;
- conhecer a organização dos sistemas de informação nas empresas;
- reconhecer os elementos do domínio do problema relevantes para a aplicação;
- conhecer os objetivos e fronteiras de um sistema de software;
- identificar quais as reais necessidades de informação de uma organização;

● **Comunicativas**

- comunicar-se bem de forma oral e escrita;
- conhecer e aplicar técnicas para o bom relacionamento entre pessoas e manter a eficácia das técnicas de trabalho em equipe;
- proficiência no idioma Inglês;

● **Comportamentais**

- disposição e postura de permanente busca da atualização profissional;
- compreender a importância da gestão dos colaboradores em um gerenciamento de projetos;
- agir de forma racional e não por impulso ou emocionalmente;
- respeitar os princípios éticos da área de computação;

● **Organizacionais**

- definir metas, objetivos, organizar recursos e definir ações antes de agir;
- atribuir tarefas certas a seus subordinados, delegando autoridade e cobrando responsabilidade;
- ser capaz de realizar / exercer várias tarefas / funções;

● **Sociais / Éticas**

- ter uma visão contextualizada da área de sistemas de informação em termos políticos, sociais e econômicos;
- compreender o mundo e a sociedade na qual está inserido;
- agir de acordo com os valores moralmente aceitos pela sociedade;

- agir com eqüidade e imparcialidade;

● **Liderança**

- possuir a capacidade de atrair, motivar e mobilizar os demais colaboradores em função de alguma atividade a ser desenvolvida;
- saber harmonizar situações e idéias conflitantes;
- conseguir estimular a equipe a superar limites;

● **Gerenciais**

- possuir a capacidade de empreender, tanto interna como externamente a uma organização;
- compreender o potencial impacto da Internet sobre as organizações e desenvolver a capacidade de promover as mudanças necessárias na empresa para que esta possa se beneficiar das oportunidades oferecidas pela grande rede, para virtualização de produtos/serviços, processos empresariais e estrutura da organização produtiva;
- preocupar-se com o horizonte, com o futuro e o caminho a seguir;
- introduzir novas idéias, atitudes, valores, experiências, processos e procedimentos;
- saber ponderar e conduzir a negociação de forma que as partes saiam satisfeitas.

4.2.2 Competências Específicas.

As competências específicas são aquelas que são totalmente direcionadas para que os indivíduos possam realizar ações para o desenvolvimento de software, ou seja, estão diretamente acopladas há alguma tarefa que tenha a finalidade de produzir software. A seguir, são apresentadas as competências específicas subdivididas em:

● **Essenciais**

- familiaridade com as tecnologias e ferramentas de análise e projeto de software e o discernimento de como, quando e quanto utilizar tais ferramentas;
- analisar o desempenho de projetos e sistemas, propostos ou implementados, por meio de modelos analíticos ou simulações;
- analisar a determinação de requisitos que um projeto deve atender, documentando-os de forma clara, organizada e de fácil uso;
- planejar, supervisionar, elaborar e coordenar projetos de software;
- conhecer a técnica de projeto estruturado;
- conhecer os fatores de risco do desenvolvimento de software;
- desenvolver sistemas utilizando fundamentos da análise e projeto Orientado a Objetos;
- projetar sistemas de informação para a Web;
- conhecer técnicas de levantamento dos requisitos e coleta de dados;
- desenvolver sistemas segundo as metodologias de engenharia de software;
- resolver problemas através da implementação de algoritmos;
- conhecer técnicas de programação;
- implementar programas de computador em uma linguagem de programação;
- desenvolver aplicações que sejam executadas em dispositivos móveis;
- validação do software para funcionar conforme projetado, através da combinação de codificação, teste de unidades e teste integrado.

- conhecer os aspectos básicos de um Sistema de Gerenciamento de Banco de Dados (SGBD), incluindo a utilização de banco de dados em WEB e ambientes distribuídos;
- modelar sistemas de banco de dados;
- conhecer as potencialidades dos sistemas de banco de dados para manipulação de dados;
- projetar interfaces de usuário utilizando conceitos de interface humano-computador;
- conhecer a potencialidade e os recursos da multimídia para o desenvolvimento de aplicações;
- selecionar modelos de construção de Web (sites dinâmicos) baseados em tecnologia de objetos distribuídos;
- ter capacidade de projetar e configurar sistemas computacionais em que sejam exigidas definições de funções a serem implementadas em software/hardware, selecionando seus componentes básicos.

● Tecnológicas

- conhecer as características dos software para sistemas distribuídos;
- reconhecer os principais componentes de hardware e de software dos sistemas distribuídos;
- conhecer o funcionamento de rede de computadores;
- conhecer sistemas de comunicação wireless, seu funcionamento e limitações;
- desenvolver políticas de segurança;
- conhecer os conceitos de tarefa e de multiprogramação;
- saber gerenciar processos que envolvam tratamento de concorrência, interrupções, paralelismo, escalonamento, manuseio de erros e exceções.

4.3 Considerações Finais do Capítulo

As competências definidas neste capítulo para os desenvolvedores de software não são definitivas. Ampliaram-se ao longo dos últimos 60 anos,

quando surgiram os primeiros desenvolvedores de software. Continuarão evoluindo, acompanhando a expansão e a importância dos sistemas de software no cotidiano da sociedade moderna, em intervalos de tempo cada vez mais reduzidos, exigindo dos profissionais de desenvolvimento a reciclagem constante de suas competências.

5 A SITUAÇÃO DO ENSINO DE DESENVOLVIMENTO DE SOFTWARE EM NÍVEL DE GRADUAÇÃO

Com o objetivo de averiguar a situação do processo de ensino-aprendizagem de desenvolvimento de software nos cursos de graduação, foi realizada uma pesquisa de campo junto a discentes de quatro instituições da região da grande São Paulo. Este capítulo sumariza e comenta os resultados.

5.1 Sobre a Pesquisa

A pesquisa foi aplicada junto aos discentes de cursos orientados ao desenvolvimento de software da região da grande São Paulo. O instrumento de pesquisa utilizado foi um questionário, que se encontra no Apêndice A.

A pesquisa foi realizada com alunos de cursos de graduação de quatro instituições de ensino superior privadas. Foram escolhidas instituições privadas pela facilidade de acesso a elas, por responderem por 70% dos alunos do país e pelo fato de nelas estarem concentrados os alunos com maior deficiência de formação básica e que mais podem ser beneficiados com novas metodologias de ensino.

Optou-se por trabalhar com vários cursos relacionados ao desenvolvimento de software, disponibilizados por instituições distintas, pois desta forma, não haverá uma homogeneidade quanto aos métodos expostos por instituições e professores aos seus discentes.

A amostra foi formada por um total de 200 alunos, sendo que um requisito obrigatório para ser selecionado foi a freqüência às aulas.

Para evitar a duplicidade de respostas foi criado um número seqüencial único que identificou os questionários e permitiu a eliminação de formulários repetidos. Os dados colhidos foram tabulados e analisados através da planilha eletrônica Excel com a finalidade de determinar os resultados obtidos e possibilitar a geração de gráficos.

5.2 Interpretação e Discussão dos Dados

Este tópico possui a finalidade de apresentar os dados tabulados através de gráficos que facilitem a compreensão, bem como, discuti-los.

5.2.1 Características do Processo de Ensino-Aprendizagem do Desenvolvimento de Software.

Esta primeira parte do questionário procurou identificar como está o processo de ensino de desenvolvimento de software, ou seja, procurou localizar quais são os principais problemas enfrentados pelos egressos para adquirirem as competências necessárias.

Questão 1: Percentualmente, as atividades realizadas no curso se concentraram em:

Teoria ( ) Exercícios ( ) Projetos ( )

Objetivo: Averiguar a distribuição das atividades propostas pelos docentes.

Discussão: Conforme o gráfico 1, é possível constatar que há uma concentração em teoria, já que para os discentes esta possui uma taxa de 53%, enquanto que exercícios e projetos, aparecem, respectivamente, com percentuais de 26% e 21%.

