MODELAGEM DO PROCESSO DE DESENVOLVIMENTO E MANUTENÇÃO DE *SOFTWARE* PARA A UINFOR/UESB

Roberto dos Santos Rocha*

RESUMO

Este trabalho tem como objetivos analisar as atividades de desenvolvimento e manutenção de *software* da Unidade Organizacional de Informática (UINFOR) da Universidade Estadual do Sudoeste da Bahia – UESB, *campus* de Vitória da Conquista, e propor a modelagem de um processo de desenvolvimento de *software*. A modalidade de pesquisa adotada foi o estudo de caso, e os principais instrumentos utilizados para a coleta de dados foram a entrevista e a observação direta. Partindo-se do entendimento de que a existência de um processo de desenvolvimento de *software* simplificado proporcionaria redução de tempo e custo para a execução dos projetos, assim como uma melhoria significativa na qualidade do *software* produzido, foram feitas comparação e análise das diversas metodologias de desenvolvimento existentes na literatura da Engenharia de *Software*, a fim de identificar a mais adequada. Desse modo, optou-se pelo processo da Fábrica de *Software* III porque apresenta uma estrutura leve, é simplificado e pode ser facilmente adaptado às reais necessidades da UINFOR. A máxima atual é produzir em larga escala, com qualidade e de forma customizada. Assim sendo, por um processo adaptado à realidade da UINFOR, chegar-se-á a uma maior produtividade com satisfação das áreas internas usuárias.

Palavras-chave: Engenharia de *Software*, Processos de *Software*, Processos Organizacionais.

INTRODUÇÃO

No cenário atual, os sistemas de informação representam um papel importante na administração das organizações, tornando-se imprescindível para os administradores compreender como esses sistemas devem ser projetados, implementados e administrados.

A produção de um *software* de qualidade que atenda às necessidades dos clientes compreende uma complexa inter-relação dos fatores organizacionais, culturais, tecnológicos e econômicos (VIDOTTI, 2003).

As funções da administração, principalmente o planejamento e o controle, são necessárias para o bom desempenho organizacional, e somente com sistemas de informação

de qualidade, alinhados à estratégia organizacional é que se conseguirão informações precisas para que os administradores possam alcançar os objetivos organizacionais de forma eficiente e eficaz nos tempos atuais.

A qualidade do *software* está intrinsecamente ligada à questão dos processos organizacionais. É imprescindível que a organização possua uma metodologia de desenvolvimento – processo de transformação do projeto de *software* no produto final –, a fim de reduzirem o tempo e o custo com o desenvolvimento, garantindo, assim, uma melhoria significativa na qualidade dos sistemas produzidos.

A tecnologia da informação (TI) permite que os processos organizacionais se tornem mais ágeis e mais produtivos, fazendo com que as necessidades dos clientes sejam satisfeitas com menor tempo e menos recursos. Entretanto, é importante que os administradores busquem sempre a melhoria desses processos.

O problema enfocado neste trabalho está relacionado com as diversas áreas de TI das instituições que ainda não se preocupam em sistematizar as atividades executadas para o desenvolvimento e manutenção de *software*. A Unidade Organizacional de Informática (UINFOR) da Universidade Estadual do Sudoeste da Bahia, *campus* de Vitória da Conquista, se apresenta fazendo parte desse contexto. Assim, o artigo tem como objetivos analisar as tarefas desempenhadas para o desenvolvimento e manutenção de *software* da UINFOR e propor a modelagem de um processo de desenvolvimento com base nas diretrizes definidas pela Engenharia de *Software*, a fim de se alcançar uma maior produtividade e satisfação dos clientes internos.

A relevância deste trabalho se deve ao pioneirismo da pesquisa na UESB, concernente ao apoio no desenvolvimento das atividades e tarefas da Unidade de Informática. A pesquisa poderá ser utilizada no âmbito da UINFOR, possibilitando uma visão sistêmica da TI, atualmente em uso, e dos processos organizacionais existentes. A partir dela, poderá ampliar a discussão entre os dirigentes da Instituição sobre a temática, com o objetivo de elevar os níveis de eficiência, melhorando consideravelmente os serviços prestados pelo setor.

O artigo está organizado em sete seções. Na seção dois, serão apresentados os conceitos básicos sobre funções administrativas, processos organizacionais e os seus principais elementos e objetivos, e também será discutida a relevância de se planejar estrategicamente a TI para as organizações. A seção três trata dos processos de desenvolvimento de *software*, paradigmas de processos e, ainda, aborda um método para desenvolvimento de *software*: o RUP – *Rational Unified Process*. A seção quatro apresenta a metodologia aplicada na pesquisa. Na seção cinco, é feita uma breve descrição da estrutura organizacional, papéis e funções desempenhadas, bem como a análise das atividades de desenvolvimento e manutenção de *software* da UINFOR. A seção seis descreve um processo de desenvolvimento e manutenção de *software* proposto à UINFOR; e, finalmente, a seção sete apresenta as conclusões e sugestões para trabalhos futuros.

FUNÇÕES DA ADMINISTRAÇÃO

As funções da administração – planejamento, organização, liderança e controle – são extremamente necessárias para o bom desempenho da organização. O planejamento e o controle são funções que merecem um grau maior de importância, tendo em vista que elas fornecem as informações precisas para os administradores poderem manter-se na direção dos objetivos organizacionais (STONER; FREEMAN, 1999).

Define-se a administração como o processo de planejar, organizar, liderar e controlar os esforços realizados pelos membros da organização e o uso de todos os recursos organizacionais para alcançar os objetivos estabelecidos (STONER; FREEMAN, 1999). Para uma melhor compreensão desse conceito, faz-se necessário a definição de cada uma das funções ou atividades que cabem ao administrador. Assim, planejar significa que os administradores pensam antecipadamente em seus objetivos e ações, e que seus

atos baseiam-se em um plano ou lógica. Organizar refere-se ao processo de identificar, dividir e alocar o trabalho. Liderar é o processo de conduzir um grupo, influenciando seu comportamento para atingirem objetivos e metas de interesse comum. Finalmente, a função controlar assegura que as atividades da organização levam-na em direção aos objetivos que foram previamente estabelecidos (LACOMBE; HEILBORN, 2003; STONER; FREEMAN, 1999).

Esses conceitos foram abordados neste trabalho com o objetivo de tornar evidente a importância de um bom planejamento, organização, liderança e controle para as organizações. Também os paradigmas de processos de *software* e a metodologia de desenvolvimento do *Rational Unified Process* (RUP), que serão expostos nas próximas seções, apresentam em sua lógica essas funções administrativas.

Processos organizacionais

Nesta subseção, são apresentados os conceitos básicos sobre processos organizacionais, seus principais elementos e objetivos, bem como é discutida a relevância de se planejar estrategicamente a tecnologia da informação para as organizações.

Diversos autores tratam de processos organizacionais. Cruz (2002, p. 106) define processo como:

> A forma pela qual um conjunto de atividades cria, trabalha ou transforma insumos (entradas), agregando-lhes valor, com a finalidade de produzir bens ou serviços, com qualidade, para serem entregues a clientes (saídas), sejam eles internos ou externos.

Ainda, segundo Cruz (2002), todo processo é composto de cinco elementos e dois objetivos. Os elementos são: *(I) Insumos* – fatores que entram na produção de bens ou serviços; *(II) Recursos* – referem-se aos elementos que dão suporte à função produção; *(III) Atividades* – são as menores partes de um processo; *(IV) Informações* – referem-se à coleção de fatos que são capturados, gerados, transmitidos ou manuseados pelo processo, como subproduto ou produto da execução das atividades que a compõe; e *(V) Tempo* – trata-se de um momento ou ocasião apropriada para que o bem ou serviço seja produzido.

