UNIVERSIDADE FEDERAL DE ITAJUBÁ

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

SISTEMA DE SOFTWARE PARA AUXÍLIO DOS DOCENTES DA UNIFEI

THALES MARTINS FERRARI

ITAJUBÁ, 2006

Universidade Federal de Itajubá
Departamento de Engenharia Elétrica

Thales Martins Ferrari

Sistema de Software para auxílio dos docentes da Unifei

Dissertação submetida ao Programa de pós-graduação em Engenharia Elétrica como requisito parcial à obtenção do título de Mestre em Engenharia Elétrica

Orientador: Prof. Dr. Otávio Augusto S. Carpinteiro

Itajubá, 2006.

Ferrari, Thales Martins. Sistema de Software para auxílio dos docentes da Unifei. Itajubá: UNIFEI 2006. 123 páginas (Dissertação de mestrado submetida ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Itajubá).

Palavras-Chave: Engenharia de Software, UML, RUP, XML, Visual Basic, Currículo Lattes

Ficha catalográfica elaborada pela Biblioteca Mauá –
Bibliotecária Margareth Ribeiro- CRB_6/1700

F375s

Ferrari, Thales Martins
Sistema de software para auxílio dos docentes UNIFEI / Thales Martins Ferrari. Itajubá (MG) : [s.n.], 2006.
138 p. : il.

Orientador : Prof. PhD. Otávio Augusto Salgado Carpinteiro.
Dissertação (Mestrado) – Universidade Federal de Itajubá.

1. Engenharia de software. 2. UML 2.0. 3. RUP. 4. XML. I. Carpinteiro, Otávio Augusto Salgado, orient. II. Universidade Federal de Itajubá. III. Título.

CDU 004.41(043)

Universidade Federal de Itajubá
Departamento de Engenharia Elétrica

Thales Martins Ferrari

Sistema de Software para auxílio dos docentes da Unifei

Dissertação submetida ao Programa de pós-graduação em Engenharia Elétrica como requisito parcial à obtenção do título de Mestre em Engenharia Elétrica

Banca Examinadora:
Prof. Dr. Otávio Augusto S. Carpinteiro (Orientador)
Prof. Dr. Fábio Protti
Prof. Dr. Edmilson Marmo Moreira

Itajubá, 2006

DEDICATÓRIA

Não apenas esse projeto, mas todas as minhas conquistas profissionais e pessoais são dedicadas aos meus pais, Cassia e Luiz Henrique, que em nenhum momento deixaram de me apoiar e a minhas irmãs e meu irmão (Amo vocês!!).

Aos meus amigos, se é que os tenho, pois para mim são irmãos incondicionais. Não necessitam de nomes citados, pois eles sabem o que são para mim e quem são.

A minha futura esposa, uma pessoa linda que sempre esteve ao meu lado.

AGRADECIMENTO

Ao orientador Prof. Otávio Augusto S. Carpinteiro, pela dedicação incondicional, apoio, paciência e também por ter acreditado em mim.

A todos que direta ou indiretamente contribuíram para a realização desta pesquisa.

A Deus.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

API	Application Programming Interface
CCI TT	Órgão internacional de padronização em telecomunicações
CNPq	Conselho Nacional de Pesquisas.
COM	Component Object Model
CPPD	Comissão Permanente de Pessoal Docente
DOM	Document Object Model
FDD	Feature-Driven Development
GED	Gratificação de Estimulo a Docência
IDE	Integrated Development Environment
MSMQ	Microsoft Message Queue
MTS	Microsoft Transaction Server
OMG	grupo de gerência de objetos
RTF	Rich Text Format
RUP	Processo Unificado da Rational
SAX	Simple API for XML
SGBD	Sistema Gerenciador de Banco de Dados
TSql	Transact Structured Query Language
UML	Unified Modeling Language
Unifei	Universidade Federal de Itajubá
VB	Visual Basic
VBA	Visual Basic for Applications
W3C	World Wide Web Consortium
XML	eXtensible Markup Language
XP	eXtreme Programming
XSD	XML Schema Definition

RESUMO

Este trabalho tem como objetivo o estudo dos modelos, processos e tecnologias da Engenharia de software para o desenvolvimento de um sistema de software que facilite a criação de relatórios de atividades dos docentes da Universidade Federal de Itajubá - Unifei.

Os docentes desta instituição já contam com um cadastro de todas as suas atividades acadêmicas no software Currículo Lattes do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq). Porém, tal software emite o currículo completo em apenas uma formatação.

Na necessidade de um currículo específico que exiba apenas as atividades acadêmicas de uma determinada Instituição, período e de outro formato, o docente teria de copiar todos os dados deste sistema ou de outra fonte qualquer. Deste modo, elaboraria o relatório manualmente, filtrando, complementando e formatando outra vez todos os dados.

Para a progressão em sua carreira, a instituição exige relatórios periódicos com formatação específica das atividades realizadas. Isto demanda tempo e esforço, tempo este que poderia estar sendo empregado em outras atividades, beneficiando não só o docente como a instituição, além de evitar a duplicação de informações, que leva a muitos erros. A coerência dos dados é de suma importância.

O sistema de software proposto foi desenvolvido para agilizar e facilitar a emissão de relatórios de atividades realizadas pelos docentes da Unifei, buscando os dados já contidos no Currículo Lattes. Este outro banco de dados auxiliar completará as informações já existentes, facilitando a adequação dos relatórios aos padrões exigidos. Informações como matrícula dos alunos, códigos das disciplinas, entre outras, estarão ali presentes.

Com a junção destes dois bancos de dados, será possível a geração automática de relatórios que poderão ser editados e impressos, satisfazendo as necessidades dos docentes e da Instituição.

ABSTRACT

This work aims at studying of the models, processes and technologies of Software Engineering for the development of a software system that facilitates the creation of reports of activities of the professors of the Federal University of Itajubá - Unifei.

The professors of this institution already count on one register in cadastre of all its academic activities in software "Currículo Lattes" of CNPq (The National Council for Scientific and Technological Development). However, such software emits the complete resume in only one formatting.

In the necessity of a specific resume that shows only the academic activities of one determined Institution, period and of another format, the professor would have to copy all the data of this system or another source any. In this way, the report would elaborate manually, filtering, complementing and formatting another time all the data.

For the progression in its career, the institution demands periodic reports with specific formatting of the carried through activities. This demand time and effort, time this that could be being employed in other activities, benefiting not only the professor as the institution, beyond preventing the duplication of information, that leads to many errors.

The data's coherence is of utmost importance.

The system of considered software was developed to speed and to facilitate to the emission of reports of activities carried through for the professors of the Unifei, searching the contained data already in the "Currículo Lattes". This another data base auxiliary will complete the existing information already, facilitating the adequacy of the reports to the demanded standards. Information as school registration of the pupils, codes of your discipline them, among others, will be gifts there.

With the junction of these two data bases, the automatic generation of reports that could be edited and printed matters will be possible, satisfying the necessities of the professors and the Institution.

1. INTRODUÇÃO

Nos dias atuais, o retrabalho e a redundância de informações são características que ocorrem, com freqüência, nos processos administrativos das Instituições. Tais características podem ocasionar vários problemas – dados inconsistentes e redundantes, duplicação de esforços – principalmente quando se gerencia informações, sejam elas digitais ou não. Estes problemas podem acarretar perda de tempo e de produção, tempo esse que poderia estar sendo utilizado na realização de outras tarefas.

Para progressão na carreira, o docente deve apresentar relatórios periódicos de sua produção e atividades. Os dados destes relatórios, em sua maioria, estão cadastrados na base de dados do Currículo Lattes do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq). O sistema Lattes gerencia informações de docentes em âmbito nacional. Seu preenchimento é praticamente obrigatório para os docentes. A base de dados do Currículo Lattes contém, portanto, a maioria das informações necessárias nos relatórios exigidos pela Unifei. O retrabalho com dados já cadastrados no Currículo Lattes pode então causar redundância de informações, dificuldades no redigitar as mesmas e perda excessiva de tempo, podendo causar erros ou até mesmo a falta de informações. Por exemplo, esquecer ou errar a data de um artigo publicado há alguns anos pode ocorrer facilmente.

É importante, para um docente, a ascensão na carreira, para se tornar um profissional mais qualificado e reconhecido pelos seus trabalhos. Para tal, é exigência da Instituição, na qual ele trabalha, a apresentação de relatórios que comprovem o merecimento de tal progressão. Estes são elaborados dentro de um certo período, na maioria das vezes em alguns anos, visando alcançar um certo número de produção e atividades que comprovem o merecimento de sua evolução. Ter que elaborar tal relatório com informações de atividades que ocorreram no passado pode ser de extrema dificuldade, pois as informações podem ser perdidas, esquecidas e duplicadas, tendo em vista que o docente terá muitas vezes que buscá-las em vários lugares ou relembrá-las. O tempo perdido para realização desses relatórios manualmente é grande, o que ocasiona perdas para o docente e para a Instituição, porque neste período poderia estar realizando outras atividades acadêmicas ou científicas, que trariam, com certeza, benefícios para ele e para a

Instituição. A Instituição também precisa controlar as atividades e produção de seus docentes, e ter relatórios precisos e formatados. Dados consistentes e organizados são de grande importância, pois, além de facilitarem o trabalho, evitam um grande esforço na análise dos mesmos.

Para resolver este problema, propomos um sistema de software que busque na base de dados do Currículo Lattes todas as informações que os relatórios de progressão precisam e os preencha com tais dados automaticamente. Teríamos então uma tarefa bem menor para entrega dos mesmos e com menos chances de erros. O sistema de software, através do padrão XML (eXtensible Markup Language) – um padrão internacional de fácil utilização e que propicia dados organizados hierarquicamente –, irá ler os dados gerados pelo Currículo Lattes e montar os relatórios para os docentes já na formatação exigida pela Instituição. Os relatórios podem ser abertos em um editor de texto comum, contido na máquina do docente, para posteriores alterações, ajustes finos e armazenamento em mídia digital ou impressão. Para possibilitar uma maior integração, utilizaremos o padrão RTF (Rich Text Format) de formatação de arquivos, um padrão internacional de fácil manipulação e aceito pela maioria dos editores de texto no mercado atualmente.

O sistema de software possui uma base de dados para complementar as informações que não estão contidas na base do Currículo Lattes. Realizará, então, uma integração dos dados contidos no arquivo XML gerado pelo Currículo Lattes, com a base de dados interna. Um exemplo de informação que está na base de dados interna e não está na base do Currículo Lattes é o número de matrícula dos alunos, que é solicitado na parte dos relatórios referentes às orientações do docente. A base de dados interna armazena somente os dados que não estão contidos na base do Currículo Lattes e alguns dados para integração com esta base.

1.1 Descrição dos capítulos

Este trabalho é composto de seis capítulos, incluindo esta introdução e anexos.

No segundo capítulo apresentamos alguns modelos e processos de Engenharia de Software utilizados no desenvolvimento do trabalho. No terceiro capítulo apresentamos os padrões XML, RTF e a Linguagem de programação Visual Basic. No quarto capítulo descrevemos o sistema desenvolvido e os resultados

obtidos com o mesmo. No quinto capítulo apresentamos uma conclusão para o trabalho. Os anexos apresentam documentos relevantes, como os modelos de relatórios e manual do software.

2. ENGENHARIA DE SOFTWARE

A Engenharia de Software é a área da Engenharia que oferece suporte em todas as fases de desenvolvimento de um sistema de software. Ela está presente na parte técnica, política e até financeira das etapas de construção e manutenção, que um sistema de software pode ter.

Não existe uma abordagem, em particular, que seja melhor para a solução do processo de produção de um sistema de software. Pressman [1995, p.30] diz que:

> *Entretanto, ao combinarmos métodos abrangentes para todas as fases de desenvolvimento do software, melhores ferramentas para automatizar esses métodos, blocos de construção mais poderosos para a implementação do software, melhores técnicas para a garantia da qualidade do software e uma filosofia de coordenação predominante, controle e administração, podemos conseguir uma disciplina para o desenvolvimento de software - disciplina esta chamada de engenharia de software.*

Segundo Sommerville [2003], Engenharia de software é uma disciplina da engenharia que se ocupa de todos os aspectos da produção de software, desde os estágios iniciais de especificação do sistema até a manutenção desse sistema, depois que ele entrou em operação. A Engenharia de software não se dedica só aos processos técnicos de desenvolvimento, mas também a atividades como o gerenciamento de projetos e o desenvolvimento de ferramentas, métodos e teorias que dêem apoio à produção de sistemas de software.

O desenvolvimento de um sistema de software pode ser realizado por diversos métodos da Engenharia de Software. Para a equipe de desenvolvimento, cabe a tarefa de escolher dentre esses métodos qual ou quais se encaixam melhor ao seu projeto e aplicá-los da melhor forma possível, seguindo os padrões da própria Engenharia de Software.

A Engenharia de Software é composta de processos e modelos de desenvolvimento, que juntos possibilitam a equipe de desenvolvimento uma base mais sólida para a construção de um sistema de software de alta qualidade. Os modelos ou paradigmas de desenvolvimento de sistemas de software são representações do seu processo de criação. Os processos são os procedimentos ou metodologias utilizadas para o desenvolvimento. Compreendem a especificação, o

desenvolvimento, a validação e a evolução do sistema de software. Um modelo de desenvolvimento é a UML (Unified Modeling Language), que oferece várias representações gráficas de um sistema de software. Tais itens citados acima, são descritos a seguir neste capítulo.

2.1 Conceitos gerais sobre engenharia de software

Um sistema de software é a implementação computacional de um modelo. Para realizar um projeto de desenvolvimento de um sistema de software é necessária uma metodologia, um conjunto de regras e padrões que devem auxiliar todo o ciclo de desenvolvimento do mesmo. Existem várias técnicas de modelagem e análise de sistema, uma das mais utilizadas, hoje em dia, é a orientada a objetos.

Um sistema de software em seu desenvolvimento passa por vários processos distintos que devem ser bem definidos e executados. Para que tudo corra bem, segundo Sommerville [2003], esses processos podem ser divididos em quatro partes sendo elas:

1. Especificação do software: A funcionalidade do sistema de software e as restrições em sua operação devem ser definidas.
2. Desenvolvimento do software: O sistema de software deve ser produzido de modo que atenda a suas especificações.
3. Validação do software: O sistema de software tem que ser validado para garantir que ele faz o que o cliente deseja.
4. Evolução do software: O sistema de software deve evoluir para atender às necessidades mutáveis do cliente.

Yang [2004], diz que: hoje em dia, software é usado na área de saúde, segurança social, defesa, entre outras. O desenvolvimento de tais softwares parte de modelar aspectos, fatos, dados e informações do mundo real. O modelo é transformado então em uma série de modelos refinados. Tal refinamento alcança eventualmente um código executável. Como o mundo real se mantém em mudança, um sistema de software necessita ser mantido constantemente evoluído, para suprir as exigências das mudanças. Desenvolver e manter tal sistema de software em evolução é obviamente difícil. Um processo bem-disciplinado de desenvolvimento e uma notação boa de modelagem são essenciais para controlar as atividades de desenvolvimento e documentar os modelos.

Quando os clientes apresentarem as idéias que necessitam de soluções do sistema, os desenvolvedores têm uma obrigação ética e profissional de ajudar o cliente a definir seu problema. Deve-se construir a melhor solução ao problema do cliente, mesmo que o cliente não compreenda ainda como ou o que pedir. O cliente deve ser incentivado a escrever um prospecto curto que indique a finalidade do sistema, seu valor, e alguns refinamentos essenciais para torná-lo útil. Este prospecto não deve ser confundido com um jogo completo dos requisitos, o qual emergirá somente com um processo iterativo posterior (Bernstein [2005]).

2.1.1 Processos de desenvolvimento de Software

No desenvolvimento de um sistema de software, um conjunto de etapas deve ser definido, sendo denominados paradigmas da engenharia de software, também conhecido como ciclo de vida de software (PRESSMAN [1995], PAULA [2000], Thiry [2001]).

A norma NBR-12207(1998) cita que o ciclo de vida de um sistema de software é uma estrutura contendo processos, atividades e tarefas envolvidas no desenvolvimento, operação e manutenção de um produto de software, abrangendo a vida do sistema desde a definição de seus requisitos até o término de seu uso.

A produção de um sistema de software é uma tarefa complexa que pode ocorrer de várias maneiras diferentes. Para isso, existem vários tipos de processos[1] para a construção de um sistema de software com qualidade. Dentre eles, destacamos cascata, prototipação, incremental, espiral, RUP, XP, FDD e Scrum.

2.1.2 Cascata

O processo de desenvolvimento em cascata, paradigma clássico da Engenharia de Software segundo Leach [2000], é o processo mais comum de desenvolvimento de sistemas de software. Possui fases separadas e distintas de especificação e de desenvolvimento (Sommerville [2004]). Tem as etapas descritas a seguir:

1 - Alguns autores referem-se a "modelos de processos". Preferimos, no entanto, usar "tipos de processos", evitando a palavra "modelo", pois pode ser confundida com "modelo do sistema".

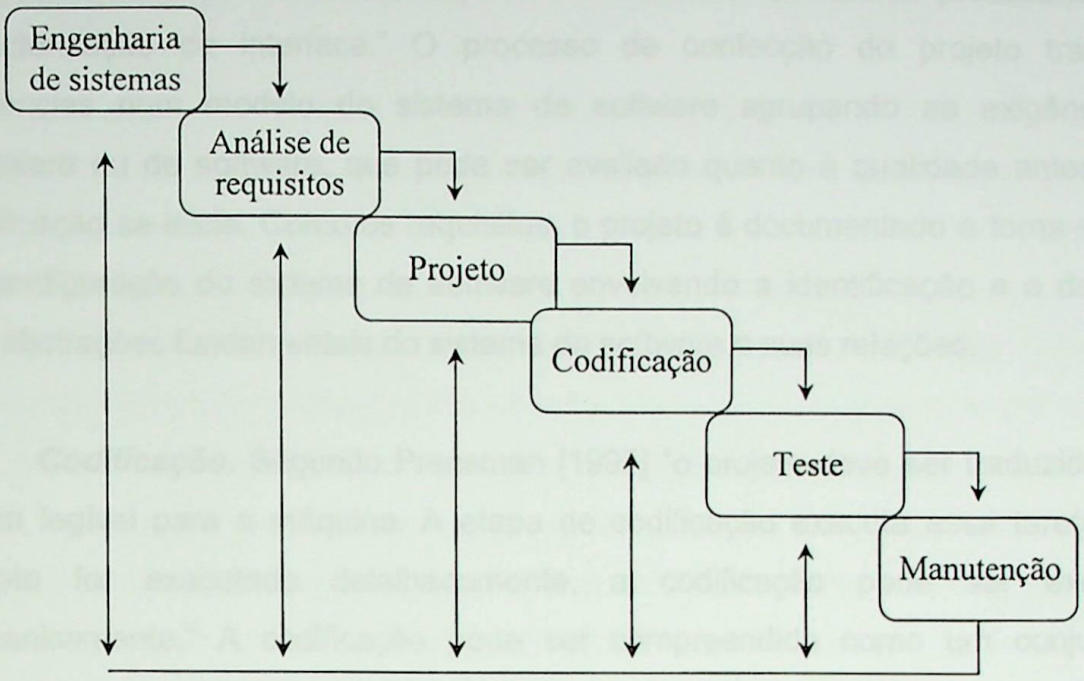

Figura 1. O modelo em cascata.

Engenharia de sistemas. Segundo Pressman [1995, p.34], "uma vez que o sistema de software sempre faz parte de um sistema mais amplo, o trabalho inicia-se com o estabelecimento dos requisitos para todos os elementos do sistema e prossegue com a atribuição de certo subconjunto desses requisitos ao software. Essa visão do sistema é essencial quando o sistema de software deve fazer interface com outros elementos, tais como hardware, pessoas e banco de dados. A análise e engenharia de sistemas envolvem a coleta dos requisitos em nível do sistema, com uma pequena quantidade de projeto e análise de alto nível".

Análise de requisitos de software. Nesta etapa, segundo Pressman [1995], o processo de coleta dos requisitos é intensificado e concentrado especificamente no sistema de software, visando um produto final com qualidade e que atenda a todas as especificações do cliente ("usuário"). Para entender a natureza do(s) programa(s) a ser(em) construído(s), o engenheiro ("analista") de um sistema de software deve compreender o domínio da informação para o software, bem como a função, desempenho e interface exigidos. As funções, as restrições e os objetivos para o sistema são documentados e revistos com o cliente.

Projeto. Segundo Pressman [1995], "o projeto de software é, de fato, um processo de múltiplos passos que se concentra em quatro atributos distintos do

programa: estrutura de dados, arquitetura de software, detalhes procedimentais e caracterização de interface." O processo de confecção do projeto traduz as exigências num modelo do sistema de software agrupando as exigências de hardware ou de software, que pode ser avaliado quanto à qualidade antes que a codificação se inicie. Como os requisitos, o projeto é documentado e torna-se parte da configuração do sistema de software envolvendo a identificação e a descrição das abstrações fundamentais do sistema de software e suas relações.

Codificação. Segundo Pressman [1995] "o projeto deve ser traduzido numa forma legível para a máquina. A etapa de codificação executa essa tarefa. Se o projeto for executado detalhadamente, a codificação pode ser executada mecanicamente." A codificação pode ser compreendida como um conjunto de programas ou unidades de programas sendo que cada uma dessas partes podem ser codificadas e testadas separadamente, porém visando um todo que é o código total do sistema de software atendendo a todos os requisitos.

Testes. Segundo Pressman [1995], assim que o código for gerado, iniciar-se-á a realização de testes do programa e testes de integração caso o programa tenha sido codificado em partes. O processo de realização de testes concentra-se nos aspectos lógicos internos do sistema de software, garantindo que todas as instruções tenham sido testadas e integradas corretamente, e concentram-se também nos aspectos funcionais externos, ou seja, realizando testes para descobrir erros e garantir que todos os requisitos foram atendidos. Depois de testado o sistema de software é entregue ao cliente.

Manutenção. Segundo Pressman [1995], normalmente, o sistema de software sofrerá mudanças depois que for entregue ao cliente, fazendo dessa fase uma das mais longas do ciclo de vida. Ocorrerão mudanças porque erros foram encontrados, porque o sistema de software deve ser adaptado a fim de acomodar mudanças em seu ambiente externo (por exemplo, uma mudança exigida por um hardware incompatível) ou porque o cliente exige acréscimos funcionais ou de desempenho. Novos requisitos podem ser descobertos ou surgirem a partir do próprio sistema de software, erros que não foram descobertos durante estágios anteriores podem ocorrer nesta fase. A manutenção do sistema de software reaplica

cada uma das etapas precedentes do ciclo de vida a um programa existente, e não a um novo.

As fases citadas acima devem seguir um cronograma de tempo e devem ser seqüenciais, ou seja, uma fase não deve ser iniciada sem que a fase precedente tenha sido concluída. Todo esse processo deve ser documentado e tais documentos aprovados. As fases devem ser complementares e ter comunicação entre si.

O processo em cascata pode oferecer opções de correção de erros e melhorias fazendo modificações do sistema. Para isso, deve-se observar em qual fase mais próxima do início do processo a modificação será realizada, e a partir desse ponto seguir seqüencialmente por todas as outras fases subseqüentes. Pode-se antes do término do desenvolvimento ser necessário voltar em uma fase anterior para modificações. Porém o retorno pode causar um impacto tão grande nas demais fases a ponto de se refazer todo o sistema nas fases subseqüentes. Geralmente, há muito intercâmbio de informações entre as fases, e este modelo não permite flexibilidade de retorno nas seqüências, tornando o desenvolvimento do sistema de software altamente engessado (REZENDE [2002]).

2.1.3 Prototipação

Segundo Pressman [1995], prototipação capacita o desenvolvedor criar um modelo do sistema de software que será implementado. É conhecido como desenvolvimento evolucionário, pois ele expõe resultados iniciais à aprovação do usuário e segue fazendo aprimoramentos através de várias versões, até que o sistema final, que atenda todos os requisitos, tenha sido desenvolvido. Prototipação pode assumir uma das três formas: (I) um protótipo em papel ou no computador que retrata a interação homem-máquina de uma forma que capacita o usuário a entender quanta interação ocorrerá; (II) um protótipo de trabalho que implementa algum subconjunto da função exigida do sistema de software desejado; ou (III) um programa existente que executa parte ou toda a função desejada, mas que tem outras características que serão melhoradas em um novo esforço de desenvolvimento.

Um protótipo é uma possível solução para o problema do software. Ele é apresentado aos usuários finais (em forma de telas, relatórios, formulários de

entrada de dados, entre outros) para que eles possam interagir e analisar o sistema de software possibilitando a verificação dos requisitos.

Segundo LAUDON [1998], a prototipagem consiste na construção de um sistema experimental, de forma rápida e barata para avaliação dos usuários finais. Interagindo com o protótipo, os usuários podem ter uma melhor idéia das informações requeridas. O protótipo validado pelos usuários pode ser usado como base para a criação do sistema final.

Na prototipação, as etapas de especificação, desenvolvimento e validação são intercaladas (Sommerville [2004]).

2.1.3.1 Tipos de Prototipação

2.1.3.1.1 Desenvolvimento exploratório

No desenvolvimento exploratório, o desenvolvimento se inicia com um protótipo das partes compreendidas e já especificadas, e através desse protótipo inicial o sistema evolui com o acréscimo de novas características na medida em que novos requisitos são estimados pelo cliente e compreendidos pela equipe de desenvolvimento.

Segundo Polloni [2001], um protótipo rápido e evolutivo é um modelo de trabalho parcialmente especificado de um sistema a ser desenvolvido, demonstrável durante a fase de análise e passível de modificação fácil e rápida, sendo capaz de evoluir para um produto final que pode ser entregue.

Segundo Sommerville [2004], o objetivo é trabalhar com o cliente e evoluir para um sistema final através de uma especificação de um esboço inicial. Começa-se com exigências bem compreendidas e adiciona características novas conforme é proposto pelo cliente.

2.1.3.1.2 Protótipos descartáveis

Outro tipo de protótipos são os protótipos descartáveis (throw away). Nesse tipo de prototipação, concentra-se em fazer protótipos com partes do sistema que estejam mal compreendidas, de forma a verificar se são aqueles requisitos que o cliente deseja ou não. Como não são partes do sistema de software propriamente e

somente para especificação de requisitos, esses protótipos são descartados, servindo somente para uma melhor compreensão dos requisitos.

2.1.3.2 Fases da prototipação

A seqüência de eventos para o paradigma de prototipação é mostrada na figura a seguir (Pressman [1995]):

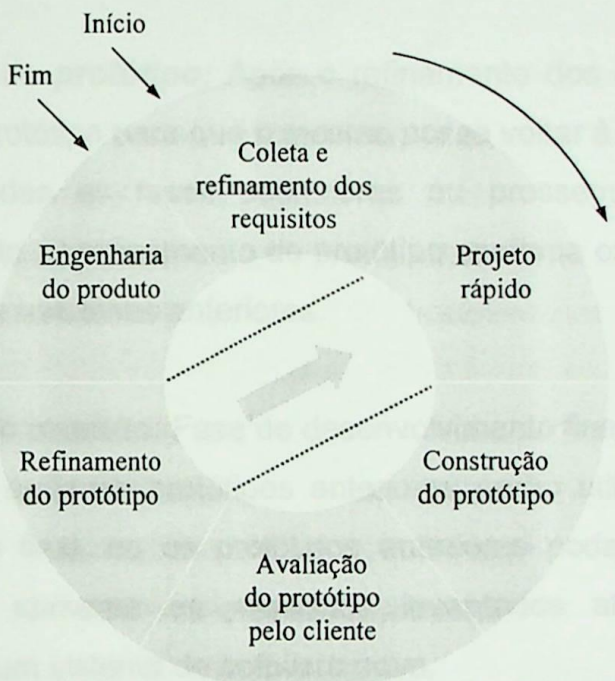

Figura 2. Prototipação.

Coleta e refinamento dos requisitos: todas as abordagens de desenvolvimento do sistema de software obrigatoriamente iniciam-se com a coleta dos requisitos. Na prototipação, a coleta dos requisitos sofre um refinamento para se estabelecer qual(is) protótipo(s) será(ão) desenvolvido(s). Os usuários, através do protótipo, podem identificar pontos positivos e negativos do sistema de software ocasionando novas idéias e refinamento nos requisitos do sistema.

Projeto Rápido: Concentra-se nas partes do sistema de software que serão visíveis ao usuário. Preocupa-se apenas em esboçar os requisitos básicos de interação com o usuário. É usado apenas para refinar os requisitos do sistema de software a ser desenvolvido. Esta fase é muito utilizada na prototipação descartável.

Construção do protótipo: Nessa fase, é construído um protótipo que pode vir a ser parte ou o início do sistema de software final, podendo ser também um melhoramento do protótipo rápido para uma melhor especificação dos requisitos.

Avaliação do protótipo pelo cliente: Ao se elaborar um protótipo, devemos apresentá-lo ao cliente para que possam ser encontrados pontos errôneos ou faltosos e pontos positivos e negativos do protótipo e assim serem redefinidos os requisitos corretamente.

Refinamento do protótipo: Após o refinamento dos requisitos, temos que dar refinamento ao protótipo para que o mesmo possa voltar à fase de projeto rápido e dela percorrer todas as fases posteriores ou prosseguir para a etapa de engenharia do produto. O refinamento do protótipo confirma os requisitos que foram sugeridos pelo cliente nas fases anteriores.

Engenharia do produto: Fase de desenvolvimento final do produto. Pode ser uma fase evolutiva, onde os protótipos anteriores serão utilizados para formar o sistema de software final, ou os protótipos anteriores podem ser descartados e levados em conta somente os requisitos levantados através deles para o desenvolvimento de um sistema de software novo.

2.1.4 Processo em Espiral

O processo em espiral foi desenvolvido para abranger as melhores características tanto do ciclo de vida clássico como da prototipação, acrescentando ao mesmo tempo, um novo elemento - a análise dos riscos – que falta a esses paradigmas (Pressman [1995]), (Cantor[1998]).

É um modelo evolucionário para o desenvolvimento de um sistema de software, baseado em uma seqüência de fases que culminam em versões incrementais do sistema de software. Este modelo foi introduzido por Barry Boehm em 1988 em um artigo publicado no IEEE (Boehm [1988]). A característica incremental é confirmada no conceito de prototipação rápida (LEACH [2000]), e em metodologias de processo mais modernas, como RUP, ICONIX e "Extreme Programming" descrita por Beck [2000].

O processo, representado pela espiral da figura a seguir, define quatro importantes atividades representadas pelos quatro quadrantes da figura:

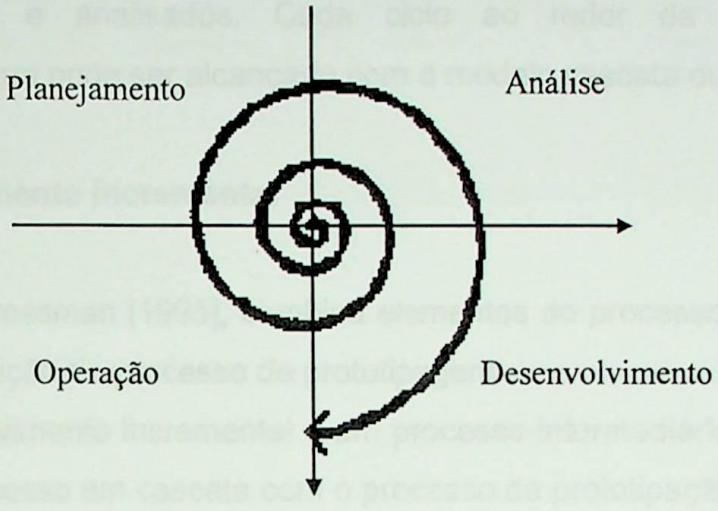

Figura 3. Processo em Espiral Simplificado

1. Planejamento: Definição dos objetivos, alternativas e restrições;
2. Análise de risco: Análise de alternativas e identificação e resolução de riscos.
3. Desenvolvimento: Desenvolvimento do produto.
4. Operação: Produto se torna operacional, é avaliado e se prepara para iniciar um novo ciclo.

Segundo BOEHM [1988], o processo em espiral era em 1988 uma inovação que combina os processos de prototipagem, convencional e incremental para ser usado em várias partes do desenvolvimento.

Segundo PAULA [2000], o produto é desenvolvido em uma série de iterações. Cada nova iteração corresponde a uma volta na espiral. Isto permite construir produtos em prazos curtos, com novas características e recursos que são agregados na medida em que a experiência descobre sua necessidade. As atividades de manutenção são usadas para identificar problemas; seus registros fornecem dados para definir os requisitos das próximas liberações. O principal problema desse ciclo de vida é a necessidade de uma gestão sofisticada para ser previsível e confiável.

Se a análise de risco indicar que há dúvidas nos requisitos, a prototipação pode ser usada no quadrante de desenvolvimento para assegurar o projeto.

Entretanto, em muitos casos, o fluxo ao redor da espiral continua, onde cada rotação indica uma evolução significativa do projeto, até a sua completa conclusão. Durante os ciclos ao redor da espiral, objetivos, alternativas e restrições são definidos, riscos são identificados e analisados. Cada ciclo ao redor da espiral, envolve desenvolvimento que pode ser alcançado com o modelo cascata ou prototipação.