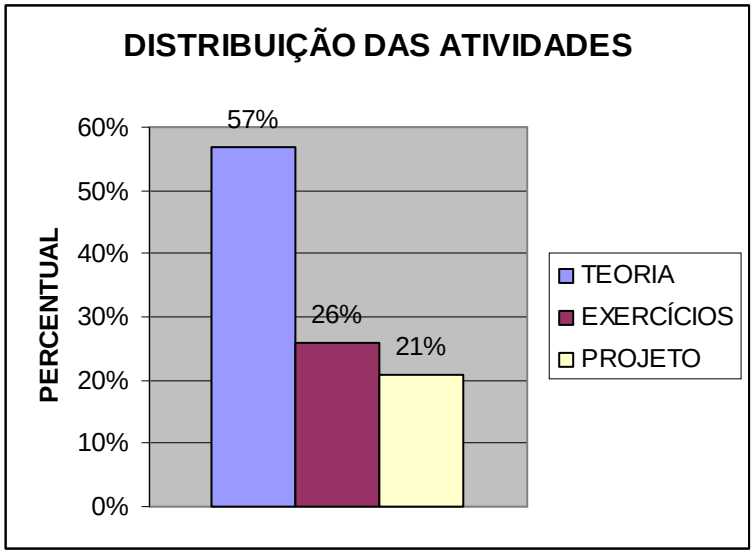


Gráfico 1 – Distribuição das Atividades

Questão 2: Quais os recursos que mais colaboram no processo de ensino-aprendizagem na área de processo de software?

Objetivo: Verificar a opinião dos alunos sobre quais recursos mais contribuem para sua aprendizagem.

Discussão: De acordo com o gráfico 2, os recursos que mais colaboram na aprendizagem de desenvolvimento de software são o laboratório com 37% e a internet com 18%.

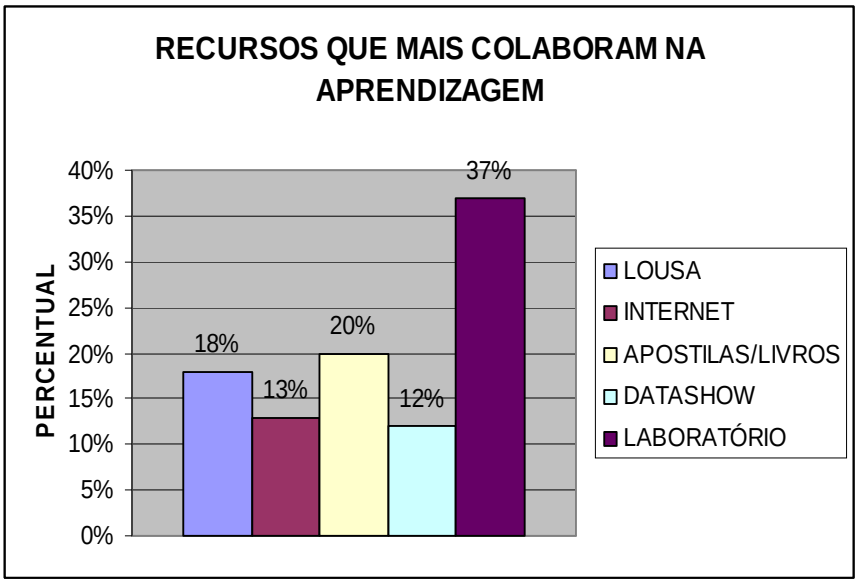


Gráfico 2 – Recursos que mais colaboram na aprendizagem

Questão 3: Considerando que a participação efetiva em Projetos de Sistemas de Software é uma das formas de experimentação que mais agrega valor para a formação dos alunos, avalie como cada um dos itens abaixo pode ser prejudicial para o sucesso?

Legenda: (a) Nada (b) Pouco (c) Médio (d) Muito

Objetivo: Verificar, entre os itens:

- Inexperiência para determinar requisitos
- Não existência de (ou dificuldade de acesso a) cliente/usuários
- Pré-requisitos de programação inadequados
- Pouco tempo de disponibilidade para envolvimento no projeto
- Dificuldade para agendar reuniões entre os alunos

- Baixa disponibilidade de recursos (hw/sw) para desenvolver projetos
- Falta de conhecimento de ambientes de desenvolvimento
- Falta de preparo para trabalhar em equipes

quais são mais prejudiciais na opinião dos alunos para o sucesso em Projetos de Sistemas de Software.

Discussão: Para os alunos todos os itens apresentam percentuais semelhantes como potenciais riscos nos projetos de desenvolvimento de sistemas de software, como pode ser observado no gráfico 3 que ilustra o percentual indicado pelos alunos para a opção “Muito”; destaca-se, porém, o “Pouco tempo para envolvimento no projeto” - (84%) - como o maior risco para o sucesso dos projetos.

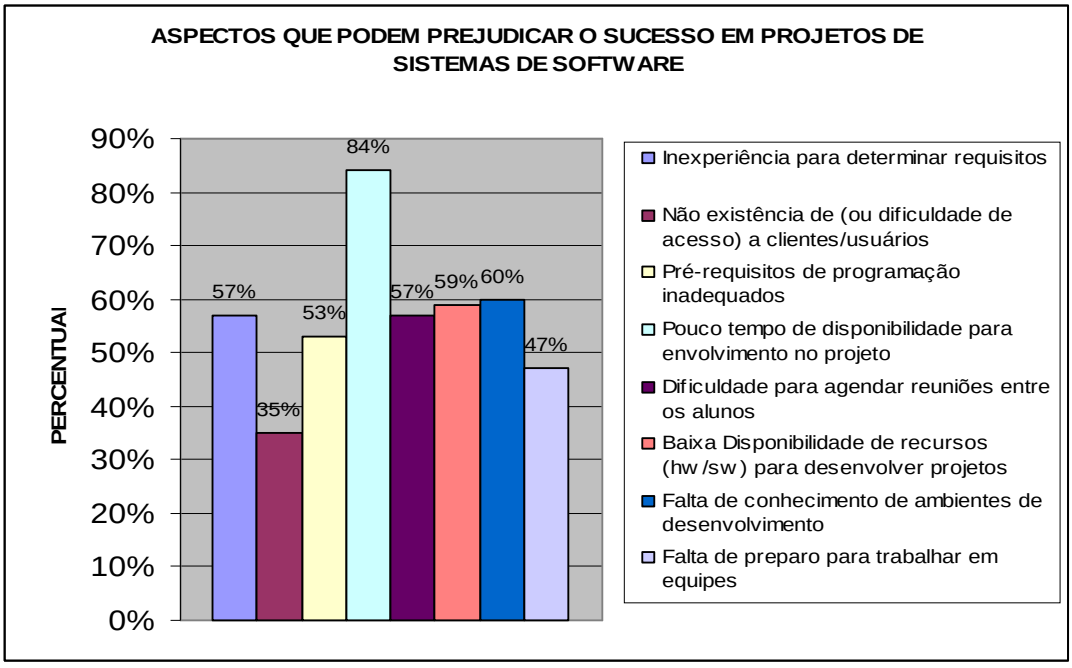


Gráfico 3 – Aspectos que podem prejudicar o sucesso em projetos de sistemas de software

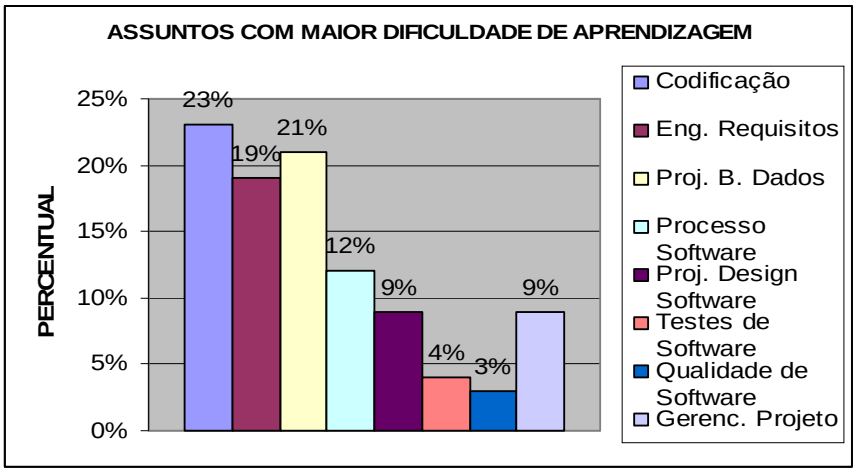


Gráfico 4 – Assuntos com maior dificuldade de aprendizagem

Questão 4: Qual o tópico da Engenharia de Software em que você encontrou maior dificuldade na aprendizagem?

Objetivo: Analisar quais são os tópicos que os discentes em sua própria opinião possuem maior dificuldade de aprendizagem.

Discussão: Quanto aos assuntos que apresentam maior índice de dificuldade na aprendizagem, 23% afirmaram que é a Codificação, 19% selecionaram a Engenharia de Requisitos, 21% escolheram Projeto de Banco de Dados e 12% optaram por Processo de Software conforme ilustra o gráfico 4.

Questão 5: É constituída por 8 itens, com respostas Sim ou Não.