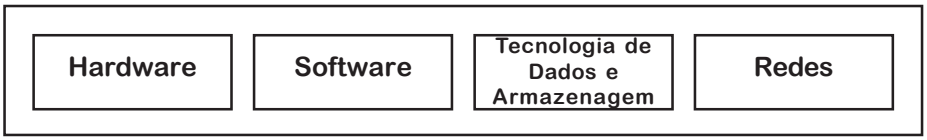
Os objetivos referem-se às metas e aos clientes. As metas são os objetivos mensuráveis do processo. O cliente é o principal objetivo de qualquer processo, e pode ser classificado em dois tipos: clientes internos – são todos aqueles que desempenham atividades dentro da organização; clientes externos – são aqueles que compram os bens ou serviços produzidos pela empresa (CRUZ, 2002).

Cruz (2002) alerta que com toda a tecnologia existente hoje, as empresas ainda correm o risco de não usar todo o potencial colocado a sua disposição e, por isso, naufraga tentando atingir ganhos de produtividade diferenciados. Para ele, os ganhos que poderiam advir com a implantação de novas tecnologias ficam aquém do esperado, e tudo isso se deve, em grande parte, ao elevado número de processos ineficientes.

Laudon e Laudon (2005) afirmam que os administradores de hoje devem saber como estruturar e coordenar as diversas tecnologias de informação para atender às necessidades da organização como um todo. Rezende (2005) conceitua a TI como "o conjunto de recursos computacionais para manipular dados, gerar informações e conhecimentos".

Os recursos que compõem a estrutura de TI são: *hardware*, *software*, tecnologia de dados e armazenagens e sistemas de telecomunicações (LAUDON; LAUDON, 2005). Apesar dos autores não enfatizarem os recursos humanos em sua conceituação, Rezende (2005) os considera como o elemento mais importante no processo de interação com os outros componentes; sem o elemento humano a tecnologia não tem funcionalidade e utilidade. Vale salientar que todo processo organizacional é realizado por pessoas que podem ser apoiadas por ferramentas, a exemplo, das tecnologias de informação e comunicação. A figura 1 mostra os principais componentes da infra-estrutura de TI.

Figura 1
Estrutura de TI



Fonte: adaptado de Laudon e Laudon, 2005.

Uma das justificativas apontadas por Cruz (2002) para se planejar estrategicamente a TI é o surgimento da preocupação pela eficiência em todos os processos.

A ciência de administração traz métodos e técnicas para se produzir em larga escala, com qualidade e de forma customizada, e a tecnologia da informação permite que os processos organizacionais se tornem mais ágeis e mais produtivos, fazendo com que as necessidades dos clientes sejam satisfeitas com menor tempo e menos recursos.

PROCESSOS DE DESENVOLVIMENTO DE *SOFTWARE*

Antes de abordar o processo de desenvolvimento de *software*, faz-se necessário definir o que vem a ser Engenharia de *Software*. Para Sommerville (2003), a engenharia de *software* é uma disciplina que se ocupa de todos os aspectos da produção de *software*, desde os estágios iniciais de especificação até a manutenção desse sistema.

Os processos no contexto da Engenharia de *Software* estão relacionados ao processo de desenvolvimento de *software*, ou seja, às etapas necessárias à conclusão ou construção de um *software* (MEDEIROS, 2004).

A qualidade dos produtos de *software* está diretamente relacionada à qualidade do seu processo de desenvolvimento. Maciel (2003 *apud* Marques *et al.*, 2004) declara que os processos de desenvolvimento bem definidos devem ser abrangentes, isto é, definir todo o ciclo de vida; ter profundidade – definir aspectos do processo em níveis diferentes de abstração; ser flexível, prático e de fácil medição; e incluir a previsão para solicitação de mudanças (aptidão ao envolvimento).

Os paradigmas da engenharia de *software* proporcionam métodos e técnicas que mostram exatamente como se deve proceder na construção do *software* (TONSIG, 2003).

Paradigmas de processos

Nesta subseção, é apresentada uma breve descrição dos modelos de processos genéricos, muitas vezes denominados, por diversos autores, de paradigmas de processos.

Para Tonsig (2003), o desenvolvimento de *software*, qualquer que seja o modelo empregado, compreende três grandes fases: requisitos, projeto/desenvolvimento, implantação e manutenção.

Figura 2
Fase Genérica dos Modelos

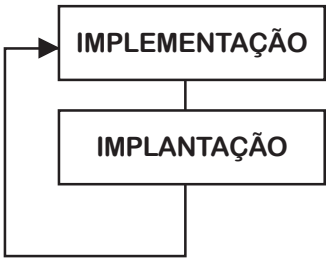
Fonte: TONSIG, 2003.

A fase de requisitos consiste no estabelecimento das fronteiras do *software*, ou seja, o que exatamente se espera que seja feito e qual será a sua abrangência. A fase de desenvolvimento é aquela em que, realmente, os analistas e os programadores irão construir o *software* propriamente dito, codificando-o em alguma linguagem de programação. A última fase refere-se à implantação e manutenção do sistema. A manutenção de *software* pode ocorrer basicamente motivada por três fatores: a correção de algum problema, sua adaptação decorrente de novas exigências e algum melhoramento funcional que seja incorporado ao *software* (TONSIG, 2003).

Modelo Balbúrdia

No modelo balbúrdia, o *software* é construído sem nenhum tipo de projeto ou documentação; normalmente, é encontrado em organizações que são administradas por crises, em que não há planejamento, nem controle com relação aos possíveis riscos.

Figura 3
Representação do modelo Balbúrdia



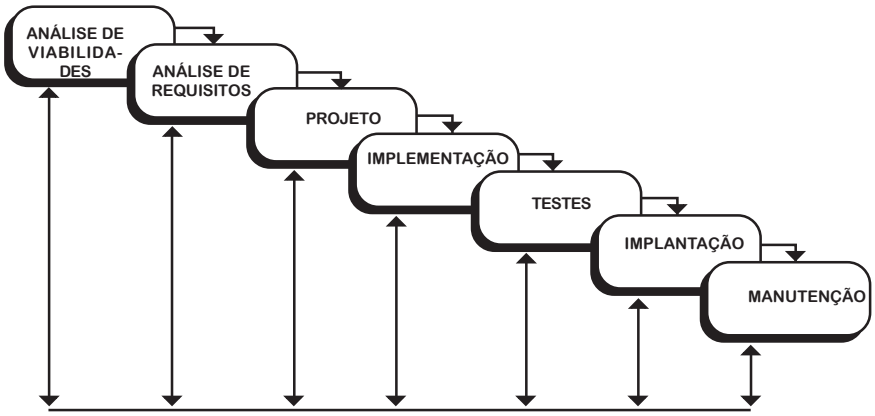
Fonte: TONSIG, 2003.

A figura 3 mostra o funcionamento desse modelo. A implementação é motivada pela urgência, desconsiderando os requisitos do sistema. Na fase de implantação do *software* já ocorrem várias mudanças que submeterão o sistema de volta à implementação. Assim, o *software* entra no que Tonsig (2003) denomina de *loop de gestação* (implementação – implantação), sem que o sistema possa atingir a maturidade ideal para uso.

