1. Introdução à Análise e Projeto de Sistemas Orientados a Objetos

O princípio dos estudos da análise de sistemas deve partir, inevitavelmente, da compreensão do que é um sistema. Em seguida pode-se enquadrar este entendimento na realidade dos sistemas computacionais, que é o foco do presente trabalho.

Em nível geral, pode-se caracterizar um sistema como sendo um conjunto de elementos interdependentes, ou um todo organizado, ou partes que interagem formando um todo unitário e complexo. Dividindo-se, ainda, os sistemas em fechados e abertos, os primeiros não estabelecem troca de "material" com o ambiente externo, ou seja, nenhum "material" entra ou deixa o sistema. Exemplos de sistemas fechados são os sistemas de simulação, os quais, em geral, são projetados com o intuito de viabilizar qualidade e otimizar posteriores projetos de sistemas abertos.

Contrariamente àqueles, os sistemas abertos se caracterizam por realizar constante troca de "materiais" com o meio em que estão inseridos. Os exemplos para estes são dos mais variados, podendo-se citar os sistemas de transporte, o corpo humano, os sistemas econômicos e sociais e, dentre os mais comuns para nossos estudos, as empresas/corporações. Portanto, na grande maioria das vezes, nas tarefas de análise e projeto, os esforços irão se concentrar nos sistemas abertos.

Um sistema aberto é composto de um conjunto de partes em constante interação (interdependência das partes), constituindo um todo orientado para determinados fins e em permanente relação de interdependência com o ambiente externo. E as empresas, por que se comportam como sistemas abertos? O ambiente em que elas se encontram é essencialmente dinâmico, forçando o sistema organizacional (para sobreviver) a responder com eficácia às pressões exercidas pelas rápidas e contínuas mudanças no ambiente. Os sistemas ainda podem ser divididos em vários subsistemas, os quais também se caracterizam em um conjunto de partes interdependentes que se relacionam entre si, compondo um sistema maior.

Usando das definições acima, podemos decompor alguns sistemas:

- Corpo humano: sistema ósseo, circulatório, muscular, etc.

- Sistema de transportes: rodoviário, metroviário, ferroviário, de carga, aéreo, marítimo, etc.

Os sistemas abertos são baseados na idéia de que determinadas entradas (inputs) são introduzidas no sistema e, uma vez processados, geram certas saídas (outputs). As empresas assim se comportam através de recursos materiais, humanos e tecnológicos, cujo processamento resulta em bens ou serviços a serem fornecidos ao mercado.

Desta forma podemos definir, agora, um sistema de informação em função de subsistemas e de empresas. Um sistema de informação é formado por outros

subsistemas, cada qual sendo um sistema de informação apoiando um processo de decisão. Assim sendo, um sistema de informação pode ser definido "como um conjunto de componentes inter-relacionados trabalhando juntos para coletar, recuperar, processar, armazenar e distribuir informação com a finalidade de facilitar o planejamento, o controle, a coordenação, a análise e o processo decisório em empresas e outras organizações" (Laudon, 1999). Portanto, cada sistema/subsistema pode ser decomposto em 3 partes principais, conforme figura 1.1, sendo:

1 - Coleta de dados de entrada;
2 - Processamento dos dados;
3 - Produção e distribuição de dados de saída (informações).

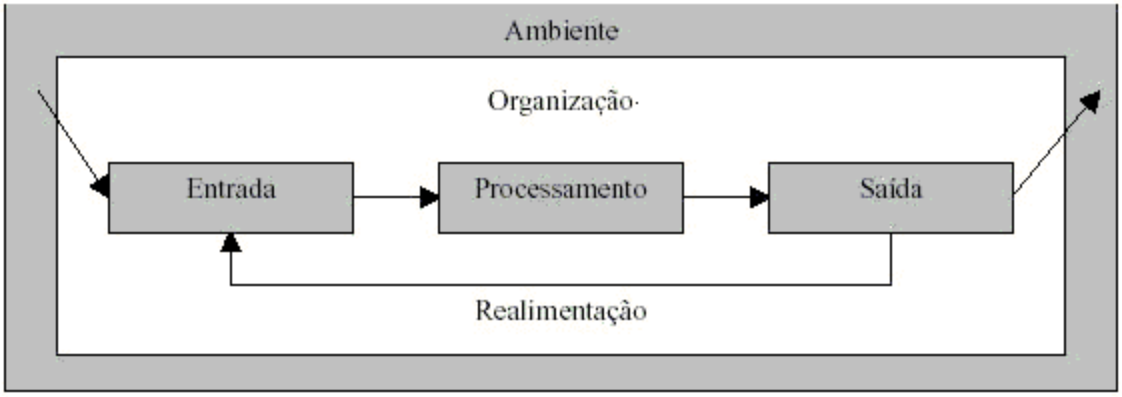

Figura 1.1 – Atividades dos Sistemas de Informação: Entrada, Processamento e Saída

De forma geral, é útil destacar os principais aspectos que descrevem o conceito dos sistemas de informações:

- O sistema total é uma extensão do processamento integrado de dados que resulta na integração de todos os subsistemas principais num único sistema;
- O sistema deve incorporar as informações necessárias para planejamento e controle;
- O sistema deve gerar informações necessárias para auxiliar os administradores de todos os níveis a atingirem seus objetivos;
- O processamento eletrônico de dados deve representar um papel importante, porque se torna necessário automatizar para prover informações exatas rapidamente.

