SISTEMAS ESPECIALISTAS PARA A ENGENHARIA DE SOFTWARE

por

Hubert Ahlert

RP-154 Abril/1991

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
Av. Osvaldo Aranha, 99
90210 - Porto Alegre - RS - BRASIL
Telefone: (0512) 271999
Telex: (051) 2680 - CCUF BR
FAX: (0512) 244164
E-MAIL: PGCC@sbu.ufrgs.anrs.br

Correspondência: UFRGS-CPGCC
Caixa Postal 1501
90001 - Porto Alegre - RS - BRASIL

Editor: Ingrid E. S. Jansch Pôrto

RESUMO

Este trabalho apresenta um estudo abrangente sobre o uso de sistemas especialistas e técnicas de inteligência artificial na área de engenharia de software. É apresentada uma seleção das principais referências bibliográficas que discutem o assunto e, com base nesta literatura, é avaliado como estes sistemas são aplicados para construir ferramentas que dão suporte ao ciclo de vida do software. No final, é apresentada uma proposta de pesquisa envolvendo uma ferramenta que utilize as técnicas discutidas nesta monografia.

ABSTRACT

This work presents a broad study about the use of expert systems and artificial intelligence techniques in the software engineering area. A selection of the relevant bibliographic references that discuss the subject is presented and, based on this literature, is appraised how these systems are applied to build tools which give support to the software lifecycle. At last, a research proposal, enclosing a tool that uses techniques discussed in this monograph, is presented.

PALAVRAS-CHAVE:

Sistemas Especialistas, Inteligência Artificial, Engenharia de Software, Ferramentas para Construção de Sistemas.

SUMÁRIO

LISTA DE FIGURAS

sistemas especialistas e avaliar sua utilização no ambiente de desenvolvimento de sistemas aplicativos e na engenharia de software de um modo geral.

O trabalho tem início apresentando, no capítulo 2, uma discussão sobre as principais idéias e conceitos ligados à tecnologia de sistemas especialistas. No capítulo 3, os sistemas especialistas são alvos de estudo sob o prisma de sua aplicação na engenharia de software. No final deste mesmo capítulo é apresentada uma proposta de pesquisa envolvendo uma ferramentas que utilize as técnicas aqui discutidas.

2. SISTEMAS ESPECIALISTAS

Neste capítulo são discutidos os principais conceitos e idéias, bem como terminologias e princípios que fundamentam a tecnologia dos sistemas especialistas e a área de inteligência artificial.

2.1 Revisão Conceitual

Antes de iniciar a discussão a respeito de sistemas especialistas e visando uma perfeita compreensão do ambiente que envolve estes sistemas é realizada, nesta seção, uma revisão de conceitos que fundamentam o assunto. Estes conceitos estão associados ao acervo tecnológico da área de Inteligência Artificial (I.A.). São eles:

Raciocínio:

- Faculdade que permite deduzir um juízo a partir de determinadas premissas, tirar conclusões a partir de um conjunto de fatos, diagnosticar causas possíveis para uma determinada condição e gerar hipóteses sobre uma situação.

Pode ser classificado em :

.raciocínio rigoroso (baseado na lógica matemática);
.raciocínio natural (baseado em lógicas flexíveis como a modal, a temporal, etc.).

Inteligência Artificial:

- Peculiaridade de um artefato ou máquina que tenha a capacidade de coletar, juntar, escolher, entender e conhecer, ou seja, característica que simula o processo de raciocínio humano.

- É a ciência de fazer com que as máquinas executem tarefas que normalmente necessitariam de inteligência (julgar, compreender, raciocinar e correlacionar) caso fossem realizadas pelos homens. Esta área da informática considera o uso de programas de computador e técnicas de programação para clarear os princípios de inteligência em geral e do pensamento humano em particular (adaptação, aprendizagem e raciocício).

- Segundo os pesquisadores da inteligência artificial, a maioria do trabalho executado na natureza é não-matemático, ou seja, não obedece a fórmulas pre-definidas. Quase todo ato de pensar profissional é feito pelo raciocínio e a maior parte do raciocínio é feito por **inferência** (derivar conclusões a partir de premissas), não por cálculo. O ser humano, na maioria de suas atividades do cotidiano, deve tomar decisões tendo como base os fatos que lhe são apresentados e tirando conclusões a partir da comparação com situações análogas, anteriormente enfrentadas, ou a partir de uma ordenação, seleção, agrupamento e correlação destes fatos. Uma tomada de decisão não se efetiva pela simples aplicação de fórmulas matemáticas mas sim pelo correlacionamento de uma série de fatos ou premissas, sujeitos à aplicação de regras, e destes extraindo as devidas conclusões, mediante um processo de inferência. Assim, é possível concluir que a inferência deve ser o melhor método para dar raciocínio à máquina e, por conseguinte, fornecer-lhe **inteligência artificial.**

- Para Schneider, Marques e Cohn [SCH 85], os estudos na área de inteligência artificial procuram direcionar esforços em duas abordagens básicas:

a) Núcleo básico da IA: considera os mecanismos básicos e técnicas de implementação que são comuns a várias aplicações dentro da inteligência artificial.

b) Tecnologias da IA: consideram abordagens específicas da IA visando, basicamente, simular atividades intelectuais executadas por seres humanos.

O núcleo básico focaliza:

. estratégias de busca da solução do problema: como encontrar uma sequência de operações para transformação de um estado inicial em um estado desejado?

. modelagem e representação do conhecimento: para a aquisição do conhecimento por parte do sistema, este conhecimento deve, de alguma forma, ser descrito e representado no ambiente deste sistema. A representação do conhecimento é uma combinação de estruturas de dados e procedimentos de interpretação que são usados no sistema. As formas de representação são influenciadas pelo tipo de aplicação e podem ser, por exemplo:

- via lógica;
- via procedimentos;
- via regras de produção;
- via redes semânticas;
- via frames.

. raciocínio e solução de problemas: encontrar as inferências relevantes à solução de um determinado problema;

. linguagens e sistemas: linguagens de programação e sistemas adequados para aplicações em IA.

As tecnologias de IA se concentram em áreas como:

- .sistemas peritos ou especialistas: sistemas inteligentes em alguma aplicação específica;

- .programação automática: a máquina simulando tarefas de analistas e programadores e gerando código executável;

- .visão do computador: a máquina tendo capacidade de "ver", ou seja, processamento da informação a partir do processamento de imagens (imagens como meio de alimentação do computador);

- .linguagem natural: a máquina compreendendo a linguagem natural escrita e falada.

Conhecimento:

- Conhecimento não tem o mesmo significado que informação. Conhecimento é a informação que foi comparada, adaptada, interpretada, selecionada e transformada, ou seja, a informação processada como um todo inter-relacionado.

- Enquanto que a noção de dado está associada aos valores não interpretados de um sistema e a informação é gerada quando os dados foram selecionados e organizados para determinado fim, a noção de conhecimento estrapola a noção de informação visto que o conceito de estrutura é incorporado com o objetivo de explorar relações entre as porções de dados, atribuindo-lhes uma forte carga semântica.

- O conhecimento é uma coleção de fatos relacionados com uma certa estrutura (taxonomia) e inclui as relações entre os tipos de objetos, de classes e de indivíduos, e determinam como as propriedades de uns se aplicam aos

outros.

- O conhecimento pode aparecer sob a forma de objetos (entidades a serem modeladas), fatos (asserções e definições), conceitos, relações lógicas, teoremas e regras, algoritmos e metaconhecimentos (conhecimento sobre conhecimento).

Fatos:

- Fatos expressam alguma coisa ou situação, assumida como verdadeira em relação a informações conhecidas.

- Representam relações válidas entre objetos.

Ex.: "José trabalha no setor de vendas"
"João é chefe do setor de vendas"

Conceitos:

- É a representação declarativa de objetos de um domínio da aplicação, com classes abstratas e instâncias concretas. Correspondem aos conhecimentos retirados de livros e incluem os termos básicos do domínio.

- Conceitos são fatos de domínio público.

Ex.: "Um automóvel tem motor, chassis e rodas"

Regras:

- Regras expressam a maneira como novos fatos podem ser inferidos.

Ex.: "O chefe de X é Y se X trabalha no setor S e Y é o chefe do setor S"

- Regras são associações empíricas ligando causas e efeitos, graus de crenças e probabilidade de conclusão, ou seja, situações desejáveis e ações a empreender. Correspondem aos conhecimentos fornecidos por um perito e é baseado na experiência.

- Conhecimento heurístico (que vem da prática; experimental).

Regras de Produção:

- Permitem a representação do conhecimento segundo a forma geral:

SE condição ENTÃO conclusão

, onde condição/conclusão são pares (atributo,valor) ou triplas (objeto, atributo, valor).

- Permitem representar situações ou condições que, por sua vez, ajudam a definir a solução do problema através da indicação das transformações a serem realizadas no conjunto de informações armazenadas sobre o problema.

Ex.: **SE** "carro não arranca" e
"faróis nao funcionam"
ENTÃO
"bateria pode estar estragada"

Redes Semânticas:

- Permitem representar o conhecimento através de modelos (coleção de regras interrelacionadas) formulados mediante o uso de grafos. O conhecimento é representado como um conjunto de nodos (objetos e/ou conceitos) ligados entre si através de um conjunto de arcos etiquetados (relações). Existem arcos especiais que transmitem a herança dos atributos na rede. Estes recebem os rótulos:

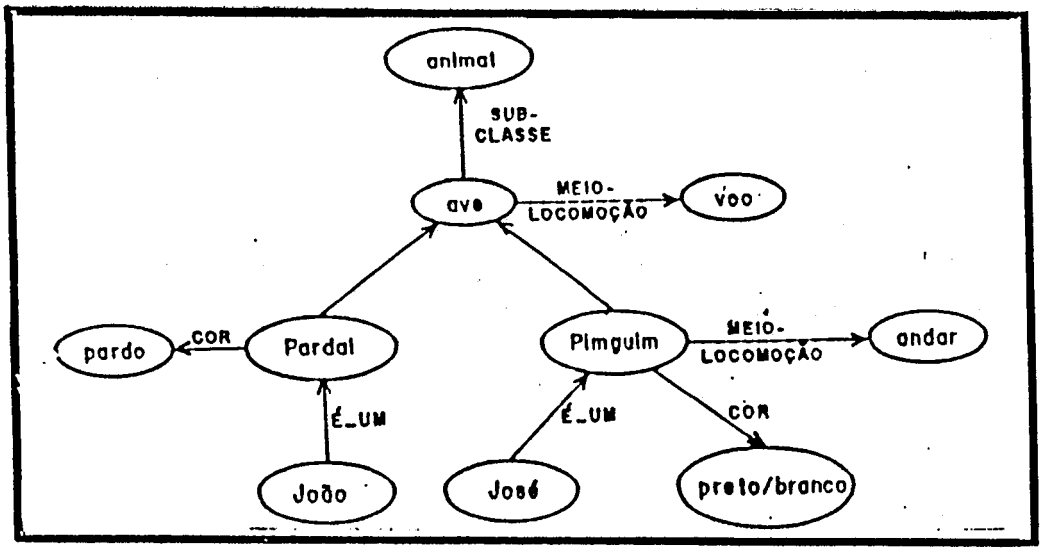
.É um/ tipo de --> É elemento de (classificação, generalização)

.Tem um/ Parte de/ Subclasse --> É subconjunto de (agregação).

A inibição da herança de um atributo é obtida através da adição de um nodo específico (quebra herança) que representa o atributo peculiar. O arco que liga este nodo deve ter o mesmo nome do atributo a ser particularizado.

- As redes semânticas facilitam as deduções sobre "hierarquias de heranças" e são uma forma natural de representação em aplicações onde o raciocínio é baseado em taxonomias muito complexas.

Ex.: (Extraído de [VIC 89])




```
é_um(pinguim,ave).
é_um(pardal,ave).
meio_loc(ave,voo).          %definição da herança
meio_loc(pinguim,andar).    %quebra de herança
cor(pinguim,preta_branca).
cor(pardal,pardo).
é_um(ave,animal).
loc(X,Y):- é_um(X,Z),loc(Z,Y),!.
loc(X,Y):- é_um(X,Z),cor(Z,Y).!.
```

```
arco(ave,é_um,animal).     %relações de um objeto
arco(pinguim,é_um,ave).
arco(pardal,é_um,ave).
     .
     .
     .
prop(X,P,Y):-arco(X,é_um,Z), prop(A,P,Y),!.
```

Frames (enquadramentos):

- Permite a representação do conecimento mediante uma estrutura (ou coleção de nodos da rede semântica e nodos vazios) que descreve um objeto (com um conjunto de propriedades), um conceito ou uma situação. Permite estender as redes semânticas através da adição de atributos fixos e/ou variáveis.