Modelo Cascata

Nesse modelo, representado por Tonsig (2003) na figura 4, o desenvolvimento de um *software* se dá de forma seqüencial após a verificação de viabilidade. Optando-se pelo desenvolvimento vem a análise de requisitos para que se possam levantar as funcionalidades que deverão estar presentes no *software*. Em seguida, inicia-se o projeto, no qual se faz uma especificação técnica do *software*. Nas fases de implementação e testes, os programas que fazem parte do sistema serão codificados e testados, respectivamente. A implantação e manutenção referem-se à liberação do *software* para utilização e treinamento na produção.

Figura 4
Representação do modelo Cascata



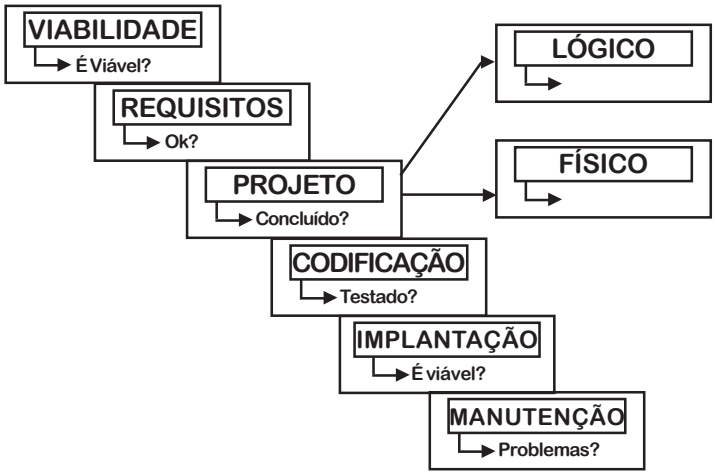
Fonte: TONSIG, 2003.

Como afirmou Sommerville (2003), o problema com o modelo em cascata é sua inflexível divisão do projeto nesses estágios distintos. Na prática, é impossível se conseguir a totalidade dos requisitos em um momento inicial do projeto, e qualquer problema que ocorra em uma das fases só poderá ser detectado após a entrega do *software* para o cliente, gerando retrabalhos ou até mesmo a extinção do projeto. Entretanto, pode-se verificar, nesse modelo, uma evolução quanto ao planejamento, organização e controle.

Modelo Incremental

O modelo incremental é uma variante do modelo cascata. Essa abordagem permite que as atividades do projeto possam ser subdivididas e que ocorram em paralelo. A fase existente no modelo em cascata foi dividida em projeto lógico e projeto físico. Como segue apresentado na figura 5:

Figura 5
Representação do modelo Incremental

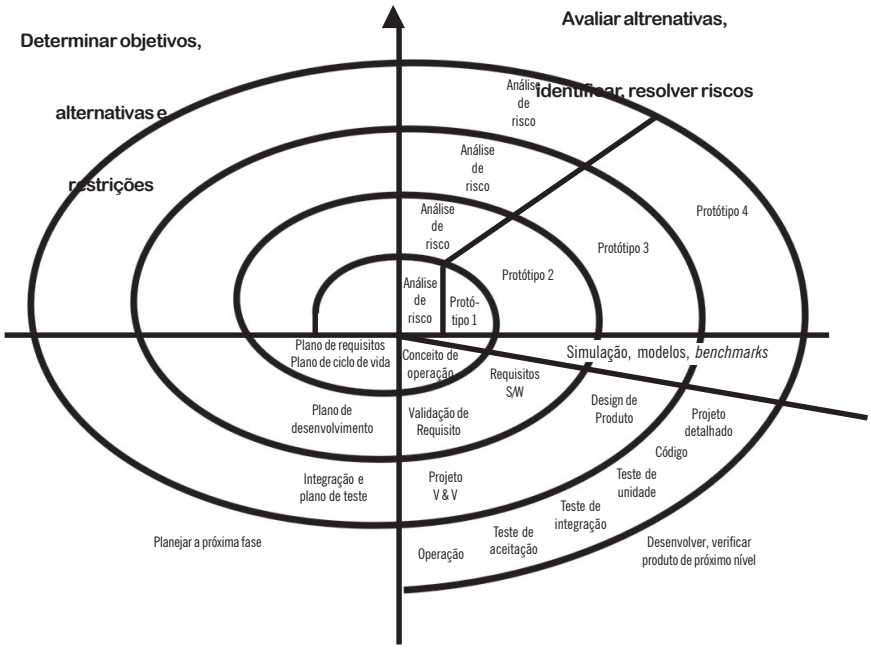


Fonte: TONSIG, 2003.

Modelo em Espiral

Na medida em que se avança pelo modelo, ocorre uma iteração, e o *software* vai evoluindo para estágios superiores, normalmente com aumento da complexidade. Cada iteração está provida das atividades determinadas pelos quadrantes que são, na seqüência: definição de objetivos, avaliação e redução de riscos, desenvolvimento e validação, e planejamento, conforme a figura 6.

Figura 6
Representação do modelo em Espiral



Fonte: SOMMERVILLE, 2003.

Esse modelo apresenta algumas vantagens em relação aos outros modelos de processos de *software*. Tonsig (2003) enfatiza que o modelo espiral induz ao desenvolvimento evolucionário do *software*, sem perder a abordagem de execução de etapas sistemáticas. Sommerville (2003) diz que a importante distinção entre o espiral e os outros modelos é a sua explícita consideração dos riscos. A sua utilização permite um maior planejamento, organização e controle sobre todas as etapas de desenvolvimento, fazendo com que os riscos sejam bem controlados. Contudo, ainda não é uma realidade em fábricas de *softwares* atuais por ser muito custosa a sua operacionalização.

Rational Unified Process - RUP

Nesta subseção são apresentadas a estrutura e as principais características do modelo de processo de desenvolvimento de *software*: o *Rational Unified Process* (RUP). Este trabalho utiliza o RUP como referência por ser um modelo que possui uma definição de atividades que contemplam desde o planejamento do projeto até os processos de testes.

Para Tonsig (2003), o RUP é um processo de engenharia de *software*, criado pela empresa *Rational Software Corporation*, centrado fortemente na arquitetura, funcionalidade e o desenvolvimento iterativo e incremental, que aplica UML para o projeto e a documentação. A *Unified Modeling Language* (UML) é uma linguagem padrão para visualizar, especificar, construir e documentar um projeto de *software.* Sommerville (2003)

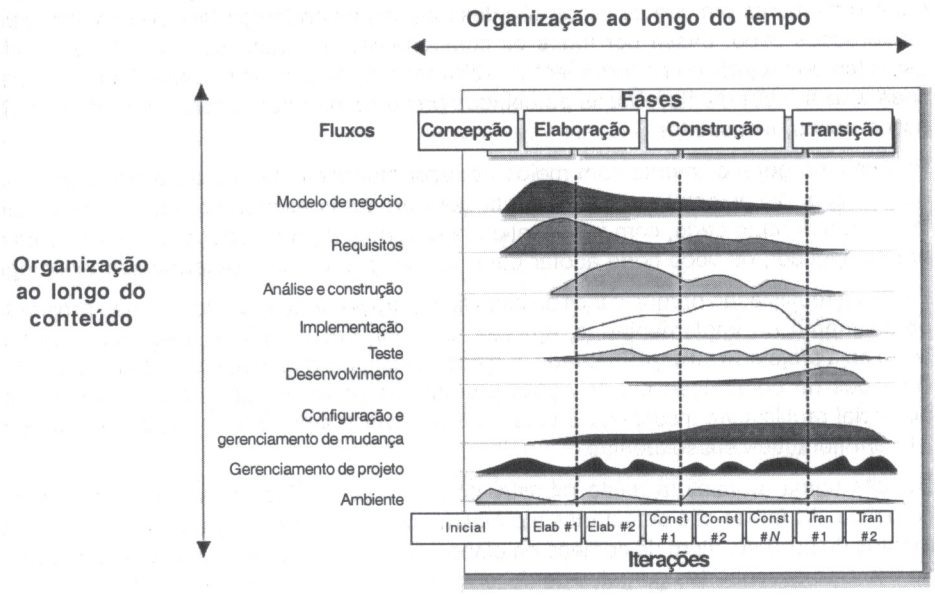
ressalta que os processos com base no modelo em cascata e no modelo incremental são amplamente utilizados para o desenvolvimento de sistemas práticos.

