Utilização da Metodologia OMT na Construção de Ferramentas CASE*

Renato Silva Cabral† Sandra de Albuquerque Jansen Patrícia Porto Carreiro
Jaelson Brelaz de Castro

Departamento de Informática
Universidade Federal de Pernambuco
Caixa Postal 7851, Recife, PE, Brasil
CEP 50732-970, Tel.:(081) 271-8430, Fax: (081) 271-4925
E-mail:rsc,saj,ppc,jbc@di.ufpe.br

Sumário

Este trabalho descreve a experiência de utilização de uma metodologia orientada a objeto no desenvolvimento de ferramentas CASE (Computer Aided Software Engeneering). Foi feito um estudo comparativo entre algumas metodologias orientadas a objeto, escolhendo-se a OMT (Object Modeling Technique) para a construção de uma ferramenta. Ao final, foi possível identificar as vantagens e desvantagens do uso da metodologia orientada a objeto OMT no processo de desenvolvimento.

Abstract

This work describes an experience in the use of an Object-Oriented methodology for development of CASE (Computer Aided Software Engeneering) tools. We provide a comparative study of some object oriented methodologies and choose OMT (Object Modeling Technique) for the construction of a tool. We conclude the paper discussing the benefits and pitfalls of the use of OMT object oriented methodology for the development process.

1 Introdução

O processo de desenvolvimento de software se beneficia do uso de ferramentas CASE na busca de uma maior produtividade e qualidade [10]. Portanto, uma das áreas ativas da engenharia de software é a produção deste tipo de ferramenta. Assim, foram identificadas propriedades necessárias às ferramentas CASE, tais como:

- facilidades de manutenção e extensão, permitindo a adaptação das várias aplicações que irá auxiliar;
- ser reusável, permitindo que suas características possam ser absorvidas para a construção de outras ferramentas;
- possuir uma interface bem definida, facilitando a interação com outras ferramentas.

O paradigma de Orientação a Objetos (OO), utilizado com sucesso em linguagens de programação, possui conceitos (encapsulamento, herança, abstração) que vêm atender às propriedades acima descritas.

Para o desenvolvimento de software é fundamental o uso de metodologias que o torna menos empírico, garantindo-lhe maior precisão e confiabilidade.

*Esse trabalho foi desenvolvido com o apoio do CNPq e da CAPES.
†Funcionário licenciado da SGA Sistemas e Serviços - Brasília/DF em programa de pós-graduação.

A integração do paradigma OO às metodologias de desenvolvimento acrescentam maior capacidade para capturar o grau de abstração necessário às aplicações. É, principalmente, neste sentido que as metodologias OO se mostram mais adequadas que as tradicionais. Por exemplo, no Departamento de Informática da UFPE várias ferramentas, utilizando tecnologias OO, estão em desenvolvimento para apoiar a produção de software. Este trabalho, em particular, apresenta a modelagem de um editor gráfico que pode se adequar a outros tipos de ferramentas CASE.

Recentemente, novas metodologias OO têm sido propostas. Entretanto, ainda não há um consenso de qual proposta possui a melhor abordagem. Na seção 2 é, então, feita uma avaliação de algumas metodologias OO, e uma destas é selecionada descrevendo-se as razões que levaram a esta escolha. Na seção 3 aborda-se mais detalhadamente a metodologia escolhida. Na seção 4 são apresentados os resultados obtidos no desenvolvimento de uma ferramenta CASE. Na seção 5 algumas considerações e críticas sobre a técnica utilizada, relativas à aplicação, são descritas.

2 Avaliação de Metodologias Orientadas a Objeto

Esta seção apresenta um levantamento parcial dos trabalhos que fazem uso do paradigma OO em metodologias de desenvolvimento de software. Foram utilizados estudos comparativos de metodologias OO a fim de escolher uma que mais se adequasse à aplicação. Uma descrição detalhada das metodologias pode ser encontrada nas próprias referências citadas a seguir ou no livro [5], que contém uma descrição mais profunda e uma análise crítica das mesmas.

Todos os trabalhos escolhidos são apresentados em livros que utilizam notações representando os conceitos fundamentais de OO. Outro critério de seleção de bibliografia, foi que as metodologias não dependessem de linguagens de implementação.

2.1 Metodologias Avaliadas

Através de alguns estudos comparativos ([1], [9], [13], [15]), foram destacadas as metodologias OO resumidas a seguir. O levantamento apresentado diz respeito às fases de Análise e Projeto do desenvolvimento das metodologias. Para cada metodologia são resumidas algumas características e as notações propostas.

2.1.1 Booch:

A metodologia de Booch [2] se baseia no modelo espiral [18] de desenvolvimento. O método não é bem definido com respeito ao processo, no entanto oferece um guia implícito através de uma descrição detalhada de cinco exemplos de projetos. Dá ênfase às fases de Projeto e Implementação. Recomenda o uso de prototipação.

