UFSM

Monografia de Graduação

UMA MODELAGEM ORIENTADA A OBJETOS PARA O AMBIENTE AMEM

Daniel Biasoli

Curso de Ciência da Computação

Santa Maria, RS, Brasil

2004

UMA MODELAGEM ORIENTADA A OBJETOS PARA O AMBIENTE AMEM

por

Daniel Biasoli

Monografia de Graduação apresentada ao Curso de Ciência da Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para a obtenção do grau de
Bacharel em Ciência da Computação

Curso de Ciência da Computação

Trabalho de Graduação nº 168

Santa Maria, RS, Brasil

2004

Universidade Federal de Santa Maria
Centro de Tecnologia
Curso de Ciência da Computação

A Comissão Examinadora, abaixo assinada,
aprova a Monografia de Graduação

UMA MODELAGEM ORIENTADA A OBJETOS PARA O AMBIENTE AMEM

elaborada por
Daniel Biasoli

como requisito parcial para a obtenção do grau de
Bacharel em Ciência da Computação

COMISSÃO EXAMINADORA:

_____
Felipe Martins Müller
(Professor Orientador)

_____
Marcos Cordeiro d'Ornellas

_____
Oni Reasilvia Sichonany

Santa Maria, março de 2004

SUMÁRIO

LISTA DE FIGURAS

RESUMO

Monografia de Graduação
Curso de Ciência da Computação
Universidade Federal de Santa Maria, RS, Brasil

UMA MODELAGEM ORIENTADA A OBJETOS PARA O AMBIENTE AMEM

Autor: Daniel Biasoli
Orientador: Felipe Martins Müller

A engenharia reversa e a reengenharia são técnicas muito eficientes para que se possa oferecer a sistemas legados uma documentação comprovada para problemas comuns. As aplicações dessas técnicas, combinadas à orientação a objetos possibilita, com facilidade, a construção de sistemas altamente flexíveis, adaptáveis e extensíveis, fazendo com que o sistema em questão possua uma coleção rica de mecanismos composicionais para formação de classes, instanciação de objetos, propriedades de herança, polimorfismo e ocultamento de informações, que estão presentes em linguagens de programação orientadas a objetos, as quais provêem apoio para criar sistemas que exibem um alto grau de reuso e facilidade de manutenção. Nesses moldes, utilizou-se no decorrer deste trabalho, técnicas de engenharia reversa e reengenharia, para que se pudesse chegar a um modelo relacional e posteriormente um modelo orientado a objetos do Ambiente AMEM.

1 INTRODUÇÃO

A Tecnologia da Informação (TI) tem se tornado, dentro de um curto intervalo de tempo, em uma das bases da sociedade industrial moderna.

Como parte integrante do desenvolvimento de uma sociedade, o processo ensino-aprendizagem sentiu a influência da TI e vem passando, nos últimos anos, por uma pequena revolução devido à utilização dos computadores como mediadores da educação.

Desta maneira, o computador pode ser usado de forma mais ativa, auxiliando o aluno a visualizar situações dinâmicas que descrevam evoluções temporais de imagens, funções ou parâmetros.

A premissa pedagógica mais utilizada em Educação à Distância (EAD) é baseada no construtivismo [2] que assume fundamentalmente a idéia de que o indivíduo é agente ativo de seu próprio conhecimento; isto é, ele constrói significados e define o seu próprio sentido e representação da realidade de acordo com suas experiências e vivências em diferentes contextos.

Desta forma, ao aluno é possibilitada a decisão sobre quais tópicos e subtópicos do domínio serão explorados, além dos métodos de estudo e das estratégias para a solução de problemas. Também se oferecem múltiplas representações dos fenômenos e problemas estudados, possibilitando que os participantes avaliem soluções alternativas e testem suas decisões. Do mesmo modo, envolve-se a

aprendizagem em contextos realistas e relevantes, ou seja, mais autênticos em relação às tarefas de aprendizagem e coloca-se o professor no papel de consultor que auxilia os participantes a organizarem seus objetivos e caminhos na aprendizagem. Contudo, envolve-se a aprendizagem em experiências sociais que reflitam a colaboração entre professores-alunos e alunos-alunos [2].

Havendo a preocupação de desenvolver um software para educação baseado na investigação-ação – não envolvendo apenas os preceitos dos meios onde esse software seria empregado e o público alvo do mesmo – pode-se evitar uma maior exclusão social, tanto no que diz respeito ao analfabetismo tecnológico quanto ao fator financeiro, já que uma aplicação mal projetada poderia exigir tecnologias de suporte que inviabilizem seu uso pela maioria. De tal maneira, um grupo multidisciplinar de professores e alunos da Universidade Federal de Santa Maria desenvolveu um Ambiente Multimídia para Educação Mediada por computador na perspectiva da investigação-ação educacional, chamado de ambiente AMEM [3].

Atualmente, o AMEM opera baseado em um modelo de dados desenvolvido de forma procedimental. Deseja-se transformá-lo em um modelo de dados orientado a objetos, utilizando-se técnicas de engenharia reversa.

A implementação atual do AMEM utiliza uma modelagem relacional. A fim de aumentar a produtividade dos programadores envolvidos com o Ambiente e a modularidade do sistema, é indispensável a construção de um modelo orientado a objetos.

Na área de bancos de dados, sabe-se que os modelos de dados clássicos como, por exemplo, o modelo relacional, não são apropriados para descrever e manipular os dados das chamadas "novas aplicações", tais como projeto assistido por computador (**CAD**[1]), manufatura de produtos (**CAM**[2]), produção de software (**CASE**[3]), automação de escritórios (**OIS**[4]), aplicações médicas e científicas, representação do conhecimento para aplicações de inteligência artificial, etc. Os dados dessas aplicações são muito complexos e evolutivos, sendo necessária a modelagem não somente de sua estrutura, mas também de seu comportamento [7].

Para desenvolver o MOO[5] do AMEM, é necessário estudar a fundo os preceitos de Engenharia Reversa e Reengenharia, amplamente utilizados na evolução deste trabalho.

Segundo Chikofsky *et al.* [5], o termo *engenharia reversa* teve origem na análise de hardware. Os autores afirmam que engenharia reversa, usada com tecnologia de desenvolvimento de software, pode fornecer ganhos significativos em termos de produtividade. Define-se engenharia reversa como um processo de análise de um sistema existente que identifica seus componentes e os representa em um nível mais alto de abstração.

A engenharia reversa tem um ótimo potencial de retorno econômico e é importante organizar e disseminar essas técnicas a fim de oferecer uma documentação comprovada para problemas comuns.

---

[1] Computer Aided Design
[2] Computer Aided Manufature
[3] Computer-Aided Software Engineering
[4] Office Information Systems
[5] Modelo Orientado a Objetos

Baseando-se nestes conceitos e nas atuais funcionalidades do ambiente AMEM, este trabalho tem por objetivo utilizar conceitos de engenharia reversa e de reengenharia a fim de, a partir do sistema legado do AMEM, conduzí-lo para um processo de reengenharia no contexto da orientação a objetos. Com sua aplicação a sistemas legados procedimentais não se consegue realizar plenamente o processo de engenharia reversa. Neste trabalho, uma Família de Padrões de Reengenharia será elaborada para conduzir esse processo a partir de sistemas legados procedimentais para sistemas alvos orientados a objetos.

A fase de implementação, da reengenharia, não será abordada neste trabalho, visto que o objetivo do mesmo se encontra na modelagem orientada a objetos. Abordaremos, portanto, parcialmente a reengenharia do AMEM.

2 REVISÃO DE LITERATURA

Os objetivos específicos deste trabalho estão concentrados primeiramente em agrupar padrões que extraiam informações a partir dos dados e do código fonte do sistema legado, gerando o Modelo Entidade Relacionamento (visão procedimental dos dados) e o Modelo de Análise do Sistema Atual – Diagrama de Pseudo-Classes (visão orientada a objetos dos dados), agrupando padrões para obter a funcionalidade do sistema, criando modelos que recuperem as regras predefinidas para a construção do ambiente AMEM contidas no sistema desenvolvido. Posteriormente, esses padrões habilitam o engenheiro de software a obter um entendimento detalhado dos componentes (partes) do sistema de software, aprofundando, assim, sua compreensão sobre o sistema legado. Desta maneira, poderemos obter de uma forma mais facilitada os cenários do sistema, a construção de diagramas de uso de casos, a elaboração da descrição de casos de usos e o tratamento de anomalias.

Por fim, deseja-se modelar o sistema, utilizando-se orientação a objetos, agrupando padrões para obter o diagrama de classes e os diagramas de seqüência do sistema, através da interação dos produtos obtidos pelas etapas anteriores. Estes padrões habilitam o engenheiro de software a obter o Modelo de Análise do Sistema orientado a objetos, servindo de suporte ao processo de engenharia avante. Fazem parte desta etapa de desenvolvimento do trabalho os seguintes

padrões: construir os diagramas de seqüência, definir classes, definir atributos, analisar hierarquias e definir métodos.

Neste capítulo, serão apresentados os principais conceitos que envolvem a proposta deste trabalho. Primeiramente, são explicados os conceitos e metodologias empregados em Educação Mediada por Computador para utilização no ambiente AMEM. Posteriormente, na seção 2.2, são explicados os conceitos de engenharia reversa e reengenharia. Na seção 2.3 é apresentado o AMEM e demonstrado o seu estado de desenvolvimento atual, bem como suas funcionalidades. Na seção 2.4, serão apresentados o modelo entidade-relacionamento do estado atual do Ambiente AMEM, abstraído através dos conceitos de Engenharia Reversa, e o modelo entidade-relacionamento modificado, projetado para que se possa dar robustez à modelagem orientada a objetos que será desenvolvida para o AMEM. Por fim, na seção 3.5, serão apresentados conceitos de Modelagem Orientada a Objetos, bem como **UML**[6] e Metodologia de Projeto.

2.1 AMEM e educação mediada por computador

A inclusão social pressupõe formação para a cidadania, o que significa que as tecnologias de informação e comunicação devem ser utilizadas também para a democratização dos processos sociais, para fomentar a transparência de políticas e ações de governo e para incentivar a mobilização dos cidadãos e sua participação ativa nas instâncias cabíveis. As tecnologias de informação e comunicação

[6] The Unified Modeling Language

devem ser utilizadas para integrar a escola e a comunidade, de tal sorte que a educação mobilize a sociedade e a clivagem entre o formal e o informal seja vencida.

Quando pensamos em resgatar a escola como espaço de luta da classe trabalhadora, imaginamo-la como instrumentalizadora do indivíduo. Concebemos a escola como necessária para a apreensão dos instrumentos teórico-metodológicos que auxiliem na compreensão da realidade, a tomada de consciência do cidadão e, principalmente, que oriente a prática social deste indivíduo / cidadão no universo mais abrangente da sociedade que ele ajuda a construir e na qual é construído.

Tal posição significa competência na extensão do domínio do conhecimento científico e tecnológico e compromisso com a articulação dos interesses da classe dominada.

O êxito dessa transformação depende, no mínimo, do domínio do conhecimento da técnica e da ciência que a classe dominante já absorveu e utiliza neste jogo de forças da sociedade de classes. É neste sentido que o intelectual das pessoas deve se desenvolver na modernidade. Sua formação há que acumular o conhecimento historicamente produzido e o sentido mais avançado deste conhecimento. O argumento da parcialidade do saber não pode ser esquecido, tampouco pode favorecer a monopolização do conhecimento por uns poucos representantes dos segmentos mais privilegiados da sociedade.

Desta forma, quando se pretende a transformação, não se pode negar à classe trabalhadora uma formação moderna e atual, o que

significa hoje: a criação, a aplicação, a pesquisa e a compreensão da tecnologia-ciência, incluindo-se aí o conhecimento da tecnologia do tratamento de informações.

As instituições de ensino devem se preocupar em fornecer instrumentos de análise daquilo que já está presente na prática social possibilitando a compreensão e a inserção do indivíduo neste momento da história.

A respeito da informática, a constatação é simples. Ela está presente no sistema de produção da sociedade capitalista moderna e, como tal, sua lógica, sua base científica, seu uso e suas implicações em todos os aspectos precisam ser compreendidos e dominados por todos os cidadãos. A este papel uma instituição compromissada com a transformação não pode se furtar.

Instrumentalizar os educadores para que possam se tornar cada vez mais sujeitos das decisões sobre como, quando e onde utilizar esta tecnologia é um dos objetivos principais da educação mediada por computador. Na época em que vivemos, numa sociedade complexa como a nossa, a quantidade de informações acumuladas e produzidas a cada dia é enorme. Tratar estas informações com a rapidez necessária de forma que os resultados possam contribuir na orientação do processo de tomada de decisão é o que podemos denominar o objeto principal da **INFORMÁTICA**.

A que mais especificamente o projeto AMEM se propunha é, sendo uma poderosa ferramenta para ensino mediado por computador, principalmente, fornecer maneiras inovadoras de utilizar a tecnologia sem que haja uma "programação intelectual do educando" mas, ao

contrário, desenvolvendo àqueles que estão envolvidos no processo ensino / aprendizagem uma imagem dinâmica de si próprios como agentes intelectuais.

O desenvolvimento do ambiente AMEM não envolveu somente a preocupação com os recursos educacionais que se deseja contemplar, mas também com a forma de contemplá-los, já que a não preocupação com o meio onde esse software será empregado bem como o público alvo do mesmo, poderá provocar uma maior exclusão social, tanto no que diz respeito ao analfabetismo tecnológico quanto ao fator financeiro, já que uma aplicação mal projetada poderá exigir tecnologias de suporte que inviabilizem seu uso pela maioria.

Desta forma, quando se pretende a transformação, não se pode negar aos cidadãos uma formação moderna e atual, o que significa hoje: a criação, a aplicação, a pesquisa e a compreensão da tecnologia-ciência, incluindo-se aí o conhecimento da tecnologia do tratamento de informações.

O ambiente AMEM foi concebido por uma equipe multidisciplinar que envolve educadores, informatas e designers na busca de um ambiente de fácil uso e baixo custo de implantação, tendo a investigação-ação como uma forma de indagação introspectiva coletiva empreendida por participantes em situações sociais com o objetivo de melhorar a racionalidade e a justiça de suas práticas sociais e/ou educativas, bem como sua compreensão dessas práticas e das situações em que tem lugar [1].

Dentre as diversas metodologias existentes para a construção do ambiente, optou-se pela prototipação, que se caracteriza por

implementar rapidamente um conjunto inicial de necessidades com a intenção declarada de expandi-las e refiná-las iterativamente à proporção do aumento do conhecimento mútuo do sistema por parte do usuário e do desenvolvedor.

Sua escolha justifica-se por sua semelhança com a prática educacional alvo do projeto, em que se tem uma espiral auto-reflexiva, que busca o constante aprimoramento das práticas educacionais baseado no conhecimento adquirido pelos sujeitos da ação.

Essas são algumas das considerações de projeto do ambiente AMEM, referentes à Educação Mediada por Computador, que buscaram garantir o seu uso nas mais diversas condições e, assim, atingir os ideais de emancipação e inclusão social e reflexão sobre suas práticas.

2.2 Engenharia reversa e reengenharia

Primeiramente deve-se considerar três conceitos dependentes: a existência de um processo de desenvolvimento de software, a presença de um sistema a ser analisado e a identificação de níveis de abstração [24].