2.1.5 Desenvolvimento Incremental

Segundo Pressman [1995], combina elementos do processo em cascata com a filosofia de repetição do processo de prototipagem.

O desenvolvimento incremental é um processo intermediário que combina as vantagens do processo em cascata com o processo de prototipação. O processo em cascata é um processo de gerenciamento simples e pode ser utilizado para sistemas de grande porte possibilitando mudanças nesse sistema posteriormente por separar projeto e implementação; mas os requisitos devem ser específicos e definidos no inicio do projeto. O processo de prototipação permite que os requisitos sejam definidos posteriormente se tornando um modelo evolucionário.

Segundo Mills [1999], o desenvolvimento incremental é um meio de reduzir o "retrabalho" no processo de desenvolvimento e de proporcionar aos clientes algumas oportunidades de adiar decisões sobre seus requisitos detalhados, até que eles tenham alguma experiência com o sistema. A figura a seguir descreve o desenvolvimento incremental.

Segundo Sommerville [2003], um processo de desenvolvimento incremental possui primeiramente um esboço onde os clientes identificam as funções a serem fornecidas pelo sistema. Eles identificam quais funções são menos ou mais importantes para eles. Em seguida é definida uma série de prazos de entrega para cada subconjunto de funcionalidades do sistema. A alocação de funções a cada estágio de entrega depende da prioridade da função. As funções prioritárias são entregues primeiramente ao cliente.

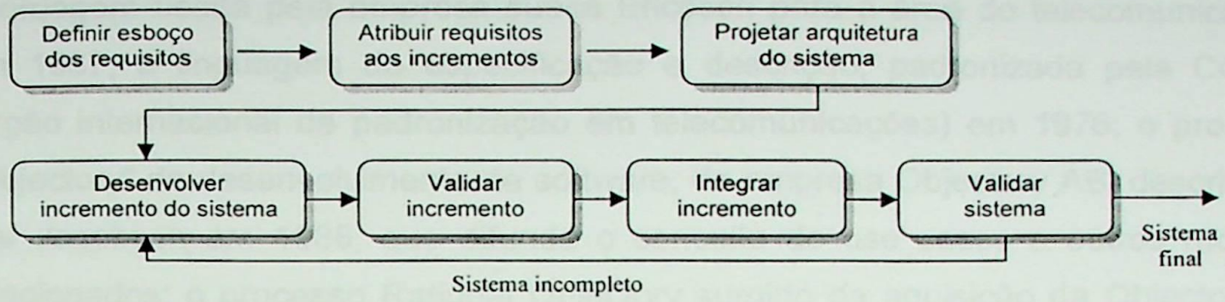

Figura 4. Desenvolvimento incremental

A definição do esboço dos requisitos é a etapa inicial do processo incremental. Através desse esboço devemos especificar incrementos que devem ser adicionados ao sistema e realizar o levantamento dos requisitos desses incrementos. Cada incremento é realizado individualmente e, depois de pronto é adicionado ao sistema; assim o cliente tem contato desde o início com o sistema de software e vai recebendo em partes as funcionalidades do sistema. A cada incremento pronto, o processo de desenvolvimento de um novo incremento é iniciado. O cliente pode experimentar o sistema enquanto o mesmo é desenvolvido, possibilitando esclarecer os requisitos para os incrementos subseqüentes e assim possibilitar uma versão posterior melhorada. A funcionalidade do sistema vai melhorando a cada incremento concluído e integrado ao sistema. Isso permite ao usuário, que tem interação com o sistema, aprender a lidar com o sistema gradativamente, facilitando seu uso posterior e compreensão maior de todas as funcionalidades do sistema.

Segundo Sommerville [2004], o desenvolvimento incremental reduz o risco de falha total do projeto.

2.1.6 RUP

O RUP (Processo Unificado da Rational) vem acompanhando a evolução dos sistemas de software e da UML, oferecendo um processo de desenvolvimento orientado por casos de uso e componentes, centrado na arquitetura, sendo interativo e incremental.

Segundo SCHNEIDER [1998], é um processo de desenvolvimento de sistema de software especificado que pode ser especializado para diferentes tamanhos, áreas e classes de projetos. Suas origens incluem, em ordem cronológica: a

abordagem usada pela empresa sueca Ericsson para a área de telecomunicações em 1967; a linguagem de especificação e descrição, padronizada pela CCI TT (órgão internacional de padronização em telecomunicações) em 1976; o processo "Objectory" de desenvolvimento de software, da empresa Objectory AB, descrito por Ivar Jacobson em 1988, que difunde o conceito de use cases e outros modelos relacionados; o processo Rational Objectory surgido da aquisição da Objectory AB pela empresa Rational em 1995 e a linguagem de modelagem UML desenvolvida por Grady Booch, Ivar Jacobson e James Rumbaugh em 1995 e considerada um padrão pelo OMG (Object Management Group) em 1997. Em 1998 o processo passou a chamar-se RUP - Rational Unified Process.

Fowler [2001] diz que existe tanto material a ser produzido para seguir uma das antigas metodologias de produção de sistemas de software, que a velocidade de desenvolvimento diminui, pois as metodologias antigas são muito burocráticas e conceituais. Como uma reação a estas metodologias, um grupo novo de metodologias apareceu nos últimos anos (tomando como referência o ano de 2001). Estes novos métodos tentam estabelecer um compromisso útil entre nenhum processo e processo demasiado, provendo apenas processo suficiente para fornecer uma vantagem razoável. As metodologias modernas de desenvolvimento, como o RUP e SCRUM, são uma reação a esses modelos antigos.

2.1.6.1 Objetivos do RUP

O RUP é um processo de Engenharia de Software que tem os seguintes objetivos:

- Atendimento às necessidades dos usuários;
- Melhoria da produtividade;
- Desenvolvimento com alta qualidade;
- Atuação segundo cronograma e orçamento previsíveis.

2.1.6.2 Fases do RUP

O RUP é composto por quatro fases incrementais (Concepção, Elaboração, Construção e Transição). Essas fases possuem fluxos de trabalhos que podem variar de acordo com cada projeto de sistema de software, mas seguem

basicamente uma seqüência lógica e um modelo. Essas fases e fluxos de trabalho são mostrados na figura seguir.

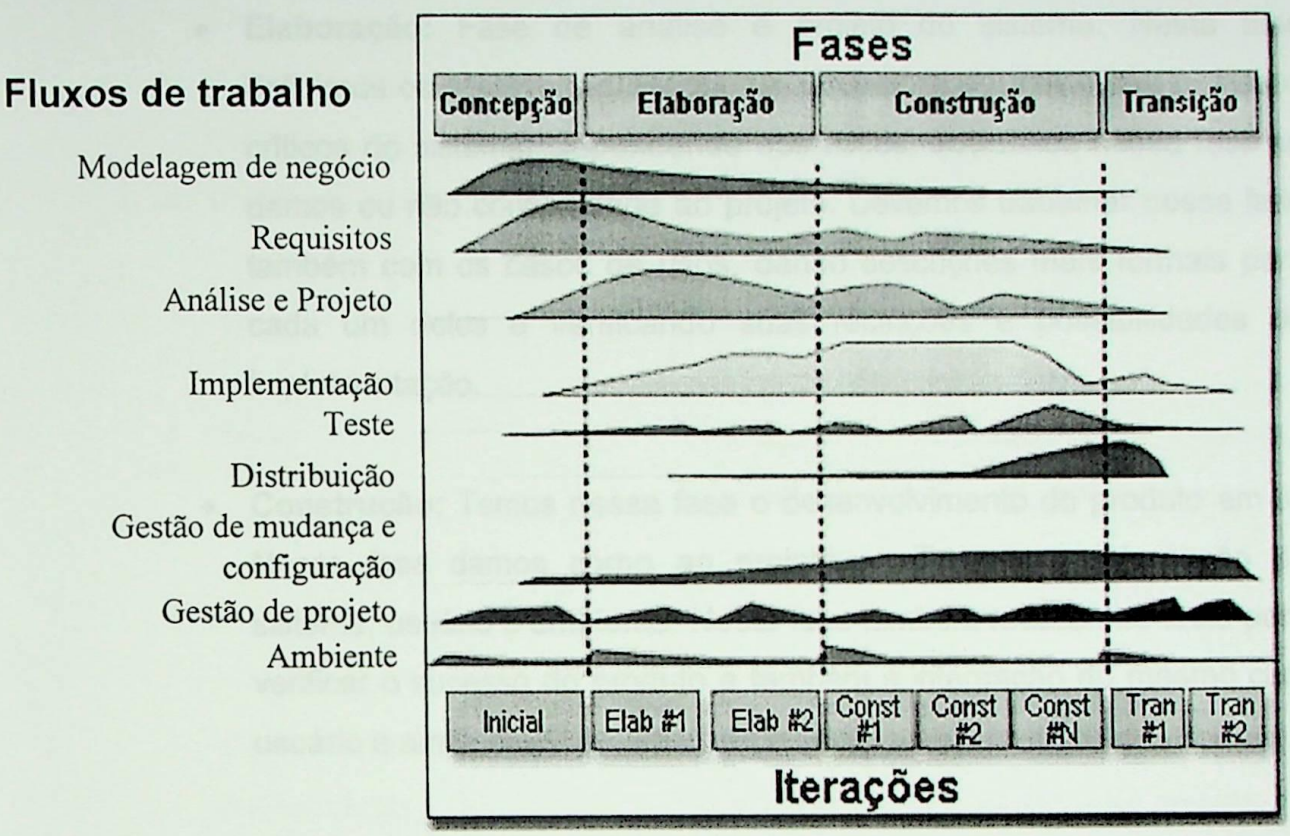

Figura 5. Fases e iterações de RUP.

As fases do RUP possuem propriedades e eventos particulares a cada uma delas, mas com interações entre as mesmas para o sucesso do projeto. As fases são incrementais e complementares, isto é, uma fase complementa a fase anterior que na verdade nada mais é do que um incremento seqüencial. As fases do RUP podem ser descritas como:

- **Concepção:** Fase inicial do projeto, onde o mesmo é concebido e definido. É a fase de definição dos requisitos. Devem ser realizadas nessa fase entrevistas, todas as técnicas para levantamento de requisitos, verificação da viabilidade do projeto, análise de Custos/Benefícios e elaboração e análise de casos de uso. Segundo ROYCE [1998], na fase de concepção são estabelecidos o caso de negócio e o escopo do projeto. O caso de negócios inclui critérios de sucesso, avaliação de riscos, estimativa de recursos necessários e um

plano para a fase. Durante a concepção é comum a criação de um protótipo executável, servindo como teste para a concepção.

- **Elaboração:** Fase de análise e projeto do sistema. Nesta fase definimos os objetivos, o escopo, a arquitetura e resolvemos os riscos críticos do sistema. Dependendo dos riscos, decidimos nessa fase se damos ou não continuidade ao projeto. Devemos trabalhar nessa fase também com os casos de usos, dando descrições mais formais para cada um deles e verificando suas restrições e possibilidades de implementação.

- **Construção:** Temos nessa fase o desenvolvimento do produto em si. Nessa fase damos corpo ao projeto, verificamos a integração do sistema, usuário e ambiente. Nesta fase também realizamos teste para verificar o sucesso do produto e também a integração do mesmo com usuário e ambiente.

- **Transição:** Fase que deixa o produto final disponível ao usuário e adaptável ao ambiente. Esta fase classicamente é iniciada com uma versão beta do sistema, pois sempre irão surgir alterações e ajuste no sistema, devido a problemas identificados em testes ou a novas características propostas. Nesta fase, também pode estar contido o treinamento ao usuário e adaptação ao sistema. Após a fase de transição o produto pode voltar a percorrer todas as fases surgindo assim um produto melhorado ou uma nova versão do produto.

A estrutura do RUP pode ser adaptável para se atender às necessidades de uma organização ou projeto de sistema de software. Um processo não é para ser seguido cegamente, pois muitas vezes pode ser inútil uma etapa e isso causará trabalho excessivo ao projeto atrapalhando o cronograma do mesmo.

A figura 5 mostra ainda os fluxos de trabalho do RUP e a intensidade com que cada um se relaciona às fases do RUP. Segundo Booch [2000], o RUP é composto por nove fluxos de trabalho de processos. São eles:

1. Modelagem de negócios	Descreve a estrutura e a dinâmica da empresa.
2. Requisitos	Descreve os requisitos através de casos de uso.
3. Análise e projeto	Descreve as várias visões da arquitetura.
4. Implementação	Leva em consideração o desenvolvimento do sistema de software, o teste da unidade e a integração.
5. Teste	Descreve casos de teste, procedimentos e medidas para acompanhamento de erros.
6. Distribuição	Abrange a configuração do sistema a ser entregue.
7. Gestão de mudança e configuração	Controla modificações e mantém a integridade dos artefatos do projeto.
8. Gestão de projeto	Descreve as várias estratégias para o trabalho com um processo interativo.
9. Ambiente	Abrange a infra-estrutura necessária para o desenvolvimento do sistema.

2.1.7 XP (eXtreme Programming)

O XP é uma metodologia de desenvolvimento rápido de sistemas proposta em 1999 por Kent Beck em seu livro "*Extreme Programming Explained*" [Beck (1999)].

XP consiste em quatro partes: valores, princípios, atividades, e práticas. A figura 6 demonstra como são dispostas entre si, com atividades funcionais em torno ou durante todo o ciclo de vida.

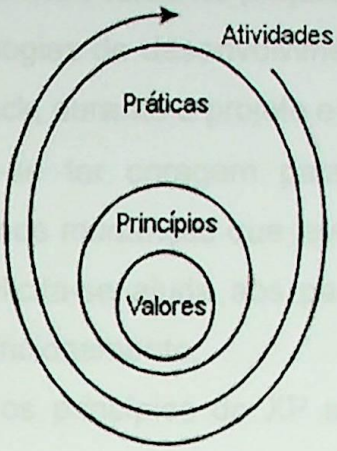

Figura 6. Partes do XP

Segundo Stephens [2003], os valores do XP são divididos em quatro partes:

- **Comunicação:** se todos estiverem na mesma sala e se comunicarem bem, então há menos possibilidades de enganos. XP supõe que a forma mais expressiva de uma comunicação é a verbal - e que uma comunicação verbal é menos susceptível ao erro e aos enganos do que uma comunicação escrita.
- **Simplicidade:** é a chave de tudo: sempre tentar fazer o mais simples possível para realizar o trabalho. Seja breve, simples e eficaz, ao invés de fazer uma coisa complicada para conseguir o mesmo objetivo. É comum mencionar que mesmo quando você segue o item mais simples, o simples absoluto não é sempre a melhor solução. Às vezes um problema complexo requer uma solução ligeiramente complexa. Exigências não funcionais (tais como disponibilidade elevada) podem afetar o projeto. Assim a coisa mais simples pode não ser possível de ser realizada, levando assim a realizar a complexa ou mais próxima dela.
- **Feedback:** Consiste no retorno ao cliente para verificar se é realmente o que ele queria. Evitando erros e evoluindo sempre para um sistema final melhor e que atenda aos requisitos do cliente. Constrói-se um protótipo dos primeiros requisitos entendidos do sistema (uma parte pequena de cada vez), testa-se, e ajusta-se o projeto até que esteja correto. A primeira versão é limitada para ser ligeiramente descartada, mas permite que se faça um rápido começo, de modo que se possam

fazer ajustes finos mais tarde no projeto. Segundo Cockburn [2001], de todas as metodologias de desenvolvimento, o XP talvez tenha a maior ênfase no feedback, durante o projeto e no projeto total.

- **Coragem:** Deve-se ter coragem para fazer mudanças no projeto. Fazem-se pequenas mudanças que levem o projeto mais rapidamente à finalização. Solicita-se ajuda aos parceiros. Melhora-se um código que já está em funcionamento.

Segundo Beck [2000], os princípios do XP são divididos em quatro partes distintas, descritas abaixo:

- **Feedback Rápido:** o tempo entre uma ação e seu feedback é crítico. Diferenças pequenas no sincronismo do feedback resultam em diferenças grandes no desenvolvimento. Assim, um dos princípios básicos de feedback é: interprete-o, e coloque-o em desenvolvimento para satisfazer o sistema tão rápido quanto possível. Os programadores aprendem como melhor projetar, testar e alimentar o sistema com o que se aprendeu em segundos ou em minutos, em vez de dias, semanas, ou meses.
- **Assuma a simplicidade:** trate cada problema como se pudesse ser resolvido com simplicidade. XP diz: faça um bom trabalho, (testes, reconstrução, comunicação,...) resolva o trabalho de hoje e confie em sua habilidade de adicionar a complexidade no futuro, onde você necessitará dela.
- **Mudança Incremental:** as grandes mudanças feitas de uma única vez não funcionam. Todo o problema deve ser resolvido com uma série de mudanças. Com partes menores, pode-se fazer a diferença. Se for necessário, divida uma mudança grande em várias outras pequenas e implemente uma de cada vez.

Segundo Beck [2000], as 12 práticas em XP são como segue:

- **O jogo de planejamento**: determinar rapidamente o escopo para o próximo passo, combinando prioridades do negócio e estimativas técnicas. O pessoal técnico precisa decidir-se sobre estimativas, conseqüências, processo, e detalhes de programação. Envolve dividir o projeto em partes.

- **Pequenas versões**: ponha um sistema simples em produção rapidamente, libere então versões novas em um ciclo muito curto.
- **Metáfora**: guie todo o desenvolvimento com uma história simples e compartilhada de como o sistema inteiro trabalha.
- **Projeto Simples**: o sistema deve ser projetado tão simples quanto possível em todos os momentos. A complexidade extra é removida assim que é descoberta.
- **Teste:** os programadores escrevem continuamente testes de unidade, os quais devem atender a todos os requisitos já propostos, para que o desenvolvimento continue. Os clientes escrevem os testes que demonstram que as características estão concluídas e satisfatórias.
- **Reconstrução**: os programadores reestruturam o sistema, sem mudar seu comportamento, para remover a duplicação, melhorar uma comunicação, simplificar, ou adicionar flexibilidade.
- **Programação em duplas:** todo o código da produção é escrito com dois programadores em uma máquina.
- **Propriedade Coletiva:** todo o código fonte pertence e é acessível a toda equipe. Assim qualquer um pode mudar partes do código em qualquer lugar dentro do sistema e em qualquer momento.
- **Integração Contínua:** integra e constrói o sistema muitas vezes em um único dia, a cada vez que uma tarefa é terminada.
- **Semana de 40 horas**: assuma como regra os desenvolvedores não trabalharem mais de 40 horas semanais.
- **Cliente Presente**: inclua um "usuário/cliente" real no projeto, disponível a tempo integral para responder a perguntas, determinar requisitos e atribuir prioridades.
- **Códigos Padrões**: os programadores escrevem todo o código de acordo com as regras que enfatizam uma comunicação com o código, e que todos conhecem, de forma a ser claro para todos.

A sinergia dessas regras dão sentido ao XP. Muitas dessas regras adotadas separadamente não teriam sentido algum, mas o conjunto delas sustenta uma revolução nas metodologias ágeis, o XP.

Segundo Beck [2000], as quatro atividades básicas do XP são descritas abaixo:

- **Codificação:** Não existe desenvolvimento do sistema de software sem codificação. Esta é uma atividade básica em qualquer processo de desenvolvimento.
- **Teste:** Em cada etapa de desenvolvimento vencida, devem ser feitos testes que comprovem a satisfação de todos os requisitos daquela etapa.
- **Escutar:** O cliente diz ao programador o que o sistema deve fazer; o programador codifica o sistema. Porque é da natureza humana basear nossas interpretações em nossa própria experiência prévia, todos devem "ouvir" as exigências do cliente (XP inovou, colocando um "cliente" entre a equipe de desenvolvimento).
- **Projeto:** você não pode apenas escutar, escrever um exemplo, testá-lo e colocá-lo para funcionar, tudo isso tem que ser planejado. Projetar é preciso, para que as coisas saiam corretamente.

Jefries [2001] diz que se deve começar com toda a equipe junta e gastar alguns minutos para esboçar um projeto de produção. Dez minutos são ideais, meia hora deve ser o tempo máximo gasto para fazer isso. Após isso, a melhor coisa a fazer é deixar a sessão de projeto e dedicar-se à codificação.

O XP tem integração constante e testes automatizados. Freqüentemente liberam-se pequenas partes que se incorporam ao produto do cliente continuamente. A união da equipe de desenvolvimento faz dele uma flexível e eficaz ferramenta de desenvolvimento de sistema de software (Beck [2000]).

2.1.8 FDD (Feature-Driven Development)

O FDD (Desenvolvimento Dirigido por Funcionalidade) surgiu também em meados de 1997, em um projeto para um grande banco em Singapura, a partir de técnicas de gerenciamento de projetos e de modelagem orientada a objetos. Foi lançado por Peter Coad, Eric Lefebvre e Jeff de Luca em "Java Modeling in Color with UML" em 1999 (Coad [1999]). É uma metodologia de desenvolvimento ágil com ciclos curtos, foco no cliente e ênfase na programação (Coad [1999]).

Segundo Coad [1999], FDD é um processo dirigido por modelos e de curta iteração. Começa com o estabelecimento de um modelo geral e continua com uma série de projetos curtos de, no máximo, duas semanas.

O FDD é um *processo ágil* que visa: Foco na satisfação do cliente; Entrega progressiva e incremental do sistema de software; Pequenas equipes altamente motivadas; Métodos informais; Simplicidade no desenvolvimento; Foco na entrega do produto final ao invés da análise. Possui um conjunto de processos com poucas e curtas iterações. Veja na figura a seguir o conjunto de processos.

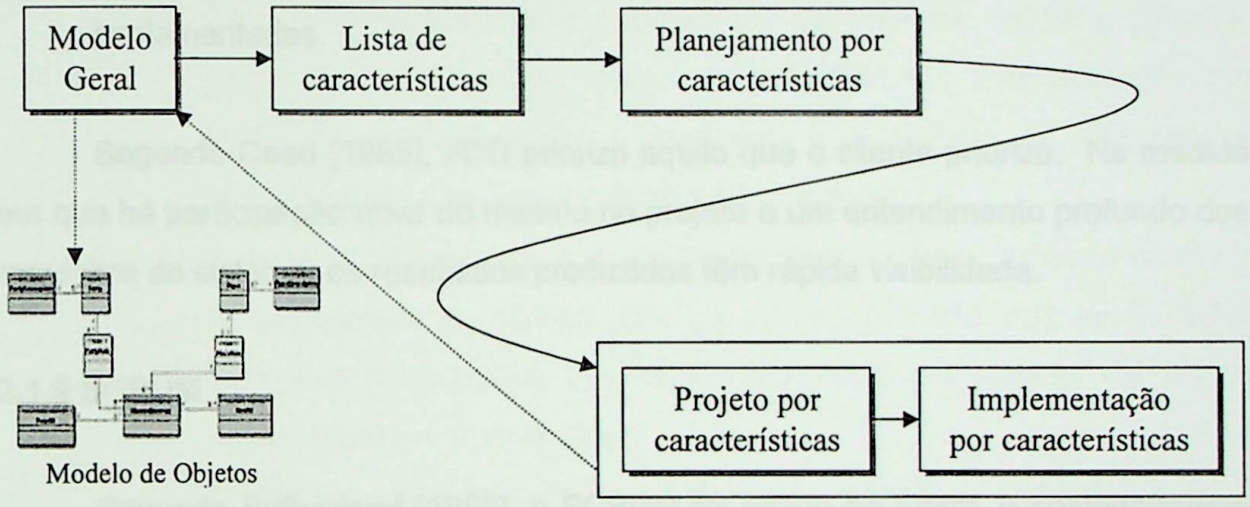

Figura 7. Conjunto de processos FDD

Segundo Coad [1999], as etapas do processo FDD são descritas abaixo:

Modelo Geral: Desenvolva um modelo geral, que atenda às características e exigências iniciais. Liste alguns componentes e focalize a forma final. Membros da equipe de desenvolvimento e o cliente, sob a orientação de um projetista experiente em modelagem de componentes e objetos, desenvolvem uma estrutura geral do sistema.

Lista de características: Construa uma lista detalhada das características do sistema. Coloque-as em ordem, estabelecendo uma hierarquia de prioridades.

Planejamento por características: Seguindo a prioridade das características, listadas no processo anterior, selecione uma quantidade

pequena de características para serem implementadas, lembrando que tal implementação não deve ultrapassar duas semanas.

Projeto por características: Projete, usando os componentes e sempre utilizando a seqüência de características da lista. Um estudo das documentações produzidas anteriormente deve ser realizado. As equipes devem ser divididas e distribui-se uma certa quantidade de características do sistema para cada equipe implementar.

Implementação por características: Resolva apenas as características selecionadas e integre-as ao sistema. Desenvolvimento de classes e métodos, inspeção do código, testes das unidades e integração ao sistema final fazem parte dessa etapa. O cliente deve aprovar as características implementadas.

Segundo Coad [1999], FDD prioriza aquilo que o cliente prioriza. Na medida em que há participação ativa do mesmo no projeto e um entendimento profundo dos requisitos do sistema, os resultados produzidos têm rápida visibilidade.

2.1.9 SCRUM

Segundo Sutherland [1998], o SCRUM foi criado na Easel, e posteriormente desenvolvido por duas empresas em conjunto: Advanced Development Methods e VMARK. Seu objetivo é fornecer um processo conveniente para projeto e desenvolvimento orientado a objeto. É uma metodologia classificada como ágil.

As etapas do SCRUM são de, no máximo, 30 dias. São chamadas de iterações ou sprints. As equipes são pequenas de até sete pessoas. Diariamente, deve ser feita uma reunião de 15 minutos, onde deve ser exposto à equipe o que será desenvolvido e avaliado o progresso do desenvolvimento (Fowler [2001]).

Segundo Schwaber [2001], SCRUM é focado nas pessoas e é indicado para ambientes em que os requisitos surgem e mudam rapidamente.

Conforme Abrahamsson [2002, p.30], os *Sprints* incluem as fases tradicionais do desenvolvimento do sistema de software: requisitos, análise, projeto e entrega.

A figura a seguir ilustra o processo SCRUM (Schwaber [2001]):

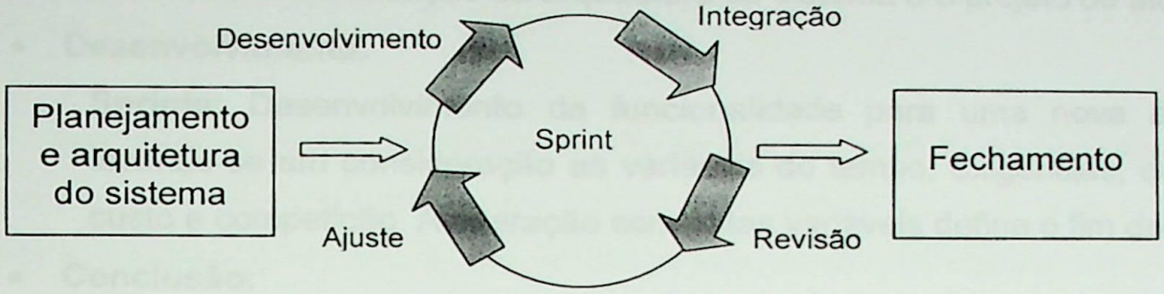

Figura 8. Processo Scrum

Segundo Schwaber [2001], as características do processo SCRUM são:

- Fase de Planejamento e Arquitetura: consiste em definir processos. Todos os processos de entrada e de saída são bem definidos e a forma de como executar o processo é explicitada. A definição de processos pode ocorrer também no fechamento;
- Fase de *sprints:* É um processo empírico, ou seja, baseado somente na experiência da equipe. São os sprints que dão vida ao SCRUM, com seus ciclos curtos. *Sprints* são flexíveis e não-lineares. Os *sprints* evoluem para o produto final;
- O projeto permanece aberto ao ambiente até a fase de fechamento. O produto a ser entregue pode ser mudado a qualquer momento durante o planejamento e a fase de *sprints* do projeto. O projeto permanece aberto à complexidade do ambiente, inclusive à competitividade, tempo, qualidade, e pressões financeiras, ao longo destas fases;
- O produto a ser entregue é determinado durante o projeto, baseado no ambiente.

Segundo Schwaber [2001], o SCRUM tem três fases descritas a seguir:

- **Início:**

  Planejamento: Definição de uma nova versão baseada no conhecimento prévio, junto com uma estimativa de programação e custo. Se um sistema novo estiver sendo desenvolvido, esta fase consiste na conceitualização e na análise. Se um sistema existente estiver sendo melhorado, esta fase consiste em análise limitada.

Arquitetura: Descreve como os itens da versão serão executados. Esta fase inclui a modificação da arquitetura do sistema e o projeto de alto nível.

- **Desenvolvimento:**

  Sprints: Desenvolvimento da funcionalidade para uma nova liberação, levando-se em consideração as variáveis do tempo, exigências, qualidade, custo e competição. A interação com estas variáveis define o fim desta fase.

- **Conclusão:**

  Fechamento: Preparação para a liberação, incluindo a documentação final, estágio de teste e liberação.

2.2. Modelagem e projetos baseados em objetos

A modelagem é o processo de construção de um modelo. Um modelo é uma abstração de algo real, podendo ser feito antes mesmo deste algo se consolidar. O modelo pode dar uma visão do projeto antes de sua execução. Essa visão externa do projeto pode ajudar a reconhecer e até a corrigir erros antes mesmo desses acontecerem, e se ter assim um projeto mais confiável.

Segundo Booch [2000], um modelo é uma simplificação da realidade. Construímos modelos para compreender melhor o sistema que estamos desenvolvendo.

2.2.1 A modelagem de Objetos

Um sistema modelado através de objetos incorpora uma estrutura estática desse sistema, mostrando os objetos, os atributos e as operações que caracterizam cada classe de objetos.

Os modelos baseados em objetos são mais aproximados do mundo real e mais adaptáveis à mudança. Os modelos de objetos apresentam uma intuitiva representação gráfica de um sistema e são úteis para a comunicação com os clientes e para a documentação da estrutura do sistema (Rumbaugh [1994]).

2.2.1.1 Objetos e Classes

2.2.1.1.1 Objetos

Um objeto é simplesmente alguma coisa que faz sentido no contexto de uma aplicação. Definimos um objeto como um conceito, uma abstração, algo com limites nítidos e significado em relação ao problema em causa. Os objetos servem a dois objetivos: eles facilitam a compreensão do mundo real e oferecem uma base real para a implementação em computador (Rumbaugh [1994]).

Um objeto é tudo que existe no mundo real e tem relação direta ou indireta entre eles ou com pessoas.

Os objetos que compartilham um mesmo conjunto integrado de atributos e serviços, mesmas hierarquias e comunicação com outros objetos, compõem uma classe; isto é, pode-se dizer que uma classe é a definição de um conjunto de objetos quase idênticos, onde os objetos são instâncias desta classe (Coad [1992]).

Segundo Sturm [1999], um programador com uma visão de orientação a objetos vê um sistema de software como um conjunto de objetos que interagem entre si. Um objeto é uma abstração de uma entidade em um domínio do problema. Como no mundo real, os objetos estão relacionados e comunicam-se também entre si.

2.2.1.1.2 Classes

Uma classe de objetos descreve um grupo de objetos com propriedades semelhantes (atributos), mesmo comportamento (operações), possibilidades de relacionamentos com outros objetos e mesma semântica (Rumbaugh [1994]).

Cada objeto pertencente a uma classe possui os mesmos atributos e padrões, porém se diferenciam pelo valor de seus atributos e do relacionamento com outros objetos.

Uma classe define objetos do mesmo tipo, porém estes objetos podem ter valores e características diferentes. Veja na figura a seguir o exemplo da classe carros e como ela pode ter objetos que se diferenciam entre si, mas não deixam de ter as mesmas características de um carro. A classe deve ter um nome sugestivo aos objetos que a compõem, facilitando assim a sua compreensão e utilização.

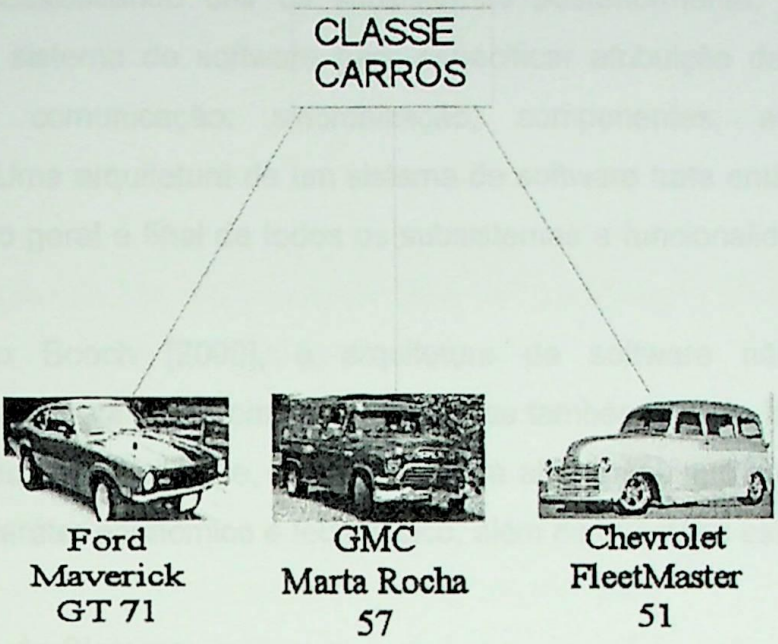

Figura 9. Exemplo da classe carros.