A notação proposta é bastante expressiva, podendo ser adaptada ao desenvolvimento em larga escala. Os elementos de sua notação são:

- Diagramas estáticos - que representam classes, objetos, módulos e processos;
- Diagrama dinâmicos - que representam transições de estados e "timing". Dá suporte à comunicação entre classes.
- Templates - mecanismo eficiente de documentação.

2.1.2 Coad & Yourdon:

Este método [8] é voltado para a fase de Análise com alguma orientação para o Projeto e Implementação. Possui um conjunto de heurísticas muito rico para identificar classes. A comunicação entre estas classes (troca de mensagens) é feita usando a visão cliente-servidor.

Duas novas terminologias são usadas: os *assuntos* e os *serviços*. A notação utilizada baseia-se em:

- Diagrama de objetos e classes - são extensões do Modelo Entidade e Relacionamento (E-R) [7]. Apresenta uma filosofia modular através do uso de 5 camadas (classes e objetos, estruturas (generalização-especialização, todo-parte), assuntos, atributos, e serviços);
- Diagrama de estado [12];
- Diagrama de serviço - auxilia na representação do modelo cliente-servidor.

2.1.3 Rumbaugh et. all:

A metodologia OMT (Object Modeling Technique) [17] é muito bem definida, cobrindo as fases da Análise, Projeto e Implementação, e dando ênfase à fase de Análise.

A notação é concisa e muito expressiva. Esta técnica procurou estender notações, já bastante difundidas na área de desenvolvimento de software, para capturar os conceitos de OO. Assim, esta metodologia representa uma mudança incremental [9] às metodologias convencionais, gerando pouco impacto aos que desejam passar destas às metodologias OO. A notação caracteriza os seguintes diagramas:

- Diagrama de objetos e classes - representa a estrutura estática do sistema. Pode ser considerado um Modelo Entidade e Relacionamento estendido [7];
- Diagrama de estado - representa o aspecto dinâmico do sistema. Utiliza a notação de "statecharts" [12];
- Diagrama de fluxo de dados [20] e [11] - representa os aspectos funcionais do sistema;
- Diagrama de arquitetura do sistema - captura o relacionamento estático entre os subsistemas.

2.1.4 Wirfs-Brock et. all:

Esta metodologia [19] utiliza a visão cliente-servidor, enfatizando o comportamento dinâmico e as responsabilidades das classes.

A ênfase é dada às fases de Projeto e Implementação. O processo é exploratório e informal, sendo mais adequado ao desenvolvimento de pequenos projetos.

A introdução de novas terminologias (*contrato, responsabilidades e colaborações*) podem dificultar a sua aceitação, por representar mudanças radicais em relação às metodologias já utilizadas, conforme analisa [9].

A sua notação é composta por:

- Cartões de classes e subsistemas;
- Diagrama de hierarquia de classes;
- Diagrama de Venn - ajudam a definir as responsabilidades das classes.
- Diagrama de colaborações entre classes - representa o comportamento dinâmico destas.
- Cartões de contrato - documentam todos os contratos entre clientes e servidores.

2.2 Observações

Nesta subseção são feitas algumas observações relativas à avaliação realizada.

- Na avaliação foi observado que os processos de Análise e/ou Projeto das metodologias OO são, de certa forma, semelhantes: Primeiro, partindo dos requerimentos, são identificados os objetos do domínio do problema, a semântica e o comportamento dos mesmos. Posteriormente, são definidos como os objetos interagem, seus relacionamentos, atributos e suas estruturas; além de como estas características serão implementadas. Porém, cada uma das metodologias possuem maneiras próprias de como identificar e representar os objetos e seus relacionamentos, com orientações, terminologias e notações diferentes.

- Na literatura não existe uma padronização para as fases de Análise e Projeto das metodologias OO, tornando-se difícil a avaliação e comparação das mesmas. Observa-se, por exemplo, que alguns processos definidos na Análise de uma metodologia são incluídos no Projeto de outra, não sendo claro o limite e a distinção entre estas etapas.

2.3 Escolha da Metodologia

A partir desta avaliação a metodologia de Rumbaugh foi a escolhida. As seguintes razões levaram a esta escolha:

- os conceitos de OMT são suportados por notações de conhecimento e uso já difundidos;
- OMT consegue um equilíbrio entre a facilidade de aprendizado e a expressividade da notação, ao contrário de outras metodologias, como a de Booch, que possui a notação mais rica, mas não é tão simples de assimilar;
- é um processo bem definido que cobre todas as fases do desenvolvimento (Análise, Projeto e Implementação);
- a fase de Análise possui uma notação muito rica para representar as visões estruturais, dinâmicas e funcionais das aplicações.

A próxima seção, tem por objetivo descrever mais alguns detalhes da metodologia OMT definida por Rumbaugh et all, apresentando suas principais etapas.

3 Object Modeling Technique - OMT

OMT é uma metodologia, onde o processo de desenvolvimento se baseia na construção de três modelos (Modelo Objeto, Modelo Dinâmico, Modelo Funcional) que descrevem o sistema, oferecendo três visões diferentes e complementares do mesmo. A base de OMT, justamente por caracterizar uma metodologia OO, é o Modelo Objeto[1].

