Uma Proposta para Manipulação de Objetos em Modeladores de Sólidos

BRUNO FEIJÓ[1]
NICK LEHTOLA[1]
JOÃO BENTO[2]
SEBASTIÃO A.L. de ANDRADE[1]

[1]ICAD - Laboratório de CAD Inteligente, PUC-Rio
Rua Marquês de São Vicente, 225
22453-900 Rio de Janeiro, RJ, Brasil
bruno@icad.puc-rio.br

[2]CMEST, Instituto Superior Técnico, Universidade Técnica de Lisboa
Av. Rovisco Pais, 1096 Lisboa Codex, Portugal

Abstract. This paper proposes a conceptual object-oriented model for solid modellers that is embedded in the design problem space and is aligned with the new directions towards product modelling. This model renders the integrated manipulation of objects more feasible to apply in practice.

Introdução

Na década de 80, a pesquisa em modelagem de sólidos tinha o seu foco nos problemas de geometria e de topologia de objetos isolados. Atualmente a atenção volta-se para o desenvolvimento de modelos completos de engenharia relativos ao produto. Esta nova direção foca as questões de modelagem de produto (*product modelling*), modelagem baseada em características (*feature-based modelling*), modelagem de tolerância (*tolerance modelling*), modelagem de restrições (*constraint modelling*), geometria variacional (*variational geometry*), métodos paramétricos (*parametric methods*) e raciocínio geométrico (*geometric reasoning*) (Wozny et alii, 1990). A definição do termo **Dados de Produto** (*Product Data*), cunhado pelos desenvolvedores do padrão emergente STEP, ajuda a esclarecer o tipo de informação que os modeladores de sólidos devem considerar atualmente: "Dados de produto são os dados requeridos para definir as características funcionais, físicas e de programação de um produto através de seu ciclo de vida" (Wilson e Kennicott, 1988) (Perkins, 1990).

Na prática, os novos modeladores de sólidos têm implementado características de modelagem de produto através do paradigma de orientação a objetos associado a técnicas de representação de conhecimento (Haase, 1992). Este tipo de abordagem é algumas vezes denominada "modelagem de produto orientada a objetos" (*object-oriented product modelling*) ou, ainda, "modelagem de produto baseada em conhecimento" (*knowledge-based product modelling*). Neste tipo de modelagem, propõem-se arquiteturas de objetos cujos relacionamentos entre si permitam a manipulação inteligente dos sólidos, tanto do ponto de vista das operações de modelagem geométrica quanto da formulação de características funcionais e físicas do produto. Esta manipulação vai desde um arrasto (*drag*) com adaptações automáticas de outras partes do artefato até alterações de funcionalidade do produto. Entretanto, a modelagem de produto orientada a objetos continua carente com relação aos seguintes tópicos: pesquisa de *design* (*design research*) e integração objetos/lógica ou objetos/λ-calculus. De fato, o processo de *design*, apesar de estar no cerne da questão de modelagem, nunca é formal e rigorosamente considerado. Por outro lado, as formas de integração de objetos com outros paradigmas são limitadas e inflexíveis.

Dentro do contexto dos processos que criam um produto, os modeladores de sólidos em sistemas de CAD devem estar imersos no espaço de problema de *design* (*DPS-Design Problem Space*). Esta imersão requer primeiro a consideração das questões de cognição e de percepção 3D. Estas questões já foram apresentadas em Feijó e Fischer (1992), através do modelador de sólidos híbrido GeneSys (Figura 1). GeneSys trata de operações locais e globais inseridas no espaço de problema de *design* (*DPS*) sem, contudo, considerar relacionamentos genéricos entre os sólidos, o que permitiria a manipulação inteligente dos mesmos.

Dentro desta linha iniciada por Feijó e Fischer (1992), o presente trabalho propõe uma arquitetura de

objetos para a manipulação inteligente dos mesmos. Esta proposta está alinhada com o rigor da pesquisa de *design* e com as necessidades de uma melhor integração de objetos com lógica.

