HOLOS

ISSN: 1518-1634

holos@ifrn.edu.br

Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte

Brasil

Pereira Pontes, Bruno; Aleixo, Fellipe; Ataíde Minora, Leonardo

PROCESSO ACADÊMICO SIMPLIFICADO: UMA PROPOSTA DE PROCESSO PARA O CEFET-RN/DATINF

HOLOS, vol. 3, diciembre, 2006, pp. 74-85

Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte

Natal, Brasil

PROCESSO ACADÊMICO SIMPLIFICADO: UMA PROPOSTA DE PROCESSO PARA O CEFET-RN/DATINF

Bruno Pereira Pontes
Graduando em Tecnologia de Desenvolvimento de Software/CEFET-RN
bppontes@hotmail.com
Fellipe Aleixo
Mestre em Engenharia da Computação/UFRN. Professor do CEFET-RN
fellipe@cefetrn.br
Leonardo Ataíde Minora
Mestre em Ciências da Computação/UFSC. Professor do CEFET-RN
minora@cefetrn.br

RESUMO

Atualmente, os cursos voltados ao desenvolvimento de sistemas computacionais não possuem um processo de *software* que atenda as suas necessidades. Processos prescritivos, como o Processo Unificado proposto pela Rational Software Corporation (*Rational Unified Process* - RUP), são complexos e burocráticos demais, dificultando o aprendizado do discente. Viu-se então, a necessidade de um processo didático para se utilizar no meio acadêmico, que permita aos discentes entenderem as etapas de desenvolvimento de um sistema computacional, aplicando, na prática, um processo mais leve, mas que abranja todas as fases do ciclo de vida de um processo de *software*. Foi desenvolvido então, um processo denominado de Processo Acadêmico Simplificado (PAS), com o intuito de facilitar o aprendizado de processo de *software* nos cursos da área de desenvolvimento de sistemas. Este artigo descreve o PAS, abordando suas principais características.

PALAVRAS-CHAVE: Engenharia de *Software*, Processo de *Software*.

SIMPLIFIED ACADEMIC PROCESS: AN PROPOSED SOFTWARE PROCESS TO CEFET-RN/DATINF

ABSTRACT

Today, the courses about software development have a generic software process. These Prescriptive Processes are too complex and bureaucratic, for example the Rational Unified Process (RUP). It's difficult the student's learning. Then, the necessity appeared of a didactic process that the student use in the academy that allows the learning to understand the stages of development of a computational system. And, this process must allow to apply a lighter and practical process but it must guarantee the use of all phases of the software process life cycle. The PAS (Processo Acadêmico Simplificado) was developed for this, with intention to facilitate the learning of process of software in the courses of development of systems. This article describes the PAS, approaching its main characteristics.

KEYWORDS: Software Engineering, Software Process.

PROCESSO ACADÊMICO SIMPLIFICADO: UMA PROPOSTA DE PROCESSO PARA O CEFET-RN/DATINF

INTRODUÇÃO

Desde o primeiro software construído aos utilizados atualmente, houve um amento na complexidade. Estes deixaram de ser apenas aplicativos com fins matemáticos para resolver problemas mais complexos como controlar sistemas aéreos, controlar calderas, sistemas de planejamento de empresas (*Enterprise Resource Planning* - ERP), entre outros.

Este aumento na complexidade do software tem ocasionado algumas mudanças no processo de construção dos softwares. Este processo de construção é denominado na literatura de Processo de Desenvolvimento de Software ou simplesmente Processo de Software. Existem diversos Processos de Software na literatura.

O Processo de Software, de uma forma simplificada, é responsável por determinar como as necessidades dos usuários serão transformadas em um produto. Este artigo tem por objetivo descrever o Processo Acadêmico Simplificado.

Objetivo do artigo

Descrever um processo de software que oriente os alunos do Curso de Tecnologia em Desenvolvimento de Software (TDS) do Centro Federal de Educação Tecnológica do Rio Grande do Norte (CEFET-RN) no desenvolvimento de software, seja em disciplinas práticas seja como projetos de extensão.