A OMT atua, principalmente, nas fases de Análise e Projeto, propondo a subseqüente fase de Implementação.

A fase de Análise enfatiza o processo conceitual de desenvolvimento de software e independe da Implementação. Os requisitos da aplicação são representados através dos modelos construídos na Análise. Estes modelos são descritos a seguir.

- O Modelo Objeto (MO) captura a estrutura estática do sistema, representando os objetos, seus relacionamentos, atributos e operações. A questão - quais são os **objetos** e suas relações ? - direciona a construção do Modelo Objeto. A notação utilizada é, na verdade, uma extensão da modelagem E-R [7] pois, combina conceitos de orientação a objetos (classe e herança) e de modelagem de informação (entidades e associação).
- O Modelo Dinâmico (MD) representa aspectos temporais e comportamentais do sistema, capturando o controle e o seqüenciamento de operações. A questão - quais são os estímulos aos **objetos** e suas respostas? - direciona a construção do Modelo Dinâmico. Emprega a notação de diagramas de estado estendidos ("statecharts") definida por [12].
- O Modelo Funcional (MF) descreve o aspecto de transformação de dados dentro do sistema. A questão - quais são as computações que cada **objeto** executa ? - direciona a construção do Modelo Funcional. Utiliza a notação da análise estruturada descrita, entre outros, em [11] e [20]

[1] Diferentemente da análise e projeto estruturado que também suporta as três visões, mas dá ênfase ao Modelo Funcional.

Nas fases de Projeto (Projeto de Sistemas e Projeto de Objeto) os modelos da Análise são transformados e expandidos determinando o Projeto, que deve ser direcionado à Implementação.

- O Projeto de Sistemas determina uma arquitetura geral do sistema, organizando-o em subsistemas. Além disso, define os aspectos mais gerais referentes à implementação, tais como: plataforma de desenvolvimento, concorrência, gerenciamento de banco de dados, definição de prioridades nas estratégias de implementação, entre outros.
- O Projeto de Objeto adiciona ao Modelo Objeto as operações determinadas pelos modelos Dinâmico e Funcional. A extensão do Modelo objeto passa pela definição de estruturas de dados, algoritmos, otimizações, e outros aspectos que direcionam a Implementação do sistema.

Por fim, a metodologia define alguns aspectos relevantes que devem ser observados na Implementação. OMT não aborda as fases de Teste e Manutenção do ciclo de desenvolvimento de software.

Uma síntese da metodologia é ilustrada na figura 1, encontrada em [3].

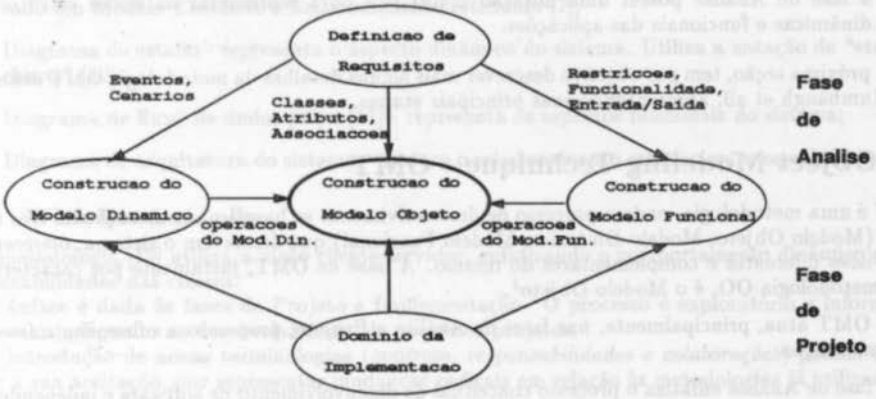

Figura 1: Síntese da metodologia OMT

A seguir, descrevemos a aplicação da metodologia OMT, para a construção de uma ferramenta CASE, considerando cada fase do processo de desenvolvimento.

4 Aplicação da OMT

A OMT foi empregada na Análise e Projeto de uma ferramenta gráfica - FERRAMENTA OOTOOL[2] - para edição de diagramas do Modelo Objeto desta metodologia. A seguir, é descrita cada uma de suas fases: Definição de Requisitos, Análise (com os Modelos Objeto,Dinâmico e Funcional) e Projeto (com os Projetos de Sistema e de Objeto).

A notação da OMT empregada neste artigo, não será detalhada por questões de espaço. Apenas a figura relacionada com o Modelo Objeto (figura 2) possui uma ilustração parcial da notação utilizada, por ser menos conhecida.