Objetivo: avaliar a situação atual do processo de ensino-aprendizagem do desenvolvimento de software nos cursos de graduação.

Discussão: Tabulados na tabela 1.

Tabela 1: Situação de atividades de projeto

Questão	**Sim**	**Não**
Durante as atividades, os docentes incentivaram a interação entre os membros de um grupo e entre os grupos?	59%	41%
Há atividades que iniciam em um período letivo e finalizadas em outro?	13%	87%
Existe continuidade à projetos/programas iniciados por alunos	9%	91%

de períodos anteriores?		
Há disponibilidade de um Ambiente (tipo Fábrica de Software) que possibilite simulação de ambientes reais de desenvolvimento?	20%	80%
Tópicos associados qualidade de software (CMM, CMMI, etc.) estão sendo abordados?	56%	44%
Situações que ocorrem no cotidiano das corporações como conflitos com usuários, prazos curtos, saída de membros da equipe etc, entre outros, são simulados no decorrer das atividades de projeto?	14%	86%
Há uma ligação entre as disciplinas do curso?	64%	36%
As competências que devem ser adquiridas ao finalizar cada disciplina está sendo mencionadas?	55%	45%

5.2.2 Conhecimentos referentes as TIC's (Tecnologias da Informação e da Comunicação).

Esta segunda parte do questionário procurou identificar os conhecimentos dos entrevistados sobre tecnologias da informação e da comunicação (TICs), bem como verificar se concorda ou não com o uso destas tecnologias no processo de ensino-aprendizagem.

A questão 6 é formada por cinco itens, com respostas Sim ou Não, que tem como objetivo avaliar o percentual de alunos com acesso a Internet, e como encaram o uso de TICs como elemento obrigatório no processo de ensino-aprendizagem.

Discussão: Tabulados na Tabela 2 e na Tabela 3.

Tabela 2: Uso da Internet

Questão	**Sim**	**Não**
Possui acesso a Internet?	98%	2%
Acessa a Internet ao menos três semanalmente?	90%	10%
Tem facilidade com a Internet?	95%	5%

Tabela 3: Opinião sobre o uso de TICs no ensino

Questão	**Sim**	**Não**
Considera que o uso de TICs podem auxiliar na aquisição de	85%	15%

competências?		
Apóia o uso de TICs como recurso obrigatório no processo de ensino-aprendizagem?	95%	5%

Questão 7: Como classifica os seus conhecimentos/habilidades no uso das seguintes TIC's: internet, e-mail, fórum, teleconferência, videoconferência, multimídia e ambientes virtuais de aprendizagem.

Objetivo: Verificar quais são as ferramentas em que os egressos já possuem um alto grau de conhecimento e usabilidade, bem como aquelas com menor índice.

Discussão:

Com relação ao conhecimento dos alunos no uso da Internet, é constatado que 90% possuem bom ou excelente conhecimento, sendo que 10% possui conhecimento razoável e, portanto, não há ninguém que esteja classificado com pouco ou nenhum conhecimento, como ilustrado no gráfico 5.

Outro tópico que não localizou nenhum indivíduo com pouco ou nenhum conhecimento refere-se a utilização de e-mail, e deste modo, 22% afirmaram terem conhecimento razoável, 30% afirmaram possuírem conhecimento bom e 48% afirmaram terem um conhecimento excelente, como ilustrado no gráfico 6.

Com relação aos conhecimentos na utilização da Teleconferência, 51% afirmaram possuírem conhecimentos entre os níveis razoável e excelente, e 49% afirmaram terem pouco conhecimento ou nenhum, como ilustrado no gráfico 7.

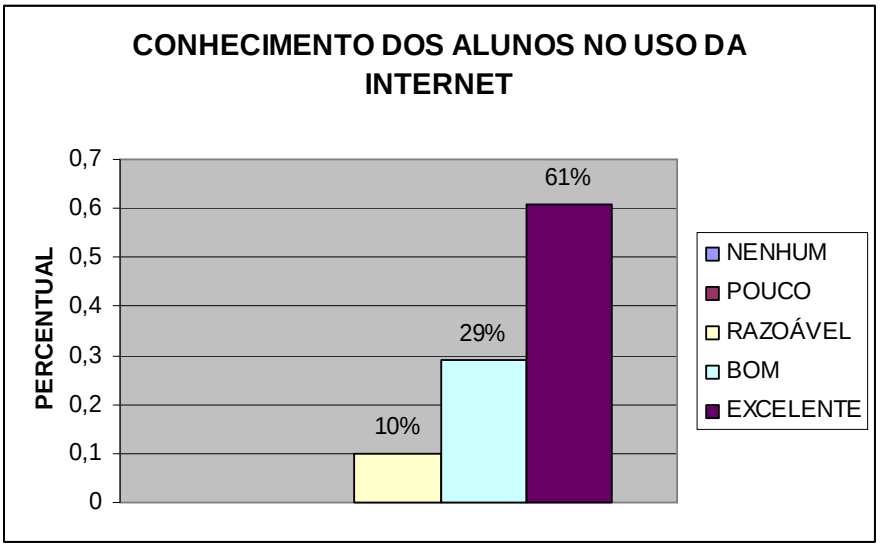
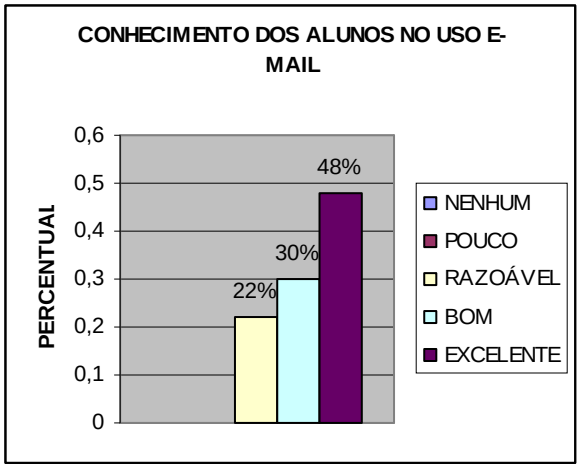


Gráfico 5 – Conhecimento dos alunos no uso da Internet



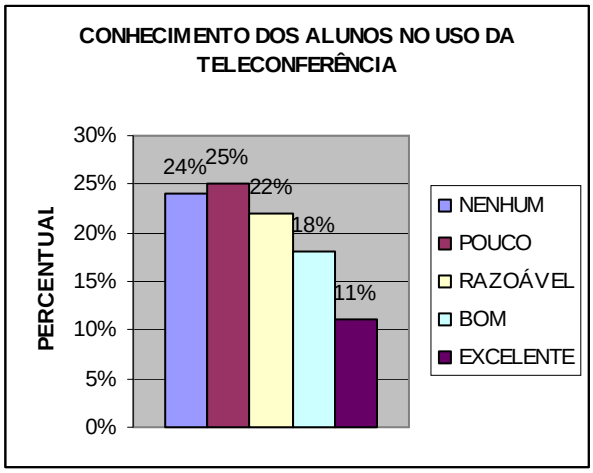


Gráfico 6 – Conhecimento dos alunos no uso do e-mail

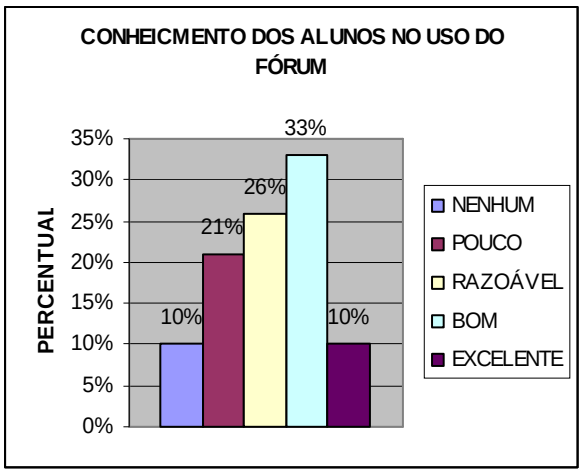
Gráfico 7 – Conhecimento dos alunos no uso da teleconferência

Quanto aos conhecimentos relacionados com o uso dos grupos de discussão, 69% classificaram seus conhecimentos entre razoável a excelente, enquanto 31% afirmaram terem pouco ou nenhum conhecimento, como ilustrado no gráfico 8.

Um dos tópicos que os alunos possuem pouco conhecimento se refere ao uso da videoconferência, pois 25% afirmaram possuírem conhecimentos entre os níveis bom e excelente, e 75% possuem conhecimentos entre os demais níveis, como ilustrado no gráfico 9.