Podemos dizer que o objeto é o que existe de fato, e a classe é uma abstração desses objetos.

2.2.2 Modelagem de Sistemas

Em 1997, o grupo de gerência de objetos (OMG) liberou a Linguagem de Modelagem Unificada (UML). Uma das finalidades da UML era fornecer à comunidade de desenvolvimento uma linguagem estável e comum de projeto que poderia ser usada para desenvolver e construir aplicações de computador. UML trouxe adiante uma notação de um padrão unificado de modelagem que profissionais quiseram por anos. Usando UML, profissionais podem agora ler e disseminar modelos da estrutura e do projeto do sistema.

Um sistema de software bem sucedido é aquele capaz de atender às necessidades dos usuários com qualidade. Para produzir tal sistema de software, de maneira eficiente, com eficácia de recursos e de maneira previsível, utilizamos a modelagem de sistemas.

Os sistemas grandes e complexos devem ser subdivididos em subsistemas, para facilitar o trabalho com os mesmos e diminuir a complexidade de cada um em particular. Resolvendo todos os pequenos subsistemas, resolvemos o sistema

grande e complexo de forma eficaz. Para tal, precisamos projetar a estrutura geral do sistema, possibilitando unir os subsistemas posteriormente. Temos então a arquitetura do sistema de software para especificar atribuição de funcionalidade, protocolo de comunicação, sincronização, componentes, escalabilidade e desempenho. Uma arquitetura de um sistema de software trata então dos requisitos para integração geral e final de todos os subsistemas e funcionalidades do sistema final.

Segundo Booch [2000], a arquitetura de software não está apenas relacionada à estrutura e ao comportamento, mas também ao uso, à funcionalidade, ao desempenho, à flexibilidade, à reutilização, à abrangência, às adequações e às restrições de caráter econômico e tecnológico, além de questões estéticas.

2.2.2.1 Visões do Sistema

A figura a seguir apresenta as cincos visões possíveis de um sistema de software, e cada visão foca um aspecto específico do mesmo, sendo todas necessárias para o sucesso do produto.

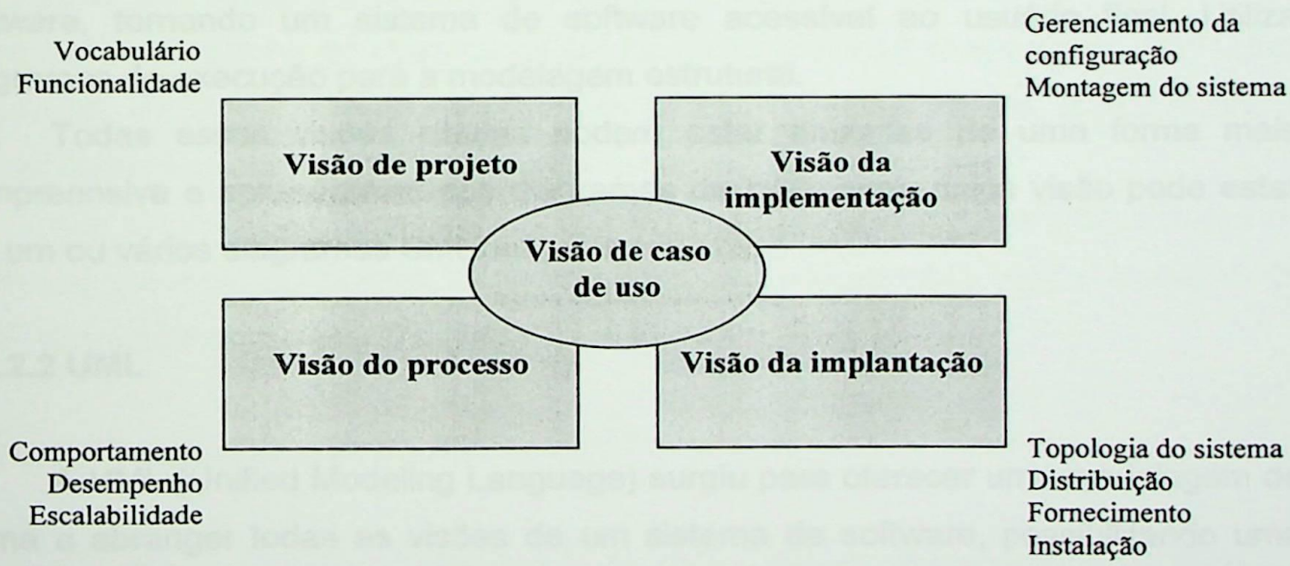

Figura 10. Modelagem da arquitetura de um sistema.

A visão do caso de uso descreve o comportamento do sistema na visão do usuário final, dos analistas e do pessoal de teste. Ela abrange um pouco de cada visão, tentando demonstrar as interações do sistema com os meios externos. Os

diagramas utilizados nessa visão são diagramas de caso de uso e diagramas de atividades, ambos para a modelagem comportamental.

A visão de projeto é uma visão lógica do sistema e é a visão de maior interesse para os analistas e projetistas. Ela abrange as classes, interface e colaborações do sistema, formando o vocabulário do problema e de sua solução. Está relacionada na UML com os diagramas de classes para a modelagem estrutural, diagramas de colaboração para a modelagem comportamental e diagramas de estados para a modelagem estrutural.

A visão do processo abrange todos os processos que tornam funcional o sistema de software, cuida do comportamento, do desempenho e da escalabilidade do sistema. Utiliza os diagramas de objetos para a modelagem estrutural e diagramas de seqüência para a modelagem comportamental.

A visão da implementação abrange gerenciamento de versões, gerenciamento de disco e arquivos para se obter um fornecimento do sistema físico do software, ou seja, um sistema final executável. Utiliza diagramas de componentes para a modelagem estrutural

A visão da implantação consiste na instalação e distribuição do sistema de software, tornando um sistema de software acessível ao usuário final. Utiliza diagramas de execução para a modelagem estrutural.

Todas essas visões citadas podem estar descritas de uma forma mais compreensiva e apresentável nos diagramas da UML; uma única visão pode estar em um ou vários diagramas UML.

2.2.2.2 UML

A UML (Unified Modeling Language) surgiu para oferecer uma modelagem de forma a abranger todas as visões de um sistema de software, possibilitando uma maior interação entre analista, projetistas, programadores e demais profissionais envolvidos no desenvolvimento de um sistema.

Segundo Reed [2000]:

> *"A UML não é um modelo de processo de software ou uma metodologia de desenvolvimento de sistemas, ela é uma notação, um mecanismo para "apontar o problema" de forma a expor a essência do domínio de um aplicativo. A combinação da UML com um bom modelo*

de processo resulta em uma poderosa combinação para a criação de aplicativos bem-sucedidos".

Segundo Pilone [2005], a UML tenta construir uma ponte entre a idéia original e uma parte do sistema de software e sua execução. A UML fornece uma maneira para capturar e discutir exigências (Diagramas de Caso do Uso). Há diagramas para capturar as partes do sistema de software que realizam determinadas exigências (Diagramas de Colaboração). Diagramas para capturar exatamente como aquelas partes do sistema realizam suas exigências (Diagramas de Seqüência e de Gráfico de Estado). Finalmente há diagramas para mostrar como tudo se une para juntos serem executados (Diagramas de Componentes e de Distribuição). Há diversos tipos, alguns com finalidades muito específicas e alguns com usos mais genéricos.

A UML modela o sistema de software através de seus vários diagramas citados a seguir.

2.2.2.3 Diagramas

A UML modela um sistema e sua interação com o usuário através de diagramas padronizados. São treze diagramas, entre eles, diagrama de caso de uso (Use Case), diagrama de classes, diagrama de seqüência e diagrama de atividades. Estes são os mais utilizados e os utilizaremos em nosso projeto, considerando a complexidade média do domínio do problema em estudo.

Os diagramas UML esboçam todas as visões de um projeto de sistema de software, possuindo diagramas estruturais, comportamentais e de modelos de gerenciamento especificados a seguir.

2.2.2.3.1 Diagramas Estruturais

Os quatro diagramas estruturais da UML existem para visualizar, especificar, construir e documentar os aspectos estáticos de um sistema (Booch [2000]). Esses aspectos estáticos podem ser comparados a uma estrutura estática relativamente estável, como por exemplo, os aspectos estáticos de um carro que incluem a existência e a colocação de chassis, carroceria, rodas, motor, fios e demais peças. Também os aspectos estáticos de um sistema de software abrangem toda sua

estrutura, a existência e a colocação de itens como classes, interfaces, colaborações e componentes. Os diagramas são

1. Diagrama de classes
2. Diagrama de objetos
3. Diagrama de componentes
4. Diagrama de implantação/Execução

Os diagramas acima serão explicados a seguir neste mesmo capítulo.

2.2.2.3.2 Diagramas Comportamentais

Os cinco diagramas comportamentais da UML são utilizados para visualizar, especificar, construir e documentar os aspectos dinâmicos de um sistema (Booch [2000]).

Diferentemente dos diagramas estruturais que são estáticos, os diagramas comportamentais focam a parte dinâmica do sistema visando o comportamento do mesmo. Os diagramas são:

1. Diagrama de caso de uso
2. Diagrama de seqüências
3. Diagrama de colaboração
4. Diagrama de gráfico de estados
5. Diagrama de atividades

Os diagramas acima serão explicados a seguir neste mesmo capítulo.

2.2.2.3.3 Diagramas de Modelos de Gerenciamento

Os quatro diagramas de modelos de gerenciamento da UML são utilizados para representar subsistemas e modelos de um sistema (Pender [2003]).

1. Diagrama de pacotes
2. Diagrama de Visão Geral de interação
3. Diagrama de temporização
4. Diagrama de estruturas compostas

Os diagramas acima serão explicados a seguir neste mesmo capítulo.

2.2.2.3.4 – Diagrama de Classe

O propósito do Diagrama de Classe é modelar a estrutura estática do sistema. O diagrama especificamente mostra as entidades dentro do sistema e junto com cada entidade sua estrutura interna e relações com outras entidades no sistema. Em um diagrama de classe não são mostrados exemplos específicos de preenchimento de cada classe, somente sua estrutura. Veja na figura a seguir o exemplo simples de uma classe em um diagrama:

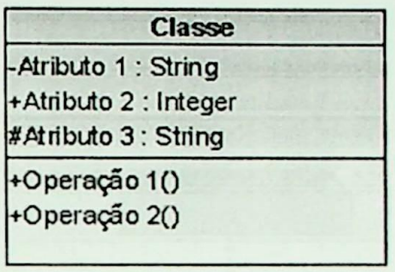

Figura 11. Classe UML.

O Diagrama de Classe é um dos mais fundamentais tipos de diagrama dentro da UML. É usado para capturar os relacionamentos estáticos de um software, ou seja, como os objetos se relacionam. Ao escrevê-lo, estaremos tomando constantemente decisões do projeto: que classes prendem referências a outras classes, que classe "possui" alguma outra classe, e assim por diante. Os Diagramas de Classe fornecem uma maneira para capturar esta estrutura "física" de um sistema (Pilone [2005]).

Uma classe possui atributos que representam os dados que aquela classe irá tratar, e possui também as operações que podem ser utilizadas com esses dados.

Os atributos devem ser descritos já com seus tipos de dados na seguinte formatação:

Permissão_do_atributo Nome_do_atributo : Tipo_do_atributo

Por exemplo:

+ Nome do Cliente : String

As permissões do atributo dizem respeito às permissões de acesso do atributo especificamente e podem ser:

- Pública, representada pelo símbolo +, um atributo público pode ser acessado por qualquer outra classe que necessite do acesso;

- Privada, representada pelo símbolo -, um atributo privado pode ser acessado somente pelas operações e objetos da sua própria classe.
- Protegida, representada pelo símbolo #, um atributo protegido pode ser acessado pelos objetos de sua classe e por todos os objetos das classes que têm ligação direta com essa classe.

Um Diagrama de Classe mostra todas as classes de um projeto e as possíveis relações entre elas, e todas as operações que cada classe pode realizar. A figura a seguir é um exemplo de classes, atributos, operações e relacionamentos entre classes:

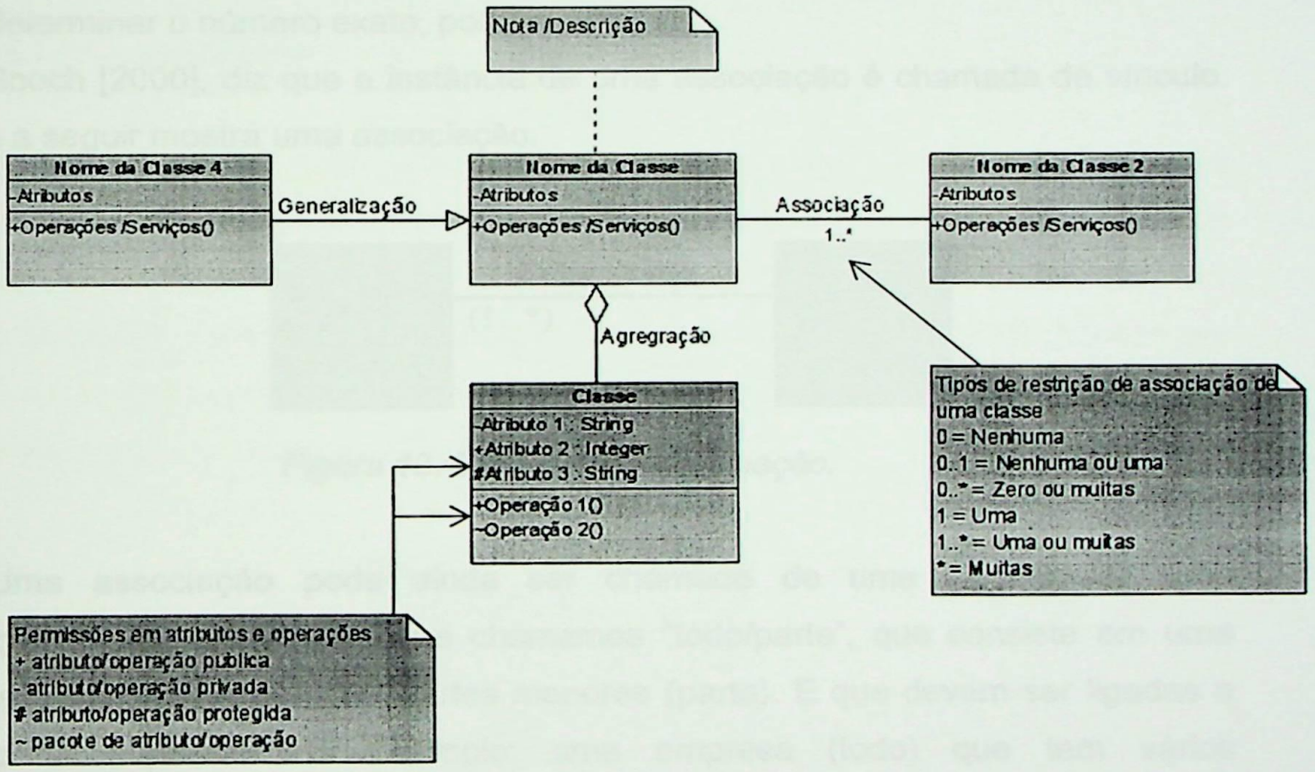

Figura 12. Exemplo de diagrama de classe.

2.2.2.3.4.1 Relacionamentos

Entre as classes quase sempre existirá um relacionamento, que pode ligar classes/objetos entre si criando relações lógicas entre estas entidades.

2.2.2.3.4.1.1 Associações

Representa a ligação (link) entre duas classes, associando objetos e permitindo que objetos de uma classe acessem objetos da outra e vice-versa, por exemplo, uma classe pessoa pode estar associada a uma classe empresa.

Uma associação representa um relacionamento estrutural existente entre dois objetos (Booch [2000]). Muitas vezes é necessário colocar restrições de acesso nessas associações determinando a quantidade de objetos de uma classe que pode esta relacionada com a outra. Essas restrições são definidas pelo número de objetos a relacionar e podem ser exatamente um (1), nenhum (0), zero ou um (0..1), zero ou muitos (0..*), um ou muitos (1..*) e simplesmente muitos (*), em alguns casos pode-se até determinar o número exato, por exemplo, (7).

Booch [2000], diz que a instância de uma associação é chamada de vínculo. A figura a seguir mostra uma associação.

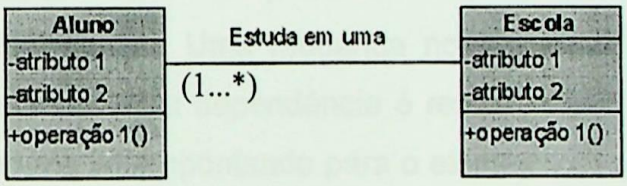

Figura 13. Exemplo de associação.

Uma associação pode ainda ser chamada de uma Agregação. Uma agregação é uma associação que chamamos "todo/parte", que consiste em uma classe superior (todo) que tem partes menores (parte). E que devem ser ligadas a essa parte superior. Por exemplo: uma empresa (todo) que tem vários departamentos distintos (partes); os departamentos são ligados à classe superior, no caso à empresa, e têm geralmente uma ligação um ou muitos (1..*), sendo que não poderá ser dos tipos que contém a opção de zero como ligação. A figura a seguir exemplifica uma agregação.

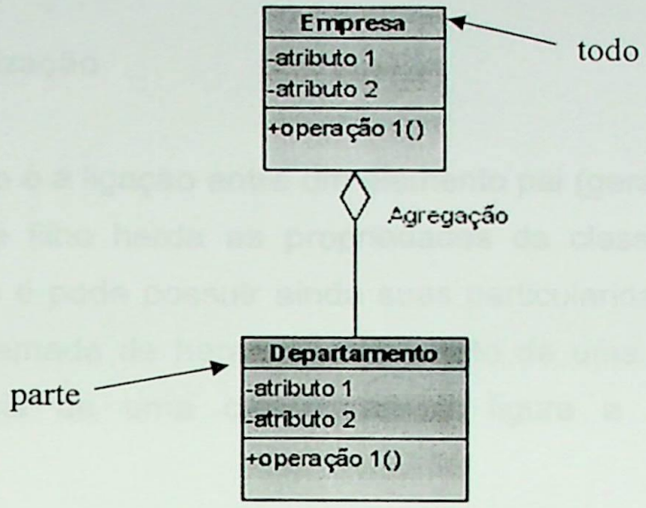

Figura 14. Exemplo de agregação

2.2.2.3.4.1.2 Dependência

A dependência é uma ligação simples entre duas classes causando na segunda uma dependência sobre a primeira. Teremos então uma classe independente e outra dependente. Uma mudança no elemento independente irá afetar o elemento dependente. Uma dependência é representada graficamente por uma linha tracejada com uma seta apontando para o elemento dependente. A figura a seguir exemplifica uma dependência.

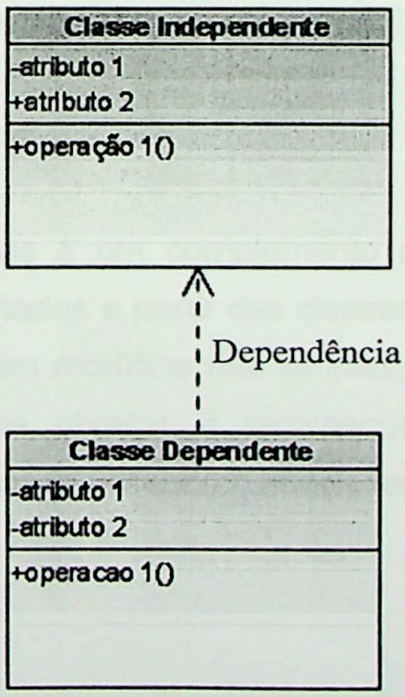

Figura 15. Exemplo de dependência.

2.2.2.3.4.1.3 Generalização

A generalização é a ligação entre um elemento pai (geral) e um elemento filho (específico). A classe filho herda as propriedades da classe pai, principalmente atributos e operações e pode possuir ainda suas particularidades. A generalização também pode ser chamada de herança. Um objeto de uma classe filho pode ser usado como instância de uma classe pai. A figura a seguir exemplifica a generalização.

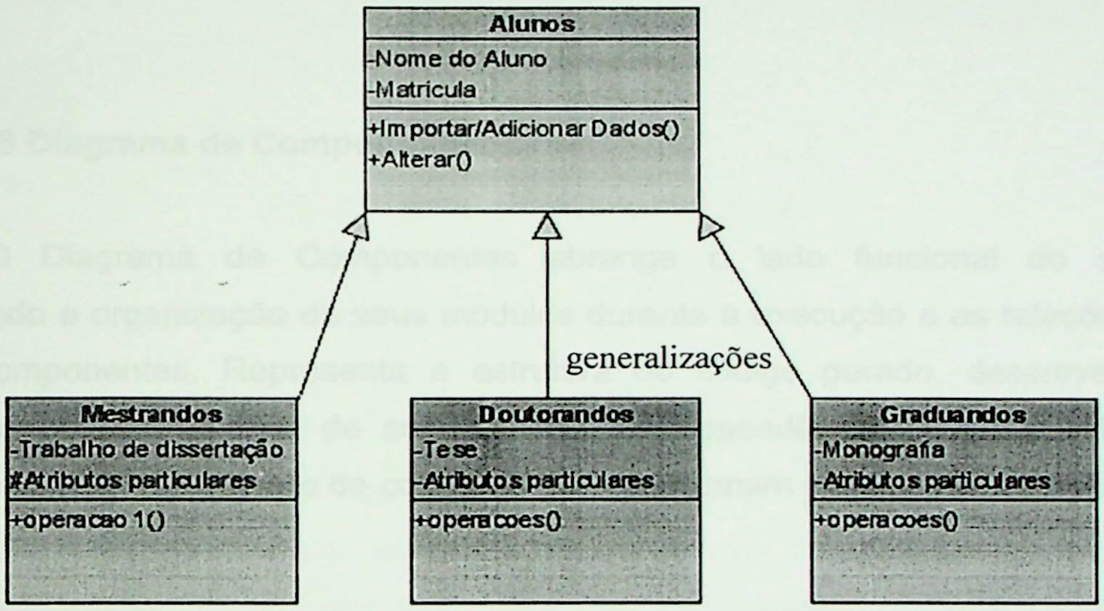

Figura 16. Exemplo de generalização.

2.2.2.3.5 Diagrama de Objetos

O Diagrama de Objetos é um complemento ao diagrama de classes. Ele apresenta todos os objetos criados a partir das classes e utiliza notação semelhante à do diagrama de classes, com modificações simples, como o sublinhado no nome dos objetos. O diagrama de objetos é recomendado para complementar um diagrama de classes complexas mostrando o relacionamento entre os objetos.

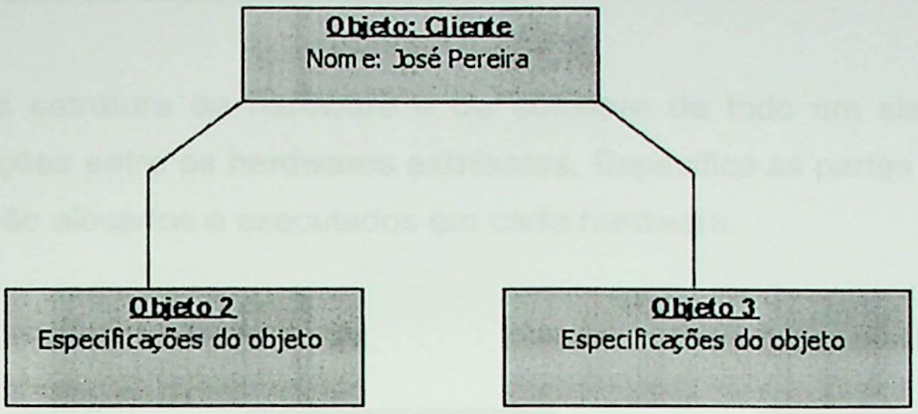

Figura 17. Diagrama de Objetos

2.2.2.3.6 Diagrama de Componentes

O Diagrama de Componentes abrange o lado funcional do sistema, mostrando a organização de seus módulos durante a execução e as relações entre seus componentes. Representa a estrutura do código gerado, descrevendo os componentes do sistema de software e suas dependências entre si. Mostra o relacionamento do conjunto de componentes que formam um sistema de software.

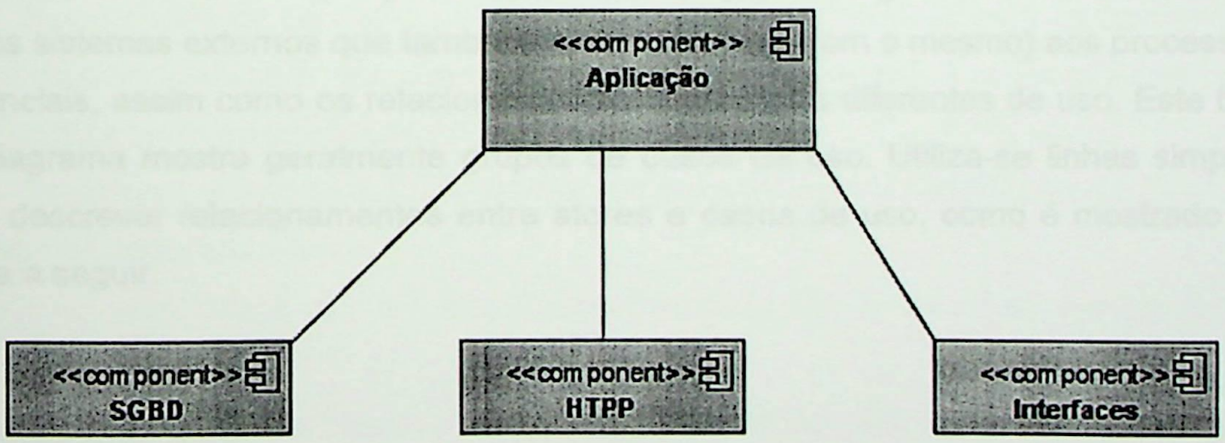

Figura 18. Diagrama de Componentes

2.2.2.3.7 Diagrama de Implantação/Execução.

Mostra a estrutura de hardware e de software de todo um sistema. Pode mostrar as ligações entre os hardwares existentes. Especifica as partes executáveis e objetos que são alocados e executados em cada hardware.

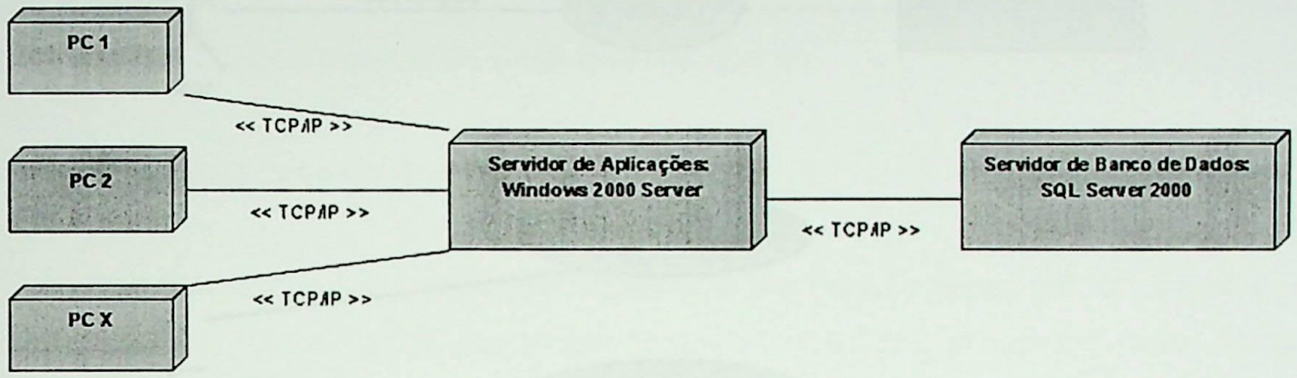

Figura 19. Diagrama de Execução

2.2.2.3.8 Diagrama de Caso de Uso

Um "Use Case" (Diagrama de Caso de Uso) modela a funcionalidade fornecida pelo sistema. A finalidade principal deste diagrama é ajudar as equipes de desenvolvimento a visualizar as exigências funcionais de um sistema, incluindo o relacionamento de "atores" (os seres humanos que interagirão com o sistema, ou outros sistemas externos que também podem interagir com o mesmo) aos processos essenciais, assim como os relacionamentos entre casos diferentes de uso. Este tipo de diagrama mostra geralmente grupos de casos de uso. Utiliza-se linhas simples para descrever relacionamentos entre atores e casos de uso, como é mostrado na figura a seguir.

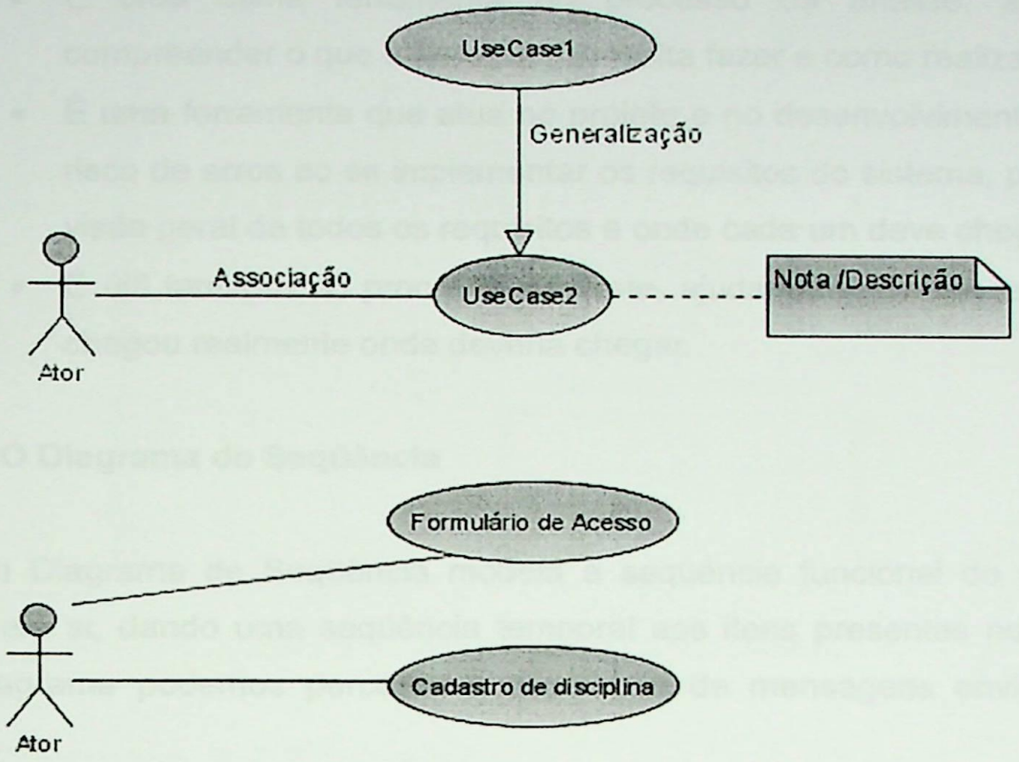

Figura 20. Diagrama "Use Case".

Em um projeto, cada participante tem a sua visão própria e para o sucesso deste, todos devem interagir e falar a mesma linguagem, focando o mesmo objetivo. A UML com o diagrama de caso de uso vem tentar exatamente isso, dar uma visão funcional do sistema como um todo e todas as interações que podem haver com ele.

Estes casos de uso são utilizados como principal artefato para estabelecer a funcionalidade desejada do sistema e realizar testes para a comunicação dos participantes além da validação final do mesmo.

Segundo Leffingwell [2003] alguns dos benefícios dos Diagramas de Caso de Uso são os descritos abaixo:

- Os Casos de Uso são relativamente fáceis de escrever e mais fáceis de ler;
- Forçam os desenvolvedores a pensar no projeto de um sistema da perspectiva de um usuário;
- Dão contexto para os requisitos do sistema. Pode-se compreender porque uma exigência existe e como o sistema chega aos seus objetivos;

- É uma ótima ferramenta no processo da análise, ajudando a compreender o que o sistema necessita fazer e como realizar isso.
- É uma ferramenta que atua no projeto e no desenvolvimento. Reduz o risco de erros ao se implementar os requisitos do sistema, por dar uma visão geral de todos os requisitos e onde cada um deve chegar.
- É útil também no processo de teste, ajudando a definir se o sistema chegou realmente onde deveria chegar.

2.2.2.3.9 O Diagrama de Seqüência

Um Diagrama de Seqüência modela a seqüência funcional do sistema de software em si, dando uma seqüência temporal aos itens presentes nele. Através desse diagrama podemos perceber a seqüência de mensagens enviadas entre objetos.

Segundo Pender [2003], os diagramas de seqüência são valiosos por que:

- Tem um foco estreito que ajuda a visualizar comandos e dados que estão sendo utilizados durante a execução de uma tarefa específica.
- Identifica explicitamente a comunicação requerida para cumprir uma interação. Isto ajuda a validar ou descobrir as relações requeridas por uma classe.
- Identifica explicitamente os objetos que participam em uma interação. Isto ajuda validar ou descobrir as características de uma classe.
- Identifica explicitamente os dados que são passados entre as partes das interações. Os dados têm que pertencer a uma classe em algum lugar no projeto.