Qualquer que seja o processo de desenvolvimento de software, espera-se que haja interação entre seus estágios e, talvez, recursão. Em um processo de desenvolvimento de software, os estágios iniciais envolvem conceitos mais gerais, independentes da implementação, enquanto os estágios finais enfatizam os detalhes de implementação [5].

O aumento de detalhes durante o processo de desenvolvimento conceitua os níveis de abstração. Estágios iniciais do sistema planejam e definem requisitos de alto nível quando comparados com a própria implementação [27].

Essa comparação é importante para deixar claro que nível de abstração e grau de abstração são grandezas distintas. Enquanto o nível de abstração é um conceito que diferencia os estágios conceituais do projeto, o grau de abstração é intrínseco a cada estágio. A evolução através das fases do processo de desenvolvimento de software envolve transições dos níveis mais altos de abstração, nos estágios iniciais, para níveis mais baixos, nos estágios posteriores. As informações podem ser representadas em qualquer estágio do desenvolvimento, seja de forma detalhada (baixo grau de abstração), seja de forma mais sucinta ou global (alto grau de abstração) [29].

Abstração é definida como a habilidade de se ignorar os aspectos de assuntos não relevantes para o propósito em questão, tornando possível uma concentração maior nos assuntos principais [5].

Para que as técnicas de manutenção de software (especificamente engenharia reversa e reengenharia) sejam descritas de forma simplificada, serão tomadas como base apenas três fases do processo de desenvolvimento de software, com níveis de abstração bem diferenciados, conforme a Figura 1:

- requisitos: especificação do problema a ser resolvido, incluindo objetivos, restrições e regras de negociação;
- projeto: especificação da solução;

- implementação: codificação, teste e adaptação ao sistema operacional.

A técnica tradicional, que avança progressivamente pelas fases do processo de desenvolvimento de software, é denominada engenharia progressiva [5]. A execução dessa técnica consiste em partir de projetos independentes da implementação, que possuem altos níveis de abstração, indo em direção a implementação física do sistema.

Em outras palavras, engenharia progressiva segue a seqüência de desenvolvimento estabelecida no projeto, visando a obtenção do sistema implementado.

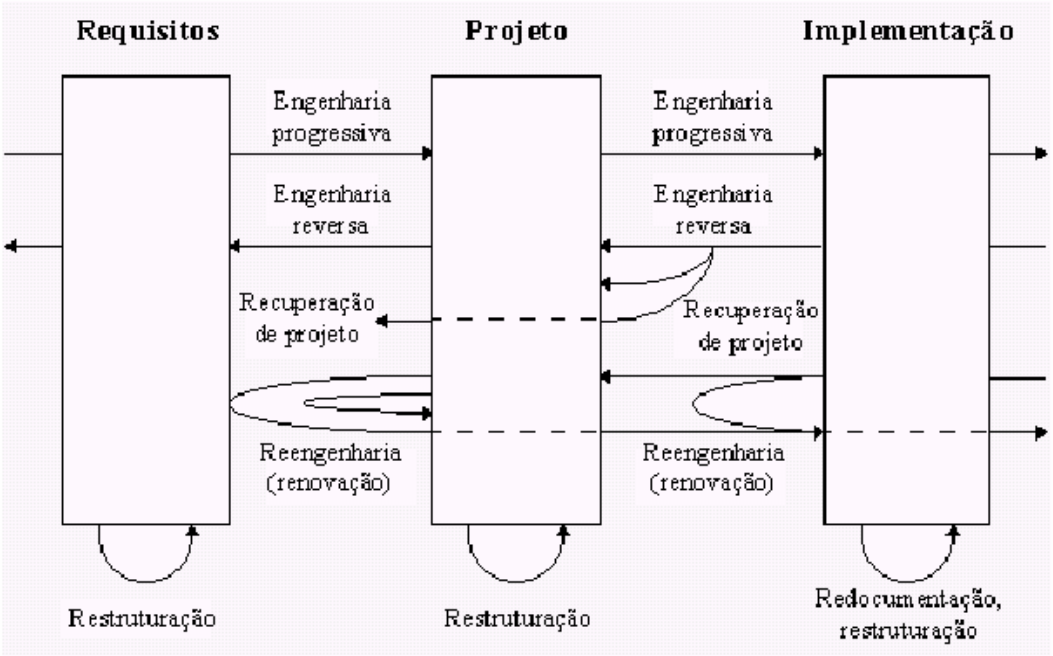


FIGURA 1 - Relacionamentos no Ciclo de Desenvolvimento de Software

Na Figura 1, com exceção da engenharia progressiva, as demais transições entre as fases de desenvolvimento são tecnologias utilizadas na manutenção de software, sendo assim definidas:

- redocumentação: como uma sub-área da engenharia reversa, é a criação ou revisão de uma representação semanticamente equivalente, dentro do mesmo nível relativo de abstração, sendo que as formas resultantes de representação são consideradas como visões alternativas, utilizadas para uma melhor compreensão humana do sistema analisado;
- recuperação de projeto: é uma sub-área da engenharia reversa na qual o conhecimento do domínio da aplicação, informações externas e dedução são adicionadas as observações referentes ao programa, para se extraírem abstrações significativas de mais alto nível, além daquelas obtidas através da observação direta do sistema;
- reestruturação: é a transformação de uma forma de representação, para outra no mesmo nível de abstração relativo, preservando o comportamento externo do sistema (funcionalidade e semântica). Geralmente usada como uma forma de manutenção preventiva, a reestruturação é aplicada em sistemas que tenham sido desenvolvidos de forma desestruturada, resultando uma representação que preserva as características do sistema, porém de forma mais bem estruturada;

- engenharia reversa: é o processo de analisar um sistema com a finalidade de criar sua representação de uma forma diferente ou em um nível mais alto de abstração do que o código fonte.
- reengenharia: é a reconstrução de algo do mundo real, tendo como propósito a busca por melhorias que permitam produzir algo de qualidade melhor ou compatível ao produto inicial.

No processo de manutenção, quando se trata de reconstruir um software (ou seja, realizar sua reengenharia), é necessário, portanto, que se proceda a engenharia reversa do sistema em questão, a fim de obter os modelos de análise baseados no software existente. Esses modelos, com as devidas correções / alterações, serão o ponto de partida para a engenharia progressiva [5].

O padrão IEEE P1219/D14 para manutenção de software define a reengenharia como um subconjunto da engenharia de software, composta por engenharia reversa e engenharia progressiva.

2.2.1 Engenharia reversa

Engenharia Reversa normalmente é empreendida com o objetivo de reprojetar um sistema para melhorar a sua manutenção ou produzir uma cópia de um sistema sem acesso às informações de seu projeto original [27].

O termo *engenharia reversa* originou-se da análise de hardware que extraía o projeto a partir do produto final. Em geral, é aplicada para melhorar um produto ou para analisar um produto concorrente ou

de um adversário (principalmente em situação militar ou de segurança nacional) [5].

Aplicando o conceito inicial de engenharia reversa a sistemas de software, muitas das técnicas utilizadas em hardware servem para obter uma compreensão básica do sistema e sua estrutura. Entretanto, enquanto o objetivo básico para hardware é duplicar o sistema, os objetivos mais freqüentes para software são obter uma compreensão suficiente em nível de projeto para auxiliar a manutenção, fortalecer o crescimento do sistema, e substituir o suporte.

Para Presmam, a engenharia reversa de software é um processo de recuperação de projeto, consistindo em analisar um programa na tentativa de criar uma representação do mesmo em um nível de abstração mais alto que o código-fonte [13].

2.2.1.1 Visões do software

A partir da engenharia reversa e com base nos diferentes níveis e graus de abstração, o software pode ser visualizado de diferentes maneiras [5]:

- visão em nível implementacional: abstrai características da linguagem de programação e características específicas da implementação;
- visão em nível estrutural: abstrai detalhes da linguagem de programação para revelar sua estrutura a partir de diferentes perspectivas. O resultado é uma representação explícita das dependências entre os componentes do sistema;

- visão em nível funcional: abstrai a função de um componente, isto é, o que o componente faz. Essa visão relaciona partes do programa às suas funções, procurando revelar as relações lógicas entre elas (diferentemente das relações sintáticas ou das estruturais);
- visão em nível de domínio: abstrai o contexto em que o sistema está operando, ou seja, o porquê do sistema a ser desenvolvido.

A Figura 2 mostra a correspondência entre as categorias de visualização do software e as diferentes atividades do ciclo de desenvolvimento de software.

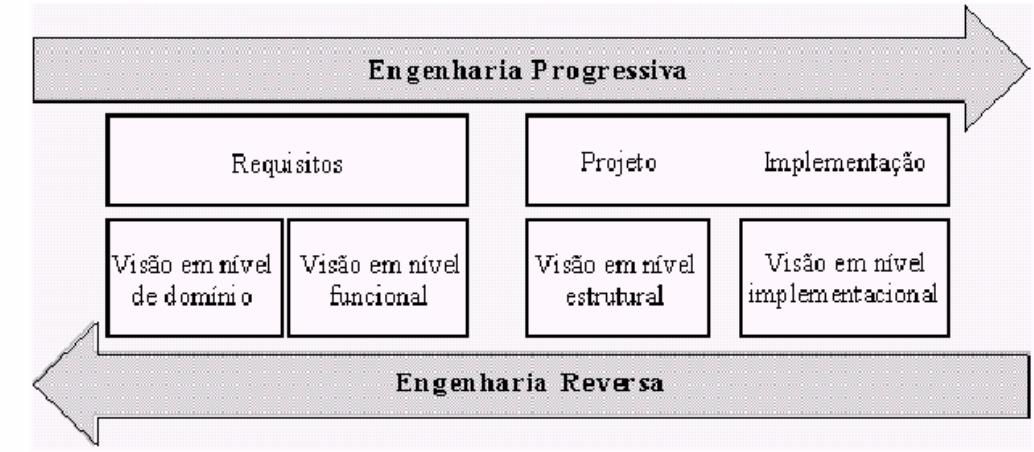


FIGURA 2 - Visualizações de Software no Ciclo de Desenvolvimento

De acordo com o nível de entendimento obtido do sistema e o escopo das informações usadas, duas categorias de engenharia reversa são definidas: visualização de código e entendimento de programa.

2.2.1.2 Visualização do código

Também denominada redocumentação, a visualização de código é a criação ou revisão de representações semanticamente equivalentes num mesmo nível de abstração [13]. O processo de visualização de código cria as representações a partir de informações obtidas apenas da análise do código fonte, embora a apresentação dessas informações possa se diversificar. As formas das representações são consideradas visões alternativas, cujo objetivo é melhorar a compreensão do sistema global.

A forma mais simples e mais antiga de engenharia reversa é a visualização de código [13]. A intenção é recuperar a documentação que já existiu, ou que deveria ter existido, sobre o sistema. A ênfase, de fato, é a criação de visões adicionais, especialmente visões gráficas, que não foram criadas durante o processo original de engenharia progressiva.

A visualização de código não transcende a visão em nível estrutural e não atribui significados ao sistema analisado. Recuperações mais ambiciosas tais como a função, os propósitos ou a essência do sistema exigem um nível de entendimento maior e são definidas como entendimento de programa.

2.2.1.3 Entendimento do programa

Nesta categoria de engenharia reversa, também denominada recuperação de projeto, o conhecimento do domínio das informações externas e as deduções são adicionadas as observações feitas sobre o

sistema através do exame do mesmo, de modo a obter informações com nível mais alto de abstração.

O entendimento do programa recria abstrações do projeto a partir de uma combinação de código, documentação existente do projeto (se disponível), experiências pessoais e conhecimentos gerais sobre o problema e o domínio de aplicação. Sintetizando, deve produzir todas as informações necessárias para se entender completamente o que, como, e por que o sistema faz.

Entendimento do programa distingue-se de visualização de código porque objetiva entender o sistema, em vez de simplesmente fornecer visões alternativas para auxiliar o usuário a entender o sistema. Esse entendimento vai além do conhecimento em nível implementacional e estrutural, buscando obter o conhecimento em nível funcional e até mesmo em nível de domínio (ambiente de operação do sistema).

Um completo entendimento do programa busca reconstruir não somente a função do sistema, mas também o processo pelo qual o sistema foi desenvolvido. É importante a recuperação de decisões de projeto tomadas durante o desenvolvimento original para uma completa estrutura de entendimento.

A categoria de entendimento do programa é a forma mais crítica de engenharia reversa, porque tenta aproximar-se do raciocínio humano na busca do entendimento.

A Figura 3 apresenta a amplitude de alcance das categorias de engenharia reversa, relacionadas com o escopo das informações utilizadas (código fonte ou base de conhecimento) e o nível de

visualização pretendida (implementacional, estrutural, funcional e de domínio).

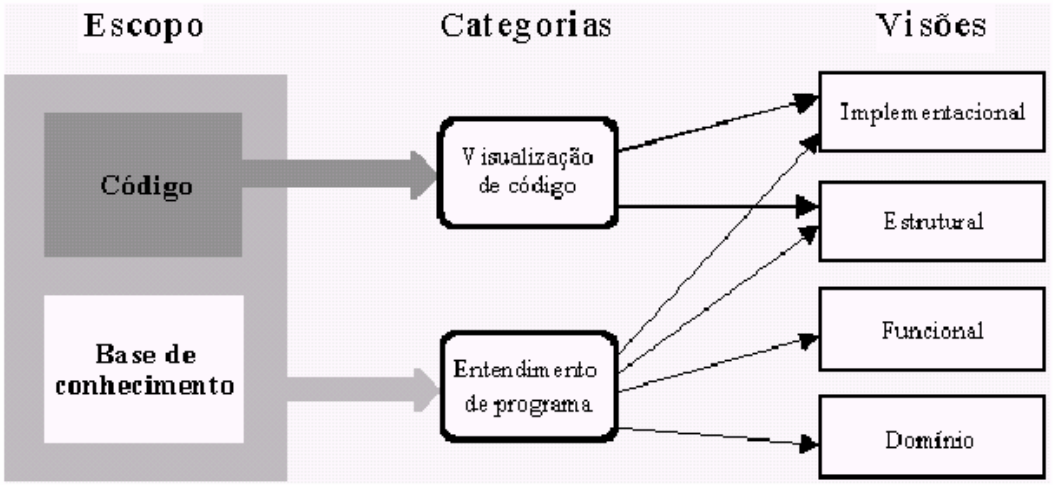


FIGURA 3 - Categorias da engenharia reversa e suas visões

2.2.2 Reengenharia

A reengenharia, conhecida também como renovação ou reconstrução, é o exame e alteração de um sistema de software, para reconstituí-lo em uma nova forma, e a subseqüente implementação dessa nova forma. Um processo de reengenharia geralmente inclui alguma forma de engenharia reversa, seguida por uma forma de engenharia progressiva ou reestruturação [26].

O principal objetivo da reengenharia é melhorar um sistema, através de alterações significantes que proporcionem melhoria, porém, sem alterar suas funções. A extração automática da descrição de uma aplicação e sua implementação em outra linguagem não é considerada reengenharia, e sim tradução de código [27]. Do mesmo modo que Chikofsky, Warden considera que a reengenharia pode ser dividida em duas fases principais: a Engenharia Reversa e a Engenharia

Progressiva; e cada uma destas fases pode ser dividida em uma série de atividades.