Quanto aos conhecimentos relacionados com a multimídia, 8% afirmaram não possuírem qualquer conhecimento, 13% afirmaram terem pouco conhecimento, e um total de 79% classificaram seus conhecimentos entre os níveis razoáveis e excelentes, como ilustrado no gráfico 10.

O tópico sobre o conhecimento no uso de gerenciadores de ambientes de aprendizagem surpreendeu, já que mais da metade, ou seja, 54% classificaram seus conhecimentos entre excelente e razoável enquanto 46% afirmaram terem pouco ou nenhum conhecimento, como ilustrado no gráfico 11.



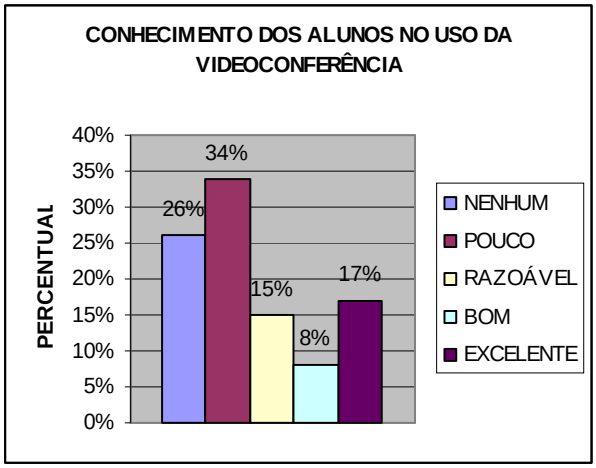


Gráfico 8 – Conhecimento dos alunos no uso do fórum

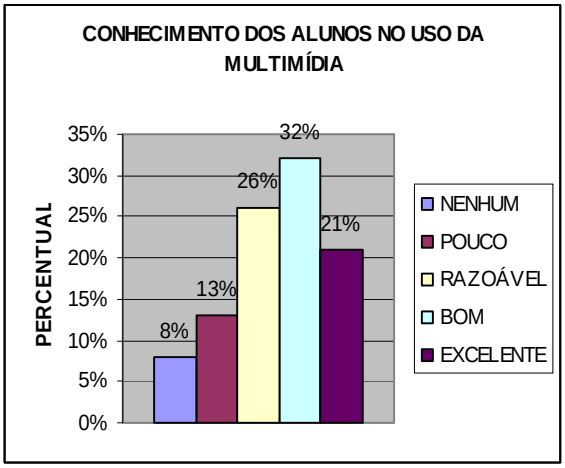


Gráfico 9 – Conhecimento dos alunos no uso da videoconferência

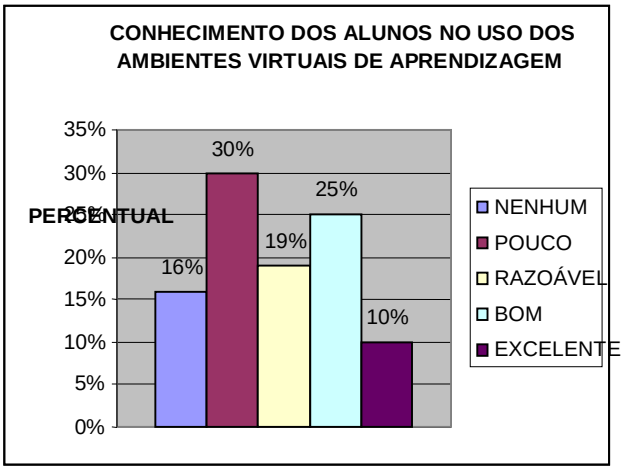


Gráfico 10 – Conhecimento dos alunos no uso da multimídia

Gráfico 11 – Conhecimento dos alunos no uso de ambientes virtuais de aprendizagem

5.3 Comentários sobre os Resultados da Pesquisa

Como pode ser constatado por meio dos dados disponibilizados nas tabelas e nos gráficos, há várias situações que estão ocasionando problemas no processo de ensino-aprendizagem de desenvolvimento de software, prejudicando o ator principal do processo, ou seja, os discentes na aquisição das competências necessárias para que possa exercer futuramente sua profissão.

Ao tratar da distribuição das atividades foi verificado que há uma concentração em teoria em detrimento da prática, implicando na diminuição de oportunidades dos egressos trabalharem cooperativamente e adquirir competências dos tipos: organizacionais (definir metas, objetivos, organizar recursos e definir ações antes de agir; atribuir tarefas a seus subordinados, delegando autoridade e cobrando responsabilidade), comportamentais (compreender a importância da gestão dos colaboradores em um gerenciamento de projetos; agir de forma racional e não por impulso ou emocionalmente), comunicativas (comunicar-se bem de forma oral e escrita; conhecer e aplicar técnicas para o bom relacionamento entre pessoas e manter a eficácia das técnicas de trabalho em equipe), de liderança (possuir a capacidade de atrair, motivar e mobilizar os demais colaboradores em função de alguma atividade a ser desenvolvida; saber harmonizar situações e idéias conflitantes e conseguir estimular a equipe a superar limites).

Em todo e qualquer estudo, sempre existem assuntos com maior facilidade de compreensão, bem como aqueles que possuem um alto grau de dificuldade. Pelos resultados da pesquisa, de acordo com os discentes, as disciplinas com maior grau de dificuldades são as seguintes: codificação, engenharia de requisitos, processo de software e banco de dados. Nestas disciplinas, se os alunos obtiverem um conhecimento limitado ou fragmentado, haverá prejuízo na aquisição de várias competências, dentre as quais vale destacar: as específicas de desenvolvimento (resolver problemas através da implementação de algoritmos, utilizando linguagens de programação; conhecer técnicas de programação; implementar programas de computador em uma

linguagem orientada a objetos; desenvolver sistemas segundo as metodologias de engenharia de software), de banco de dados (conhecer os aspectos básicos de um Sistema de Gerenciamento de Banco de Dados (SGBD), incluindo a utilização de banco de dados em WEB e ambientes distribuídos; modelar sistemas de banco de dados; conhecer as potencialidades dos sistemas de banco de dados para manipulação de dados), de análise e projeto (analisar a determinação de requisitos que um projeto deve atender, documentando-os de forma clara, organizada e de fácil uso; conhecer os fatores de risco do desenvolvimento de software).

Atualmente, praticamente todas as atividades a serem realizadas por profissionais da área de desenvolvimento de software envolvem comunicação e/ou interação entre colegas, e somente 59% dos alunos pesquisados afirmaram que houve incentivo a interação entre os membros de um grupo e entre os grupos. Este número não é o ideal, devido à importância que o trabalho em equipe adquiriu nos últimos anos nos projetos de desenvolvimento de software.

Este pouco incentivo ao trabalho em equipe pode prejudicar a aquisição de várias competências, com destaque para: competências organizacionais (ser capaz de realizar / exercer várias tarefas / funções); comunicativas (comunicar-se bem de forma oral e escrita; conhecer e aplicar técnicas para o bom relacionamento entre pessoas e manter a eficácia das técnicas de trabalho em equipe); comportamentais (agir de forma racional e não por impulso ou emocionalmente); sociais/éticas (agir de acordo com os valores moralmente aceitos pela sociedade; agir com equidade e imparcialidade); de liderança (possuir a capacidade de atrair, motivar e mobilizar os demais colaboradores em função de alguma atividade a ser desenvolvida; saber harmonizar situações e idéias conflitantes); e gerenciais (introduzir novas idéias, atitudes, valores, experiências, processos e procedimentos).

Há uma rotação significativa de profissionais de desenvolvimento entre as organizações devido ao acirrado mercado de trabalho. Portanto, há uma constante mudança nas atividades realizadas por um profissional. Suas responsabilidades podem ser modificadas, acrescidas ou diminuídas em breves

intervalos de tempo, o que sugere que o profissional deve estar muito bem preparado para dar seqüência a novas atividades, que muitas vezes foram iniciadas por outros colegas e não concluídas. Este tipo comum de situação indica que o processo de ensino-aprendizagem deve contemplar atividades que sejam iniciadas em um período letivo por um grupo de alunos e concluídas, ou apenas continuadas, em outro período por outro grupo. Estas situações, conforme pode ser observado pelos resultados da pesquisa, não são exploradas nos cursos pesquisados.

Segundo os alunos, raramente são simuladas situações que podem surgir no dia a dia de projetos e de implantações de sistemas de software. Vivenciar situações típicas do cotidiano de profissionais, mesmo que em ambientes simulados, é de fundamental importância para a formação dos futuros egressos, pois praticamente todos os conhecimentos, todas as habilidades e todas as atitudes exigidas de um profissional podem ser explorados nessas simulações.