Metodologia

A experiência ocorreu no Curso de Tecnologia em Desenvolvimento de Software (TDS) do Centro Federal de Educação Tecnológica do Rio Grande do Norte (CEFET-RN). A disciplina de Processo de Desenvolvimento de Software não é uma disciplina regular do curso, mas foi oferecida no caráter optativa no semestre 2006.2. O objetivo da disciplina foi o desenvolvimento de uma customização do Processo Unificado, para ser aplicada nas disciplinas e projetos de desenvolvimento de software do curso de TDS. A intenção no desenvolvimento de tal customização foi a especificação de um processo de software com as seguintes características: simples, minimamente formal e ágil, que possibilitasse a experiência acadêmica de construção de software orientado a um processo.

A disciplina transcorreu em reuniões semanais de 4 horas-aula (3 horas), num total de 72 horas-aula (54 horas). É importante ressaltar que todos os alunos matriculados em tal disciplina já haviam cursado anteriormente a disciplina de "Engenharia de Software I", disciplina do quarto período do curso de TDS. Nesta disciplina o aluno recebeu todo o conhecimento fundamental sobre Processos de Desenvolvimento de Software e conheceu os principais processos da literatura. A maioria dos alunos possuía também o conhecimento prático da aplicação de Processo de Software, explorado na disciplina de "Prática de Desenvolvimento de Software I", onde foram desenvolvidos projetos, fazendo-se uso de um subconjunto, não documentado, do Processo Unificado. O conhecimento e a experiência anterior dos alunos envolvidos foram as bases para o desenvolvimento da disciplina.

A metodologia utilizada na disciplina baseou-se na construção coletiva de conhecimento. A cada aula os alunos apresentavam o resultado de tarefas realizadas fora de sala de aula. Tais resultados eram discutidos e modificados para refletir todas as contribuições e reflexões realizadas no debate, sempre com a mediação do professor da disciplina. Cada tarefa realizada pelos alunos teve associada a si uma nota, utilizada para compor as médias parciais e finais dos alunos na disciplina.

No primeiro encontro da disciplina de Processo de Software foi realizada uma revisão geral de todos os conceitos importantes relativos ao tema central da disciplina. No final do primeiro encontro foram passadas as tarefas individuais para serem apresentadas na aula seguinte. As atividades iniciais da disciplina objetivaram definir a estrutura base para o Processo de Software proposto, suas fases e disciplinas, bem como um nome de fantasia para o mesmo. Com a estrutura base do processo definida, as atividades passadas para os alunos foram de três tipos: (i) descrever características e objetivos de fases, disciplinas e artefatos; (ii) proposição de modelos de artefatos; e (iii) proposição de fluxo de atividades para uma dada disciplina em uma dada fase. A partir da primeira aula, cada encontro foi dividido em dois momentos: no primeiro momento os alunos expunham os resultados das suas tarefas, e no segundo momento o grupo discutia o que foi apresentado. Na discussão, as propostas de melhoramento também foram discutidas, em busca de um consenso, que buscava definir as características específicas do processo proposto.

Todas as informações geradas pelos alunos foram publicadas em um sítio Internet[1], equipado com uma ferramenta simplificada de gerenciamento de conteúdo – DokuWiki[2]. A medida que os debates e definições foram acontecendo, as informações foram sendo atualizadas no site da disciplina. Como resultado final da disciplina, foi gerado um site sobre a customização proposta para o Processo Unificado, contendo descrições, objetivos, fluxos de atividades e modelos de artefatos para cada disciplina em cada fase do processo.

Justificativa

Para ser eficaz e conduzir a construção de produtos de boa qualidade, um processo deve ser adequado ao domínio da aplicação e ao projeto específico (AMBLER, 2004). Deste modo, processos devem ser definidos caso a caso, considerando-se as especificidades da aplicação, a tecnologia a ser adotada na sua construção, a organização onde o produto será desenvolvido e o grupo de desenvolvimento. Ainda que diferentes projetos requeiram processos com características específicas para contemplar suas peculiaridades, conhecendo a tecnologia de desenvolvimento e a organização, é possível estabelecer um modelo básico de processo a ser configurado e adaptado para os projetos (AMBLER, 2004).