Também, para alguns autores, pode-se reutilizar automaticamente os esforços de desenvolvimento passados, objetivando reduzir custos de manutenção e melhoria na flexibilidade do software utilizando reengenharia [28].

Segundo [13], a reengenharia, também chamada de recuperação ou renovação, recupera informações de projeto de um software existente e usa essas informações para alterar ou reconstituir o sistema, preservando as funções existentes, ao mesmo tempo em que adiciona novas funções ao software, num esforço para melhorar sua qualidade global.

Para [29], a reengenharia de software é descrita como a reorganização e modificação de sistemas de software existentes, parcial ou totalmente, para torná-los mais manuteníveis.

Das diversas definições dispostas, percebe-se que existe clara distinção entre o desenvolvimento de um novo software e reengenharia de software. A distinção está relacionada ao ponto de partida de cada um dos processos. O desenvolvimento de um novo software (definido como engenharia progressiva) inicia-se com uma especificação escrita do software que será construído, enquanto que a reengenharia inicia-se tomando como base um sistema já desenvolvido.

O objetivo da reengenharia é produzir um sistema novo com maior facilidade de manutenção e a engenharia reversa é usada como

parte do processo de reengenharia, pois fornece o entendimento do sistema a ser reconstruído.

2.2.2.1 Engenharia de software legado

Um sistema se torna legado na medida em que nenhuma pessoa que participou de sua concepção está mais disponível. Por falta de documentação do sistema legado e falta de conhecimento sobre os detalhes de implementação por parte dos programadores atuais, sistemas legados podem apresentar um ou vários dos problemas apresentados no início da seção anterior.

Tradicionalmente, a engenharia reversa trabalha com velhos sistemas escritos em Cobol, Fortran, Assembler, etc. Mas essa definição inclui até sistemas mais recentes que foram desenvolvidos com linguagens modernas (com orientação a objeto por exemplo).

Um software legado pode ser informalmente definido como aquele que executa tarefas úteis para a organização, mas que foi desenvolvido utilizando-se técnicas atualmente consideradas obsoletas. A migração e/ou alteração desse tipo de software gera desafios técnicos (por exemplo: verificar efeitos colaterais de uma alteração) e não-técnicos (por ex.: estimar o custo de alteração) a todos os envolvidos com manutenção de software.

Existe um grande dilema na decisão sobre o futuro de software legado. Ao mesmo tempo em que ele traz incorporado o acúmulo de anos de experiência e refinamento, traz também todos os vícios e defeitos vigentes na época de seu desenvolvimento, mesmo que naquela época, o que hoje é chamado de vício e defeito, fosse a

melhor indicação para o desenvolvimento de software. Por exemplo, o tamanho do programa, que devido a limitação imposta pelo custo do hardware, tinha que ser reduzido, atualmente não é crucial (devido ao barateamento dos componentes de hardware). Houve ainda a época em que se prezava a eficiência, em detrimento da clareza e estruturação do programa. Adicione-se a isso as sucessivas manutenções e conseqüentes degenerações sofridas pelo software. Sabe-se hoje que todos esses fatores estão diretamente relacionados com a dificuldade de entendimento de qualquer software legado.

Muitos programas legados são críticos para os negócios das organizações que os possuem. Eles têm embutidas informações dos negócios e procedimentos, que podem não estar documentados. O risco de remover e reescrever tais programas é grande, pois muita informação teria que ser redescoberta por tentativa e erro. Conseqüentemente, as organizações, de um modo geral, não ”aposentam” seus sistemas legados, preferindo mantê-los em operação, com adaptações às novas necessidades.

As dificuldades devem ser enfrentadas, pois qualquer software que não evoluir continuamente (sofrer manutenções) tornar-se-á menos útil no mundo real.

Os maiores problemas em manter software legado são, segundo [28]:

- desestruturação e dificuldade de entendimento do código, muitas vezes porque o software foi desenvolvido antes da introdução dos métodos de programação estruturada;

- programadores que não participaram do desenvolvimento de um produto de software sentem dificuldade em entender e mapear a funcionalidade para o código fonte;
- documentação desatualizada, não auxiliando em nada a equipe de manutenção;
- dificuldade de predizer as conseqüências de efeitos colaterais;
- dificuldade de administrar múltiplas alterações concorrentes.

Em alguns negócios, estima-se que 80% de todos os gastos com software são consumidos pelas manutenções e evoluções dos sistemas. O número de sistemas que precisa ser modificado está aumentando gradualmente. Existe fila de espera para pedidos de manutenção. Isto significa que, algumas vezes, é impossível para as organizações investirem em novos sistemas para melhorar a eficiência organizacional [28].

O problema de manutenção de sistemas legados é complexo porque, geralmente, esses sistemas não são simples programas, desenvolvidos e mantidos. Muitos deles são compostos de outros diferentes programas que, de alguma forma, compartilham dados. Também ocorre que esses programas foram desenvolvidos por diferentes pessoas ao longo dos anos, as quais não estão mais na organização. Muitas vezes foi usado um sistema de administração de bases de dados que pode estar obsoleto, ou depende de arquivos armazenados separadamente. No caso de arquivos separados, cada um tem seu próprio formato e, freqüentemente, as mesmas informações são duplicadas e representadas em diferentes formas e em diferentes

arquivos. Essa duplicação usualmente acontece porque as informações são fortemente integradas com as estruturas de dados dos programas [13].

Com o objetivo de minimizar os problemas gerados por manutenções difíceis e, algumas vezes, degenerativas da estrutura do sistema, muitas organizações estão optando por refazer seus sistemas. A idéia básica dessa reconstrução ou reengenharia é que as informações de projeto e especificação sejam extraídas do código fonte, reformuladas e reconstruídas, resultando um software mais fácil de ser mantido.

Aplicando-se a reengenharia de software, o sistema pode ser redocumentado ou reestruturado, os programas podem ser traduzidos para uma linguagem de programação mais moderna e implementados em uma plataforma distribuída, bem como seus dados podem ser migrados para uma base de dados diferente. A reengenharia de software objetiva fazer sistemas flexíveis, fáceis de modificar, frente às constantes mudanças das necessidades dos usuários [5].

2.2.2.2 Migração do paradigma procedimental para o de orientação a objetos

Atualmente, a fase de transição que a comunidade de informática está enfrentando para migrar software das plataformas centralizadas (mainframes) para ambientes de processamento distribuído em arquitetura cliente/servidor, mostra a importância da reengenharia, o que implica, muitas vezes, o mover de uma arquitetura orientada a ação (paradigma procedimental) para uma arquitetura orientada a

objetos (paradigma de orientação a objetos). É exatamente o objetivo fundamental deste trabalho.

A programação orientada a objetos tem muitas vantagens sobre a programação procedimental, pois possibilita construir sistemas altamente flexíveis, adaptáveis e extensíveis; possui uma coleção rica de mecanismos composicionais para formação de classes, instanciação de objetos, propriedades de herança, polimorfismo e ocultamento de informações, que estão presentes em linguagens de programação orientadas a objetos, as quais provêem apoio para criar sistemas que exibem um alto grau de reuso e facilidade de manutenção.

Existem vários motivos para migração do paradigma procedimental para o paradigma de orientação a objetos:

- em programas procedimentais, é muito difícil prever efeitos colaterais, pois freqüentemente os relacionamentos não são visíveis;
- manutenções sucessivas requerem um conhecimento detalhado dos componentes do sistema, como foram criados e modificados e seus inter-relacionamentos;
- a reusabilidade de software também é seriamente afetada pela estrutura própria dos programas procedimentais;
- a orientação a objetos oferece algumas características úteis, tais como meios de abstração bem definidos, conceito de encapsulamento e comportamentos que efetivamente apóiam o processo de manutenção de software.

A identificação de objetos nos programas procedimentais, exibindo explicitamente suas dependências, ajuda a entender o projeto do sistema, evita a degradação do projeto original durante as manutenções e facilita o processo de reuso.

2.2.3 Considerações finais

O enfoque maior está nas definições de:

- "o que" é a reengenharia e outros termos relacionados a manutenção de software;
- na aplicabilidade dessa atividade;
- "por que" realizar a reengenharia de um software.

O "como", ou seja, a forma de realizar a reengenharia em um software depende de muitos fatores, relacionados, por exemplo, com a forma de trabalho da empresa, a origem do software em questão e com o desenvolvimento de tecnologias para apoiar essa atividade.

A importância da reengenharia está em possibilitar que todo conhecimento agregado ao software legado não seja perdido, o que aconteceria se a opção fosse pelo desenvolvimento de um novo software.

O processo de extração do conhecimento de um software legado é realizado pela engenharia reversa, que fornece visões mais abstratas do que o código do sistema e/ou alguma outra documentação possivelmente existente. Com essas visões é possível compreender o software e o sistema em que está inserido, possibilitando a produção de modelos de análise que servirão a reengenharia.

Da reengenharia de um software resulta uma nova versão, que agrega toda a informação do software legado, as inovações tecnológicas e novas funcionalidades desejadas. Dessa forma, a reengenharia de software não trata apenas de modernizá-lo, mas também adaptá-lo de acordo com as novas necessidades do processo em que está inserido.

Baseado principalmente nos conceitos acima, adotou-se a prática da Engenharia Reversa e da Reengenharia para o desenvolvimento da modelagem do sistema atual do AMEM, que embora já esteja em perfeito funcionamento e possua uma documentação, de suas funcionalidades, muito boa, não possuía uma modelagem relacional, a qual também foi desenvolvida no decorrer do projeto da modelagem e será descrita na próxima seção.

3 O AMBIENTE AMEM

O Ambiente Multimídia para Educação Mediada por computador, chamado de ambiente AMEM, foi desenvolvido na perspectiva da investigação-ação educacional, concebido por uma equipe multidisciplinar que envolve educadores, informatas e designers, a fim de buscar um ambiente de fácil uso e baixo custo de implantação, com o objetivo de melhorar a racionalidade e a justiça de suas práticas sociais e ou educativas, bem como sua compreensão dessas práticas e das situações em que tem lugar [1].

No desenvolvimento deste software, adotaram-se metodologias de projeto empregadas em Engenharia de Software para auxiliar a equipe de desenvolvedores a obter economicamente um software que fosse confiável e que funcionasse eficientemente, de fácil utilização e manutenção.

A idéia era ultrapassar os limites do ensino tradicional que perpassava nos cursos organizados pela comunidade da área tecnológica, os quais estavam pautados, predominantemente, pelos manuais dos fabricantes.

A etapa de planejamento envolveu a discussão, em grupo, dos requisitos desejados no ambiente AMEM e a forma como esses requisitos foram apresentados e desenvolvidos, procurando tornar a equipe multidisciplinar mais homogênea e o intercâmbio de conhecimento mais efetivo [1].

A etapa de análise e projeto constou da modelagem dos requisitos educacionais levantados para um conjunto de procedimentos sistematizados para a lógica computacional. O modelo foi discutido e tendo ocorrido uma concordância deu-se início a etapa de prototipação, ou seja, a implementação, que envolveu a codificação do requisito levantado na lógica de computador através de uma linguagem computacional. Não houve uma modelagem relacional do ambiente.

A etapa de avaliação buscou avaliar cada um dos requisitos implementados até o momento na prática do dia-a-dia, dessa forma, os problemas detectados foram inseridos na próxima etapa da espiral e o sistema evolui sobre o seu uso e a reflexão sobre o mesmo [1].

Para o desenvolvimento do AMEM, o apoio de especialistas da área de design foi fundamental, para que se fizesse um estudo ergonômico das Interfaces homem-máquina do ambiente, em busca de uma apresentação mais amigável e que reduzisse o tempo de adaptação as características, recursos e uso do mesmo. Esse estudo ergonômico que deu origem as atuais interfaces do ambiente AMEM, é apresentado em [14].

Quanto ao primeiro obstáculo, foi de consenso do grupo AMEM, o desenvolvimento de um ambiente baseado em software livre ou ***freeware***. O primeiro refere-se a uma filosofia de desenvolvimento de software e não a gratuidade do mesmo, uma vez que esse pode ser comercializado. Essa filosofia prega a liberdade de uso do software independentemente do fim, desde que este continue sendo distribuído com seus códigos fonte (lógica do sistema escrita em linguagem de

computador), o que permite sua modificação e adaptação a realidade do usuário final, já o segundo, restringe-se a permitir sua distribuição e uso sem custos.

No entanto, o fato de serem livres não resolve completamente o problema, pois, existia, ainda, o fator dependência de plataforma, ou seja, se o ambiente AMEM fosse dependente de um tipo específico de hardware (computador) e / ou software (aplicativos de suporte: sistema operacional) ou ambos, e esses não fossem os possuídos pelo usuário interessado em seu uso, essas características não seriam suficientes.

Para solucionar esse fato, foi procurado o uso de tecnologias multiplataforma, ou seja, que se adaptassem às mais variadas configurações de hardware e software. Dessa forma, optou-se pelo desenvolvimento de um ambiente baseado na Web, já que uma aplicação desse tipo permite atingir um público maior e sua utilização fica dependente somente da presença de um navegador Web, comumente usado para navegação na Internet. Assim, qualquer usuário que tenha acesso a um computador conectado a Internet ou a uma rede de computadores, que a simule, e um navegador Web poderá acessar o ambiente AMEM e usufruir todos os seus recursos, sem, para isso, precisar instalar nenhum outro tipo de tecnologia [1].

Nestes moldes, o ambiente fornece ferramentas automáticas para customização dos materiais didáticos, inclusive na forma de desafio, melhor solução no momento e desafio mais amplo, organizando o processo de aprendizagem pedagogicamente.

O emprego desse procedimento metodológico engaja os educandos na resolução de problemas e projetos (desafiando-os educacionalmente no âmbito do ambiente), que exige do educando a pesquisa e reflexão das informações obtidas para a elaboração da solução, sendo que quando esse possuir qualquer dúvida poderá interagir com o educador, assim como debater esses questionamentos com outros educandos.

Após a conclusão do sistema, propriamente dito, seguiu-se e ainda segue, a fase de treinamento dos educadores para a utilização desse sistema na aplicação de cursos estipulados.

Para esse fim, o projeto foi estruturado em tarefas modulares descritas a seguir:

- Módulo I: Instrumentalização dos educadores fazendo uso do ambiente semi-presencial, ministrando cursos de Sistema Operacional, Editor de Textos, Editor de Recursos Multimídia e Tecnologias de acesso à Internet (tempo);
- Módulo II: Observação e avaliação dos resultados obtidos no módulo I, assim como o estudo de ferramentas e linguagens para o desenvolvimento de ambientes distribuídos (tempo);
- Módulo III: Criação e desenvolvimento de um ambiente para educação à distância (EAD) (tempo).

O ambiente contém:

- **Uma Interface de Identificação de Usuário**: serve para preparar o ambiente de acordo com as prioridades de cada um

dos usuários, ou seja, interface do educando ou interface do educador;

- **Uma Interface para o Educando**: fornece ao educando todas as ferramentas e materiais didáticos disponibilizados pelo educador;
- **Uma Inferface para o Educador**: fornece ao educador as ferramentas necessárias à disponibilização de um curso, assim como formas de avaliação do progresso dos educandos;
- **Mecanismos de comunicação**: oferece os meios possíveis para a realização da comunicação entre educando/educador e educando/educando no ambiente, como e-mail, fórum de discussão, debates,...
- **Mecanismos de coordenação**: oferece formas de agendamento de eventos e controle do andamento do curso, tais como: quadro de avisos, plano de aulas, tarefas, avaliação e relatórios de participação;
- **Mecanismos de cooperação**: oferece formas de compartilhamento e de participação externa ao material do curso, como: bibliotecas virtuais, co-autoria de educadores e educandos.
- Módulo IV: Implementação de um curso a ser definido aplicando o ambiente desenvolvido para EAD e a disseminação deste ambiente e desta metodologia aos envolvidos no projeto (tempo).

