Estruturação do Processo de Desenvolvimento de *Software*: Uma Proposta para o Trabalho em Equipe

Maria Cristina Delgado Preti[1]
Cristiano Maciel[2]

Resumo: Este artigo discute o processo de desenvolvimento de *software* e o ganho que se obtém com a padronização nos seus processos. O estudo em questão sugere a estruturação de um processo de desenvolvimento de *software* para a Diretoria de Tecnologia de Informação do Instituto Federal de Educação, Ciência e Tecnologia de Mato Grosso (IFMT). Para tal, discute alguns modelos de processos de desenvolvimento de *software*, os elementos da linguagem UML e o paradigma da orientação a objetos, apresentando o resumo de alguns trabalhos relacionados ao tema proposto.

Palavras-chave: Processo de desenvolvimento de *software*, modelo de processos de *software*.

Abstract: This paper discusses the process of the software development and the gain is that obtained by the standardization in its processes. This study suggests the structuration of a process for software development for the Information Technology Directory at the Federal Institute of Education, Science and Technology of Mato Grosso (IFMT). Based on this, this study presents some models of software development processes, the elements of the UML language, the object-oriented paradigm. The summary of some studies related to the mentioned theme was presented.

Keywords: Software development processes, models of software processes.

1 Especialista em Gestão de Projetos em Engenharia de *Software*, pelo Centro Universitário Cândido Rondon; analista de Tecnologia da Informação da Gerência de Tecnologia da Informação do IFMT – Campus Cuiabá. E-mail: maria.preti@cba.ifmt.edu.br.

2 Doutor em Computação, pela Universidade Federal Fluminense; professor do Instituto de Computação da Universidade Federal de Mato Grosso (UFMT). E-mail: cmaciel@ufmt.br.

Introdução

A falta de adoção de métodos, de ferramentas e de procedimentos no desenvolvimento de *softwares* resulta em números expressivos de projetos não concluídos ou em projetos concluídos que não atendem às necessidades do cliente (PRESSMAN, 2006). Para solucionar esse problema, existe uma área do conhecimento da computação denominada Engenharia de *Software*, que surgiu em meados dos anos 1970, propondo a criação e a utilização de sólidos princípios de engenharia, a fim de obter um *software* de maneira econômica e confiável. Uma das proposições dessa ciência é o processo de desenvolvimento de *softwares*, foco deste artigo.

De acordo com Paula Filho (2003), o processo de desenvolvimento de *software* é um conjunto de passos parcialmente ordenados, constituídos por atividades, métodos, práticas e transformações, usadas para atingir uma meta. A consolidação dessa forma de trabalho com vistas à padronização permitirá que o produto final seja entregue dentro do prazo estabelecido com eficiência e melhor qualidade, visto que o fluxo dos processos e das atividades ligados ao desenvolvimento de um sistema será mais ágil e a produtividade dos agentes envolvidos aumentada, devido ao uso racional da mão de obra.

Com a recente estruturação ocorrida no IFMT, observou-se a carência de um modelo de desenvolvimento de processo de *software* que permitisse a padronização de processos e artefatos a serem usados pela equipe de desenvolvimento. Portanto, este estudo propõe à Diretoria de Gestão de Tecnologia de Informação (DGTI) a implantação de um modelo de processo de desenvolvimento de *software* tendo por finalidade manter integradas as equipes do IFMT, guiadas por um modelo único, trabalhando de forma mais produtiva, com artefatos úteis e adequados, utilizando ainda a mesma linguagem na documentação, na codificação e na comunicação sistêmica. Tal pesquisa foi realizada com base em referências bibliográficas, investigando, entre outros, experiências nesta área.

Para atingir o objetivo almejado, a seguir, serão abordados os modelos de processos de desenvolvimento de *software*, os artefatos da linguagem

UML, o paradigma da orientação a objetos e um resumo de alguns trabalhos relacionados ao tema investigado. Além disso, também será feita a proposta de um processo de desenvolvimento de *software* para o IFMT.

Processos de Desenvolvimento de *Software*

O processo de desenvolvimento de *software* é um conjunto de atividades e de resultados associados necessários para construir aplicações de alta qualidade. Ele forma a base para o controle gerencial do projeto, estabelecendo o contexto no qual os métodos técnicos são aplicados, os produtos de trabalho (modelos, documentos, dados, relatórios etc.) produzidos e os marcos estabelecidos; então, a qualidade é assegurada e as modificações são adequadamente geradas (PRESSMAN, 2006; SOMMERVILLE, 2003).