Segundo Booch [2000], o Diagrama de Seqüência consiste em um número de objetos mostrados em linhas verticais. O decorrer do tempo é visualizado observando-se o diagrama no sentido vertical, de cima para baixo. As mensagens enviadas por cada objeto são simbolizadas por setas entre os objetos que se relacionam.

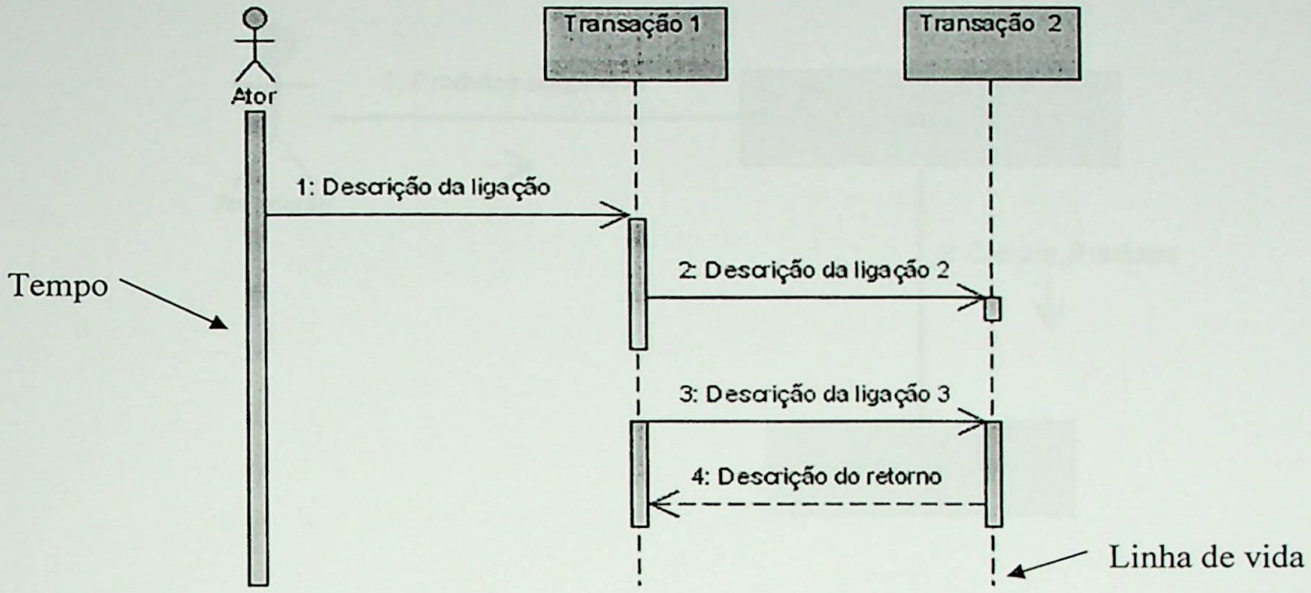

Figura 21. Diagrama de Seqüência

2.2.2.3.10 Diagrama de Colaboração

O Diagrama de Colaboração mostra a interação dinâmica entre objetos. Ele mostra a relação entre um conjunto de objetos, e a troca de mensagens existentes entre eles. O diagrama de colaboração pode ser utilizado para substituir o diagrama de seqüência.

Segundo Pender [2003], o diagrama de colaboração é quase o mesmo que o diagrama de seqüência. A diferença é a perspectiva. Ambos os diagramas modelam interações entre objetos para uma tarefa específica, mas enquanto o diagrama de seqüência enfatiza o arranjo na seqüência de interações sobre o tempo, o diagrama de colaboração modela como as interações utilizam a estrutura dos objetos e a participação em seus relacionamentos.

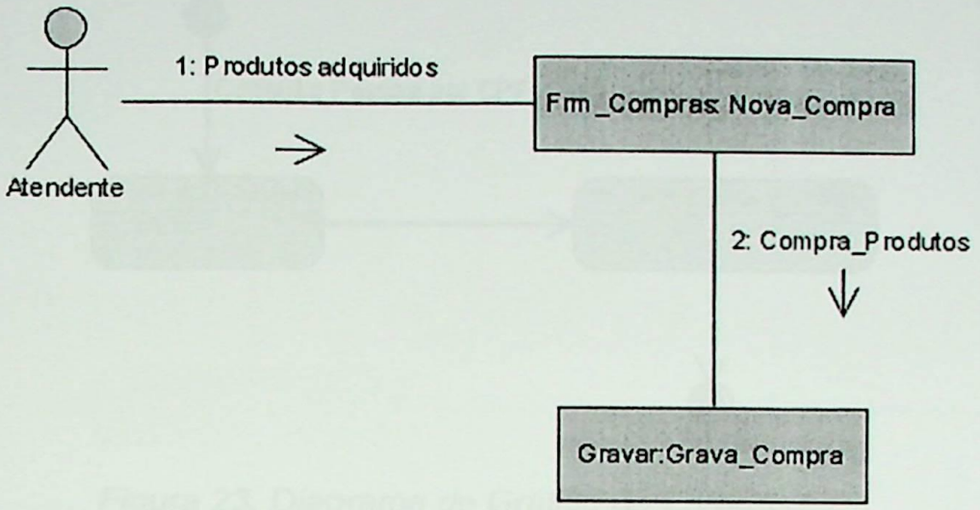

Figura 22. Diagrama de Colaboração

2.2.2.3.11 Diagrama de Gráfico de Estados

Um estado em um objeto é uma condição ou uma situação que pode ocorrer durante o tempo de vida de um objeto.

O Diagrama de Estados mostra todos os estados que um objeto pode passar durante seu tempo de vida e as respostas e ações que pode gerar com estas mudanças, e ainda quais eventos do sistema provocam essas mudanças. O Diagrama de Estados complementa um Diagrama de Classes com uma descrição mais detalhada de seus objetos. Ele mostra o que chamamos de máquina de estados, que podem ser estados, transições, eventos e atividades que ocorrem com os objetos.

Segundo Sommerville [2004], os objetos respondem aos diferentes pedidos de serviços e as transições de estado provocadas por estes pedidos.

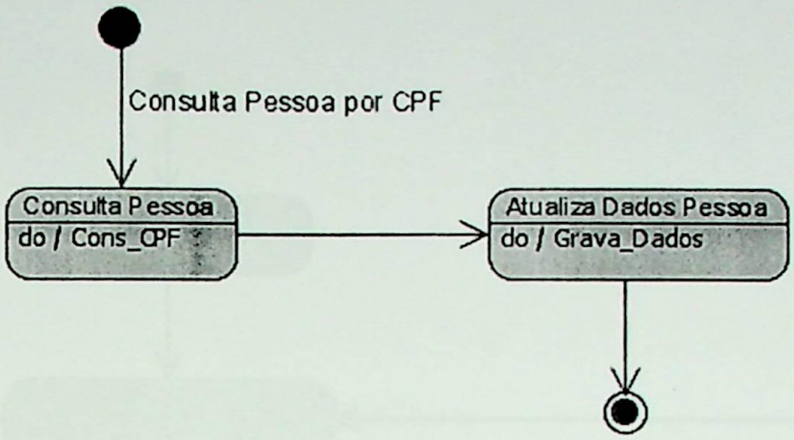

Figura 23. Diagrama de Gráfico de Estados

2.2.2.3.12 Diagrama de Atividades

O Diagrama de Atividades representa as ações (atividades) dos objetos e seus resultados. É uma variação do Diagrama de Estados, porém representa as ações e os resultados que ocorrem em determinada mudança de estado de um objeto. Mostra um conjunto de atividades, o fluxo de cada atividade para com uma outra, e os objetos que realizam ou sofrem ações.

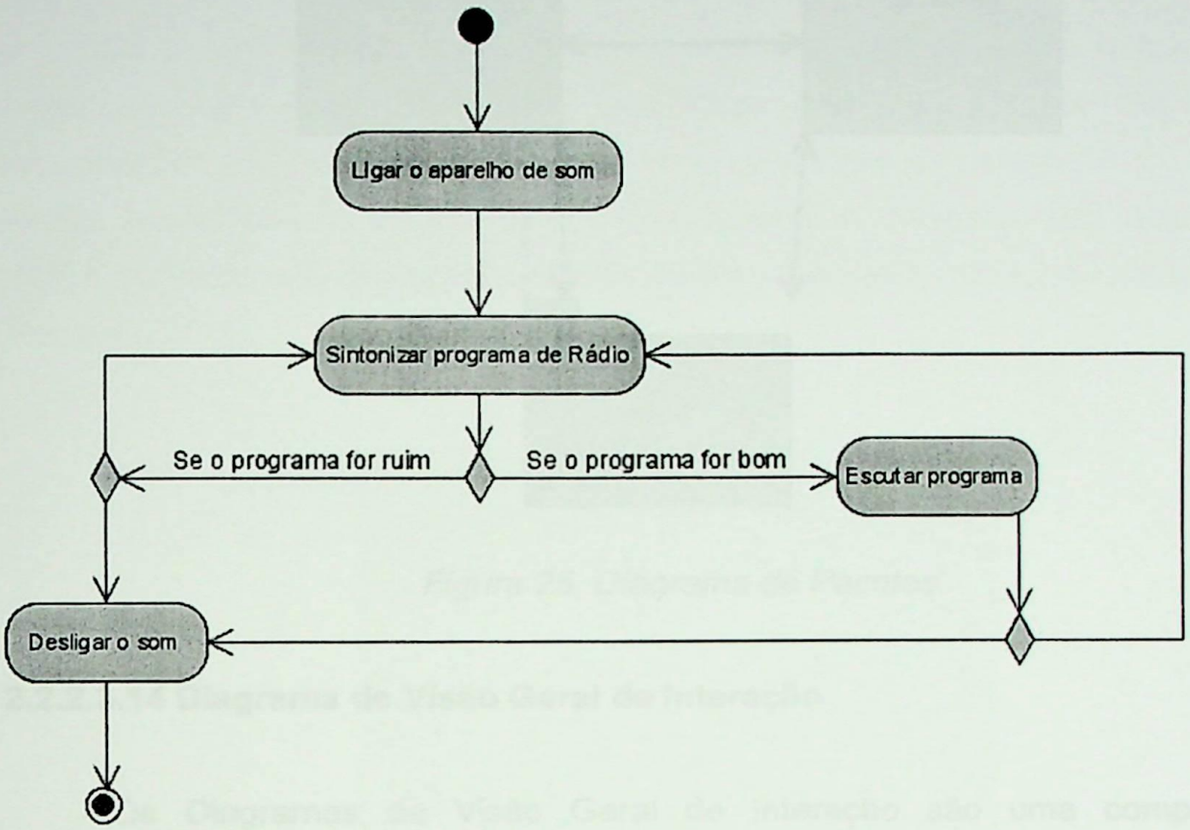

Figura 24. Diagrama de Atividades.

2.2.2.3.13 Diagrama de Pacotes

Os Diagramas de Pacotes representam os pacotes do sistema com ou sem seus constituintes. Podem também representar a dependência entre pacotes ou subsistemas (Pender [2003]).

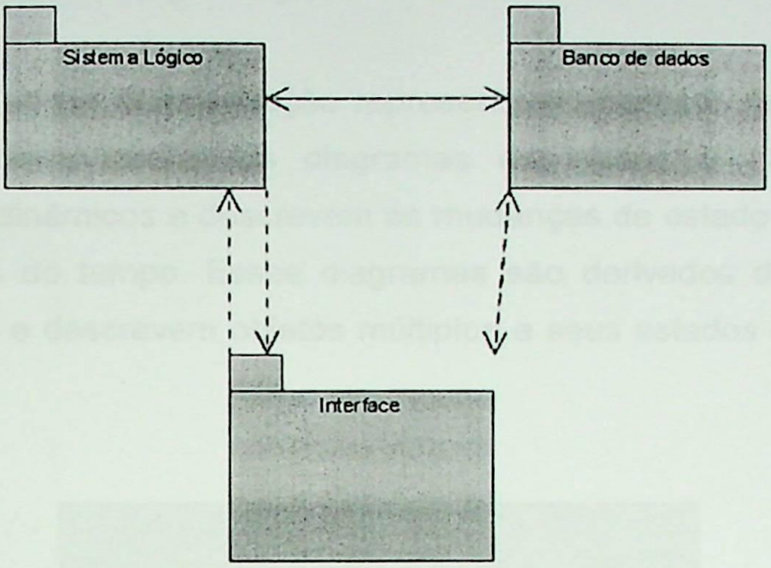

Figura 25. Diagrama de Pacotes.

2.2.2.3.14 Diagrama de Visão Geral de Interação

Os Diagramas de Visão Geral de Interação são uma composição de diagramas de atividades e seqüência, nos quais as atividades são substituídas por pequenos diagramas de seqüência (Pender [2003]).

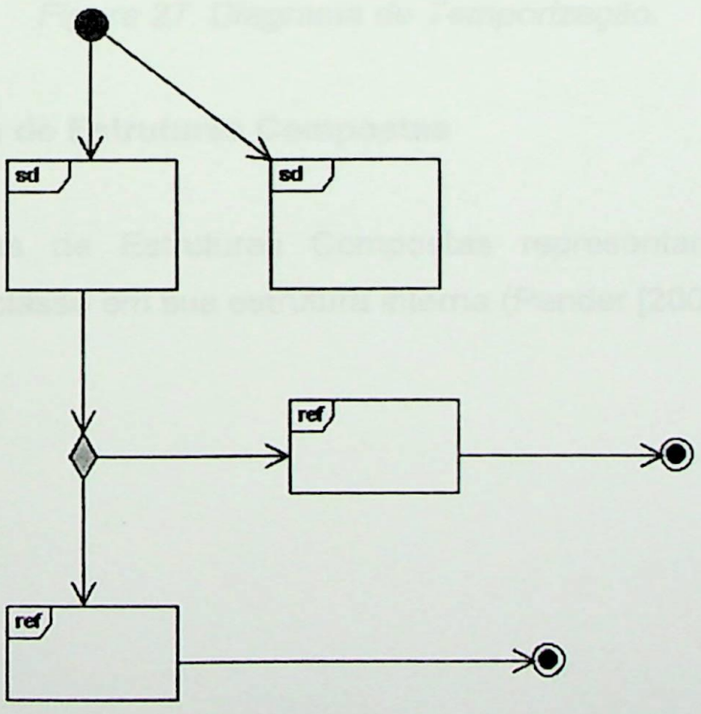

Figura 26. Diagrama de Visão Geral de Interação.

2.2.2.3.15 Diagrama de Temporização

Os Diagramas de Temporização representam restrições de temporização e constituem uma outra forma dos diagramas de interação. Os diagramas de temporização são dinâmicos e descrevem as mudanças de estado de um objeto em pontos específicos do tempo. Esses diagramas são derivados dos diagramas de gráfico de estados e descrevem objetos múltiplos e seus estados ao mesmo tempo (Pender [2003]).

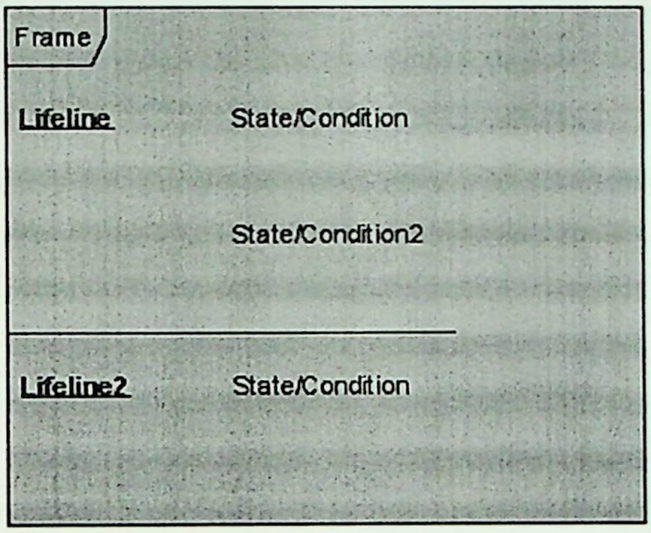

Figura 27. Diagrama de Temporização.

2.2.2.3.16 Diagrama de Estruturas Compostas

Os Diagramas de Estruturas Compostas representam a decomposição hierárquica de uma classe em sua estrutura interna (Pender [2003]).

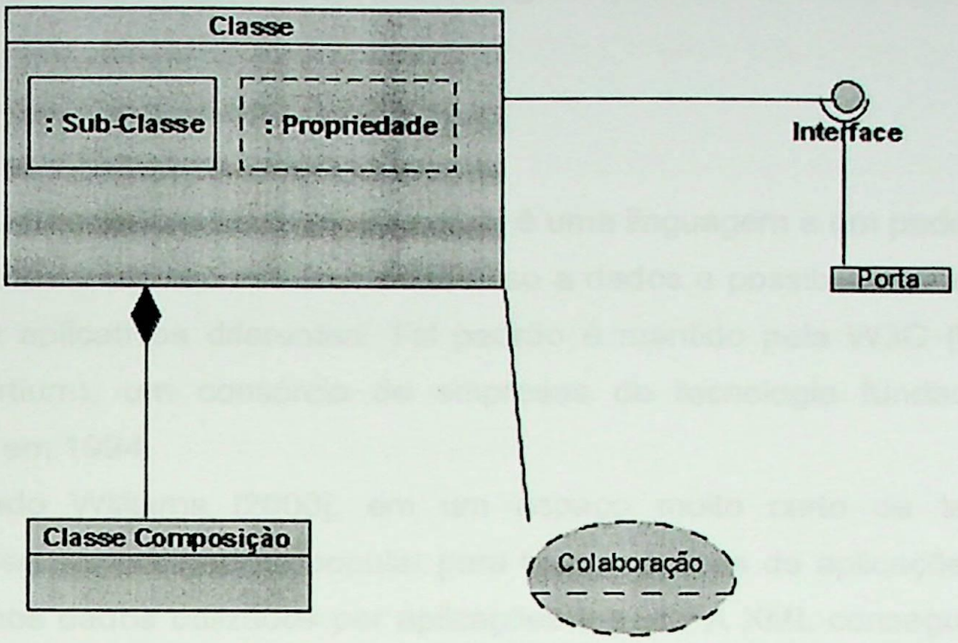

Figura 28. Diagrama de Estruturas Compostas.

3-TECNOLOGIAS E PADRÕES UTILIZADOS

3.1 XML

A XML (Extensible Markup Language) é uma linguagem e um padrão inovador feito para oferecer um recurso leve de acesso a dados e possibilitar a transferência destes entre aplicativos diferentes. Tal padrão é mantido pela W3C (World Wide Web Consortium), um consórcio de empresas de tecnologia fundada por Tim Berners Lee em 1994.

Segundo Williams [2000], em um espaço muito curto de tempo, XML transformou-se em um padrão popular para todos os tipos de aplicações desde os dados web aos dados utilizados por aplicações locais. A XML conseguiu abranger todos os requisitos para um bom desenvolvimento: armazenamento, transporte e exposição dos dados. A XML está sendo usada por programadores do mundo todo nas mais diversas linguagens de programação.

Os arquivos XML têm uma estrutura hierárquica em forma de árvore, e identada o que facilita a visualização do arquivo. Os elementos são separados por tags. As tags são definidas entre os sinais de menor (<) e maior (>). Tags são representações de elementos XML e podem conter ainda alguns atributos (Birbeck [2001]).

A tabela da direita abaixo exemplifica um arquivo XML

1	<?xml version="1.0" encoding="utf-8" ?>
2	<dados>
3	<aluno id= "1" matriculado= "sim" >
4	<nome>Jose da Silva</nome>
5	<curso>Mestrado</curso>
6	</aluno>
7	</dados>

Tabela 1. Exemplo arquivo XML.

Na linha número 1 temos o prólogo do arquivo XML. Ele é de uso obrigatório e contêm informações sobre a versão do arquivo XML e a codificação de idioma utilizada no mesmo.

Na linha número 2 está o elemento raiz, também de uso obrigatório. Ele define o nome da estrutura de dados. Deve existir somente um elemento raiz em um documento XML.

Na linha 3 temos um elemento filho. Esse elemento possui dois atributos, sendo eles "id" e "matriculado". O elemento filho possui ainda mais dois elementos filhos "nome" e "curso" nas linhas 4 e 5 respectivamente.

Na linha 6 temos o item que finaliza o elemento filho "aluno". Quando abrimos um elemento temos obrigatoriamente que fechá-lo. Para isto basta escrever uma tag com uma barra (/) antes do nome do elemento.

Na linha 7 fechamos o elemento raiz, indicando que o arquivo XML chegou ao fim.

Podemos definir um valor padrão para os atributos, o que não ocorre com os elementos. Já os atributos não podem ser repetidos, dispostos em ordem, indexados ou possuir filhos, ao passo que os elementos podem.

Os elementos em um documento XML devem ser aninhados de forma correta e todos devem ser devidamente encerrados. Já em relação aos atributos, precisamos nos certificar de que um mesmo atributo não seja definido em um elemento mais de uma vez e que seus valores sejam delimitados por aspas. Os elementos possuem uma característica denominada "case sensitive". Essa característica define que seus elementos diferenciam maiúsculas de minúsculas. Então, em uma tag aberta com letra maiúscula, deve ser fechada com letra maiúscula obrigatoriamente.

Um documento XML pode conter associado a ele um esquema de apresentação e/ou definição dos dados. Este esquema é chamado de XSD (XML Schema Definition). Os arquivos XSD nada mais são do que outros arquivos XML com estrutura de definição de dados.

3.1.1 Vantagens do XML

Os documentos XML possuem várias vantagens que o tornaram um padrão conhecido e utilizado mundialmente. Entre elas a de serem uma das melhores formas para troca de informações entre aplicações. Citamos ainda outras vantagens:

- São de fácil interpretação e manipulação por serem geralmente autodescritivos.

- Permitem a criação de estruturas complexas como árvores de profundidade arbitrária, apesar de sua simplicidade.
- São flexíveis, permitindo a representação de quaisquer dados estruturados ou semi-estruturados.
- Recorrendo a um esquema, é possível verificar suas validações.
- São facilmente manipuláveis através de várias ferramentas existentes no mercado hoje em dia.
- São um padrão aberto, independente de aplicações e sistemas operacionais, facilitando a integração de sistemas heterogêneos.

Segundo Harold [2001], XML é ideal para grandes e complexos documentos porque os dados são estruturados e por deixar também especificar as relações entre elementos. Porém pode ser utilizado em sistemas de qualquer complexidade.

Segundo Deitel [2000], o XML embora tenha muitas potencialidades avançadas, é acessível a todos os níveis de programadores por causa de sua simplicidade inerente. É baseado em texto. Assim qualquer um pode criar um arquivo XML tanto com ferramentas especificas quanto com os mais primitivos processadores de texto. Entretanto, XML não é limitado a descrever somente dados textuais, mas também pode descrever imagens, gráficos de vetor, animações ou algum outro tipo de dados que for proposto. XML é um padrão aberto. Há uma ampla gama de ferramentas capazes de executá-lo. Um usuário pode escolher a que se ajusta às suas necessidades.

3.1.2 Simple API for XML (SAX)

Para a manipulação de arquivos XML utilizamos o SAX (Simple API for XML), uma API (Application Programming Interface) que manipula os arquivos XML de forma simples. Ela possui o que chamamos de "parser". Um parser é um leitor que ajuda na conversão do arquivo para manipulação dos dados contidos no mesmo. O parser é composto de vários eventos que possibilitam a manipulação de um arquivo XML, tais como início e fim do documento, início e fim do elemento, nó de comentário, entre outros vários disponíveis através do SAX.

Existe um outro parser para arquivos XML chamado DOM (Document Object Model). O DOM por sua vez carrega todo o documento XML na memória para uso posterior.

Benz [2003] diz que SAX é muito bom para utilizar com um grande arquivo XML. SAX lê sequencialmente o arquivo uma única vez para coletar dados, não armazenando os mesmos, e caso necessite de um novo dado, todo processo de leitura terá que ser reiniciado. DOM, ao contrário, armazena uma árvore com os dados do arquivo na memória até que a aplicação elimine-a. Assim, uma vez analisado, as partes do arquivo podem ser recuperadas sem ter que reanalisá-lo. Referente a qual é melhor para uma aplicação específica, a escolha pode variar, dependendo dos dados com que se está trabalhando. Se você estiver trabalhando com um arquivo grande de dados estruturados, como a saída do XML pode ser uma lista com centenas de milhares de registros, SAX é provavelmente o método parser para se tentar utilizar primeiramente, para evitar uso excessivo de memória para armazenar, por um longo período, dados com o DOM.

O SAX nada mais é do que um conjunto de rotinas e padrões estabelecidos para se acessar um arquivo XML e permitir de forma simples várias funcionalidades. O SAX é baseado em eventos e funciona de forma seqüencial percorrendo todos os elementos (nós) de uma árvore XML. Os eventos são agrupados em "Handlers" e os principais do SAX são ContentHandler e ErrorHandler.

Dentro do ContentHandler de um SAX temos vários eventos. Citamos os principais abaixo:

1. startDocument – Quando a leitura do documento XML é iniciada, este evento é disparado. Este evento ocorre uma única vez durante a leitura do documento XML e é o primeiro evento a ocorrer.
2. startElement – Esse evento ocorre todas as vezes que um novo elemento é encontrado em um documento XML, não importando se é um elemento pai ou elemento filho. Como a leitura do SAX é seqüencial esse evento ocorrerá sucessivas vezes, e pode receber elementos e atributos como parâmetros. O nome do elemento seria o primeiro parâmetro e o(s) atributo(s) se existir(em) o segundo parâmetro, agrupados entre colchetes, como no exemplo a seguir:

   startElement ("Elemento", [Attribute ("atributo1", "valor_atributo1")])

3. characters – Este evento ocorre quando é encontrado, após um elemento, uma seqüência de caracteres. Ele simplesmente armazena em uma variável temporária essa seqüência.
4. endElement – Evento que ocorre quando um elemento é finalizado. Tem como parâmetro o nome do elemento que está sendo finalizado.
5. endDocument – Evento que ocorre quando o último elemento é finalizado e o final do arquivo é encontrado. É o ultimo evento a ocorrer.

1	<?xml version="1.0" encoding="utf-8" ?>
2	<mestrado>
3	<aluno id="1">João Pereira
4	</aluno>
5	</mestrado>

Tabela 2. Exemplo de parte de um arquivo XML

Teríamos com o SAX os seguintes eventos para leitura do documento acima:

startDocument ()
startElement ("mestrado", [ ])
startElement ("aluno", [Attribute ("id","1")])
characters ("João Pereira")
endElement ("aluno")
endElement ("mestrado")
endDocument ()

Dentro do errorHandler, temos os eventos de erros que podem ocorrer ao fazermos a leitura de um arquivo XML. Temos por exemplo o "fatalerror", que é o evento acionado quando ocorre um erro que não pode ser tratado e a leitura será interrompida. Temos também o "ignorableWarning", que ocorre somente quando um evento pode ocasionar um provável erro, não implicando que o erro realmente ocorrerá. O evento "ignorableWarning" não para a leitura do documento XML.

3.2 RTF (Rich Text Format)

O RTF é um formato simples que oferece suporte a formatações de texto e criação de tabelas em arquivos textos. Ele tem a grande vantagem de ser suportado pela maioria dos editores de texto no mercado hoje em dia. O RTF é uma linguagem de marcação utilizada, por exemplo, pelo MS Word e compatível com dezenas de processadores de texto. RTF é um formato universal que está presente em qualquer sistema operacional e por ser texto, é muito mais fácil de ser gerado e processado do que arquivos binários.

O padrão RTF encontra-se atualmente em sua versão 1.8, versão mantida pela Microsoft. O RTF oferece formatações ricas de texto com possibilidades de inserção de gráficos, figuras e tabelas em um documento de texto. O RTF utiliza geralmente o padrão ASCII como base para suas formatações de texto, possuindo palavras, símbolos e grupos de controle de texto.

Uma palavra de controle em RTF é um comando destinado a formatar uma determinada parte do texto, de forma a exibir ou imprimir esse texto no formato desejado. As palavras de controle em RTF têm no máximo 32 caracteres de extensão e são padronizadas de acordo com cada versão. As palavras de controle são precedidas por uma barra ("\"), e têm como delimitadores de espaço, traço ("-"), outra barra ou um número. O padrão RTF é "CaseSensitive", ou seja, faz diferenciação entre minúsculo e maiúsculo em seus caracteres. É um exemplo de palavra de controle "\clFitText". A palavra de controle "\ansicpg", por exemplo, é a palavra que define a página de códigos utilizada pelo documento. A palavra que representa português é "\ansicpg860".

Além das palavras de controle temos ainda os símbolos de controle, que são caracteres simples para controlar uma função do RTF. Possuem a mesma formatação de uma palavra de controle começando com barra ("\") e possuindo seus delimitadores posteriormente. Um exemplo de símbolo de controle é o "\b" que transforma o texto a seguir em negrito e o "\b0" é o delimitador deste símbolo.

Podemos ter também grupos de controle. Um grupo consiste em textos e palavras controle ou em símbolos de controle incluídos entre chaves "{ }". A chave de abertura "{" indica o começo do grupo e a chave de fechamento "}" indica a extremidade do grupo. Cada grupo especifica o texto afetado pelo grupo e pelos atributos diferentes deste texto. Temos como exemplo o grupo

{\b \red0\green0\blue0; Texto formatado. \b0}

Através então de grupos, palavras e símbolos de controle podemos gerar um documento de texto que pode ser exibido corretamente formatado em um editor de texto qualquer, que suporte a versão do RTF utilizado. Basta gerarmos um arquivo de texto com os controles desejados e salvá-los com a extensão RTF.

3.3 Visual Basic

Ferramenta de desenvolvimento produzida pela Microsoft atualmente em sua versão 7, o Visual Basic, conhecido simplesmente como VB, é uma ferramenta de desenvolvimento rápido de aplicações. É utilizada inclusive para a produção de protótipos de desenvolvimento de sistemas de software. É um IDE (Integrated Development Environment) poderoso. Até sua versão 6, o VB não era considerado por muitos um IDE orientado a objetos, porém a versão 7 (conhecida como Vb.Net) já oferece todos os recursos de orientação a objetos.

"Visual Basic não é apenas uma linguagem. É um ambiente de desenvolvimento integrado no qual você pode desenvolver, executar, testar e depurar seus aplicativos." (Petroutsos [1999])

Segundo Reed [2000], o VB é um IDE que tem várias características orientadas a objetos. Muitos na indústria chamam produtos como o VB, baseado em objeto, e não orientado a objeto. Reed na frase anterior se referia a versão 6 do VB, versão essa utilizada em nosso projeto. Vários escritores debatem sobre este assunto de uma linguagem ser ou não orientada a objetos e não há uma definição global precisa e aceita por todos. Existem definições de um conjunto de itens que uma linguagem orientada a objetos deve conter, itens como classes, herança, encapsulamento entre outros.

Segundo Reed [2000], a aceitação do Visual Basic no mercado se deu pelo acréscimo de novos recursos e capacidades. Podemos citar alguns aspectos de amadurecimento:

- Introdução do Component Object Model (COM) que permite trabalhar tanto com objetos distribuídos como orientados a objetos.
- Microsoft Transaction Server (MTS) e o Microsoft Message Queue (MSMQ) que permite suporte de comunicação e gerenciamento de recursos.

- Uso do Visual Basic for Applications (VBA) como linguagem de integração com vários softwares, como por exemplo, Word, Excel, Access e Power Point.
- Licença do VBA para ser utilizados em vários aplicativos como Corel, Visio e AutoCAD.
- Uso de VBScript em aplicativos para intranet e internet.
- Oferta de três diferentes versões (edições) com recursos e preços distintos. Segundo Holzner [1998]:

    - Visual Basic Learning edition: é introdutório e permite criar aplicativos facilmente com alguns dos componentes básicos; edição de estudante.
    - Visual Basic Professional edition: possui recursos mais avançados do que o anterior; permite criar controles para internet e ActiveX. Mas ainda com algumas limitações.
    - Visual Basic Enterprise edition: é o mais avançado e permite criar aplicativos distribuídos em um ambiente de equipe e é a versão completa do VB.

O VB oferece recursos para desenvolvimento de aplicativos locais e aplicativos integrados. Ele tem suporte a várias tecnologias do mercado, como o XML, e pode trabalhar com o mesmo de forma clara. O Visual Basic oferece também um fácil acesso e controle de banco de dados, podendo acessar a maioria dos bancos de dados no mercado hoje em dia.

3.4 MS Access

O Access é um SGBD (Sistema Gerenciador de Banco de Dados) local desenvolvido e mantido pela Microsoft. O Access é de fácil acesso através do Visual Basic, não precisando nem mesmo estar instalado na máquina para ser acessado. Basta termos o arquivo do banco de dados gerado em Access e o Visual Basic conseguirá acessá-lo de forma simples e clara.