O RUP fornece, também, uma abordagem disciplinada para assumir tarefas e responsabilidades dentro de uma organização de desenvolvimento (KRUCHTEN, 2003). Esse processo permite um total planejamento e controle sobre os projetos de desenvolvimento de *software* porque é fortemente avaliado e documentado, o que assegura a produção de *software* de alta qualidade que satisfaça as necessidades do cliente dentro do prazo e orçamentos previsíveis.

Medeiros (2004) aponta três importantes características do Processo Unificado: *(i)* o RUP é dirigido por Casos de Uso – sucessão de ações executadas por um sistema, para a realização de uma macroatividade; *(ii)* propõe o desenvolvimento de *software* baseado em arquiteturas de *software* e componentes – o desenvolvimento baseado em componente é uma abordagem importante para a arquitetura de *software*, pois capacita a reutilização de componentes de milhares de fontes disponíveis comercialmente; *(iii)* é uma metodologia de desenvolvimento de *software* iterativa e incremental – a iteração é como se fosse um miniprojeto quase completo no qual se passa por todos os fluxos principais, e que resulta num sistema executável (incremento), porém incompleto (KRUCHTEN, 2003).

Para Tonsig (2003), o processo de desenvolvimento utilizando o RUP se dá mediante uma série de ciclos que constituem uma versão do produto. Cada ciclo é constituído de quatro fases: concepção, elaboração, construção e transição. Em cada uma dessas fases realizam-se as iterações, abrangendo uma série de *workflows* (atividades).

FIGURA 7
ESTRUTURA GLOBAL DO RUP



Fonte: KRUCHTEN, 2003.

Na fase Concepção, é construído o documento Visão do *Software*, em que serão apresentados os riscos principais ou mais aparentes e a detecção das áreas mais críticas a serem tratadas. Na fase Elaboração, ocorre o levantamento de requisitos das áreas mais críticas da organização. Na fase Construção, serão apresentados os protótipos, bem como seus relacionamentos com o banco de dados. E na fase Transição, o *software* sai da versão beta e poderá ser avaliado como versão de produção; se homologado significa que o ciclo terminou.

As atividades que compõem o RUP são as seguintes: *(i) Requisitos* – envolve a busca, compreensão e documentação dos requisitos do sistema; *(ii) Análise* – transformação dos requisitos em especificações técnicas que descrevem como implementar o *software*; *(iii) Projeto* – trata-se de atividades de planejamento, acompanhamento e desenvolvimento do projeto de gerenciamento dos riscos; *(iv) Implementação* – criação de programas de computador, segundo as especificações técnicas existentes; e *(v) Testes* – devem ser realizadas atividades de validação a fim de verificar se os resultados apresentados são compatíveis com os requisitos especificados (MEDEIROS, 2004).

É importante ressaltar que ao longo das fases existem atividades que ocorrem em intensidades diferentes. Medeiros (2004) diz que as quatro fases devem existir sempre, elas não podem ser suprimidas, entretanto as atividades dentro de uma fase podem ocorrer ou não.

METODOLOGIA

A pesquisa realizada na Unidade Organizacional de Informática da UESB pode ser classificada como exploratória, pois como afirmou Gil (2002), tais pesquisas têm como objetivo proporcionar maior familiaridade com o problema, com vistas a torná-lo mais explícito ou a constituir hipóteses.

A modalidade da pesquisa adotada é o estudo de caso, pois os fatos são analisados de forma precisa e objetiva dentro de uma única instituição, a UINFOR.

De acordo com Gil (2002), a coleta de dados no estudo de caso é feita mediante o concurso dos mais diversos procedimentos – a observação, a análise de documentos, a entrevista e a história de vida. Para Cruz (2002), a entrevista é a primeira e, talvez, a mais importante técnica de levantamento e documentação das atividades de um processo.

Nesta pesquisa são utilizados como instrumentos de coleta de dados a observação direta – o pesquisador faz parte do quadro funcional da Instituição –, entrevistas e, também, a análise de documentos. Lakatos e Marconi (1995) classificam a observação direta e a entrevista como observação direta intensiva.

A partir da construção do referencial teórico e da análise da organização do setor, procedimentos e informações existentes, bem como os recursos envolvidos, é proposto um processo de desenvolvimento de *software* para a UINFOR, buscando sua aplicação prática posterior para medição, análise e retroalimentação do mesmo.

UINFOR – UNIDADE ORGANIZACIONAL DE INFORMÁTICA - UESB

A UESB é uma instituição pública de ensino superior, que se originou, na década de 1970, das antigas Faculdades de Formação de Professores de Vitória da Conquista e de Jequié. São 35 cursos de graduação instalados em três campi – Vitória da Conquista, Jequié e Itapetinga.

A Unidade Organizacional de Informática (UINFOR) é responsável pelo planejamento, coordenação, execução e controle das atividades pertinentes à área de tecnologia da informação da UESB. Essa área ainda não possui um processo de desenvolvimento de *software* para a execução dos seus projetos; utiliza apenas como referência um documento denominado de "Padrão para Desenvolvimento de Aplicações", o qual fica disponível na *intranet* para os programadores se orientarem quanto às questões de nome de projetos, modelos de formulários, componentes, nomes de variáveis, enfim, relacionadas à documentação na implementação de programas apenas.

PAPÉIS E FUNÇÕES DESEMPENHADOS PELA UINFOR

A UINFOR se divide em três grandes áreas: desenvolvimento de sistemas, redes e manutenção. A Gerência de Desenvolvimento e Suporte é responsável pelo desenvolvi-

mento de sistemas próprios e suporte aos terceirizados que fazem funcionar os vários setores da UESB. A Coordenação de Redes e os operadores desenvolvem atividades de manutenção e ampliação da parte física e lógica da rede corporativa e suporte aos usuários. A área de Manutenção é responsável pela manutenção física dos computadores e instalação de sistemas.

Neste trabalho é dado destaque para a área de desenvolvimento de sistemas, na qual se compreende como são executadas as atividades de desenvolvimento e manutenção de *software* da UINFOR e, a partir desse entendimento, propõe-se a modelagem do processo que sistematiza as tarefas a serem executadas pelos perfis funcionais definidos.

Descrição da equipe de desenvolvimento de *software*

O Gerente de Desenvolvimento e Suporte da UINFOR tem como atividades: o levantamento de requisitos junto aos usuários, viabilização de recursos para aceleração de novos projetos e a manutenção de código-fonte.

O Programador é o profissional responsável pela codificação dos novos sistemas. Suas principais atribuições são: o desenvolvimento de programas, a partir de uma análise de sistemas definida por eles mesmos e aprovada pelo gerente de desenvolvimento; manutenção do código-fonte; planejamento e prototipação da interface gráfica dos sistemas.