Os modelos foram originalmente propostos para colocar ordem no caos do desenvolvimento de *software*. A construção dessas representações gráficas tem por finalidade comunicar a estrutura e o comportamento do sistema, visualizar e controlar a sua arquitetura, gerenciar os riscos, compreendê-lo melhor e descobrir oportunidades de simplificação e de reuso. Além disso, proporciona um guia para a construção do sistema, permitindo a documentação das decisões tomadas.

Há alguns modelos de processo produzidos pela Engenharia de *Software*, com diferentes perspectivas para ajudar gestores e desenvolvedores de sistemas. A seguir, os modelos genéricos de desenvolvimento de sistemas são abordados.

O modelo em cascata sugere uma abordagem sistemática e sequencial para o desenvolvimento de *softwares*, que começa com a análise e a especificação dos requisitos estabelecidos por meio de consulta ao cliente e progride ao longo do planejamento, da modelagem, da construção e da implantação, culminando na manutenção progressiva do *software* acabado. É adequado para servir como modelo de processo útil em situações nas quais os requisitos são fixos,

bem compreendidos e os trabalhos devem prosseguir de forma linear (PRESSMAN, 2006).

No desenvolvimento evolucionário, as atividades de especificação, de desenvolvimento e de validação são realizadas concorrentemente, com rápido *feedback* por meio dessas atividades. Há dois tipos: o exploratório, em que o sistema evolui com o acréscimo de novas características, à medida que elas são propostas pelo cliente; e prototipagem, em que o objetivo é compreender os requisitos do cliente e, a partir disso, desenvolver uma melhor definição de requisitos para o sistema (SOMMERVILLE, 2003).

O processo de iteração da abordagem evolucionária apoia-se em dois modelos: incremental e espiral (ibid.).

O modelo incremental combina elementos do modelo em cascata, aplicado de maneira iterativa. Aplica sequências lineares que produzem "incrementos" do *software* passíveis de serem entregues. Um plano é desenvolvido para o próximo incremento como resultado do uso e/ou da avaliação. Esse processo é repetido após a realização de cada incremento, até que o produto completo seja produzido. É indicado quando não há mão de obra disponível para a implementação completa.

Já o modelo espiral mantém a abordagem sistemática passo a passo, sugerida pelo modelo cascata, mas o incorpora a um arcabouço iterativo que reflete mais realisticamente o mundo real. Cada *loop* da espiral é dividido em quatro setores: definição de objetivos, avaliação e redução de riscos, desenvolvimento e validação, e planejamento.

O desenvolvimento baseado em componentes incorpora muito das características do modelo espiral e propõe soluções através da combinação de componentes de *software* previamente preparados. As atividades de modelagem e de construção começam com a identificação de componentes candidatos. Esses componentes podem ser projetados como módulos de *software* convencional, ou como classes, ou pacotes de classes orientados a objetos. Esse modelo leva ao reuso de *software*. Contudo, as adequações nos requisitos são inevitáveis (PRESSMAN, 2006).

Considerando que os sistemas implantados no IFMT variam de pequeno a médio porte e que, para essa situação, Sommerville (2003)

recomenda a adoção da abordagem de desenvolvimento evolucionária, será proposto neste trabalho um processo híbrido: desenvolvimento evolucionário do tipo prototipagem com uso do modelo espiral para a iteração das fases do processo e o desenvolvimento baseado em componentes, essencial na agilidade do desenvolvimento.

A linguagem de modelagem adotada mundialmente para a visualização, a documentação, a especificação e a construção de sistemas orientados a objetos é a *Unified Modeling Language* (UML) – Linguagem de Modelagem Unificada, dado que fornece vocabulário e regras para a representação conceitual e física de um sistema. Ela indica como criar e ler modelos de um sistema, mas não indica quais modelos devem ser criados nem quando serão criados, pois isso é responsabilidade do processo de desenvolvimento de *software*.

Atualmente, o paradigma de desenvolvimento de sistemas mais usado é o Orientado a Objetos (OO), tendo o objeto e a classe como principal bloco de construção de todos os sistemas. Ressalte-se a facilidade de comunicação entre desenvolvedores e usuários, pela associação de objetos físicos com os de sistema.

Os sistemas Orientados a Objetos, quando construídos corretamente, são flexíveis a mudanças, possuem estruturas bem conhecidas, reduzem a complexidade no desenvolvimento de *softwares* e aumentam a produtividade de programadores. A probabilidade de o sistema ser implementado efetivamente também é considerável.

Nas seções a seguir, serão apresentadas a UML, a linguagem padrão para a modelagem de sistemas Orientados a Objetos adotada pelo *Object Management Group* (OMG) desde 1997, e alguns trabalhos relacionados a este estudo.

