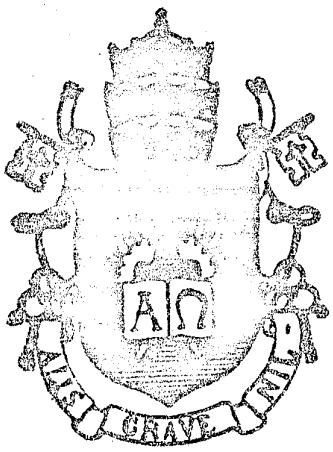

PUC

Série: Monografias em Ciência da Computação,
No. 5/90

O MODELO DE "DESIGN" DO SISTEMA MIDAS DE DESENVOLVIMENTO DE INTERFACES DE USUÁRIO

Regina H.B. Cabral

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO
RUA MARQUÊS DE SÃO VICENTE, 225 - CEP-22453
RIO DE JANEIRO - BRASIL

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

Departamento de Informática

Série: Monografias em Ciência da Computação, Nº 5/90
Editor: Paulo A. S. Veloso Junho, 1990

O Modelo de *Design* do Sistema MIDAS de Desenvolvimento de Interfaces de Usuário

Regina Helena Bastos Cabral

DISCIPLINA : INF 3004 – Exame de Qualificação

ORIENTADOR: Prof. Dr. Bruno Feijó

Junho de 1990

Responsável por publicações:

Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC RIO, Departamento de Informática
Rua Marquês de São Vicente, 225 - Gávea
22453 - Rio de Janeiro, RJ
BRASIL

Tel.: (021)529-9386 TELEX: 31078 FAX: (021)274-4546
BITNET: userrtlc@lncc.bitnet

O Modelo de *Design* do Sistema MIDAS de Desenvolvimento de Interfaces de Usuário

Regina H. B. Cabral

Junho 1990

Resumo

O sistema MIDAS *(Merging Interface Development with Application Specification)* [CABR90] visa possibilitar o *design* interativo de interface de usuário específica da aplicação, em um ambiente orientado para prototipagem, enquanto refina a especificação da própria aplicação pretendida. Para guiar o projetista *(designer)*, MIDAS deverá incorporar em sua base de conhecimento um modelo de ciclo de vida de software. Como sistema para projeto/desenvolvimento de software, sua utilidade será diretamente dependente do quanto ele satisfará às necessidades do projetista na condução do processo de *design*. Neste texto é discutido o modelo de ciclo de vida a ser adotado por MIDAS sob a perspectiva de modelos de *design*, sendo consideradas necessidades cognitivas e computacionais de projetistas de interfaces e de sistemas.

Palavras-chave:. Metodologias de *Design*, Modelos de *Design*, Prototipagem, Modelos de Ciclo de Vida de Desenvolvimento de Software, Sistemas de Desenvolvimento de Interfaces de Usuário

1 Introdução

Uma interface de usuário é considerada como qualquer sistema de software de computador cuja principal função é proporcionar suporte e assistência no uso de algum outro sistema de software, denominado aplicação. O paradigma de interface de usuário é um tipo de notação de especificação que expressa as intenções e desejos do usuário, definindo implicitamente grande parte dos requisitos funcionais do software de aplicacao. Em outras palavras, a especificação da interface de usuário frequentemente é suficiente para obter uma especificação quase completa do sistema de aplicação. Isto é especialmente verdade para aplicações altamente interativas.

MIDAS será um ambiente baseado em conhecimento que pretende dar suporte a projetistas de sistemas em princípios de projeto tanto de interfaces quanto de software de aplicação, tendo como base um modelo de ciclo de vida orientado para prototipagem, que deverá encorajar o desenvolvimento simultâneo da interface com a especificação precisa da aplicação.

Os produtos finais de MIDAS serão uma interface específica da aplicação que poderá posteriormente ser adaptada pelo usuário final (alterando flexivelmente a aparência padrão dos cenários do diálogo), o código de ligação entre a interface e os módulos provisórios *(stubs)* da aplicação, e a especificação do software de aplicação até a lista do que é importado-exportado e a especificação semântica de cada módulo da aplicação.

O entendimento do processo de *design*, i.e., sobre como ele é e pode ser conduzido, é essencial para possibilitar o desenvolvimento de ferramentas, em particular de MIDAS, que auxiliem o projetista no desempenho de sua tarefa. Isto inclui o estudo de como projetistas trabalham e pensam, ou seja, o entendimento de suas necessidades cognitivas. A partir deste estudo, é interessante definir um modelo de *design* que possa ser incorporado (utilizado) na ferramenta de suporte a esta atividade.

Em termos gerais, *design* é aceito na literatura ([AKIN79], [BENT89], [BOEH86], [CROS89], [DAVI87], [GOEL89], [KOEG87], [NADI87], [TAKA87], [VETH87]) como um processo evolutivo a partir de um caso central (protótipo ou requisitos abstratos) que é refinado em direção à especificação (desenvolvimento) de um artefato que atinja desempenho desejado e seja realizável com alto grau de confiança. Como ressaltado em [AKIN79] e [BENT89], as atividades cognitivas de síntese, análise e avaliação são naturais e essenciais ao refinamento entre as etapas deste processo e envolvem formas de raciocínio como dedução, indução, criação e intuição que são dependentes, por exemplo, da experiência do projetista. Em outras palavras, *design* é um processo extremamente complexo que não comporta uma definição absoluta.