5.4 Considerações Finais do Capítulo

O objetivo deste capítulo foi verificar como está a situação do ensino de desenvolvimento de software nos cursos de graduação e avaliar os conhecimentos dos discentes sobre as tecnologias da informação e da comunicação.

Por meio dos resultados obtidos foi possível constatar que há vários problemas interferindo de maneira negativa no processo de ensino-aprendizagem de desenvolvimento de software, e dificultando a aquisição de competências por parte dos discentes, porém em contrapartida os discentes são a favor do uso das TICs para auxiliar o processo de ensino-aprendizagem e possuem índices relevantes de conhecimentos sobre essas tecnologias, com exceção feita à teleconferência e à videoconferência.

Esses resultados foram muito úteis para formular a proposta apresentada no próximo capítulo.

6 PROPOSTAS DE MELHORIA PARA O ENSINO-APRENDIZAGEM DE DESENVOLVIMENTO DE SOFTWARE

Este capítulo apresenta as sugestões de aperfeiçoamento do processo de ensino-aprendizagem dos cursos de nível superior que têm como objetivo principal a formação de desenvolvedores de software. Inicialmente discute o perfil do egresso considerando quais competências serão importantes para sua entrada no mercado de trabalho e quais serão importantes para sua evolução profissional. Em seguida, apresenta a proposta de que sejam usados processos de software para nortear cursos de ensino-aprendizagem de desenvolvimento de software. Na seqüência, discute como estruturar esses cursos sob essa ótica, considerando os aspectos de seqüenciamento de disciplinas e processo de avaliação. Conclui, discutindo a importância de ambientes de ensino-aprendizagem colaborativos para a implementação da proposta.

6.1 O Perfil do Egresso

O objetivo dos cursos da área de computação e informática é a formação de profissionais capacitados para o desenvolvimento de sistemas de computação que atendam as necessidades da sociedade. O perfil do desenvolvedor de software, especificamente, pode ser delineado a partir das competências exigidas pelo mercado de empresas que contrata programadores, analistas de sistemas, analistas de negócios, arquitetos de software, administradores de banco de dados, entre outros profissionais. No capítulo 4 foram delineadas as principais competências exigidas do desenvolvedor de software, divididas em competências específicas essenciais, competências específicas tecnológicas e competências genéricas.

As competências específicas têm como finalidade permitir que os egressos possam assumir de imediato os cargos acima relacionados. Evidentemente essas competências deverão ser aperfeiçoadas com a participação em atividades de desenvolvimento e manutenção de sistemas de

software. Também, deverão ser ajustadas às necessidades específicas de cada organização e aperfeiçoadas continuamente por meio de reciclagem de conhecimento.

As competências genéricas - em particular, as competências intelectuais, comunicativas, comportamentais, organizacionais e sociais e éticas - deverão em um momento inicial ajudar o egresso a rapidamente ajustar-se às exigências do mercado profissional. Em um segundo momento, complementadas pelas competências gerenciais e de liderança e amparadas pelo aperfeiçoamento das competências específicas, serão o sustentáculo da evolução profissional do egresso.

6.2 O Processo de Software como Norteador do Aprendizado

Para a aquisição do amplo leque de competências exigidos do egresso, este trabalho propõe, entre outras sugestões, que processos de software sejam usados para nortear o ensino-aprendizagem de desenvolvimento de software. Antes de apresentar a proposta será discutida a seqüência tradicional de ensino de desenvolvimento de software, serão apresentados os principais elementos que compõem um processo de software, será discutida a importância dos modelos como elementos de abstração e será discutido o processo de software como atividade tipicamente iterativa e incremental.

6.2.1 A Seqüência do aprendizado

Tradicionalmente, o ensino de desenvolvimento de software inicia-se pelas disciplinas que ensinam lógica de programação e linguagem de programação. Evoluem para as disciplinas que ensinam análise de sistemas - tanto engenharia de requisitos, como modelagem de sistemas -, que são ensinadas, geralmente em paralelo com disciplinas que ensinam modelagem e desenvolvimento de bancos de dados. E, em seguida, concluem o ensino com técnicas de projeto de software e a aplicação no desenvolvimento de pequenos

projetos. Raramente, são aprofundadas técnicas de testes, implantação de sistemas e manutenção. Geralmente, no final do ciclo de aprendizagem, são ministrados aspectos teóricos de gerenciamento de projetos, gerenciamento de configuração, estudos de viabilidade, avaliação de riscos de projetos, entre outros temas relacionados ao desenvolvimento de software.

Pode-se observar que esta metodologia segue dos aspectos mais concretos para os mais abstratos do desenvolvimento. Este procedimento parece natural e correto, se for considerado que os alunos estão, ainda, em formação de seus processos de abstração em desenvolvimento de software. Porém, a falha que pode ser observada é que as disciplinas são apresentadas como assuntos individuais e estanques, sem a preocupação de enquadrar os conceitos ministrados no contexto mais abrangente dos processos de software.

Outra crítica que pode ser feita ao procedimento é que ele é essencialmente cascata, ou seja, segue as características do ciclo clássico de desenvolvimento, deixando de aproveitar os aspectos interativos, iterativos e incrementais dos ciclos de vida mais modernos, como o processo unificado e a *extreme programming*.

6.2.2 Os Principais Elementos de Processos de Software

Para introduzir processos de software como norteadores do ensino-aprendizagem de desenvolvimento de software, é interessante relembrar os principais elementos que os compõem.

Um processo de software é constituído por fases seqüenciais. Nessas fases são realizadas atividades, divididas em tarefas elaboradas por profissionais específicos. Ao final de cada fase, são liberados artefatos específicos, que, devidamente avaliados e validados, serão usadas nas fases seguintes. Um processo de software determina também marcos de controle para possibilitam seu gerenciamento e pontos de controle para a avaliação de qualidade. Atividades de apoio são realizadas ao longo de todas as fases, como, por exemplo, o gerenciamento de configuração.

6.2.3 Modelos como Elementos de Abstração

A quantidade de informações coletadas e compiladas durante o desenvolvimento de um sistema de software é muito grande. Apresentá-las textualmente torna a comunicação entre os participantes praticamente inviável. Por esse motivo há décadas são adotados modelos, normalmente, gráficos, para representar as abstrações realizadas durante o desenvolvimento. Para citar como exemplo, entre as linguagens de modelagem mais difundidas atualmente encontra-se a UML (*Unified Modeling Language*), embora muitas outras técnicas de modelagem sejam usadas para abstrair modelos.

6.2.4 O Processo de Software como Atividade Iterativa e Incremental

Sistemas de software estão continuamente evoluindo, procurando acompanhar as modificações que ocorrem no contexto sistêmico em que estão inseridos e as constantes evoluções das tecnologias que os implementam. Apresentam, também, uma gama muito grande de requisitos que nem sempre podem ser inferidos ou implementados em uma única etapa de desenvolvimento. Por estas razões, os processos de software são tipicamente iterativos, ou seja, realizados por meio da repetição de seqüências de atividades que implementam incrementos a artefatos já desenvolvidos. Essa característica, que não está presente no ciclo clássico, é comum a praticamente todos os processos de software atualmente utilizados pela indústria de software, porém não é explorada no ensino devido aos projetos de desenvolvimento estarem circunscritos a períodos letivos e, normalmente, não serem continuados em períodos seguintes, pelo mesmo grupo ou por outros grupos de alunos. Este fato impede geralmente a realização de mais de um ciclo de iteração, ficando o conhecimento dos alunos restrito ás características do ciclo clássico.

6.2.5 O Procedimento Proposto

Como alternativa a abordagem descrita acima, propõe-se que o conceito de processo de software seja introduzido aos alunos desde os primeiros dias de aula. Os primeiros algoritmos desenvolvidos devem ser abordados sob o conceito de processo de software, trabalho desenvolvido em equipe, com fases de elaboração bem determinadas, artefatos intermediários bem definidos, atividades claramente delineadas, atribuições de funções para membros de grupos, modelagem quando for possível, trabalhos de análise bem realizados. Ou seja, explorando todas as características de processos de software bem definidos.

Para o início do ensino de desenvolvimento de software, nas disciplinas de lógica de programação e linguagem de programação sugere-se a adoção da metodologia *Extreme Programming*, pelas suas características de se adequar bem a pequenos projetos desenvolvidos por equipes não muito grandes.