Utilizando a UML

A UML é uma linguagem gráfica, complexa e rica em recursos, própria para a modelagem de sistemas. É utilizada pelo projetista de *software*

na decomposição do sistema em pedaços compreensíveis, porque reduz a complexidade e facilita a visualização do desenho e da comunicação entre os objetos.

Na UML, um modelo é composto de elementos (atores, casos de uso, classes, objetos, estados, atividades etc.), relacionamentos (dependências, associações, generalizações e realizações) e diagramas (casos de uso, classe, objeto, colaboração, componentes, implantação, pacotes, atividade, sequência e estados) (LARMAN, 2000), assim descritos:

Caso de uso é um documento narrativo que descreve a sequência de eventos de um ator (um agente externo que usa um sistema para completar um processo). Auxilia na compreensão, na administração de riscos, na delimitação do escopo e na elaboração de estimativas.

A ferramenta gráfica, diagrama de casos de uso, ilustra os atores com figuras de traço simples, os casos de uso com formas ovais e a relação entre eles, com flechas que indicam o fluxo de informação ou estímulo.

Para representar conceitos ou objetos no domínio do problema, são usadas estruturas estáticas que definem operações. Ao conjunto dessas estruturas, dá-se o nome de modelo conceitual, que é o artefato mais importante da fase de análise Orientada a Objetos. Pode ser construído em paralelo ao desenvolvimento dos casos de uso.

A seguir, tem-se o diagrama de sequência, para ilustrar os eventos gerados pelos atores que são reconhecidos pelo sistema. Sua criação é dependente do desenvolvimento prévio dos casos de uso.

Por fim, o diagrama de classes ilustra as especificações para a classe de *software* e para as interfaces da aplicação. Inclui as seguintes informações: classes, associações, atributo e tipos de atributo; interfaces, com suas operações e constantes; métodos; navegabilidade; e dependências.

As ferramentas CASE (*Computer-Aided Software Engineering*) são ferramentas automatizadas de uso da Engenharia de *Software* para o desenvolvimento de sistemas. São indispensáveis na modelagem visual e também servem para apoiar a depuração e os testes.

Em acréscimo à definição da proposta de processo de desenvolvimento de *software* para o IFMT feita neste artigo, que adota o modelo

híbrido como solução, indica-se a UML para documentação dos processos e o paradigma da OO para o desenvolvimento de sistemas.

Trabalhos Relacionados

Aqui, apresenta-se um resumo da pesquisa feita sobre alguns trabalhos relacionados ao estudo em questão. Ressalta-se que não foram encontrados na literatura, em busca via internet, muitos dos trabalhos que descrevem experiências similares a esta.

Oliveira, Vasconcelos e Rouiller (2009) apresentam a proposta de um ambiente de desenvolvimento de *software* centrado no processo. Esse tipo de desenvolvimento surgiu com o objetivo de proporcionar uma estrutura computacional que gerencie o intercâmbio de informações entre os desenvolvedores, controlando as atividades realizadas, envolvendo, assim, os recursos consumidos, os prazos determinados e as datas de início de cada atividade.

A arquitetura contempla quatro tipos de usuários: projetista de processo, gerente de processo, gerente de projetos e equipe de desenvolvimento; e quatro tipos de componentes: mecanismo de interação com o usuário, mecanismo para o gerenciamento do processo no ambiente, mecanismo de repositório do ambiente e mecanismo para a integração de ferramentas ao ambiente. Embora a estrutura apresentada possa estar caracterizada em um nível de completude capaz de atender às necessidades de todas as atividades do ciclo de vida de um processo de *software*, nesse estudo, ela não foi testada empiricamente. Para ajudar a organização da implementação progressiva desse processo, utilizou-se a tecnologia CASE.

Os resultados de projetos de implantação de processo de *software* em microempresas do Estado de Goiás, descritos por Souza e Oliveira (2009), referem-se às experiências realizadas entre 2000 e 2004, envolvendo empresas de diferentes portes e com objetivos de negócio distintos. A principal conclusão é a de que modelos genéricos, direcionados para

grandes empresas, não atendem aos requisitos das empresas goianas. Além disso, subestimar os riscos de implantação de processo de *software*, ou seja, não definir um plano de contingência para os problemas reduz drasticamente as chances de sucesso. Uma boa análise de riscos, associada a um plano para a monitoração e o controle desses riscos, pode evitar o fracasso do projeto. A continuação do trabalho previu o desenvolvimento de um método de implantação de processo de *software*, baseado no MPS.BR. Os resultados obtidos nessa experiência servem de base para o planejamento de projetos de implantação de processo de *software* em empresas situadas em áreas pouco industrializadas.