3.1 Processo de reengenharia e engenharia reversa do AMEM

Um dos objetivos deste trabalho, inicialmente, foi através de técnicas de Engenharia Reversa e Reengenharia, projetar um modelo entidade-relacionamento para o AMEM, para que, sua estrutura inicial pudesse ser modificada.

Juntamente com o desenvolvimento desta etapa, espera-se expor as modificações efetuadas na estrutura original do projeto, sem explicitar cada funcionalidade do AMEM, pelo simples motivo da existência desta documentação em trabalhos publicados anteriormente por outras pessoas.

3.1.1 Experiência com engenharia reversa e reengenharia

Nesta etapa do trabalho, pesquisaram-se padrões para o desenvolvimento da engenharia reversa do AMEM.

Baseando-se nesses padrões, foram gerados processos de engenharia reversa e de engenharia avante orientados a objetos, a partir de sistemas legados procedimentais. Primeiramente, foram modelados os dados do legado, agrupando padrões que extraíssem informações a partir dos dados, do código fonte e do Modelo de Análise existente do sistema atual do sistema legado, gerando o Modelo Entidade Relacionamento (visão procedimental dos dados).

Posteriormente, foram modeladas novas funcionalidades ao sistema, incrementadas novas capacitações técnicas e ajustados alguns relacionamentos desnecessários, os quais serão documentados a seguir

quando será feita a comparação do sistema atual com o modelo novo proposto neste documento.

3.1.1.1 Modelagem atual e aperfeiçoamentos

Os modelos entidade-relacionamento do sistema AMEM foram refeitos, para buscar um aperfeiçoamento do mesmo. Baseado nas funções atuais, já publicadas, foram aperfeiçoadas algumas funções.

Por exemplo: na biblioteca, atualmente, não é dado suporte para a inclusão de arquivos no banco de dados e sim em disco. Desse modo é bem mais complicado fornecer portabilidade e principalmente facilidade de busca de conteúdos. Isto porque o sistema atual é dependente das permissões do administrador do Sistema Operacional em que o AMEM está sendo utilizado.

Além disso, para que sejam gravados novos arquivos em disco, no sistema atual, são necessárias permissões de leitura e escrita no sistema operacional, o que infelizmente pode comprometer a segurança do sistema.

Nesse sentido, a primeira modificação na estrutura da modelagem relacional foi incrementar o suporte a inclusão de arquivos em Banco de Dados e não mais em disco.

Esta modificação também é fundamental para que se possa dar portabilidade ao sistema no momento de um backup das informações armazenadas no AMEM, estando o mesmo instalado em qualquer tipo de Sistema Operacional.

A modificação na Biblioteca pode ser visualizada através da figura 4, que representa a biblioteca no modelo atual e na figura 5, que representa a biblioteca no modelo modificado:

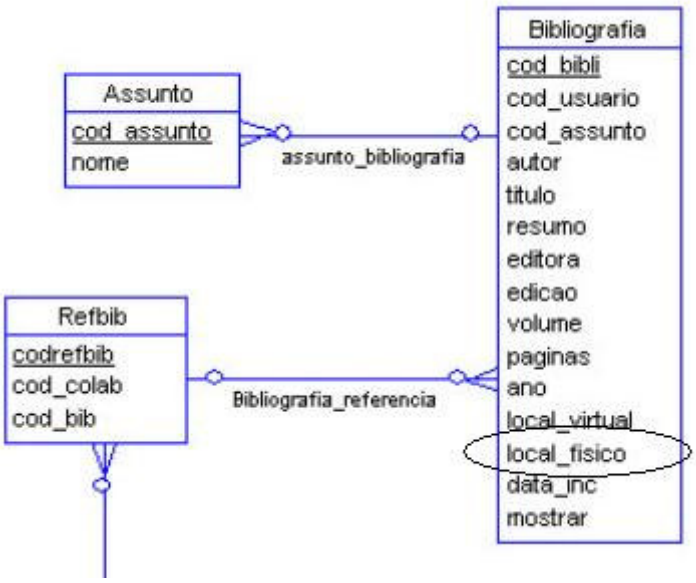


FIGURA 4 – Biblioteca sendo armazenada em um endereço físico do servidor

FIGURA 5 – Biblioteca sendo armazenada em banco de dados

Nota-se, na figura 5, uma diferença bastante significativa em relação ao modelo atual, da figura 4. No modelo proposto, foi adicionada uma nova tabela, denominada “Arquivo_bib”, em que existe a manutenção de arquivos dentro de campo de armazenamento de objetos, denominado dados, ao invés de se continuar mantendo um campo “local_fisico”, como no modelo atual, em que é guardado somente o caminho onde o arquivo está gravado em disco e não o arquivo por completo na base de dados.

Da mesma maneira, tabelas em que haviam referências a arquivos contidos em disco tiveram suas estruturas alteradas para que tivessem este tipo de facilidade, conforme é mostrado nas figuras 6 e 7:

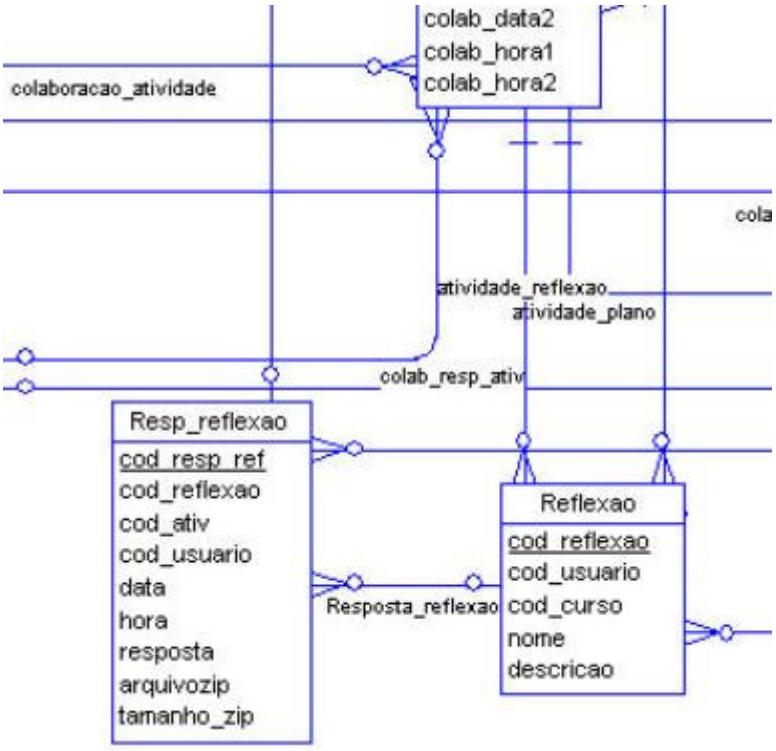


FIGURA 6 – Tabela de reflexão das atividades atualmente

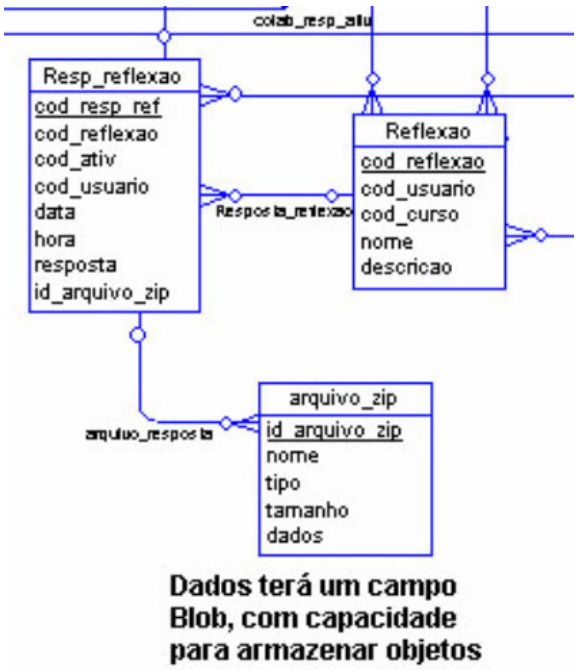


FIGURA 7 – Tabela adicionada para inclusão de arquivos

Nos mesmos moldes, foi alterada a tabela de reflexão nas atividades, da mesma forma que na biblioteca, adicionando um identificador para uma tabela que contém o nome do arquivo a ser armazenado no banco, o tipo de arquivo (zip, doc, txt, pdf, etc), o tamanho em bytes e um campo para armazenamento binário desse arquivo.

Além desse tipo de correção no projeto, também foram corrigidas, na modelagem relacional, questões como chaves estrangeiras desnecessárias, como é o caso de haver dois identificadores em tabelas referenciando turmas e cursos respectivamente, o que é totalmente desnecessário e pode causar uma grande confusão para o analista de sistemas, pois a tabela de turmas já contém uma referência para cursos (No modelo Entidade-Relacionamento do AMEM, cursos são tratados como disciplinas), tornando totalmente desnecessárias em várias tabelas referências a cursos.

Um exemplo pode ser mostrado pela figuras 8, 9 e 10:

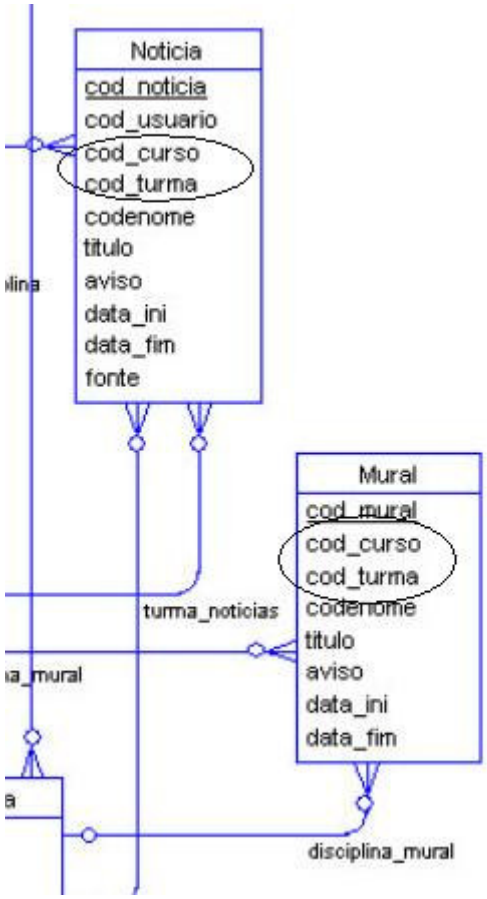


FIGURA 8 – Identifica duplicação de informação (desnecessário).

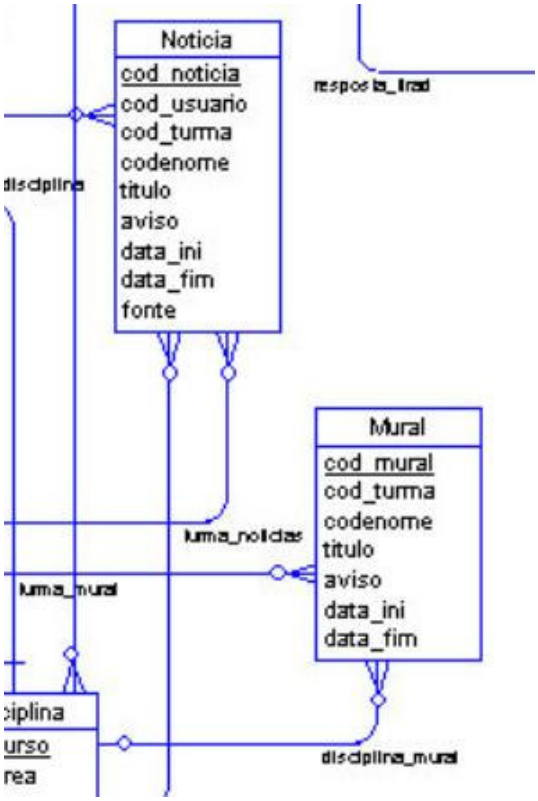


FIGURA 9 – Duplicações de referências removidas no modelo novo

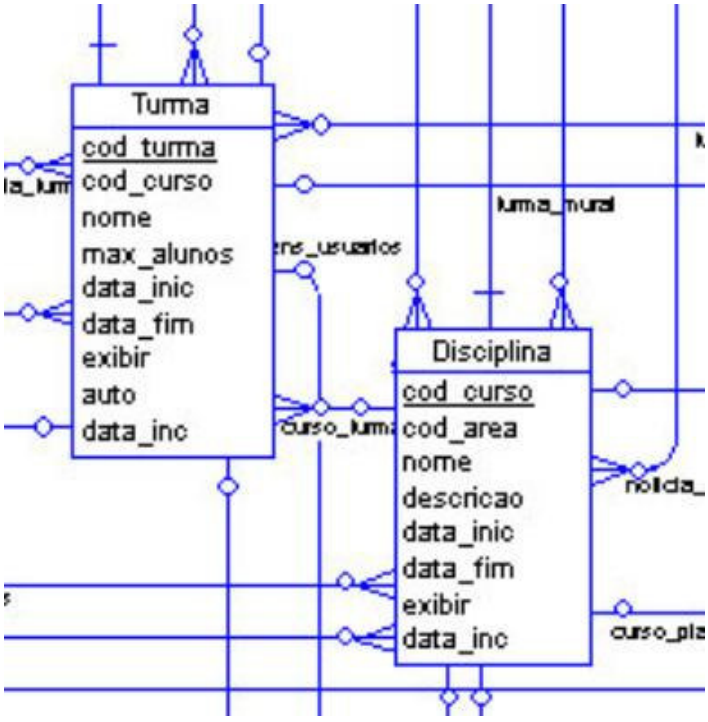


FIGURA 10 – Toda a turma está contida em uma disciplina / Curso

Através da figura 10, pode-se observar que um registro da tabela de turmas, automaticamente estará referenciando a algum curso correspondente àquela turma.

Salvo algumas correções já citadas anteriormente, como a manutenção de arquivos em campos das tabelas SQL e duplicações de referências removidas, foram adicionadas novas funcionalidades ao modelo atual. Ferramentas como um Tira-Dúvidas (Figura 11), com a possibilidade de envio de perguntas para uma **FAQ**[7] e um Comunicador Instantâneo (Figura 12), com a possibilidade de troca de arquivos, também foi incrementado ao modelo.