- Frames são semelhantes a registros do Pascal ou listas de propriedades no Lisp e visam colecionar todas as propriedades do objetos em um único local.

- Segundo Viccari [VIC 89], a terminologia adotada nesta forma de representação do conhecimento é:

.Frame : Estrutura, registro
.Slot : Campo, item, atributo, relacionamento
.Faceta : Subitem
.Conteúdo: Valor, procedimentos
.Scripts : Aglomeração de conhecimentos descrevendo relações entre frames.

```
Ex.: frame (automóvel,(
          [tipo,veículo],
          [nro_rodas,quatro],
          [objetivo,transporte_de_passageiros],
          [mecânica,motor_de_combustão],
          [fonte_de_energia,combustível_volátil]
          )).
```

Base de Conhecimento (BC):

- Base de conhecimento não é o mesmo que base de dados, termo já corriqueiro na informática. A diferença pode ser ilustrada mediante uma analogia com a atividade de um médico em uma situação de visita ao leito do paciente. A base de dados do médico, naquele instante, é o prontuário onde está registrado o histórico clínico, medicação, sinais vitais, resposta do organismo, etc. Este conjunto de dados ainda deve ser interpretado. Para tal, o médico utiliza-se de uma base de conhecimentos médicos, obtidos através de sua formação acadêmica e através de anos de experiência. Trata-se de um conjunto de fatos, crenças (tudo que pode ser representado através de algum aparato lógico e nem sempre é verdadeiro) e, talvez o mais importante, conhecimento heurístico (por descoberta) adquirido ao longo do tempo.

- A base de conhecimentos é um conjunto de fatos e um conjunto de regras capazes de permitir a derivação de outros fatos. Numa base de connhecimentos são mantidas informações e conhecimento sobre um determinado domínio (problema). Desta forma, um sistema baseado no conhecimento pode desempenhar uma função de acompanhamento, diagnóstico e assistência. Nesta base de conhecimentos, os dados estão estruturados de forma irregular, existem mais descrições do que dados, o esquema conceitual se integra aos dados, a informação está associada à função de interpretação existente no sistema, os dados e as regras estão organizados em unidades que facilitam a inferência de novos fatos, a consulta proporciona resultados e os fatos são pontos de partida para a aplicação de regras.

- Santos e Gonçalves discutem em [SAN 89] os fatores que influenciam na construção de Bases de Conhecimentos e colocam como requisitos para a obtenção das regras que irão compor a BC, os seguintes:

a) a apropriação dos fatos e heurísticas que são normalmente considerados pelo especialista;
b) a identificação do especialista e a verificação de seu estilo cognitivo e grau de competência;
c) as formas de investigação adotadas na solução de problemas (relacionadas com o requisito do item a) e o nível de competência a ser incorporado à BC;
d) a natureza dos problemas resolvidos pelo especialista;
e) a quem se destina o sistema, ou seja, quem será o usuário do sistema produzido.

Sistema baseado em conhecimentos (SBC):

- É um sistema que incorpora conhecimentos, experiências e processos de tomada de decisão de peritos em determinada área profissional. Processa conhecimentos e tenta simular algumas formas de raciocínio humano na solução de problemas e, normalmente, não associadas a saídas numéricas. Uma base de conhecimentos (fatos, relações lógicas e regras) fornece o suporte para que o conhecimento possa ser processado por este sistema.

- Um sistema especialista, por sua vez, é um sistema baseado em conhecimentos que atua em áreas e tarefas bem definidas e requer as aptidões especiais fornecidas por um perito no assunto.

- Como ilustrado na Fig. 2.1, para Bibel [BIB 84] e Forsyth [FOR 86], a arquitetura de um SBC é composto por:

.Módulo de aquisição de conhecimento .....
(estratégias de indução e aprendizagem);

.Módulo base de conhecimentos .....
(métodos de representação do conhecimento);

.Módulo de inferência/resolução de problemas .... (métodos de raciocínio);

.Módulo de explanação ..... (janela para o usuário).

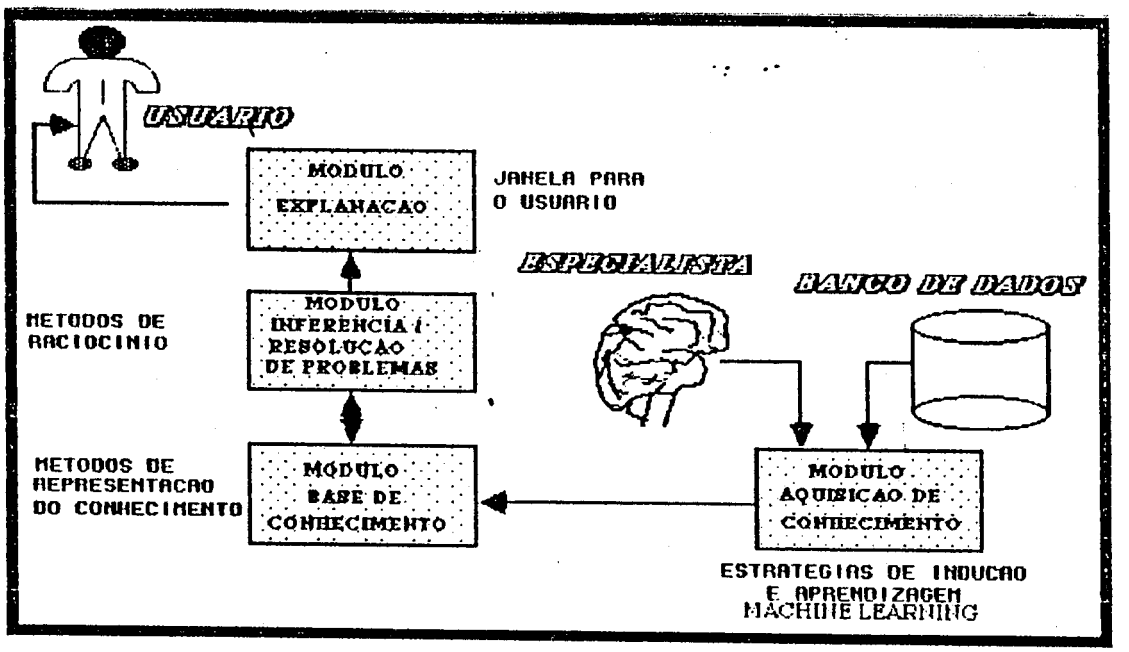


Fig. 2.1 - Arquitetura de um sistema baseado em conhecimentos (adaptado de [FOR 86])

Bibel [BIB 84] adiciona mais um componente na arquitetura como módulo de interface com o usuário. Forsyth [FOR 86] representa esta interface no próprio módulo de explanação.

Sistema de Representação do Conhecimento:

- É o conjunto de recursos operacionais que permitem uma representação do conhecimento. Esta representação pode ser:

.Lógica: o conhecimento é descrito via asserções postuladas como verdadeiras (um conjunto de

axiomas) a respeito do que deve ser modelado. Se baseiam em uma semântica bem definida (por exemplo, lógica de primeira ordem).

.Estrutural: o conhecimento é descrito com objetos e relações sobre as entidades a serem modeladas (conceitos, hierarquia de estruturas, descrições de classes de elementos e instâncias individuais ou componentes de objetos).

.Procedimental: o conhecimento é descrito através de procedimentos (Estados e transições, operações e mensagens).

Aprendizagem:

- Consiste em adquirir novos conhecimentos, reestruturar conhecimentos anteriores ou melhorar o desempenho de tarefas.

- A partir de uma aquisição de conhecimentos (colocar a perícia humana numa forma que possa ser processada pelo computador), a aprendizagem força a transformação e a expansão do conhecimento existente, ou seja, a partir de informações obtidas do ambiente externo ou do perito, o sistema de aprendizagem modifica e/ou amplia a base de conhecimentos e, com isto, as alterações provocadas na base resultam em novo comportamento do mecanismo de inferência.

- O processo de aprendizagem pode ser por:

.Instrução: assimilação a partir de fonte externa (tutor).

.Dedução: a partir de uma teoria formal é realizada uma inferência dedutiva para expandir o conhecimento (aplicação de regras de inferência).

.Analogia: a partir de comparação de situações semelhantes, pressupondo a existência de conhecimento prévio sobre o problema e suas respectivas soluções nos mais diversos domínios.

.Indução: inferências feitas também a partir de fatos não verdadeiros (ao contrário do caso dedutivo).

.Implantação direta: a partir de um conhecimento nulo ou quase nulo (alfabetização) e onde as necessidades inferenciais são quase nulas (Ex.: inclusão de novas regras na base de conhecimentos).

Tem autores, como Oliveira [OLI 90], que relacionam as técnicas de aprendizagem por instrução e implantação direta como casos de aquisição de conhecimento, reservando o termo aprendizagem para situações onde a inferência é mais forte.

- A aprendizagem por indução pode ser subdividida em:

.Por exemplos: a partir de um conjunto de exemplos fornecidos por um instrutor, o sistema abstrai regras gerais para descrever o conceito a ser aprendido.

.Por observação e descoberta: a figura do instutor desaparece e o sistema obtém um conjunto de exemplos a partir da interação direta com o ambiente externo podendo provocar interferência (experimentação ativa) ou não (observação passiva) neste ambiente.

- Na aprendizagem por dedução, um dos métodos mais difundidos é a aprendizagem baseada em explicações onde, a partir de uma descrição e exemplo inicial do conceito, de uma teoria de descrição do domínio e de um critério de aceitação da descrição do conceito, o sistema constrói, aplicando um mecanismo de prova (explicação), a descrição do conceito a ser aprendido dentro do critério de operacionalidade.

2.2 Programação em Lógica: Princípios e Noções Básicas

A programação em lógica parte do princípio que um programa é um modelo (coleção de regras interrelacionadas, associadas à hipótese ou diagnóstico) de um determinado problema ou situação expresso através de um conjunto finito de sentenças lógicas.

Ao contrário de programas escritos em Fortran, Algol ou Pascal, um programa em lógica não é a descrição de um procedimento para obter soluções do problema. Um programa em lógica assemelha-se a um banco de dados, a exceção do fato de que um banco de dados descreve somente observações como "Pedro é o gerente de Jorge", enquanto que as sentenças de um programa em lógica podem também retratar um escopo mais genérico (regras), como: " o gerente é um superior hierárquico do funcionário".

Um programa em lógica, na sua forma mais geral, nada mais é do que uma teoria completamente formalizada e descrita por um número finito de axiomas. A programação em lógica se baseia nos resultados da mecanização da lógica de primeira ordem. Cabe relembrar que a lógica de primeira ordem ou cálculo de predicados pode ser caracterizada como um sistema formal apropriado à definição de teorias do universo de discurso da matemática e cuja linguagem consiste de:

- .Símbolos lógicos: variáveis, conectivos lógicos e quantificadores.

- .Símbolos não lógicos: constantes (para denotar objetos), símbolos predicativos (para denotar relacionamento entre objetos) e símbolos funcionais (para denotar funções sobre os objetos).

A programação em lógica exemplifica um estilo de programação denominado programação declarativa (assercional ou não-procidemental) em contraste com a programação procidemental (imperativa), típica das linguagens tradicionais. A programação declarativa engloba também a programação funcional que tem a linguagem Lisp como o seu exemplo mais conhecido.

O interesse no paradigma de programação em lógica começou a ser despertado com o advento da linguagem PROLOG (PROgramming in LOgic).

Dentro do desenvolvimento atual, programação em lógica inclui também alguns comandos extra-lógicos (caso da linguagem Prolog) [CAS 87] para suprir as facilidades tradicionais de linguagens de programação, como entrada e saída, por exemplo. Estes comandos fogem da proposta original de programação declarativa.

Os conceitos de chamada (ou consulta) e de resposta também diferem das noções encontradas nas linguagens tradicionais. Uma consulta a um programa em lógica é uma afirmação exprimindo as condições a serem satisfeitas por uma resposta correta em presença da informação descrita pelo programa. A maioria dos sistemas para programação em lógica reduzem a busca de respostas corretas à pesquisa de refutações (refutação = dedução de uma contradição) a partir das sentenças do programa e da negação da consulta.