Benitti, Seara e Schlindwein (2005) apresentam uma proposta metodológica para o processo de desenvolvimento de *Software* Educacional (SE) que compreende tanto as questões pertinentes aos processos quanto os aspectos relativos à concepção de SE. A proposta expõe um processo constituído basicamente de quatro etapas que se comunicam de forma iterativa e incremental: concepção, elaboração, finalização e viabilização.

Na fase de concepção, a primeira atividade concentra-se em definir os objetivos de aprendizagem e os requisitos do *software*, além de definir o escopo e o público-alvo, e identificar a infraestrutura disponível na escola.

A fase da elaboração tem como objetivo a criação de um protótipo funcional do *software* norteado pelos requisitos identificados na fase da concepção. Essa fase subdivide-se, basicamente, em quatro atividades: especificação do incremento, construção do protótipo, avaliação do protótipo e validação preliminar.

A fase da finalização ocorre após uma análise positiva da avaliação do uso do *software* pelos alunos. Essa fase subdivide-se em duas atividades: integração e elaboração da documentação.

A viabilização é a última fase do processo e é destinada aos usuários do *software*. Segundo os envolvidos nesse projeto, a aplicação desse processo tem demonstrado grande desempenho e facilidade de utilização em desenvolvimentos realizados.

Coelho (2007) descreve os passos utilizados para a implantação de um processo de desenvolvimento de *software* em uma cooperati-

va de *software* livre, denominada TecnoLivre. Partindo da análise da cooperativa, de seus projetos e de suas equipes, ressalta a importância da utilização de um processo de desenvolvimento de *software* que tenha por meta a padronização do desenvolvimento, a fim de garantir o atendimento do prazo, do preço e dos requisitos estabelecidos para o produto. O estudo de alguns modelos de processos existentes foi realizado com a finalidade de avaliá-los e escolher o que melhor se enquadrava na realidade da TecnoLivre, ou até mesmo formalizar um modelo customizado dentre as várias características dos modelos estudados. Não tendo aplicação prática, funcionou como um exercício que propôs um modelo de processo.

Todos os trabalhos acima mencionados, independentemente da estrutura de processo adotada por cada autor, ratificam a necessidade de padronização do desenvolvimento de *software* para que o produto esteja pronto dentro do prazo e conforme os parâmetros estabelecidos pela empresa.

Em seguida, será apresentada a proposta de um processo de desenvolvimento de *software* para o IFMT – Campus Cuiabá.

Proposta de um Processo de Desenvolvimento de *Software* (PDS) no IFMT

Para se propor um Processo de Desenvolvimento de *Software* (PDS) ao IFMT – Campus Cuiabá, é importante considerar aspectos como o seu ambiente (infraestrutura), a dinâmica de integração de sistemas e as atividades e/ou artefatos para esse processo.

Ambiente

Os Institutos Federais são instituições de educação superior, básica e profissional pluricurriculares e *multicampi*, especializados na oferta

de educação profissional e tecnológica nas diferentes modalidades de ensino, com base na conjugação de conhecimentos técnicos e tecnológicos com as suas práticas pedagógicas, nos termos da Lei nº 11.892, de 29 de dezembro de 2008.

A sua estrutura básica organizacional compreende, entre outros departamentos, a Diretoria de Gestão de Tecnologia de Informação (DGTI), que possui em seu rol de competências: o desenvolvimento e a implantação de sistemas informatizados, o dimensionamento de requisitos e de funcionalidades, a especificação da arquitetura, a escolha das ferramentas de desenvolvimento e a codificação de aplicativos.

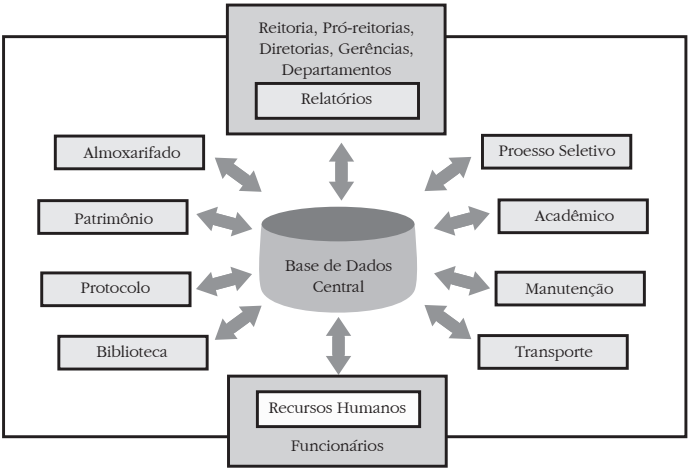
Vale ressaltar que o IFMT sofreu recente processo de mudança, passando de Centro Tecnológico para Instituto Federal. Houve a criação da Reitoria como órgão executivo. As escolas técnicas e as unidades descentralizadas de ensino deixaram de existir para formar os Institutos Federais, ocasionando a integração das equipes de desenvolvimento, que passaram a ser gerenciadas pela DGTI.