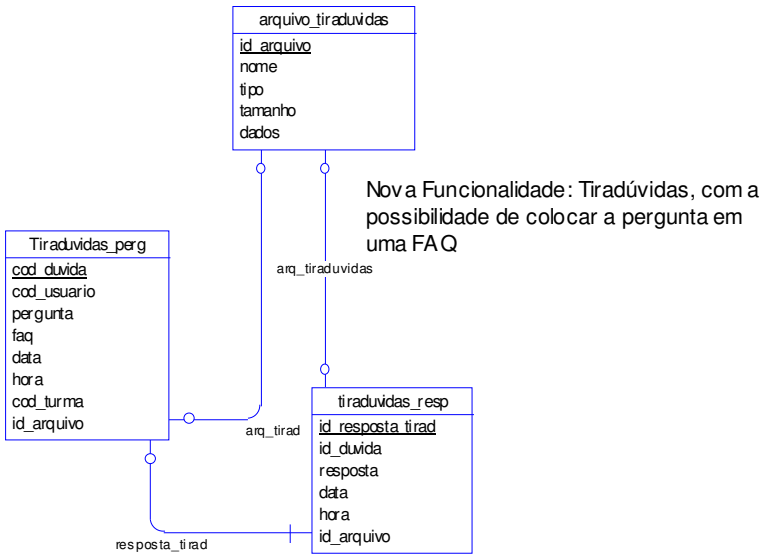


FIGURA 11 – Tira-Dúvidas

Este modelo de Tira-Dúvidas foi desenvolvido e implementado no Portal Matemático, o qual se encontra no endereço

[7] Frequently Asked Questions

http://www.matematica.com.br, com o objetivo similar ao que será acrescentado no AMEM.

Atualmente os alunos usuários do ambiente AMEM não possuem uma ferramenta capaz de proporcionar total interação entre alunos e professores via Web. A qualquer hora, o aluno poderá solucionar suas dúvidas em qualquer disciplina que esteja cursando, caso haja algum professor de plantão. Sendo uma ferramenta para a Web, proporciona que o professor tenha um completo conforto podendo responder a seus alunos a qualquer hora e em qualquer lugar.

O funcionamento do sistema é intuitivo e eficiente. O aluno obtém uma tela na qual escolhe a disciplina de sua dúvida e entra com a questão em uma caixa de edição, podendo vincular arquivos pré-concebidos. Automaticamente, o professor que responde pela disciplina naquele determinado momento recebe uma notificação de sua existência. Em seguida o aluno estará recebendo o retorno. Ao receber uma notificação de questionamento, o professor entra no sistema e imediatamente terá acesso a todas as solicitações pendentes. Ele pode abrir os documentos anexados pelo aluno e consultar sua dúvida. Então formula a resposta podendo também vincular arquivos de apoio e a retorna para o aluno. Todas as questões juntamente com suas respostas e arquivos anexados são mantidas em uma base de dados, de onde os professores regularmente extraem as mais comuns para montar outra base de questões freqüentemente enviadas.

Este processo é bem amigável também para professores de instituições com contratos que permitem interação neste nível do sistema. Com este histórico de questões e lista das mais comumente

enviadas, o aluno tem acesso a informações privilegiadas, algumas vezes solucionando por completo sua dúvida antes mesmo de solicitar ao professor, o que pode ser feito em todos os casos.

A figura 12 mostra o modelo proposto para a construção de um comunicador on-line para o AMEM:

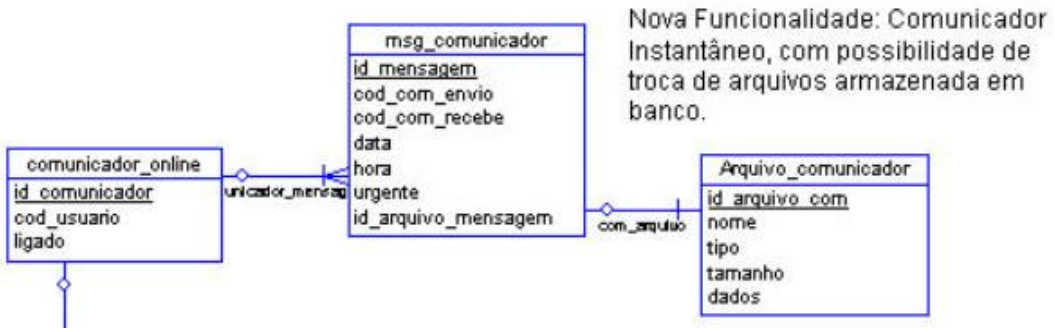


FIGURA 12 – Modelagem de um Comunicador On-line

A proposta deste comunicador on-line é facilitar o diálogo alunos – alunos, professores – professores e alunos – professores. Com a possibilidade de troca de informação e troca de arquivos.

A comunicação não é instantânea, devido ao fato de que antes de o usuário receber sua mensagem, esta será armazenada na base de dados, possibilitando um histórico on-line das informações trocadas via AMEM.

Os arquivos que poderão ser enviados via comunicador também são armazenados em banco de dados o que retira do usuário a responsabilidade de configuração de portas para recebimento de arquivos, facilitando a capacidade de segurança dos usuários.

A figura 13 representa a miniatura da modelagem do sistema atual do AMEM:

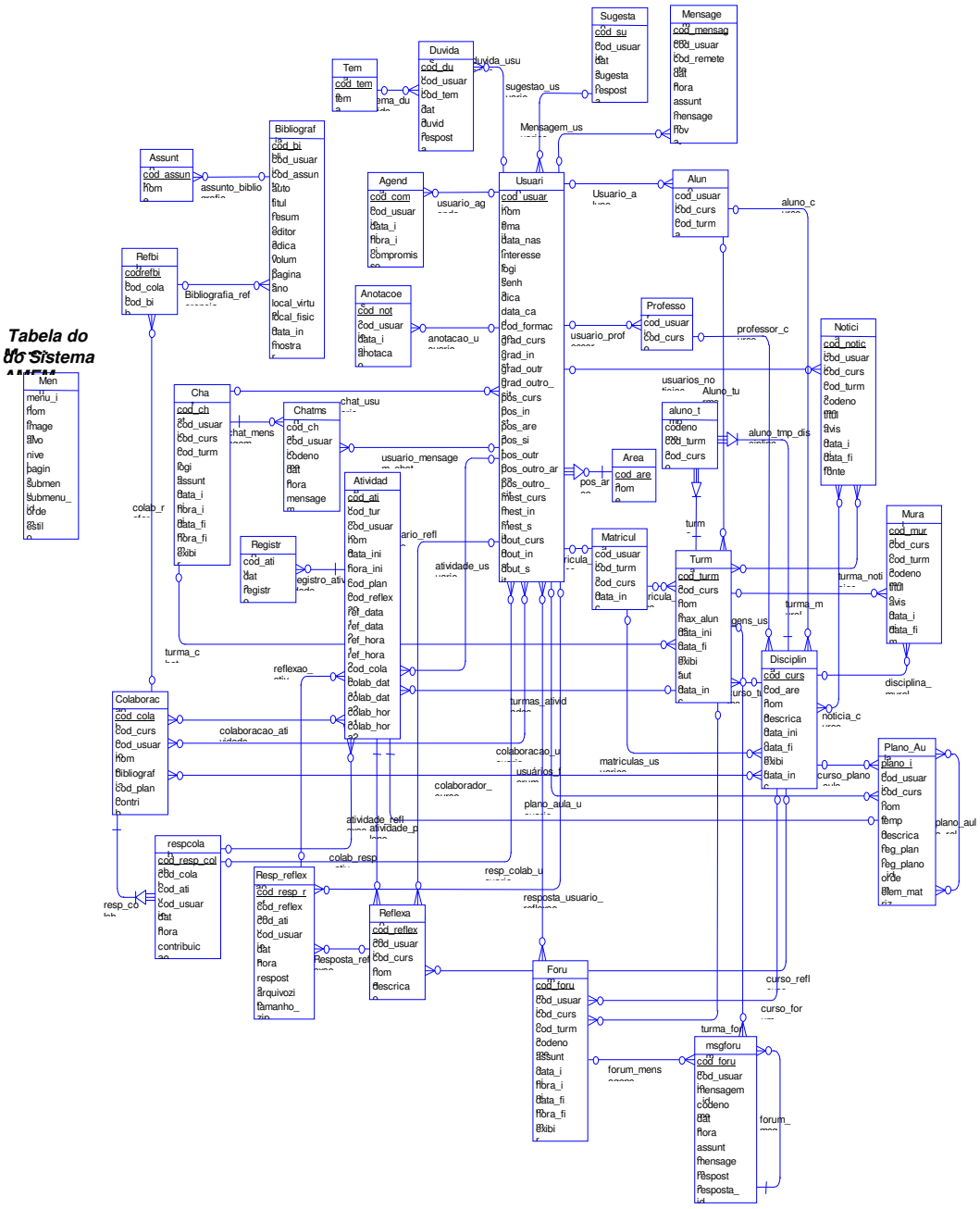
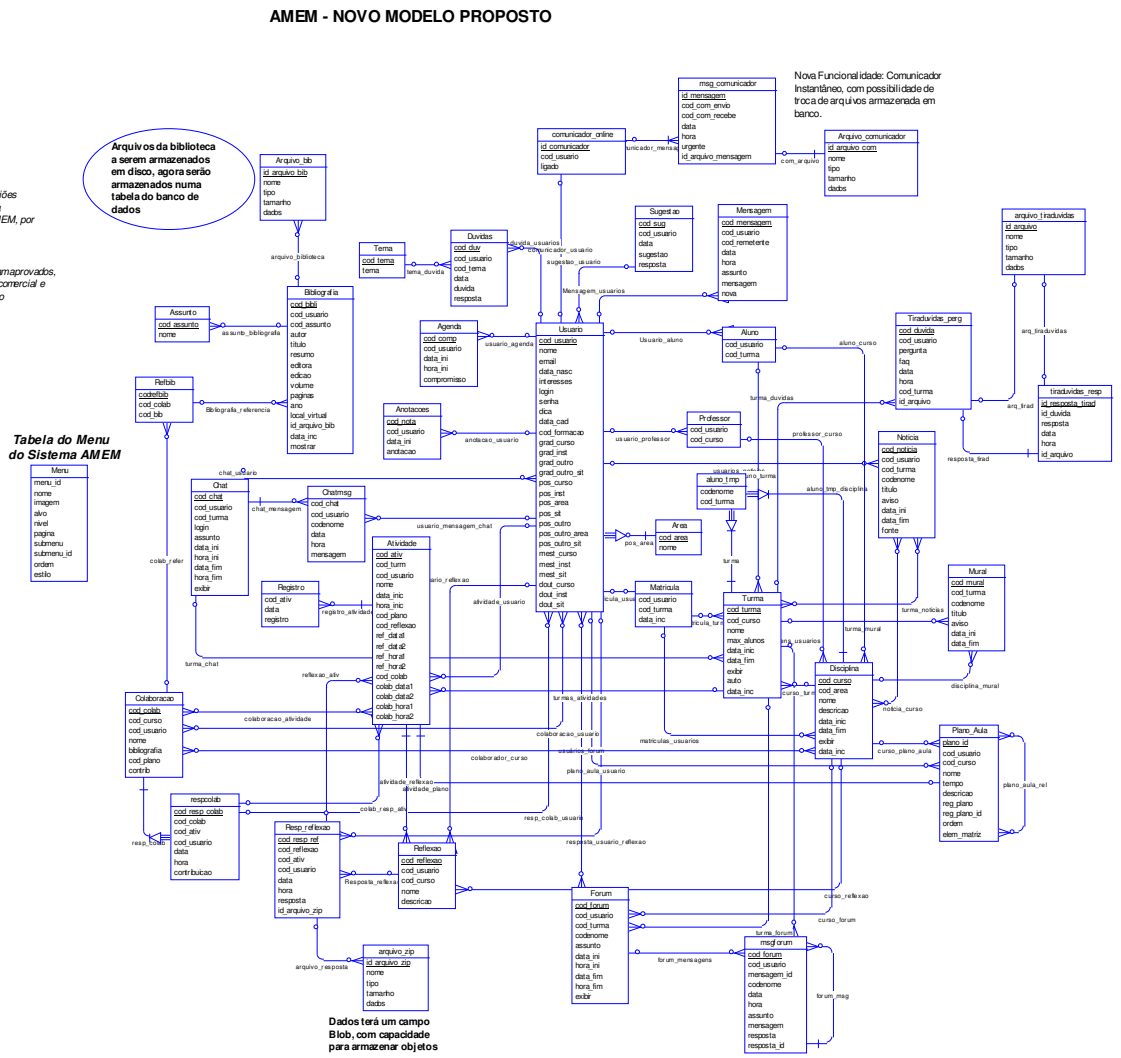


FIGURA 13 - Modelagem do Sistema Atual

A figura 14, representa a miniatura da modelagem do novo modelo proposto do AMEM:

FIGURA 14 – Modelagem do Novo Modelo Proposto



3.1.1.2 O que não foi aprovado ou elaborado

Neste item do trabalho procurar-se-á relatar idéias que o grupo do AMEM não aprovou.

A primeira delas foi com relação a Agenda. Foi dada a sugestão de construir uma busca de horários livres nas agendas de outros usuários cadastrados no AMEM, para que se pudessem marcar reuniões em horários não agendados (Principalmente explorar mais as reuniões virtuais), por exemplo, interagindo mais a agenda com o restante do sistema.

Por questões de privacidade e principalmente pelo não uso da Agenda por parte de alguns usuários, esta funcionalidade não foi aprovada.

3.1.1.3 O que pode agregado ao sistema com o novo modelo

O novo modelo proposto possibilita ao programador construir um sistema de relatórios, visto que com a modelagem do AMEM pronta, tem-se uma visão mais ampla do sistema.

Com o armazenamento de arquivos em tabelas do banco e não mais em disco, é facilitada a construção de uma ferramenta de backup, o que a exemplo dos relatórios, inexiste no ambiente AMEM.

3.2 A modelagem Orientada a Objetos

Os modelos de dados orientados s objetos são mais expressivos e flexíveis que o modelo relacional. O modelo de dados relacional foi

criado para permitir a representação de uma grande variedade de problemas usando um pequeno conjunto de conceitos simples. Por outro lado, os modelos orientados a objetos foram projetados para a criação e representação de estruturas bem mais complexas de uma maneira coerente e uniforme.

Hoje, com inúmeras vantagens encontradas ao se utilizar o paradigma de orientação a objetos, os SGBDs [8] Orientados a Objetos começam a se destacar comercialmente, embora os SGBDs relacionais ainda representem o estado da arte na tecnologia tradicional de bancos de dados e serem os mais estudados na literatura. Suas maiores vantagens são a simplicidade e a portabilidade entre implementações. Esses sistemas continuam a dominar o mercado, sendo utilizados nas mais diversas aplicações.

Entretanto, os conceitos de orientação a objetos podem ser utilizados como mecanismos de abstração para o projeto de bancos de dados relacionais. A modelagem orientada a objetos, por utilizar conceitos mais claros e naturais, permite produzir bancos de dados relacionais mais adequados às aplicações do mundo real, evitando os problemas de normalização freqüentemente associados ao projeto relacional. Além disso, a modelagem orientada a objetos aumenta a integração entre os dados e as aplicações. Dessa forma, uma abordagem orientada a objetos para o projeto lógico de bancos de dados relacionais permite que as aplicações sejam projetadas de forma integrada, em que dados e operações possam ser projetados ao mesmo tempo.

[8] Sistemas de Gerenciamento de Banco de Dados

3.2.1 Diagrama de Caso de Uso do AMEM

O diagrama a seguir (figura 15) mostra as ações entre um aluno, o professor e o administrador dos sistemas integrados do AMEM.

Quando se fala em sistemas integrados do AMEM, denota-se todas as funcionalidades do mesmo, como o fórum, a biblioteca, o sistema de envio de trabalhos, o sistema de envio de mensagens instantâneas, etc.