[2]A definição completa desta ferramenta pode ser encontrada em [4], [16]

4.1 Definição de Requisitos

Foram definidas as características necessárias de uma ferramenta CASE, para dar suporte à edição gráfica dos diagramas. Estes diagramas devem representar a notação específica do Modelo Objeto. A estrutura básica tratada pela ferramenta é a seguinte (a definição de requisitos completa pode ser encontrada em [4], [16]):

O editor manipula cada diagrama como um conjunto de páginas de tamanho fixo. As páginas podem conter caixas, representando classes/objetos, e ligações, correspondendo às associações entre as classes. Relacionados às caixas e às ligações podem existir textos com tipos e posições distintas, de acordo com suas funções (ver figura 2).

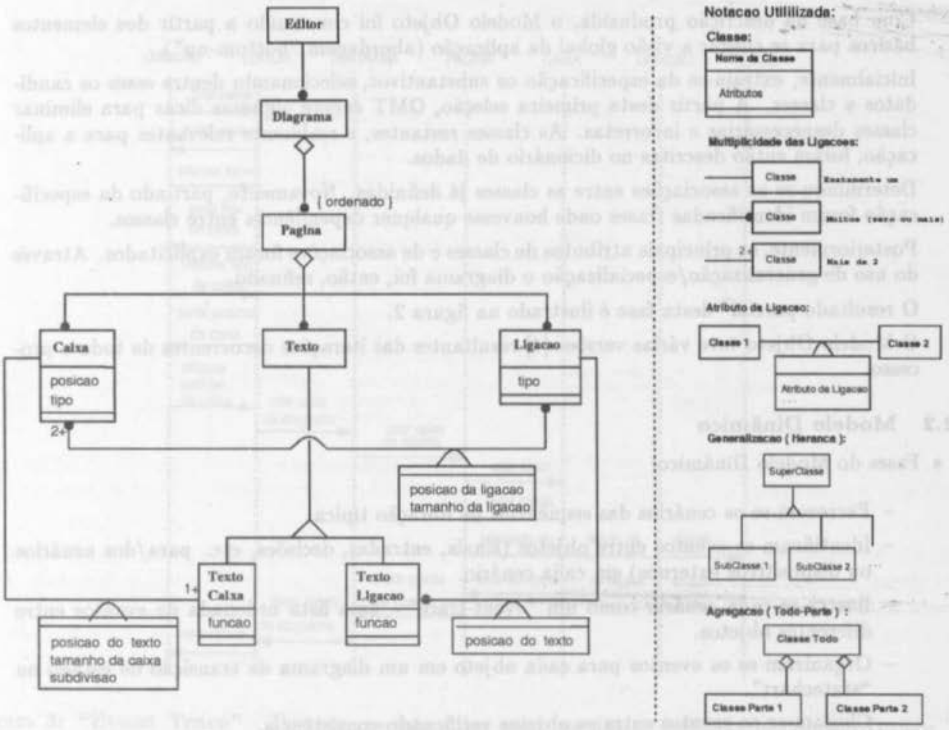

Figura 2: **Classes principais com associações e atributos** - A classe Editor é responsável pela interface com o usuário, podendo tratar vários diagramas que são compostos por Páginas, que por sua vez são compostas por Caixas, Textos e Ligações.

A fim de situar os leitores que desconhecem a metodologia, nas subseções seguintes, são descritos resumidamente os passos de cada fase, juntamente com a aplicação destes à construção da ferramenta.

4.2 Análise

Esta etapa se baseia na Definição dos Requisitos gerando um modelo (composto por três visões) do mundo real, contendo suas principais características. Se divide nas três subetapas descritas a seguir.

4.2.1 Modelo Objeto

- Fases do Modelo Objeto:
  - A partir da descrição inicial do problema, identificam-se objetos e classes.
  - Prepara-se um dicionário de dados que consista das descrições de cada classe.
  - Identificam-se associações entre classes e atributos de classes.
  - Organizam-se classes usando herança.
  - Repete-se este processo eliminando classes e associações redundantes.
- Elaboração do Modelo Objeto de OOTOOL:

  Com base na descrição produzida, o Modelo Objeto foi construído a partir dos elementos básicos para se chegar a visão global da aplicação (abordagem "bottom-up").

  Inicialmente, extraiu-se da especificação os substantivos, selecionando dentre esses os candidatos a classes. A partir desta primeira seleção, OMT sugere algumas dicas para eliminar classes desnecessárias e incorretas. As classes restantes, e realmente relevantes para a aplicação, foram então descritas no dicionário de dados.

  Determinou-se as associações entre as classes já definidas. Novamente, partindo da especificação foram identificadas frases onde houvesse qualquer dependência entre classes.

  Posteriormente, os principais atributos de classes e de associações foram explicitados. Através do uso de generalização/especialização o diagrama foi, então, refinado.

  O resultado parcial[3] desta fase é ilustrado na figura 2.

  O Modelo Objeto teve várias versões [4] resultantes das iterações decorrentes de todo o processo.