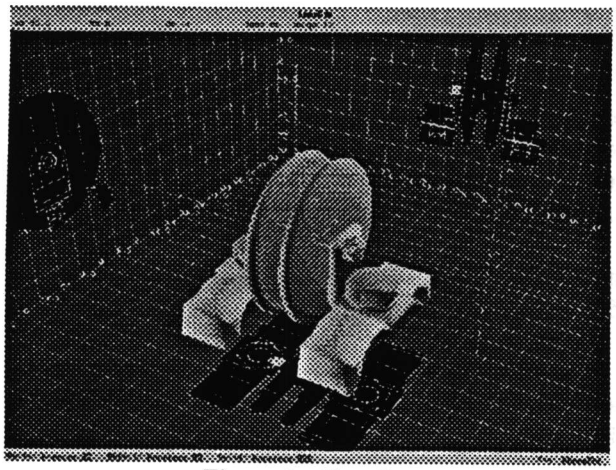

Figura 1 - GeneSys

Espaço de Problema de *Design*

Design foi primeiro identificado com Resolução de Problemas (*Problem Solving*) em Simon (1969). De acordo com sua proposta, um espaço de estados representa todos os possíveis estados do problema (i.e. todas as possíveis descrições do problema) que precisam ser considerados quando uma solução é tentada.

No paradigma de Resolução de Problemas, *design* é um processo de busca adaptativa motivado por metas.

No modelo *SAE* (Feijó e Bento, 1991) (Bento, 1992) (Scheer, 1993) (Prates, 1993) os objetivos são representados por entidades de *design* que, por estarem dentro do paradigma de Resolução de Problemas, capturam ambas as noções de **forma** (i.e. essencialmente atributos físicos) e de **função** (i.e. especificação da função a ser realizada pela forma).

A definição de entidade de *design* é a seguinte:

Def. Entidade de *design* $\mathbf{e_j}$ é um terno $\mathbf{e_j(I,F,f)}$ onde **I**=atributos de identificação, **F**=atributos de forma e **f**=atributos de função.

Uma entidade de *design* $\mathbf{e_j}$ pode representar um objetivo, um sub-objetivo ou uma solução parcial.

Um estado é definido como se segue:

Def. Um estado $\mathbf{T_i}$ é um conjunto de entidades de *design* $\mathbf{e_j}$.

Neste contexto, *design* é um processo evolucionário que começa com um conjunto de especificações de entrada, $\mathbf{T_0}$, gera uma idéia núcleo (*kernel idea*) e a refina (por decomposição, geração ou transformação) com o intuito de chegar à descrição final do artefato $\mathbf{T_n}$, i.e.: $\mathbf{T_0 \rightarrow T_1 \rightarrow ... \rightarrow T_n}$. Os conceitos de história e de espaço são apresentados a seguir:

Def. Uma história de *design* **H** é uma seqüência de estados $\mathbf{T_0 \rightarrow T_1 \rightarrow ... \rightarrow T_n}$.

Def. O espaço de problema de *design* (**DPS**) é o conjunto de todas as histórias de *design*.

No GeneSys (Feijó e Fischer, 1992), o esquema de representação CGS modificada representa a história de *design* dentro de um DPS. Os nós da árvore CSG são entidades de *design* com atributos físicos, geométricos e de relações semânticas. Na implementação do GeneSys tomou-se a decisão de limitar o nível de aninhamento de relações *part-of* até o terceiro nível (*World*, *Object* and *Solid*). A questão do nível de aninhamento continua, entretanto, uma questão em aberto, visto que um excesso de níveis pode se tornar uma fonte de complexidade ao invés de modularidade e clareza.

Integração Objetos/Lógica

A integração de objetos com lógica é usualmente realizada de duas formas, ambas incompletas: (1) como sistemas de produção orientados a objetos; (2) como adaptações de Prolog. Na primeira forma, tem-se uma combinação simplista de objetos com regras de produção, sem a consideração de lógica matemática, quer como formalismo de representação quer como paradigma de programação. Algumas ferramentas típicas desta linha são as seguintes: ART (Inference Co, 1985), NEXPERT Object (Neuron Data, 1990), KEE (Fikes and Kehler, 1985) (Intellicorp, 1986) e STRATA (GEC Research, 1987). As adaptações de Prolog (Shapiro and Takeuchi, 1983) (Zaniolo, 1984) (McCabe, 1989) (Watson and Chan, 1991) (Koegel, 1987), que têm o compromisso de manter o paradigma de lógica como linguagem de programação, impõem várias restrições ao uso de objetos e continuam limitadas à lógica de Horn.