Um banco de dados é um conjunto de informações armazenadas de forma organizada, podendo estar relacionadas, de modo a oferecer um fácil processamento (inclusão, exclusão e alteração) e recuperação dos dados. Prague [2003] diz que um banco de dados é um termo da computação utilizado para uma coleção de dados a respeito de um determinado tópico ou aplicação de negócio. As

bases de dados ajudam a organizar estas informações relacionadas em uma forma lógica para o acesso e a recuperação fáceis.

O Access é um banco de dados relacional que oferece recursos de relação entre tabelas e ligação dos dados. Trabalha com Sql (Structured Query Language) que é a linguagem padrão da maioria dos bancos de dados no mercado hoje em dia. Através dessa linguagem podemos alterar, adicionar, excluir e consultar dados em um banco de dados.

4- SISTEMA PROPOSTO

O sistema proposto consiste em um Sistema de Software para auxílio dos docentes da Unifei (Universidade Federal de Itajubá). Ele irá gerar relatórios de progressão dos docentes da instituição segundo a base de dados do currículo Lattes e complementar com sua base própria de dados. O Currículo Lattes é um sistema de software do CNPq (Conselho Nacional de Desenvolvimento Cientifico e Tecnológico) que contém todos os dados dos docentes cadastrados. Neste capítulo, serão apresentados o sistema proposto, as etapas vencidas, as tecnologias utilizadas em cada uma delas, o processo de desenvolvimento utilizado (prototipação), a linguagem de modelagem utilizada (UML), o banco de dados utiliizado (Ms Access) e os resultados obtidos. Foi utilizada a prototipação por ser um desenvolvimento individual e evolucionário e foi utilizado o Access por ser um banco de dados de fácil integração com o Visual Basic (também utilizado nesse projeto).

4.1 Análise dos requisitos

A primeira etapa foi uma entrevista, feita com o Professor Otávio A. S. Carpinteiro, sobre o sistema utilizado como fornecedor de dados, o Currículo Lattes. Este sistema possui os dados dos docentes e é utilizado pela Capes, CNPq e outras entidades. Tal sistema possui uma exportação de dados através da XML, utilizada para fornecer dados para o Sistema de Software para auxílio dos docentes da Unifei, que é o foco deste trabalho. Foi esclarecido nessa entrevista inicial como o Currículo Lattes funciona, onde estão cada uma das informações especificamente e como gerar o arquivo XML, para utilização posterior.

A análise dos requisitos foi feita com base em documentos e nos modelos de relatório de progressão, que os docentes têm que preencher e entregar nos departamentos corretos da instituição.

4.2 Proposta de desenvolvimento do sistema de software

Com base na análise dos requisitos e no sistema de software Currículo Lattes podemos então capturar informações sobre o sistema atual de emissão dos relatórios periódicos e elaborar uma proposta para o novo sistema.

4.2.1 Descrição do sistema atual

Os docentes da Unifei já possuem cadastrados no sistema Currículo Lattes o seu currículo e todas as atividades realizadas com relação à instituição. Mas este sistema de software somente emite um relatório padrão.

Atualmente a CPPD (Comissão Permanente de Pessoal Docente) solicita relatórios de produção dos docentes em um formato específico. Então surge o seguinte problema: duplicidade de dados ao preencher o sistema de software Currículo Lattes e também os relatórios da CPPD com praticamente os mesmos dados. Isto pode causar vários outros problemas.

É um serviço árduo preencher o sistema Currículo Lattes e ficar alternando entre telas e colando informações do sistema para um relatório do Microsoft Word, pois os campos estão em caixas de textos distintas dentro do sistema. Por exemplo: o título de uma publicação está em uma caixa, já a editora, o ano de publicação e outros dados podem estar em caixas de texto distintas e inclusive em telas ou abas diferentes, ocasionando assim uma maior alternância de telas. A duplicidade de dados de tal forma pode causar vários erros, entre eles, pular alguns campos, copiar várias vezes a mesma informação, não saber qual informação copiar ou onde copiar tal informação. Preencher todo o relatório de progressão para entrega na Unifei sem ter uma base de consulta não seria muito recomendado, pois pode ocorrer esquecimento e ainda informações erradas.

Para preenchimento do sistema de software Currículo Lattes temos toda uma seqüência lógica, que deve ser seguida na primeira vez que o sistema de software é preenchido, posteriormente precisamos somente atualizar ou adicionar mais informações nesse software.

O Currículo Lattes oferece cadastro via web ou através de um sistema de software instalado localmente e com transmissão de dados para a web. Estes dados podem ser exportados do sistema através da XML e podem também serem gravados em um arquivo localmente.

Os relatórios de progressão preenchidos pelos docentes permitem que a instituição (Unifei) faça um acompanhamento quanto à evolução e o plano de carreira dos mesmos. Tais relatórios representam um número específico de pontos conquistados com cada atividade e com ele pode-se progredir na carreira de docente.

4.2.2 Problemas observados

Os principais problemas apresentados pelo atual sistema são:

- Duplicidade de informações;
- Dificuldade no preenchimento de dois sistemas;
- Possível perda ou esquecimento de informações;
- O sistema Currículo Lattes não emite o relatório no formato que a instituição adotou;
- Preencher o relatório de progressão com todas as publicações, atividades administrativas, orientações, participações em bancas entre outras, levará uma grande quantidade de tempo, que poderia ser empregado em outras atividades acadêmicas.

4.3 Sistema Proposto

O sistema proposto visa fazer uma interação com o arquivo XML gerado pelo Currículo Lattes e utilizar o mesmo para gerar os relatórios exigidos pela Unifei.

4.3.1 Objetivo

O objetivo do nosso sistema é realizar a tarefa, ou pelo menos, boa parte dela, no preenchimento dos relatórios de progressão do docente Unifei, através dos dados XML exportados do sistema de Currículo Lattes. Sendo para tal objetivo executadas as seguintes tarefas:

- Importação dos dados XML gerados pelo sistema Currículo Lattes;
- Complementação das informações através de cadastro em banco de dados próprio do número de matrícula dos alunos, o qual não consta na base de dados do Currículo Lattes;
- Complementação das informações através de cadastro em banco de dados próprio do código, as horas/aulas e quantidade de alunos das disciplinas lecionadas, os quais não constam na base de dados do Currículo Lattes;

- Gerar os relatórios da CPPD e da GED (Gratificação de Estímulo a Docência) por um período especificado pelo docente.

4.3.2 Justificativas

Atualmente com o avanço da computação e das tecnologias, não há porque preencher um sistema com todos os dados pertinentes e depois refazer o serviço preenchendo novos relatórios. Isto é inconcebível.

4.3.3 Premissas e restrições

Este projeto está sendo desenvolvido para fins acadêmicos, sem fins lucrativos. O desenvolvimento abrangerá recursos humanos, de sistema de software e de hardware limitados sob as seguintes condições:

- Sistema operacional Windows 98 ou superior.
- Linguagem de codificação Visual Basic 6.0.
- Banco de dados Access 97.
- A modelagem do sistema será feita utilizando a tecnologia UML.
- Os dados serão importados utilizando tecnologia XML.
- Os relatórios serão gerados utilizando o formato RTF (Rich Text Format).

4.3.4 Benefícios

O sistema oferecerá como beneficio a geração automática dos relatórios da CPPD e da GED, os quais eram anteriormente gerados manualmente.

Oferecerá agilidade, rapidez e confiança na geração dos relatórios e eliminará redundância e erros de informações nos relatórios.

4.3.5 Alternativas de Implementação

4.3.5.1 Marcos, Pontos de Controle

- Estudo inicial – Estudo de viabilidade – Coleta de informações.
- Elaboração da proposta de desenvolvimento.

- Modelagem do sistema.
- Definição das premissas e dos modos de acesso aos dados do sistema Currículo Lattes.
- Implementação do banco de dados que receberá e armazenará algumas informações oriundas do Currículo Lattes e outras complementares adicionadas pelo usuário.
- Implementação da integração com o XML.
- Implementação do modelo de sistema de software (interface).
- Implementação do modelo de relatório para a CPPD.
- Implementação do modelo de relatório para a GED.
- Testes e correções de erros.
- Testes finais.

4.3.6 Relação do Sistema com o Ambiente

4.3.6.1 Singularidade

A solução apresentada, por ser específica para os relatórios utilizados pela Unifei, não está disponível para venda ou aquisição de qualquer forma.

4.3.6.2 Integrabilidade

Este projeto tem integrabilidade com o sistema de software Currículo Lattes do CNPq através da tecnologia XML.

4.3.7 Análises de Requisitos do Sistema de Software

As atividades realizadas nesta fase são:

- Determinação do contexto.
- Definição do escopo.
- Definição dos requisitos funcionais.
- Definição dos atores e criação dos use cases.
- Levantamento das classes candidatas.

- Especificação do layout sugerido.
- Diagrama de seqüência de atividades

4.3.7.1 Modelo Descritivo

Após a entrevista e a elaboração da Proposta de desenvolvimento, foi criada a primeira modelagem do sistema, criados os diagramas de uso de caso, diagrama de classes e o diagrama de seqüência, sendo desenvolvido em cima destes dados o primeiro modelo do sistema.

4.3.7.1.1 Funcionamento do Sistema de Informação dos Docentes

O Sistema deverá permitir importar os dados do arquivo XML gerado pelo Currículo Lattes, tratar estes dados para uso posterior, e ainda:

- Buscar as informações de todas as disciplinas lecionadas pelo professor em questão, separando-as por ano e por curso, para possível cadastro de número de alunos em determinado ano, número de horas/aula e o código da disciplina. O preenchimento dos dados das disciplinas não é obrigatório, pois estes não constam no Currículo Lattes.
- Buscar informações a respeito dos alunos do docente em questão para preenchimento do número de matrícula. O sistema necessita somente dos alunos que foram orientados no período específico e o preenchimento não é obrigatório; os dados de matrícula dos alunos não constam na base de dados do Currículo Lattes.
- Seguindo os modelos de relatórios fornecidos pela Instituição, mesclar os dados do XML com os dados complementares da base de dados própria (disciplinas e alunos) e gerar o relatório com todos os dados possíveis para o docente.
- Visualizar os relatórios gerados em formato RTF.

4.3.8 Modelagem do sistema

4.3.8.1 Diagrama Use Case

O sistema é de fácil interação com o usuário e tem ainda uma interação com o sistema Currículo Lattes. Estas interações são apresentadas no Diagrama de Caso de Uso a seguir.

A primeira interação do ator principal do sistema (Docente) é com Currículo Lattes para solicitar a exportação de dados no formato XML. Após gerar esse documento, o docente precisa especificar qual arquivo XML será utilizado no nosso sistema, fazendo assim a importação dos dados do arquivo XML para uso posterior. A próxima etapa é de caráter opcional, onde os dados dos alunos podem ser preenchidos diretamente no sistema para cruzamento dos dados com o arquivo XML. O preenchimento das disciplinas também é opcional e supre as informações que o arquivo XML não contém.

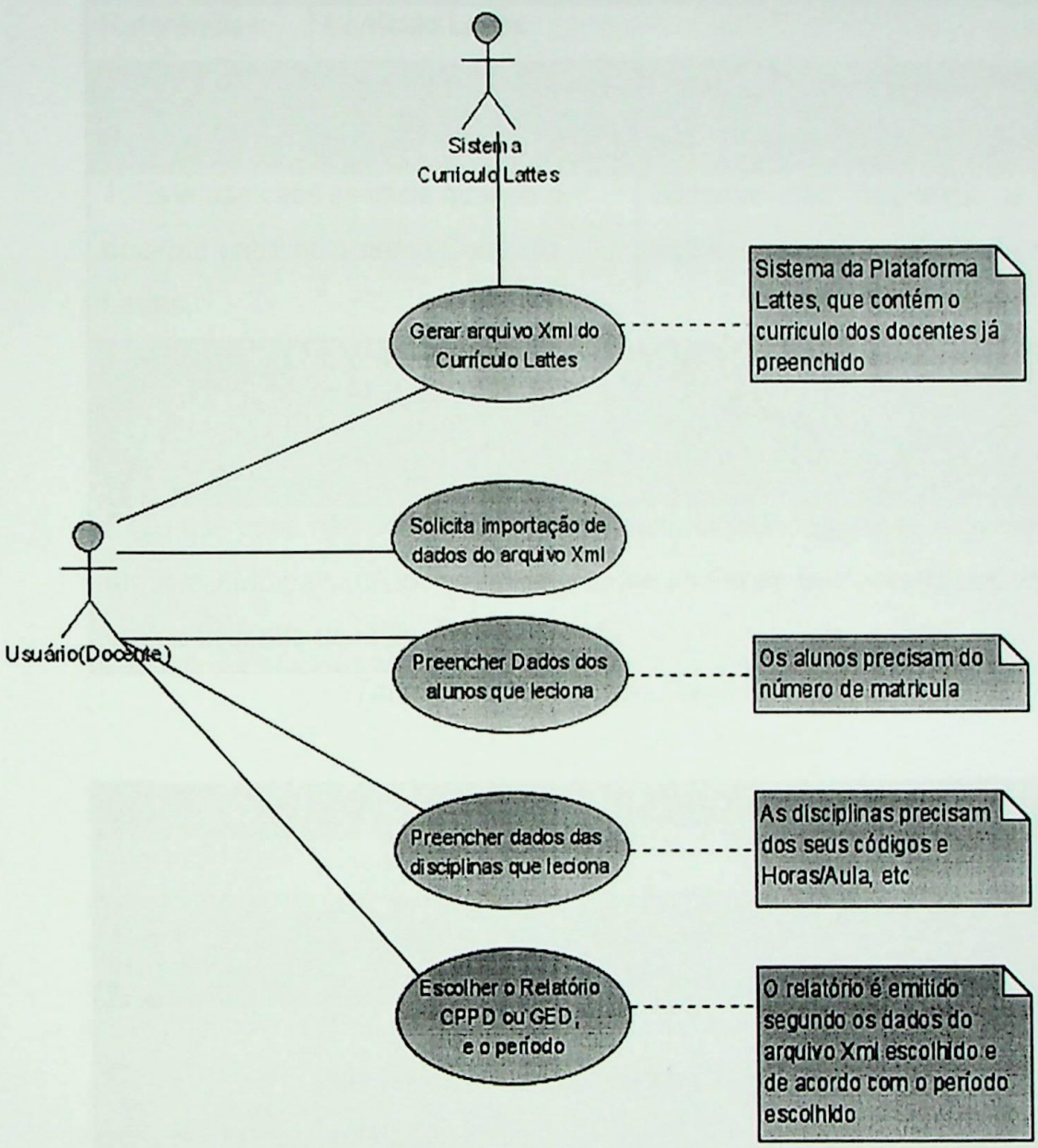

Figura 29. Diagrama Use Case do sistema

4.3.8.1.1 Cenário Use Case

Use Case:	**Gerar Arquivo XML do Currículo Lattes**
Atores:	Currículo Lattes / Docente
Objetivo:	Através do Currículo Lattes gerar o arquivo XML
Descrição:	O docente terá que entrar no sistema Currículo Lattes, onde seus dados já devem estar cadastrados e atualizados, e gerar o arquivo XML para uso do sistema.

Tipo:	Essencial
Referências:	Currículo Lattes
Curso de Eventos Típicos	
Ação do Ator	**Resposta do Sistema**
1. Este use case se inicia quando o docente entra no sistema Currículo Lattes.	Sistema não responde a essa ação.
2. Docente solicita exportação dos seus dados através de arquivo XML.	Sistema gera arquivo XML com todos os dados do docente.
Cursos Alternativos	
Esse use case não possui curso alternativo, sendo obrigatório a geração do arquivo XML pelo menos uma vez e todas as vezes que o sistema Currículo Lattes tiver sua base de dados alterada.	

Tabela 3. Descrição do Caso de Uso 1

Use Case:	**Solicitar importação de dados do arquivo XML**
Atores:	Docente
Objetivo:	Através do sistema encontrar o arquivo XML gravado em disco
Descrição:	O docente terá que entrar no sistema e escolher a opção de selecionar o arquivo XML que será usado para gerar os relatórios finais
Tipo:	Principal e Essencial
Referências:	Arquivo XML
Curso de Eventos Típicos	
Ação do Ator	**Resposta do Sistema**
1. Este use case se inicia quando o docente escolhe o arquivo XML	Sistema armazena referência da localização em disco do arquivo para uso posterior.
2. O sistema utiliza como pasta de saída dos relatórios gerados a pasta onde se encontra o XML.	Sistema armazena referência da pasta para utilizar na geração final dos relatórios.

Cursos Alternativos
Esse use case não possui curso alternativo, sendo obrigatório a escolha do arquivo XML.

Tabela 4. Descrição do Caso de Uso 2.

Use Case:	**Preenche dados dos alunos que leciona**
Atores:	Docente
Objetivo:	Através do sistema completar a base de dados com informações dos alunos.
Descrição:	O docente terá que entrar no sistema, solicitar a importação dos dados dos seus alunos para o sistema e depois atualizar os dados dos alunos que não constam no arquivo XML.
Tipo:	Opcional
Referências:	Arquivo XML
Curso de Eventos Típicos	
Ação do Ator	**Resposta do Sistema**
1. Este use case se inicia quando o docente solicita importação das informações do arquivo XML	Sistema armazena em sua base de dados os dados dos alunos do arquivo XML.
2. Completar informações dos alunos	O docente deve completar informações referentes a cada aluno, como o número de matrícula.
Cursos Alternativos	
Esse use case não possui curso alternativo, porém é opcional o complemento dos dados dos alunos.	

Tabela 5. Descrição do Caso de Uso 3.

Use Case:	**Preenche dados das disciplinas que leciona**
Atores:	Docente
Objetivo:	Através do sistema completar a base de dados com informações das disciplinas.

Descrição:	O docente terá que entrar no sistema, solicitar a importação dos dados das suas disciplinas e depois atualizar os dados destas que não constam no arquivo XML, como a quantidade de alunos, código da disciplina, entre outros.
Tipo:	Opcional
Referências:	Arquivo XML
Curso de Eventos Típicos	
Ação do Ator	**Resposta do Sistema**
1. Este use case se inicia quando o docente solicita importação das informações do arquivo XML	Sistema armazena em sua base de dados os dados das disciplinas do arquivo XML.
2. Completar informações das disciplinas	O docente deve completar informações referentes a cada disciplina, informações como código da disciplina, horas/aula e número de alunos. O sistema armazena essas informações em uma base de dados.
Cursos Alternativos	
Esse use case não possui curso alternativo, porém é opcional o complemento dos dados das disciplinas.	

Tabela 6. Descrição do Caso de Uso 4.

Use Case:	**Escolher o relatório CPPD ou GED e o período.**
Atores:	Docente
Objetivo:	Escolher o relatório a ser gerado e o período dos dados que irão compor esse relatório.
Descrição:	Após o preenchimento (opcional) dos complementos de alunos e disciplinas, o sistema está pronto a gerar os relatórios, cruzando os dados do arquivo XML e de sua base de dados. Basta escolher o período e o relatório que deseja gerar.

Tipo:	Principal e essencial
Referências:	Arquivo XML e Base de dados interna
Curso de Eventos Típicos	
Ação do Ator	**Resposta do Sistema**
1. Este use case se inicia quando o docente escolhe o relatório a ser gerado, CPPD ou GED.	O docente deve escolher o relatório a ser gerado pelo sistema.
2. Docente escolhe um período para solicitar o relatório deste período.	Sistema através das datas informadas vai fazer a busca dos dados no arquivo XML.
3. Docente solicita a geração do relatório.	O Sistema cruza os dados do arquivo XML com a base de dados interna e gera o relatório segundo o modelo escolhido, colocando os dados em locais específicos.
Cursos Alternativos	
Esse use case não possui curso alternativo, é de uso principal.	

Tabela 7. Descrição do Caso de Uso 5

4.3.8.2 Diagrama de Classes

As classes utilizadas em nosso sistema são as apresentadas na figura a seguir.

O Diagrama de Classe possui a classe Alunos, classe essa que tem por funcionalidade armazenar os dados dos alunos para cruzamento posterior com os oriundos do arquivo XML e geração de um relatório mais completo.

A classe XML é a classe referente ao arquivo XML importado pelo sistema. Ela tem somente atributos.

A classe Disciplinas trata dos dados referentes às disciplinas lecionadas pelo docente. Os dados são provenientes do arquivo XML. A classe Disciplinas tem uma classe derivada chamada Parâmetros das Disciplinas. Esta classe guarda dados específicos de cada disciplina importada do XML.

As classes CPPD e GED são as classes responsáveis pelo cruzamento de informações entre as classes e a geração dos relatórios. Estas classes possuem somente operações.

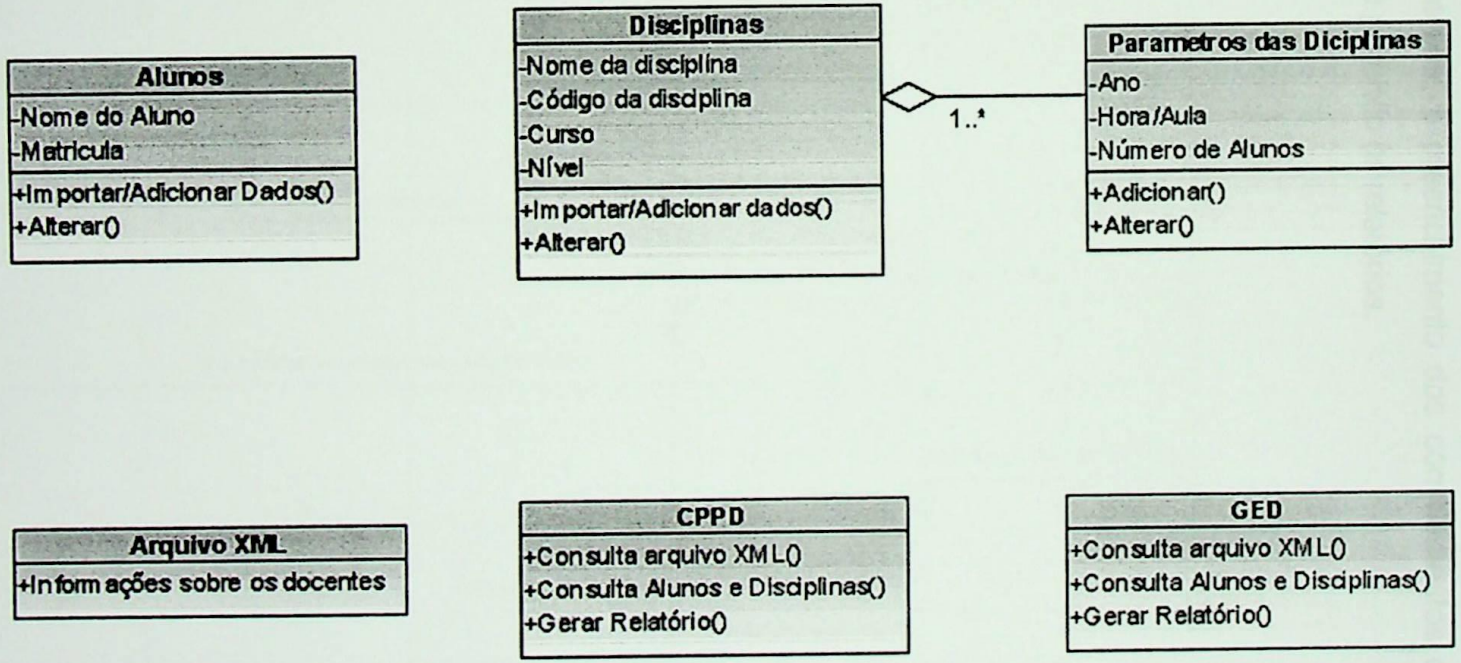

Figura 30. Diagrama de Classes do Sistema.

4.3.8.3 Diagrama de Seqüência

O Diagrama de Seqüência do sistema é apresentado a seguir. Ele ilustra toda a trajetória que o docente deverá realizar, para gerar os relatórios da CPPD e da GED, iniciando com a interação que deve haver com o Currículo Lattes, a escolha do arquivo XML gerado, o preenchimento dos complementos das disciplinas e dos alunos e finaliza gerando os relatórios.

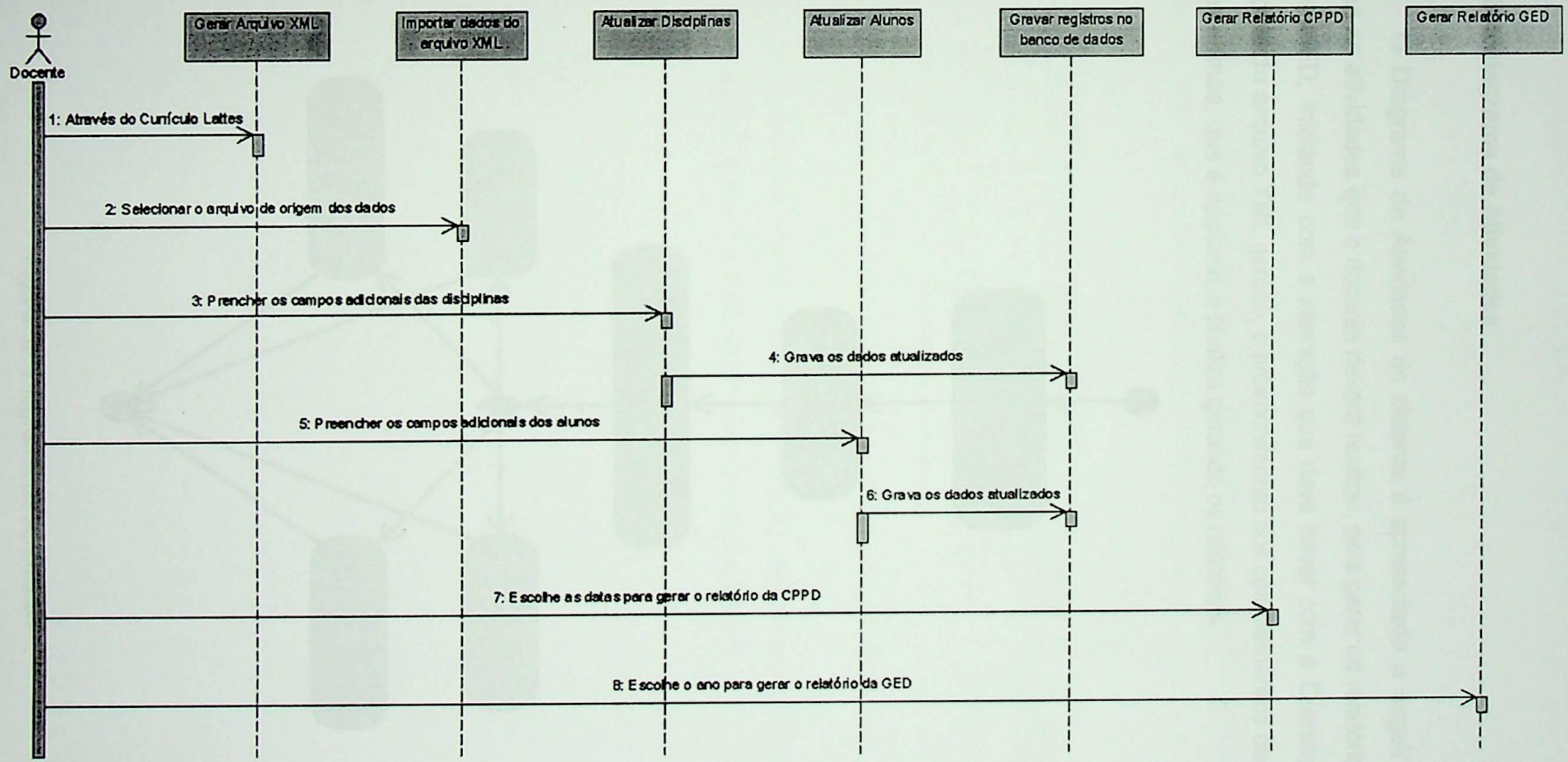

Figura 31. Diagrama de seqüência do sistema.

4.3.8.4 Diagrama de Atividades

O Diagrama de Atividades do sistema é apresentado a seguir. Ele ilustra todas as atividades que o docente deverá realizar, para gerar os relatórios da CPPD e da GED, iniciando com a interação que deve haver com o Currículo Lattes, a escolha do arquivo XML gerado, o preenchimento dos complementos das disciplinas e dos alunos, que é opcional, e finaliza gerando os relatórios.

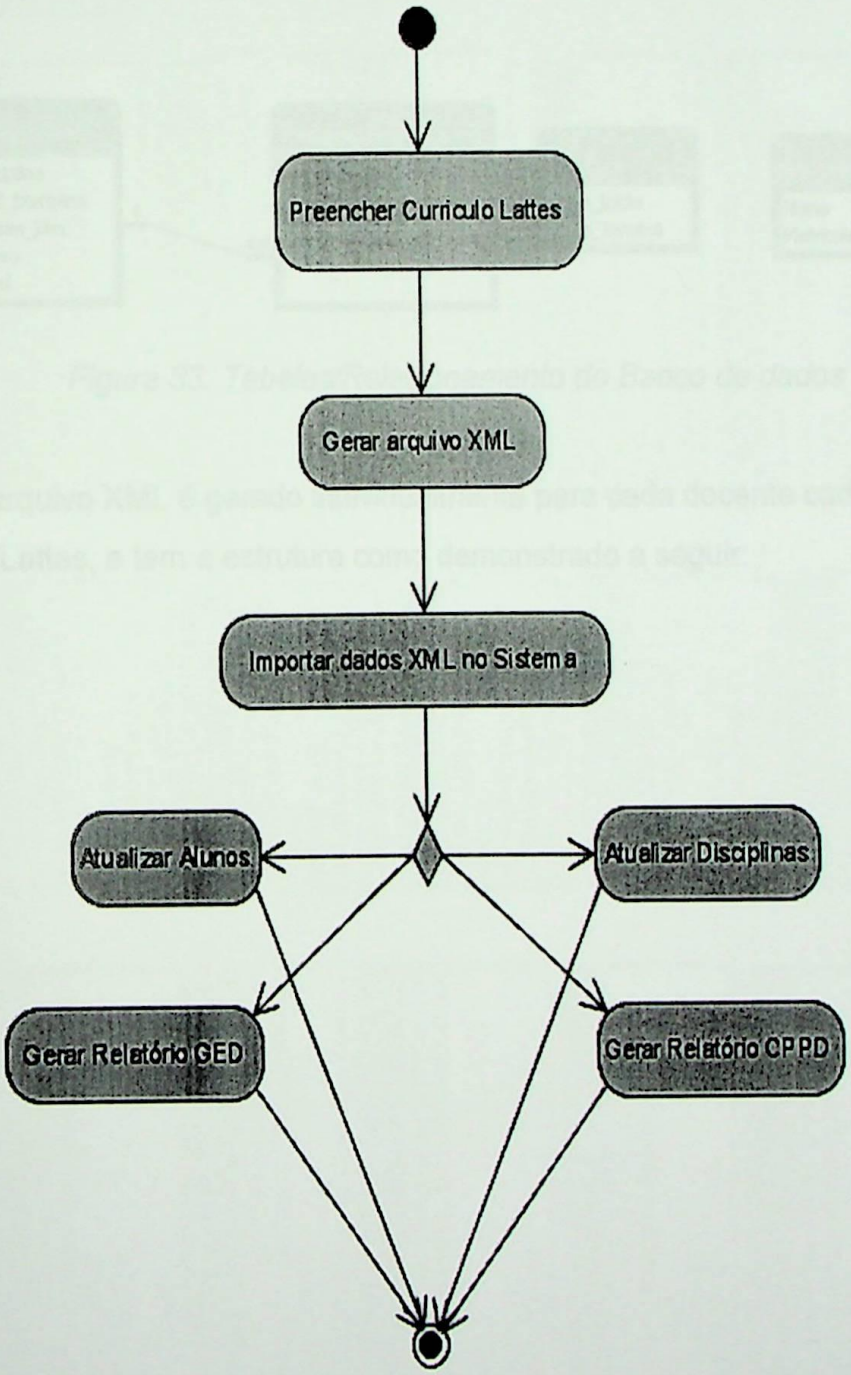

Figura 32. Diagrama de Atividades

4.4 Banco de Dados do Sistema

O arquivo XML importado do Currículo Lattes não possui todos os dados necessários para gerar os relatórios. Uma base de dados auxiliar foi desenvolvida para mesclar os dados com o arquivo XML proporcionando um relatório mais completo.

O banco de dados utilizado foi o Microsoft Access, um banco relacional para utilização local. As tabelas e o relacionamento entre elas são mostrados a seguir.