PROCESSO DE SOFTWARE

Software é tudo aquilo que é necessário para que um sistema computacional funcione, e eles existem devido às necessidades de clientes (SOMMERVILLE, 2003). Tais necessidades devem sofrer um processo de transformação, para que o *software* seja construído (SOMMERVILLE, 2003; PRESSMAN, 2006). Esse processo de transformação deve ter um

[1] Temporariamente disponível em http://www.cefetrn.br/~fellipe/processo/
[2] Ferramenta de Wiki disponível em http://wili.splitbrain.org/

início e um fim determinado (PMBOK, 2004). Essa variável temporal é denominada de ciclo de vida e determina as fases do desenvolvimento (transformação) (PMBOK, 2004).

Pode-se definir um processo de *software* como sendo a especificação do processo de transformar as necessidades do cliente em um produto de *software*. Ele especifica as atividades envolvidas, bem como o ciclo de vida de desenvolvimento. Uma definição mais formal é dada como sendo um conjunto coerente de atividades que objetivam transformar os requisitos do cliente em um sistema de *software* (SOMMERVILLE, 2003). Sua utilização é de fundamental importância, pois fornece estabilidade, controle e organização para atividades que, se deixadas sem controle, podem tornar-se bastante caóticas (SOMMERVILLE, 2003; PMBOK, 2004, PRESSMAN, 2006).

Diferentes processos de *software* organizam essas atividades de maneira diversas (AMBLER, 2004). No entanto, existe um conjunto delas que está presente em todos eles. Segundo PMBOK(2004), as atividades básicas de todo e qualquer processo são: (i) entender as necessidades do cliente, (ii) planejar a solução, (iii) implementar a solução, (iv) validar esta solução e (v) entregar o produto ao cliente. Tais atividades podem ser reescritas usando termos da Engenharia de *Software* como atividades de: (i) requisito, (ii) análise e projeto, (iii) implementação e (iv) teste (SOMMERVILLE, 2003).

Após conhecer as atividades básicas de um processo de *software*, é necessário entender como elas acontecem dentro deste. Essa organização é definida pelas fases (miniprojetos) do ciclo de vida de desenvolvimento. Para cada miniprojeto as atividades que serão executadas são ordenadas para que os objetivos da fase sejam alcançados (SCOTT, 2003; PMBOK, 2004). Durante o restante do trabalho será usado o termo disciplina no lugar de atividade. Isso se deve ao fato de que o conceito de atividade em processo de software envolve várias atividades, o que poderá gerar alguma confusão. Pode-se então definir disciplina como sendo uma coleção de atividades relacionadas a uma "área de interesse" principal (SCOTT, 2003).

O processo de *software* é representado de forma abstrata através de um modelo. Um modelo define um ciclo de vida, mesmo que implícito, os papéis dos desenvolvedores, o resultado das disciplinas (saída) e o que é necessário para executar as disciplinas (entrada) (SOMMERVILLE, 2003). Essas entradas e saídas são conhecidas como artefatos (SCOTT, 2003). Esses artefatos podem ser diagramas gerados, códigos fontes, lista de requisitos ou documentos (AMBLER, 2004).

SOMMERVILE(2003) classifica os modelos segundo seu ciclo em: modelo em cascata, desenvolvimento evolucionário, desenvolvimento formal de sistemas, desenvolvimento orientado a reuso e modelos híbridos. Os modelos híbridos são o espiral e o iterativo e incremental. Será considerado neste trabalho apenas o modelo iterativo e incremental, pois o processo proposto está baseado em tal modelo.

O Modelo Iterativo e Incremental

Iteração é um miniprojeto que resulta em uma versão do sistema, que contém funcionalidade adicionada ou melhorada em comparação com a versão anterior (incremento) (SCOTT, 2003; SOMMERVILLE, 2003; AMBLER, 2004). Em um processo de desenvolvimento incremental, a equipe, juntamente com o cliente, identifica as funções

mais importantes e quais são menos importantes (AMBLER, 2004). Em seguida são definidas as iterações e as funções são alocadas em cada iteração de acordo com sua prioridade (AMBLER, 2004). CANTOR(1998) apresenta as disciplinas do modelo iterativo e incremental distribuídas em quatro fases: concepção, elaboração, construção e transição.