O modelo de caso de uso consiste de atores e casos de uso. Os atores representam usuários e outros sistemas que interagem com sistema modelado. Eles são representados graficamente por um "homem palito". Os casos de uso mostram o comportamento do sistema, cenários que o sistema percorre em resposta ao estímulo de um ator.

No modelo de caso de uso o relacionamento entre um ator e um caso de uso representa a participação deste ator no caso de uso. Além deste relacionamento, existe um outro tipo de relacionamento entre casos de uso:

- O relacionamento estender: é representado graficamente por uma seta com o estereótipo ***<<extend>>***, mostrando que o caso de uso destino pode incluir o comportamento especificado pelo caso de uso origem.

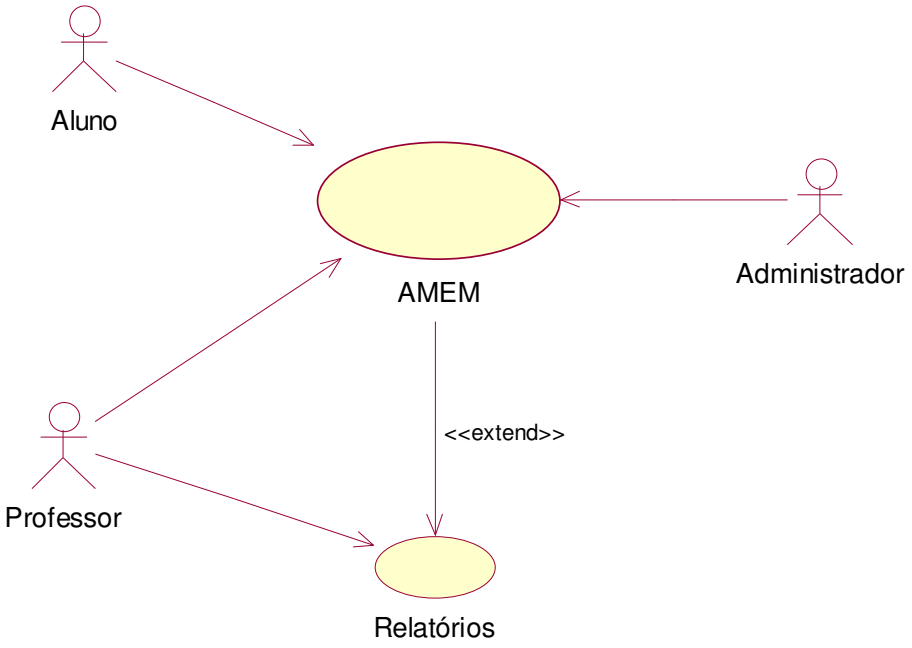


FIGURA 15 – Diagrama de Caso de Uso do AMEM

3.2.2 Diagrama de Seqüência do Sistema

Os diagramas de seqüências são usados para modelar a interação entre objetos em um sistema. Tipicamente, um diagrama de seqüência captura o comportamento de um único caso de uso. O diagrama apresenta os objetos e as mensagens que são passadas entre estes objetos dentro do caso de uso da sessão anterior. Objetos são representados por linhas tracejadas verticais, e a passagem de mensagens entre dois objetos é representada por vetores horizontais. As mensagens são desenhadas cronologicamente do topo à base do diagrama.

A figura 16 apresenta um modelo simples do funcionamento geral do AMEM, utilizando as entidades apresentadas no Diagrama de caso de uso deste trabalho.

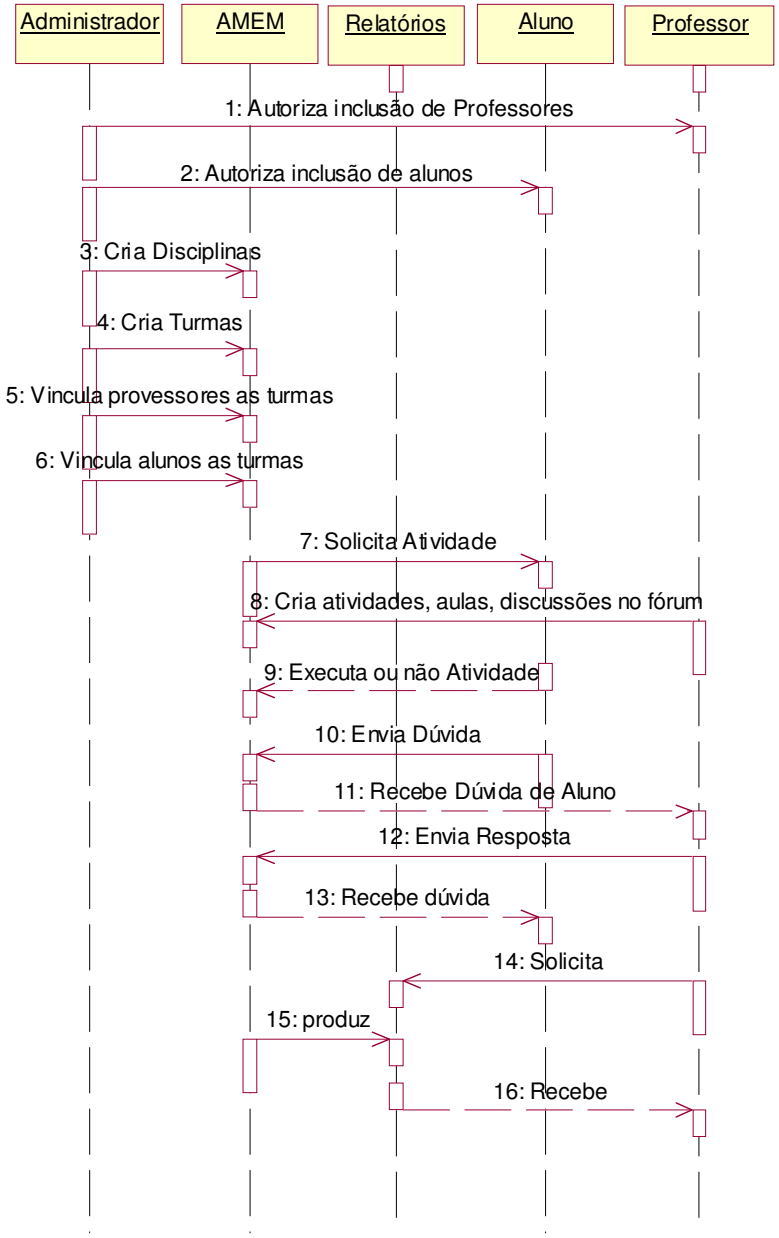


FIGURA 16 – Diagrama de Seqüência do AMEM

Percebe-se pelo diagrama de seqüência, que não foram expostos no mesmo, todas as funcionalidades do AMEM. Abstraiu-se subsistemas do ambiente, como o Fórum, o Tiradúvidas, o Mural On-line e outros, para demonstrar o funcionamento do mesmo externamente, sem o conhecimento prévio de seu conteúdo.

No diagrama, pode-se constatar claramente o perfil e a função de cada entidade, seja ela do administrador, do aluno, do professor, do AMEM ou mesmo a funcionalidade dos relatórios para com os educadores, que nada mais são do que consultas à base de dados.

No próximo item deste trabalho, será mostrado o diagrama de classes da modelagem do sistema do AMEM.

3.2.3 Diagrama de Classes

Uma classe em UML é representada por uma caixa retangular com três compartimentos: um com o nome da classe, o outro com a lista de atributos da classe e o último com a lista de operações da classe.

As associações representam relacionamentos estruturados entre objetos de diferentes classes, e são representados graficamente através de uma linha conectando as classes. Uma associação pode ter um nome. E as extremidades da linha que representa uma associação pode ter nome de papéis mostrando como a classe é vista pelas outras classes na associação.

A multiplicidade de uma associação especifica quantas instâncias de uma classe relacionam-se a uma única instância de uma classe associada. Uma multiplicidade é representada por um intervalo de

valores possíveis, no seguinte formato: limite_inferior. .limite_superior, onde esses limites são valores inteiros (o caracter * pode ser usado como limite_superior para indicar falta de limite).

A agregação é uma forma especial de associação que representa o relacionamento todo-parte entre objetos. Ela é representada incluindo-se um losango na extremidade do objeto todo do relacionamento todo-parte.

A generalização é uma ferramenta poderosa para a abstração. A generalização é um relacionamento existente entre uma classe mais geral (superclasse) e uma classe mais específica (subclasse), onde a classe mais específica é consistente com a mais geral e adiciona informações a ela. Uma generalização é representada por uma linha com um triângulo, que liga a classe mais específica a mais genérica.

De acordo com estas convenções, modelou-se o diagrama de classes do coração do ambiente AMEM:

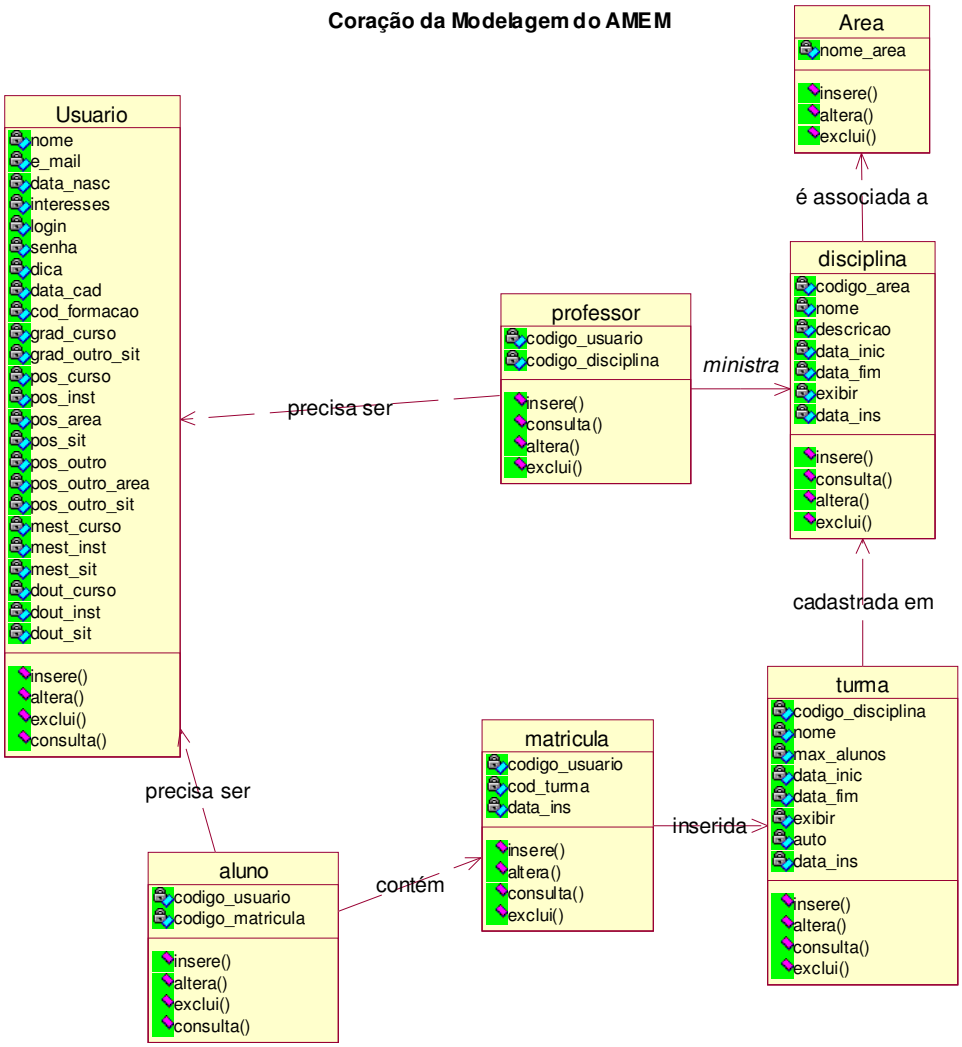


FIGURA 17 – Coração da modelagem do AMEM

Uma vez modelado o diagrama de classes do chamado “Coração do AMEM”, pode-se modelar por partes as outras funcionalidades do AMEM.

Na figura 18, será mostrado o diagrama de classes do sistema de atividades, já demonstrado no modelo entidade-relacionamento descrito neste trabalho anteriormente:

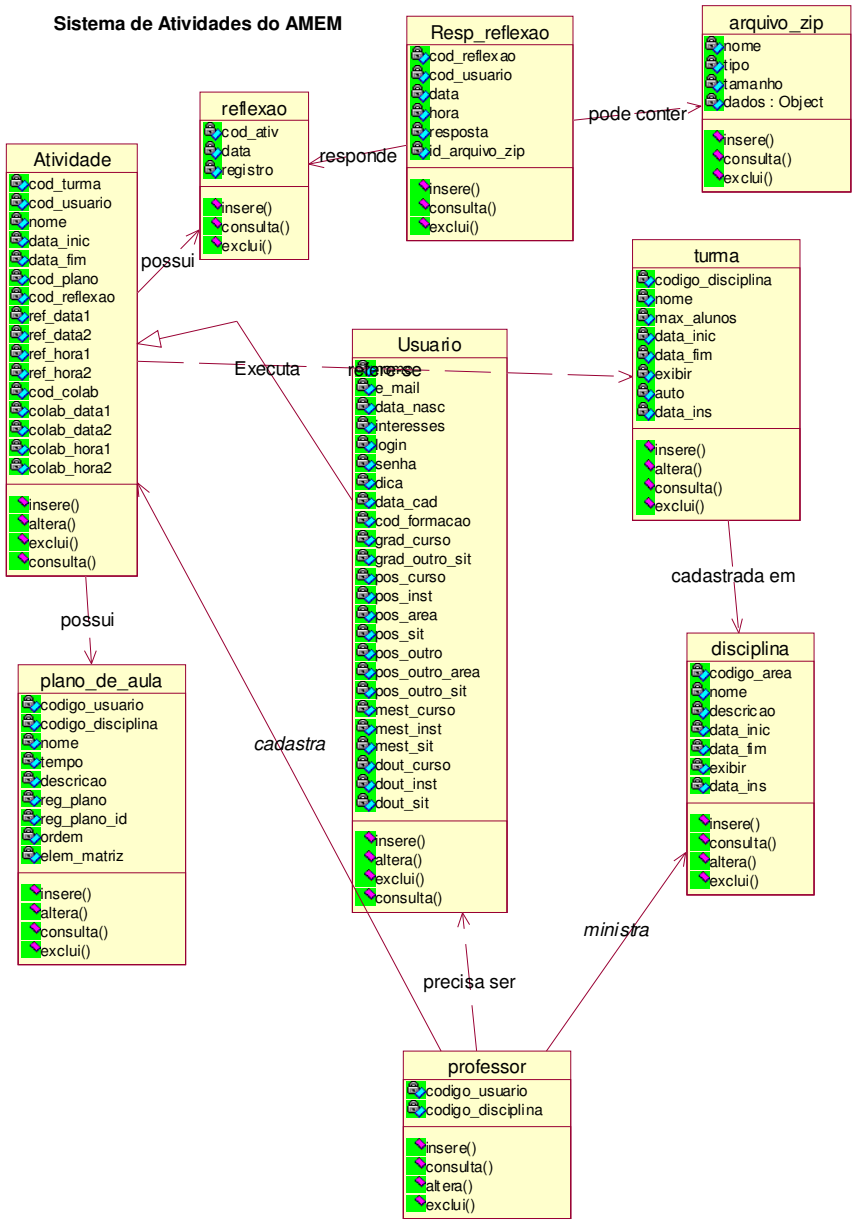


FIGURA 18 – Diagrama de Classes do Sistema de Atividades

A figura abaixo se refere ao sistema da biblioteca do AMEM, integrado com o sistema de atividades, tendo abertura para a inserção de colaboradores, quanto à indicação de bibliografia e atividades.