Não obstante, em [GOEL89] é apresentada uma discussão sobre características de *design* resultando em um interessante gabarito (*template*) útil para reconhecimento de atividades de *design* em geral e para definição de modelos de *design*, conforme descrito a seguir.

2 Um "gabarito" para modelos de *design*

O gabarito resultante da discussão apresentada em [GOEL89] evidencia aspectos específicos que devem ser considerados em um modelo de *design*. Ele se baseia na identificação de invariantes do ambiente de *design (design task environment)*, i.e. algumas características comuns em ambientes de tarefas consensualmente reconhecidas como atividades de *design*, e dos consequentes invariantes em necessidades cognitivas de projetistas daqueles tipos de tarefas, conforme a Figura 1 (uma adaptação da Figura 2 apresentada em [GOEL89] de modo a se encaixar neste texto). Como invariantes do ambiente de *design* estão:

ID*1*: tamanho e complexidade do problema;

ID*2*: objetivos como entrada e uma especificação do artefato como saída;

ID*3*: separação temporal entre as fases de *design* e de distribuição do artefato;

ID*4*: demora e limitação de *feedback* proveniente do mundo durante a evolução do processo de *design*;

ID*5*: funcionamento do artefato de modo independente do projetista;

ID*6*: custos associados a cada ação;

ID*7*: inexistência de definição de respostas certas/erradas mas apenas de respostas melhores/piores;

ID*8*: muitos graus de liberdade no estabelecimento do problema de *design*, seja por falta de informação ou por restrições impostas pelo mundo real.

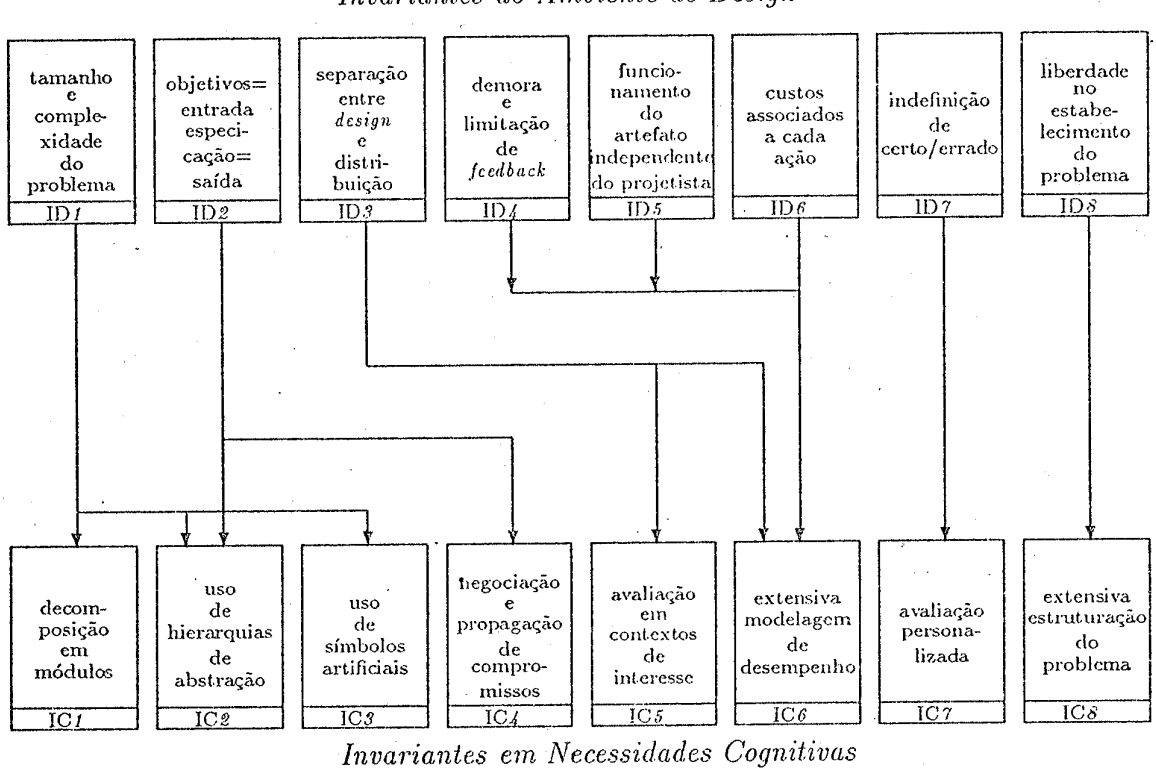

Figura 1 - "Gabarito" para modelos de *design*

Como invariantes em necessidades cognitivas de projetistas estão:

IC*1*: decomposição da solução em módulos relacionados;

IC*2*: uso de hierarquias de abstração (possibilitando o desenvolvimento do artefato e seus componentes em diferentes níveis de detalhe);

IC*3*: uso de sistemas de símbolos artificiais (possibilitando a focalização e a representação de informações relevantes em diferentes níveis de abstração);

IC*4*: negociação e propagação dinâmica de compromissos (objetivos, restrições e decisões);

IC*5*: avaliação contínua de componentes gerados ou focalizados em todos os contextos de interesse (local, atual e futuro);

IC*6*: extensiva modelagem de desempenho (permitindo que o projetista antecipe o desempenho do artefato e as consequências de sua distribuição no mercado);

IC7: funções de avaliação (e regras de finalização) personalizadas (ressaltando que uma avaliação do que é melhor ou pior depende de habilidades pessoais do projetista);

IC8: extensiva estruturação do problema (possibilitando aquisição e uso gradativo de informação para estabelecimento do problema).