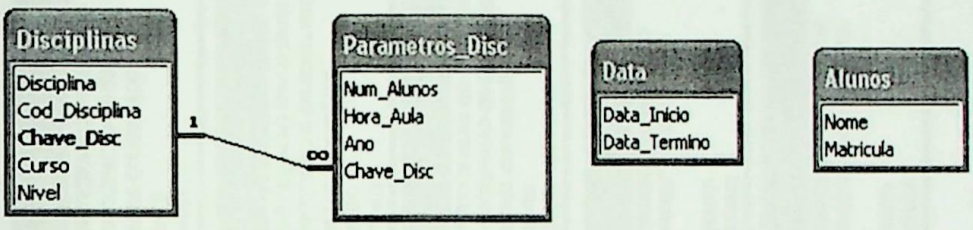

Figura 33. Tabelas/Relacionamento do Banco de dados

O arquivo XML é gerado individualmente para cada docente cadastrado no Currículo Lattes, e tem a estrutura como demonstrado a seguir:

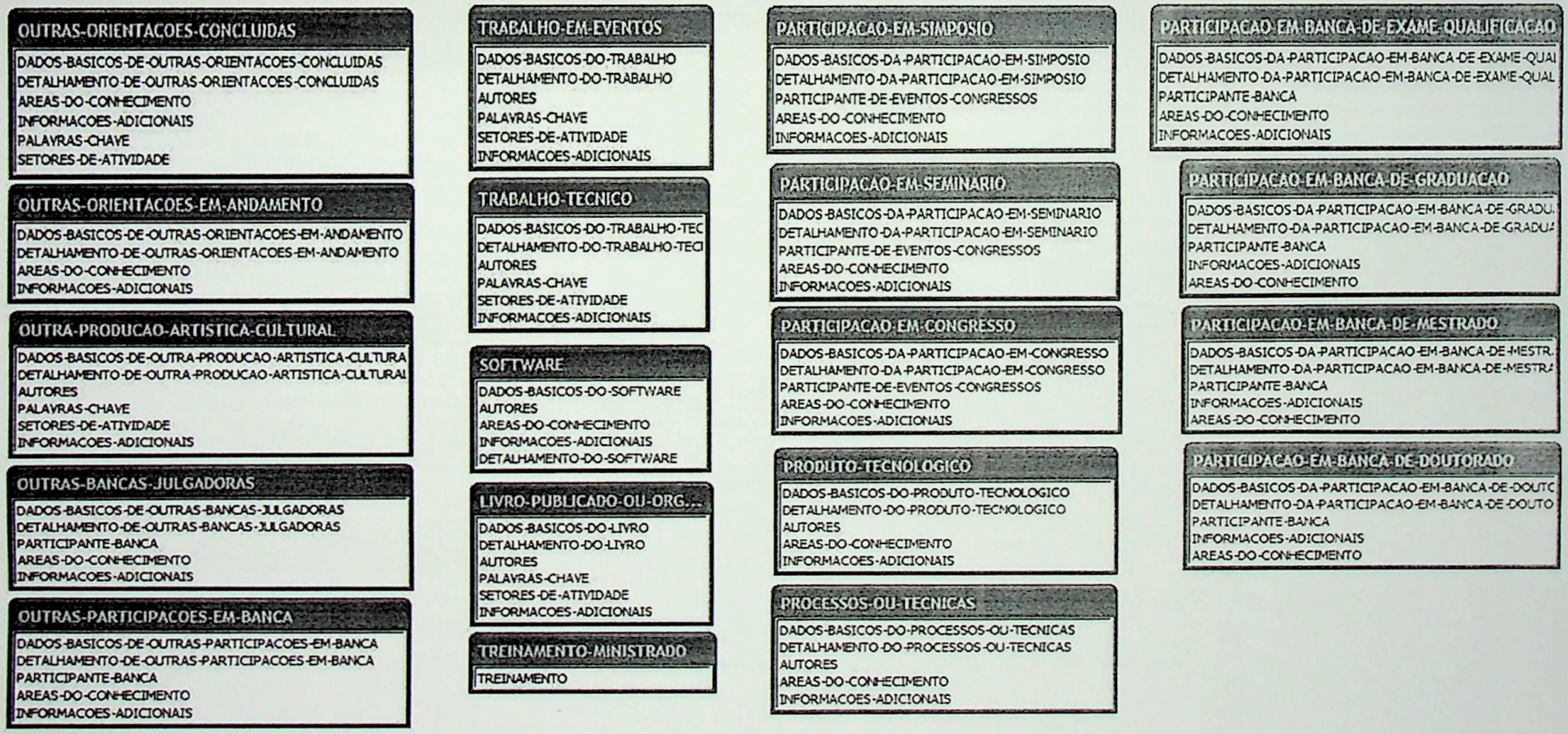

Figura 34. Estrutura do arquivo XML do Currículo Lattes.

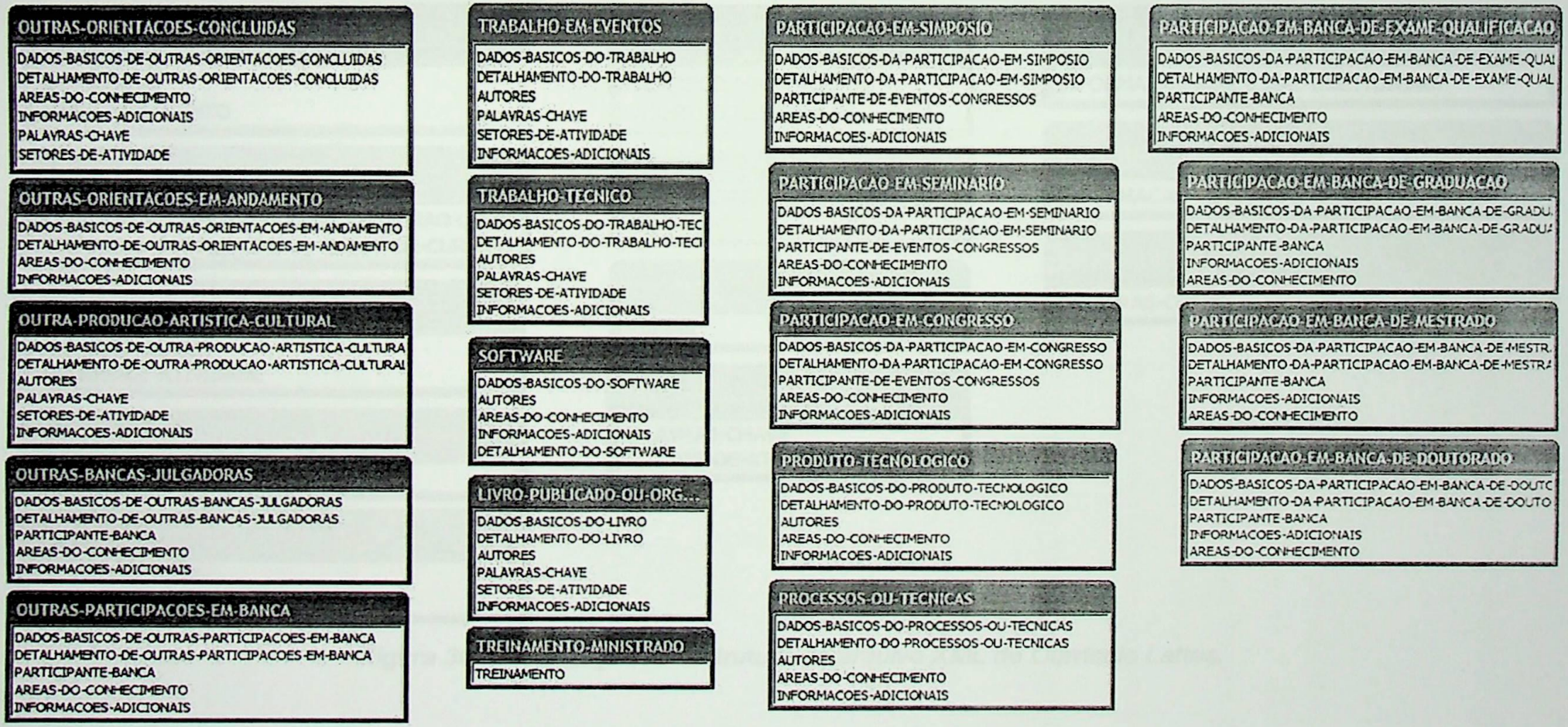

Figura 35. Continuação 1 - Estrutura do arquivo XML do Currículo Lattes.

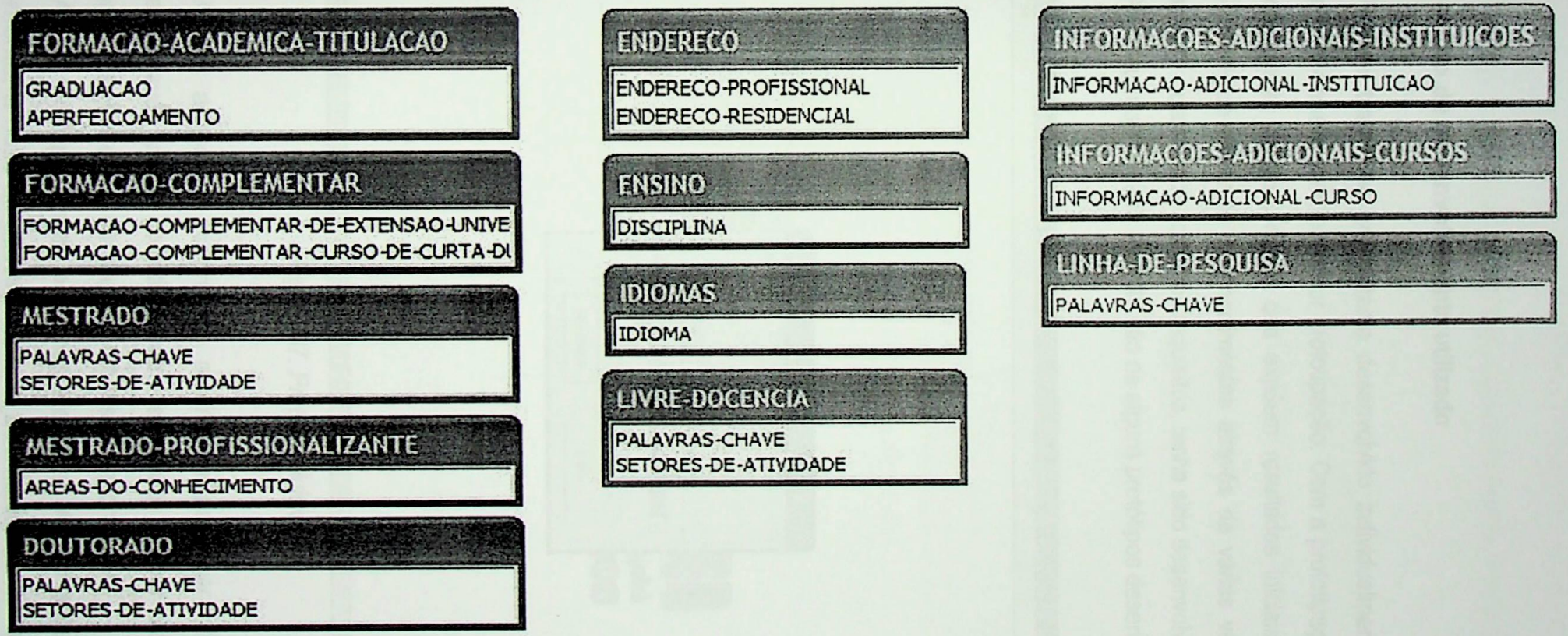

Figura 36. Continuação 2 - Estrutura do arquivo XML do Currículo Lattes.

4.5 Processo de desenvolvimento utilizado

Por se tratar de um sistema desenvolvido individualmente, foi adotado o processo de desenvolvimento por prototipação. Com a prototipação o sistema teve desenvolvimento evolucionário, que expõem resultados iniciais a aprovação do usuário e segue fazendo aprimoramentos através de várias versões, até que o sistema final, que atenda todos os requisitos, tenha sido desenvolvido.

As telas das figuras 37 à 42 são de alguns protótipos desenvolvidos:

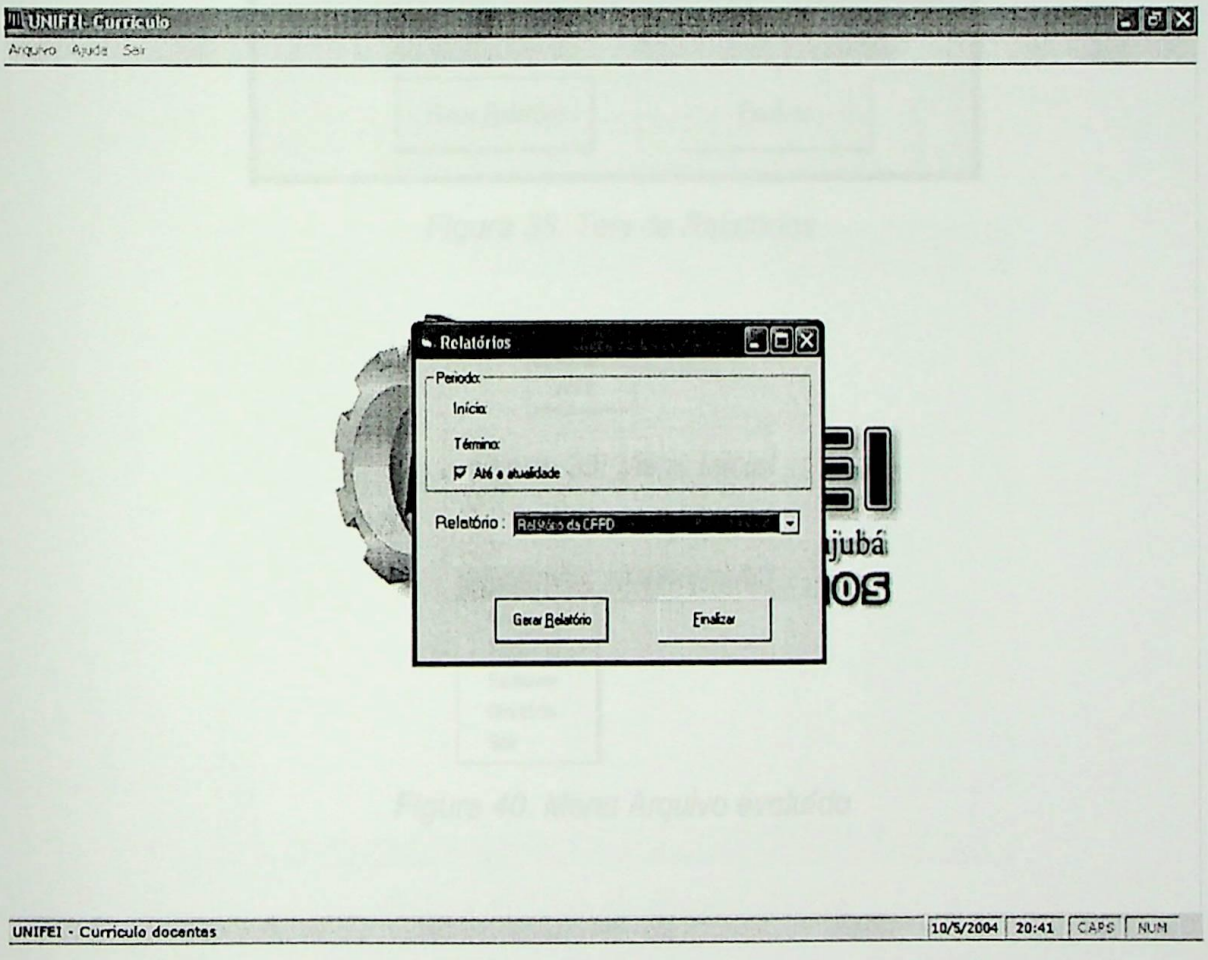

Figura 37. Protótipo inicial.

A tela acima foi do primeiro protótipo utilizado para desenvolvimento do sistema. Através deste protótipo foram levantados alguns requisitos e algumas mudanças que deveriam ser implementadas posteriormente.

A tela principal sofreu pequenas alterações. As mudanças foram realizadas internamente, a base de dados auxiliar foi adicionada e o menu foi alterado com a

seqüência dos protótipos. As figuras a seguir mostram alguns menus e telas que foram modificadas com a prototipação.

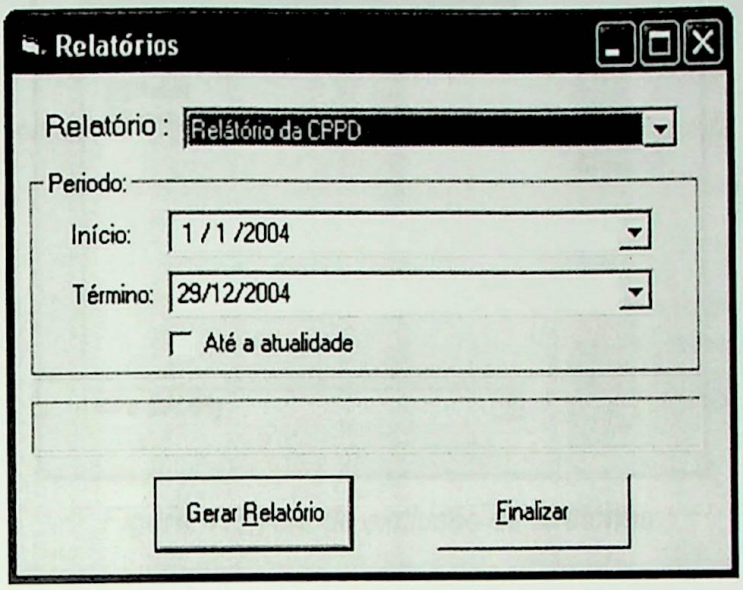

Figura 38. Tela de Relatórios

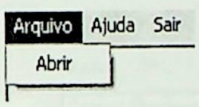

Figura 39. Menu Inicial

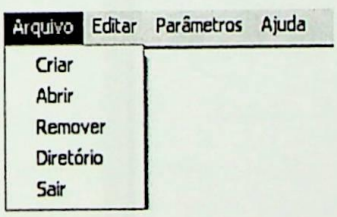

Figura 40. Menu Arquivo evoluído

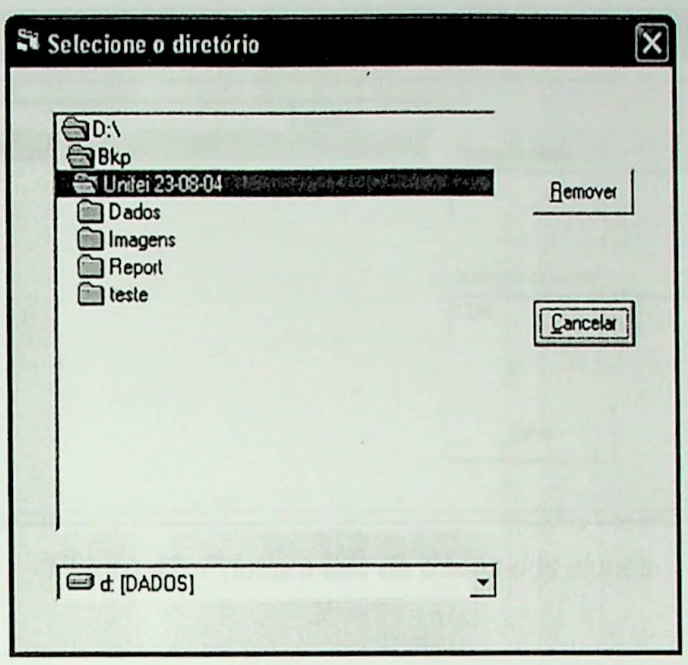

Figura 41. Tela de exclusão de diretórios

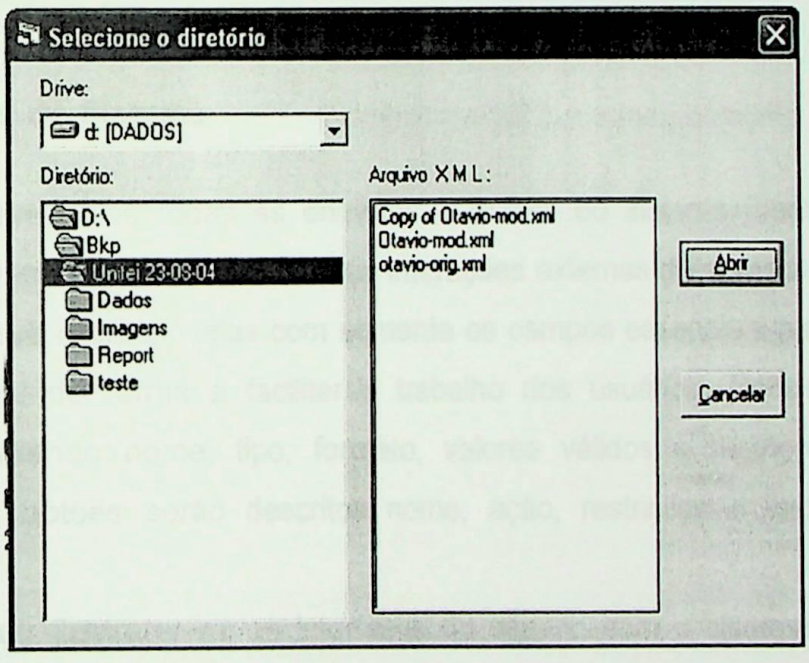

Figura 42. Tela de escolha do XML

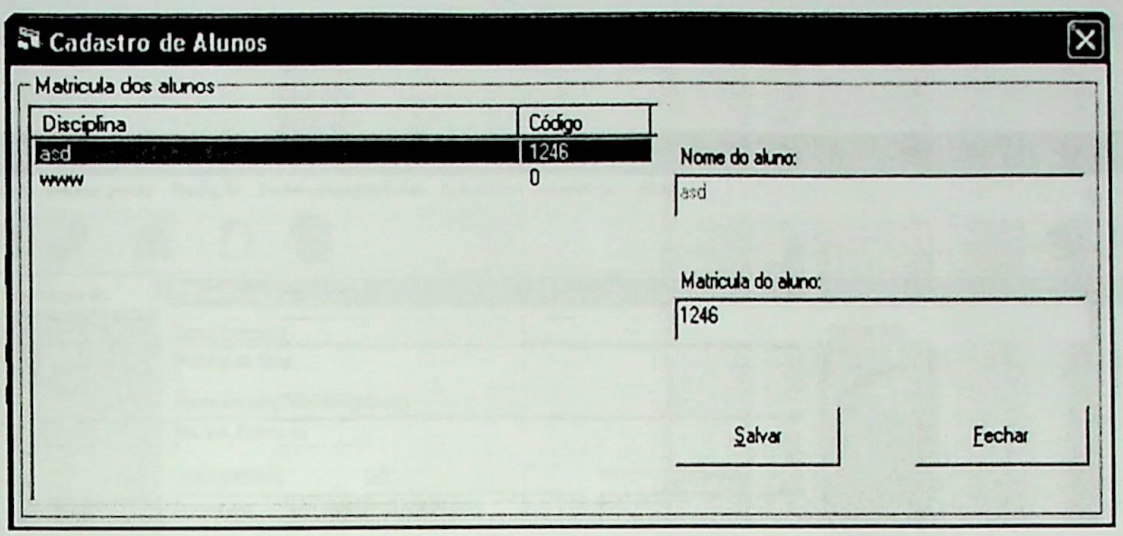

Figura 43. Primeira tela de dados dos alunos

As telas acima mostram algumas partes do processo de prototipação pelo qual o sistema foi desenvolvido. As telas finais são apresentadas a seguir no próximo item.

4.6 Interfaces do Sistema

Descreveremos todas as entradas e saídas do sistema, bem como suas interações. Descreveremos também as interações externas do sistema de software. O sistema deve oferecer telas com somente os campos essenciais para a geração dos relatórios, de forma a facilitar o trabalho dos usuários. Os campos serão descritos contendo nome, tipo, formato, valores válidos e restrições. Para os comandos e botões serão descritos nome, ação, restrições e seqüências dos mesmos.

A seguir ilustraremos as interfaces do usuário com o sistema de software Currículo Lattes e com o nosso sistema.

4.6.1 Tela do Sistema Currículo Lattes

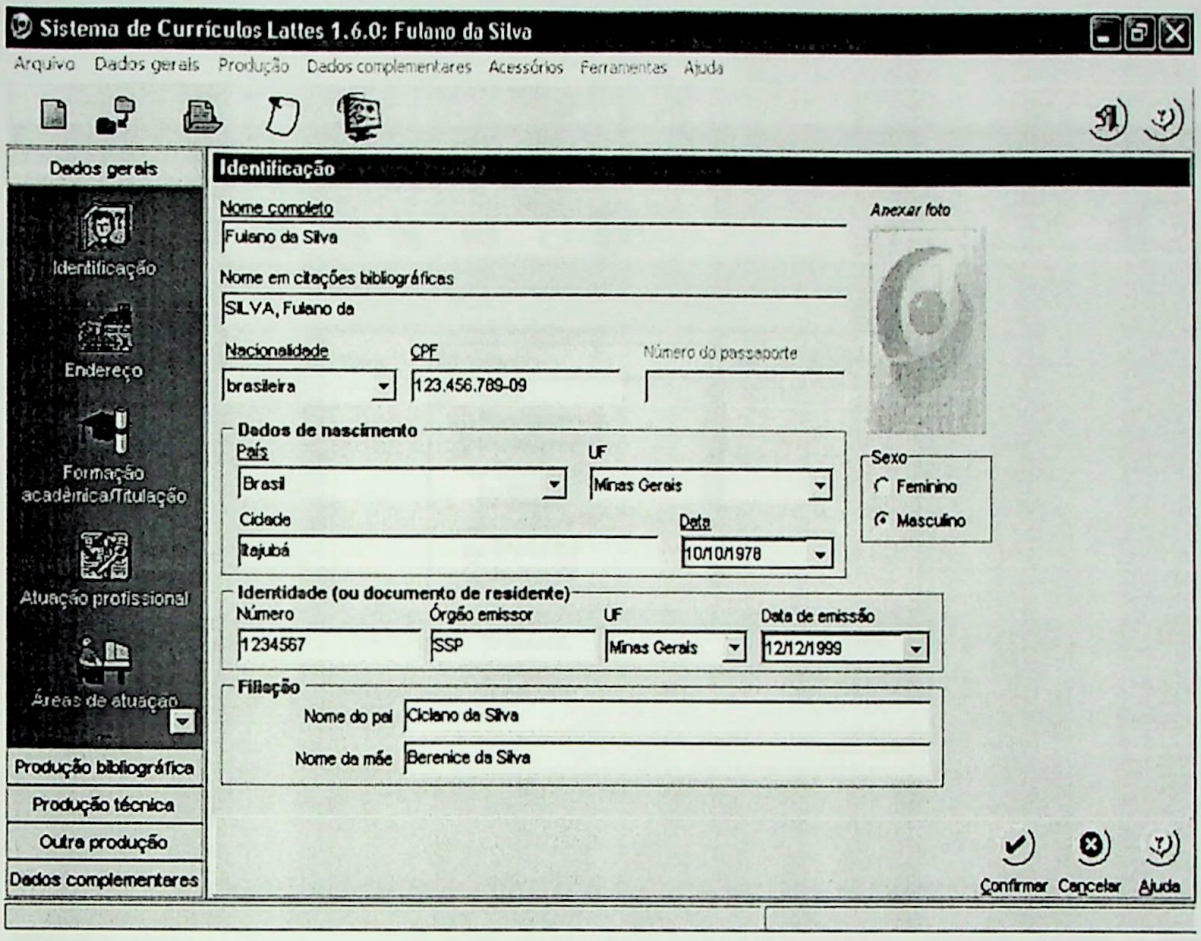

Figura 44. Tela do Currículo Lattes.

No sistema do Currículo Lattes, devem ser cadastradas todas as informações referentes ao docente, suas atividades e produções. Estando os dados cadastrados no Currículo Lattes, pode-se a qualquer momento exportá-los para um arquivo XML, o qual será utilizado em nosso sistema. Não entraremos em detalhes sobre o preenchimento de dados no Currículo Lattes por não ser o foco do nosso trabalho.

4.6.2 Exportação de dados do Currículo Lattes

A tela a seguir apresenta a tela de exportação dos dados do Currículo Lattes.

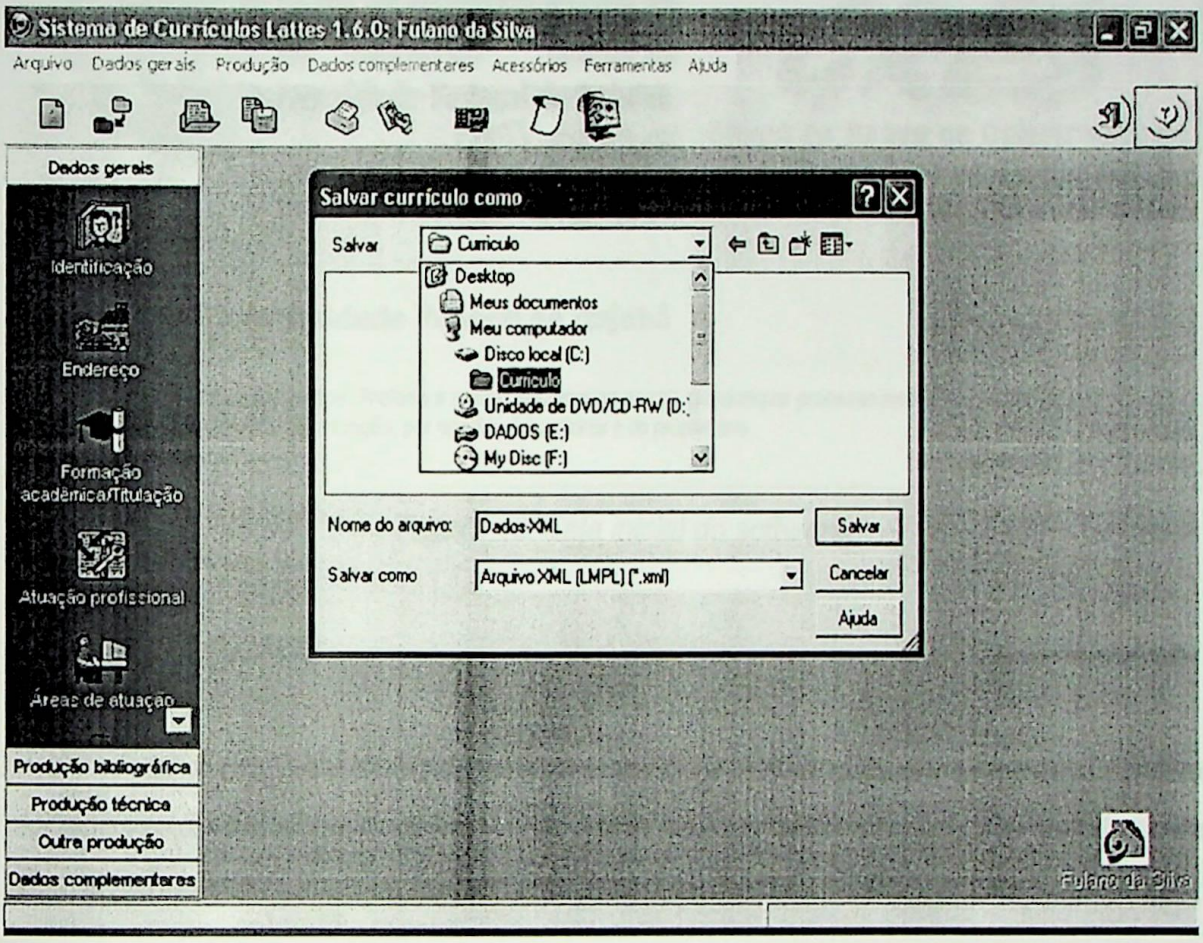

Figura 45. Exportar arquivo XML no Currículo Lattes.

A exportação dos dados é bem simples. Após abrir os seus dados no sistema de software Currículo Lattes, escolha no menu Arquivo a opção Exportar Dados e depois clique em Arquivo XML. Aparecerá então a tela acima para escolher o local e o nome do arquivo a ser salvo. Apesar de simples, esta etapa é fundamental para a geração dos relatórios no nosso sistema, pois teremos posteriormente que localizar a pasta e o arquivo que foi salvo.

4.6.3 Tela de abertura do sistema

A tela a seguir é a tela de abertura do software. Ela tem somente a função de mostrar os dados referentes ao software.

Figura 46. Tela inicial do software.

4.6.4 Tela Principal

A tela principal do sistema de software tem os logotipos e oferece um menu de acesso às funções do mesmo que serão apresentadas a seguir. Ela se relaciona diretamente com todas as outras interfaces do software.