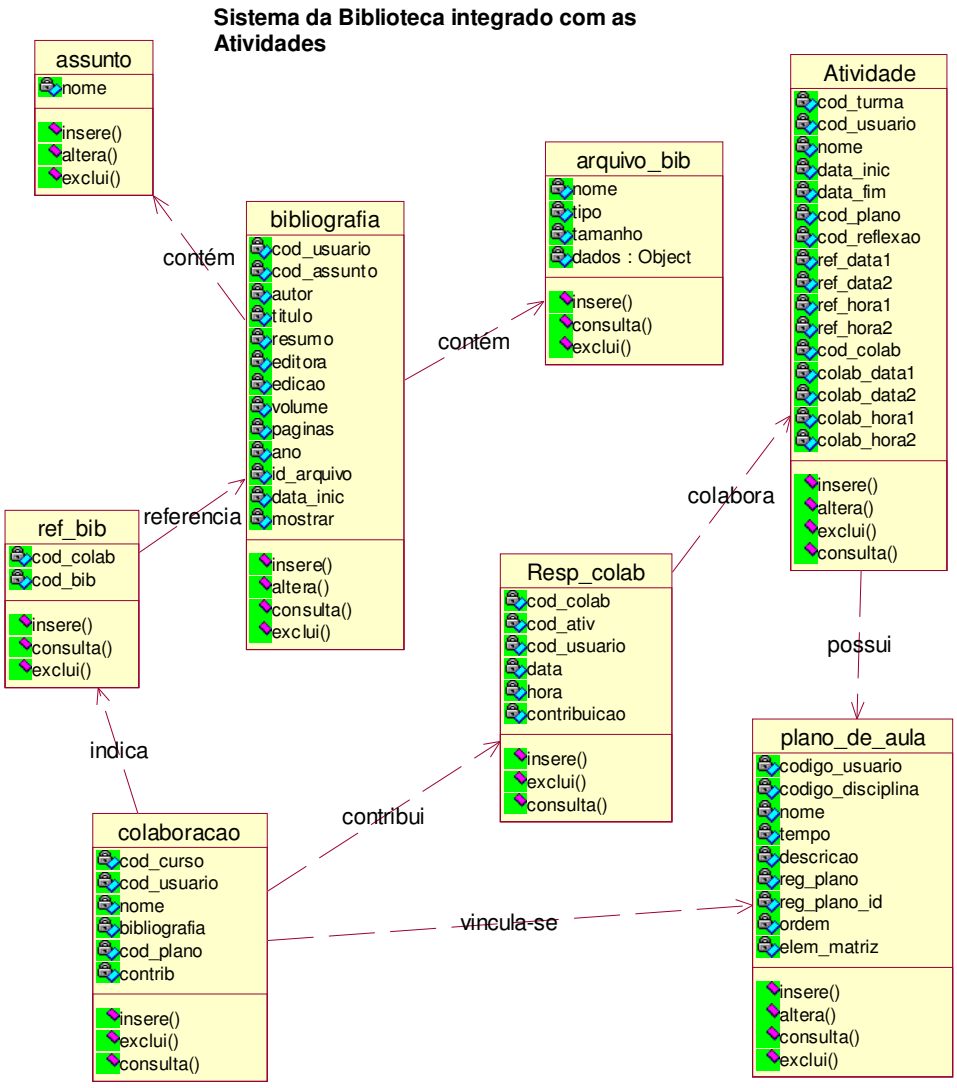


FIGURA 19 – Diagrama de classes da Biblioteca

Nos mesmos moldes anteriores, serão apresentados nas figuras subseqüentes, 20, 21, 22 e 23, a agenda, o mural, o sistema de reuniões on-line e o fórum:

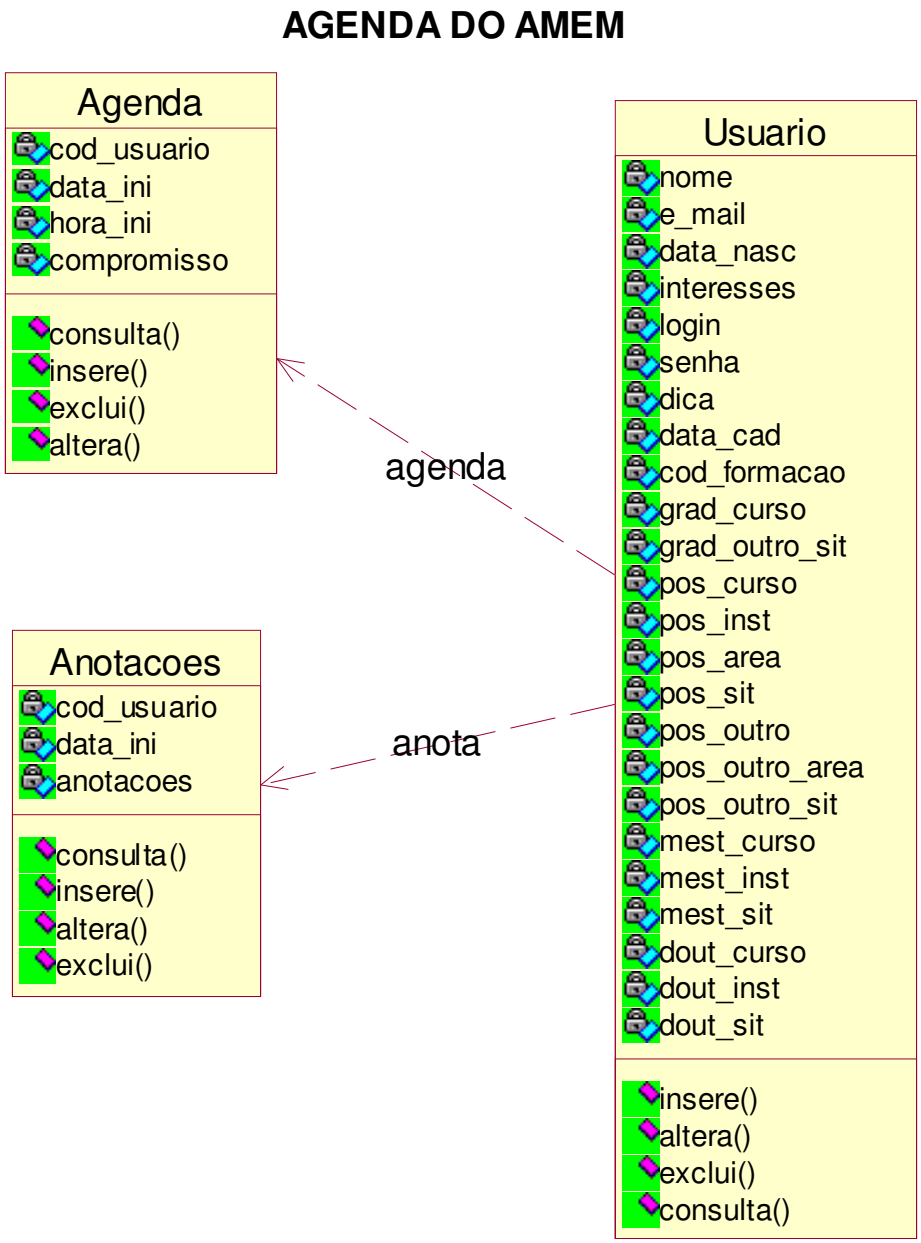


FIGURA 20 – Diagrama de classes da Agenda

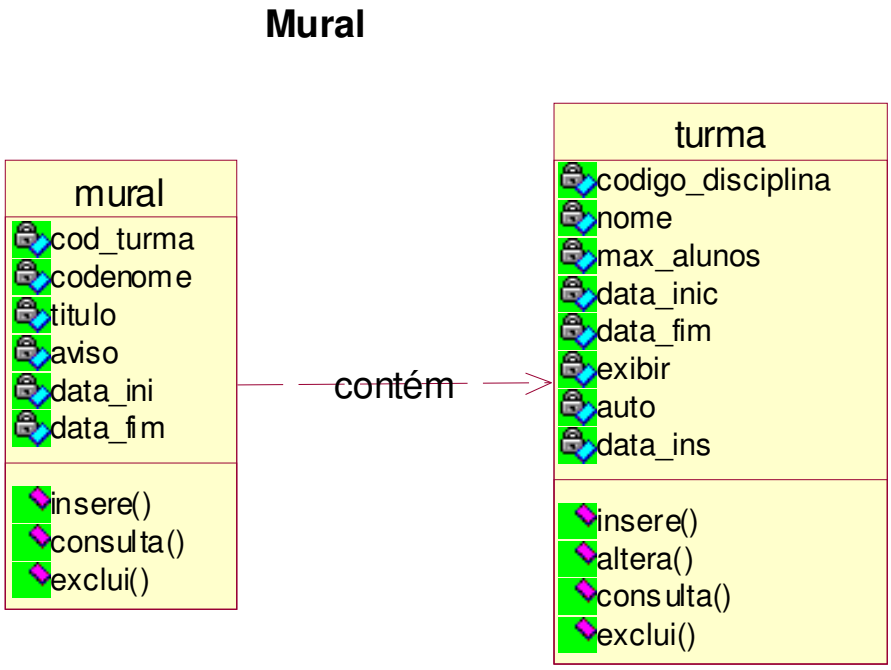


FIGURA 21 – Diagrama de classes do Mural

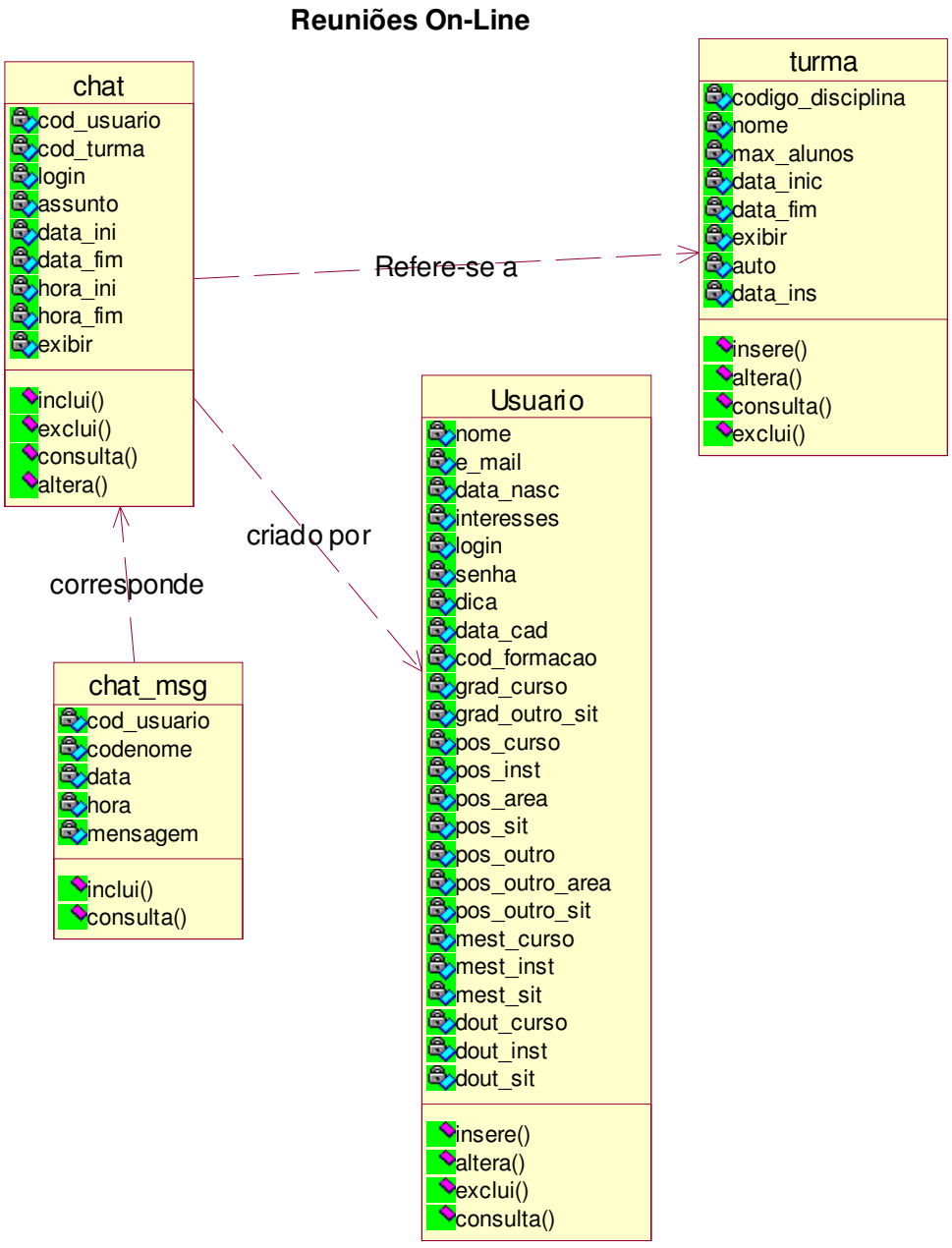


FIGURA 22 – Diagrama de classes das Reuniões On-Line

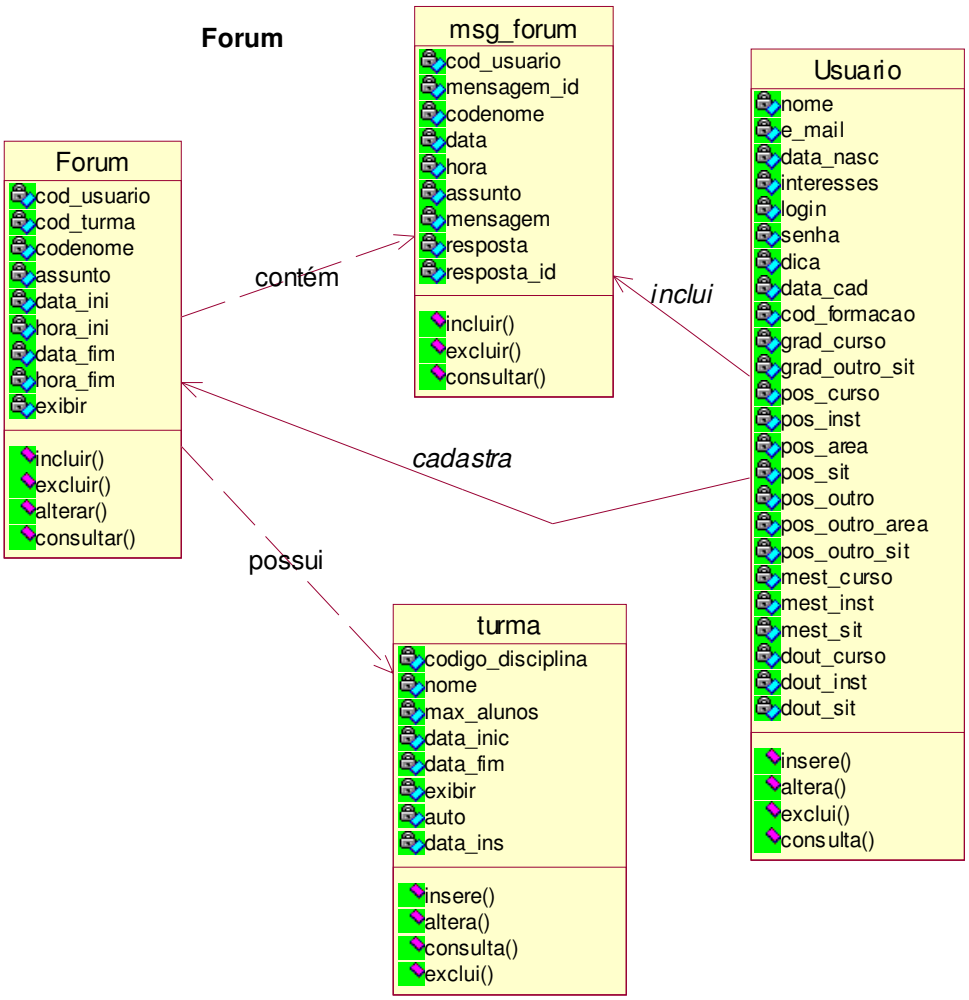


FIGURA 23 – Diagrama de classes do Fórum

A partir de agora, serão inclusos no trabalho, os diagramas de classes das novas funcionalidades propostas ao AMEM, já descritas anteriormente quando apresentado o diagrama Entidade-Relacionamento.