Embora na Figura 1 estejam assinaladas apenas as relações do tipo "IDi requer ICj", observe-se que, por sua própria natureza, os invariantes em necessidades cognitivas são fortemente relacionados entre si.

Uma comparação dos invariantes identificados em ambientes de *design* com características consistentemente reconhecidas em ambientes de desenvolvimento de software [KOUB89] reforça que estes últimos sejam considerados ambientes de *design* . Consequentemente, de acordo com a discussão anterior, ambientes para desenvolvimento de software devem adotar (ou incorporar) modelos de *design* que visem a satisfação dos correspondentes invariantes em necessidades cognitivas de seus usuários. Em particular, isto é válido para o sistema MIDAS cujo modelo de ciclo de vida de software pode ser considerado um modelo de *design*, conforme discutido a seguir.

3 O modelo de ciclo de vida de MIDAS como modelo de *design*

MIDAS pretende ser um ambiente no qual um conjunto de ferramentas dará suporte a uma definição completa de requisitos da aplicação do usuário através do desenvolvimento de sua interface. O modelo de ciclo de vida de software orientado para prototipagem a ser adotado por este ambiente é mostrado na Figura 2. Ele é uma adaptação do modelo proposto em [BISC89] e ambos se encaixam na abordagem geral de expressar o processo de projeto/desenvolvimento de software através de uma série de pequenos subprocessos de prototipagem introduzida anteriormente em [BOEH86] com seu modelo espiral de desenvolvimento e melhoria de software. Em outras palavras, este modelo de ciclo de vida pretende possibilitar a evolução do artefato de *design* através de *loops* de síntese, análise e avaliação [BENT89].

Entidades, atributos das entidades e o relacionamento entre elas são três importantes componentes do artefato que precisam ser identificados durante a evolução do processo de *design* [BENT89], [VETH87]. Na terminologia de orientação a objetos [GOLD83], paradigma de programação a ser usado em MIDAS, estes três componentes podem ser associados a objetos, suas variáveis de classe e de instância, relacionamento hierárquico entre classes e mensagens entendidas pelos objetos.

Os principais resultados do desenvolvimento da interface de usuário, mostrados no interior do grande retângulo na Figura 2, são um protótipo da interface de usuário (eventualmente a interface final), a arquitetura do programa de aplicação e um componente protótipo da aplicação já "ligado" *(bound)* à interface. O desenvolvimento completo de uma interface de usuário é a base para a definição dos requisitos da aplicação no ambiente MIDAS.

As cinco primeiras fases do ciclo de vida proposto são altamente integradas. A primeira