A resposta de uma consulta a um programa em lógica não se limita apenas a indicar que uma suposição acerca da informação contida no programa é falsa ou verdadeira. A resposta efetivamente exige informações extraídas do programa e pode vir acompanhada de uma explicação sobre como foi obtida, expressa em termos da refutação que a gerou.

Em vista das colocações aqui feitas, referentes aos princípios que fundamentam a programação em lógica, é possível salientar que as linguagens que seguem esta abordagem incorporam características que são relevantes para o desenvolvimento de sistemas especialistas baseados na representação de conhecimentos. O que reforça este argumento é o fato de que a lógica formal utiliza um método matemático que permite representar, de forma natural, todo o conhecimento (fatos, regras, conceitos) através de predicados.

2.3 Os Sistemas Especialistas

Sistema especialista é um programa ou conjunto de programas inteligentes que usam conhecimentos e mecanismos de inferência na solução de problemas que, normalmente, só podem ser resolvidos por peritos humanos.

O poder dos sistemas especialistas está no conhecimento que contêm. Extrair este conhecimento da cabeça dos especialistas humanos e coloca-lo em um computador é um problema de aquisição de conhecimento. Esta aquisição de conhecimento está sendo um dos gargalos enfrentados pelos profissionais da informática interessados em ambientes de I.A.

Existem dois tipos de conhecimento. O primeiro é composto pelos fatos de domínio público, registrado em livros e jornais, ou mesmo o que o professor ensina em sala de aula. O segundo tipo é o conhecimento heurístico, que vem da boa prática e do bom julgamento sobre algum assunto específico. É o conhecimento experimental que os especialistas adquirem ao longo de anos de experiência.

Para a representação deste conhecimento (peça fundamental dos sistemas especialistas) em computador, torna-se necessária a utilização de linguagens e estuturas de armazenamento voltadas aos conceitos apresentados na

seção 2.1.

A construção de sistemas especialistas impõe certas condições como, por exemplo:

.deve haver pelo menos um perito humano reconhecido que seja especialista na tarefa a ser automatizada;

.a fonte básica do conhecimento procurado no perito é a experiência e o julgamento (conhecimento heurístico);

.o perito deve ser capaz de explicar seus conhecimentos e experiências e os métodos usados para aplicá-los em problemas específicos;

.o domínio da aplicação deve ter fronteiras bem definidas.

Para Viccari [VIC 90], os sistemas especialistas apresentam características comuns:

.são limitados a um domínio específico;
.podem fazer inferências com base em dados incertos;
.podem explicar seu caminho de resolução;
.são projetados para expansão por acréscimo de conhecimento;
.contemplam hipóteses múltiplas simultaneamente;
.o raciocínio heurístico é implementado através de regras práticas.

Nesta seção, os sistemas especialistas são abordados numa visão panorâmica sobre as aplicações tradicionais onde são utilizados, as linguagens para este tipo de sistema e as limitações atualmente presentes nestes sistemas.

2.3.1 Paradigmas de Programação e Linguagens

O esforço de desenvolver uma linguagem para a inteligência artificial começou em 1965 num trabalho de parceria entre o cientista da computação Edward Feigenbaum e Joshua Lederberg, professor de genética. Há muito tempo eles pensavam, fascinados, na possibilidade de usar computadores para modelar e assistir o pensamento científico. Juntos começaram a escrever programas que pudessem inferir hipóteses moleculares a partir de dados químicos, isto é, criar e estudar moléculas a partir da união de átomos de substâncias químicas. Eles perceberam imediatamente que seus programas não poderiam ter um desempenho efetivo e prático sem que possuíssem em seu interior um considerável conhecimento de fisica e química.

Lederberg recrutou, então, os talentos e especialização de outro cientista, o bioquímico Carl Djerassi, mais conhecido como o pai da pílula de controle da natalidade. O time interdisciplinar, reunindo ciência da computação, genética e química, trabalhou vários anos para produzir um programa inteligente (sistema especialista) para explicar os detalhes de uma estrutura molecular, a partir de dados químicos.

Muitos pesquisadores de IA desenvolvem suas próprias linguagens com peculiaridades projetadas para lidar com problemas específicos na área. A primeira idéia das linguagens de programação para IA foi a de usar o computador para manipular símbolos arbitrários e não somente números. Esta idéia, e as técnicas de processamento de listas derivadas desta idéia, são originárias de linguagens como Lisp e IPL, uma das primeiras linguagens de programação em IA.

Quando se fala em linguagens de programação para IA é importante analisar aspectos como estruturas de dados e ambientes de apoio à programação.

As estruturas de dados que irão representar o conhecimento devem:

.espelhar, numa maneira natural e conveniente, certas peculiaridades das entidades a serem manipuladas;
.ser eficientes no gerenciamento do espaço de armazenamento e no tempo de acesso.

As características importantes em um sistema de apoio à programação de sistemas especialistas são:

.diálogo interativo;
.facilidades interativas de depuração;
.um editor prático;
.rotinas de entrada/saida, preferencialmente com recursos gráficos.

Analisando o elenco de linguagens atualmente disponíveis no mercado e o paradigma de programação associado a estas linguagens, poderia ser questionado: Qual o modelo de programação e quais as linguagens são mais indicadas ao desenvolvimento de sistemas especialistas?

O quadro abaixo, extraído de [ALT 90], ilustra os paradigmas de programação habituais e as características que estes modelos incorporam:

Paradigma de Programação Procedural:

Linguagens	:	Fortran, Pascal, C, Simula, ADA
Usos	:	Processamento numérico, processamento de dados clássico.
Características:		Variáveis, possíveis efeitos colaterais em vista de variáveis globais, estruturação, organização, orientado a fluxo de controle, modificação dinâmica de listas e funções.

Paradigma de Programação Funcional:

Linguagens : Lisp (puro), FP, HOPE, ML, Miranda

Usos : Processamento de símbolos,processamento de linguagem natural, sistemas especialistas.

Características: Funções, parâmetros, sem variáveis, sem atribuições, manipulação uniforme de dados e programas, formalmente definido pelo lambda-calculus na representação da teoria de funções de árvores recursivas.

Paradigma de Programação em Lógica:

Linguagens : PLANNER, PROLOG, HSRL, HCPRVR

Usos : Prototipação, prova de teoremas, regras de gramática, sistemas especialistas.

Características: Cláusulas de Horn como regras de inferência, encadeamento reversivo, relevância na sequência de regras, sem organização, sem estruturação, sem modularização, declarativa, atômica, formalmente definida pelo cálculo de predicados.

Paradigma de Programação orientada a objetos:

Linguagens : Simula, Flavors, LOOPS, Smalltalk.

Usos : Sincronização, comunicação entre processos, descrição de topologia de redes, aprendizado assistido por computador, CAD, compiladores, sistemas operacionais, aplicações de escritório.

Características: Conceito de classe, hierarquia,herança, comunicação entre objetos através de passagem de mensagens, estruturação, modularização, baseado em tipos abstratos de dados.

Avaliando este quadro comparativo de paradigmas de programação constata-se que, para o desenvolvimento de sistemas especialistas, tanto a programação funcional (LISP) quanto a programação em lógica (PROLOG) tem características que facilitam a representação do conhecimento e, portanto, estão sendo qualificados mais frequentemente para este tipo de aplicação.

A programação em lógica, no entanto, permite, através de uma notação simples (lógica orientada a regras) a representação de qualquer fato do mundo real e uma fácil compreensão do que está sendo modelado. Também, através do mecanismo de regras de inferência, é possivel dar inteligência aos sistemas, permitindo que novos fatos e regras sejam criadas e, com isto, oportunizando uma aprendizagem atomática por parte do sistema.

Uma linguagem bastante expressiva no contexto de linguagens de programação em lógica e a mais difundida no momento dentro do paradigma de programação em lógica, presente em um significativo número de sistemas especialistas, é a linguagem Prolog (projetado por Colmerauer na Universidade de Marseille em 1973). Esta linguagem deve ser vista como uma implementação das idéias de programação em lógica dentro de um subconjunto de cláusulas definidas.

Como principais características da linguagem Prolog [CAS 87] que podem ser estendidas para peculiaridades da programação em lógica e, conseguentemente para os sistemas especialistas, convém salientar as seguintes:

- .apresenta uma semântica declarativa inerente à lógica;
- .representa uma implementação da lógica como linguagem de programação;
- .permite a definição de programas inversíveis, ou

seja, não faz distinção entre argumentos de entrada e saída e, assim, permite definição de programas com mais de uma finalidade;
.permite a obtenção de respostas alternativas;
.permite recuperação dedutiva de informações;
.representa programas e dados através do mesmo formalismo (cláusulas);
.incorpora um mecanismo uniforme para análise, seleção e criação de estruturas de dados;

Por outro lado, Altenkrueger [ALT 90] salienta que diferentes tipos de conhecimentos necessitam de diferentes formas de representação deste conhecimento e, portanto, é interessante realizar-se uma avaliação dos recursos que cada linguagem oferece para que a escolha recaia sobre aquela que melhor atenda as necessidades do sistema especialista em questão.

Segundo o mesmo autor [ALT 90], na escolha de ambientes de desenvolvimento de sistemas especialistas é interessante considerar:

.tamanho e complexidade do sistema;
.desempenho;
.prototipação vs. ciclo de vida;
.modularização, flexibilidade e portabilidade;
.integração com outros sistemas;
.formas de representação do conhecimento e mecanismos de inferência;
.interfaces do usuário (menu, gráfica, linguagem natural).

Convém salientar que, dentro do ambiente de sistemas especialistas, surge também um novo paradigma de construção de sistemas. Este paradigma é representado pelo conceito de "Shell" (embrião, gerador de sistema). Trata-se de um software gerador de sistemas especialistas que recebe os fatos e regras e, a partir deles, elabora toda a

especificação, gerência e manipulação dos conhecimentos necessários à execução do sistema sendo concebido. Em outras palavras, "shell" é um sistema especialista vazio que deve ainda receber, através de uma linguagem própria, o conjunto de regras para seu funcionamento.

Este gerador de sistemas especialistas (expert system shell), a exemplo do TWAICE [MEL 90], vem munido de um bem selecionado formalismo de representação do conhecimento, uma estratégia predefinida de solução de problema (descrição da forma como os dados ou conhecimentos sobre o domínio da aplicação são usados para obter uma solução para o problema), mecanismos de explanação e "trace" e facilidades de apresentação de resultados (saída) adaptado ao formalismo de representação do conhecimento e à estratégia de solução do problema.

Como aplicações pertinentes a inteligência artificial que podem ser concebidas com ferramentas de construção de sistemas especialistas (shell), Gevarter [GEV 87] coloca:

.classificação: seleção de respostas a indagações a partir de um conjunto de alternativas e com base em fatos informados. Neste contexto aparece:
  - seleção de hipóteses baseado em evidências;
  - diagnose;
  - depuração recomendando medidas de correção para situações adversas diagnosticadas;
  - aconselhamento de auxílio ao usuário principiante sobre o uso de determinadas funções;

.projeto e síntese: configuração do sistema especialista com base em possibilidades alternativas;

.assistente inteligente para suporte à decisão;

.prognóstico sobre possibilidades futuras com

base na informação corrente;

.ordenação de tempo (scheduling) para tarefas com base nos recursos existentes;

.monitoração numa atividade de constante observação para permitir os prognósticos;

.controle de processos: monitoração e ações de ajuste;

.descoberta e digestão da informação (avaliação da situação) para criar novos conceitos, relações e ordenações;

.raciocínio através de exemplos;

.seleção de uma série de ações (planejamento) para atender aos objetivos do usuário a partir de um conjunto complexo de alternativas;

Alguns exemplos de "Shell", disponíveis no mercado, estão ilustrados na figura 2.2.