A ênfase na programação é a principal característica da *Extreme Programming,* o que a torna ideal para as etapas iniciais de aprendizado. Outros benefícios, sugeridos por Teles (2006) e adaptados ao contexto de ensino, são os decorrentes da programação em pares, que possibilita: detecção de erros com maior facilidade quando dois alunos estão trabalhando juntos; simplicidade, uma vez que a programação em pares auxilia os alunos a elaborarem soluções mais simples e mais rápidas de implementar; pressão do par, pois a pressão do trabalho em par faz com que os alunos fiquem mais focados nas atividades e por mais tempo; disseminação do conhecimento, pode-se dizer que uma das principais características da programação em pares é a sua capacidade de disseminação do conhecimento, pois há constante troca de pares, fato que promove um maior compartilhamento de informações; confiança, pois trabalhando em pares os alunos têm mais confiança nos códigos desenvolvidos; velocidade, como conseqüência natural do grupo de características citados, há aceleração na solução dos trabalhos sugeridos.

Nos primeiros projetos de sistemas de software, nas disciplinas de análise e projeto, sugere-se que se continue o uso da *extreme programming*, uma vez

que ela ainda é a mais adequada para esses primeiros projetos.

Nas etapas seguintes do processo de ensino, sugere-se a adoção de processos mais sofisticados, como o processo unificado, aplicados a projetos que envolvam mais de uma disciplina. É interessante que esses projetos tenham continuidade em outros períodos letivos, ou que já sejam continuação de projetos desenvolvidos por outros grupos de alunos em períodos anteriores. Projetos mais robustos exigem da equipe que os desenvolve maiores cuidados com a documentação e com o gerenciamento, forçando as equipes a realizarem todas as atividades previstas no ciclo de desenvolvimento dos processos adotados. Possibilitando, desta forma, que os discentes aperfeiçoem as competências mais valorizadas pelo mercado profissional.

6.3 Estrutura de Curso

Dois aspectos importantes devem ser considerados ao propor uma estrutura de curso: primeiro como serão seqüenciadas as disciplinas, e, segundo, como será o processo de avaliação.

O seqüenciamento das disciplinas deve manter o tradicional, citado no item 6.2.1, uma vez que essa seqüência tem se mostrado ao longo dos anos como mais efetiva para o ensino de desenvolvimento de software. Por outro lado, organizações curriculares por módulos têm sido defendidas nos últimos anos para o ensino técnico e tecnológico (Novelli, 2006). Sugere-se, então que as disciplinas sejam agrupadas em módulos consistentes que permitam a realização de projetos integradores, que explorem conjuntamente todas as competências características de cada disciplina individual. O agrupamento das disciplinas pode apresentar diferentes conformações, devido a diferentes aspectos, como por exemplo: o número de anos em que é ministrado o curso, se ele é seriado ou por crédito, se as disciplinas são anuais ou semestrais, se possui disciplinas optativas ou não, se apresenta um elenco de disciplinas complementares que procura direcionar o curso para um tipo de contexto específico, entre outros. Porém, é importante que incluam, além de disciplinas

que respondem pela aquisição das competências específicas essenciais, as disciplinas que são responsáveis pelas disciplinas específicas tecnológicas e as disciplinas que contribuem para a aquisição das competências genéricas.

Quanto ao processo de avaliação, a sugestão para este tipo de estrutura de curso é a avaliação por competências. Este processo avaliativo procura determinar se o aluno adquiriu as competências previstas nas disciplinas que esta cursando. Estes procedimentos de avaliação, em geral, envolvem situações práticas em que as habilidades, conhecimentos e atitudes dos alunos são observados e valorizados. Para a proposta em questão, o procedimento mais adequado é avaliar a participação dos alunos nos projetos em que estão participando. Procedimento que exige uma participação do avaliador muito próxima aos alunos ou um registro muito efetivo e confiável das atividades realizadas e dos resultados obtidos por cada aluno individualmente.

6.4 A Proposta e o Ensino-Aprendizagem Colaborativo

O conjunto de competências necessárias a um bom desenvolvedor de software é amplo e exige um longo período para se completar. A utilização da aprendizagem colaborativa nos cursos de desenvolvimento de software é de grande valia ao processo de ensino-aprendizagem, pois auxilia e facilita os discentes na aquisição e assimilação das várias competências. O desenvolvimento de software ensinado de modo colaborativo aproxima da realidade o cotidiano dos desenvolvedores e torna as aulas mais interessantes, investigativas, interativas e participativas. Possibilita, também, a discussão e a troca de idéias, incentivando a pesquisa e fazendo com que os discentes exerçam maior interação junto aos colegas, bem como junto aos docentes, e deste modo, gerando conhecimento de forma coletiva.

Entre as principais competências que podem ser obtidas e aperfeiçoadas por meio da aprendizagem colaborativa, podem-se citar:

- Reconhecer os elementos do domínio do problema relevantes para a aplicação;

- Conhecer os objetivos e fronteiras de um sistema de software;
- Ser capaz de realizar / exercer várias tarefas / funções;
- Comunicar-se bem de forma oral e escrita;
- Conhecer e aplicar técnicas para o bom relacionamento entre pessoas e manter a eficácia das técnicas de trabalho em equipe;
- Compreender a importância da gestão dos colaboradores em um gerenciamento de projetos;
- Atribuir tarefas certas a seus subordinados, delegando autoridade e cobrando responsabilidade;
- Possuir a capacidade de atrair, motivar e mobilizar os demais colaboradores em função de alguma atividade a ser desenvolvida;
- Saber harmonizar situações e idéias conflitantes;
- Conseguir estimular a equipe a superar limites;
- Introduzir novas idéias, atitudes, valores, experiências, processos e procedimentos;
- Saber ponderar e conduzir negociações de forma que as partes saiam satisfeitas;
- Familiarizar-se com as tecnologias e ferramentas de análise e projeto de software e o discernimento de como, quando e quanto utilizar tais ferramentas;
- Determinar e analisar requisitos que um projeto deve atender, documentando-os de forma clara, organizada e de fácil uso;
- Planejar, supervisionar, elaborar e coordenar projetos de software;
- Aperfeiçoar técnicas de levantamento de requisitos e coleta de dados;
- Desenvolver sistemas segundo as metodologias de engenharia de software;
- Implementar programas de computador em uma linguagem de programação;
- Validar o software para funcionar conforme projetado, por meio da

combinação de codificação, teste de unidades e teste integrado;

- Modelar sistemas de banco de dados;
- Projetar interfaces de usuário utilizando conceitos de interface humano-computador.

Pode-se observar que o elenco de competências que podem ser aperfeiçoadas por meio da aprendizagem colaborativa inclui não só as competências específicas essenciais, mas também as competências genéricas, em particular aquelas que desenvolvem a capacidade de trabalhar em equipe, característica extremamente valorizada atualmente em profissionais de desenvolvimento de software.

6.5 A Inclusão de Ambientes de Ensino-Aprendizagem Colaborativo

A inclusão de ambientes de ensino-aprendizagem colaborativo é determinante para o sucesso da proposta, uma vez que gerenciar tal contexto sem recursos computadorizados é praticamente impossível.

Ambientes CSCL possuem as características necessárias para implementar a proposta aqui apresentada. Esses ambientes são implementados com Tecnologias da Informação e da Comunicação que são do domínio dos alunos, como pode ser observado pela pesquisa apresentada no quinto capítulo deste trabalho. Portanto, a adaptação dos alunos a estes ambientes é relativamente rápida e possibilita que realizem projetos em equipe, não apenas no ambiente físico da instituição de ensino, mas também, fora dele, explorando os recursos de comunicação que estes ambientes disponibilizam.

A coordenação dos trabalhos pelos professores também pode ser dinamizada, uma vez que os ambientes em geral disponibilizam recursos de gerenciamento e monitoria que podem ser acessados de qualquer lugar e a qualquer momento. Porém, aqui se esbarra em problemas importantes, como a rejeição dos professores a ambientes que promovem o ensino a distância, a rejeição de alguns professores de se envolverem com novas tecnologias e a

rejeição em mudarem a sua forma de ministrar aulas. Basicamente, todas essas formas de rejeição estão associadas à preocupação constante dos professores com a descaracterização da profissão e a substituição de suas atividades por sistemas computadorizados, mesmo com professores que atuam na área de computadorização de sistemas.

6.6 Considerações Finais do Capítulo

A proposta apresentada procurou levar em consideração a opinião dos alunos que consideram o laboratório o recurso mais significativo para o aprendizado e freqüentam cursos preponderantemente teóricos, como pode ser observado nos gráficos 1 e 2, no capítulo 5 . A proposta sugere o uso de ambientes colaborativos que podem minimizar os efeitos da falta de tempo que os alunos tem para envolver-se em projetos. E propõe a continuidade de projetos - possibilitada pelo seu desenvolvimento em ambientes que podem manter seus resultados - como forma de simular situações típicas do dia a dia dos profissionais.