4.2.2 Modelo Dinâmico

- Fases do Modelo Dinâmico:
  - Escrevem-se os cenários das sequências de iteração típica.
  - Identificam-se eventos entre objetos (sinais, entradas, decisões, etc. para/dos usuários ou dispositivos externos) em cada cenário.
  - Ilustra-se cada cenário como um "event-trace" - uma lista ordenada de eventos entre diferentes objetos.
  - Organizam-se os eventos para cada objeto em um diagrama de transição de estado ou "statechart".
  - Checam-se os eventos entre os objetos verificando consistência.
- Elaboração do Modelo Dinâmico de OOTOOL:

  Para construção do Modelo Dinâmico é necessário a idealização de como se dará a interação sistema-usuário. Um esboço da interface foi, deste modo, construído. Em seguida, foram elaborados os cenários de eventos representando esta interação somente considerando os casos normais. Como exemplo, considere o cenário abaixo, que representa a operação de criação de Caixa[4], relacionando o Usuário com o Sistema (composto por todos os objetos significantes):

[3] A versão completa está apresentada em [4], [16].

[4] Este objeto se refere a notação da OMT que representa classes ou objetos, conforme pode ser observado na figura 2.

Usuário pede para criar Caixa

Sistema pede nome, tipo e posição da Caixa

Usuário informa nome, tipo e posição da Caixa

Sistema cria Caixa

Estes cenários são traduzidos para os "event-traces", onde é ilustrado como um evento, que parte do usuário, interage com as classes do Modelo Objeto (ver figura 2). Nesta etapa não foram considerados os casos de exceção, como por exemplo, o cancelamento de uma operação pelo usuário, disco cheio, etc. O exemplo do "event-trace"[5] correspondente ao cenário anterior, é indicado na figura 3.

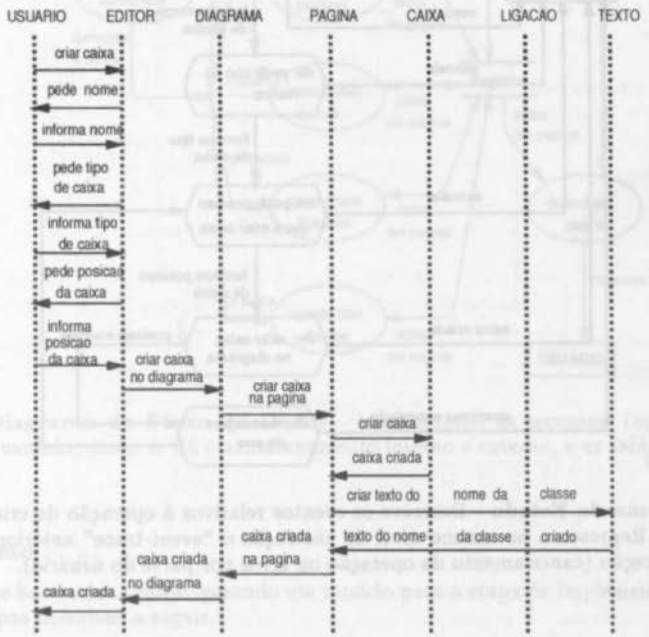

Figura 3: **"Event Trace"** - Descreve as interações entre as classes definidas no Modelo Objeto para a operação de criação de Caixa requerida pelo usuário. Representa o cenário descrito anteriormente, onde o Sistema foi decomposto nas classes Editor, Diagrama, Página, Caixa, Ligação, Texto. Observe que a operação é propagada atingindo todos os objetos (Diagrama, Página, Caixa e Texto) que necessita para a sua realização

[5]Todos os "event-traces" com seus respectivos cenários estão apresentados em [4], [16].

Partindo dos "event-traces" os diagramas de estado foram construídos para cada classe do sistema com comportamento significativo. Neste momento, torna-se importante a inclusão dos casos de exceção. O diagrama de estado, correspondente ao "event-trace" anterior, para a classe Editor, é mostrado na figura 4.

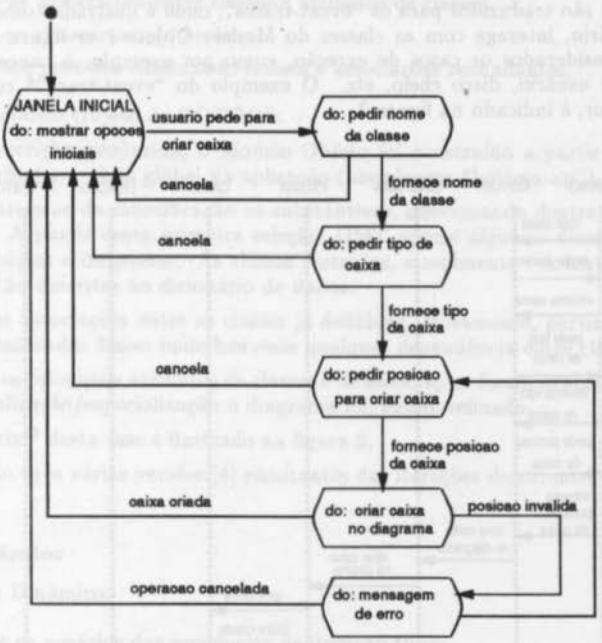

Figura 4: **Diagrama de Estado** - Descreve os eventos relativos à operação de criação de Caixa da classe Editor. Representa no mesmo nível de abstração o "event-trace" anterior, incluindo os casos de erro e exceção (cancelamento da operação ou erros por parte do usuário).