O modelo iterativo e incremental proporciona vários benefícios, dentre os quais se podem destacar (MARTINS, 1999):

- Redução dos riscos envolvendo custos a um único incremento;
- Redução do risco de lançar o produto no mercado fora do cronograma previsto;
- Aceleração do tempo de desenvolvimento do projeto como um todo;
- Reconhecimento de uma realidade freqüentemente ignorada: as mudanças de requisitos.

O modelo iterativo e incremental é um modelo emergente (AMBLER, 2004). Seu enfoque é interessante no sentido de que usa a flexibilidade e a modularidade do desenvolvimento orientado a objeto. Assim, provê um ciclo de vida que combina a preocupação com "como as pessoas trabalham" e concede o controle de gerenciamento (CANTOR, 1999). Essas características estão presentes em grande parte das metodologias atuais, que focam os aspectos práticos do desenvolvimento, e não em questões filosóficas profundas (AMBLER, 2004).

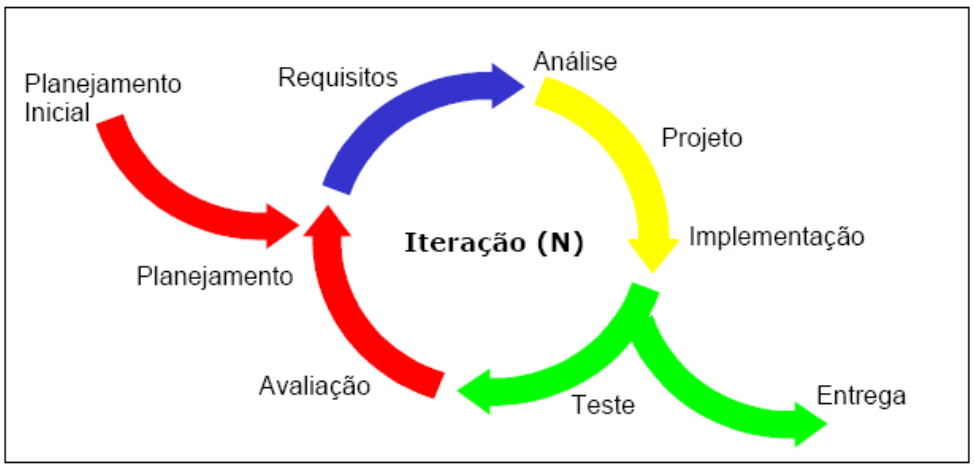

Figura 11 - Modelo Iterativo e Incremental.

O PROCESSO ACADÊMICO SIMPLIFICADO

Introdução

O Processo Acadêmico Simplificado (PAS) é um processo de desenvolvimento de *software* didático, desenvolvido para ser aplicado no meio acadêmico. Ele é baseado no modelo iterativo e incremental e tem por objetivo ser mais simplificado e menos burocrático, moldando-se as necessidades dos cursos voltados ao desenvolvimento de sistemas computacionais.

Idéias, práticas e modelos de documentos de outros processos foram inseridas no mesmo, visando o seu enriquecimento. Embora o processo seja baseado, essencialmente, no modelo

iterativo e incremental, são utilizados, mais especificamente, dois processos de desenvolvimento de *software* como orientadores na construção do PAS, são eles:

- RUP (*Rational Unified Process*) - O RUP é um processo proprietário de Engenharia de *Software* criado pela *Rational Software Corporation*, que tem como meta garantir a produção de *software* de alta qualidade que atenda às necessidades dos usuários dentro de um cronograma e de um orçamento previsível. Ele é centrado em uma arquitetura baseada em componentes, facilmente extensível, promovendo a reutilização de *software* e um entendimento intuitivo. Oferece uma abordagem baseada em disciplinas para atribuir tarefas e responsabilidades dentro de uma organização de desenvolvimento. O RUP é guiado por casos de uso, projetado e documentado utilizando a notação UML (*Unified Modeling Language*) para ilustrar os processos em ação, utilizando técnicas e práticas aprovadas comercialmente. Para maiores detalhes sobre RUP ver JACOBSON, BOOCH e RUMBAUGH(1999).

- XP (*eXtreme Programming*) - O XP é um processo leve, eficiente, flexível e de baixo risco para times pequenos e médios, que desenvolvem *software* com requisitos dinâmicos ou em constante mudança. A metodologia XP foi criada por Kent Beck, que no início dos anos 90 pensava sobre caminhos melhores para desenvolver *software*. Ele é baseado nos seguintes valores: simplicidade, comunicação, *feedback* e coragem. Para maiores detalhes sobre este processo ver BECK(2004).