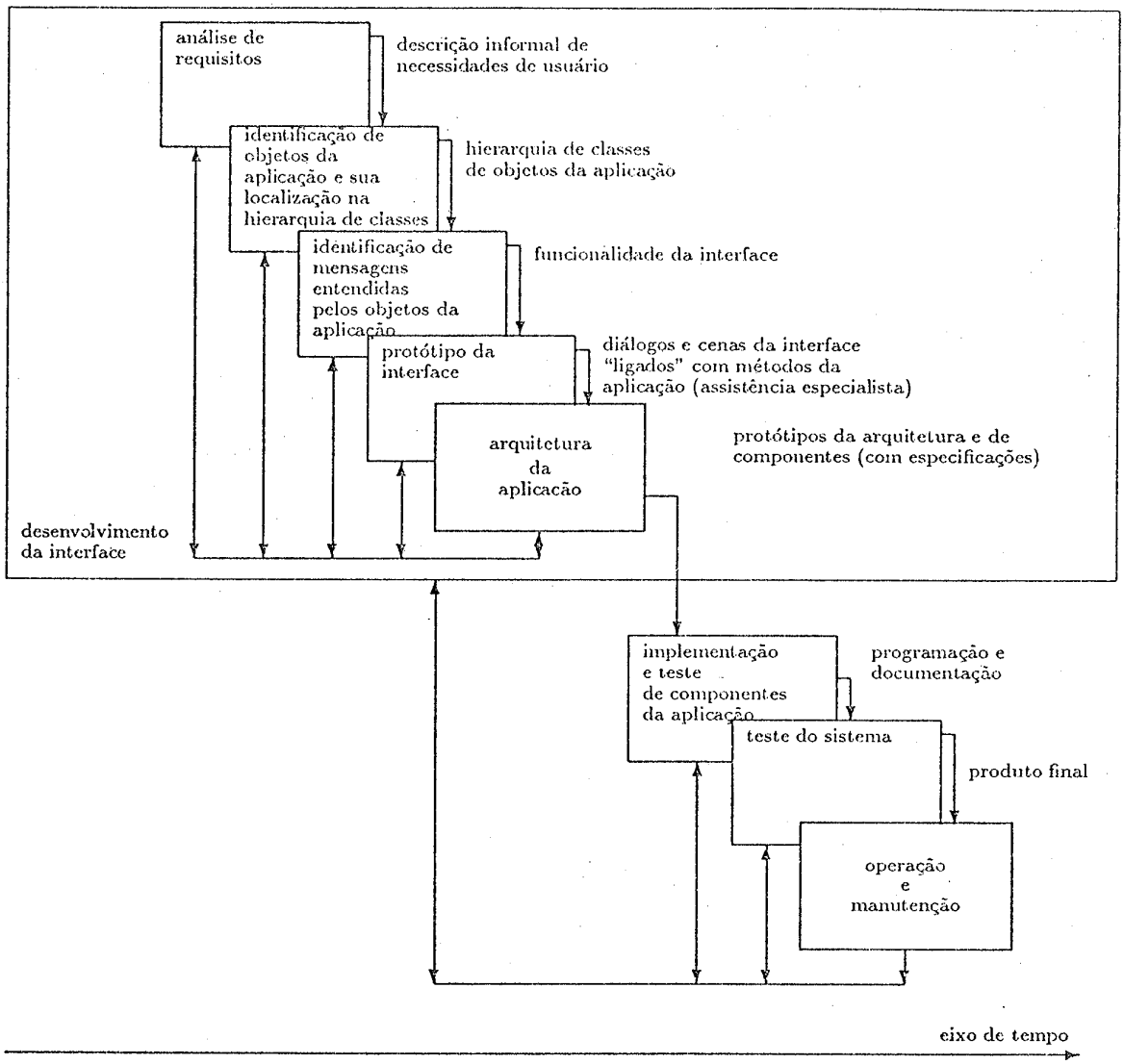

Figura 2 - Modelo de ciclo de vida de MIDAS

fase consiste em uma descrição informal das necessidades do usuário, embutida na análise de requisitos. A partir desta descrição, o projetista de interfaces começa a identificar os objetos da aplicação (segunda fase). Na terceira fase, os objetos identificados anteriormente são caracterizados em maior detalhe pela atribuição de funcionalidade, i.e., pela identificação das mensagens por eles entendidas, de suas variáveis de classe e de instância, e de quaisquer globais que a aplicação possa precisar. Na quarta fase, o projetista de interfaces gera um protótipo da interface pelo estabelecimento das "cenas" e do "*script*" do diálogo entre usuário e aplicação. A partir deste protótipo de interface o projetista é capaz de gerar um protótipo da arquitetura e uma especificação do sistema de aplicação (quinta fase).

MIDAS deverá incorporar em sua base de conhecimento um conjunto de regras que expressem este modelo de ciclo de vida e, com auxílio de uma estratégia de cooperação

e através de uma interface de usuário, dar suporte ao projetista de interfaces para seguir este modelo, implementando o comportamento de um especialista em processos de desenvolvimento de software.

A abordagem evolutiva por prototipagem e a organização em etapas adotada no ciclo de vida de MIDAS facilitam a decomposição do projeto/desenvolvimento de interfaces em conjunto com a especificação do aplicativo em módulos ou sub-problemas (IC*1*), e incentivam a utilização de hierarquias de níveis de abstração (IC*2*), cada um dos quais podendo ser expresso em um sistema simbólico (IC*3*) definido convenientemente.

Por outro lado, a evolução por prototipagem e a organização do modelo em etapas associadas à liberdade de movimentação entre as etapas possibilitam a satisfação simultânea dos invariantes IC*4* (negociação e propagação de compromissos), IC*5* (avaliação contínua de componentes em contextos de interesse), IC*6* (extensiva modelagem de desempenho), IC*7* (avaliação personalizada) e IC*8* (extensiva estruturação do problema).

A inexistência de uma definição de certo/errado (ID*7*), seja na escolha de estilo de interação ou na especificação de uma arquitetura do sistema aplicativo, motiva a utilização de funções de avaliação personalizadas ou, no máximo, aceitas como consenso (IC*7*). Assim, o projetista, preferivelmente em conjunto com o usuário do sistema em projeto, define o momento de parar o desenvolvimento da interface/especificação do sistema em projeto e de passar para as etapas de implementação e teste dos componentes específicos do aplicativo (sexta, sétima e oitava etapas do ciclo de vida de MIDAS).

Acreditamos que este modelo é computacionalmente viável através da integração das abordagens de:

- prototipagem [BISC89] - possibilitando evolução de um protótipo que eventualmente se tornará o próprio artefato;
- desenvolvimento orientado a objetos [GOLD83] - que por sua natureza e, em particular, por suas características de hereditariedade e polimorfismo, facilita o acréscimo gradual de classes de objetos à biblioteca de classes já existente, à medida em que o problema seja melhor estruturado e conforme sejam assumidos compromissos;
- estilo de interação com manipulação direta [SHNE83] - possibilitando que o usuário se aproxime cognitivamente da semântica da aplicação, devido tanto ao fato de o usuário manipular diretamente os objetos exibidos quanto à transparência na representação natural destes objetos e das operações sobre eles;
- sistemas multi-especialistas [DAVI87] - possibilitando a utilização de um conjunto de sistemas especialistas em domínios específicos que atuam cooperativamente entre si e com o projetista. Por exemplo, além de um sistema especialista em processos de desenvolvimento de software, MIDAS também deverá fornecer um sistema especialista em projeto de interfaces.
- processamento dirigido por eventos [SCHM86] - permitindo tanto a liberdade de caminhamento entre etapas na evolução do processo de *design* quanto a implementação de diálogos assíncronos;

Uma justificativa mais detalhada sobre a utilização destas abordagens pode ser encontrada em [CABR90].