4.2.3 Modelo Funcional

- Fases do Modelo Funcional:
  - Identificam-se os valores de entrada e saída (parâmetros de eventos entre o sistema e fora dele) na descrição do problema.
  - Constroem-se diagramas de fluxo de dados mostrando como cada valor de saída é computado partindo dos valores de entrada.
  - Escreve-se uma descrição de cada função.
  - Identificam-se restrições entre objetos.
  - Especifica-se um critério de otimização.

- Elaboração do Modelo Funcional de OOTOOL:

  O Modelo Funcional foi baseado principalmente nas interações modeladas pelos diagramas de estado do Modelo Dinâmico. A partir destes, foi possível determinar as entradas e saídas do sistema e montar os diagramas de fluxo de dados (DFD's).

O DFD de mais alto nível de abstração[6] é o indicado na figura 5.

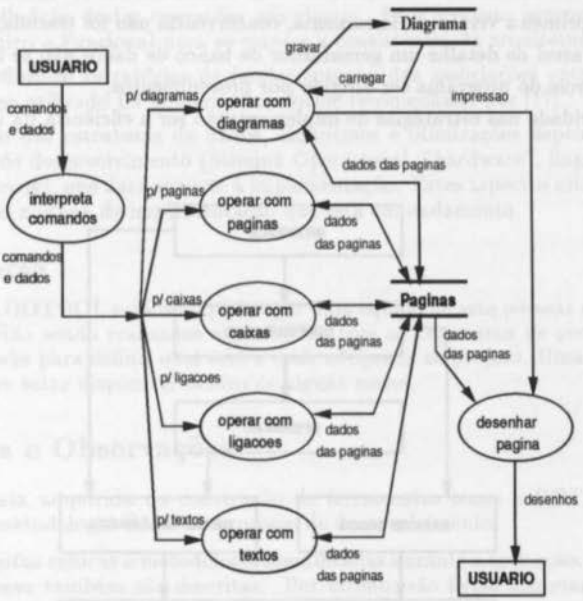

Figura 5: **Diagrama de Fluxo de Dados** - Mostra como os processos (operações) tratam as entradas do usuário, como se dá o armazenamento interno e externo, e as saídas para o usuário

4.3 Projeto

Esta etapa se baseia na Análise, gerando um modelo para a etapa de Implementação. Se divide em duas subetapas descritas a seguir.

4.3.1 Projeto de Sistema

- Fases do Projeto de Sistema:
  - Organiza-se o sistema em subsistemas.
  - Identifica-se concorrência.
  - Alocam-se subsistemas e tarefas para os processadores.
  - Escolhe-se uma aproximação para gerenciamento dos dados armazenados (ex.: arquivos ou banco de dados) e recursos globais (ex.: unidades físicas tais como processadores, espaço tais como espaço do disco, e recursos compartilhados tais como banco de dados).
  - Escolhe-se a implementação de controle: dirigido por evento, sistemas concorrentes ou dirigido por procedimento.
  - Definem-se condições limites e prioridades estratégicas para implementação.
  - O resultado desta etapa é normalmente um diagrama da arquitetura do sistema que captura o relacionamento estático entre os subsistemas.

[6] A expansão dos processos representados neste DFD podem ser encontradas em [4], [16].

- Elaboração do Projeto de Sistema de OOTOOL:

  A arquitetura do sistema foi montada através do agrupamento das classes em subsistemas, de acordo com suas funcionalidades, como indica a figura 6. Outros aspectos da estrutura geral da implementação foram definidos:

  - Nesta primeira versão da ferramenta, concorrência não foi tratada;
  - A este nível de detalhe um gerenciador de banco de dados não se tornou necessário;
  - O controle do programa ser dirigido por procedimentos;
  - A prioridade nas estratégias de implementação ser a eficiência da interface.

Figura 6: **Arquitetura do Sistema** - O Subsistema de Interface é composto pela classe Editor. É responsável por toda interação com o usuário. O Subsistema de Configuração corresponde à classe Diagrama. É responsável pelas operações que lidam com o diagrama como um todo. O Subsistema de Operação é formado pelas classes: Página, Caixa, Ligação, Texto e as respectivas subclasses das três últimas. É responsável pela avaliação dos pedidos de modificação da página corrente. Os Subsistemas de Base de dados e Biblioteca gráfica não são tratados por nenhuma classe definida na análise. São considerados como uma biblioteca da linguagem de programação, ou mesmo independente da mesma. São responsáveis pela interface com o Sistema operacional e Hardware.