Os dois processos que guiam o PAS representam dois tipos de processos: os prescritivos e os ágeis. AMBLER(2004) afirma que os processos prescritivos enfocam as disciplinas e seus resultados, possuem um paradigma de comando e controle e minimizam o papel do cliente; enquanto os processos ágeis possuem um foco nos indivíduos e nas suas interações, no *software* e não nos documentos. O cliente para ele é parte da equipe de desenvolvimento.

Dessa forma, o PAS pretende unir as virtudes dos processos prescritivos e ágeis. Ele é centrado em uma arquitetura baseada em componentes, oferece uma abordagem baseada em disciplinas, é guiado por casos de uso, seu ciclo de vida é iterativo e incremental, projetado e documentado utilizando a notação UML, características herdadas do RUP. No entanto, ele é leve, eficiente, flexível e possui os valores da simplicidade, comunicação e *feedback* definido em processos ágeis.

Fases do PAS

Foram definidas quatro fases para o PAS: Concepção, Elaboração, Construção e Validação. Elas diferenciam-se das fases do RUP basicamente pela abragência e pela nomenclatura. As diferenças existem principalmente por que o PAS possui um foco didático, enquanto o outro é focado na área comercial.

A fase de concepção para o RUP define o estudo de viabilidade e parte da análise, enquanto o PAS não considera a viabilidade, busca apenas uma visão inicial para se trabalhar o sistema. Ela deverá deixar como resultado a clara visualização do sistema a ser desenvolvido (seu escopo, seus principais requisitos e desejos do cliente). Todas as atividades desenvolvidas nesta fase têm este fim, e para tanto necessitam da participação ativa do cliente. Trata-se de uma fase marcada por reuniões e discussões onde inicialmente o cliente apresenta sua visão

do sistema, e num segundo momento os projetistas apresentam a visão que puderam capturar do mesmo. A fase é dada por terminada não necessariamente tendo passado o tempo estipulado para a mesma, mas tendo o cliente e projetistas concordados acerca do que é o sistema. Esta fase se torna opcional para o aluno quando o professor realiza as atividades.

A fase de elaboração tem como propósito analisar o domínio do problema, estabelecer uma arquitetura e desenvolver o plano de projeto. A arquitetura se desenvolve a partir de um exame dos requisitos mais significativos (aqueles que têm grande impacto na arquitetura do sistema). A estabilidade da arquitetura é avaliada através de um ou mais protótipos de arquitetura.

Embora o processo sempre tenha que acomodar mudanças, as atividades da fase de elaboração asseguram que a arquitetura, requisitos e planos são bastante estáveis, podendo-se então, determinar o custo de maneira previsível e programar a conclusão do desenvolvimento. Ao término desta fase, a "engenharia pesada" é considerada completa, e o projeto sofre seu ajuste de contas mais importante: a decisão de se comprometer à construção.

Durante a fase de construção, todos os componentes e características restantes da aplicação são desenvolvidos, testados e integrados ao produto. A fase de construção é, de certo modo, um processo industrial, no qual é enfocado o gerenciamento de recursos, controle dos custos, prazos e qualidade. Certamente, o quebra-cabeça do gerenciamento sofre uma transição do desenvolvimento de propriedade intelectual durante a concepção e a elaboração para o desenvolvimento de produtos para distribuição durante a construção e a validação. A meta da fase de construção é esclarecer os requisitos restantes e concluir o desenvolvimento do sistema com base na arquitetura.

O foco da Fase de Validação é assegurar que o *software* esteja disponível para seus usuários finais. Ela pode atravessar várias iterações e inclui testes do produto em preparação para *release* e ajustes pequenos com base no *feedback* do usuário. Nesse momento do ciclo de vida, o *feedback* do usuário deve priorizar o ajuste fino do produto, a configuração, a instalação e os problemas de usabilidade; todos os problemas estruturais mais graves devem ter sido trabalhados antes no ciclo de vida do projeto.