Dinâmica de Integração

O IFMT pretende integrar os sistemas administrativos, a fim de possibilitar a automação e o armazenamento de todas as informações de negócios, proporcionando maior confiabilidade dos dados e diminuição do trabalho.

A integração é possível pelo compartilhamento de informações comuns entre os diversos módulos que estão armazenados em uma base de dados central, conforme ilustrado na Figura 1. Isso significa que, após a entrada e armazenamento dos dados, o *software* integrado disponibiliza a informação para todos que dela necessitem na empresa. Faz-se necessária, por conseguinte, a implantação de critérios de segurança efetivos, claros, consistentes, em tempo real e com qualidade (REZENDE e ABREU, 2009).

Figura 1. Integração de informações entre os módulos no IFMT.



Além dos pressupostos acima, é preciso que essa integração dê exatidão e credibilidade às informações geradas, bem como propicie a eliminação da informalidade de comunicação e de papéis. A tomada de decisões também ganha outra dinâmica, pois, na execução de uma alteração, todos os módulos são informados e se preparam, de forma integrada, para determinado evento – tudo realizado em muito menos tempo do que seria possível sem a presença do sistema.

A DGTI implantou, em alguns *campi*, o SIGA-EPCT, sistema integrado de gestão acadêmica desenvolvido com tecnologias livres e de forma colaborativa pelas próprias instituições participantes da Rede de Educação Profissional, Científica e Tecnológica. Esse projeto tem o apoio do Ministério da Educação (Mec), por meio da Secretaria de Educação Profissional e Tecnológica (Setec). O SIGA-EPCT contempla dois sistemas: o SIGA-EDU, que automatiza a gestão dos processos institucionais acadêmicos (Ensino, Pesquisa e Extensão), e o SIGA-ADM, que gere os processos administrativos (Protocolo, Recursos Humanos, Almoxarifado, Compras, Patrimônio etc.).

Além do SIGA-EPCT, também foi implantado o *Enterprise Resource Planning* (ERP) *ADempiere*, que é um *framework* de código aberto.

A proposta de PDS em questão visa propiciar efetividade, continuidade, segurança e transparência à equipe de desenvolvimento.

De forma geral, os sistemas devem fornecer a visão do estado do projeto a qualquer instante, servir como meio de comunicação entre os envolvidos, indicando o seu nível de participação, manter um histórico documental do projeto, sistema ou *software*, e ser a base para a fase e subfases seguintes.

Para prover ao ambiente o controle e a evolução do PDS, serão utilizadas ferramentas específicas para repositório e integração. A definição das tecnologias, bem como as ferramentas mais adequadas para cada etapa do processo, deve acontecer no momento da sua implantação. Desta forma, tem-se um PDS independente de tecnologia, não havendo risco de defasagem tecnológica.

A equipe de desenvolvimento pode seguir o arcabouço definido neste estudo para alcançar as metas de desenvolvimento. A estrutura inclui cinco atividades: análise, projeto, implementação, implantação e manutenção evolutiva.

A análise ou concepção é a primeira etapa do PDS. Ela enfatiza a investigação do problema, produzindo compreensão ampla, mas pouco profunda do sistema. Sugere-se que essa fase não dure mais do que umas poucas semanas e produza um relatório sucinto de problemas, informando se vale a pena continuar trabalhando no problema, se há condições técnicas e financeiras, e se o cronograma é exequível. Em suma, informa se o projeto é passível de ser concluído. Antecipando-se aos problemas, o analista deve responder basicamente às seguintes perguntas: O projeto é realizável? A equipe de desenvolvimento tem condições de realizá-lo? Há tempo disponível? Pode-se comprar um pacote e adaptá-lo à necessidade do Instituto, ao invés de construir outro totalmente novo? Há riscos na execução desse projeto?

A fase de Análise prevê, para cada sistema, um conjunto de atividades e/ou artefatos, abaixo descritos.

Relatório com uma análise da viabilidade: Elaborar um documento que faça uma análise panorâmica do problema a ser tratado, descrevendo os pontos críticos do projeto, as diferentes alternativas de soluções para o problema e se o projeto será levado adiante ou não.