4.3.2 Projeto de Objeto

- Fases do Projeto de Objeto:
  - Combinam-se os três modelos para obter operações das classes.
  - Elaboram-se algoritmos para implementar as operações e para otimizar medidas tais como facilidade de implementação, entendimento e perfomance.
  - Ajusta-se a estrutura da classe para melhorar a herança.
  - Define-se como será implementado as associações entre classes.
  - Divide-se em módulos os subsistemas.

- Elaboração do Projeto de Objeto de OOTOOL:

  Esta etapa partiu da complementação do Modelo Objeto com as operações definidas nos outros dois modelos. A funcionalidade dos subsistemas, definidas no Projeto de Sistemas, norteou a distribuição destas operações nas classes. Este processo provocou alterações nos Modelos Dinâmico e Funcional para se manter a consistência da arquitetura.

  Além disso, definiu-se estratégias de implementação das associações entre classes, onde o principal recurso utilizado foi ponteiros, conforme recomendado por [17].

  A determinação das estruturas de dados, algoritmos e otimizações dependem da definição da plataforma de desenvolvimento (Sistema Operacional, "hardware", linguagem de programação e bibliotecas), que dará suporte à implementação. Estes aspectos estão sendo definidos juntamente com a etapa de implementação que está em andamento.

4.4 Implementação

A implementação de OOTOOL está sendo feita por uma equipe de sete pessoas em ambiente Unix de estações SUN. Estão sendo realizadas experiências com as linguagens de programação C++ e Smalltalk/ObjectWorks para definir qual será a mais adequada ao projeto. Uma versão preliminar desta ferramenta deve estar disponível dentro de alguns meses.

5 Conclusões e Observações

Através da experiência adquirida na construção de ferramentas como a OOTOOL, foi possível avaliar o auxílio da metodologia OMT no processo de desenvolvimento.

Nesta seção são feitas críticas à metodologia identificadas durante a aplicação. Algumas decisões tomadas neste processo também são descritas. Por último, são feitas algumas considerações de caráter geral.

- Com relação à fase de Definição de Requisitos:
  - A OMT não orienta como obter e descrever a definição dos requisitos, apesar de enfatizar a importância da mesma. Para esta aplicação foram seguidas as orientações de [18].
- Com relação à fase de Análise:
  - É a fase melhor definida pela metodologia. Seguindo seus passos foi possível construir o resultado da Análise proposto.
  - OMT tem sido criticada em se concentrar demais nesta fase. Entretanto, para esta aplicação, o detalhamento realizado na Análise auxiliou bastante a simplificação das fases posteriores e o entendimento do problema como um todo.
  - A modelagem de certas entidades como atributos ou classes foi uma dificuldade percebida. A metodologia não oferece uma forma eficaz de identificar esta diferença. Como exemplo, para OOTOOL decidiu-se modelar o texto como uma classe (vide figura 2) e não como atributo, por conta da variedade de fontes e funções com localizações específicas que os textos podem ter na notação de OMT para o Modelo Objeto.
  - Uma das vantagens da OMT é a possibilidade de descrever o comportamento dos objetos através do Modelo Dinâmico. No caso de aplicações típicas como ferramentas gráficas, o Modelo Dinâmico tem uma importância fundamental pois, ajuda a modelar todas as interações do sistema com o usuário. Tornou-se assim necessário, para esta aplicação, um maior detalhamento deste modelo [4], [16].
  - Os cenários e os "event-traces" facilitaram bastante a elaboração dos diagramas de estado. Entretanto, os "event-traces" não permitem a visualização de uma operação que atinja mais de um objeto, o que não ocorreu nesta aplicação.

- Para que as operações sejam realizadas, dentro de cada objeto, é necessário que o objeto colabore com outros, através da troca de mensagens. A notação da OMT para o Modelo Objeto não dá suporte a este tipo de interação, como realizado, por exemplo, utilizando-se as *colaborações* de [19].
- A metodologia não aborda devidamente onde deve ser definida a interface do sistema (uma tarefa fundamental para qualquer desenvolvimento), apesar desta informação estar distribuída nos seus vários modelos.
- Foi observado que o Modelo Objeto só pode ser realmente completado e refinado a partir da construção dos outros dois modelos.
- Um ponto forte de OMT é utilizar vários pontos de vista (Modelo Objeto, Modelo Dinâmico e Modelo Funcional) complementarmente, apesar disto, o mapeamento entre eles não é bem definido.

• Com relação à fase de Projeto:

- Apesar da metodologia cobrir as fases de Análise, Projeto e Implementação, as duas últimas não possuem um nível de detalhamento tão amplo quanto a primeira.
- A divisão da fase de Projeto em duas subfases (Projeto de Sistemas e Projeto de Objetos) ajuda a diminuir a diferença do nível de abstração entre as fases de Análise e Implementação.
- Não há correspondência explícita entre a hierarquia das classes definida no Modelo Objeto e as camadas da Arquitetura do Sistema.