Portanto, a proposta de nortear o ensino de desenvolvimento de sistemas de software por processos de software em ambientes de ensino-aprendizagem colaborativo tem como finalidade oferecer alternativas às IES que mantém cursos na área de computação e informática. Considerando que os resultados, por elas obtidos na formação de egressos capacitados para enfrentar os desafios de um mercado de trabalho dinâmico e desafiador, não tem sido satisfatórios. É consistente com a opinião de educadores como Wolynec (2007) para quem é necessário adotar novas metodologias de ensino centradas em projetos e atividades, na qual os estudantes utilizem ambientes digitais para criar, apresentar e compartilhar sua aprendizagem, interagindo com seus pares.

CONSIDERAÇÕES FINAIS

Nestas considerações finais serão resumidas as principais conclusões e contribuições deste trabalho, e serão apresentadas algumas oportunidades de continuação e estudos complementares futuros.

Neste trabalho foi explorada a hipótese de que para cursos de computação e informática com ênfase no desenvolvimento de software é possível desenvolver ambientes de ensino-aprendizagem alternativos aos atuais, que comprovadamente não atendem as expectativas do mercado de trabalho. A proposta de um ambiente norteado por processos de software e sustentado por ambientes colaborativos baseados em modernas Tecnologias da Informação e da Comunicação foi apresentada como uma possível alternativa para a formação de profissionais que venham a possuir as competências necessárias para rapidamente assumir funções nas organizações desenvolvedoras de sistemas de software.

O ambiente de ensino-aprendizagem proposto alinha-se a opiniões como a de Wolynec (2006) para quem “o único caminho que permite formar o tipo de profissional que o mercado atual necessita, com competência para atacar problemas complexos e para negociar consenso em trabalhos em equipe, consiste na diminuição de aulas expositivas, substituindo-se o ensino transmissional pela aprendizagem ativa”.

O trabalho também propôs vários objetivos específicos que foram alcançados e apresentados ao longo dos capítulos. Esses objetivos foram alcançados por meio de revisões bibliográficas, de pesquisas, de observações e de reflexões do autor. Resumindo: a) os principais temas associados ao processo de desenvolvimento de software foram apresentados e serviram de base para a proposição principal; b) as principais disciplinas que estruturam a formação de desenvolvedores de software foram apresentadas e subdivididas em essenciais, tecnológicas e complementares, para facilitar as reflexões sobre o ambiente proposto; c) a educação colaborativa foi apresentada, bem como discutidas as principais características de ambientes de ensino-aprendizagem

colaborativos, denominados CSCL; d) as competências mais significativas para desenvolvedores de software foram discutidas e divididas em genéricas e específicas para facilitar sua análise frente aos problemas dos atuais ambientes e à proposta apresentada; d) a pesquisa realizada junto aos discentes foi tabulada e analisada e seus resultados foram utilizados para dar sustentação a proposta principal do trabalho.

Como sugestões para dar continuidade ao trabalho podem-se relacionar:

- A pesquisa de campo foi aplicada somente em instituições privadas localizadas na região da grande São Paulo, pode-se ampliá-la para outras regiões e cobrir instituições públicas que apresentam um perfil discente bem diferente;
- Pode-se realizar uma pesquisa com os docentes e com administradores de IES para identificar seus principais problemas e limitações no uso de TICs e de ambientes colaborativos;
- Pode-se ampliar o universo de ambientes colaborativos, bem como o nível de profundidade nos estudos realizados com esses ambientes;
- Pode-se procurar aplicar os conceitos do processo proposto, mesmo que parcialmente, em um módulo de um curso, para validá-lo.

Como pesquisas futuras, sugerem-se:

- A criação de um ambiente colaborativo específico para dar sustentação ao processo de ensino-aprendizagem proposto, ou alternativamente personalizar ambientes abertos já existentes para servirem de sustentação ao processo proposto;
- Estudar o novo papel do professor (tutor) frente às novas abordagens de ensino-aprendizagem;
- Avaliar como é o desenvolvimento dos egressos que estudam a distância através de cursos totalmente baseados em um sistema de gerenciamento de cursos a distância frente aos discentes que estudam de maneira convencional;
- Estender o estudo realizado a outros tipos de cursos.

Concluindo, este trabalho alcançou os objetivos propostos, contudo pelo número de propostas de seqüência do trabalho e pelo número de opções de novas pesquisas que ele abre, há muito trabalho ainda por se fazer nesta área.

REFERÊNCIAS

ARETIO, Lorenzo Garcia. **La educación a distancia – De la teoría a la práctica.** Barcelona: Editorial Ariel, 2002.

ASSIS, Maria Luiza Fava Lopes Camargo de. **A Influência do Curso de Extensão PROEPRE – Fundamentos Teóricos e Prática Pedagógica para a Educação Infantil – na Formação Continuada de Professores.** Campinas: 2007. (Dissertação de Mestrado em Educação defendida em 2007 na Faculdade de Educação da Universidade de Campinas.

BAPTISTA, Renan Martins; STAA, Arndt Von; FIORINI, Soeli T. **Engenharia de Software com CMM.** Rio de Janeiro: Brasport, 1998.

BEGOSSO, Luiz Ricardo. **Ambiente para o Desenvolvimento de Maturidade em Engenharia de Software em um Curso de Ciência da Computação.** São Paulo: 2002. (Dissertação de Doutorado defendida em 2002 na Escola Politécnica da Universidade de São Paulo).

BITTENCOURT, Dênia Falcão de. **A construção de um modelo de curso "lato sensu" via internet – a experiência com o curso de especialização para gestores de instituições de ensino técnico UFSC/Senai.** 1999. Dissertação (Mestrado em Engenharia de Produção) – Programa de Pós-Graduação em Engenharia de Produção, UFSC, Florianópolis.

BOOCH, Grady; RUMBAUGH James; JACOBSON, Ivar. **UML Guia do Usuário.** Rio de Janeiro: Campus, 2000.

BOYATZIS, Richard E. Self-Directed Learning. Executive Excellence. Feb 2004. Vol. 21. No. 2. pp. 11 e 12.

BRAUDE, Eric. J. **Software Engenieering: an object-oriented perspective.** New York: John Wiley & Sons Inc, 2001.

CAMPOS, Fernanda C. A; SANTORO, Flávia Maria; Borges, Marcos R. S; SANTOS, Neide. **Cooperação e Aprendizagem On-line.** Rio de Janeiro: DP&A, 2003.

CASTORINA, Jose Antonio; FERREIRO, Emilia; LERNER, Delia; OLIVEIRA, Marta de Kohl**. Piaget e Vygotsky.** 4 º Edição.São Paulo: Atica, 1998.

FLEURY, Maria Tereza Leme; FLEURY, Afonso. **Estratégias Empresariais e Formação de Competências.** 3º Edição. São Paulo: Atlas, 2004.

GRAMIGNA, Maria Rita. **Modelo de competências e gestão de talentos.** São Paulo: Makron Books, 2002.

HEERDT, Ana Paula Szpoganicz. **Competências Essenciais dos Coordenadores de Curso em uma Instituição de Ensino Superior.** Florianópolis: 2002.
(Dissertação de Mestrado defendida em 2002 no PPGEP/UFSC).

KENSKI, Vânia Moreira. **Tecnologias e ensino presencial e a distância.** 4º edição. Campinas: Papirus, 2003.

KOMOSINSKI, Leandro José. **Um Novo Significado para a Educação Tecnológica fundamentado na Informática como Artefato Mediador da Aprendizagem.** (Dissertação de Doutorado defendida em 2000 no PPGEP/UFSC).

JAVAFREE. Obtendo Qualidade de Software com o RUP. Disponível em http://www.javafree.org/content/view.jf?idContent=7. Acesso em 24/03/2006.

KRUCHTEN, Philippe. **The rational unified process: na introduction.** 3 ed. Boston, MA: Addison Wesley, 2003.

LEVY, Pierre. Cibercultura. São Paulo: Editora 34, 1999.

NOVELLI, Gisele. Currículo por Módulos e Educação Profissional Técnica: Crítica da Formação para o Mercado de Trabalho. São Paulo 2006. (Dissertação de Doutorado defendida em 2006 na Pontifícia Universidade Católica de São Paulo).

UNIVERSIDADE DE MOGI DAS CRUZES. Matriz Curricular do Curso de Bacharelado em Sistemas de Informação. Disponível em http://web1.umc.br/ensino/graduacao/campus villa/bach_sist_info/bach_sist_info.htm. Acesso em 28/01/2006.