No fim do ciclo de vida da Fase de Validação, os objetivos devem ter sido atendidos e o projeto deve estar em uma posição de fechamento. Em alguns casos, o fim do ciclo de vida atual pode coincidir com o início de outro ciclo de vida no mesmo produto, conduzindo à nova geração ou versão do produto. Para outros projetos, o fim da Validação pode coincidir com uma liberação total dos artefatos a terceiros que poderão ser responsáveis pela operação, manutenção e melhorias no sistema liberado.

Tabela 1: Objetivos básicos das fases do PAS

FASE	OBJETIVOS
Concepção	- Delimitar o sistema; - Iniciar a arquitetura; - Criar um protótipo.
Elaboração	- Definir, validar e delinear a arquitetura; - Demonstrar que a arquitetura suportará os requisitos do sistema a um custo justo e em tempo justo; - Estabelecer um ambiente de suporte.
Construção	- Atingir a qualidade adequada com rapidez e eficiência; - Concluir a análise, o desenvolvimento e o teste de todas as funcionalidades necessárias; - Desenvolver de modo iterativo e incremental um produto completo que esteja pronto para a validação; - Minimizar custos de desenvolvimento, aperfeiçoando recursos e evitando fragmentar e refazer o desnecessário.
Validação	- Entregar e instalar o produto gerado no cliente; - Corrigir defeitos e modificar o sistema para corrigir problemas não identificados previamente;

Disciplinas do PAS

Como já foi dito, uma disciplina é uma coleção de atividades relacionadas a uma "área de interesse" principal. As disciplinas do PAS são: (i) Requisitos, (ii) Análise e Projeto e (iii) Implementação e Testes. Nesse ponto cabe uma explicação sobre a quantidade de disciplinas. Como visto anteriormente, existem 5 disciplinas básicas que todo processo de *software* possui. No entanto, visando simplificar e tornar o processo didático, as disciplinas de análise e projeto foram fundidas, bem como a de implementação e teste.

A disciplina de Requisitos é responsável pela obtenção de um conjunto de requisitos de um produto de *software* (sejam eles funcionais ou não-funcionais) que devem ser acordados entre cliente e desenvolvedor. É ela quem ajuda o desenvolvedor a entender e delimitar o escopo e as funcionalidades do sistema a ser desenvolvido. Fornece uma base para planejar o conteúdo técnico e o gerenciamento das iterações e para estimar o custo e o tempo de desenvolvimento do sistema. Nela podemos definir também um protótipo de interface de usuário para o sistema, focando nas necessidades e metas do usuário.

A disciplina de Análise e Projeto visa o detalhamento, a estruturação do sistema, além da validação dos requisitos que foram eliciados durante a disciplina de requisitos. Essa disciplina dá forma ao domínio do sistema gerando um modelo conceitual do problema, tudo isso orientado pelos casos de uso.

Na disciplina de Implementação e Testes, o projeto elaborado na disciplina anterior, é de fato concretizado em uma linguagem de programação orientada a objetos. A relação entre essas duas disciplinas é bastante estreita, pois a implementação do projeto poderá levar a ajustes e adaptações no mesmo. Dentro dos aspectos a serem considerados na disciplina de Implementação e Testes, destaca-se o planejamento e execução de testes e a aplicação das

seguintes práticas de Programação Extrema - XP: "desenvolvimento orientado a testes", "refatoração", "integração contínua", "código comunitário", "padrão de codificação" e "*releases* pequenos".

Em cada fase do desenvolvimento as disciplinas apresentam atividades diferentes e geram artefatos diferentes. As tabelas a seguir relacionam as fases e as disciplinas. A primeira (tabela 2) foca as atividades e a segunda (tabela 3) os artefatos.