4 Um exercício de uso do modelo

O projeto de MIDAS ainda está na fase inicial de análise de requisitos. Em outras palavras, MIDAS ainda não tem um protótipo que permita simular sua interação com um projetista. Não obstante, no que se segue, pretendemos ilustrar alguns resultados obtidos durante uma evolução inicial das quatro primeiras etapas do modelo de *design* a ser adotado por este ambiente. Para isto vamos supor que o usuário de MIDAS queira desenvolver um sistema para desenhar interativamente figuras semelhantes às apresentadas neste texto e gerar o código T_EX [KNUT84] correspondente para possibilitar sua inserção no texto. A escolha de um sistema com esta funcionalidade como exemplo se deve à sua simplicidade e natureza altamente interativa. Por outro lado, e exatamente por ser altamente interativo, sua especificação fica quase completa quando sua interface estiver desenvolvida. Entretanto, vale observar que não temos o objetivo de realmente projetar/desenvolver o sistema exemplo.

A primeira etapa do ciclo de vida de MIDAS corresponde a uma definição informal das necessidades do usuário do sistema a ser desenvolvido, embutida na análise de requisitos onde os objetos da aplicação começam a ser identificados. Esta definição informal pode ser expressa por um texto em linguagem natural editado para documentar os objetivos do sistema a ser desenvolvido, dando início à definição do caso central a partir do qual o processo de design começará. No exemplo em questão, o resultado de uma primeira execução desta etapa pode ser:

"O sistema a ser projetado/desenvolvido é uma ferramenta para desenhar interativamente figuras que aparecem em textos como este e gerar código fonte (por exemplo, em T_EX) para possibilitar a inclusão das figuras em textos. Estas figuras incluem objetos como pontos, poligonais, setas, retângulos, círculos, números e textos. Cada um destes objetos tem atributos, tais como tipo de linha usada no contorno do objeto, fonte e tamanho de caracteres, a serem definidos pelo usuário durante a utilização do sistema. O usuário do sistema desenha cada figura interativamente por manipulação direta, jamais se preocupando com o código correspondente."

Observe-se que esta descrição informal já possibilita a identificação de alguns objetos do aplicativo e de alguns de seus atributos (o que virtualmente ocorre na segunda etapa). Não obstante, a utilização da segunda etapa continua válida para que o projetista do sistema em desenvolvimento confirme os objetos identificados informalmente na etapa anterior e, possivelmente, identifique novos objetos no mundo da aplicação além de encaixá-los na hierarquia de classes da biblioteca disponível. O resultado desta etapa depende não só do entendimento do projetista sobre o problema focalizado mas também de seu conhecimento sobre as classes já existentes na biblioteca. Por exemplo, no aplicativo focalizado, o projetista poderia confirmar pontos, poligonais, setas, retângulos, círculos e caracteres alfa-numéricos como objetos e identificar elipses como mais um objeto. Vale observar que a inclusão deste novo objeto pode motivar o projetista a um retorno à etapa anterior para a modificação correspondente. Ao analisar a biblioteca de classes disponível, aqui assumida como a de Smalltalk-80 [GOLD83], talvez auxiliado por um sistema especialista, o projetista, por exemplo, os localizaria no trecho da hierarquia mostrado na Figura 3.

```
Object
   Point              (para pontos)
   Rectangle          (para retângulos)
   BitBlt
      CharacterScanner
      Pen             (para poligonais)
   DisplayObject
      DisplayMedium
         Form         (para setas e outros ícones)
            Cursor
            DisplayScreen
      Path
         Arc
            Circle    (para círculos e elipses)
         Line
```

Figura 3 - Localização de Objetos na Hierarquia de Classes

De maneira natural, o projetista começa a identificar detalhes característicos de cada um destes objetos e funções que devam desempenhar ou mensagens que devam entender (terceira etapa do ciclo de vida de MIDAS). Por exemplo, o projetista pode identificar a necessidade de cada objeto entender uma mensagem que defina sua posição em uma figura ou ainda que todos os objetos identificados anteriormente, com exceção de ponto e seta, precisam entender uma mensagem que defina seu tamanho. Ao mesmo tempo, como consequência da necessidade de definir a posição de cada objeto, é identificada a necessidade de um atributo que a armazene. Por exemplo, um ponto ser definido por duas coordenadas que o posicionam em uma figura e, para cada objeto pode ser especificado um ponto como referência para seu posicionamento (por exemplo, o canto inferior esquerdo para retângulos).