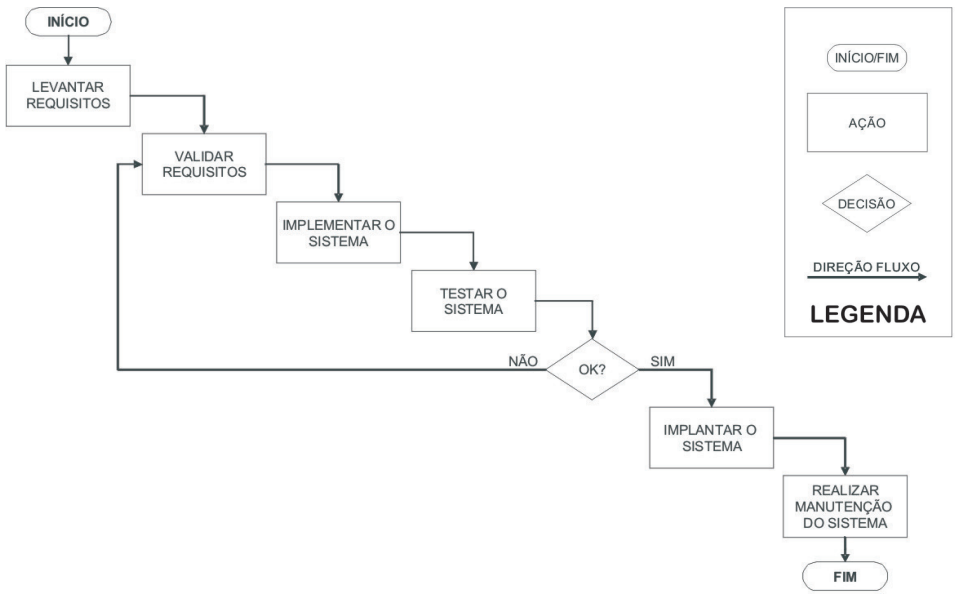
Os Operadores de Suporte são responsáveis pelo acompanhamento dos sistemas desenvolvidos pela própria Instituição e pelos sistemas terceirizados.

Análise das atividades desempenhadas pela equipe de desenvolvimento de *software*

Após observação direta intensiva e análises de documentos, foi possível compreender as atividades de desenvolvimento e manutenção de *software* da UINFOR. Para tanto, fez-se um mapeamento do fluxo das atividades utilizando a técnica de fluxogramação apontada, por Cruz (2002) como uma ferramenta de representação gráfica que permite ao analista uma visão completa das tarefas, dos responsáveis pela execução e das informações geradas, bem como sua análise e redesenho. A figura 8 retrata o desenvolvimento de *software* da UINFOR de uma maneira sintética. Proporciona, apenas, informações em um nível alto de abstração, ou seja, apresenta sua estrutura e oculta os detalhes das atividades.

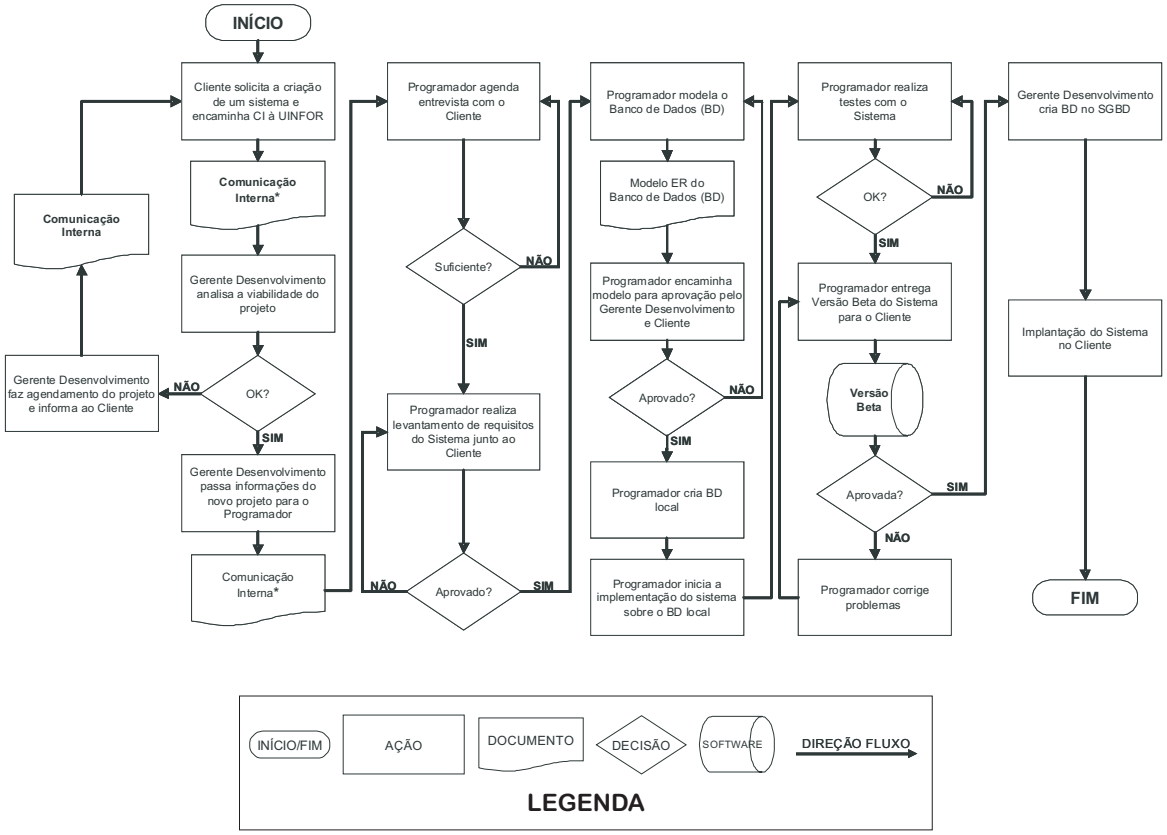
Figura 8

Fluxograma resumido do processo atual de desenvolvimento



A figura 9 abaixo mostra o mesmo fluxo com um nível maior de detalhes, em que aparecem as tarefas e ações desempenhadas pela equipe de desenvolvimento.

Figura 9
Mapeamento do processo atual de desenvolvimento de *software* da UINFOR



A partir do mapeamento do processo atual de desenvolvimento de *software* da UINFOR (figura 9), foi possível perceber que o modelo existente se assemelha muito ao balbúrdia, descrito por Tonsig (2003), pois não há planejamento e controle bem definidos para os projetos. Pode-se observar que o *software* é construído com pouca documentação e há um retrabalho contínuo, principalmente, na etapa de levantamento das necessidades dos clientes, o que certamente remete o trabalho de volta à implementação, constantemente, e sem nenhum tipo de padronização das tarefas e atividades realizadas. Verificou-se que as principais fases do projeto ficam quase que exclusivamente sob a responsabilidade do programador e sem nenhuma documentação, exceto as de programas-fontes, o que poderá inviabilizar a futura evolução do *software*.

Por meio dos fluxogramas, percebem-se, ainda, que o processo da UINFOR apresenta certa similaridade com o modelo em cascata, haja vista que as fases ocorrem de forma seqüencial, e qualquer problema que ocorra em qualquer uma das fases – levantamento dos requisitos, implementação do sistema, teste e implantação – só é detectado quando o *software* é disponibilizado ao cliente, gerando retrabalhos contínuos ou mesmo a extinção do projeto.

Desta maneira, ficou evidente que a UINFOR não utiliza nenhum processo para a produção e manutenção dos seus sistemas. Um dos resultados dessa falta de sistematização é a dificuldade na entrega do *software* dentro dos prazos e custos que deveriam ser predefinidos, o que acaba gerando uma insatisfação das áreas internas usuárias.