Tabela 2: Atividades das disciplinas nas fases

ATIVIDADES		DISCIPLINAS		
		Requisitos	**Análise e Projeto**	**Implementação e Testes**
FASES	**Concepção**	- definição de qual será o projeto pelo cliente; - definir escopo do problema; - definir as funções e as restrições do sistema; - dividir o problema em casos de uso; - classificar casos de uso por módulo de sistema ou prioridade do cliente; - descrever brevemente os casos de uso.	- analisar ambiente físico e computacional; - analisar restrições citadas pelo cliente; - analisar modelo de casos de uso; - reunir equipe e cliente para avaliação da arquitetura candidata; - elaborar documento de arquitetura do sistema.	- capturar as telas; - confeccionar um protótipo contendo as telas iniciais do novo sistema.
	Elaboração	- detalhar caso(s) de uso escolhido(s) para validar; - consultar ou entrevistar cliente; - revisar requisitos e rastreabilidade para caso de uso; - atualizar documentos de visão em função da revisão em requisitos.	- revisar documentos de visão, arquitetura e detalhamento de casos de uso; - identificar qual caso de uso se adequa às especificações da arquitetura; - ler caso de uso; - identificar conceitos; - identificar operações; - elaborar diagramas de interação de objetos; - elaborar diagrama de classe de objeto; - implementar caso de uso.	- ler documento de arquitetura do sistema, identificar camadas e componentes; - estudar o que foi analisado e projetado na disciplina de analise e projeto; - planejar teste funcional; - planejar teste unitário; - implementar as classes do componente; - efetuar teste unitário para a classe.

ATIVIDADES		DISCIPLINAS		
		Requisitos	**Análise e Projeto**	**Implementação e Testes**
	Construção	- detalhar caso de uso; - identificar mudanças nos requisitos; - atualizar documento de visão, se for o caso; - atualizar diagrama de casos de uso, se for o caso.	- ler caso de uso; - identificar conceitos; - identificar operações; - identificar componentes; - elaborar diagramas de interação de objetos; - elaborar diagrama de classe de objeto; - implementar caso de uso.	- planejar integração; - dividir componentes entre as equipes; - construir teste unitário; - implementar componente; - realizar teste unitário; - integrar o sistema; - realizar teste de integração.
	Validação	Caso haja mudança nos requisitos: - identificar mudanças nos requisitos; - atualizar documento de visão, se for o caso; - atualizar diagrama de casos de uso, se for o caso.	Caso haja acréscimo de novas características: - ler caso de uso; - identificar conceitos; - identificar operações; - identificar componentes; - elaborar diagramas de interação de objetos; - elaborar diagrama de classe de objeto; - implementar caso de uso.	- planejar testes *alfa* em ambiente de desenvolvimento; - executar teste *alfa* em *build* estável no ambiente de desenvolvimento; - executar *build's* necessários, integrar e criar um *release*; - acordar junto ao usuário testes *beta* que rasteiem a aceitação, a usabilidade e a satisfação do cliente; - executar teste *beta* e coletar informações; - gerar relatório com dados coletados; - no caso de erros, - verificar se necessita passar por implementação ou remodelagem.

Tabela 3: Artefatos das disciplinas nas fases

ARTEFATOS		DISCIPLINAS		
		Requisitos	**Análise e Projeto**	**Implementação e Testes**
FASES	**Concepção**	- documento de visão	- documento de arquitetura do sistema	- telas do sistema
	Elaboração	- documento de visão revisado - documento de detalhamento de caso de uso	- documento de arquitetura do sistema	- plano de testes
	Construção	- documento de visão revisado - documento de detalhamento de caso de uso	-	- código - plano de integração
	Validação	-	-	- código

Princípios Básicos

Como já dito anteriormente, o PAS é centrado em arquitetura, oferece uma abordagem baseada em disciplinas, é guiado por casos de uso, iterativo e incremental e possui os valores da simplicidade, comunicação e *feedback*. Tanto a abordagem baseada em disciplinas, bem como o conceito de iterativo e incremental já foram explanados. Cabe uma breve explicação sobre os demais conceitos.

Para entender o conceito de "dirigido por casos de uso", se faz necessário definir o que vem a ser um caso de uso. SCOTT(2003) define um caso de uso como sendo uma seqüência de ações executadas por um ou mais atores (pessoas ou entidades não-humanas fora do sistema) e pelo próprio sistema, que produz um ou mais resultados de valor para um ou mais atores. Casos de uso capturam requisitos funcionais e todos juntos resultam no modelo de casos de uso, o qual descreve a funcionalidade completa do sistema. Para maiores detalhes sobre casos de uso ver JACOBSON(1992) e COCKBURN(2001)

Para um processo de *software*, ser direcionado a casos de uso significa que o processo executa uma seqüência de tarefas derivadas deles. Eles são especificados, projetados e servem de base para a construção dos casos de teste. Os casos de uso direcionam o processo de desenvolvimento, já que, baseados no modelo de casos de uso os desenvolvedores criam uma série de modelos de projeto e implementação que os realizam efetivamente. Os responsáveis pelos testes realizam seu trabalho com o propósito de garantir que os componentes do modelo de implementação cumpram corretamente os objetivos estabelecidos nos casos de uso. Desta forma, os casos de uso não somente iniciam o processo de desenvolvimento, mas também o mantém coeso.