Para o tratamento híbrido de lógica e orientação a objetos, este trabalho propõe o uso do ambiente de programação XLOG+. Este ambiente foi desenvolvido por Bento (1992) a partir dos trabalhos de Feijó (1989).

XLOG+ não é uma adaptação de Prolog e considera (em contraste com Prolog) a lógica não-Horn. XLOG+ é totalmente aberto e é inserido em qualquer sistema como uma função em C.

Herança de propriedades, em XLOG+, é automaticamente estabelecida por ocasião da criação de um objeto (i.e. não precisa ser declarada explicitamente). Em XLOG+, a herança é cancelada (*inheritance overriding*) pela inclusão local de um

específico método ou atributo (nulo ou não). Ao contrário da orientação a objetos convencional, a herança em XLOG+ pode ser dinâmica.

Mensagens são passadas a objetos em XLOG+ através de perguntas (*queries*) diretas ou de deduções lógicas. Métodos em XLOG+ são, essencialmente, predicados lógicos agregados.

Finalmente, XLOG+ considera objetos mutáveis, no sentido de que suas propriedades podem mudar ao longo do tempo, e objetos compostos.

A apresentação do XLOG+ não é o objetivo deste trabalho. Apenas as suas possibilidades são relevantes como garantia para a arquitetura de objetos proposta.

Modelo de Objetos

O modelo proposto para a manipulação dos sólidos baseia-se nos trabalhos de Bento (1992), Scheer (1993) e Prates (1993). A proposta é considerar as entidades de *design* $e_j(I,F,f)$ como objetos que apresentam a seguinte estrutura:

I = atributos de identificação
- nome da entidade
- descrição da entidade
- estado da entidade

F = atributos de forma = {r,s}

r = atributos de relação
- parentesco
- alternativas
- versões
- decomposição
- trans-of
- term-of
- de ligação

s = atributos de estrutura
- atributos físicos e geométricos
- atributos de comportamento
- atributos de descrição

f = atributos de função
- propósito
- especificações funcionais
- especificações de desempenho

A evolução dos estados de *design* tendem a criar objetos que têm apenas atributos de forma e que estão na forma de sub-árvores CSG modificadas (Feijó e Fischer, 1992). Uma outra tendência nesta evolução, é a transformação de especificações funcionais (p.ex. "temperatura agradável") para especificações de desempenho (p.ex. $18 \leq$ temperatura ≤ 25). O conceito de desempenho é o elo entre função e forma e, também, o motor da dinâmica dos ciclos de Síntese-Análise-Avaliação do Modelo SAE.

Estes atributos podem ser **primitivos** (apresentam valores booleanos, inteiros, reais, cadeias de caracteres e ponteiros) ou **compostos** (lista, vetores, matrizes, estruturas de dados, etc ...). Os atributos também apresentam **predicados** definidos como procedimentos agregados (*attached procedures*) os quais são utilizados para calcular, deduzir ou verificar os valores dos atributos correspondentes. Apenas dois tipos de procedimentos são necessários: **if_needed** e **if_changed**.

Os procedimentos agregados são usados como disparos em acesso de variável, inspirados no estilo de valores ativos e técnicas de programação orientada a acesso (Stefik et alii, 1986) (Inference Corporation, 1985). Os procedimentos **if-needed** são avaliados antes da tentativa de recuperar o valor de um atributo Os procedimentos **if-changed** são definidos como objetivos a serem cumpridos no caso de um valor de atributo estar sofrendo a tentativa de ser trocado. Este último procedimento espalha as conseqüências de uma mudança de valor e também funciona como zelador do atributo (i.e. não permite mudanças que violem a integridade do produto).Os procedimentos agregados ou são predicados em lógica de primeira ordem ou são funções em C.