NOME	FABRICANTE	Formalismos p/ Repres.do Conhec.					Interface c/ outro software
		Fatos	Regras	Frames	Lógica	Proc.	
Arity/Expert Development Package	Arity Corp.	SIM (*)	SIM	SIM	SIM		Prolog
PATER	TECSIS-Tecnologia de Sistemas Inteligentes LTDA	SIM	SIM		..		
Intelligence Compiler	Intelligence Ware, Inc.	SIM	SIM	SIM (**)	SIM	SIM	Pascal,DOS Lotus 123, DBase III
GURU (***)	Micro Data Base Systems, Inc.	SIM	SIM			SIM	SQL, Pascal, C
KEE	IntelliCorp	SIM	SIM	SIM	SIM	SIM	
KES	Software Arch. and Engineering, Inc.	SIM	SIM			SIM	C
ESP Frame Engine	Expert Systems International	SIM	SIM	SIM	SIM	SIM	Prolog

(*) - Fatos em termos de conceitos, definições e propriedades
(**) - Um frame pode ser do tipo objeto, tabela e "B-tree"
(***)- Outras facilidades: banco de dados, linguagem de programação estruturada, gráficos estatísticos, linguagem natural

Fig. 2.2 - Exemplos de "Shell"

2.3.2 Aplicações Tradicionais

O sistema especialista já é hoje um suporte de alto nível para especialistas humanos, o que explica a terminologia "assistente inteligente" também comumente

utilizada. São programas de inteligência artificial geralmente construídos com a capacidade de explicar as linhas de raciocínio que levaram às decisões. Alguns podem explicar até mesmo porque rejeitaram certos caminhos de raciocínio e escolheram outros.

O maior grupo de sistemas especialistas em uso hoje está centrado na medicina. Uma das principais razões para o desenvolvimento dos sistemas especialistas é que eles podem vir a substituir uma mão-de-obra das mais caras do mundo, porque o valor agregado do trabalho desses profissionais é alto e exige anos de educação, treinamento e experiência.

Além da medicina, principalmente no fornecimento de diagnósticos, outras áreas do conhecimento humano são alvos da aplicação dos sistemas especialistas. Dentre as aplicações tradicionais, podem ser mencionadas:

.tradução automática de linguagens;
.demonstração de teoremas;
.jogos;
.reconhecimento da fala e compreensão da lingua natural;
.composição musical;
.sistemas na área jurídica.

2.3.3 Limitações

As limitações e obstáculos atuais, enfrentados pelas pesquisas em sistemas especialistas, acontecem principalmente em áreas como:

.desenvolvimento de processadores de língua natural;
.comunicações orais;
.implantação de alguns aspectos mentais como o

aprendizado e a criatividade.

Segundo Viccari [VIC 89], dificuldades ambientais também devem ser consideradas e podem ser agrupadas nos seguintes itens:

- .dificuldades de uso da máquina: treinamento técnico e assistência a usuários não qualificados;
- .dificuldades de integração de sistemas;
- .ineficiência de codificação: linguagens atuais com deficiências na representação do conhecimento;
- .insuficiência de arquitetura: arquitetura baseada nas máquinas de Von Neumann são muito limitadas para as aplicações da inteligência artificial.

Todas as aplicações atuais dos sistemas especialistas envolvem características de análise do problema onde, a partir de fatos conhecidos, procura-se determinar as regras que satisfazem estes fatos (dos efeitos para as causas; decompor os fatos nas regras que os suportam). Por outro lado, as dificuldades reais, atualmente presentes na área de I.A., estão relacionadas com a síntese do problema onde o objetivo é elaborar novas regras a partir de um conjunto de fatos (das causas para os efeitos; agrupar fatos particulares para gerar um todo que os resume).

A característica de síntese envolve a utilização de mecanismos de aprendizagem a serem incorporados aos sistemas especialistas e a dificuldade atual está relacionada com o desenvolvimento de estratégias eficientes de aprendizado (machine learning) e de estratégias de aquisição automática de conhecimento.

3. SISTEMAS ESPECIALISTAS E TÉCNICAS DE I.A. NA ENGENHARIA DE SOFTWARE

O desenvolvimento de software pode ser visto como uma transformação de uma especificação informal, expressa em linguagem natural, para outra especificação, formal e sem ambiguidades, que possa ser processada pelo computador.

A engenharia de software, através de suas técnicas e metodologias, procura conciliar as características de uma linguagem natural (informal e imprecisa) com as exigidas pelos computadores (formal e consistente). Neste sentido, procura-se obter recursos que tornem a atividade de programação de computadores mais próxima das formas de raciocínio e expressão humanas. Sob este enfoque, os sistemas especialistas passam a ser alvo de estudo e um dos maiores obstáculos que irão enfrentar junto à construção de sistemas é a amplitude do domínio deste tipo de aplicação.

Considerando esta proposta, os sistemas especialistas são analisados, neste capítulo, sob o enfoque de sua aplicabilidade na área de engenharia de software.

3.1 Aplicação da I.A. na Engenharia de Software

O apêndice "A" registra um levantamento de publicações de autores que pesquisam o uso e aplicações de sistemas especialistas e técnicas de IA na área de engenharia de software. Esta coletânea de referências bibliográficas foram analisadas e sobre elas registrados alguns breves comentários. De posse desta análise bibliográfica, a presente seção visa consolidar as idéias expostas pelos autores em seus trabalhos com a intenção de identificar os temas de maior relevância e as linhas de pensamento que direcionam as atuais pesquisas nesta área.

3.1.1 Aplicações em Destaque

O estudo da literatura constante no apêndice "A" pode ser sintetizada pelas seguintes idéias e aplicações práticas que se enquadram no contexto da inteligência artificial aplicada à engenharia de software:

a) técnicas de IA na formalização de especificações;

b) sistemas especialistas e a programação em lógica como ferramentas para rápida prototipação;

c) sistemas especialistas como mecanismos para gerenciamento de reusabilidade;

d) sistemas especialistas e técnicas de IA no paradigma de construção automática de sistemas;

e) sistemas especialistas no apoio a fases específicas do ciclo de vida do software;

f) computadores de quinta geração: a programação em lógica no papel da engenharia de software.

g) modelos de dados baseados em conhecimentos e sistemas de banco de dados especialistas

Um comentário sobre cada uma destas idéias é, a seguir, apresentado.

a) técnicas de IA na formalização de especificações

A idéia de formalização de especificações está associada ao processo de descrição feita em alguma linguagem, gráfica ou textual, com construções e expressões restritas, que utiliza algum formalismo no qual a

preocupação básica reside na definição precisa e sem ambiguidades.

Uma especificação formal permite verificação automática de consistência, completeza e correção e, aplicando a ela ferramentas apropriadas, é possível conseguir tradução automática de especificações e geração automática de testes.

Na área de inteligência artificial, esta formalização é realizada mediante o uso de uma ou mais formas de representação do conhecimento (fatos, regras, redes semânticas, procedimentos, frames) [ALT 90] que são descritas através de uma linguagem de especificação apropriada.

Por exemplo:

Ex.1: - Uso de regras

.Especificação informal:

"A tarefa de monitoração de um reator atômico se resume a:

- .caso a temperatura do reator estiver elevada, baixar a cadeia de reações;
- .caso for detectado um excesso de radiação no ar, ordenar uma retirada de emergência do local e isolar a área."

.Formalização em Prolog:

```
ctr_radiacao:-temperatura(reator,elevada),!,
              ordem(diminuir_reacoes).
ctr_radiacao:-radiacao(ar,elevada),!,
              ordem(evacuar_local),
              ordem(isolar_area).
loop_monitoracao:-repeat, ctr_radiacao, fail.
```

Ex.2: - Uso de frames

.Especificação informal:

"Carro é um veículo com quatro rodas projetado para o transporte de passageiros e acionado por um motor de combustão interna que utiliza um combustível volátil"

.Formalização em Prolog:

```
frame(carro,
      ([tipo,veiculo],
       [nro_rodas,quatro],
       [objetivo,transporte_passageiros],
       [mecanica,motor_de_combustao],
       [fonte_de_energia,combustivel_volatil]
      )).
```

Quando a engenharia de software passa a ser alvo de formalização, todo conhecimento a ser armazenado na B.C se refere ao conhecimento sobre métodos e técnicas utilizadas na engenharia de software e ao conhecimento sobre o domínio da aplicação.

Em [WOH 88], por exemplo, o protótipo de diagnose de esquemas conceituais possui um editor gráfico que utiliza uma linguagem gráfica (SIMOL), ao estilo do modelo E/R, para representar o esquema conceitual. Esta especificação do modelo conceitual é então formalizada em quatro diferentes predicados Prolog:

Em Simol:

.Retângulo: representa tipos de dados (Ex.: inteiro, data, denominação)

.Círculo..: representa entidades ou objetos (Ex.: carro, pessoa)

.Triângulo: representa eventos ou atividades sobre os objetos (Ex.: cria novo carro, registro de carro)

.Seta.....: representa atributos (Ex.: nome de pessoa, modelo do carro). Além do nome do atributo, a seta vem rotulada, também, com a cardinalidade mínima e máxima do atributo e do seu inverso (opcionalmente especificado para o sentido inverso)

Em Prolog:

.Tipos de dados:

```
Data_object(nome, tipo).
```

.Entidades:

```
Entity(nome,lista_eventos_cria_instan,
       lista_eventos_excl_instan,
       lista_super_classes).
```

.Eventos:

```
Event(nome,lista_super_classes).
```

.Atributos:

```
Attribute(nome,nome_inverso, origem,
       destino,cadinalidade(min1,max1,
       min2,max2),lista_eventos_inic,
       lista_eventos_altera).
```

A rede semântica poderia ser uma outra alternativa interessante para esta formalização, apresentada em [WOH 88], pelas características de herança de atributos e relações entre objetos que esta forma de representação do conhecimento incorpora (Veja seção 2.1).

Os sistemas especialistas podem, também, ser uma opção para transformar especificações informais em correspondentes especificações formais e, no processo, realizar as necessárias validações.

Em [CAU 88], o sistema OICSI, por exemplo, recebe sentenças da linguagem natural (especificação informal em francês) e, mediante aplicação de regras de produção para análise léxica e sintática, a gramática é avaliada (veja detalhes na seção 3.3.1). Feita a análise da gramática, a especificação informal (fatos iniciais) é formalizada através de regras semânticas de interpretação. As sentenças são mapeadas para representações conceituais (classes de fatos) dos aspectos estáticos e dinâmicos do sistema como entidades, ações, eventos e associações entre estes aspectos. A descrição é feita mediante o uso de redes semânticas.

Uma transformação de especificação se baseia na localização de estruturas de dados e operações semanticamente equivalentes na linguagem alvo para cada construção sintática da especificação de origem, ou seja, procura mecanismos que permitam um mapeamento entre as especificações que, comprovadamente (sujeito a algoritmos de prova), reflitam a intenção modelada, sem efeitos colaterais indesejáveis.

Um processo de transformação pode ser aplicado a diferentes níveis de especificação. Além da transformação de informal para formal, sistemas transformacionais (responsáveis por automatizar um processo de transformação de especificações) podem também ser considerados para o mapeamento entre especificações formais (linguagens de programação, por exemplo). Neste contexto, os sistemas especialistas podem ser uma opção interessante para implementar os sistemas transformacionais. Apesar de não existir o aspecto de informalidade, presente no âmbito do processo de transformação de especificação informal para formal, o sistema especialista quando usado para converter especificações formais (de formal para formal) deverá conhecer a sintaxe e a semântica das duas linguagens e, através de regras de análise e interpretação (como em [CAU 88]) fazer o devido mapeamento.

Por exemplo:

Converter Prolog...

```
regra:-C1,(C2;C3;(C4,C5)),!,(A1;A2),A3
```

...para Algol

```
IF C1 AND ((C2 OR C3 OR (C4 AND C5))
THEN REGRA:- (A1 OR A2) AND A3
```

Deve haver uma ou mais regras de conversão para cada comando da linguagem de origem e a quantidade de regras a serem necessárias para efetivar o mapeamento completo entre as linguagens dependerá da complexidade das linguagens envolvidas.

Como possível aplicação de sistemas especialistas para permitir mapeamento entre especificações pode ser exemplificada a geração de programas fonte em linguagens de terceira geração a partir de uma especificação em linguagens de quarta geração (L4G).

As pesquisas na área de processamento da lingua natural, uma vez consolidadas, podem também trazer contribuições para a engenharia de software (por exemplo, estabelecer mecanismos que eliminem ambiguidades de interpretações) pois, na fase de modelagem e de formalização de uma solução, aparecem ainda dificuldades de interpretar as reais necessidades do usuário, em vista da característica de informalidade presente em sentenças da língua natural. Interpretações ambíguas conduzem a formalizações incorretas.

b) sistemas especialistas e programação em lógica como ferramentas para rápida prototipação

A prototipação é um processo de contrução de um modelo operacional do sistema de forma que, através da obtenção rápida de uma versão preliminar deste sistema, possam ser avaliados os requisitos do sistema em colaboração com o próprio usuário. Isto melhora a compreensão do ambiente do usuário, por parte do projetista, e permite que o usuário acompanhe o desenvolvimento do sistema, principalmente nas fases preliminares do projeto onde um modelo do sistema deve ser colocado em uma representação formal.