Levantamento dos requisitos: Prevê consultas com o cliente e com os usuários finais do sistema para a definição: dos requisitos funcionais, ou seja, as funções básicas que o sistema deve oferecer; dos requisitos não-funcionais, ou seja, as restrições colocadas sobre como o sistema deve realizar seus requisitos funcionais; das características que o sistema não deve exibir. Segundo Wazlawick (2004), os requisitos funcionais podem ser, ainda, classificados em dois grupos: evidentes, que são efetuados com o conhecimento do usuário; e ocultos, que são efetuados pelo sistema sem o conhecimento do usuário. Os requisitos não funcionais podem ser classificados como obrigatórios e desejáveis. Deve-se também estabelecer um conjunto de objetivos gerais que o sistema deve cumprir. Sugere-se utilizar um *template* em formato de tabela para elencar os requisitos: o código do requisito funcional ("F", seguido de um número); nome do requisito funcional (especificação curta); descrição (especificação longa e detalhamento do requisito); categoria funcional (evidente ou oculto). Adicionalmente, a cada requisito funcional, deve-se acrescentar uma lista de requisitos não funcionais, contendo: código do requisito não funcional ("NF", seguido do número do requisito funcional, e de um ponto e o número do requisito não funcional); nome do requisito não funcional (especificação curta); restrição (especificação longa do requisito não funcional); categoria (segurança, performance, compatibilidade, etc.); obrigatoriedade (se o requisito é desejável ou obrigatório); permanência (se o requisito é permanente ou transitório).

Organização dos requisitos: Presume-se que os requisitos já foram elencados, para, em seguida, organizá-los em grupos correlacionados de forma a abordá-los nos ciclos iterativos. Os requisitos serão agrupados do

seguinte modo: identificação dos atores e dos casos de uso, elaboração do diagrama de casos de uso e a construção do modelo conceitual que envolve as ações: listar e desenhar os conceitos candidatos, acrescentar as associações necessárias para registrar os relacionamentos e os atributos necessários para combater os requisitos de informação.

Análise de componentes: Leva em consideração o levantamento de requisitos, para fazer uma busca de componentes e implementar a análise.

Criação do glossário: Cria um documento simples, que define os termos que requerem esclarecimentos, de modo a melhorar a comunicação e padronizar a linguagem compartilhada pela equipe. Deve-se aperfeiçoá-lo em cada ciclo de desenvolvimento, à medida que novos termos são encontrados.

Elaboração de um documento de análise de riscos: Lista-se o que pode ocorrer de errado na gerência do projeto, visando transformar problemas em oportunidades. Por exemplo: um plano de contingência, de ações e de pessoas responsáveis por tratar os riscos quando ocorrerem.

Elaboração de um documento para o planejamento da comunicação: Define-se o que será comunicado, para quem, quando e de que forma.

Cronograma de execução e custos: A construção do cronograma depende dos seguintes fatores: tempo total estimado para o projeto (em hora/pessoa); tempo disponível (em semanas ou meses); tamanho da equipe; e estruturação da equipe.

A fase de Projeto visa produzir uma solução para o problema identificado na fase de Análise. Nesse momento, resta projetar uma arquitetura de *software* e de *hardware* para realizar concretamente o sistema, isto

é, apresentar uma solução para o problema enunciado. As atividades concernentes a essa etapa são:

Termo de abertura: Consiste na identificação do projeto, data e versão; participações e responsabilidades; missão, justificativa, objetivo, metas, requisitos, premissas, escopo, exclusões, riscos; prazo estimado, previsão e assinatura.

Expansão de casos de uso: Consiste em descrever o caso de uso, passo a passo. Primeiro, o caso de uso para cada requisito funcional; em seguida, o(s) ator(es) e os interessados; depois, a pré-condição e pós-condição; na sequência de passos principal (fluxo principal), descrever o processo no qual tudo dá certo; por fim, tratar as sequências alternativas associadas às possíveis exceções, ou seja, identificar e tratar as possíveis exceções de interação (fluxos alternativos). Nessa fase, um *template* deve ser usado pela equipe, para a padronização da documentação, podendo adicionar novos itens ao *template* a partir do seu uso.

Elaboração de diagrama de sequência: Deve ser construído para o fluxo principal de cada caso de uso. Nessa fase, o importante é identificar as operações e consultas de sistema necessárias.

Elaboração de modelo conceitual: Descreve a informação que o sistema vai gerenciar, baseada no domínio do problema, apresentando somente o aspecto estático da informação. Deve-se representar a informação definindo os seguintes elementos: os conceitos, que são representados pelas classes; os atributos; e as associações.

Construção de diagrama de classe: Baseia-se no modelo conceitual, adicionado de algumas informações cuja obtenção não foi possível na fase de análise. Acrescer os seguintes elementos: os métodos; as flechas de navegabilidade das associações; e a informação sobre os tipos dos atributos e dos métodos.