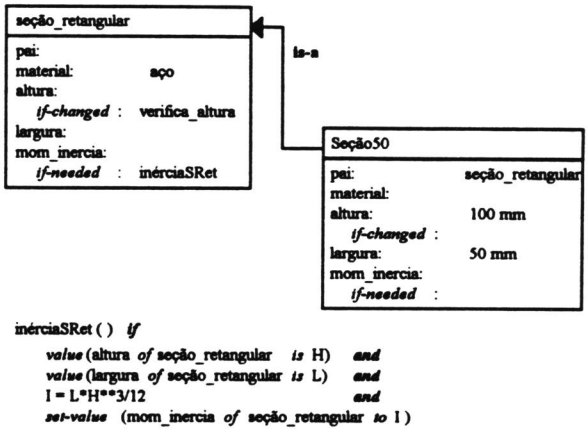

Figura 2 - Exemplo de Objetos e Predicados

A Figura 2 apresenta um objeto **Seção50** que herda propriedades do seu pai **seção_retangular** através de uma relação de parentesco (**is-a**). Neste exemplo, o procedimento **inerciaSRet** está especificado em termos de lógica (o predicado ***value*** coloca na variável **H** o valor do atributo altura do objeto **seção_retangular**).

As entidades e_j, na evolução do processo de *design*, podem ser criadas ou destruídas. Nestas ações as relações entre entidades quase sempre são mobilizadas.

Estas relações entre entidades (ou atributos de relação) são facilmente identificados com auxílios de **grafos de *design*** [Prates (1993)]. Estes grafos permitem, para cada estado de *design*, representar adequadamente a complexa relação entre as diversas entidades de design envolvidas.

As relações de parentesco (**is-a**) caracterizam a herança entre entidades, a qual pode ser simples (um ancestral) ou múltipla (vários ancestrais). Por exemplo, a viga VS250x56.8 herda propriedades de três ancestrais: VIGA, SEÇÃO Tipo I e material AÇO.

As relações de alternativas (**or**) são utilizadas para representar possibilidades de opções em estados de *design*. As relações de versões (**ex**) são utilizadas para manter informações sobre versões anteriores, ou seja, manter informações sobre etapas a que se deseja retornar (Figura 3). As relações de decomposição (**part-of**) caracterizam a decomposição de uma entidade em entidades componentes (por ex. VIGA V1 e COLUNA C1 são **part-of** do objeto PRÉDIO). As relações de ligação (**linked-to**) definem uma relação de causa e efeito para as entidades associadas (Figura 4). As relações **trans-of** e **term-of** representam parte da semântica da Árvore CSG Modificada proposta pelo GeneSys (Feijó e Fischer, 1992).

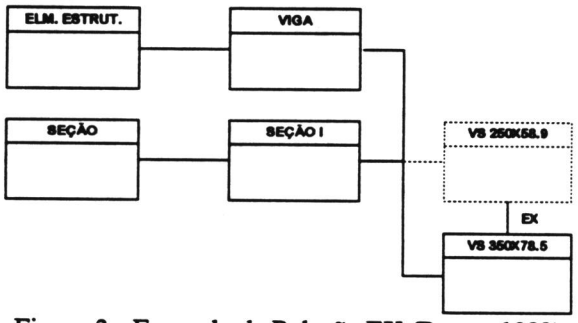

Figura 3 - Exemplo de Relação **EX** (Prates, 1993)

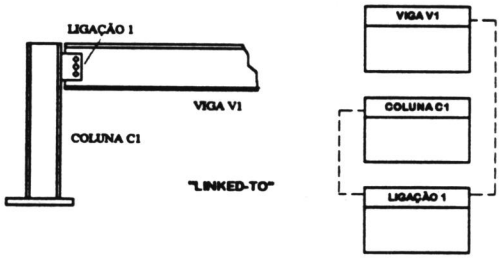

Figura 4 - Exemplo de Relação de Ligação

Manipulação Integrada de Objetos

O problema da manipulação está associado às relações de ligação (**linked-to**). O exemplo a seguir consite em se avaliar a consistência das relações de ligação. Para isso será tomado como modelo uma estrutura metálica composta de uma viga, duas colunas e elementos de ligação (placas de base e chapas de ligação), conforme ilustra a Figura 5. Os sólidos são mostrados em 2D para simplificar a explicação.