Seguindo o ciclo de vida, o projetista entra na quarta etapa onde ocorre a escolha do estilo de comando de interação com o usuário do sistema em projeto, ou seja, o estabelecimento das "cenas" e do "script" do diálogo do sistema em desenvolvimento. Aqui o projetista pode definir janelas e *menus* que componham o protótipo da interface. Por exemplo, o projetista pode escolher uma janela de trabalho onde o usuário do sistema em projeto "desenhará" figuras, um *menu* com ícones representantes dos objetos identificados nas fases anteriores e um *menu* que implemente a funcionalidade destes objetos. Neste ponto, MIDAS pretende prover auxílio especialista em técnicas de interação, em escolha de objetos da tela que melhor conduzam a funcionalidade desejada, em posicionamento de janelas, em superposição de cores, etc. O resultado desta etapa é um protótipo da interface de usuário do sistema em desenvolvimento "já ligado" aos métodos da aplicação identificados por seu relacionamento com a interface.

Neste ponto, o projetista pode passar à quinta etapa do ciclo de vida para projeto arquitetônico do aplicativo ou resolver testar o protótipo da interface. Suponhamos que ele resolva testar o protótipo. Durante o uso do protótipo, o projetista pode perceber a necessidade de alguns objetos manipulados entenderem mensagens para modificação

(ampliação e redução em uma ou mais de suas dimensões) de sua forma, ou ainda mensagens para apagá-los da figura focalizada. A partir da identificação destes requisitos, o projetista pode voltar a etapas anteriores e modificá-las adequadamente, para que o novo protótipo esteja mais próximo do artefato desejado. Lembrando que não temos o objetivo de realmente projetar/desenvolver o sistema exemplo interrompemos aqui este exercício.

Para finalizar esta seção vale observar que:

- MIDAS deverá incorporar em sua base de conhecimento regras para dar suporte ao projetista para seguir o modelo de *design* discutido.
- Atividades cognitivas de síntese, análise e avaliação [BENT89] são naturais em um processo de *design* e isto justifica a liberdade oferecida por MIDAS para o caminhamento entre as etapas de seu ciclo de vida e o relacionamento íntimo entre suas cinco primeiras etapas. Este processo e relacionamento entre etapas pode ser notado muitas vezes pela interseção entre resultados obtidos em uma etapa e que seriam objetivos a atingir na etapa seguinte.
- Na quarta etapa deve ser ressaltado o valor do auxílio que MIDAS pretende oferecer a seu usuário, pois em sua biblioteca de classes para construção de interfaces de usuário deverão estar incluídos objetos comando e objetos visão (*command objects* e *view objects*, na terminologia de *MacApp*) [SCHM86] que farão tudo relacionado à mecânica de diálogos de interfaces (como detetar *clicks* no *mouse*, exibir *menus* e capturar opções do usuário, arrastar imagens, etc.), permitindo que o projetista de interfaces se dedique à tarefa de prover a semântica da aplicação pela implementação do comportamento de todos os objetos da aplicação.
- Nem todo trecho de código da aplicação é relacionado à interface, ainda que possa agora ser suficientemente claro quão entrelaçadas são a escolha dos objetos no mundo da aplicação, a resultante funcionalidade da interface, e a estrutura do código da aplicação. A troca de mensagens entre objetos da aplicação deverá ser território totalmente livre, deixando à disposição do projetista amplo espaço para criatividade arquitetônica. Para aqueles objetos da aplicação que recebem mensagens originárias de uma interação da interface, MIDAS terá todas as informações necessárias para caracterizar aquele objeto (classe) da interface, e um módulo provisório (*stub*) poderá imediatamente ser gerado para efeitos de simulação. É nesta fase que a estrutura interna de código puramente aplicativo é deixada para o projetista de sistemas.
- A ligação entre o código de interface de usuário e o código de aplicação é implementável usando mecanismos de processamento dirigido por eventos em conjunto com programação orientada a objetos, conforme discutido em [CABR90].
- MIDAS pretende prover uma notação de especificação executável [HOFF88] que deverá permitir ao projetista de sistemas completar a caracterização de todos os objetos da aplicação e simular o comportamento do produto final. Como o projetista de interfaces terá à disposição as especificações dos objetos da aplicação então existentes (na biblioteca de classes), MIDAS deverá fornecer métodos para gerência de configurações e validação de programas [ALEN90].

5 Análise crítica e conclusões

Design é aceito como um processo evolutivo a partir de um caso central (uma solução inicial para o artefato ou objeto de *design*) em direção a uma especificação do artefato desejado [BENT89], [BOEH86], [GOEL89], [NADI87], [TAKA87], [VETH87]. A solução inicial é baseada no conhecimento do projetista sobre tipos de soluções, tecnologias disponíveis e relações de dependência entre eles. Em outras palavras, o processo de *design* depende da experiência do projetista em *design*, do seu conhecimento em relação à natureza do objeto de *design*, de seu desempenho em atividades cognitivas como indução, dedução, criação e intuição, e das ferramentas de que dispõe.