NUNES, Luiz Eduardo Perfeito. **Revisão pelos Pares na Aprendizagem de Análise e Projeto de Sistemas: UM Estudo de Caso.** Florianópolis: 2005. (Dissertação de Mestrado defendida em 2005 no PPGEP/UFSC).

PETERS, James F; PEDRYCZ, Witold. **Engenharia de Software.** Rio de Janeiro: Campus, 2001.

PFLLEGER, Shari Lawrence. **Engenharia de Software: Teoria e Prática –** 2º Edição. São Paulo: Prentice Hall, 2004.

PRESSMAN, Roger A**. Engenharia de Software.** 5º Edição. Rio de Janeiro: McGraw-Hill, 2002.

PRESSMAN, Roger A. **Engenharia de Software.** 6º Edição. São Paulo: McGraw-Hill, 2006.

PRIKLADNICKI, Rafael; Audy, Jorge. **Desenvolvimento distribuído de software.** Rio de Janeiro: Elsevier, 2008.

SILVA, Marilise Krueger. **O Comprometimento com a Qualidade dos Sistemas de Informação: Um Enfoque nas Competências das Pessoas.** Florianópolis: 2001.
(Dissertação de Mestrado defendida em 2001 no PPGEP/UFSC).

SOCIEDADE BRASILEIRA DE COMPUTAÇÃO. Currículo de Referência da Sociedade Brasileira de Computação para os Cursos em Graduação em Computação e Informática.
Disponível em http://www.sbc.org.br/index.php?language=1&subject=28. Acesso em 15/05/2006.

SOFTEX. Empresários do setor de software reúnem-se em Campinas.
Disponível em http://cps.softex.br/noticia_interna.php?id=577. Acesso em 11/04/2007.

SOMMERVILLE, Ian. **Engenharia de Software -** 6º Edição. São Paulo: Addison Wesley, 2003.

SOUZA, Cleidson de. Desenvolvimento Colaborativo de Software.
Disponível em http://www2.ufpa.br/cdesouza/teaching/cscw-2006-2/12-DesenvolvimentoColaborativoDeSoftware.ppt#256,1,Desenvolvimento Colaborativo de Software. Acesso em 22/06/2007.

SVEIBY, Karl Erik. **A Nova Riqueza das Organizações: gerenciando e avaliando patrimônio de conhecimento.** Rio de Janeiro: Campus, 1998.

TELES, Vinícius Manhães. Programação em Par. Disponível em http://www.improveit.com.br/xp/praticas/programacao_par. Acesso em 20/11/2006.

UNIVERSIDADE DE BRASÍLIA. Metodologias Ágeis. Disponível em http://www.redes.unb.br/material/ESOO/Metodologias%20%C1geis.pdf. Acesso em 15/06/2007.

UNIVERSIDADE FEDERAL DE UBERLÂNDIA. Diretrizes Curriculares de Cursos da Área de Computação e Informática. Disponível em http://www.pp.ufu.br/Computacao.htm. Acesso em 10/03/2007.

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ. Projeto Pedagógico do Curso Superior em Tecnologia em Sistemas de Informação. Disponível em http://www.pg.cefetpr.br/coinf/disciplinas.php. Acesso em 20/05/2008.

WOLYNEC, Elisa. A Educação na Era da Interatividade. Disponível em http://www.techne.com.br/artigos/A%20Educ%20_Era_Interatividade.pdf. Acesso em 02/05/2007.

WOLYNEC, Elisa. Educando os profissionais do século 21 com a metodologia do
século18. Disponível em http://www.techne.com.br/artigos/Formando%20o%20profissional%20do%20século%2021%20com%20a%20metodologia%20do%20século%2018.pdf. Acesso em 05/11/2007.

WOLYNEC, Elisa. Entraves à evolução da Educação Superior. Disponível em http://www.techne.com.br/artigos/Entraves_Evolucao.pdf. Acesso em 14/04/2007.

WOLYNEC, Elisa. Formação profissional em sintonia com o mercado. Disponível em http://www.techne.com.br/artigos/Formação%20profissional%20em%20sintonia%20com%20o%20mercado.pdf. Acesso em 10/10/2007.

WOLYNEC, Elisa. Mercado de Ensino Superior em estado de Alerta. Disponível em http://www.techne.com.br/artigos/Mercado_ensino_alerta.pdf.
Acesso em 10/10/2006.

WOLYNEC, Elisa. Mudanças estratégicas para a evolução das IES. Disponível em http://www.techne.com.br/artigos/Mudancas_estrategicas_IES.pdf.
Acesso em 26/10/2006.

APÊNDICE A – Modelo do questionário de avaliação do curso direcionado ao Processo de Software aplicado aos discentes.

Prezado(a) Aluno(a):

Este questionário foi concebido com a objetivo de conhecer a sua opinião sobre o processo de ensino/aprendizagem de um Curso direcionado a aquisição de competências referente a atividades de Processo de Software. Não é necessário identificar-se. Suas respostas serão utilizadas para consolidar o trabalho e também para identificar mecanismos que possam contribuir e aperfeiçoar o processo de ensino/aprendizagem.

Grato pela colaboração.

Número Sequencial Único: XXXXXX

Data:
Curso: Ano que cursa:

Questões pertinentes ao processo de ensino-aprendizagem de processo de software.

1) Percentualmente, as atividades realizadas no curso se concentraram em:
Teoria ( ) Exercícios ( ) Projetos ( )

2) Quais os recursos que mais colaboram no processo de ensino/aprendizagem na área de processo de software?
( ) Lousa ( ) Apostilas ( ) Retro projetor
( ) Internet ( ) Data-Show ( ) Outros _____

3) Considerando que a participação efetiva em Projetos de Sistemas de Software é uma das formas de experimentação que mais agregam valor para a formação dos alunos, avalie como cada um dos itens abaixo pode ser prejudicial para o sucesso da atividade?
Legenda: (a) Nada (b) Pouco (c) médio (d) Muito

( ) Inexperiência para determinar requisitos
( ) Não existência de (ou dificuldade de acesso a) cliente/usuários
( ) Pré-requisitos de programação inadequados
( ) Pouco tempo de disponibilidade para envolvimento no projeto
( ) Dificuldade para agendar reuniões entre os alunos
( ) Baixa disponibilidade de recursos (hw/sw) para desenvolver projetos
( ) Falta de conhecimento de ambientes de desenvolvimento
( ) Falta de preparo para trabalhar em equipes
( ) Outros _____

4) Qual o tópico da Engenharia de Software que encontrou maior dificuldade de aprendizagem? (marque com X)

( ) Engenharia de Requisitos
( ) Processo de Software
( ) Prototipação
( ) Codificação
( ) Outros _____

( ) Projeto (“Design”) do Software
( ) Gerenciamento de Projeto
( ) Testes de Software
( ) Qualidade do Software

5) Responda (S – sim ou N – não) para as seguintes questões:

( ) Durante as atividades, os docentes incentivaram a interação entre os membros de um grupo e entre os grupos?
( ) Há atividades que iniciam em um período letivo e finalizadas em outro?
( ) Existe continuidade à projetos/programas iniciados por alunos de períodos anteriores?
( ) Há disponibilidade de um Ambiente (tipo Fábrica de Software) que possibilite simulação de ambientes reais de desenvolvimento?
( ) Tópicos associados qualidade de software (CMM, CMMI, etc.) estão sendo abordados?
( ) Situações que ocorrem no cotidiano das corporações como conflitos com usuários, prazos curtos, saída de membros da equipe etc, entre outros, são simulados no decorrer das atividades de projeto?
( ) Há uma ligação entre as disciplinas do curso?
( ) As competências que devem ser adquiridas ao finalizar cada disciplina está sendo mencionadas?

Questões pertinentes ao conhecimento e habilidade na utilização das tecnologias da informação e da comunicação.

6) Responda (S – sim ou N – não) para as seguintes questões:

( ) Possui acesso a Internet?
( ) Acessa a Internet ao menos três semanalmente?
( ) Tem facilidade com a Internet?
( ) Considera que o uso de TICs podem auxiliar na aquisição de competências?
( ) Apóia o uso de TICs como recurso obrigatório no processo de ensino-aprendizagem?

7) Como classifica os seus conhecimentos/habilidades no uso das seguintes TIC’s: internet, e-mail, fórum, teleconferência, videoconferência, multimídia e ambientes virtuais de aprendizagem.

Legenda: (1) Nenhum (2) Pouco (3) Razoável (4) Bom (5) Excelente

( ) Internet ( ) E-mail ( ) Fórum

( ) Teleconferência ( ) Videoconferência ( ) Multimídia
( ) Ambientes Virtuais de Aprendizagem