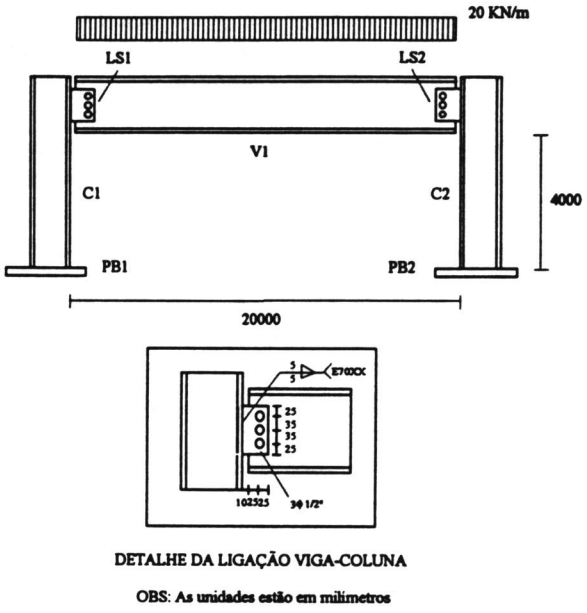

Figura 5 - Modelo de Estrutura Metálica

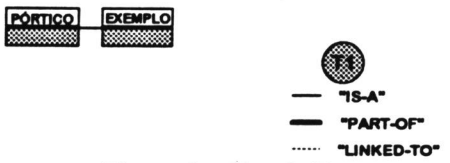

Figura 6a - Estado T_1

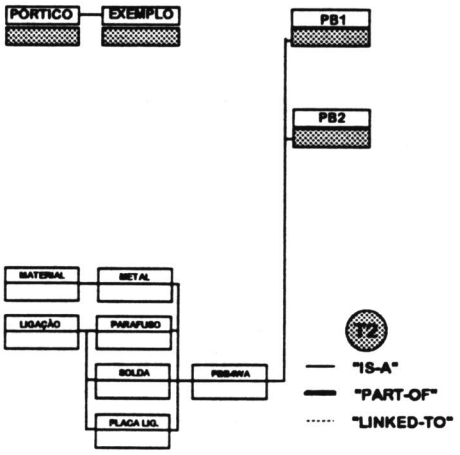

Figura 6b - Estado T_2

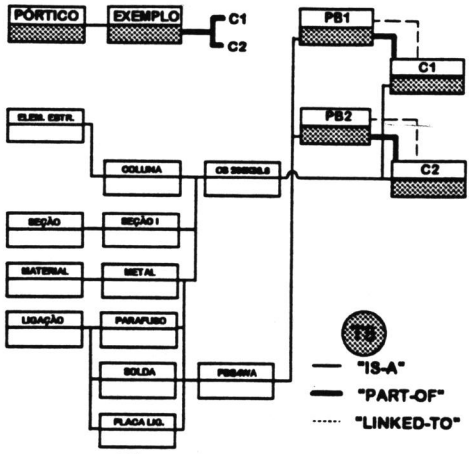

Figura 6c - Estado T_3

O primeiro passo nesta explanação é avaliar a evolução do processso de design, até à configuração final. O grafo de design associado a esta evolução (T_1 a T_4) está mostrado nas Figuras 6a, 6b, 6c e 6d. A montagem virtual da estrutura metálica no modelador deve ocorrer de maneira similar à situação real (i.e. primeiros as placas de base, depois as colunas e, finalmente, a viga com as chapas de ligação).