Elaboração de projeto da camada de interface: Deve-se dividir a camada de interface em duas subcamadas: de apresentação e de aplicação. A primeira é uma camada com as classes que representam os objetos gráficos de interface, cujas responsabilidades se resumem em receber dados e comandos do usuário e em apresentar-lhe resultados. Já a segunda é uma camada que controla a lógica de interface, abrindo e fechando janelas, habilitando e desabilitando botões. Sugere-se utilizar a avaliação heurística para a inspeção de usabilidade das interfaces, considerando os fatores expostos por Maciel et al. (2004), para a qual devem ser elaborados instrumentos e roteiros a fim de orientar os especialistas que realizarão os testes.

Elaboração de projeto da camada de persistência: Cabe ao projetista indicar quais ferramentas de persistência serão adotadas e apresentá-las aos demais membros da equipe. Essa atividade deve construir o esquema do banco de dados, criar os índices, especificar o armazenamento físico dos dados, definir as visões sobre os dados armazenados, atribuir os direitos de acesso e definir as políticas de backup dos dados.

Especificação de framework*:* Seleciona uma arquitetura reusável, que fornece a estrutura e o comportamento genéricos para uma família de abstrações de *software* dentro de um contexto.

Durante a fase de Implementação, o projeto de *software* é compreendido como um conjunto de programas ou unidades de programa, abrangendo as atividades:

Codificação: Utilizar o paradigma da orientação a objetos, elegendo a linguagem para uso de acordo com a habilidade da equipe no momento.

Testes: Utilizar o modelo espiral, iniciando pelo teste de unidade que se concentra em cada unidade do *software* implementado. Após essa

etapa, deve-se progredir para o teste de integração, em que o foco fica no projeto e na construção da arquitetura de *software*. Ato contínuo, realizar o teste de validação, em que os requisitos estabelecidos como parte da análise dos requisitos são validados em contraste com o *software* que acabou de ser construído. Por último, realizar o teste integrado de sistema: o *software* e os outros elementos do sistema são testados como um todo. Nessa fase, as interfaces podem ser novamente testadas, possivelmente, já com os usuários finais.

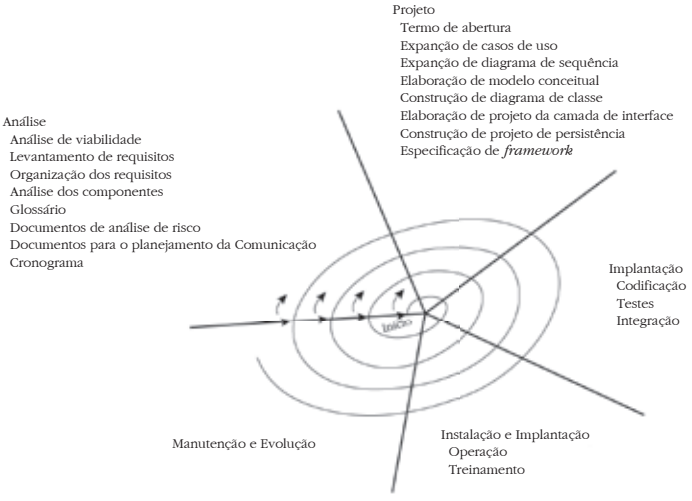
Integração: Agrupar os sistemas abordados no item 'Dinâmica de Integração', para a formação de um sistema único.

Na fase de Instalação e Implantação, o sistema é implantado no ambiente no qual deverá operar. Para essa etapa, duas atividades são necessárias: operação e treinamento. A operação envolve, quando necessário, a migração do banco de dados existente. Por sua vez, o treinamento se comprometerá com a organização de sessões de treinamento para os usuários.

Na fase de Manutenção e Evolução, devem-se corrigir eventuais erros e/ou na melhoria do sistema, ao longo da vida útil do *software*. Quando necessário, deve haver correções e atualizações nos artefatos provenientes das fases de análise e de projeto do sistema.

Os ciclos de análise, de projeto, de implementação e de testes são repetidos tantas vezes quantas forem necessárias para desenvolver todo o sistema, conforme mostra a Figura 2. Em cada ciclo, um conjunto diferente de casos de uso é abordado, segundo as prioridades do momento. Dessa forma, procura-se produzir uma solução completa e funcional para cada um dos processos associados ao ciclo.

Figura 2. Proposta para o processo de desenvolvimento de *software* no IFMT.



Conforme já relatado, para implantar esse processo com eficiência e consistência, é importante definir as ferramentas que automatizarão os processos e artefatos, e que farão o controle de versão do projeto de desenvolvimento do *software*.