É possível verificar que a visão sistêmica não necessariamente passaria pelos ambientes computacionais, pois já existiam antes destes. No entanto, há muito já se tornou difícil conceber um sistema de informações sem o suporte de ferramentas e

sistemas de software e hardware. Portanto, não é demais relacionar os conceitos vistos até então, com o campo de processamento de dados, e afirmar que os sistemas serão tratados como um conjunto de pessoas, máquinas e métodos organizados de modo a cumprir um certo número de funções específicas.

Todavia, é exatamente a visão sistêmica, independente da tecnologia em questão, que deve ser adotada quando se começar a estudar e proceder com as tarefas iniciais da análise de sistemas. Então, o que é fazer análise? Confunde-se, muitas vezes, que analisar é somente resolver problemas. Porém, análise é o estudo de um problema, que antecede à tomada de uma decisão/ação. Já em relação ao nosso foco de trabalho - os sistemas computacionais -, análise é o estudo de alguma área de trabalho ou de uma aplicação, levando geralmente à especificação de um novo sistema. Esta ação será o projeto e a implementação do mesmo.

No princípio da informática, geralmente os problemas a serem resolvidos eram de caráter especificamente matemáticos. Eram adotados métodos científicos de resolução de problemas e passava-se à implementação das soluções. Os programadores começaram a adotar ferramentas de modelagem, sendo os fluxogramas as mais utilizadas. Estes se mostraram voltados unicamente à especificação da solução algorítmica sem, contudo, permitir aos "analistas" da época que se concentrassem no domínio do problema em que se estava envolvido.

Neste momento da história do desenvolvimento de software surgiu a necessidade de se ter ferramentas e, principalmente, métodos sistemáticos que auxiliassem o profissional de software a percorrer o caminho da análise e projeto de sistemas. Foram então elaboradas abordagens mais completas. Eram as primeiras metodologias de análise e projeto de sistemas, as quais usavam de técnicas e ferramentas de modelagem, sendo as mais expressivas os diagramas de fluxo de dados (DFD) e os modelos entidade-relacionamento (E-R). Surgia, assim, a análise e projeto de sistemas estruturados.

Nesta época já havia as primeiras propostas de processos de construção de software. Portanto, desde então, a adoção de métodos sistemáticos de análise e projeto se mostrou extremamente necessária. Além do processo de programação, um método sistemático deve abranger também as demais fases do processo de construção de software. Em geral, a grande maioria das metodologias deve pressupor as fases de definição de requisitos, análise, projeto, implementação e testes. Esta abordagem pode, contudo, variar, dependendo dos objetivos de quem propôs o método e, claro, de quem o utiliza. Além disso, várias atividades específicas à engenharia de software também deverão ser consideradas quando se planeja construir um produto de software com qualidade, o que será tratado na seção a seguir.

Como principais características em se ter um método, cita-se:

- Oferecer sistemática para as fases de definição de requisitos, análise, projeto,

implementação e testes;

- Evitar codificação precoce: programas de difícil manutenção e que não satisfazem os requisitos do usuário;

- Métodos sistemáticos consomem mais tempo nas fases iniciais do desenvolvimento de software, mas minimizam erros no projeto e implementação e viabilizam software com maior qualidade.

Com o surgimento do paradigma de programação orientado a objetos, a história quase se repetiu, ou seja, se implementava sistemas sob um determinado prisma, mas a especificação do mesmo estava sendo feita, ainda, de maneira estruturada. Contudo, isto é normal no processo de evolução tecnológica, o que logo gerou a necessidade por métodos de análise e projeto orientados a objetos.

Várias metodologias foram desenvolvidas e apresentavam inúmeras vantagens, conforme elencados por Coleman (1996):

- Verificação de requisitos;
- Conceitos mais claros;
- Maior adequação do projeto ao problema;
- Melhor decomposição do projeto para trabalho em equipe;
- Melhor comunicação da equipe de desenvolvimento;
- Menor esforço de manutenção;

Todo o assunto até aqui abordado tem como único objetivo situar o leitor para o que virá a frente. Contudo, o tema discutido sugere leituras complementares em relação a sistemas de informação e a outras metodologias de análise e projeto, já que este aprofundamento sairia do escopo deste projeto, restringindo-nos em apresentar uma metodologia em particular de análise e projeto de sistemas orientados a objetos.