Balzer propõe um novo paradigma de desenvolvimento de software baseado na automatização (veja também o item "d" desta seção) e, em seu artigo [BAL 83a], ele discute também a utilização da prototipação como forma de validar a intenção do projeto e como forma de, automaticamente, gerar uma implementação. Nesta proposta do Balzer, após a análise de requisitos, constrói-se um protótipo que é submetido à validação pelo usuário que tem, assim, condições de julgar de forma concreta se o que está sendo projetado atende às suas necessidades. Após a aprovação, o protótipo passa por um processo de otimização para se obter o produto final.

Assim, dentro da abordagem proposta por Balzer, os sistemas especialistas podem assumir o papel de ferramentas para permitir uma rápida implementação de especificações.

Revertendo o enfoque, sob outro ponto de vista, é possível questionar como o paradigma da prototipação pode contribuir na construção de sistemas especialistas. Neste contexto, como visto na seção 2.3.1, figuram os geradores de sistemas especialistas ou "Shell" que permitem uma rápida construção destes sistemas.

Por outro lado, um "Shell" pode ser rotulado, também, como um sistema especialista para construir outros sistemas especialistas dentro da idéia de rápida prototipação e, portanto, é uma ferramenta para gerar protótipos de uma classe específica de sistemas.

Kowalski [KOW 84b], por sua vez, prega a prototipação via programação em lógica onde enfatiza que a utilização deste paradigma de programação torna mais simples e barata a confecção de protótipos rápidos visto que, com isto, é utilizada uma linguagem semanticamente equivalente a uma linguagem normalmente empregada para especificar os requisitos do usuário (DFD, por exemplo).

Esta posição do Kowalski, no entanto, é questionável visto que o paradigma de programação em lógica representa uma reformulação parcial da cultura de programação atualmente sendo praticada pelos profissionais de informática. Programadores que utilizam linguagens tradicionais (Fortran, Algol, Pascal etc) se familiarizaram com o uso da programação procedural e, para uma efetiva utilização deste novo paradigma , os conceitos associados a programação declarativa (lógica orientada a regras) devem ser, agora, assimilados. A questão do treinamento, nessa nova forma de pensar e programar, deve ser considerada quando se fala em simplificar e baratear a confecção de protótipos.

c) sistemas especialistas como mecanismos para gerenciamento de reusabilidade

A reusabilidade é uma prática da engenharia de software que está sendo muito considerada por pesquisadores e profissionais de informática. É uma das características básicas de sistemas que seguem o paradigma de orientação a objetos [AHL 90b] e está sendo agora também

discutida, com interesse, na área de inteligência artificial.

A reusabilidade permite concentrar esforços em novos aspectos do produto e, em termos de benefícios à produtividade, permite abreviar os testes e liberar funções com menos esforços e erros.

Hoje a reusabilidade é tratada sob um escopo bastante amplo onde não só programas e dados são considerados peças reusáveis mas também as especificações funcionais e o próprio projeto fisico. Dentro desta idéia, Balzer [BAL 83a] propõe construir um banco de módulos reusáveis (especificação funcional, projeto físico e programas) e um configurador inteligente (sistema especialista). Os módulos devem desempenhar funções suficientemente gerais para que possam ser úteis em uma variedade de situações. Uma vez constituida esta base de conhecimentos, que acumula uma série de conhecimentos sobre peças de software reusáveis (catálogo de rotinas e módulos), o configurador inteligente pode compor um sistema juntando os módulos e resolvendo as interfaces. Caso um módulo específico ainda não esteja disponível na B.C, este configurador pode se encarregar de introduzí-lo a partir de uma especificação fornecida pelo usuário. Para manutenção, a idéia é utilizar as técnicas de reusabilidade e reimplementação (Replay's) para alterar somente detalhes da especificação que são afetados pela modificação. Para esta tarefa, propõe também a utilização de um sistema especialista.

d) sistemas especialistas e técnicas de IA no paradigma de construção automática de sistemas

O paradigma de construção automática de sistemas [AHL 90a] pressupõe uma programação automática, ou seja, um processo de construção de programas/sistemas onde, a partir

de uma especificação informal ou semi-formal, submetida a sistemas transformacionais (sistemas de transformação de linguagens/especificações), é obtido automaticamente o código executável com a mínima intervenção humana.

O uso de sistemas especialistas no paradigma de construção automática de sistemas está associado,também, ao aspecto de formalização, anteriormente discutido no item "a" dessa seção, pois este paradigma depende, inicialmente, de uma formalização da especificação (processo de aquisição da especificação) e, em seguida, de sucessivos refinamentos (transformações) desta.

O artigo do Goldberg [GOL 86], por exemplo, analisa a abordagem transformacional (refinamentos de especificação), base do paradigma de programação automática, e a programação baseada em conhecimentos sob a ótica do projeto de algoritmos e otimização de programas onde cita o caso do uso de técnicas para seleção de estruturas do projeto PSI [KAN 81] nas quais a seleção é feita mediante aplicação de regras de escolha. O projeto PSI/SYN, descrito por Kant e Barstow em [KAN 81] é um sistema transformacional composto por uma coleção de programas especialistas escritos em Lisp.

Em [BAL 81], Balzer propõe um mecanismo de sucessivos refinamentos de especificação. Este mecanismo é implementado no sistema Glitter (GoaL-directed-jITTERed), [FIC 85] que tenciona automatizar a maior parte das sucessivas transformações de especificações que o método do Balzer sugere.

Balzer propõe também, em [BAL 85a], um ciclo de desenvolvimento de sistemas fundamentado no princípio da implementação transformacional [BAL 81]. A arquitetura proposta por Balzer [BAL 85a] está ilustrada na figura 3.1.

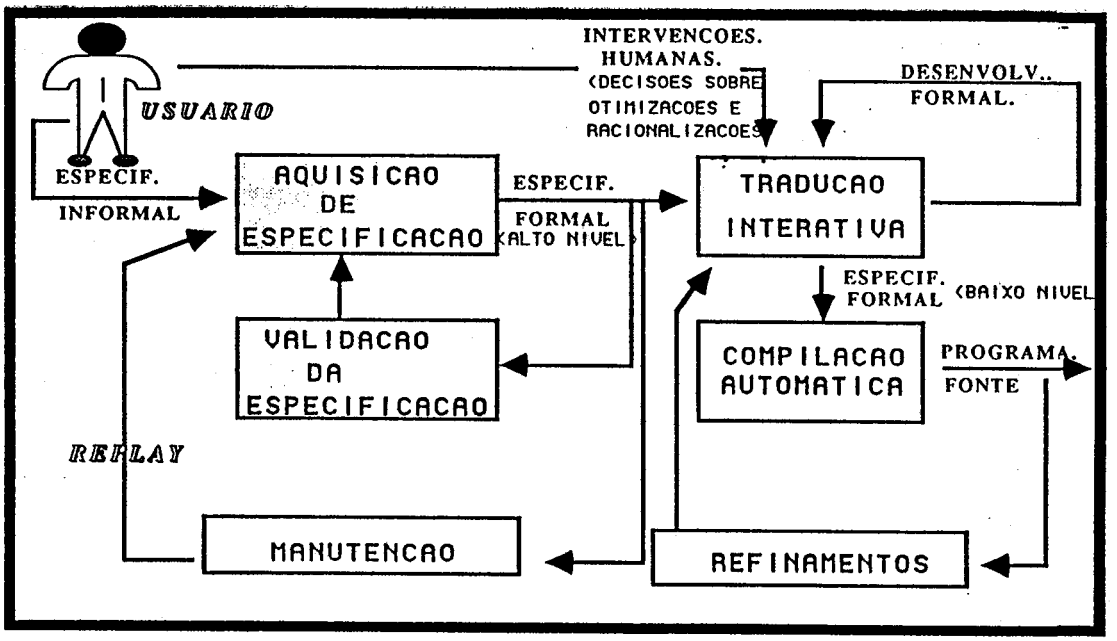


Fig.3.1 - Arquitetura de um modelo de programação automática (Adapatado de [BAL 85a])

Nesta arquitetura tem-se intervenção humana somente: 1)na fase de aquisição da especificação, onde o usuário fornece a descrição do problema em uma linguagem informal, e 2)durante a tradução interativa, alimentando o sistema com decisões sobre otimizações e racionalizações do processo de transformação para aqueles detalhes que os algoritmos de refinamento não possam ser autosuficientes. O restante do processo é automaticamente executado sob controle e gerência do sistema transformacional.

Na estrutura de desenvolvimento formal, associada à tradução interativa, é registrada a história de derivações a partir da sequência de transformações

aplicadas. Com isso permite-se que uma nova implementação seja rederivada de uma especificação modificada e somente os procedimentos envolvidos na alteração sejam reeditados mantendo os demais inalterados.

Todos os conhecimentos sobre a prática de programação são mantidos em uma base de conhecimentos, configurada a partir de fatos e regras, e gerenciado por um sistema especialista.

O conjunto de ferramentas de suporte à construção automática de software deve incorporar uma diversidade de conhecimentos referentes à, por exemplo, análise de requisitos, especificação, validação, implementação e manutenção e, portanto, Balzer considera que este assistente inteligente de suporte ao paradigma proposto deva ser um ou mais sistemas especialistas.

e) sistemas especialistas no apoio a fases específicas do ciclo de vida do software

Durante o ciclo de vida de um sistema, sob o enfoque do modelo em cascata [BOE 88], é necessário estabelecer um ambiente completo de desenvolvimento e implantação no qual um conjunto de ferramentas pode auxiliar a este, como suporte a atividades específicas nas várias fases do projeto. Estas ferramentas compõem o elenco de instrumentos utilizados na engenharia de software no apoio a metodologias e técnicas para construir e manter um sistema de informações em computador.

Podem, por exemplo, ser salientadas ferramentas como:

- .editores e tradutores de linguagens de especificação;
- .ferramentas de modelagem;

.ferramentas de análise;
.gerenciadores de projetos;
.ferramentas de documentação etc.

A fase de modelagem de sistemas agrupa um conjunto das principais atividades no ciclo de desenvolvimento de sistemas pois é nesta fase que são descritas, formalmente, as principais propriedades estáticas e dinâmicas do sistema e o analista/projetista se familiariza com o ambiente do usuário.

Para esta fase, Alvares e Giraudin [ALV 89] propõem uma abordagem de sistema especialista, para condução do processo de modelagem, com o uso de uma ferramenta inteligente que oriente o usuário e valide suas tarefas na atividade de especificação do sistema. No sistema proposto, implementado em Prolog, o chefe do projeto pode configurar seu próprio método de modelagem usando um conjunto de definições do meta-modelo (base de especificação), um conjunto de restrições à linguagem de especificação (base de regras) e meta-regras de formação do método (base de meta-regras de controle do método). A partir deste aparato, gera uma base de conhecimentos sobre um método particular. Com isto pode adaptar a ferramenta às necessidades específicas e à abordagem particular de uma equipe de analistas. Após a configuração, um módulo de atualização e controle da especificação permite que a equipe de projeto comece a modelar o sistema interagindo com uma base de fatos de especificação da aplicação.

Cauvet, Proix e Rolland [CAU 88] também apresentam um sistema especialista para modelagem conceitual de sistemas. A ferramenta faz uma tradução da descrição realizada em linguagem natural (mediante uso de regras de análise e interpretação) levando a uma rede descritiva (rede semântica) de todos os elementos do esquema conceitual. Usa regras de projeto (estruturação, validação e diálogo) para completar e transformar a rede

descritiva.

Uma ferramenta de suporte à fase de análise aparece descrita no artigo do Puncello [PUN 88]. O ASPIS (Application Software Prototype Impelentation System) é um sistema especialista que considera os conceitos da análise estruturada para definição das funções e Diagramas E/R para representar os dados e visa fazer uma transformação de especificação informal para uma linguagem formal RSL. Em seguida, os axiomas da linguagem RSL são convertidos para código Prolog.

A arquitetura do ASPIS considera a utilização de quatro sistemas especialistas básicos:

- .assistente de análise (manipula conhecimentos sobre o método e sobre o domínio da aplicação);
- .assistente de prototipação;
- .assistente de reusabilidade de especificações;
- .assistente de projeto (manipula a mesma BC do assistente de análise).