A figura 24 mostra o diagrama de classes do Tiradúvidas e a figura 25, o diagrama de classes do comunicador on-line.

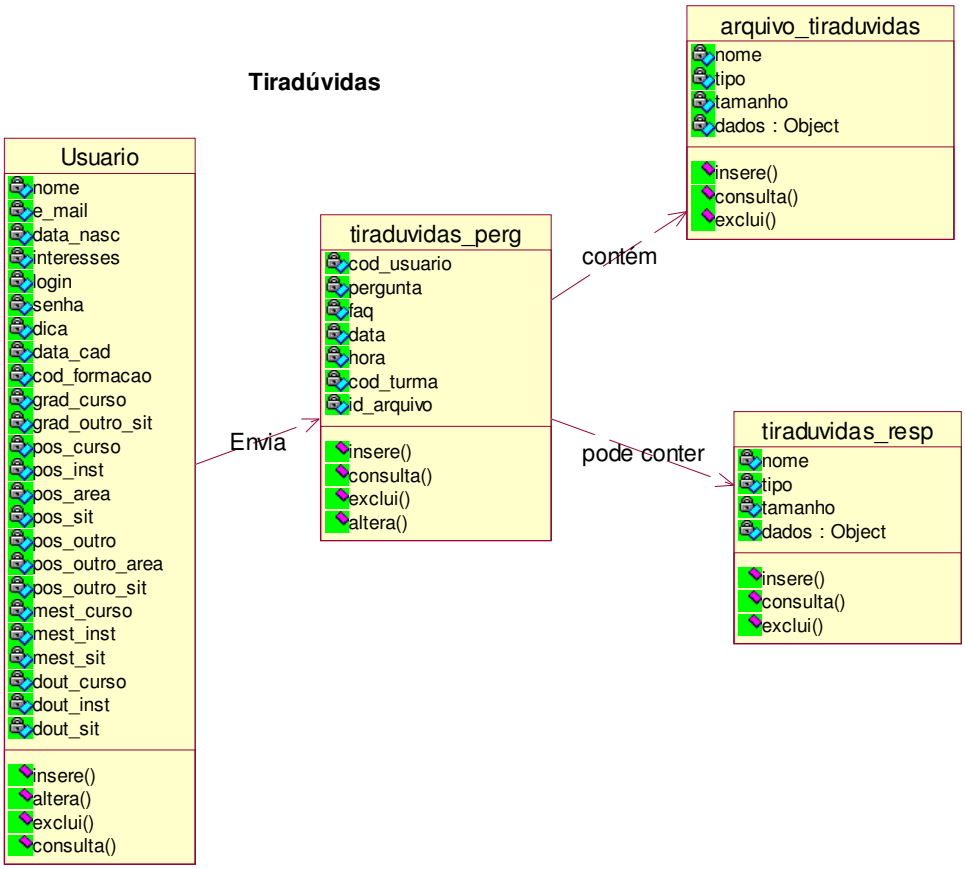


FIGURA 24 – Ferramenta de Tiradúvidas do AMEM

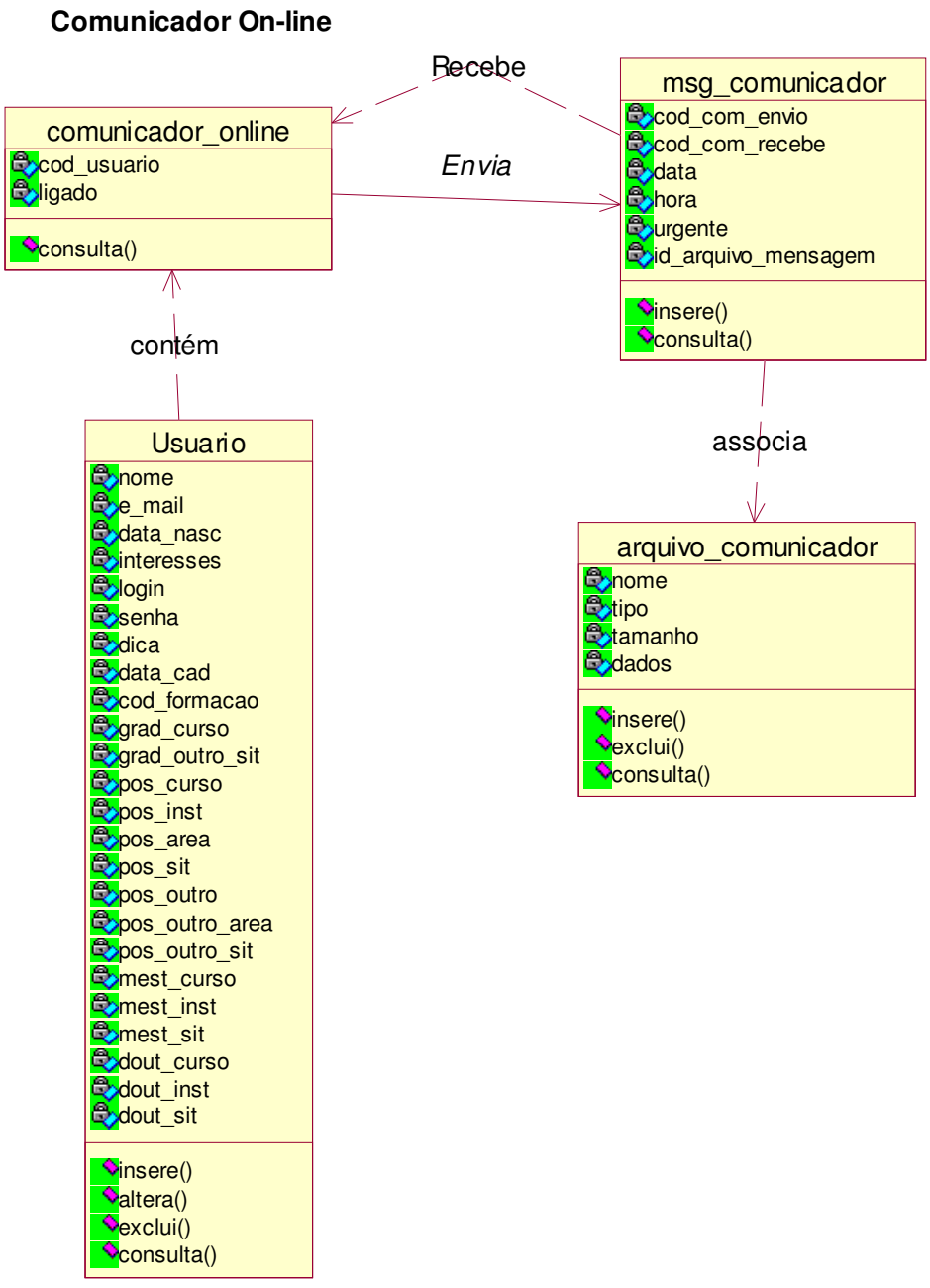


FIGURA 25 – Diagrama de classes do Comunicador On-line

A especificação das funcionalidades de cada módulo do AMEM não será descrita neste trabalho, por já constar em trabalhos anteriores.

4 CONCLUSÃO

O Trabalho de Conclusão do Curso de Ciência da Computação apresentado tratou do processo de engenharia reversa e reengenharia de um ambiente educacional desenvolvido por uma equipe multidisciplinar da UFSM, o qual está em funcionamento baseado em um modelo de dados projetado de forma procedimental.

Com o objetivo de poder portar o ambiente AMEM, de uma plataforma centralizada a ambientes de processamento distribuído em arquiteturas cliente/servidor, por exemplo, o estudo minucioso dos conceitos de engenharia reversa e reengenharia foi fundamental.

Após concluir a modelagem orientada a objetos, baseando-se na reengenharia empregada, é possível mover o sistema de uma arquitetura orientada a ação, para uma arquitetura orientada a objetos, tornando o sistema altamente flexível, adaptável e extensível.

Uma vez implementado o sistema do AMEM, orientado a objetos, será muito mais fácil prever efeitos colaterais, pois os relacionamentos são muito mais visíveis.

A manutenção dos sistemas integrados do AMEM também é facilitada, bem como a reusabilidade do software.

A partir de agora se percebe a existência de meios de abstração bem definidos, proporcionados pela modelagem orientada a objetos. Uma vez aplicada a modelagem, adota-se o conceito de

encapsulamento e comportamentos que efetivamente apóiam o processo de manutenção de software.

Para trabalhos futuros, será muito mais fácil identificar objetos, principalmente porque através deste documento também foram exibidas, explicitamente, as dependências procedimentais do ambiente em questão, evitando a degradação do projeto original durante as manutenções e facilitando o processo de reuso.

5 REFERÊNCIAS BIBLIOGRÁFICAS

[1] DA SILVA, R. T. **Educação mediada por computador** - CCEN/UFPB, Departamento de Física – Brasil.

[2] WILSON, Brent G. (1995). **Constructivist Learning Environments:** Case Studies in Instructional Design - Educational Technology Publications.

[3] FERNÁNDEZ, G. E. **Considerações de Projeto do Ambiente Amem na Perspectiva da Investigação-Ação** – UFSM/RS, Departamento de Eletrônica e Computação. Santa Maria - Brasil.

[4] RECCHIA, L. E. & PENTEADO, R. **FaPRE/OO:** Uma Família de Padrões para Reengenharia Orientada a Objetos de Sistemas Legados Procedimentais – DC/SP, Universidade Federal de São Carlos. São Carlos – Brasil.

[5] CHIKOFSKY, J. E. & CROSS, J. H. **Reverse Engineering and Design Recovery:** A Taxonomy. IEEE Software, v. 7, n. 1, p. 13-17, Jan. 1990.

[6] BRAGAGLIA, M. U. **Software para Modelagem de Banco de Dados Objeto-Orientados usando IDEFobjetc** – UNIVALI, Universidade do Vale do Itajaí, Curso de Ciência da Computação. Itajaí – SC.

[7] BURLESON, Donald K. **Inside the Database Object Model**. 1. ed. Boca Raton: CRC Press LLC, 1998. 226p. ISBN 0-8493-1807-6.

[8] KERN, Vinícius M. **Bancos de Dados Relacionais - Teoria e Prática de Projeto**. São Paulo: Érica, 1994. 236 p.

[9] FERNÁNDEZ, G. E. **A Informática o Desenvolvimento de Ambientes Web** – UFSM/RS, Departamento de Eletrônica e Computação. Santa Maria – Brasil, 2002.

[10] YOURDON, Edward. **Análise Estruturada Moderna**. Rio de Janeiro: Ed. Campus, 1992.

[11] FERNÁNDEZ, G. E.. **Ambiente Multimidia para Educação mediada por computador na perspectiva da IAE:modelagem e implementação** – UFSM/RS, Departamento de Eletrônica e Computação. Santa Maria – Brasil, 2003.

[12] LAUERMANN, C. A. R. **Ambiente Multimídia para Educação Mediada por Computador na Perspectiva da IA:** Avaliação e Tutorial – UFSM/RS, Departamento de Eletrônica e Computação. Santa Maria – Brasil, 2002.

[13] PRESMAM, Edward R. **Engenharia de Software.** 3. ed. Ed. Makron Book, 1995.

[14] AZOLIN, Beatriz R. **Estudo e Implementação das Interfaces Homem-Máquina de um Ambiente Multimídia Para Educação Mediada Por Computador**. Trabalho de graduação do curso de Informática da Universidade Federal de Santa Maria, Santa Maria, RS: UFSM, 2002.

[15] Blooch, Grady. **Object-oriented analysis and design with aplications** - Second Edition, The Benjamin/Cummings Publishing Company, Inc., 1994.

[16] RUMBAUGH, James *et al*. **Object-Oriented Modeling and Design** - Prentice-Hall, Inc., 1991.

[17] ALFONSECA, M. Alcalá, A. **Programación Orientada a Objetos**. Anaya Multimedia, Madrid, 1992.

[18] BECK, K. & CUNNINGHAM, W. A laboratory for teaching object-oriented thinking. Proc. of **Object-Oriented Programming Systems, Languages and Applications** 1989

(OOPSLA '89). SIGPLAN Notices, Vol. 24, No. 10, October 89, pp 1-6.

[19] ALFONSECA, M. **Multimedia Ediciones S.A. Curso IBM de Programación**. Unidades 38 a 41.

[20] ALFONSECA, M. Frames. **Semantic Networks and Object-Oriented Programming in APL2. IBM** J. Res. Dev., 33:5, p. 502-510, Sep. 1989.

[21] COX, Brad. **Object-oriented Programming:** an evolutionary approach. Addison-Wesley, 1986.

[22] HOPKINS, T. **A first Course in Smalltalk 80**. Prentice Hall, Inc., Englewood Cliffs, NJ, 1991.

[23] BURNS, A. & DAVIES, G. **Concurrent Programming**. Ed. Addison-Wesley, 1993.

[24] FERNANDEZ, L. & ARROYO, F. **Teoría y Práctica del Módulo de Programación Orientada a Objetos. Programación en PIIPOO**. Departamento de Publicaciones de la E.U.I. de la Universidad Politécnica de Madrid, 1997.

[25] BOOCH, G. **Análisis y Diseño Orientado a Objetos**. Ed. Addison Wesley, 1996.

[26] CHIKOFSKY, Elliot J. **Computer-aided software engineering (CASE)**, 1993.

[27] WARDEN, R. **Re-engineering - A Pratical Methodology With Commercial Applications. In: Applied Information Tecnology 12 (Software Reuse and Reverse Engineering in Practice).** (P. A. V. Hall, ed.) - Chapman@Hall, 1992.

[28] PREMERLANI, W. J. e BLAHA, M. R. **An Approach for Reverse Engineering of Relational Databases. Communications of the ACM**, v. 37, n. 5, 1994, p. 42-49.

[29] SOMMERVILLE, I. **Software Engineering (International Computer Science Series)**. 5a Edição. Reading: Addison-Wesley, 1995.