No restante deste capítulo são desenvolvidos conceitos introdutórios para o trabalho orientado a objetos, na seção 1.1, sem aprofundarmo-nos no assunto, o que será feito no momento adequado do processo. Em seguida, na seção 1.2, é feito um apanhado geral sobre o histórico das metodologias de desenvolvimento orientadas a objetos e sobre o método Fusion, dando seqüência com um breve resumo sobre a UML na seção 1.3, sendo estas duas abordagens os pilares para o projeto FILM. O presente capítulo é finalizado com a seção 1.4, onde se apresenta a estrutura resultante proposta pelo projeto FILM - Método Fusion Expandido e Adaptado à UML por Galimberti (2002), no qual é fornecida uma visão geral do processo que poderá ser adotado no transcorrer dos trabalhos de análise e projeto de sistemas orientados a objetos.

1.1. Análise e Projeto Orientados a Objetos

A orientação a objetos surgiu como uma abordagem de programação que procura explorar o nosso lado intuitivo. Os "átomos" da computação orientada a objetos (os próprios objetos), são análogos aos objetos existentes no mundo físico, o que produz um modelo de programação muito diferente da tradicional visão "funcional" e "procedimental" (Coleman, 1996).

A visão do modelo computacional orientado a funções poderia ser assim resumida:

- Os "átomos" da computação são as funções;
- As funções atuam sobre um único estado compartilhado;
- Qualquer função pode atuar sobre qualquer parte do estado compartilhado.

Já no modelo computacional orientado a objetos:

- A única maneira de se obter ou alterar o estado de um objeto é pelo envio de mensagens;
- Cada objeto "é responsável" pela resposta a uma determinada mensagem;
- Os objetos, quando compartilham uma única interface, são agrupados em classes.

Durante a execução do sistema, os objetos podem ser construídos ou destruídos, podem executar ações ou se tornar inacessíveis: estas características o tornam um modelo computacional dinâmico. Assim sendo, os "átomos" do processo de computação são os objetos:

- Os objetos trocam mensagens entre si;
- As mensagens resultam na ativação de métodos;
- O emissor da mensagem não precisa saber como o objeto receptor organiza o seu estado interno;
- Ao emissor da mensagem basta saber que o objeto receptor/servidor responde a certas mensagens de maneira bem definida.

Entretanto, o paradigma orientado a objetos deve, além do processo de programação, abranger também as demais fases do processo de desenvolvimento de sistemas computacionais. Um dos aspectos mais importantes e comuns entre os métodos de análise e projeto de sistemas orientados a objetos é viabilizar uma hierarquia com as especificações das classes de objetos que deverão ser implementadas para formar o software. Para que isto ocorra, cada metodologia deve especificar caminhos entre a definição de requisitos e a definição da arquitetura do software orientado a objetos, sendo que neste trajeto vários são os artefatos de software produzidos com o objetivo de otimizar o processo de estruturação das classes de objetos.

Historicamente foram desenvolvidas metodologias para tratar o ciclo de desenvolvimento de software sob o ponto de vista de objetos. Contudo, como um processo natural de evolução de "novas" tecnologias, este tema ainda proporciona novas propostas de abordagem para o assunto, gerando, ao mesmo tempo, divergências e similaridades entre as mesmas, sejam métodos ou mesmo processos completos de engenharia de software.

Voltando a referenciar nossa breve conversa sobre a análise de sistemas e sua história, vários estudiosos concluíram sobre a necessidade de abordagens que auxiliassem o profissional da área a construir software. Foram então desenvolvidas inúmeras pesquisas e experiências, tanto na academia como na indústria. Como resultados dos projetos de pesquisa surgiram diversas metodologias de análise e projeto de sistemas, mas também linguagens de especificação, ferramentas e técnicas de modelagem de sistemas orientados a objetos. Assim, vários trabalhos não se caracterizam necessariamente como métodos de análise e projeto, mas também contribuem em muito para este campo da informática. Assim sendo, até especificar suas características, trataremos estes trabalhos, genericamente, como abordagens de análise e projeto orientados a objetos.

Dentre as principais abordagens (métodos, técnicas, linguagens de especificação) conhecidas mundialmente, e altamente influentes para este trabalho, citam-se: "*Booch*", de Grady Booch; "OMT (*Object Modeling Technique*)", de James Rumbaugh; "*Objectory*", de Ivar Jacobson; "CRC (*Class- Responsability-Collaboration*)", de Kent Beck; "RUP (Rational Unified Process)", dos "Tree Amigos", assim comumente conhecidos os pesquisadores Booch, Rumbaugh e Jacobson, que juntos eram proprietários da Rational Software Corp. Além destas, tem-se como atualmente influentes para avanços do FILM as abordagens ditas ágeis, sendo XP - *Extreme Programming* e AM - *Agile Modeling*, respectivamente de Kent Beck e Scott Ambler.