O conhecimento sobre o domínio da aplicação é explorado pelo assistente de análise através do conhecimento sobre o método e está organizado em três subconjuntos de redes semâticas:

- .Rede para representação dos diagramas (documentos) da análise estruturada: Cada diagrama e cada componente do diagrama, (processos e fluxos) são representados, na rede semântica, como objetos separados que se interligam refletindo a estrutura dos modelos da análise estruturada. Um processo (box) pode ser componente de um diagrama (boxlist) ou ser origem de outro diagrama (refinamento). Um fluxo (arrow) tem origem e tem destino.

.Rede para representação dos conceitos do domínio: Mantém as possíveis combinações entre funções básicas do sistema e os dados ou as possíveis alternativas de refinamentos de funções e de dados.

.Rede para representação do conhecimento sobre procedimentos de análise: Cada fase do método de análise e cada conhecimento heurístico da aplicação (por exemplo, um modelo de controle de acesso ao sistema) são representados por uma classe na rede semântica. As classes representativas de fases da análise estão interligadas conforme a sua sequência lógica no método. As classes representando conhecimento sobre o domínio da aplicação são especializações de alguma classe do tipo fases da análise.

Além do ASPIS, outra ferramenta para a fase de análise é o Sys-Aide apresentado por Shemer em [SHE 87] e também o "Analist" apresentado por Stephens e Whitehead em [STE 85].

Para a fase de programação aparecem os trabalhos de Barstow. O livro do Barstow [BAR 79] descreve com detalhes os experimentos de investigação da construção de programas baseada em conhecimentos. A idéia básica contida nos experimentos de Barstow, e relatada nesse livro, é representar toda a experiência que os programadores possuem a respeito da atividade de programação na forma de fatos e regras para que possam ser submetidas a uma ferramenta de construção automática dos programas (sistema PECOS no projeto PSI/SYN). Barstow alerta, no entanto, em [BAR 85], que no processo de elaboração das regras de representação do conhecimento (base para um ambiente de programação automática) deve haver uma preocupação com a racionalização do processo evitando redundâncias na especificação, ou seja, evitar que a mesma regra seja especificada em

contextos diferentes com nomes diferentes. O autor salienta que é preocupante o número de regras em constante expansão em uma base de conhecimentos e, portanto, propõe a busca de uma convergência das regras nas diferentes áreas do conhecimento. Propõe o estabelecimento de padrões de definição de regras que evidenciem a prática de reusabilidade.

Para gerência e acompanhamento de projetos de software, Lin e Levary [LIN 89] apresentam um sistema especialista, denominado HESS, onde o processo de desenvolvimento de software é assistido pelo computador. Através desta ferramenta, composta por um simulador do ciclo de vida (para experimentação de modelos em confronto com procedimentos e ações de gerência e testes de fatores ambientais), um módulo para apropriação da especificação, um módulo para fornecer as explanações (recomendações para o processo de desenvolvimento) e um SGBC, o projeto recebe suporte em termos de planejamento, gerência e controle.

O sistema se baseia na aplicação de regras que caracterizam cada fase do projeto.

Por exemplo:

Regra 1
SE "número de páginas de documentação previstas para o projeto preliminar é A"
ENTÃO
"previsão para o projeto detalhado é A+Constante_tipo_projeto"

Regra 2
SE "codificação for feita na linguagem L"
E "treinamento será via empresa especializada"

ENTÃO

"tempo previsto para treinamento é de Tmin(Linguagem,Tipo_treino)"

Uma ferramenta inteligente para diagnose de esquemas conceituais é proposta por Wohed [WOH 88]. O sistema representa, conforme visto anteriormente nesta seção, o esquema conceitual (que vem modelado na linguagem gráfica Simol) através de predicados Prolog que descrevem o esquema em termos de tipos de dados, entidades, eventos e atributos. Os predicados Prolog são então passados para o sistema especialista de Diagnose que se encarrega de validar o esquema conceitual aplicando regras (regras de produção) de sintaxe e de semântica para o Simol, regras de domínio da aplicação e regras de controle de qualidade (por exemplo, verificação de completeza, verificação de sinônimos, etc) da metodologia usada.

Exemplo de regras usadas no sistema:

```
"comentário sobre a regra"
Id  R9a
Rule
  If  Entity is Missing_id_attributes
  And Entity Violates_rule R8a
  And Superclasses Of Entity Is_equal_to  None
  Then Entity Violates_this_rule
```

Realizada a validação, o sistema emite um relatório de erros detectados no esquema fornecido e um conjunto de sugestões de incremento potencial deste esquema.

f) computadores de quinta geração:a programação em lógica no papel da engenharia de software

Com o surgimento da proposta dos japoneses para o

projeto de computadores de quinta geração, baseados em uma arquitetura altamente paralela, a tecnologia de IA passou a merecer a atenção no que se refere a aplicação de conceitos e técnicas para concepção de ferramentas utilizadas nas mais diversas áreas da informática (hardware, comunicação de dados, banco de dados, etc.).

Neste contexto, Kowalski [KOW 84b] propõe uma fusão do modelo convencional de contrução de sistemas, onde as aplicações são geradas a partir das práticas da engenharia de software, com o modelo proposto para a nova geração de computadores onde a programação em lógica se caracteriza como instrumento para geração de aplicações em IA. No modelo alternativo de construção de aplicações (nova tecnologia de software), a programação em lógica passa a ser usada tanto para sistemas de IA como para permitir a implementação de aplicações tradicionais. Kowalski propõe que a programação em lógica seja usada na análise de requisitos do usuário e que o produto desta análise possa ser executado diretamente pelo computador (veja também item "b" desta seção).

O quadro abaixo ilustra esta fusão:

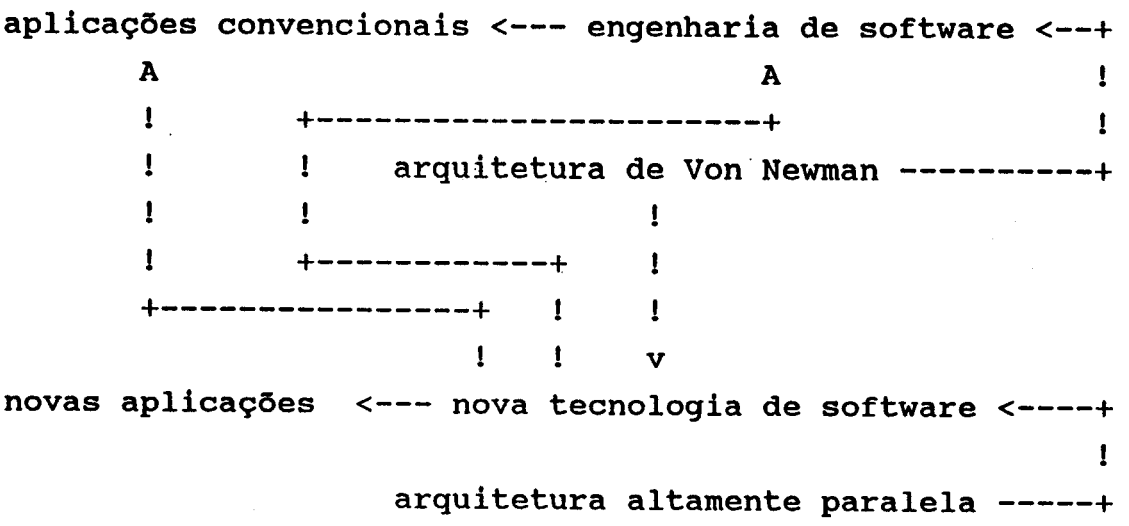
visão convencional:

```
aplicações convencionais <--- engenharia de software <--+
                                                        !
                          arquitetura de Von Newman -----+
```

visão japonesa:

```
aplicações IA <---- programação em lógica <-------------+
                                                        !
                          arquitetura altamente paralela -----+
```

visão alternativa:



Na visão alternativa proposta por Kowalski, as "novas aplicações" estão no âmbito dos usos típicos da inteligência artificial que necessitam manipular uma B.C e aplicar mecanismos de inferência e a "nova tecnologia de software" configura o uso da programação em lógica como base para a construção de sistemas.

Esta é uma questão realmente muito polêmica. Será que a programação em lógica, as técnicas de I.A e os sistemas especialistas são a solução para todos os problemas da engenharia de sotware? Existe toda uma cultura gerada na área de engenharia de sotware que certamente não pode ser descartada com o surgimento de alguma idéia revolucionária. Cabe avaliar como as tecnologias podem se complementar, uma às outras.

g) modelos de dados baseados em conhecimentos e sistemas de banco de dados especialistas

Este tema está associado a investigação de ferramentas, técnicas e mecanismos que tornem os banco de dados agentes "ativos" (com capacidades dedutivas) e que suportem aplicações de IA, e representa a convergência das tecnologias de banco de dados e inteligência artificial na

intenção de aproveitar o acervo tecnológico criado, ao longo do tempo, nestas duas áreas para a construção de um ambiente unificado (BD + IA).

O artigo do Kerschberg [KER 90] discute o assunto e descreve algumas possibilidades de incorporar conhecimento e técnicas de I.A. aos banco de dados gerando, assim, BDs especialistas.

Cita, por exemplo:

a) sistema especialista fracamente acoplado a um BD, cada qual mantendo sua funcionalidade e com interfaces bem determinadas (por exemplo, um sistema especialista com interface para consultas SQL);

b) mecanismos de acesso e manipulação de banco de dados incorporados aos sistemas especialistas (extensão nos sistemas especialistas);

c) capacidades de raciocínio, para inferência de conhecimentos, incorporadas a um S.G.B.D (extensão nos S.G.B.D);

d) sistema especialista fortemente acoplado a um BD para especificação, gerência e manipulação de BD integrado, baseado em conhecimento (dados + conhecimentos integrados);

Uma tentativa para integração destas tecnologias, dentro da alternativa "d", é o KDM (Knowledge Data Model) que é um banco de dados semântico que proporciona uma visão orientada a objetos para os dados e para os conhecimentos. O KDM permite modelar primitivas incorporadas em modelos semânticos, em modelos orientados a objetos e nos formalismos da I.A. para representação do conhecimento.

Dayal, Buchmann e McCarthy também discutem o tema em [DAY 88]. Eles sugerem o HIPAC como um modelo de dados estendido que inclui construções para representar regras. Neste modelo, as regras são tratadas como objetos, segundo o contexto do paradigma de orientação a objetos, e configuram regras, que denominam de Ação-condição-evento, para generalizar mecanismos (asserções, gatilhos, procedimentos de BD, etc) para o suporte a funções de um S.G.B.D ativo como:

-controle de integridade;
-controle de acesso;
-gerência de dados derivados;
-inferência (raciocínio), etc.

3.1.2 Horizonte de Investigações

Na seção 3.1.1 foram enumeradas algumas aplicações práticas da IA no contexto da engenharia de software com base nas idéias sintetizadas a partir das publicações que analisam o assunto.

A intenção não foi a de esgotar completamente o assunto nos breves comentários realizados sobre cada aplicação relacionada. Estes temas podem ser, de forma agrupada ou individualmente, mais profundamente explorados em futuras pesquisas, gerando publicações (artigos, trabalhos individuais, relatórios etc.) e inclusive eventuais ferramentas.

O item "b" (prototipação), por exemplo, pode ser avaliado como uma especialização do tema abordado no item "e" (fases do ciclo de vida).

O item "a" (formalização), por sua vez, pode ser explorado em conjunto com o item "d" (programação automática) visto que a formalização é peça fundamental

para o processo de aquisição da especificação, fase inicial no paradigma de programação automática (Figura 3.1).

Com relação a este último item, dentro do modelo de programação automática, o estágio de aquisição de especificação, com certeza, deve receber uma atenção especial por parte das pesquisas considerando que é deste processo que será obtido uma descrição formal que reflita as reais intenções do sistema a partir de uma especificação informal feita por algum mecanismo linguístico ou gráfico. É este estágio que garante a legibilidade, a completeza, a correção, a verificabilidade e outras qualidades de uma boa especificação. Para este estágio, as técnicas de IA podem ter uma efetiva contribuição e as investigações devem direcionar as atenções em, por exemplo:

.como extrair da IA suporte semântico para resolver ambiguidades;

Fonte de consulta:

. processadores de língua natural;

.como combinar as formas de representação do conhecimento (regras, frames, procedimentos etc) para formalizar uma especificação que seja, ao mesmo tempo, legível, completa e suceptível ao mecanismo de transformação de especificações com o mínimo de traduções possíveis;

Fonte de consulta:

. formalização de bases de conhecimentos;
. especificações formais;

.como extrair da IA algum suporte para validação do comportamento da especificação;

Fonte de consulta:

. execução simbólica de especificações (prototipação);
. provadores de teoremas;

Para o estágio de tradução interativa, também um possível alvo da aplicação dos sistemas especialistas, devem ser analisados os aspectos de, por exemplo:

.como gerar linhas de programa a partir da especificação;

Fonte de consulta:

. regras semânticas de interpretação e tradução;

. compiladores;

.como evitar reedição de especificações em fases de manutenção;

Fonte de consulta:

. bases de conhecimentos históricas;

. reusabilidade de especificações;

. aprendizagem;

.como selecionar estruturas de dados para a especificação a ser gerada;

Fonte de consulta:

. regras de seleção;

. mecanismos de inferência.