Sugere-se que manuais e documentos sejam elaborados durante todo o processo de desenvolvimento do *software*, como medida de precaução contra a eventual ausência ou troca de pessoas do projeto, minimizando, assim, a dependência dos recursos humanos envolvidos no projeto.

Considerações Finais

O IFMT pretende unificar os sistemas, com o intuito de alcançar a integração total entre eles, a fim de agilizar os processos administrativos internos. Esse processo deve ser implantado gradativamente, em cada área funcional, para melhor controle e acompanhamento do processo. Para isso, a definição de estratégias depende de fatores humanos organizacionais e temporais, relacionados a cada área funcional.

O estabelecimento de um PDS é essencial nessa integração, porque permite organizar e guiar a equipe de desenvolvimento de forma ordenada, além de ser de extrema importância no desenvolvimento de novos sistemas.

A proposta do PDS sugerida neste artigo deverá ser discutida em reunião com todos os membros da equipe de desenvolvimento e com os dirigentes da Reitoria que estiverem diretamente ligados ao assunto, para aperfeiçoá-la e implantá-la. Ela é uma iniciativa de apoio à equipe de desenvolvimento.

Cabe destacar que, em meio às dificuldades enfrentadas na elaboração dessa proposta, a recente reestruturação organizacional por que passou o IFMT impactou sobremaneira o PDS, devido às constantes modificações exigidas e que representam um risco iminente, uma vez que ainda não foi consolidada.

Referências

BENITTI, F. B. V.; SEARA, E. F. R; SCHLINDWEIN, L. M. Processo de desenvolvimento de *software* educacional: proposta e experimentação. *Revista Novas Tecnologias na Educação*, Porto Alegre: UFRGS, n. 1, v. 3, mai. 2005. (Texto apresentado no V Ciclo de Palestras Novas Tecnologias na Educação.)

COELHO, A. C. *Estudo e proposta de um Processo de Desenvolvimento de Software em uma Cooperativa de Software Livre*. Lavras, 2007. Monografia (Graduação em Ciência da Computação), UFLA.

FERRAMENTAS CASE. Disponível em: imasters.uol.com.br/artigo/3048/uml/ferramentas_case/. Acesso em: 10 out. 2009.

GOVERNO FEDERAL. Lei nº 11.892, de 29 de dezembro de 2008. *Lex*: coletânea de legislação e jurisprudência. Disponível em: www.planalto.gov.br/ccivil_03/_ato2007-2010/2008/lei/l11892.htm. Acesso em: 7 ago. 2009.

LARMAN, C. *Utilizando UML e Padrões*: uma introdução à análise e ao projeto Orientados a Objetos. [s.l.]: Bookman, 2000.

MACIEL, C.; et al. Avaliação heurística de sítios na Web. Cuiabá: SUCESU-MT, 2004. (Conferência originalmente proferida na Sociedade do Conhecimento. Cuiabá: PAK Multimídia, 2004.)

OLIVEIRA, S. R. B.; VASCONCELOS, A. M. L. de; ROUILLER, A. C. *Uma proposta de um ambiente de implementação de processo de software*. Disponível em: www.dcc.ufla.br/infocomp/artigos/v4.1/art09.pdf. Acesso em: 10 out. 2009.

PAULA FILHO, W. *Engenharia de Software e fundamentos, métodos e padrões*. 2 ed. [s.l.]: LTC – Livros Técnicos e Científicos Editora S.A., 2003.

PRESSMAN, Roger S. *Engenharia de Software*. 6. ed. [s.l.]: McGraw, 2006.

RENAPI – Rede de Pesquisa e Inovação em Tecnologias Digitais. Disponível em: www.renapi.gov.br/sigaepct/o-projeto/conheca-o-projeto. Acesso em: 9 ago. 2011.

REZENDE, D. A.; ABREU, A. F. de. *Tecnologia da informação aplicada a sistemas de informações empresariais*: o papel estratégico da informação e dos sistemas de informação nas empresas. 6. ed. São Paulo: Atlas, 2009.

SOMMERVILLE, I. *Engenharia de Software*. 6. ed. [s.l.]: Addison Wesley, 2003.

SOUZA, A. S. de; OLIVEIRA, J. L. de. *Experiências de implantação de processo de software em Goiás*. Disponível em: www.softex.br/portal/softexweb/upload Documents/_mpsbr/ArtigoMPSBR.pdf. Acesso em: 11 out. 2009. (Publicado na Sociedade Brasileira para Promoção da Exportação de Software – SOFTEX.)

WAZLAWICK, R. S. *Análise e projeto de sistemas de informação Orientado a Objetos. 6*. ed. [s.l.]: Elsevier, 2004.