Devido a esta miscelânea de métodos, surgiram duas boas abordagens com propostas de fusão de conceitos, sendo elas: Fusion (Coleman, 1996) e UML (*Unified Modeling Language*) (OMG, 2001). Estas se caracterizam como a base de nossas atividades e são apresentadas a seguir.

Atualmente sendo adotada como base para a prática da orientação a objetos nas disciplinas de Análise e Projeto de Sistemas, seguindo os currículos dos cursos de Bacharelado em Ciência da Computação e Tecnologia em Processamento de Dados da Universidade de Caxias do Sul, o método Fusion se apresenta bastante sistemático e didático para o processo de desenvolvimento de software orientado a objetos, estabelecendo uma rota direta entre a definição dos requisitos e a implementação do sistema. No entanto, os modelos por ele criados sugerem adequações devido a similaridade de notação entre os mesmos e, principalmente, por não ter a padronização necessária ao segmento de objetos. Além disso, o método não propõe abordagem para a definição de requisitos, que é uma das fases iniciais do processo de produção de

software, que precede as atividades de "construção" do mesmo.

Já em relação à UML, esta caracteriza-se como um conjunto de notações bem definidas e específicas para a construção dos diversos artefatos necessários desde a definição de um sistema orientado a objetos até a sua implementação. A UML necessita, contudo, de um método que indique quais os passos a serem dados para se construir o sistema de software, conforme será melhor abordado na seção seguinte.

1.2. Método Fusion de Análise e Projeto Orientado a Objetos

O método Fusion de análise e projeto de sistemas orientados a objetos foi concebido dentro dos laboratórios da empresa Hewlett-Packard tendo como motivação sistematizar o processo de construção dos sistemas de software em demanda na corporação.

O método Fusion caracteriza o centro dos estudos para nossa abordagem de orientação a objetos. Neste contexto, apresenta-se na tradução de uma fusão de conceitos, sintetizando os melhores recursos de abordagens populares de modelagem de sistemas orientados a objetos.

O método Fusion não "engloba" a captura de requisitos, não possuindo uma fase destinada a isto. Portanto, o método prevê que os requisitos sejam fornecidos por usuários e documentados seguindo-se alguma técnica de elicitação e de especificação de requisitos "qualquer", passando-se em seguida à fase de análise.

Já na na fase de análise, o propósito é definir "o que" o sistema faz e não "como" é feito. Em contraste com outros métodos, na análise do Fusion não são associados métodos às classes, o que deve ser feito na fase de projeto. Em geral, a fase de análise comporta os seguintes aspectos:

- Identificar os objetos e as classes existentes no sistema e seus relacionamentos;
- Definir o comportamento esperado do sistema em termos de eventos de entrada e saída;
- Os modelos do sistema são produzidos descrevendo os seguintes elementos:
  - Classes de objetos que existem no sistema;
  - Relacionamentos entre essas classes;
  - Operações que podem ser executadas no sistema;
  - Sequências permissíveis para eventos de entrada e de saída;

Na fase de projeto o objetivo é definir como serão implementadas as operações do sistema através do comportamento dos objetos em tempo de execução. Durante a

fase de projeto, as operações são associadas às classes e os modelos gerados documentam o que segue:

- Operações são divididas em interações de objetos;
- Objetos fazem referências mútuas entre si;
- As operações são implementadas pelas interações entre os objetos;
- As classes se relacionam por herança;
- Os atributos constam nas classes juntamente com os métodos;

O método Fusion tem ao seu final uma fase de implementação. O autor propõe o uso de máquinas de estados, complementando a fase de projeto, mas tem como meta transformar o projeto em código-fonte orientado a objetos. Segue abaixo as orientações do Fusion para esta transformação do projeto em código:

- Herança, referências e atributos das classes são codificados nas classes específicas;
- Interações entre objetos são transformadas nos métodos pertencentes a uma classe selecionada;
- Máquinas de estado são usadas para se reconhecer as sequências permitidas para as operações;
- Também são consideradas técnicas de inspeção e teste para avaliação de sistemas orientados a objetos.

Dicionário de Dados: caracteriza-se como um artefato de software que é adotado e "alimentado" durante todo o processo de construção do software e identifica as entidades existentes no sistema. Este artefato serve de referência durante todo o processo de construção do software.

1.3. Padrões de Modelagem de Objetos usando UML - Unified Modeling Language

A UML teve seu desenvolvimento iniciado em outubro de 1994 quando Grady Booch e Jim Rumbaugh, da Rational Software Corporation, começaram seus trabalhos de unificar os métodos Booch e OMT. Em 1995 juntou-se com aqueles Ivar Jacobson, unindo sua companhia à Rational e sua proposta de Engenharia de Software Orientada a Objetos (OOSE), através do processo *Objectory 3.8*, passou a colaborar com a proposta da UML.