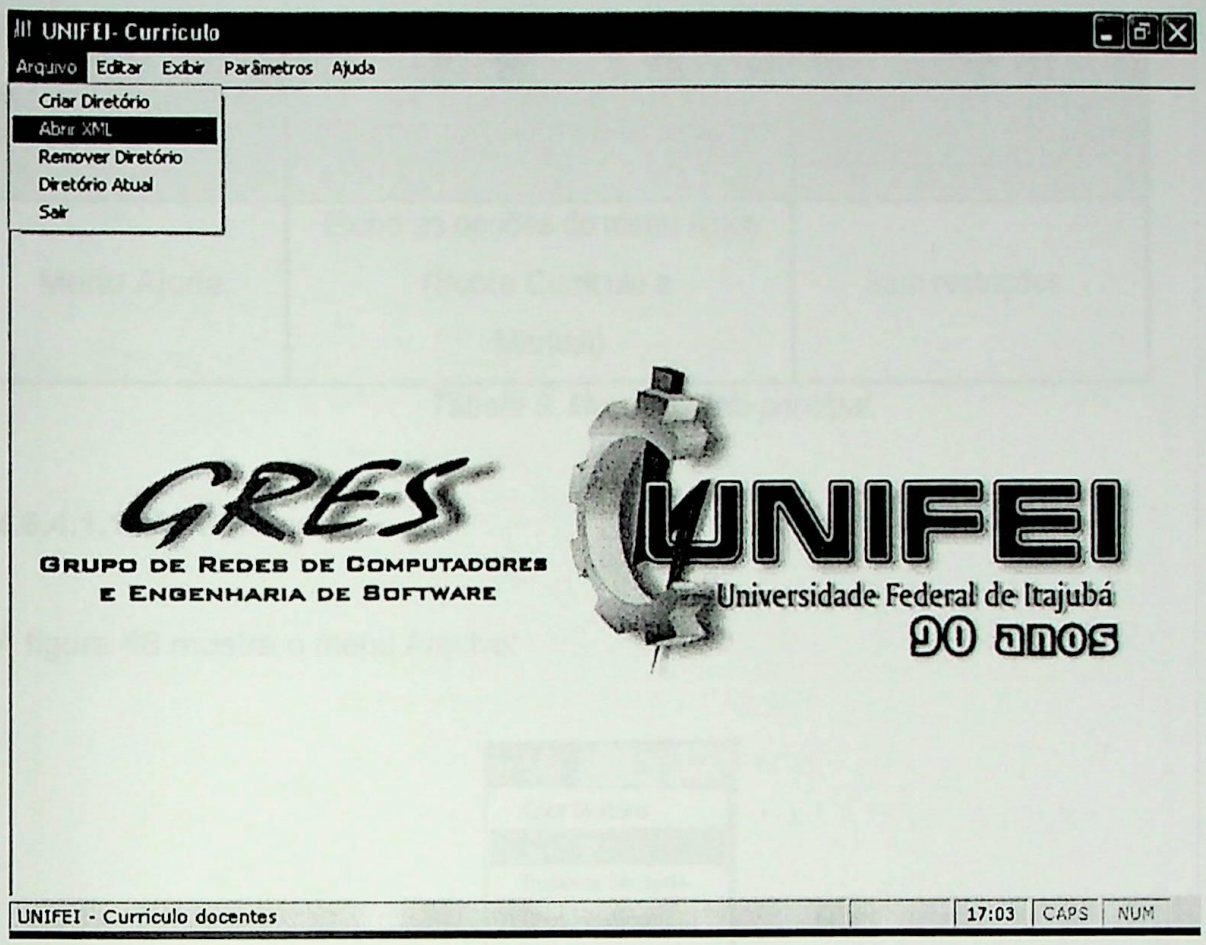

Figura 47. Tela principal do software.

4.6.4.1 Comando e ações da tela principal

Nome	Ação	Restrições
Menu Arquivo	Exibe as opções do menu Arquivo (Criar Diretório, Abrir XML, Remover Diretório, Diretório Atual e Sair)	Sem restrições
Menu Editar	Exibe as opções do menu editar (CPPD e GED)	Deve ser escolhido anteriormente o arquivo XML a ser utilizado.
Menu Exibir	Exibe as opções do menu Exibir (Relatórios)	O relatório deve ser gerado anteriormente para que o mesmo possa ser exibido.

Menu Parâmetros	Exibe as opções do menu parâmetros (Alunos e Disciplinas)	Deve ser escolhido anteriormente o arquivo XML a ser utilizado.
Menu Ajuda	Exibe as opções do menu Ajuda (Sobre Currículo e Manual)	Sem restrições.

Tabela 8. Menus da tela principal.

4.6.4.1.1 Menu Arquivo

A figura 48 mostra o menu Arquivo:

Figura 48. Menu Arquivo.

4.6.4.1.1.1 Comando e ações dos itens do menu Arquivo

Nome	Ação	Restrições
Criar Diretório	Oferece a opção de criar um novo diretório em um local escolhido pelo usuário. Esse diretório será utilizado para armazenar seus relatórios.	Sem restrições
Abrir XML	Menu de uso essencial do programa que oferece a opção de escolha de um arquivo XML, que deve ser localizado em uma pasta ou unidade do computador. Será utilizado para gerar os relatórios e preencher alguns dados da base de dados	O arquivo deve ser gerado anteriormente no sistema de software Currículo Lattes.

	interna. Ao se escolher um arquivo XML, automaticamente o diretório do arquivo passa a ser o diretório de trabalho do sistema, onde os relatórios serão gerados.	
Remover Diretório	Neste menu, temos a opção de excluir um diretório com todos os seus arquivos.	Sem restrições.
Diretório Atual	Mostra uma mensagem na tela com o diretório que está sendo utilizado para gerar os relatórios no momento.	Sem restrições.
Sair	Exibe uma mensagem perguntando se deseja realmente sair do sistema. Se clicar na opção Sim o sistema será fechado. Clicando em não o sistema continuará aberto.	Sem restrições.

Tabela 9. Menu Arquivo.

4.6.4.1.2 Menu Editar

O menu Editar, como mostrado na figura 49, tem as opções de escolha dos dois relatórios gerados pelo sistema.

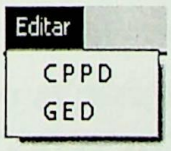

Figura 49. Menu Editar.

Comando e ações dos itens do menu Editar.

Nome	Ação	Restrições
CPPD	Exibe a tela de escolha do período para geração do relatório da CPPD	Deve ser escolhido anteriormente o arquivo XML a ser utilizado.

GED	Exibe a tela para escolha de qual ano você quer gerar o relatório da GED	Deve ser escolhido anteriormente o arquivo XML a ser utilizado.

Tabela 10. Menu Editar.

4.6.4.1.3 Menu Exibir

O menu Exibir, como mostrado na figura 50 tem apenas uma opção descrita na tabela 11.

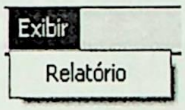

Figura 50. Menu Exibir.

Comando e ações dos itens do menu Exibir:

Nome	Ação	Restrições
Relatório	Exibe uma tela para escolha de um arquivo gerado pelo sistema anteriormente. Após escolhido o arquivo, abre o mesmo no editor de texto padrão do sistema para edição ou visualização.	Deve ser gerado anteriormente o relatório a ser exibido.

Tabela 11. Menu Exibir.

4.6.4.1.4 Menu Parâmetros

O menu Parâmetros, como mostrado na figura 51, tem as opções de escolha dos dois relatórios gerados pelo sistema.

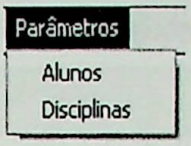

Figura 51. Menu Parâmetros.

Comando e ações dos itens do menu Parâmetros:

Nome	Ação	Restrições
Alunos	Exibe a tela Cadastro de Alunos. Nesta tela, cadastramos informações sobre os alunos que não constam no arquivo XML.	Deve ser escolhido anteriormente o arquivo XML a ser utilizado.
Disciplinas	Exibe a tela Cadastro de Disciplinas. Nesta tela, cadastramos informações sobre as disciplinas que não constam no arquivo XML.	Deve ser escolhido anteriormente o arquivo XML a ser utilizado.

Tabela 12. Menu Editar.

4.6.4.1.5 Menu Ajuda

O menu Ajuda, como mostrado na figura 52, tem as opções de exibir informações sobre o sistema de software e o manual do mesmo.

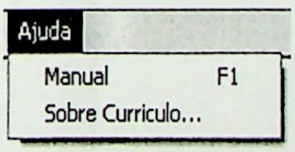

Figura 52. Menu Ajuda.

Comando e ações dos itens do menu Ajuda:

Nome	Ação	Restrições
Manual	Exibe um documento digital no editor de texto padrão, contendo o manual do software, com todas as referências de como utilizá-lo.	Sem restrições.
Sobre Currículo	Exibe uma tela com informações sobre o sistema de software (Tela mostrada a seguir).	Sem restrições.

Tabela 13. Menu Ajuda.

4.6.5 Tela Escolha do XML

A tela de escolha do arquivo XML a ser utilizado é mostrada na figura 53. Ela é de uso obrigatório para geração dos relatórios.

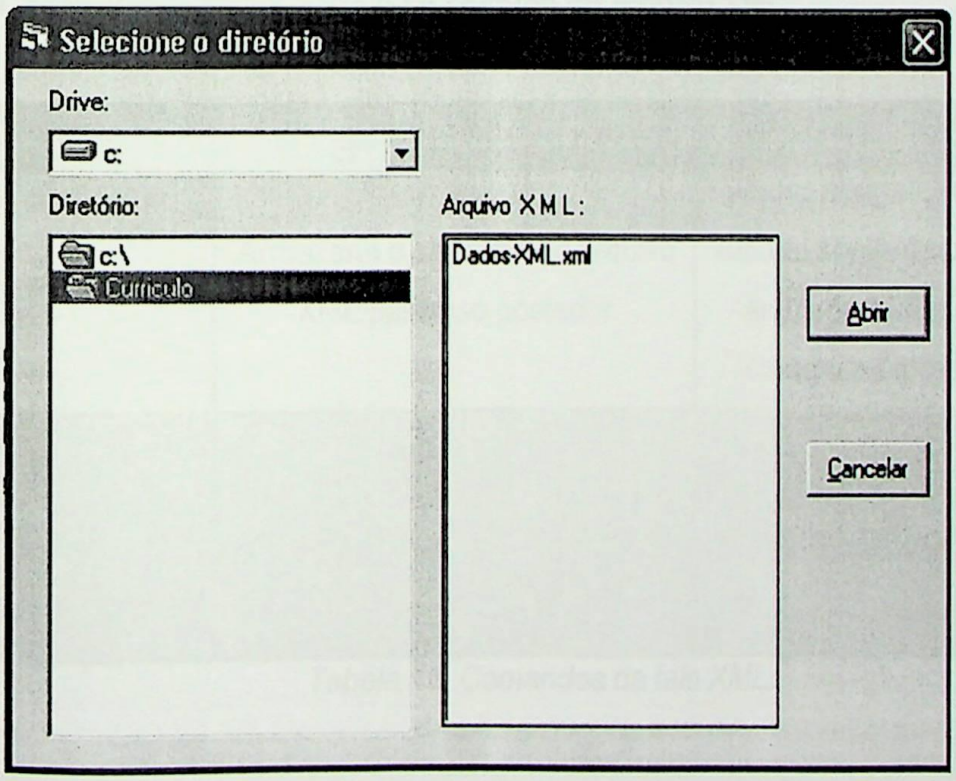

Figura 53. Tela de Escolha do arquivo XML.

4.6.5.1 Descrição dos itens da tela de Escolha do arquivo XML

Nome	Valores válidos	Formato	Tipo	Restrições
Drive	Todas as unidades de armazenamento contidas no computador local.	-	-	Sem restrições.
Diretório	Todos os diretórios e subdiretórios do drive escolhido acima.	-	-	Sem restrições.
Arquivo XML	Todos os arquivos XML do diretório escolhido no item acima	-	XML	Apenas arquivos XML

Tabela 14. Descrição dos itens da tela de escolha do arquivo XML.

4.6.5.2 Comando e ações da tela de escolha do arquivo XML

Nome	Ação	Restrições
Abrir	Armazena o caminho do arquivo XML para uso posterior.	De uso obrigatório e suporte somente para arquivos XML do Currículo Lattes.
Cancelar	Cancela a escolha do arquivo XML. Se outro arquivo tiver sido escolhido anteriormente ele continuará a ser válido.	Sem restrições.

Tabela 15. Comandos da tela XML.

4.6.6 Tela de exclusão de diretórios

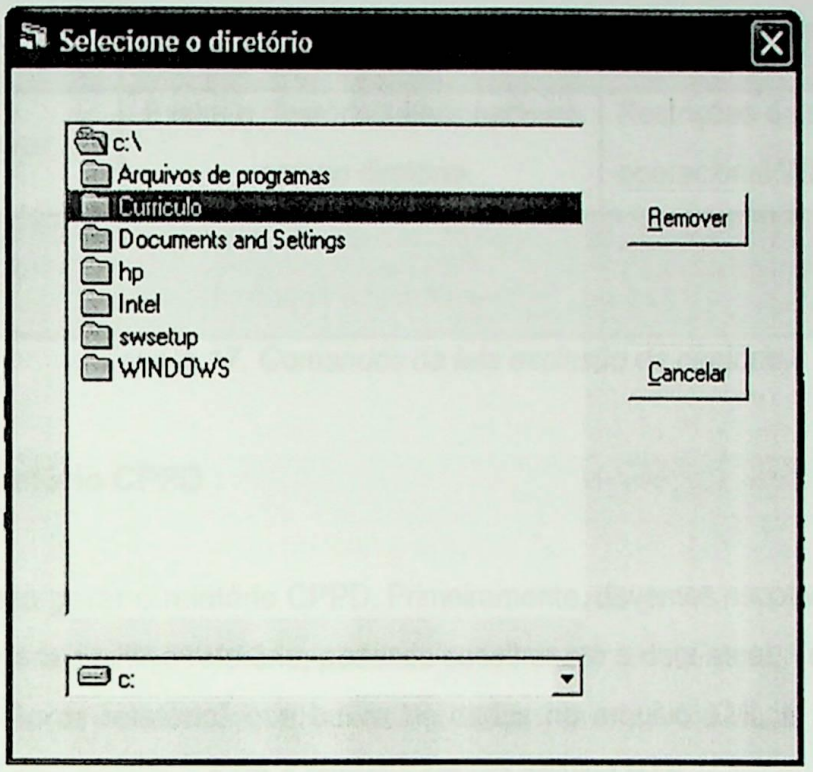

Figura 54. Tela de exclusão de diretórios.

4.6.6.1 Descrição dos itens da tela de exclusão de diretórios

Nome	Valores válidos	Formato	Tipo	Restrições
Diretório	Todos os diretórios e subdiretórios do drive escolhido no item abaixo.	-	-	Sem restrições.
Drive	Todas as unidades de armazenamento contidas no computador local.	-	-	Sem restrições.

Tabela 16. Descrição dos itens da tela de exclusão de diretórios.

4.6.6.2 Comando e ações da tela de exclusão de diretórios

Nome	Ação	Restrições
Remover	Exclui o diretório selecionado no campo diretório.	Restrições do sistema operacional Windows.
Cancelar	Cancela a exclusão dos diretórios e fecha a tela.	Sem restrições.

Tabela 17. Comandos da tela exclusão de diretório.

4.6.7 Tela Relatório CPPD

Tela para gerar o relatório CPPD. Primeiramente, devemos escolher o período inicial e final para emitir o relatório, podendo escolher até a data atual. Temos então, o comando "Gerar relatório" que busca os dados no arquivo XML e na base de dados interna do sistema e gera o relatório em um arquivo RTF.

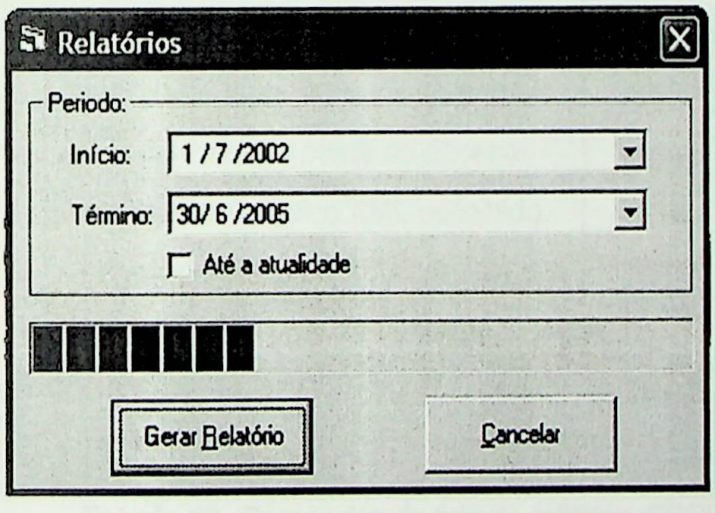

Figura 55. Tela do relatório CPPD.

4.6.7.1 Descrição dos itens da tela do relatório CPPD

Nome	Valores válidos	Formato	Tipo	Restrições
Data de início	Datas	dd/mm/yyyy (dia/mês/ano)	Date	Datas entre 1/1/1893 até 31/12/9999

Data de término	Datas	dd/mm/yyyy (dia/mês/ano)	Date	Datas entre 1/1/1893 até 31/12/9999
Barra de progressão	0 a 100	Valores numéricos de 0 a 100	Inteiro	Valores entre 0 e 100

Tabela 18. Descrição dos itens da tela do relatório CPPD.

4.6.7.2 Comando e ações da tela do relatório CPPD

Nome	Ação	Restrições
Gerar Relatório	Busca os dados no arquivo XML escolhido anteriormente e cruza esses dados com os cadastrados na base interna. Gera o relatório do período escolhido nos campos data de início e data de término. O relatório é salvo na pasta de trabalho escolhida anteriormente no formato RTF e com o nome do docente contido no arquivo XML escolhido, seguido da sigla CPPD e a data de início e data de término.	Escolha do arquivo XML e do diretório de trabalho no qual será gerado o relatório.
Cancelar	Cancela a emissão do relatório CPPD e fecha a tela.	Sem restrições.

Tabela 19. Comandos da tela do relatório CPPD.

4.6.8 Tela relatório GED

Tela para gerar o relatório GED. Primeiramente, devemos escolher o ano para emitir o relatório GED. Temos então, o comando "Gerar relatório" que busca os dados no arquivo XML e na base de dados interna do sistema e gera o relatório em um arquivo RTF segundo o formato contido no anexo.

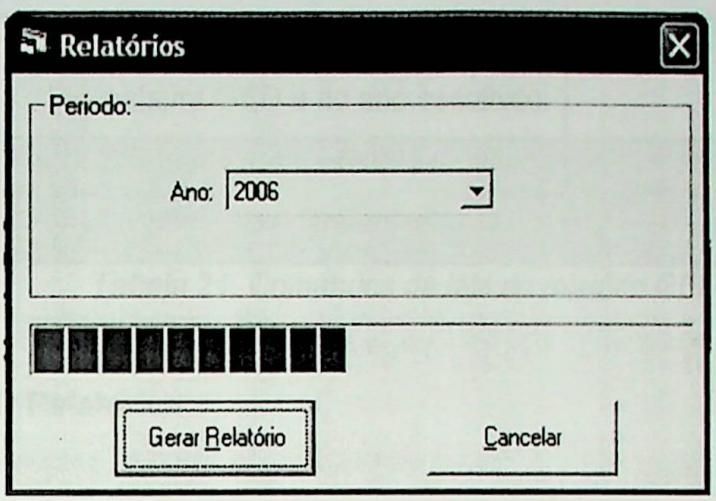

Figura 56. Tela do relatório GED.

4.6.8.1 Descrição dos itens da tela do relatório GED

Nome	Valores válidos	Formato	Tipo	Restrições
Ano	Datas	yyyy (ano)	Date	Datas entre 1893 até 9999
Barra de progresso	0 a 100	Valores numéricos de 0 a 100	Inteiro	Valores entre 0 e 100

Tabela 20. Descrição dos itens da tela do relatório GED.

4.6.8.2 Comando e ações da tela do relatório GED

Nome	Ação	Restrições
Gerar Relatório	Busca os dados no arquivo XML escolhido anteriormente e cruza esses dados com os cadastrados na base interna. Gera o relatório do ano escolhido no campo ano. O relatório é salvo na pasta de trabalho escolhida anteriormente no formato RTF e com o nome do docente contido no arquivo	Escolha do arquivo XML e do diretório de trabalho no qual será gerado o relatório.

	XML escolhido, seguido da palavra GED e do ano escolhido.	
Cancelar	Cancela a emissão do relatório GED e fecha a tela.	Sem restrições.

Tabela 21. Comandos da tela do relatório GED.

4.6.9 Tela Exibir Relatórios

Tela para exibir os relatórios gerados anteriormente. Escolhemos o drive e o diretório onde serão exibidos todos os nomes dos arquivos RTF contidos no diretório. Se ele tiver vários arquivos podemos abrir qualquer um dos arquivos, apenas escolhendo o arquivo correto.

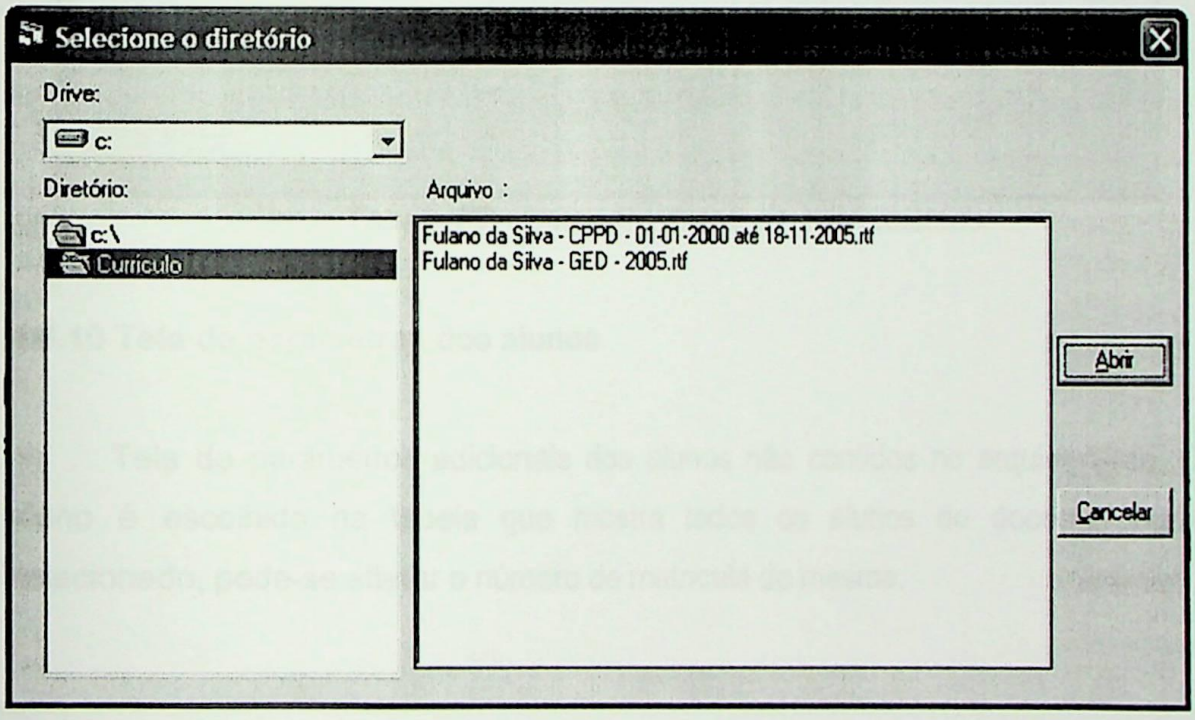

Figura 57. Tela exibir relatório.

4.6.9.1 Descrição dos itens da tela de exibir relatórios

Nome	Valores válidos	Formato	Tipo	Restrições
Drive	Unidades de armazenamento do computador local	-	-	-

Diretório	Diretórios da unidade escolhida acima	-	-	-
Arquivo	Arquivos RTF	Nome dos arquivos	RTF	Somente arquivos RTF

Tabela 22. Descrição dos itens da tela de exibir relatório.

4.6.9.2 Comando e ações da tela exibir relatório

Nome	Ação	Restrições
Abrir	Exibe o arquivo RTF no editor de texto padrão do sistema. O arquivo, após aberto, pode ser editado, impresso e salvo.	Sem restrições.
Cancelar	Cancela a exibição do relatório e fecha a tela.	Sem restrições.

Tabela 23. Comandos da tela de exibir relatório.

4.6.10 Tela de parâmetros dos alunos

Tela de parâmetros adicionais dos alunos não contidos no arquivo XML. O aluno é escolhido na tabela que mostra todos os alunos do docente. Após selecionado, pode-se alterar o número de matrícula do mesmo.

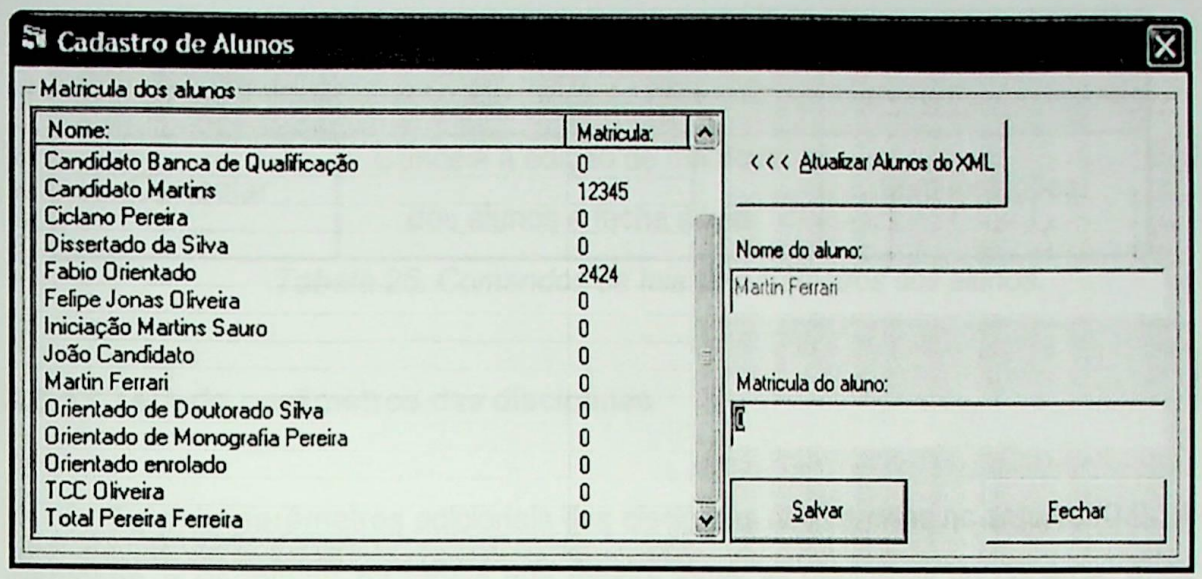

Figura 58. Tela de Cadastro de parâmetros dos alunos.

4.6.10.1 Descrição dos itens da tela de cadastro de parâmetros dos alunos

Nome	Valores válidos	Formato	Tipo	Restrições
Matrícula	Valores numéricos.	Inteiros positivos	Inteiro	Números positivos
Nome do Aluno	-	150 caracteres	String	Nome não pode ser alterado, por ser utilizado para fazer ligação com o arquivo XML
Tabela	Nome do aluno e matricula	2 colunas de caracteres	String e inteiro	Somente leitura.

Tabela 24. Descrição dos itens da tela de parâmetros dos alunos.

4.6.10.2 Comando e ações da tela de parâmetros dos alunos

Nome	Ação	Restrições
Atualizar alunos do XML	Busca novos alunos no XML escolhido anteriormente. Alunos já adicionados não são alterados.	O arquivo XML deve ser escolhido anteriormente.

Salvar	Salva alterações no número de matrícula do aluno escolhido.	Escolha do aluno.
Cancelar	Cancela a edição de matrícula dos alunos e fecha a tela.	Sem restrições.

Tabela 25. Comandos da tela de parâmetros dos alunos.

4.6.11 Tela de parâmetros das disciplinas

Tela de parâmetros adicionais das disciplinas não contidas no arquivo XML. A disciplina é escolhida na tabela que mostra todas as disciplinas do docente. Após selecionada pode-se alterar o código da disciplina, o número de Horas/Aula por mês e o número de alunos em um ano específico. Os nomes das disciplinas, curso, nível e os anos de cada uma são oriundos do arquivo XML.

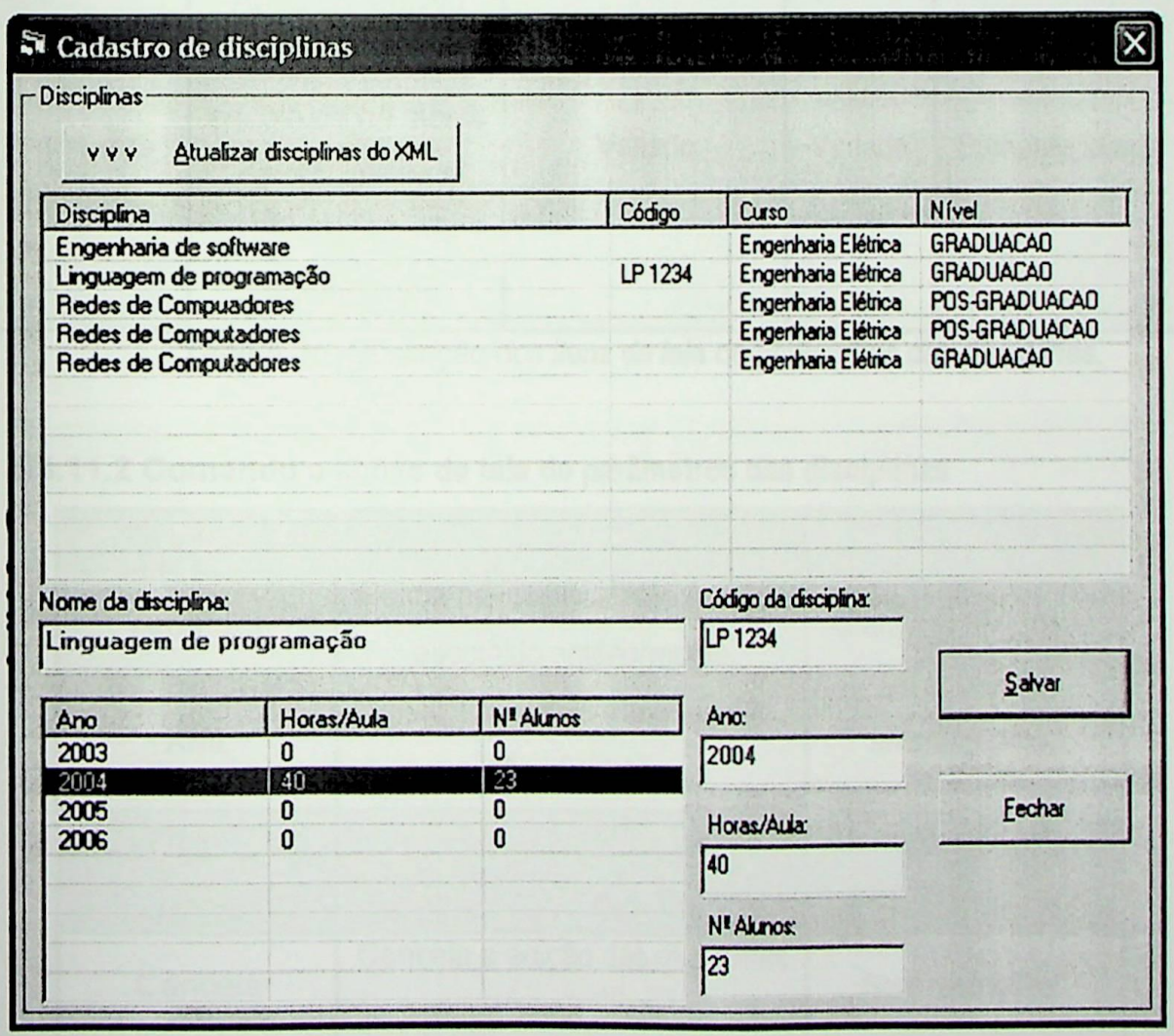

Figura 59. Tela de Cadastro de parâmetros das disciplinas.

4.6.11.1 Descrição dos itens da tela de cadastro de parâmetros das disciplinas

Nome	Valores válidos	Formato	Tipo	Restrições
Nome da disciplina	Caracteres alfanuméricos	150 caracteres	String	Somente leitura
Código da disciplina	Caracteres alfanuméricos	10 caracteres	String	Sem restrições
Ano	Numéricos	4 dígitos (yyyy)	Date	Somente leitura.
Horas/Aula	Numéricos	2 dígitos	Inteiro	2 dígitos
Número de Alunos	Numéricos	3 dígitos	Inteiro	3 dígitos
Tabela de Disciplinas	Disciplinas do arquivo XML	Variado	Variado	Somente leitura
Tabela de anos das disciplinas	Anos do arquivo XML e da disciplina escolhida na tabela de disciplinas e demais valores preenchidos	Variado	Variado	Somente leitura

Tabela 26. Descrição dos itens da tela de parâmetros das disciplinas.

4.6.11.2 Comando e ações da tela de parâmetros das disciplinas

Nome	Ação	Restrições
Atualizar disciplinas do XML	Busca novas disciplinas no XML escolhido anteriormente. Disciplinas já adicionadas não são alteradas.	O arquivo XML deve ser escolhido anteriormente.
Salvar	Salva alterações dos dados da disciplina escolhida e do ano.	Escolha da disciplina e do ano.
Cancelar	Cancela a edição das disciplinas e fecha a tela.	Sem restrições.

Tabela 27. Comandos da tela de parâmetros das disciplinas.

4.6.12 Tela de informações sobre o sistema de software desenvolvido

Figura 60. Tela de informações sobre o sistema de software desenvolvido.

4.6.12.1 Comando e ações da tela de informações sobre o sistema de software

Nome	Ação	Restrições
OK	Fecha a tela de informações	Sem restrições
System Info...	Exibe a tela de informações do sistema operacional Windows	Sem restrições

Tabela 28. Comandos da tela de informações sobre o software.