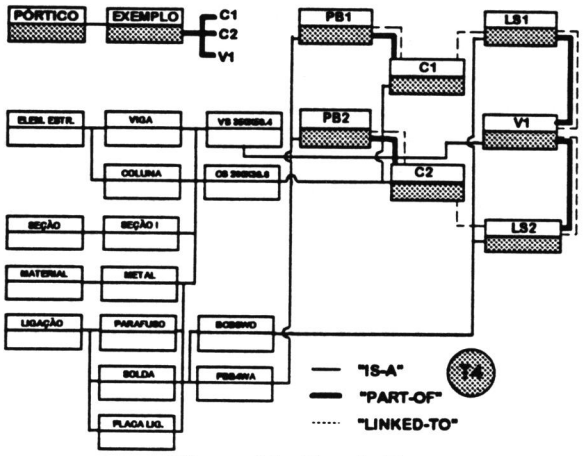

Figura 6d - Estado T_4

As entidades de *design* e seus principais atributos, no estado final, estão resumidos na Tabela 1.

PB1 e PB2
pai: PBB4WA;
material: AÇO MR-250;
dimensão: 250x250x10.

C1
pai: CS 200x38.8 (i.e. coluna soldada de seção I);
material: AÇO MR-250;
comprimento: 4350 mm.
linked-to: **[PB1, LS1]** (no caso de **C2**: **[PB2, LS2]**);

LS1 e LS2
pai: BCBSWD (ligação simples para viga-coluna, formada por PARAFUSOS, SOLDAS e PLACA DE LIGAÇÃO);
material placa: AÇO MR-250;
material solda: METAL E70XX;
material parafusos: AÇO ASTM A325;
tipo solda: dois filetes com 120 mm de comprimento e perna de 5 mm;
parafusos: 3 parafusos de 1/2";
dimensão da placa: 60x120 mm;
linked-to: **[C1, V1]** (no caso de **LS2**: **[C2, V1]**);

V1
pai: VS 350x50.4 (viga soldada de seção I);
material: AÇO MR-250;
comprimento: 8000 mm;
linked-to: **[LS1, LS2]**;

EXEMPLO
sistema formado por **V1, C1, C2, LS1, LS2, PB1** e **PB2**

Tabela 1 - Principais atributos (Ex. da Figura 9)

No exemplo acima, o estado final foi alcançado a partir de uma especificação inicial, recomendações normativas, um banco de dados de consulta e um protótipo implementado de forma bem específica visando apenas verificar as idéias propostas. A especificação inicial é o próprio problema a ser resolvido, isto é: projetar um pórtico com vão de 20 m, com pé-direito mínimo de 4 m e suportando uma carga acidental de 20 KN/m (exemplo hipotético). As recomendações normativas representam as cláusulas de norma, no caso a **NBR-8800** [NBR-8800 (1986)], utilizadas para a avaliação de cargas, dimensionamento de peças, etc. O banco de dados de consulta, no caso, representa um conjunto de entidades já especificadas e disponíveis no DPS, tais como o conjunto de materiais metálicos, as seções tipo I, etc.

O protótipo foi implementado utilizando-se a linguagem C++. A sua finalidade principal foi a de verificar os atributos de relação entre as entidades de *design*, principalmente as de relação de ligação. A linguagem C++ foi adotada, principalmente por suas características afins com o modelo proposto (herança, decomposição, etc.). Neste protótipo, os procedimentos agregados (**if-needed** e **if-changed**) foram implementados com o objetivo de simular, de uma forma simples, o ambiente que poderia ser proporcionado com ambientes híbridos baseados em lógica e orientação a objetos, tais como XLOG+ (Bento, 1992).

Com o intuito de se verificar as conseqüências de uma manipulação integrada através das relações de ligação (**linked-to**), promoveu-se, a partir do estado de *design* corrente T_4, um deslocamento de 1 m para a esquerda da coluna **C1**. Com isto, a partir dos processos agregados associados e das relações de ligação, uma nova evolução no DPS ocorreu à procura de um novo estado final, movido pelas novas condições, pelos propósitos inicialmente impostos ao processo e pelas restrições associadas (Figura 7).