Booch, Rumbaugh e Jacobson, três autores de renome internacional, conhecidos e referenciados mundialmente como "The Three Amigos" (isso, assim mesmo), extraíram de suas metodologias anteriores aspectos positivos de cada uma delas de maneira a formular a UML. As experiências destes profissionais mostraram ser ineficiente uma metodologia estanque e que não se adequasse a outros processos de desenvolvimento

de software. O que os motivou à criação da UML como uma linguagem flexível, que pudesse se adaptar a outras metodologias já existentes ou que por ventura viessem a ser criadas.

A partir deste quadro mundial, havia sido criada uma comissão internacional, denominada *Object Management Group* (OMG) (OMG, 2004), que iniciou um processo de padronização da área de orientação a objetos. Assim, a OMG reconheceu a UML 1.3 com exclusividade, em 1996, como padrão de linguagem notacional para construção de artefatos de sistemas orientados a objetos. Atualmente a UML encontra-se na versão 1.5, estando em processo constante de avaliação e ajustes junto à OMG para evoluções em sua proposta. Em 2001, membros da OMG iniciaram o trabalho de uma evolução para a UML 2.0, o qual é mantido organizado por quatro RFPs (*Requests for Proposal*) sendo: *UML Infrastructure*, *UML Superstructure*, *Object Constraint Language*, e *UML Diagram Interchange*.

Partindo para a sua definição, a UML "é uma linguagem gráfica para visualizar, especificar, construir e documentar os artefatos de um sistema de software. A UML oferece uma forma padrão para se escrever/modelar projetos de sistemas, incluindo conceitos, tais como processos de negócios e funções de sistema, bem como 'coisas' concretas como sentenças de linguagem de programação, projetos de bancos de dados e componentes reutilizáveis de software" (Booch, 1998).

Em termos de estrutura, para este trabalho é suficiente, por enquanto, sabermos que a UML define doze tipos de diagramas, divididos em três categorias: quatro tipos de diagramas representam a estrutura estática de uma aplicação; cinco outros diagramas representam diferentes aspectos do comportamento dinâmico da aplicação; e outros três representam maneiras como você pode organizar e gerenciar os módulos da aplicação (OMG, 2004).

- Diagramas Estruturais:
    - Diagramas de Classes;
    - Diagramas de Objetos;
    - Diagramas de Componentes;
    - Diagramas de Implantação / Disposição Física (*Deployment Diagram*);
- Diagramas Comportamentais:
    - Diagramas *Use Case*;
    - Diagramas de Sequência;
    - Diagramas de Atividade;

    - Diagramas de Colaboração;
    - Diagramas de Estado;
- Diagramas Gerenciais:
    - Pacotes;
    - Subsistemas;
    - Modelos;

Além dos diagramas, a UML viabiliza outros mecanismos gerais tais como notas, contratos, etc. Contudo não iremos abordar cada pormenor da UML, pois contém um conjunto de notações relativamente extenso. No restante dos capítulos serão apresentados os aspectos diagramáticos necessários aos nossos trabalhos, já que não é necessária a utilização de todas as notações e diagramas da UML para a customização de um processo ou método de desenvolvimento de software.

A UML, portanto, não se caracteriza como um método ou processo de construção de software e sim uma linguagem comum para expressar os diversos modelos, ou seja, não informa "como se faz" para desenvolver software. Conseqüentemente, diversos métodos e processos de desenvolvimento efetivo de software continuarão a ser criados, porém poderão ser feitos usando a UML como uma linguagem comum. De pronto podem ser citados o RUP, de Jacobson (1998) e a proposta, bastante didática, de Larman (2000). Esta realidade também vem a motivar que empresas de desenvolvimento, que adotam métodos e processos proprietários/particulares para construção de software, possam se adequar a um padrão mundial de orientação a objetos de maneira customizada aos seus ambientes de desenvolvimento.

A flexibilidade da UML e sua padronização justificam sua utilização em nosso trabalho. A partir da próxima seção começaremos a abordar o processo que utilizaremos para construir software e quais serão os diagramas e notações UML que faremos uso para tornar a modelagem dos artefatos do software inteligíveis.