5. Conclusão

Os docentes da Unifei já possuem cadastro de todas as informações acadêmicas no sistema de software Currículo Lattes. Porém, tal sistema de software emite apenas o currículo completo somente em uma formatação. No caso de se emitir um outro relatório, que exiba apenas algumas atividades acadêmicas específicas, em um determinado período e com outro formato, o docente teria que copiar todos os dados desse sistema ou de outra fonte qualquer e fazer o relatório manualmente, filtrando, complementando e formatando outra vez todos os dados. Para progressão em sua carreira, a Instituição exige relatórios periódicos de produção e atividades realizadas, cujos mesmos têm um formato específico. Para tal tarefa o docente demandaria muito tempo e esforço, tempo esse que poderia estar sendo empregado em outras atividades acadêmicas, trazendo benefícios para o mesmo e para a Instituição. A replicação de informações pode também causar vários erros advindos da redundância de informações, tais como esquecimento ou informações errôneas. A consistência dos dados é importante.

Com o sistema de software proposto, visamos eliminar as falhas no preenchimento dos relatórios, economizar tempo e evitar erros. Ele foi desenvolvido para agilizar e facilitar a emissão de relatórios de produção e de atividades realizadas pelos docentes da Unifei, buscando os dados já contidos no Currículo Lattes. Como nem todas as informações estão contidas neste último, utilizamos uma base de dados complementar. Temos complementos como matricula dos alunos e código das disciplinas entre outras, não contidas no Currículo Lattes. O sistema de software irá gerar então, automaticamente, os relatórios e possibilitar a edição e impressão dos mesmos.

O sistema de software foi desenvolvido utilizando as tecnologias de desenvolvimento e padrões citados nos capítulos anteriores. Através dos conceitos de engenharia de software, como modelagem e processos de desenvolvimento, abordados no capítulo 2, foi possível verificar a viabilidade e modelar o mesmo através da UML (Unified Modeling Language), ter uma visão geral sobre o que seria desenvolvido, e só então, fazer a implementação. Através do capítulo 3, apresentamos as tecnologias e padrões que fizeram parte do desenvolvimento do produto. Foram utilizados XML (Extensible Markup Language), RTF (Rich Text Format), Visual Basic e Microsoft Access.

Através do levantamento dos requisitos, da análise já existente (Currículo Lattes) e da modelagem do novo sistema, conseguimos uma visão geral do problema, verificando que a redundância de dados e de tarefas podiam causar vários erros e perda de tempo.

No capítulo 4, apresentamos o sistema desenvolvido, com todos os requisitos, problemas, dificuldades e etapas vencidas. Mostramos ainda, as interfaces do sistema de software e todos os diagramas desenvolvidos e ainda sua interação com o usuário. Através das telas contidas e explicadas nesse capítulo, podemos entender melhor o sistema e verificar sua funcionalidade.

Com a implementação e implantação do sistema de software, os resultados esperados foram alcançados, pois com simples cliques, o mesmo importa dados de um arquivo XML gerado pelo sistema já existente (Currículo Lattes); cruza os dados com a base de dados complementar e assim pode gerar dois relatórios: o relatório da CPPD (Comissão Permanente de Pessoal Docente) e o relatório da GED (Gratificação de Estimulo a Docência) de forma rápida, sem erros e no formato exigido pela Instituição. Mais detalhadamente, para gerar os relatórios, o docente escolhe o arquivo XML gerado pelo Currículo Lattes, preenche opcionalmente a base de dados complementar (com dados referentes às disciplinas e alunos com que ele trabalha), escolhe o relatório que deseja gerar e o período a que se refere. O sistema automaticamente gera o relatório com todas as informações que encontrar no XML e na base de dados complementar no formato específico. O docente pode então abrir o relatório em um editor de texto qualquer que suporte o formato RTF e fazer ajustes finos, alterações e impressões. Com isso, acreditamos que o sistema de software melhorou e agilizou o trabalho dos docentes e a produção dos relatórios de atividades.

5.1 Trabalhos Futuros

5.1.1 Adição de novos relatórios

O sistema de software elaborado foi documentado e desenvolvido através de técnicas de engenharia de software padronizadas e de fácil manutenção. A adição de novos relatórios pode ser elaborada de forma fácil e rápida, pois, sistemas documentados facilitam a manutenção evolutiva.

Os relatórios possuem um formato RTF o que possibilita criar modelos em vários editores de texto.

5.1.2 Realizar contagem dos pontos de forma automática

Cada atividade realizada pelo docente equivale a um certo número de pontos para progressão na sua carreira. Os pontos variam de acordo com tabelas de pontuação específicas de cada relatório. Tais pontos poderiam ser computados automaticamente através do sistema de software.

5.1.3 Importação/ Exportação, no padrão XML, da base de dados interna

Tal como no Currículo Lattes, também visualiza-se a possibilidade de importar ou exportar no padrão XML, a base de dados interna armazenada no SGBD Access.

5.1.4 Criação de um Sistema Curricular Completo

Vislumbra-se, também, a possibilidade de se criar um outro sistema curricular completo, que engloba todas as atividades dos docentes, inclusive as não contempladas no Lattes. Tal sistema poderia importar ou exportar dados para o Lattes.

REFERÊNCIAS BIBLIOGRÁFICAS:

1. ABRAHAMSSON, Pekka, SALO Outi. *Agile Software Development Methods – Review and Analysis.* Espoo VTT Publications, 2002.

2. BOEHM, B.W. *Spiral Model of software Development and Enhancement.* IEEE Computer, v. 21, n. 5, Maio 1988.

3. BECK, Kent; FOWLER, Martin. *Planning Extreme Programming.* Addison Wesley, USA, 2000.

4. BECK, Kent. *Extreme Programming Explained: Embrace Change.* Addison Wesley, USA, 1999.

5. BENZ, Brian; DURANT, John R. *XML Programming Bible.* Wiley Publishing Inc., New York- NY- USA, 2003.

6. BERNSTEIN, Lawrence. *Trustworthy Systems – Through Quantitative – Software Engineering.* Wiley-IEEE Computer Society Press, 2005.

7. BIRBECK, Mark; KAY, Michael; Et al. *Professional XML – Second Edition.* Peer Information Inc., USA, 2001.

8. BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. *Uml, Guia do usuário*, Rio de Janeiro, Elsevier, 2000.

9. CANTOR, Murray R. *Object-Oriented Project Management with UML.* New York: Ed. John Wiley & Sons, 1998.

10. COAD, P.; YOURDON, E. *Análise baseada em objetos*, 2ª edição, Rio de Janeiro, Editora Campus, 1992.

11. COAD, Peter; LEFEBVRE, Eric; DE LUCA, Jeff. *Java Modeling in Color with UML.* Prentice Hall, USA, 1999.

12. COCKBURN, Alistair. *Agile Software Development.* Addison Wesley, USA, 2001.

13. DEITEL, Harvey M.; DEITEL, Paul J.; et al. *XML How to Program.* Prentice Hall, USA, 2000.

14. FOWLER, Martin; SCOTT, Kendall. *UML Essencial: Um breve guia para a linguagem-padrão de modelagem de objetos.* Porto Alegre: Ed. Bookman, 2001.

15. HAROLD, Elliotte Rusty. *The XML Bible – 2th edition.* Wiley Publishing Inc., New York- NY- USA, 2001.

16. HOLZNER, Steven. *Visual Basic 6 Black Book.* Coriolis Group Books, USA, 1998.

17. JEFRIES, Ron; ANDERSON, Ann; HENDRICKSON, Chet. *Extreme Programming Installed.* Addison- Wesley. 2001

18. LAUDON, Kenneth C.; LAUDON, Jane P. *Management information systems: new approaches to organization & technology. 5 Edition*, USA, Prentice Hall Inc., 1998.

19. LEACH, Ronald J. *Introduction to Software Engineering.* Florida: CRC Press LLC, 2000.

20. LEFFINGWELL, Dean; WIDRIG, Don. *Managing Software Requirements: A Use Case Approach, Second Edition.* Addison Wesley, Toronto- Canada, 2003.

21. MILLS, Harlan D. *The Management of Software Engineering.* IBM System Journals, vol. 38, no. 2 e 3, p. 289-295, 1999.

22. NBR 12207. *Tecnologia de informação – Processos de ciclo de vida de software.* ABNT, Rio de Janeiro - RJ, 1998.

23. PAULA W. P. F. *Manual do engenheiro de Software – Métodos Gerenciais.* DDC – UFMG. 2000.

24. PENDER, Tom. *UML Bible.* Wiley Publishing Inc., New York - NY - USA, 2003.

25. PETROUTSOS, Evangelos. *Dominando o Visual Basic 6: a Bíblia.* Makron Books, São Paulo, Brasil, 1999.

26. PILONE, Dan; PITMAN, Neil. *UML 2.0 in a Nutshell.* O'Reilly Media, Sebastopol- CA - USA, 2005.

27. POLLONI, Elliotte Rusty. *XML Bible.* New York, IDG Books, 2001.

28. PRAGUE, Cary N.; IRWIN, Michael R.; REARDON, Jennifer. *Access 2003 Bible.* Wiley Publishing Inc., Hoboken- NJ- USA, 2003.

29. PRESSMAN, Roger S. *Engenharia de Software.* São Paulo, Makron Books, 1995.

30. REED, Paul R. Jr. *Desenvolvendo Aplicativos com Visual Basic e UML.* Makron Books, São Paulo, Brasil, 2000.

31. REZENDE, Denis A. *Engenharia de Software e Sistemas de Informação.* Rio de Janeiro: Brasport, 2002. p.122-151.

32. ROYCE, W. *Software Project Management A Unified Framework*, Addison Wesley, 1998.

33. RUMBAUGH, James. *Modelagem e projetos baseados em objetos.* Rio de Janeiro, Campus, 1994.

34. SCHNEIDER, G.; WINTERS, J. P.; JACOBSON I. *Applying Use Cases: A Pratical Guide.* Addison-Wesley, 1998.

35. SCHWABER, Ken; Beedle, M. *Agile Software Development with Scrum.* Prentice Hall, 2001.

36. SOMMERVILLE, Ian. *Engenharia de Software- 6ª Edição.* São Paulo, Editora Pearson Education do Brasil, 2003.

37. SOMMERVILLE, Ian. *Software Engineering – 7th edition.* Addison Wesley, USA, 2004.

38. STEPHENS, Matt; ROSENBERG, Doug. *Extreme Programming Refactored: The case Against XP.* Apress, USA, 2003.

39. STURM, Jake. *Vb6. Design and Development.* Wrox Press Ltd, Birmingham-UK, 1999.

40. SUTHERLAND, Jeff. *Sutherland's COMDEX/Object World Tutorial – SCRUM.* 1998. Disponível em: <http://www.jeffsutherland.org/objwld98/ow_scrum.html>. Acesso em 15 mar. 2003.

41. THIRY, Marcello. *Processo de Desenvolvimento de Software com UML*, 2001. Disponível em: <http://www.eps.ufsc.br/disc/procuml/>. Acesso em: 29 abr.2005.

42. WILLIAMS, Kevin. *Professional XML Databases.* Wrox Press Ltd, Birmingham-UK, 2000.

43. YANG, Hongji. *Software Evolution with UML and XML.* Idea Group Publishing, Hershey- USA, 2004.

ANEXOS

ANEXO I

Formulário para solicitação da GED e modelo utilizado pelo software.

UNIVERSIDADE FEDERAL DE ITAJUBÁ – UNIFEI				
CIAG – UNIFEI	FORMULÁRIO PARA SOLICITAÇÃO DA GED			Ano:
Docente:				Título:
Instituto:	Departamento:	Classe:	Nível:	Reg. Trab:

Grupo I – Atividades de ensino (Máximo de 120 pontos)

DISCIPLINAS MINISTRADAS DURANTE O ANO			
Sigla	Nome	Nível[1]	Horas aula no ano[2]

1 – GR (Graduação) / ESP (Especialização) / MÊS (Mestrado) / DR (Doutorado) / EXT (Extensão). 2 – Número total de horas-aula lecionadas pelo professor nesta disciplina durante o ano. Não é o número de horas-aula semanais.

Grupo I – Atividades de ensino (continuação...)

ORIENTAÇÃO E SUPERVISÃO DE ALUNOS					
Matrícula	Nome do aluno	Tipo[3]	Or/Cor[4]	Início[5]	Término[5]

3 – IC(Iniciação Científica) / TD (Trabalho de Diploma) / ES (estágio supervisionado) / MES (Mestrado) / DR (Doutorado) / ESP (Especialização). 4 – OR (Orientador) / COR (Co-orientador). 5 – Indicar apenas mês/ano.

Grupo II – Produção Intelectual (Máximo de 60 pontos)

PRODUÇÃO BIBLIOGRÁFICA PUBLICAÇÕES				
Título:				
Tipo[6]:	Editora/Revista/Anais/Instituição:			
Data:	Nº de autores:	Principal (S/N):	Indexada (S/N):	Internacional (S/N):
Título:				
Tipo[6]:	Editora/Revista/Anais/Instituição:			
Data:	Nº de autores:	Principal (S/N):	Indexada (S/N)	Internacional (S/N):
Título:				
Tipo[6]:	Editora/Revista/Anais/Instituição:			
Data:	Nº de autores:	Principal (S/N):	Indexada (S/N)	Internacional (S/N):

6 – LI (Livro) / CL (Capítulo de livro) /AR (Artigo em Revista) / TA (Artigo em anais) / AP (Apostila completa). Atenção! Artigos ainda não computados para GED, publicados no ano imediatamente anterior, podem ser computados para a GED deste ano.

Grupo II – continuação...

PRODUÇÃO TÉCNICA PARTICIPAÇÃO EM EVENTOS TÉCNICOS-CIENTÍFICOS COM APRESENTAÇÃO DE TRABALHOS[7]	
Título do trab.:	Data:
Evento:	Internacional (S/N):
Título do trab.:	Data:
Evento:	Internacional (S/N):

7 – Congresso / Seminário / Workshop / Palestra / Conferência. OBS. Publicação e apresentação de trabalhos são pontuados em separado. Assim, quando for o caso, lance nos dois quadros.

Grupo II – continuação...

OUTRAS PRODUÇÕES TÉCNICAS[8]	
Título do trab.:	Data:
Título do trab.:	Data:

8 – Desenvolvimento de sistema de software produto. Desenvolvimento de material didático e instrucional. Desenvolvimento de outros produtos. Desenvolvimento de novas técnicas, e Relatório final de projeto de pesquisa.

Grupo III – Atividades de Pesquisa e Extensão (Máximo de 30 pontos)

PROJETOS DE PESQUISA COM REGISTRO NA CPq[9]				
Título:				
Função[10]:	Registro CPq (S/N):	Início:	Término:	Agência financ.:
Título:				
Função[10]:	Registro CPq (S/N):	Início:	Término:	Agência financ.:

9 – Não considerar trabalhos de iniciação científica como pesquisa de docente. 10 – **Coord** (Coordenador de projetos de pesquisa) / **Pesq** (Pesquisador em projetos de pesquisa). OBS. Lançar dados de **projetos de pesquisa** com e sem financiamento.

Grupo III – continuação...

ATIVIDADES DE EXTENSÃO[11]
Descrição

11 – Inclui coordenação de convênios e de projetos de cooperação científica. Exige-se registro na UNIFEI. É vedada a inclusão de atividades notadamente caracterizadas como de prestação remunerada de serviços.

Grupo IV – Atividades de Qualificação (Máximo de 140 pontos)

ATIVIDADES DE QUALIFICAÇÃO *Stricto Sensu*		
Nível (MÊS/DR/PD):	Instituição:	Dispensa (T/P):[12]
Relatório aprovado pela CCD/PPG? (S/N):	Início:	Término:

12 – Dispensa total (T) ou parcial (P) das atividades de ensino. Para a obtenção dos pontos, exige-se relatório aprovado pela CCD.

Grupo V – Atividades Administrativas e de Representação (Máximo de 20 pontos)

ATIVIDADES DE APOIO ADMINISTRATIVO[13]			
Função/Cargo	Órgão	Início	Término

13 – Coordenação: geral de laboratórios, de GEPE's, de Trabalhos de Diplomas, e de eventos institucionais. Chefia de laboratório. Presidência de seções ou equivalente, em eventos técnicos/científicos. Participação em comissões permanentes. Participação em outras comissões. Sub-Chefia de departamento. Consultoria Adhoc. Direção de órgão da UNIFEI, e outras atividades administrativas não consideradas no Grupo VIII.

Grupo V – continuação ...

ATIVIDADES DE REPRESENTAÇÃO[14]			
Função/Cargo	Órgão/entidade	Início	Término

14 – Representação acadêmica. Participação em órgãos colegiados. Representação sindical. Participação não remunerada em conselhos ou comissões de órgãos governamentais e de entidades científicas, culturais e profissionais.

Grupo VI – Avaliação Qualitativa das Atividades de Ensino (Máximo de 10 pontos)

AVALIAÇÃO QUALITATIVA[15]
Descrição

15 – Através de projeto institucional de avaliação do desempenho docente pelos discentes

Grupo VII – Outras Atividades (Máximo de 10 pontos)

PARTICIPAÇÃO EM BANCAS EXAMINADORAS		
Local	Data	Tipo[16]

16 – TD (de trabalho de diploma) / ESP (de avaliação da monografia final) / MES (Mestrado) / DR (Doutorado) / QD (Exame de qualificação de doutorado) / CP (Concurso público) e LD (Livre docência)

Grupo VII – continuação ...

ORIENTAÇÃO E SUPERVISÃO DE ALUNOS				
Matrícula	Nome do aluno	Tipo[17]	Início	Término

17 – Alunos em Monitoria (MON), Estagiários oficiais (EST) e Outros trabalhos de Alunos com bolsa de trabalho (BT).

Grupo VII – continuação ...

OUTRAS ATIVIDADES[18]
Descrição

18 – Cursos de qualificação não incluídos no grupo IV. Palestras proferidas. Participação como "referee", etc.

Grupo VIII – Atividades Administrativas Previstas no § 1° do art. 4° da Lei n° 9.678 (Máximo de 84 pontos)

ATIVIDADES ADMINISTRATIVAS[19]				
Função/Cargo	FG/CD/outra	Órgão	Início	Término

19 – Liste aqui apenas as seguintes atividades: Cargo de direção (CD). Cargo de Chefia com função gratificada (FG). Coordenação de curso de graduação e pós-graduação. Presidência de comissão institucional, prevista em Lei (por exemplo, CCD, CPPD, etc). Atenção! Se ocupar mais de um cargo, dos previstos neste grupo, para efeito de contagem de pontos considere apenas um deles.

ASSINATURAS

Do professor:
Solicito a avaliação de minhas atividades com vistas à concessão da gratificação de estímulo à docência.
Declaro que as informações acima são verdadeiras.

Data ___/___/___ Assinatura do docente:

_____

Do chefe do departamento:
De acordo.

Data ___/___/___ Assinatura do chefe do departamento:

_____

ANEXO II

Modelo e Exemplo de relatório CPPD

Relatório de Atividades

Este relatório destina-se a reportar à Comissão de Avaliação de Desempenho para Progressão na Carreira Docente, as minhas atividades como Professor PREENCHER, no período compreendido entre 01 de janeiro de 2000 a 18 de novembro de 2005. Enumero, a seguir, as atividades classificadas por categorias.

1- Capacitação [*xx pt*]

1.1- Mestrado [*xx pt*]

1.1.1- Afastamento em tempo integral

1.1.2- Afastamento em tempo parcial

1.1.3- Disciplinas cursadas

- Pós-Doutorado em andamento na Universidade Federal do Rio de Janeiro - . [xx pt]
  PREENCHER QUANTIDADE DE ANOS CURSADOS

1.1.4- Pela obtenção da titulação

- Mestrado concluido no curso de Engenharia Elétrica da Universidade Federal de Itajubá - Título: "Produção de Sistema de software com UML" - 2003. [xx pt]

1.2- Doutorado [*xx pt*]

1.2.1- Afastamento em tempo integral

1.2.2- Afastamento em tempo parcial

1.2.3- Disciplinas cursadas

1.2.4- Pela aprovação no exame de qualificação

1.2.5- Pela obtenção da titulação

- Doutorado concluído no curso de Administração da Universidade de São Paulo - Título: "Administração de Empresas de Software" - 2005. [xx pt]

1.3- Pós-Doutorado [*xx pt*]

1.4- Cursos [*xx pt*]

1.5- Estágios [*xx pt*]

2- Orientações [*xx pt*]

2.1- Doutorado [*xx pt*]

2.1.1- Por orientado

- Orientado enrolado (Matr. PREENCHER) - XXX/2002 a XXX/2005 [xx pt]
- Total Pereira Ferreira (Matr. PREENCHER) - XXX/2004 a XXX/2005 [xx pt]
- Orientado de Doutorado Silva (Matr. PREENCHER) - XXX/2003 a XXX/2005 [xx pt]

2.1.2- Por tese concluída

- Total Pereira Ferreira (Matr. PREENCHER) - XXX/2004 [xx pt]
- Orientado de Doutorado Silva (Matr. PREENCHER) - XXX/2003 [xx pt]

2.2- Mestrado [*xx pt*]

2.2.1- Por orientado

- Martin Ferrari (Matr. PREENCHER) - XXX/2002 a XXX/2005 [xx pt]
- Autor DESCONHECIDO (Matr. PREENCHER) - XXX/2001 a XXX/2005 [xx pt]
- Dissertado da Silva (Matr. PREENCHER) - XXX/2004 a XXX/2005 [xx pt]
- Fabio Orientado (Matr. 2424) - XXX/2003 a XXX/2005 [xx pt]
- Felipe Jonas Oliveira (Matr. PREENCHER) - XXX/2000 a XXX/2005 [xx pt]
- Ciclano Pereira (Matr. PREENCHER) - XXX/2000 a XXX/2005 [xx pt]

2.2.2- Por dissertação concluída

- Dissertado da Silva (Matr. PREENCHER) - XXX/2004 [xx pt]
- Fabio Orientado (Matr. 2424) - XXX/2003 [xx pt]
- Felipe Jonas Oliveira (Matr. PREENCHER) - XXX/2000 [xx pt]
- Ciclano Pereira (Matr. PREENCHER) - XXX/2000 [xx pt]

2.3- Trabalho de diploma [*xx pt*]

2.3.1- Por TD em andamento

- Ciclano Pereira (Matr. PREENCHER) - XXX/2005 a XXX/2005 [xx pt]
- TCC Oliveira (Matr. PREENCHER) - XXX/2000 a XXX/2005 [xx pt]

2.3.2- Por TD concluído

- TCC Oliveira (Matr. PREENCHER) - XXX/2000 [xx pt]

2.4- Pesquisa de iniciação cientifica [*xx pt*]

2.4.1- Por pesquisa em andamento

- Felipe Jonas Oliveira (Matr. PREENCHER) - XXX/2001 a XXX/2005 [xx pt]
- Total Pereira Ferreira (Matr. PREENCHER) - XXX/2003 a XXX/2005 [xx pt]
- Iniciação Martins Sauro (Matr. PREENCHER) - XXX/2002 a XXX/2005 [xx pt]

2.4.2- Por pesquisa concluída

- Iniciação Martins Sauro (Matr. PREENCHER) - XXX/2002 [xx pt]

2.5- Monografia [*xx pt*]

2.5.1- Por monografia em andamento

- TCC Oliveira (Matr. PREENCHER) - XXX/2004 a XXX/2005 [xx pt]
- Iniciação Martins Sauro (Matr. PREENCHER) - XXX/2004 a XXX/2005 [xx pt]
- Martin Ferrari (Matr. PREENCHER) - XXX/2005 a XXX/2005 [xx pt]
- Orientado de Monografia Pereira (Matr. PREENCHER) - XXX/2000 a XXX/2005 [xx pt]

2.5.2- Por monografia concluída

- Martin Ferrari (Matr. PREENCHER) - XXX/2005 [xx pt]
- Orientado de Monografia Pereira (Matr. PREENCHER) - XXX/2000 [xx pt]

2.6- Estágio na Unifei [*xx pt*]

2.6.1- Por estagiário em andamento

- Ciclano Pereira (Matr. PREENCHER) - XXX/2003 a XXX/2005 [xx pt]

2.6.2- Por relatório de estágio aprovado

- Ciclano Pereira (Matr. PREENCHER) - XXX/2003 [xx pt]

2.7- Monitor [*xx pt*]

3- Produção Científica [*xx pt*]

3.1- Livros [*xx pt*]

- Livro Publicado 1. Editora 1. SP. 2003 [xx pt]

3.2- Capítulo de livro [*xx pt*]

- Capítulo Capitulo do Livro 2. Do livro Livro Publicado 2. Editora 2. RJ. 2000 [xx pt]

3.3- Tradução de livro [*xx pt*]

3.4- Artigos técnicos e científicos [*xx pt*]

3.4.1- Em periódicos e revistas de circulação internacional

3.4.1.1- Com árbitro

3.4.1.2- Sem árbitro

3.4.2- Em periódicos e revistas de circulação nacional

3.4.2.1- Com árbitro

- Artigo Publicado 4, - Anais da Academia Brasileira de Ciências, vol.1, 1 - 2004 [xx pt]
- Artigo Publicado 3, - Infocapes, vol.1, 1 - 2003 [xx pt]
- Artigo Publicado 2, br - Periodico 1, vol.1, 1 - 2000 [xx pt]
- Artigo publicado 1, Rio de janeiro - Matematica aplicada e computacional, vol.222222, 2222 - 2002 [xx pt]

3.4.2.2- Sem árbitro

3.4.3- Em anais de congressos, seminários e simpósios de relevância internacional
- Completo Internacional, INT-EVENT - Dallas - 2003 [xx pt]
- Resumo de Anais, INT-EVENT - San Marino - 2002 [xx pt]

3.4.4- Em anais de congressos, seminários e simpósios de relevância nacional
- Trabalho Publicado Completo, Evento Regional - São Lourenço - 2002 [xx pt]
- Trabalho em evento, EVENTO NAC - ITAJUBÁ - 2002 [xx pt]

3.5- Apostila completa de disciplina na Unifei [*xx pt*]

3.6- Disciplinas preparadas utilizando novas técnicas [*xx pt*]

3.7- Resumo em anais de congresso [*xx pt*]

3.7.1- De relevância internacional

3.7.2- De relevância nacional

3.8- Pesquisa registrada na PPG, que envolva financiamento de órgãos governamentais ou de terceiros **[*xx pt*]**

3.8.1- Coordenador
- Relatório Técnico - xxx/2003 a xxx/2003 [xx pt]

3.8.2- Pesquisador
- Pesquisa em Sistema de software de controle de tensão - dezembro de 2000 [xx pt]
  VERIFICAR POSICIONAMENTO DA PESQUISA
- Pesquisa em Afundamentos de tensão - dezembro de 2000 [xx pt]
  VERIFICAR POSICIONAMENTO DA PESQUISA

3.8.3- Conclusão com relatório aprovado pelo órgão financiador
- Relatório Técnico - xxx/2003 a xxx/2003 [xx pt]

3.9- Pesquisa registrada na PPG, que não envolva nenhum tipo de financiamento **[*xx pt*]**

3.9.1- Coordenador, com relatório semestral aprovado pela CPq

3.9.2- Pesquisador, com relatório semestral aprovado pela CPq

3.9.3- Conclusão com relatório aprovado pela CPq

3.10- Obtenção de patente ou registro de sistema de software junto aos órgãos competentes [*xx pt*]

- **Sistema de software Sistema de software Multimidia. 2005 [xx pt]**
  VERIFICAR
- **Sistema de software Sistema de software Computacional. 2004 [xx pt]**
  VERIFICAR

4- Participação e organização de eventos oficiais [*xx pt*]

4.1- Organização de seminários, congressos e eventos de mesma natureza [*xx pt*]

4.1.1- De relevância internacional

4.1.2- De relevância nacional

4.1.1- De relevância regional/local

4.2- Coordenação de sessões em seminários, congressos e eventos de mesma natureza [*xx pt*]

4.2.1- De relevância internacional

4.2.2- De relevância nacional

4.2.1- De relevância regional/local

4.3- Participação em seminários, congressos e eventos de mesma natureza (pontos por trabalho apresentado) [*xx pt*]

4.3.1- De relevância internacional

4.3.2- De relevância nacional

4.3.1- De relevância regional/local

4.4- Participação em seminários, congressos e eventos de mesma natureza (sem apresentação de trabalhos) [*xx pt*]

4.4.1- De relevância internacional

4.4.2- De relevância nacional

4.4.1- De relevância regional/local

5- Atividades administrativas [*xx pt*]

5.1- Direção geral da Unifei [*xx pt*]

5.1.1- Diretor geral, Vice-diretor, Pró-Diretores e diretores de instituto

- Diretor de Instituto - dezembro/2004 a novembro/2005 [xx pt]

VERIFICAR POSICIONAMENTO DA ATIVIDADE ADMINISTRATIVA

5.1.2- Assessores

5.2- Presidências de comissões permanentes da Unifei [*xx pt*]

5.3- Chefias de departamento [*xx pt*]

5.4- Participação em comissões permanentes da Unifei [*xx pt*]

5.5- Coordenação geral da comissão de vestibular da Unifei [*xx pt*]

5.6- Coordenação dos cursos de graduação e pós-graduação [*xx pt*]

5.7- Coordenador de grupo de ensino, pesquisa e extensão [*xx pt*]

5.8- Coordenador de trabalho de diploma [*xx pt*]

5.9- Representação oficial da Unifei em: [*xx pt*]

5.9.1- Órgãos de fomento

5.9.2- Conselhos, associações ou comissões externas

5.10- Participação na congregação da Unifei [*xx pt*]

- **Cargo exercido - janeiro/2004 a novembro/2005 [xx pt]**

5.11- Coordenação e chefia de laboratórios na Unifei [*xx pt*]

5.11.1- Coordenação geral de laboratório

5.11.2- Chefia de laboratório

6- Atividades de ensino vinculadas à instituição [*xx pt*]

- Redes de Compuadores (2004) - 0 ha - 0 alunos. [xx pt]
- Redes de Compuadores (2005) - 0 ha - 0 alunos. [xx pt]
- Redes de Computadores (2004) - 0 ha - 0 alunos. [xx pt]
- Redes de Computadores (2005) - 0 ha - 0 alunos. [xx pt]
- Engenharia de sistema de software (2003) - 0 ha - 0 alunos. [xx pt]
- Engenharia de sistema de software (2004) - 0 ha - 0 alunos. [xx pt]
- Engenharia de sistema de software (2005) - 0 ha - 0 alunos. [xx pt]
- LP 1234 Linguagem de programação (2003) - 0 ha - 0 alunos. [xx pt]
- LP 1234 Linguagem de programação (2004) - 40 ha - 23 alunos. [xx pt]
- LP 1234 Linguagem de programação (2005) - 0 ha - 0 alunos. [xx pt]
- Redes de Computadores (2002) - 0 ha - 0 alunos. [xx pt]
- Redes de Computadores (2003) - 0 ha - 0 alunos. [xx pt]

7- Atividades de extensão da instituição [*xx pt*]

- Atividade de Extensão na Unifei. Instituto de Engenharia Elétrica. 2000 [xx pt]

8- Atividades especiais [*xx pt*]

8.1- Atividades em laboratórios da Unifei **[*xx pt*]**

8.1.1- Projeto e implementação

8.1.2- Ampliação e reestruturação

8.2- Participação em bancas examinadoras [*xx pt*]

8.2.1- Doutorado

- João Candidato. Universidade Federal de Itajubá. Engenharia Elétrica. 2003 [xx pt]

8.2.2- Qualificação de doutorado

- Candidato Banca de Qualificação. . . 2002 [xx pt]

8.2.3- Mestrado

- Aluno Mestrado 2. . . 2003 [xx pt]
- Candidato Martins. Universidade Federal de Itajubá. Engenharia de Produção. 2001 [xx pt]

8.2.4- Concurso público para magistério superior

- Banca Livre-docência. Livre-docência. 2003 [xx pt]
  VERIFICAR
- Banca de Concurso Público. Concurso público. xxx/2003 [xx pt]
  VERIFICAR
- Banca de Professor titular. Professor titular. 2002 [xx pt]
  VERIFICAR

8.2.5- Seleção de candidatos para cursos em pós-graduação

8.2.6- Trabalho de diploma

- Aluno de graduação. . . 2004 [xx pt]

8.2.7- Relatório técnico ou de estágio

8.3- Participação como revisor [*xx pt*]

8.3.1- Livro

8.3.1.1- De relevância internacional

8.3.1.2- De relevância nacional

8.3.2 Capítulo de livro

8.3.2.1- De relevância internacional

8.3.2.2- De relevância nacional

8.3.3- Artigos técnicos e científicos em periódicos e revistas

8.3.3.1- De relevância internacional

8.3.3.2- De relevância nacional

8.3.4- Artigos técnicos e científicos em anais de congressos e seminários

8.3.4.1- De relevância internacional

8.3.4.2- De relevância nacional

8.3.5- Resumo de artigos técnicos e científicos em anais de congressos e seminários

8.3.5.1- De relevância internacional

8.3.5.2- De relevância nacional

8.3.6- Consultor Ad-hoc de órgão de fomento à pesquisa

9- Outras atividades [*xx pt*]

- Livro 3 - 2005 [xx pt]

10- Total de pontos [*xx pt*]

PREENCHER TODOS OS ITENS ANTERIORES COM AS DEVIDAS PONTUAÇÕES, BEM COMO ESTE ITEM COM A SOMA DAS PONTUAÇÕES ANTERIORES.

Fulano da Silva