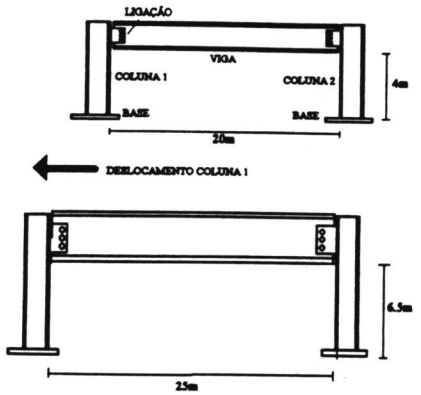

Figura 7 - Conseqüências de uma Manipulação

Com o afastamento da coluna **C1**, as ligações também se deslocaram. A viga **V1** teve o seu vão alterado e, conseqüentemente, a resultante de forças atuantes (carga distribuída) aumentou. Com isto, os elementos estruturais, que a princípio não suportaram a nova condição de carregamento, tiveram que melhorar as suas capacidades de suporte. A alteração das propriedades geométricas foi o caminho escolhido para se obeter esta melhora. A entidade **PB2** não sofreu nenhum tipo de alteração. As entidades **PB1**, **LS1** e **LS2** tiveram suas localizações no espaço alteradas. A viga **V1** passou a ser um VS 400x57.8, com uma seção transversal mais robusta. Com a alteração de **V1**, as entidades **C1** e **C2**, passaram a ter 4400 mm de comprimento.

Comentários Finais

A arquitetura de objetos proposta neste trabalho é consistente com o processo de *design*, com a árvore CSG modificada do GeneSys e com os sistemas baseados em conhecimento para engenharia que exigem tratamentos híbridos de objetos/lógica mais flexíveis. Esta arquitetura considera os conceitos mais formais em pesquisa de *design*, tais como o dilema função x forma e o papel do comportamento. Estes conceitos não são tratados em nenhum sistema de modelagem de produtos de conhecimento dos autores.

No modelo proposto, o problema da manipulação está associado às relações de ligação (**linked-to**) e aos procedimentos agregados (**if-needed** e **if-changed**). As relações de ligação ajudam a organizar a composição do artefato. Os procedimentos agregados revelam-se apropriados para aplicar o conhecimento das restrições e das funcionalidades impostas ao artefato. A integração entre objetos e lógica proposta é original no sentido de que exige um ambiente mais flexível do que os encontrados na literatura. Neste particular, XLOG+ parece ser o mais indicado. Um trabalho futuro seria a evolução do XLOG+ para programação com restrições (*constraint logic programming*).

Os problemas revelados pelo protótipo está na modelagem dos próprios objetos que pode conduzir a um sistema rico em objetos, mas computacionalmente ineficiente e complexo. Para cada área de aplicação deve-se estudar o melhor equilíbrio entre as representações intensionais (i.e. mais atributos e menos objetos) e extensionais (i.e. menos atributos e mais objetos) do conhecimento.

Outra questão está na aplicação de ambientes híbridos de lógica e orientação a objetos que, por um lado, oferecem a criação dinâmica de relações (part-of, is-a, linked-to) e, por outro lado, criam métodos não-convencionais do ponto de vista de linguagem de programação orientada a objetos. De fato, uma instância ou uma classe criadas em C++ só podem ser mudadas via compilação. Em XLOG+ (Bento, 1992), entretanto, as instâncias e classes são dinamicamente criadas - o que é essencial no exercício de *design*..

As idéias aqui apresentadas serão utilizadas na concepção de um sistema baseado em conhecimento para edificações de aço. O objetivo é oferecer ao projetista uma ferramenta que permita a simulação da construção real do artefato. Nesta simulação, o projetista manipulará os sólidos básicos que compõem a construção de um edifício de aço, antevendo os problemas do construtor. As definições de funcionalidade, as restrições e o conhecimento inerente a este tipo de artefato permitirão um aumento de automação no processo de criação do produto final. A topologia do GeneSys, neste caso, deve ser, entretanto, degenerada em uma topologia muito mais simples, visto que os objetos são padronizados e as interseções são encontradas por algoritmos muito simples.

Agradecimentos

Os autores agradecem ao CNPq, FINEP, FAPERJ, The British Council e JNICT pelo suporte a esta pesquisa. Ficam também os agradecimentos a Rolf Fischer por sua constante assistência ao sistema GeneSys e ao Expert Systems Laboratory (Imperial College, Londres) onde o primeiro autor é Honorary Lecturer. Agradecimentos também são devidos aos revisores, cujas críticas foram bastante construtivas.