1.4. Método Fusion Expandido e Adaptado à UML

A partir desta seção é abordado o tema principal deste trabalho, o qual tem como alicerce os resultados obtidos com o projeto FILM – forma simbólica de referenciar o projeto denominado "Método Fusion Expandido e Adaptado à UML"–, tendo sido institucionalizado na Universidade de Caxias do Sul, dentro do Departamento de Informática, com apoio da Pró-Reitoria de Pesquisa e Pós-Graduação e pela FAPERGS - Fundo de Amparo à Pesquisa do Estado do Rio Grande do Sul (Galimberti, 1999). O Projeto FILM, oficialmente iniciado no ano 2000, tinha a proposta original de adaptar o método Fusion de análise e projeto de sistemas orientados a objetos com uma

linguagem padrão de modelagem, sendo que a proposta do projeto e as primeiras atividades desenvolvidas tiveram resultados preliminares ainda em 1999, com o trabalho acadêmico de Migot (1999). A importância da reformulação do método se concentrava na eficiência e clareza dos modelos gerados por aquele método, sendo que os mesmos podiam ser considerados, às vezes, como incompletos e com notações pouco especificadas para o trabalho didático em cursos de graduação. A proposta também expandia o espectro do método Fusion, acrescentando ao mesmo a abordagem da fase inicial de especificação de requisitos, a qual não era considerada na versão original do método, e desenvolvida em sua primeira versão pelo acadêmico Franzen (1999).

Para a expansão do método e adequação de seus modelos, utilizou-se da *Unified Modeling Language* (UML). Conforme já visto, a UML é considerada padrão mundial para modelagem de sistemas pelo Object Management Group (OMG), e propõe flexibilidade aos diversos processos de desenvolvimento de software, possibilitando expansão dos métodos orientados a objetos e disponibilizando um conjunto de notações para a construção dos modelos da orientação a objetos.

Este contexto, aliado ao fato de existir proposta semelhante, quando da adequação da UML ao processo Objectory de Engenharia de Software, através do trabalho "*UML Extension for Objectory Process for Software Engineering*", veio a motivar o presente trabalho, com o qual buscou-se utilizar a UML para acrescentar clareza, eficiência e padronização à construção dos artefatos propostos pelo método Fusion. Além disso, viabilizou-se a expansão do método Fusion em relação à fase inicial de definição de requisitos, enquadrando-se notações da UML para a geração de artefatos para esta fase.

No entanto, a principal motivação para a criação deste trabalho vem da atual estrutura curricular dos cursos do Departamento de Informática desta universidade e, consequentemente, da inserção de seus egressos no mercado de desenvolvimento de software. Sendo estes cursos formadores de profissionais empreendedores com perfis de analistas e projetistas de software, considera-se que os mesmos devem estar sintonizados com o contexto mundial da tecnologia e dos padrões de orientação a objetos. Deve-se, portanto, considerar o processo gradativo de mudança do paradigma estruturado pelo orientado a objetos, pelo qual passam as empresas do setor de informática da região, o que as está levando a adotar padrões estabelecidos para o segmento de produção de software.

Assim sendo, para que a adaptação do método fosse calibrada e concluída, foram desenvolvidos muitos estudos de caso, em várias frentes acadêmicas e outros projetos. Vale citar os mais de 40 trabalhos de estágio, no curso de Tecnologia em Processamento de Dados, que utilizaram dos resultados deste projeto para guiarem o desenvolvimento de seus software.

O Projeto FILM também foi validado com a construção de uma ferramenta de software de cálculo de motores elétricos, em parceria com a empresa Eberle e financiamento da FAPERGS, através do projeto Ferramenta de Cálculo de Motores Elétricos de Indução Trifásicos e Monofásicos (ProMot). Este projeto permitiu o amadurecimento do método FILM, originando estudos para acadêmicos de graduação em Ciência da Computação sobre técnicas de inspeção de artefatos de software (Zanotto, 2001) e no campo de mapeamento de modelos de objetos para camada de armazenamento de dados (Mattiuz, 2002). Contudo, a contribuição maior ainda estava por vir, que foi a evolução de um método para um processo mais completo de produção de software. Definiu-se, então, um processo misto de desenvolvimento de software, sendo composto, em um primeiro estágio, por um Processo Linear e, em seguida, por um Processo Cíclico (mais especificamente, um Processo Incremental-Iterativo).

Outra agradável surpresa com o projeto FILM foi o início do desenvolvimento de um interessante estudo de caso que se caracterizou em uma experiência de excelentes resultados, sendo que não estava previsto quando da proposta deste projeto e foi idealizado com o desenrolar das pesquisas. Trata-se de módulos de uma ferramenta de software de auxílio à engenharia de software, comumente conhecida como CASE (*Computer Aided Software Engineering*), e que também gerou os trabalhos dos acadêmicos Lazzari (2001) e Vieira (2002). Esta ferramenta, chamada CASE FILM, foi construída conforme as orientações do método FILM e se propõe a auxiliar os profissionais e estudantes da área de engenharia de software a conduzirem seus projetos de software conforme o método defendido por esta pesquisa.