Estas perspectivas comprovam que o horizonte de pesquisa, para as aplicações discutidas na seção anterior, é bastante amplo e, portanto, necessita de um maior aprofundamento no assunto.

3.2 Tendências

Em termos de perspectivas para futuros trabalhos, dentro do escopo de sistemas especialistas, Altenkrueger [ALT 90] acredita que as investigações focalizarão temas

como:

- .interfaces inteligentes em diferentes níveis (menu, linguagem natural);
- .métodos avançados de aquisição de conhecimento:
  - -aquisição direta a partir de uma literatura especializada (processamento da linguagem natural);
  - -o sistema descobrindo novos conhecimentos a partir de suas próprias consultas;
  - -em períodos de ociosidade, o sistema procurando resolver contradições (análogo ao subconsciente humano durante o sono)

Estas perspectivas podem também ser estendidas para uma aplicabilidade dentro do contexto de sistemas especialistas na engenharia de software.

Aperfeiçoando, por exemplo, os sistemas especialistas para tratamento de interfaces inteligentes, os avanços obtidos neste sentido podem, também, ser aplicados para melhorar a comunicação com/entre ferramentas inteligentes concebidas para apoio a fases específicas do ciclo de desenvolvimento de software.

Avanços no processamento da linguagem natural podem trazer contribuições para ferramentas que façam transformações de especificações informal para formal dentro do contexto de programação automática.

O processo de aprendizagem dos sistemas especialistas, por sua vez, pode ser explorado em fases como análise de requisitos e modelagem do ambiente dos sistema para que, a cada novo projeto de desenvolvimento de software, conhecimentos adquiridos anteriormente, de outros projetos, possam ser aproveitados pelos mecanismos de inferência.

Novas aplicações da inteligência artificial em ambientes da engenharia de software são constantemente avaliadas e, dentre as perspectivas enumeradas em [RID 86], cabe ressaltar:

- registro interativo das decisões humanas de projeto durante o desenvolvimento e utilizar este registro para dar assistência à manutenção;

- representação do conhecimento sobre programação.

- análise e identificação de metodologias e estratégias de projeto buscando representá-las e simulá-las no computador.

- captura de informações sobre requisitos (especificação, projeto e codificação) e sobre técnicas de análise e representá-las em B.C.

Com certeza, todas as fases do ciclo de desenvolvimento de sistemas serão alvos de pesquisas que considerem a aplicação de técnicas de I.A. na construção de ferramentas que deem suporte a tarefas especializadas em cada fase.

Martin afirma, em seu manifesto [MAR 83], que os analistas de sistemas procuram automatizar todos os trabalhos, exceto o seu próprio. Neste contexto, prega a automação da engenharia de software em todos os sentidos. Para este intuito, os sistemas especialistas certamente serão cogitados.

3.3 Estudo de Caso: Sistemas Especialistas na Fase de Modelagem de Sistemas

Conforme visto na seção 3.1, os sistemas especialistas vem sendo cogitados para a construção de ferramentas de suporte para as diferentes fases do ciclo de vida do software. Direcionando o foco das atenções sobre a fase de modelagem de sistemas, nesta seção é feita uma análise da forma como algumas ferramentas, concebidas com a tecnologia IA, manipulam a base de conhecimentos sobre técnicas de modelagem de sistemas e como representam estes conhecimentos.

Também é discutido o processo de aprendizagem como módulo auxiliar integrado ao sistema especialista para fins de, a partir de informações do ambiente externo, expandir sua base de conhecimentos sobre a modelagem de sistemas e sobre o domínio da aplicação.

O estudo de caso se baseia nas proposições apresentadas por Alvares e Giraudin [ALV 89] e Cauvet, Proix e Rolland [CAU 88] em suas publicações anteriormente já comentadas.

3.3.1 A representação do conhecimento

Consta na literatura analisada [CAU 88],[ALV 89] que a representação do conhecimento, nos sistemas ora em estudo, é feita da seguinte forma:

a) no sistema OICSI [CAU 88]

O sistema especialista para modelagem conceitual de sistemas de informações, OICSI, recebe uma especificação em um subconjunto de sentenças da linguagem natural (Francês), utilizadas segundo alguns critérios:

.sentenças devem estar no presente do indicativo, na terceira pessoa do singular ou plural;
.uso de metáforas são proibidas;
.sentenças que compõem o texto devem ser, tanto quanto possível, independentes uma da outra;
.sentenças que incluem inversão de sujeito em relação ao verbo não são consideradas válidas.

Esta especificação compõe os fatos iniciais que possibilitam transferir o conhecimento sobre o domínio da aplicação para o projetista.

A linguagem natural é submetida a um conjunto de regras de análise (léxicas e sintáticas), do tipo "regras de produção", para averiguar a natureza gramatical de cada palavra, verbo e proposição e verificar se pertencem à linguagem autorizada.

Exemplo:

Regra1: SE a primeira palavra de uma proposição é "when" ou um sinônimo
E o verbo da proposição expressa uma aparição
ENTÃO trata-se de uma proposição indicando tempo

Regra2: SE o verbo da proposição expressa um estado ("as to be", "to stay", ...)
ENTÃO trata-se de uma proposição indicando condição

Em seguida este conhecimento dos fatos iniciais é formalizado mediante regras de interpretação que executam a tarefa de abstração. Elas mapeiam as sentenças em representações conceituais (classes de fatos) dos aspectos estáticos e dinâmicos do sistema (entidades, ações, eventos

e associações entre eles), usando uma estrutura gramatical do tipo rede semântica. Operam, portanto, com fatos iniciais e geram classes de fatos.

Exemplo:

Regra3: SE há somente uma proposição principal, e
um sujeito e um predicado
E o verbo expressa uma posse
ENTÃO -o sujeito descreve uma entidade;
-o predicado descreve uma entidade;
-a entidade descrita pelo predicado
é um atributo da entidade descrita
pelo sujeito.

As classes de fatos são submetidas a regras de estruturação a fim de gerar os elementos do esquema conceitual, descritos através de uma rede semântica normalizada. São regras para estruturação de classes, de associações entre classes de entidades e de aspectos dinâmicos de classes de entidades e visam gerar uma versão normalizada do modelo conceitual. As classes de fatos são estruturadas através de abstrações do tipo agregação, generalização e associação. O modelo conceitual resultante é uma hierarquia de entidades, uma hierarquia de ações e uma hierarquia de eventos.

Cada regra de estruturação associa situações alternativas a uma situação inicial obtida da base de fatos. Estas regras de estruturação podem levar o mecanismo de inferência a disparar:

- regras de validação: para eliminar estruturas incorretas ou ambíguas e garantir a conformidade com as normas do modelo e a coerência semântica da representação com o fenômeno real.

.regras de diálogo: para aquisição de informações complementares, obtidas junto ao projetista, que podem auxiliar na escolha de estruturas adequadas.

b) no sistema de condução da modelagem de sistemas de informações [ALV 89]

Na arquitetura do sistema de condução apresentado em [ALV 89], o sistema especialista manipula dois tipos de conhecimentos:

.conhecimento de um método específico;
.conhecimento de um domínio de aplicação.

O conhecimento do método considera as noções de modelos, linguagens e sequência de operações ("demarche"). Estes conceitos são formalizados através de enunciados em linguagem Z. Esta linguagem permite a representação de classes de informações, denominadas conjuntos de entidades, e a definição de relações binárias (associações) entre dois conjuntos de entidades.

A partir de características informais dos modelos e uma linguagem de especificação de sistemas de informações ,na forma de enunciados Z, estes modelos são formalizados e ordenados em sequências de operações culminando na geração do método.

O sub-sistema de configuração, responsável por criar uma base de conhecimentos sobre um método personalisado, necessita, para a configuração deste método, de uma base da linguagem de especificação (conjunto de definições que fixam o quadro conceitual dos modelos e métodos, ou seja, objetos e ligações entre objetos para definir um meta-modelo), uma base de regras elementares (restrições à linguagem de especificação) e uma base de

meta-regras de controle de métodos (meta-regras para formar um método).

A base de conhecimentos do método contém dois tipos de conhecimentos: conhecimento de controle (regras elementares de gestão de cada modelo, sequência de modelos e modo de ativação das regras) e conhecimento de explicação (regras que expõem um método ao usuário).

Na arquitetura proposta para o sistema de condução, uma base de fatos com as especificações da aplicação (entidades e ligações entre elas) é mantida por um módulo de atualização e controle de especificação, ligado ao sub-sistema de condução. O funcionamento do mecanismo de condução da modelagem (tutoramento) é gerido pela base de conhecimentos do método particular, anteriormente criada e mantida pelo sub-sistema de configuração. Representar, sob a forma de uma base de conhecimentos, o domínio da aplicação a modelar é uma das metas ainda previstas para o sistema de condução.

3.3.2 O processo de aprendizagem

Como visto anteriormente, o processo de aprendizagem trata das estratégias e mecanismos, incorporados aos sistemas especialistas, que lhe dão a capacidade de modificar e ampliar a base de conhecimentos, a partir de estímulos externos, gerando mudanças de comportamento do motor de inferência em um ciclo de realimentação do processo (conhecimentos auto-alimentados).

Na presente seção, o assunto é retomado com o intuito de avaliar o estado da arte para os sistemas especialistas em análise neste estudo de caso, além de discutir algumas propostas de melhorias.

3.3.2.1 O estado da arte

Para os dois sistemas especialistas, alvos deste estudo de caso, o estado da arte no processo de aprendizagem apresenta o seguinte perfil:

a) no sistema OICSI [CAU 88]

O sistema especialista OICSI se baseia em um método específico de modelagem conceitual, sem maiores pretenções de incorporar, dinamicamente, novas técnicas e métodos. O conhecimento sobre o método é estático e predefinido.

Para o conhecimento sobre o domínio da aplicação, Cauvet, Proix e Rolland [CAU 88] não fazem nenhuma referência sobre aprendizado automático a partir do mecanismo de inferência e das regras de diálogo. Não há, também, referências a registros históricos de soluções anteriores para um reaproveitamento em um novo projeto de aplicativo.

b) no sistema de condução da modelagem de sistema de informações [ALV 89]

Alvares e Giraudin [ALV 89] consideram que um sistema de condução da modelagem de sistemas de informações deva permitir uma adapatação a diferentes formas de trabalho dos analistas e que se enquadre num modo misto de comportamento entre a condução e controle permanentes, por parte do sistema, das ações do usuário durante a modelagem e entre a liberdade total das atividades de modelagem com validações a posteriori.

Na proposta apresentada pelos autores, a intenção foi a de permitir que um gerente de projetos tenha liberdade de definir seus próprios métodos, conforme

necessidades da instalação. A idéia é que ele possa definir e afinar precisamente um método com a ajuda de regras e, durante a modelagem, verificar se estas regras são respeitadas.

O aprendizado de um método, por parte do sistema especialista, se realiza através da especificação de regras (em um processo de aprendizagem por instrução) em um módulo de configuração de métodos. Neste módulo,o chefe do projeto estabelece um conjunto de regras para formar os modelos do método e ele define a sequência destes modelos. O módulo de configuração verifica o método assim definido utilizando as meta-regras de controle e gera uma base de conhecimentos particular a este método personalizado que será utilizada, posteriormente, pelo sub-sistema de condução.

O conhecimento sobre o método, embora definido previamente, antes de uma aplicação no processo de modelagem do sistema de informações, pode ser expandido e modificado a qualquer momento mediante mudanças nas regras que o configuram. Não existe, no entanto, nenhum mecanismo que permita um aprendizado automático do tipo dedução, analogia ou indução.

3.3.2.2 Proposições

Como fundamentação teórica para a proposição de melhorias no processo de aprendizagem dos sistemas especialistas, cabe citar o trabalho de Oliveira [OLI 90].