PROPOSTA DE MODELAGEM DE UM PROCESSO PARA A UINFOR

Nesta seção é proposto um processo de desenvolvimento e manutenção de *software* para a UINFOR, com vistas a maximizar a produtividade do setor com qualidade, levando em consideração os princípios da Engenharia de *Software*.

Para tanto, foi estudado e adaptado um processo de desenvolvimento de uma fábrica de s*oftware* (FÁBRICA DE SOFTWARE III, 2003). O conceito de fábrica de *software* não é novo, pois surgiu no fim da década de 60 quando foram criadas as primeiras fábricas nos Estados Unidos. Atualmente o termo deve ser interpretado como uma organização projetada de forma particular e sistematizada, composta por pessoas envolvidas em um esforço comum, cujas tarefas são organizadas e a padronização é utilizada com o objetivo de auxiliar na coordenação e formalização do processo (AAEN; BOTTCHER; MATHIASSEN, 1997).

O processo pesquisado da Fábrica de *Software* III foi definido e aplicado em um ambiente acadêmico – a Universidade Federal de Pernambuco (UFPE) – no programa de pós-graduação em nível *stricto sensu* de Ciência da Computação, em 2003. Optou-se por esse processo, principalmente, porque ele está fundamentado no RUP e, também, por apresentar uma estrutura leve; é simplificado e pode ser facilmente adaptado às necessidades da UINFOR.

O primeiro momento dessa proposta consiste na especificação dos papéis funcionais. Uma mesma pessoa poderá desempenhar mais de uma função dentro do processo, pois a equipe da UINFOR é restrita. A seguir, serão apresentados os papéis funcionais sugeridos à equipe, ampliando as funções já apresentadas anteriormente, passando de três para seis perfis.

QUADRO 1
PERFIS FUNCIONAIS

PAPEL	RESPONSABILIDADE
Gerente de Desenvolvimento	Responsável pelo planejamento, análise da viabilidade de desenvolvimento do projeto e acompanhamento das atividades. Define os custos e prazos, e estima o esforço do projeto.
Analista de Sistemas	Responsável pelo levantamento e análise dos requisitos de *software*.
Analista de Qualidade	Responsável pela definição do processo que garante a qualidade do *software* que está sendo produzido. Realiza auditorias de qualidade e coleta métricas ao longo de todo o projeto.
Engenheiro de *Software*	Responsável pelo projeto e desenvolvimento do *software*.
Engenheiro de Testes	Executa os testes de codificação para verificar e validar o *software* produzido.
Líder de Equipe	Coordenação e atribuição de tarefas dentro de um grupo específico, relatando ao Gerente de Desenvolvimento o andamento das atividades.

Fonte: adaptado de Medeiros *et al.*, 2004.

O segundo momento da proposta consiste na definição das fases do processo de desenvolvimento e manutenção de *software* para a UINFOR. Conforme Medeiros *et al*. (2004), a Fábrica de *Software* III possui uma metodologia de desenvolvimento de *software* dividida em quatro fases: comercial; planejamento e gerenciamento; desenvolvimento de componentes; e testes e validação. Pode-se perceber que a metodologia de desenvolvimento da Fábrica III é voltada para a produção comercial, mesmo num ambiente acadêmico. Assim, fez-se necessário a realização de adaptações das fases para adequar à realidade da UINFOR; são as seguintes: Pré-operacional; Planejamento e Gerenciamento; Desenvolvimento e Testes e Validação.

O terceiro e último momento da proposta consiste na elaboração do plano de processos. Esse plano detalha cada fase, mostrando o fluxo de atividades e tarefas a serem executadas, os documentos que deverão ser gerados em cada fase e os respectivos perfis funcionais. Medeiros *et al*. (2004) esclarecem que as atividades descritas no plano de processos expõem as informações das quais são os seus objetivos, insumos necessários à sua realização, seqüência de passos a serem executados e os resultados gerados. A compreensão do plano de processos é de fundamental importância, uma vez que irá sistematizar as tarefas, as quais são executas sem uniformização pela equipe da UINFOR atualmente. Os quadros e os fluxogramas a seguir mostram o plano de processos proposto para ordenação das tarefas.

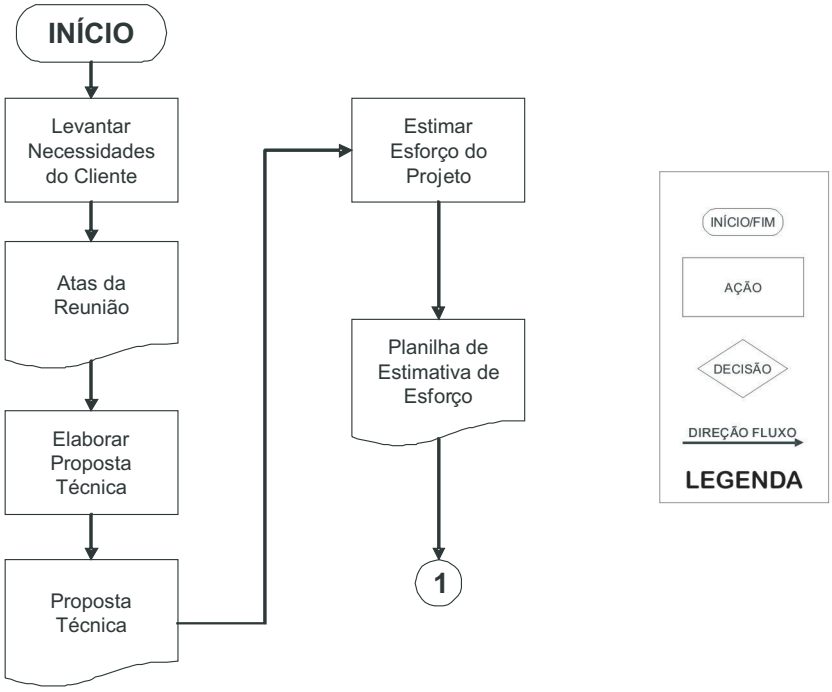
Quadro 2
Fase pré-operacional

FASE	ATIVIDADE	RESPONSÁVEL	DOCUMENTOS
Pré-operacional	Levantar Necessidades dos Clientes	Gerente de Desenvolvimento	Ata de Reunião
	Elaborar Proposta Técnica	Líder de Equipe	Proposta Técnica
	Estimar Esforço do Projeto	Gerente de Desenvolvimento	Planilha Estimativa de Esforço

Fonte: adaptado de Medeiros *et al.*, 2004.

Na fase **Pré-operacional,** o gerente de desenvolvimento será o responsável pelo levantamento das necessidades dos clientes e pela estimativa do esforço do projeto. Essa última atividade consiste basicamente do uso da métrica e da documentação de parâmetros e valores usados, para que sejam possíveis se fazer análises futuras e gerar dados históricos das estimativas. Os líderes de equipe serão os responsáveis pela elaboração da proposta técnica. A figura 10 mostra o fluxo de atividades da fase Pré-operacional:

Figura 10
Fluxograma da fase pré-operacional



Na fase de **Planejamento e Gerenciamento** o gerente de desenvolvimento será o responsável pela definição, acompanhamento, controle, comunicação do andamento do projeto e validação da proposta junto ao cliente. O analista de qualidade ficará responsável, também, pelo acompanhamento e validação do projeto junto ao cliente, gerenciamento do projeto e revisão periódica de toda a documentação. Já os líderes de equipe participarão das reuniões periódicas, a fim de comunicar o andamento do projeto e, também, serão responsáveis pela validação do projeto junto ao cliente. O quadro 3 retrata essa fase.