Ressalta-se, ainda, o objetivo de viabilizar ao profissional habilidades suficientes para que possa optar por outros processos de construção de software. Isto poderia ser feito a qualquer momento, usando-se, por exemplo, o processo RUP (*Rational Unified Process*), o qual está bastante difundido mundialmente. No entanto, sob o ponto de vista didático, justifica-se a adoção de mais de uma abordagem de APOO, no nosso caso o Fusion, a UML, a Definição de Requisitos por Objetivos e, mais recentemente, adaptação de práticas de modelagem ágil, pois assim o acadêmico e profissional de engenharia de software poderá visualizar as vantagens em se conhecer como adaptar um determinado método à realidade de construção de software nas empresas em que irão atuar.

Numa relação de troca permanente, este projeto está auxiliando o ensino da disciplina de engenharia de software e, em específico, guiando as disciplinas de análise e projeto de sistemas através do Guia FILM de Análise e Projeto de Sistemas Orientados a Objetos. Por outro lado, a aplicação do "novo" método como conteúdo curricular está ajudando na validação e adequação progressiva do mesmo. Este objetivo se completa ao situar o egresso dos cursos supra citados no cenário atual de desenvolvimento de software, viabilizando ao mesmo a construção de conhecimento atualizado e sintonizado com os padrões internacionais já estabelecidos, como é o caso da UML que está sendo adotada há muito pela indústria de informática mundial e mais

recentemente na região de Caxias do Sul.

Para a fase de Definição e Especificação de Requisitos o projeto colaborou na construção da proposta "Um Modelo de Estruturação de Requisitos para o Método Fusion" (Rocco, 2001), sendo seus resultados utilizados pelo FILM na fase inicial do processo. No entanto, dependendo dos objetivos e carga horária de um curso de APOO, é possível partir de especificações textuais e, na Análise e Projeto Orientados a Objetos, preferencialmente, adotar *use cases* para a definição de requisitos.

A estrutura geral dos resultados alcançados com o projeto FILM será apresentada com o que há em termos de processos e fases de construção de sistemas de software orientados a objetos, conforme podemos analisar na figura 1.2. Em seguida, dentro de cada capítulo, cada fase será tratada de forma separada e expandida, permitindo uma visualização detalhada do que deve ser modelado em cada uma de suas atividades internas, o que será expresso a partir de diagramas de atividades.

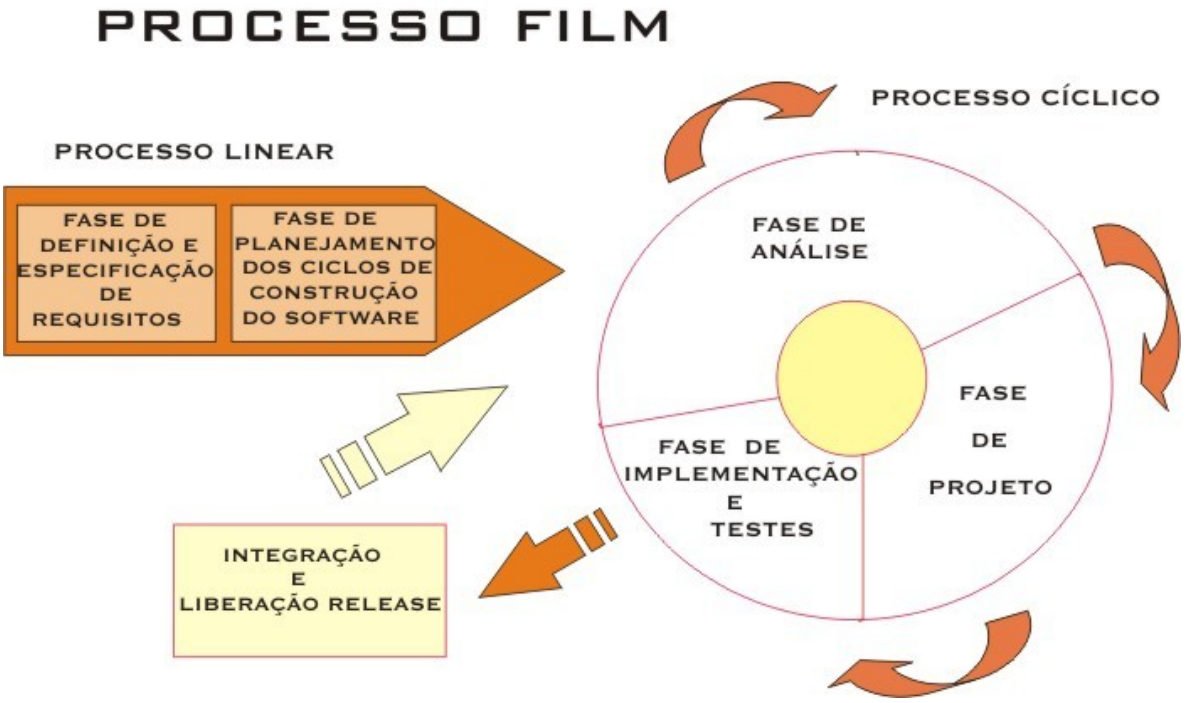

Figura 1.2 – Modelo do Processo FILM para Construção de Software

O modelo acima permite visualizar a estrutura geral do FILM através dos seus processos e de suas fases. O FILM prevê um processo misto para tratamento das fases do método, sendo inicialmente conduzido por um Processo Linear e, em seguida, por várias iterações em um Processo Cíclico (Incremental-Iterativo).