Referências

Bento,J.,1992. Intelligent CAD in Structural Steel: a cognitive Approach, Phd Thesis, Expert Systems Laboratory, Imperial College, UK.

Fikes,R. and Kehler,T.,1985. The role of frame-based oriented representation in reasoning, Comm.ACM, 9(28), 904-920.

Feijó,B.,1988. Fundamental Steps Towards an Intelligent CAD System in Structural Steel, PhD Thesis, Expert Systems Laboratory, Imperial College, London, UK.

Feijó,B. and Bento,J.,1991. A Cognitive Approach to Design, CMEST Report, AI5/91, IST, Lisbon, Portugal.

Feijó,B. e Fischer,R.,1992. Cognição e Percepção 3D em Modeladores de Sólidos, SIBGRAPI V, 97-103.

GEC Research, 1987. An Introduction to the STRATA AI Toolkit, Marconi Research Centre, Essex, UK.

Haase,B.,1992. Who and What is Smart - Intelligent CAD Capabilities Range from Associativity to "KBE", Design Net, 1(6), June 92, 19-25.

Inference Corporation,1985. ART Programming Manual, Inference Corporation, Los Angeles, USA.

Intellicorp,1986. KEE Software Development System User's Manual, Doc No. 3.0-U-1, IntelliCorp, Mountain View, California, USA.

Koegel,J.F.,1987. POOL: Parallel object-oriented approach to structural analysis and design, Computers & Structures, 40(1), Pergamon Press, 75-82.

MaCabe,F.,1989. Logic and Objects: Language, Applications and Implementation, PhD Thesis, Dept. of Comp., Imperial College, London, UK.

Mostow,J.,1985.Towards Better Models of Design Process, in AI Magazine, Spring 85, 44-57, American Association for Artificial Intelligence, USA.

NBR-8800, 1986. Projeto e Execução de Estruturas de Aço de Edifícios, Associação Brasileira de Norma Técnicas, Brasil.

Neuron Data,1990. NEXPERT Object v.2.0 - User's Guide, Neuron Data Inc., Palo Alto, California, USA.

Perkins,J.(ed.),1990. An Introduction to Product Data Exchange, AE/91/0014 Version 1, CAD-CAM Data Exchange Technical Centre, Leeds.

Prates,A.J.B.P,1993. Fundamentos e Especificações de um Ambiente para Design Baseado em Lógica e Objetos, Dissertação de Mestrado, Departamento de Engenharia Civil, PUC-Rio, Brasil.

Shapiro,E. and Takeuchi,A.,1983. Object-oriented programming in cocurrent prolog, New Generation Computing, 1(1), Springer-Verlag.

Simon,H.A.,1969.The Sciences of the Artificial, MIT, Press, Massachusets, USA.

Scheer,S.,1993. Uma Análise Crítica sobre o Tratamento Cognitivo de Design em Sistemas CAD, Tese de Doutorado, Laboratório de CAD Inteligente, Dept. de Informática, PUC-Rio, Rio de Janeiro, RJ.

Stefik,M., Bobrow,D. and Khan,K.,1986. Integrating access-oriented programming in a multi-paradigm environment, IEEE Software, The Institution of Electrical and Electronic Engineers, Inc., New Jersey, U.S.A.

Watson,A. and Chan,S.,1991. A Prolog-based object-oriented engineering DBMS, Computers and Structures, 40(1), Pergamon Press, 11-21.

Wilson,P.R. and Kennicott,P.R.,1988. Integrated Product Information Model, STEP/PDES Testing Draft, ISO TC184/SC4/WG1.

Wozny,M.J., Turner,J.U. and Preiss,K. (eds.),1990. Geometric Modelling for Product Engineering, Elsevier Science Publishers B.V., The Netherlands.

Zaniolo,C.,1984. Object-oriented programming in Prolog, In: Proc. Int. Symp. on Logic Programming, Atlantic City, USA.