A utilização de sistemas especialistas, por exemplo, em estratégias de *design* e de resolução de problemas relacionados ao mundo do artefato focalizado, é consensualmente aceita como essencial para auxiliar projetistas nas atividades de indução e dedução [BENT89], [DAVI87], [KOEG87], [NADI87], [TAKA87], [VETH87]. Técnicas de inteligência artificial tais como representação de fatos através de regras armazenadas em bases de conhecimento, geração de hipóteses a serem testadas e, se confirmadas, incorporadas à base como fatos, pesquisas nestas bases através de encadeamento para trás (*backward*) e para frente (*forward*) para verificação de satisfação de requisitos, já estão sendo consideradas básicas em ferramentas de auxílio em raciocínio por indução e dedução.

Por outro lado, as atividades de criação e intuição, por sua própria natureza, continuam sob responsabilidade do projetista [BENT89]. Não obstante, é ressaltada a importância da consideração das demandas cognitivas de projetistas na construção de modelos, metodologias e ferramentas de *design* que os auxiliem inclusive nestas duas atividades [TAKA87]. Neste caso, pode-se considerar consenso a tendência em oferecer meios para que o projetista se preocupe o mínimo possível com as tarefas de manter, atualizar, organizar e recuperar dados e requisitos, mantendo a consistência entre eles, e tenha liberdade e facilidades para criar e intuir. [TAKA87] sugere que a base de conhecimento de um sistema especialista seja composta de "fatos verdadeiros" (verdades e adivinhações confirmadas) e "adivinhações" (modificações randômicas em fatos, sujeitas a confirmações para serem assumidas como fatos verdadeiros) de modo a auxiliar a atividade de criação (na verdade, ele faz esta sugestão como uma forma de possibilitar criatividade artificial). É importante ressaltar que estas alterações randômicas devem ser controladas por um método de filtragem, de modo que apenas uma parte restrita de conhecimento seja alterada para gerar alguma coisa nova, não permitindo resultados sem sentido.

Todas estas atividades têm natureza dinâmica e envolvem aquisição, reformulação, ou mesmo, rejeição de informações e decisões anteriores, implicando a necessidade de retorno a etapas para as alterações pertinentes.

Assim, modelos (ou ferramentas) de *design* devem incorporar estratégias que visem a:

- oferecer ao projetista linguagens de representação do artefato adequadas a cada etapa do processo de *design* [NADI87], [WOOD87], conforme o nível de abstração ou a quantidade de informação disponível, possibilitando desde o estabelecimento do contexto de *design* até a obtenção do artefato, passando pela pesquisa por subsoluções. Tais linguagens devem facilitar a obtenção gradual de informação sobre as

entidades, atributos das entidades e relacionamentos entre elas, considerados elementos básicos do objeto a ser projetado [BENT89], [VETH87]. Além disto, [NADI87] ressalta que *design* é uma atividade dominantemente visual, o que motiva a definição de representações visuais como, por exemplo, através de ícones e diagramas.

- possibilitar a organização do processo de *design* de modo que as atividades de análise (reconhecimento de propriedades, requisitos e restrições relevantes no objeto de *design*) e síntese (resolução de problema para encontrar uma solução para as especificações e restrições identificadas na análise), naturais e necessárias em *design*, possam ocorrer de forma adequada a cada uma delas. Este aspecto precisa ser considerado cuidadosamente pois, virtualmente, como argumentado em [AKIN79], [DARK79] e [LAWS79], as atividades de análise e síntese são dinâmicas e fazem parte de todas as etapas no processo de *design*, i.e., geralmente, não há necessidade da análise do problema estar completa para o projetista iniciar a atividade de síntese. Dirigindo estas duas atividades, ocorre naturalmente a atividade de avaliação que procura verificar a consistência entre os elementos obtidos nas atividades de análise e síntese. Assim, é interessante que o modelo de *design* como um todo e cada uma de suas etapas possibilitem a ocorrência de vários *loops* de análise, síntese e avaliação [BENT89], conforme o projetista deseje.
- possibilitar o gerenciamento da evolução do processo garantindo a cooperação entre os elementos envolvidos (sistemas especialistas e projetista) [BENT89], [NADI87] e mantendo coerência nos aspectos de dependência, correspondência, transformação, acessabilidade conceitual e organizacional entre as etapas [DAVI87].
- oferecer facilidades para documentação da evolução do processo de *design*, de modo a permitir a explicação de decisões de projeto e a volta a etapas anteriores para acréscimo, transformação ou eliminação de objetivos e restrições, e consequente transformação de decisões [NADI87], com garantias de manutenção de coerência entre estes elementos [DAVI87]. Neste sentido, em cada etapa, objetivos e restrições podem ser usados para descrever de modo preciso a natureza da informação que aparece nas especificações do objeto em projeto.

O ciclo de vida de *software* a ser adotado por MIDAS se bascia na abordagem geral de evolução de protótipos na qual o protótipo eventualmente se tornará o artefato de *design*. Acreditamos que a integração das abordagens de sistemas multi-especialistas, programação orientada a objetos, interação com manipulação direta e processamento dirigido por eventos possibilitem a definição de estratégias úteis para o atendimento dos demais aspectos discutidos acima e será explorada na continuidade do desenvolvimento de MIDAS.