Quadro 3
Fase planejamento e gerenciamento

FASE	ATIVIDADES	RESPONSÁVEL	DOCUMENTOS
Planejamento e Gerenciamento	Elaborar Plano de Projetos Preliminar	Gerente de Desenvolvimento	- Plano de Projeto Preliminar
	Definir o Controle do Projeto	Gerente de Desenvolvimento e Equipe de Trabalho	- Estrutura de Desmembramento de Trabalho (WBS); - Definição do Plano Acompanhamento e Controle; - Plano de Gerenciamento de Riscos; - Plano de Gerenciamento de Configuração; - Plano de Comunicação.

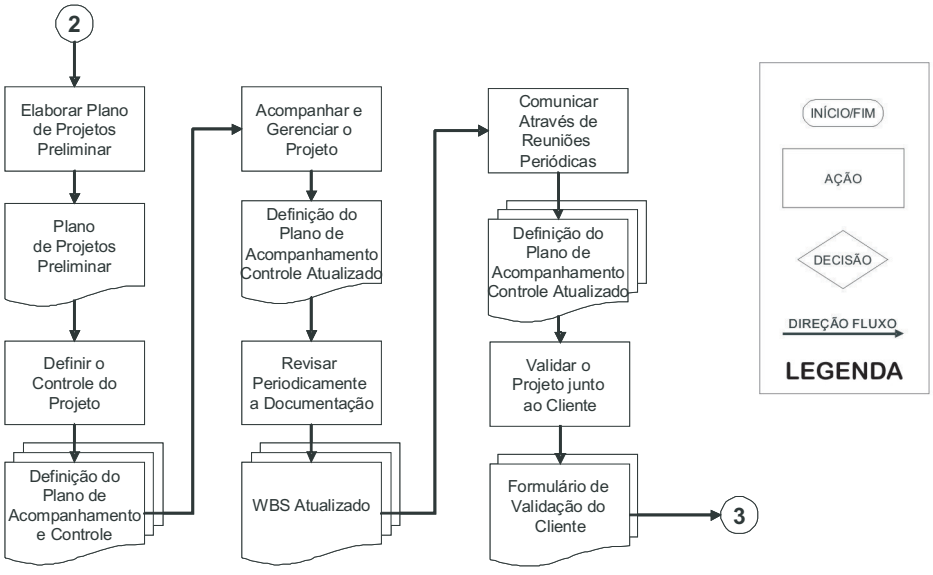
Planejamento e Gerenciamento	Acompanhar e Gerenciar o Projeto	Gerente de Desenvolvimento Analista de Qualidade	- WBS Atualizado; - Plano de Acompanhamento e Controle Atualizado; - Plano de Gerenciamento de Impactos Atualizado; - Plano de Gerenciamento de Configuração Atualizado; - Formulário de Controle de Mudanças.
	Revisar Periodicamente a Documentação	Analista de Qualidade	- WBS Atualizado; - Plano de Acompanhamento e Controle Atualizado; - Plano de Gerenciamento de Impactos Atualizado; - Plano de Gerenciamento de Configuração Atualizado; - Formulário de Controle de Mudanças Atualizado.
	Comunicar através de Reuniões Periódicas	Gerente de Desenvolvimento e Líderes de Equipe	- Ata de Reuniões; - Plano de Acompanhamento e Controle; - Plano de Gerenciamento de Riscos; - Plano de Gerenciamento e Configuração
	Validar o Projeto junto ao Cliente	Analista de Qualidade, Gerente de Desenvolvimento e Líder de Equipe	- Formulário de Validação do Cliente; - Documento Avaliando o Processo Adotado.

Fonte: adaptado de Medeiros *et al*., 2004.

A figura 11 retrata o fluxo de atividades da fase Planejamento e Gerenciamento:

FIGURA 11

FLUXOGRAMA DA FASE PLANEJAMENTO E GERENCIAMENTO



Na fase de **Desenvolvimento**, o analista de sistema é o responsável por analisar o problema em questão, definir requisitos e projetar o sistema. Para tais tarefas, o analista irá manipular os documentos correspondentes às atividades "Definir Problema". O engenheiro de *software* será o profissional responsável pela implementação do sistema. O produto da atividade desse profissional é o código gerado.

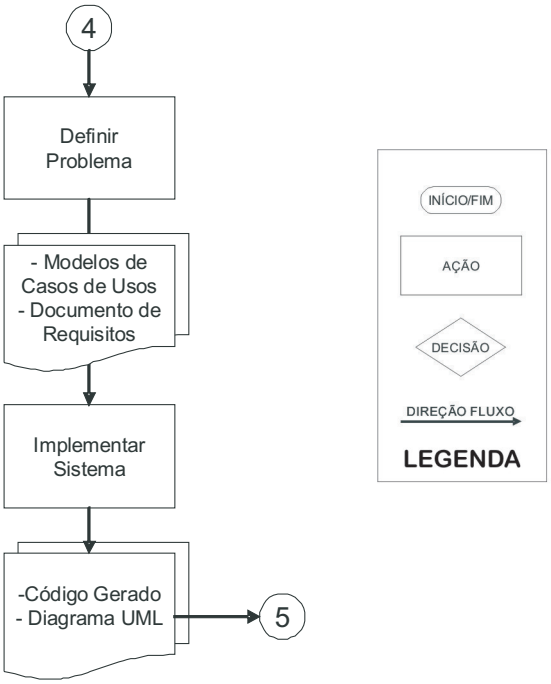
Quadro 4
Fase desenvolvimento

FASE	ATIVIDADE	RESPONSÁVEL	DOCUMENTOS
Desenvolvimento	Definir Problema	Analista de Sistemas	- Modelo de casos de uso - Documento de Requisitos
	Implementar o Sistema	Engenheiro de *Software*	- Código Gerado; - Diagramas UML.

Fonte: adaptado de Medeiros *et. al.*, 2004.

A figura 12 mostra o fluxo de atividades da fase Desenvolvimento:

Figura 12
Fluxograma da fase desenvolvimento



Na fase de **Testes e Validação** é possível detectar erros antes de o *software* ser disponibilizado aos usuários. O engenheiro de testes será o responsável pela elaboração, execução e avaliação do plano de testes. O engenheiro de *software* realizará a implementação dos testes. O usuário validador executará os testes de aceitação.