• Com relação à metodologia como um todo:

- Apesar deste trabalho ter apenas se centrado na experiência em elaborar uma ferramenta para a edição do Modelo Objeto, o resultado deste desenvolvimento pode ser reutilizado para a construção das ferramentas de edição de diagramas dos demais modelos de OMT (Modelos Dinâmico e Funcional).
- Assim como todas as metodologias de desenvolvimento de software, o sucesso da OMT depende de ferramentas que viabilizem o uso das notações propostas, como também de ferramentas que auxiliem a manter a documentação de todas as iterações do processo. Isto reforça a importância de trabalhos como este, que vêm diminuir a dificuldade em utilizar amplamente as orientações que as metodologias dispõem.
- A abordagem utilizada por OMT está dentro da linha de pesquisa de usar vários esquemas de representação para modelar a abstração das aplicações. Como foi constatado, na OMT a integração dos três modelos não está devidamente representada. Neste sentido, alguns estudos estão sendo realizados, relativos a integração de pontos de vista diferentes de várias metodologias [6], [14].

Outra experiência da utilização da OMT, é relatada em [3]. Neste caso, OMT foi empregada com sucesso, por uma grande equipe, no desenvolvimento de um projeto de porte médio, até a fase de Implementação. Nesse trabalho, foi observado que a notação de OMT facilitou a comunicação entre os componentes da equipe de desenvolvimento, e entre os mesmos e os clientes. Além disso, ressaltam que o uso da herança deve ser feito com muito cuidado para que não atrapalhe, em vez de ajudar, a modelagem e implementação da aplicação.

Neste trabalho, procurou-se averigüar o uso e os benefícios de algumas metodologias OO aplicadas ao desenvolvimento de ferramentas CASE. O relato de experiências deste tipo torna-se muito importante para a orientação de projetistas na escolha de metodologias e a elaboração de novas propostas destas.

Referências

[1] P. Arnold, S. Bodoff, D. Coleman, H. Gilchrist, and F. Hayes. An Evaluation of Five Object-Oriented Development Methods. Technical report, Hewlett-Packard - Information Management Lab., 1991.

[2] G. Booch. *Object-Oriented Design with Applications.* Benjamim/Cummings, 1991.

[3] B. Bruegge, J. Blythe, J. Jackson, and J. Shufelt. Object-Oriented System Modeling with OMT. In *OOPSLA*, pages 359–376, 1992.

[4] P. P. Carreiro, R. S. Cabral, S. A. Jansen, and S. P. Sad. Ferramenta OOTOOL - versão 1.0. Technical report, Depto. de Informática -UFPE, Fevereiro, 1993.

[5] J. B. Castro. *Análise e Projeto Orientado a Objeto.* XXVI Congresso Nacional de Informática e Telecomunicações - SUCESU93, October 1993. (Em impressão).

[6] J. B. Castro and A. Finkelstein. Requirements Elicitation and Formalisation. In *XIX SEMISH*, pages 214–228, September 1992.

[7] P. P. S. Chen. The Entity-Relationship Model - Toward a Unified View of Data. *ACM Transactions on Database Systems*, March 1976.

[8] P. Coad and E. Yourdon. *Object-Oriented Design.* Prentice Hall, Englewood Cliffs, N.J., 1991.

[9] R. G. Fichman and C. F. Kemerer. Object-Oriented and Conventional Analysis and Design Methodologies. *Computer - IEEE*, pages 22–39, October 1992.

[10] G. Forte and R. J. Norman. A Self-Assessment by the Software Engineering Community. *Communications of the ACM*, 35(4):28–32, April 1992.

[11] C. Gane and T. Sarson. *Structured Systems Analysis: Tools and Techniques.* Prentice Hall, Englewood Cliffs, N.J., 1978.

[12] D. Harel. Statecharts: A Visual Formalism for Complex Systems. *Science of Computer Programming*, pages 231–274, 1987.

[13] J. Batista A. Jr. and D. Fonseca. Um Estudo de Metodologias para Desenvolvimento Orientado a Objetos. Technical report, Departamento de Informática - UFPE, 1992.

[14] J. C. S. P. Leite and A. F. Prado. Design Recovery - A Multi-Paradigm Approach. In *First International Workshop on Software Reusability*, Dormund, Germany, July 1991.

[15] D. E. Monarchi and G. I. Puhr. A Research Typology for Object-Oriented Analysis. *Communications of The ACM*, 35(9):35–47, September 1992.

[16] A. J. B. Neto and F. Pimentel et all. Editor de Diagramas para o Modelo Objeto. Technical report, Depto. de Informática - UFPE, Julho, 1993.

[17] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Object-Oriented Modeling and Design.* Prentice-Hall International Editions, 1991.

[18] I. Sommerville. *Software Engineering.* Addison-Wesley, third edition, 1989.

[19] R. J. Wirfs-Brock, B. Wilkerson, and L. Wiener. *Designing Object-Oriented Software.* Prentice Hall, Englewood Cliffs, N.J., 1990.

[20] E. Yourdon. *Modern Structured Analisys.* Yourdon Press, Englewood Cliffs, N.J., 1989.