O Processo Linear, ao ter como meta a estruturação dos objetivos de negócio e os requisitos do sistema, preparando o início da construção iterativa e incremental, está

composto por duas fases, sendo: fase de Definição e Especificação de Requisitos e a fase de Planejamento dos Ciclos de Construção do Software.

Já o Processo Cíclico caracteriza-se pela divisão do projeto em construção em unidades menores, mas com propriedades de versões operacionais, as quais podem ser liberadas aos clientes/usuários. Uma iteração é como um miniprojeto que resulta em uma versão operacional incremental (e, espera-se, melhorada) do sistema em relação a iteração anterior. Cada iteração, em geral, deve percorrer todas as fases do Processo Cíclico, sendo as fases de Análise, Projeto, Implementação e Testes. Vale enfatizar que o FILM tem como foco as fase de Análise e Projeto, buscando estruturar e orientar o prosseguimento para as fases de Implementação e Testes.

Este processos e fases serão devidamente conceituados ao longo do trabalho. Nos dois próximos capítulos serão tratadas as duas primeiras fases do Processo Linear. A partir do capítulo 4 inicia-se o Processo Cíclico, sendo dedicado, também, um capítulo para cada uma de suas fases.

A organização dos demais capítulos compreende o detalhamento das fases e suas atividades no sentido de facilitar ao leitor sua compreensão e aplicação. Serão apresentados os refinamentos das fases do processo de desenvovimento de software, sendo fornecidos fluxos de atividades, descrição de artefatos de entrada e saída e orientações estratégicas para a construção dos artefatos de software. O Processo FILM de Construção de Software será apresentado, portanto, na seqüência de seus processos Linear e Cíclico, onde para cada uma das fases e atividades serão tratados os seguintes tópicos:

- Fundamentação conceitual da fase do processo: é feita uma introdução que oferece uma visão do que está incluído na fase, seus principais objetivos e correlação com o fluxo de atividades da fase (*workflow*).
- *Workflow* da fase: visualização da seqüência de realização das atividades da fase.
- Cada atividade apresentada no fluxo de atividades será detalhada conforme os itens a seguir:
    - Fundamentação conceitual da atividade e descrição dos objetivos da mesma;
    - Descrição dos trabalhadores envolvidos na atividade;
    - Descrição dos artefatos de entrada necessários à atividade;
    - Descrição dos artefatos produzidos durante a atividade, ou seja, os artefato de saída da atividade;

- Estratégias e Notações para Modelagem: este tópico tem o propósito de orientar o leitor para a realização da atividade e modelagem de artefatos, além de especificar a estrutura notacional a ser utilizada na atividade;

- Estudo de caso: buscando facilitar o aprendizado, foram trabalhados em sala de aula diversos estudos de casos desenvolvidos no Processo FILM. Contudo, verificou-se que poderiam haver benefícios didáticos em se ter um trabalho mais simples e compacto para sala de aula. Então foi iniciada a especificação de um sistema para controle de uma biblioteca virtual departamental, comumente sendo chamada MiniBib neste trabalho. Em seguida este tema foi oferecido como sugestão para trabalho de estágio para auxiliar o projeto FILM. Esta proposta foi continuada e disponibilizada por este projeto através do acadêmico DeBona (2002) com o intuito de oferecer um material menos complexo e mais aplicável pelos estudantes de análise e projeto de sistemas. Ao longo das atividades deste guia serão selecionados pequenos trechos do estudo de caso, adaptando-se o mesmo, em sua totalidade (exceto a própria implementação por não ser o escopo deste guia), no apêndice deste trabalho.

- Inspeção dos artefatos da fase: este tópico é apresentado ao final de cada capítulo e tem como objetivo prover qualidade final ao produto de software através do tratamento da qualidade parcial de cada um dos artefatos desenvolvidos ao longo das fases. São prestadas orientações aos trabalhadores da fase no sentido de dar subsídios técnicos para a verificação e inspeção dos artefatos desenvolvidos em cada atividade. Optou-se por abordar este tema ao final dos capítulos, mas as orientações tratarão de todas as atividades da fase, o que deverá fazer com que os revisores sugiram mudanças nos artefatos da respectiva fase e do ciclo em questão. A necessidade por técnicas de inspeção adaptadas ao FILM também originou o trabalho do acadêmico Zanotto (2001), o qual permitiu os primeiros debates em torno de um módulo de inspeção para a ferramenta CASE FILM.