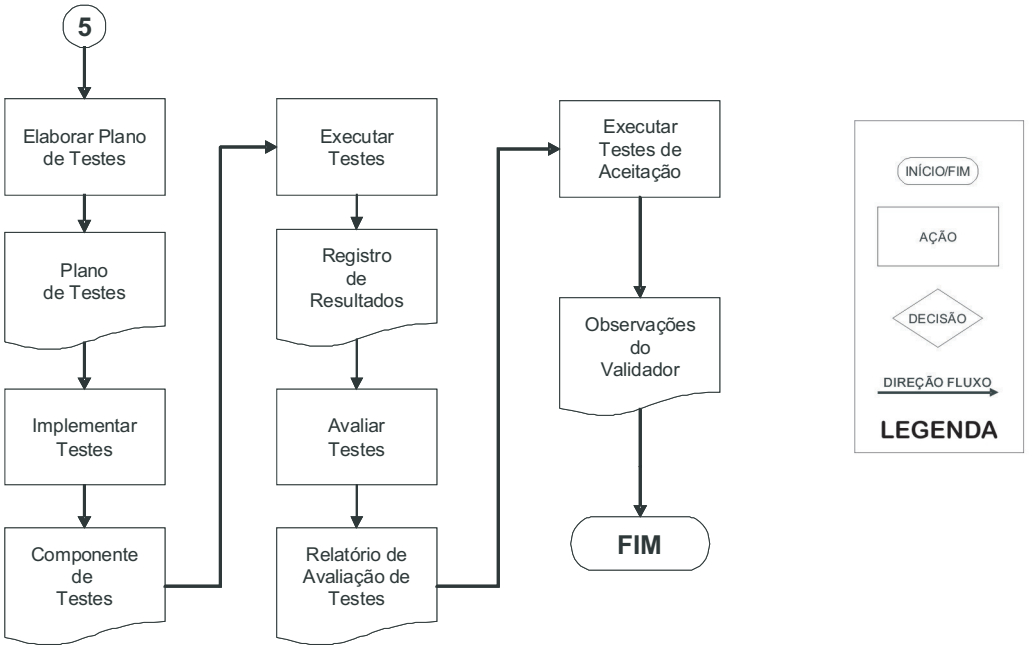
Quadro 5
Fase teste e validação

FASE	ATIVIDADE	RESPONSÁVEL	DOCUMENTOS
Testes e Validação	Elaborar Plano de Testes	Engenheiro de Testes	Plano de Testes
	Implementar Testes	Engenheiro de *Software*	Componentes de Testes
	Executar Testes	Engenheiro de Testes	Registro de Resultados
	Avaliar Testes	Engenheiro de Testes	Relatório de Avaliação de Testes
	Executar Testes de Aceitação	Usuário Validador	Observação do Validador

Fonte: adaptado de Medeiros *et. al.*, 2004.

A figura 13 mostra o fluxo de atividades da fase Teste e Validação:

Figura 13
Fluxograma da fase teste e validação



CONCLUSÃO E TRABALHOS FUTUROS

Neste artigo foi apresentada uma proposta de modelagem do processo de desenvolvimento e manutenção de *software* para a UINFOR – Unidade Organizacional de Informática, baseada na metodologia de desenvolvimento da Fábrica de *Software* III, com o objetivo de maximizar a produtividade, qualidade, eficácia e eficiência do setor.

Após a análise dos procedimentos atuais, foi possível perceber que é imprescindível à UINFOR a adoção de um processo de desenvolvimento bem estruturado para que os gestores possam planejar e controlar os projetos presentes e futuros. Somente assim, poder-se-á aumentar a produtividade da equipe de desenvolvimento e minimizar retrabalhos com a manutenção dos sistemas desenvolvidos.

As funções da administração são extremamente necessárias para o bom desempenho de uma organização e, conforme visto na literatura especializada da Engenharia de *Software*, um processo de desenvolvimento bem definido contempla todas essas funções. Dessa forma, o processo proposto à UINFOR permitirá o compartilhamento de informações entre os membros da equipe e o cumprimento de prazos e estimativas de custos. Como o processo de desenvolvimento proposto é documentado, facilitará o entendimento e continuidade dos projetos por parte dos novos programadores que ingressarem na Instituição *a posteriori*.

Convém ressaltar que alguns desafios poderão ser enfrentados para implementação dessa proposta devido aos fatores como: necessidade da mudança na percepção atual dos gestores da UINFOR/UESB para o novo paradigma de desenvolvimento de sistemas; resistências às mudanças pelo fator cultural, com o agravante da ocorrência de treinamentos simultâneos à continuidade dos trabalhos; e, ainda, dificuldades estruturais como a contratação de mão-de-obra especializada, baixos salários e restrições de vagas num ambiente público como a UESB.

Para trabalhos futuros, recomenda-se uma análise dos documentos gerados em cada fase do processo, com um diagnóstico quantitativo para mensuração da qualidade do mesmo; o desmembramento da fase de desenvolvimento, utilizando métodos e técnicas a partir do estado da arte da computação; além da realização de um Projeto Piloto na UINFOR, buscando a validação dessa proposta, futuras melhorias e novas adaptações.

NOTA

* Bacharel em Administração – Faculdade de Tecnologia e Ciências de Vitória da Conquista / Graduando do curso de Ciência da Computação – Universidade Estadual do Sudoeste da Bahia / Orientação: Prof.ª Vanessa Bittencourt Xavier de Moura / E-mail: rsantos.rocha@gmail.com

REFERÊNCIAS

AAEN, I.; BOTTCHER, P.; MATHIASSEN, L. Software factories. In: *Proceedings of the Twentieth Information Systems Research Seminar in Scandinavia,* Oslo, 1997. Disponível em: <http://www.cs.aau.dk/~larsm/Dr_Techn/Volume_II/17.pdf> Acesso em: 08 set. 06.

CRUZ, Tadeu. *Sistemas, organização & métodos:* estudo integrado das novas tecnologias da informação. 3. ed. São Paulo: Atlas, 2002.

FÁBRICA DE SOFTWARE III. *Metodologia de desenvolvimento de software.* Recife: UFPE, 2003. Disponível em: <http://www.cin.ufpe.br/~fabrica3/homePage/planoprocesso.htm> Acesso em: 19 mai. 06.

GIL, Antonio Carlos. *Como elaborar projetos de pesquisa.* 4 ed. São Paulo: Atlas, 2002.

KRUCHTEN, Phillippe. *Introdução ao RUP.* Rio de Janeiro: Ciência Moderna, 2003.

LACOMBE, Francisco; HEILBORN, Gilberto. *Administração:* princípios e tendências. São Paulo: Saraiva, 2003.

LAKATOS, Eva Maria; MARCONI, Marina de A. *Metodologia do trabalho científico.* 4. ed. São Paulo: Atlas, 1995.

LAUDON, Kenneth; LAUDON, Jane P. *Sistemas de informações gerenciais:* administrando a empresa digital. 5. ed. São Paulo: Prentice Hall, 2005.

MARQUES, Helena M. et al. Adaptação de um processo de desenvolvimento para fábricas de *software* distribuídas. In: *7º Workshop Ibero-Americano de Engenharia de Requisitos e Ambientes de Software,* IDEAS´2004, Arequipa, Peru. 3-7 maio 2004. Disponível em: <http://www.cin.ufpe.br/~in953/> Acesso em: 26 mai. 06.

MEDEIROS, E. *Desenvolvendo software com UML 2.0.* São Paulo: Makron Books, 2004.

MEDEIROS, Vivianne da Nóbrega et al. Construindo uma fábrica de software: da concepção às lições aprendidas. In: *CLEI2004 – XXX Latin-American Conference on Informatics,* Arequipa, Peru, oct-2004. Disponível em <http://www.cin.ufpe.br/~in953/> Acesso em 03 jun. 06.

REZENDE, Denis Alcides. *Sistemas de informações organizacionais:* guia prático para projetos em cursos de administração, contabilidade e informática. São Paulo: Atlas, 2005.

SOMMERVILLE, Ian. *Engenharia de software.* 6. ed. São Paulo: Addison Wesley, 2003.

STONER, James A. F.; FREEMAN, R. E. *Administração.* 5. ed. Rio de Janeiro: LTC, 1999.

TONSIG, Sergio L. *Engenharia de software:* análise e projeto de sistemas. S. Paulo: Futura, 2003.

UNIVERSIDADE ESTADUAL DO SUDOESTE DA BAHIA. *Unidade Organizacional de Informática.* Disponível em: <http://www.uesb.br/uinfor/> Acesso em: 14 mar. 06.

VIDOTTI, Cleber. *Metodologia simplificada de desenvolvimento de software para empresas de pequeno e médio porte:* uma aplicação prática na WOPM Informática. Florianópolis, 2003, 106 f. Dissertação (Mestrado em Engenharia de Produção), Programa de Pós-Graduação em Engenharia de Produção, UFSC, 2003.