Já o termo arquitetura, dentro do contexto de *software*, possui vários significados. A melhor forma de entendê-lo é fazer um paralelo com a arquitetura na construção civil. As construções são observadas sob vários pontos de vista: estrutura, serviços, condução de calor, encanamento, eletricidade, etc. Da mesma forma, a arquitetura em um sistema de computacional é descrita como sendo as diferentes visões desse sistema. Entre os elementos da arquitetura de *software*, estão incluídos elementos estáticos, elementos dinâmicos, o modo como esses elementos trabalham juntos e o estilo arquitetônico total que guia a organização do sistema.

A arquitetura é bastante importante para o PAS, pois a partir dela se tem: (i) um entendimento da visão global, (ii) o esforço de desenvolvimento é organizado, e (iii) as possibilidades de reuso e evolução do sistema são facilitadas. Embora seja verdadeiro o fato dos casos de uso dirigem o processo, eles não são selecionados isoladamente. São desenvolvidos juntamente com a arquitetura do sistema. Ou seja, os casos de uso direcionam a arquitetura do sistema, que por sua vez influencia a seleção dos casos de uso. Portanto, ambos amadurecem no decorrer do ciclo de vida do sistema.

CONCLUSÃO

Seguir um processo de *software* é de fundamental importância para que o produto que está sendo desenvolvido seja funcional, cumpra com os seus requisitos e tenha qualidade. Na

área de *softwares* comerciais já existem muitos processos conhecidos e que funcionam, garantindo o sucesso do *software*. No entanto, na área acadêmica, muitas vezes esses processos já existentes são difíceis de serem aplicados, pois são focados em situações que não pertencem ao escopo da academia, além de muitas vezes serem burocráticos. Dessa forma, o Processo Acadêmico Simplificado (PAS) é um processo que visa o aprendizado do aluno, não deixando de lado, porém o que os processos já existentes possuem de melhor.

Ao se adotar como processo de desenvolvimento o PAS é possível utilizá-lo de acordo com os objetivos específicos da disciplina que o está usando, pois ele foi desenvolvido de modo bem flexível, deixando aberta a possibilidade de ser aplicado tanto para alunos iniciantes como para aqueles mais experientes.

REFERÊNCIAS BIBLIOGRÁFICAS

1. AMBLER, S.W. Modelagem Ágil: Práticas eficazes para a Programação eXtrema e o Processo Unificado. Bookman. 2004.
2. PRESSMAN, R.S. Engenharia de *Software*. McGraw-Hill. 2006.
3. SOMMERVILLE, I. Engenharia de *Software*. Pearson Addison Wesley. 2003.
4. CANTOR, Murray R. Object-Oriented Project Management with UML. New York: Ed. John Wiley & Sons. Cap.03, p.93-95. 1998.
5. MARTINS, Vidal. Bate Byte 89 - O Processo Unificado de Desenvolvimento de *Software*. 1999. Disponível em: <http://www.pr.gov.br/batebyte/edicoes/1999/bb89/*software*.htm>. Acesso em: 15 nov. 2006.
6. BECK, K. Programação extrema explicada: acolha as mudanças. Bookman. 2004.
7. SCOTT, K. O processo unificado explicado. Bookman, 2003.
8. JACOHSON, Ivar, BOOCH, Grady, and RUMBAUGH, James. The Unified Software Development Process. Addison-Wesley 1999.
9. JACOBSON, Ivar, et al. Object-Oriented Software Engineering: A Use Case Driven Approach. Addison-Wesley, 1992.
10. COCKBURN, Alistair. Writing Effective Use Cases. Reading, MA.: Addison-Wesley, 2001.
11. Project Managment Body of Knowledge Guide (PMBOK'2004). Project Managment Institute, Inc (PMI), 2004.