Referências

[AKIN79] Akin, O.; "An Exploration of the Design Process", *Design Methods and Theories*, Vol. 13, N. 3/4, 1979, 115-119

[ALEN90] Alencar, P.; Lucena, C.; "A Logical Approach for Evolving Software Systems", a aparecer

[ALEX64] Alexander, C.; Notes on the Synthesis of Form, Harvard University Press, Cambridge, Mass., U.S.A., 1964

[ARCH65] Archer, L. B.; "Systematic Method for Designers", The Design Council, London, U. K., 1965

[BENT89] Bento, J.; Feijó, B.; Dowling, P. J.; "Knowledge based design of steel portal frames for agricultural buildings", 1989, a aparecer

[BISC89] Bischofberger, W.; Keller, R.; "Enhancing the Software Life Cycle by Prototyping", *Structured Programming* 1:47-59, 1989

[BOEH86] Boehm, B.; "A Spiral Model of Software Development and Enhancement", *ACM Software Engineering Notes* Vol.11, No.4, August 1986

[CABR90] Cabral, R. H. B.; Campos, I. M.; Cowan, D. D.; Lucena, C. J. P.; "Interfaces as Specifications in the MIDAS User Interface Development System", a aparecer.

[CROS89] Cross, N.; "Understanding Design: The Lessons of Design Methodology", *Design Methods and Theories*, 13, 13 4, 1989??

[DARK79] Darke, J.; "The Primary Generator and the Design Process", *Design Studies*, V.1, No.1, 36-44, 1979

[DAVI87] David, B. T.; "Multi-Expert Systems for CAD", *Proceedings* 1^{st} *EUROGRAPHICS Workshop on ICAD Systems - Theoretical and Methodological Aspects*, The Netherlands, April 21-24, 1987

[DERE75] DeRemer, F.; Kron, H.H.; "Programming-in-the-Large versus Programming-in-the-Small", *IEEE Transactions on Software Engineering* V. SE-2, No.2, June 1976

[GOEL89] Goel, V.; Pirolli, P.; "Design within Information-Processing Theory: The Design Problem Space", *AI Magazine*, Spring 1989, 19-36.

[GOLD83] Goldberg, A.; Robson, D.; Smalltalk-80: the language and its implementation. Xerox Corporation, 1983.

[HART89] Hartson, H. R.; Hix, D.; "Human-Computer Interface Development: Concepts and Systems for its Management", *ACM Computing Surveys*, V.21, No. 1, 1989.

[HILL72] Hillier, B.; Musgrove, J.; O'Sullivan, O.; "Knowledge and Design" in Mitchell, W. J. (ed), Environmental Design: Research and Practice, University of California, Los Angeles, U.S.A., 1972

[HOFF88] Hoffman, D.; "Practical Interface Specification", *Software - Practice and Experience* Vol.19(2), February 1989

[JONE63] Jones, J. C.; "A Method of Systematic Design" in Jones, J. C. and Thornley, D. (Eds), Conference on Design Methods, Pergamon Press, Oxford, U. K., 1963

[KNUT84] Knuth, D. E.; The *TEXbook*. Addison-Wesley, Reading, Massachusetts, 1984.

[KOEG87] Koegel, J. F.; "A Theoretical Model for Intelligent CAD", *Proceedings 1st EUROGRAPHICS Workshop on ICAD Systems - Theoretical and Methodological Aspects*, The Netherlands, April 21-24, 1987

[KOUB89] Koubek, R. J.; Salvendy, G.; Dunsmore, H. E.; LeBold, W. K.; "Cognitive issues in the process of software development: review and reappraisal", *International Journal of Man-Machine Studies*, 30, 1989, 171-191

[LAWS79] Lawson, B.; "Cognitive Strategies in Architectural Design", *Ergonomics*, V.22, No.1, 59-68, 1979

[NADI87] Nadin, M.; Novak, M.; "MIND: A Design Machine -Conceptual Framework-", *Proceedings 1st EUROGRAPHICS Workshop on ICAD Systems - Theoretical and Methodological Aspects*, The Netherlands, April 21-24, 1987

[RITT73] Rittel, H.; Webber, M.; "Dilemmas in a General Theory of Planning", *Policy Sciences*, V.4, 181-200, 1973.

[SCHM86] Schmucker, K.J.; Object-Oriented Programming for the Macintosh, Productivity Products International, Inc., 1986

[SHNE83] Shneiderman, B.; "Direct Manipulation: a Step Beyond Programming Languages", *IEEE Computer* 16, August 1983, 57-69

[TAKA87] Takala, T.; "Intelligence beyond Expert Systems: A Physiological Model with Applications in Design", *Proceedings 1st EUROGRAPHICS Workshop on ICAD Systems - Theoretical and Methodological Aspects*, The Netherlands, April 21-24, 1987

[THOM79] Thomas, J.C.; Carrol, J. M.; "The Psychological Study of Design", *Design Studies*, V.1, No.1, 5-11, 1979

[VETH87] Veth, B.; "An Integrated Data Description Language for Coding Design Knowledge", *Proceedings 1st EUROGRAPHICS Workshop on ICAD Systems - Theoretical and Methodological Aspects*, The Netherlands, April 21-24, 1987

[WOOD87] Woods, D. D.; "Commentary: Cognitive engineering in complex and dynamic worlds", *International Journal of Man-Machine Studies*, 27, 1987, 571-585.