Oliveira [OLI 90] apresenta um estudo sobre a utilização de metaconhecimento (conhecimento sobre o conhecimento) em sistemas de aprendizagem simbólica automática. Neste trabalho, o autor discute as técnicas mais conhecidas de aprendizagem, comparando suas vantagens e desvantagens. Apresenta, também, os fundamentos sobre metaconhecimento e nível meta onde salienta que um Nível

Meta é uma estrutura de representação que descreve a estrutura de representação de um determinado nível objeto. Nesta abordagem meta, um ambiente é descrito por uma base de conhecimentos (BC) que por sua vez é descrita por uma meta-BC.

Este estudo aplicado a sistemas especialistas na engenharia de software implica na construção de uma meta-BC que contenha as regras de definição de métodos que podem ser criados e expandidos dinamicamente, ou seja, uma meta-BC contém os conhecimentos sobre como descrever um determinado método. Por outro lado, um método particular, descrito a partir da meta-BC, possui regras (mantidas numa BC) que descrevem a forma como o ambiente do aplicativo (domínio da aplicação) deve ser descrito. Sob esta abordagem, o conhecimento sobre a engenharia de software fica representado em três níveis:

- .Meta-BC: regras para definir métodos.
- .BC sobre o método particular: regras do método que indicam como descrever uma aplicação.
- .BC sobre o domínio da aplicação: conhecimento relativo à aplicação.

Partindo deste pressuposto, o processo de aprendizagem nos sistemas estudados poderia receber alguns incrementos. Como proposições podem ser mencionadas:

a) referente a aprendizagem em relação a conhecimentos sobre o método

Neste contexto seria interessante incorporar mecanismos de aquisição dinâmica de conhecimento sobre o método e técnicas de modelagem, à semelhança do apresentado em [ALV 89], a fim de que estes possam ser flexibilizados, com liberdades de expansão do método existente ou incorporação de novos métodos. Como justificativa desta

proposição, pode ser argumentado que uma metodologia de modelagem e projeto muitas vezes deve ser adaptada às características de uma determinada instalação e deve haver uma evolução gradativa incorporando novas expectativas.

Como incremento deste mecanismo proposto, pode-se pensar também em incorporar facilidades que permitam um aprendizado automático a partir de situações semelhantes detectadas anteriormente em outros métodos mantidos na base de conhecimentos.

b) referente a aprendizagem em relação a conhecimentos sobre o domínio da aplicação

A solução adotada em [LIN 89], onde a base de conhecimentos mantém registro de soluções (recomendações) dadas a determinado problema apresentado (vetor de entrada), parece ser bastante atrativa. Esta idéia pode ser estendida para uma aplicação no reaproveitamento de especificações de componentes do sistema aplicativo, dentro dos princípios de reusabilidade, numa intersecção com alguns pressupostos do paradigma de orientação a objetos. Por exemplo, considerando uma situação análoga, registrada numa base de conhecimentos histórica, onde foi diagnosticada uma solução na qual utilizou-se um conjunto de objetos devidamente configurados, os mecanismos de aprendizagem automática podem aproveitar este conhecimento para inferir novos conhecimentos para a situação atual, após uma confirmação por parte do usuário/projetista.

Exemplo:

Situação:

--> manipulação de atributos chave em documentos.

Solução anterior:

--> um atributo chave de uma entidade deve ser manipulado, em um documento de entrada de dados, acrescido de um dígito de controle.

3.3.3 Análise comparativa

Confrontando as peculiaridades dos dois sistemas estudados, a partir das informações extraídas da literatura, pode-se sintetizar as principais características no seguinte quadro comparativo:

I)Manipulação do método de modelagem

1. O método e a linguagem de especificação

[CAU 88]: Método estático e predefinido.

[ALV 89]: Método configurado dinamicamente (módulo de configuração que manipula uma meta-BC) a partir de uma linguagem de especificação configurada dinamicamente (módulo de manut. da ling.) e formalizada em Z.

2. A representação do conhecimento

[CAU 88]: Conjunto de regras para análise (léxica e sintática), interpretação, estruturação, validação e diálogo.

[ALV 89]: Conjunto de regras de controle (modelos, sequência de modelos, modo de ativação das regras) e de explicação. Para o sistema, um método é a organização de modelos sob a forma de sequência de operações.

3. O processo de aprendizagem

[CAU 88]: Por implantação direta.

[ALV 89]: Por implantação direta e por instrução.

II) Manipulação do domínio da aplicação

1. A Linguagem de especificação

[CAU 88]: Subconjunto da linguagem natural (francês).

[ALV 89]: Diálogo assistido (tutorado) com base no método particular configurado.

2. A representação do conhecimento

[CAU 88]: Fatos iniciais (regras de produção), classes de fatos (rede semântica) e elementos do esquema conceitual (rede semântica normalizada).

[ALV 89]: Base de fatos contendo fatos que descrevem entidades e ligações entre entidades.

3. O processo de aprendizagem

[CAU 88]: Por instrução e implantação direta.

[ALV 89]: Por instrução e implantação direta.

3.4 Aplicação Prática: Uma Ferramenta, Baseada em Conhecimentos, no Apoio à Integração de Visões durante a Modelagem com Redes de Petri - Uma proposta de P.E.P

Esta seção descreve uma proposição de um plano de estudo e pesquisa (PEP) para a caracterização de uma ferramenta, baseada em conhecimentos, que apoie o processo de integração de visões na modelagem com Redes de Petri.

O texto deste PEP é um estudo preliminar sobre a utilização de sistemas especialistas como alternativa para sanar um problema específico no uso de Redes de Petri. As idéias colocadas nesta proposta tiveram origem nas dificuldades normalmente enfrentadas, por um projetista de sistemas que utiliza Redes de Petri para modelagem, quando da definição das fronteiras entre visões.

Uma ferramenta de apoio, neste caso, pode vir a otimizar e padronizar o processo de integração e/ou individualização destas visões evitando, assim, que a tarefa dependa exclusivamente da experiência e senso crítico do projetista.

A intenção aqui é a de relatar superficialmente a idéia, expondo o problema e sugerindo uma possibilidade de solução. Os detalhes de uma eventual implementação, como especificação das interfaces, estruturas de representação do conhecimento, mecanismos de inferência, etc, não foram alvos deste estudo.

Caso venha a ser considerada por uma futura pesquisa, torna-se necessário estabelecer um mapeamento do ambiente descrito pela Rede para uma forma adequada de representação do conhecimento (determinar como representar os modelos) e prover um detalhamento nas características básicas da ferramenta proposta.

3.4.1 Motivação

A modelagem conceitual de um sistema é etapa vital no ciclo de desenvolvimento de uma aplicação pois é, a partir dela, que o analista/projetista do sistema se familiariza com o ambiente do usuário absorvendo os conhecimentos necessários mediante uma descrição completa das principais funções que deverão ser supridas pelo novo sistema.

Neste contexto, existe um interesse crescente pela comunidade de informática no uso de metodologias e ferramentas que auxiliem a realizar mais rapidamente, e de forma mais precisa, esta tarefa de "fotografar" o sistema em questão.

Dentre o elenco de ferramentas disponíveis para esta atividade, figuram as Redes de Petri que estão despertando grande interesse por parte de pesquisadores. Isto porque, na modelagem conceitual, além do embasamento formal e da representação gráfica de fácil assimilação que caracterizam este método de especificação, oferecem a possibilidade de descrever tanto as propriedades estáticas como também as propriedades dinâmicas de um sistema. Com isto passam a despertar interesse também na área de automação de um ambiente de escritório, onde é relevante, além da modelagem do sistema de informações, também uma descrição do sistema sócio-técnico.

Quando é feita a descrição completa de um sistema de informações, em muitos casos, distintas áreas de interesse passam a merecer atenção por parte do analista e, com isto, necessitando a geração de vários modelos específicos representando, cada qual, uma área sendo estudada. Aparece aí o conceito de visões da aplicação.

Diferentes visões, representadas por modelos parciais, podem ser construídas com redes, focalizando

áreas de atuação distintas para o sistema [RIC 85] [VOS 86].

As visões podem, por exemplo, caracterizar áreas departamentais de uma organização ou então, como visto em [VOS 86], descreverem perspectivas diferentes como informações, recursos e pessoal.

Seja qual for a finalidade de se usar visões para particularizar determinados aspectos do sistema, em algum momento pode haver a necessidade de integrar todas elas, como ilustrado na figura 3.2, ou então a partir do modelo completo, individualizar uma visão para fins de uma aplicação específica.

Surge aí a questão de como otimizar este processo de integração/individualização de visões e que mecanismos usar para esta tarefa muitas vezes trabalhosa e sujeita a erros.

O artigo do Voss demonstra que esta etapa pode vir a ser muito onerosa e minuciosa, exigindo do projetista um alto grau de concentração e cuidado, principalmente se cada visão for concebida por uma pessoa ou grupo diferente.

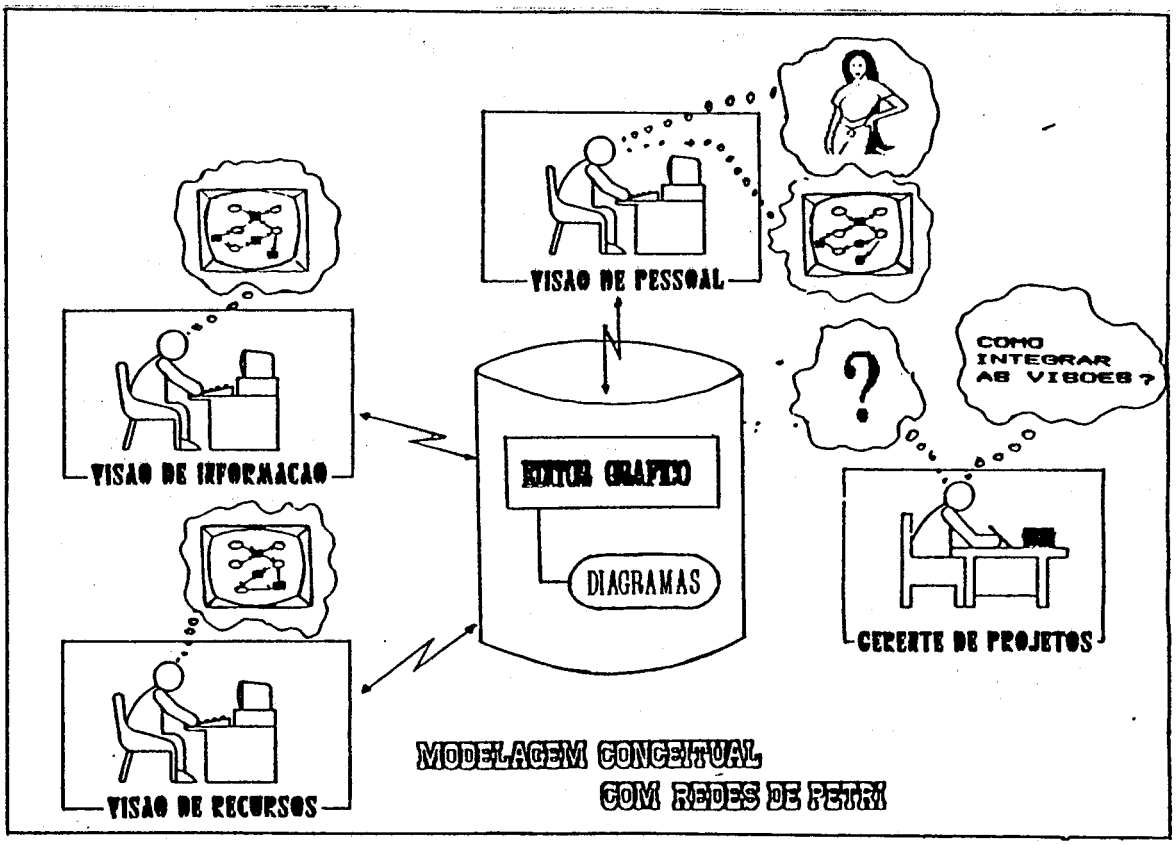


Fig. 3.2 - Uso de visões na modelagem com Redes

3.4.2 A Proposta de Pesquisa

Considerando o fato de que os modelos parciais, representativos das diferentes visões sobre o sistema aplicativo em desenvolvimento, serem eventualmente construídos por pessoas ou equipes distintas, uma das primeiras exigências impostas por uma ferramenta automatizada de apoio à integração que venha a ser utilizada é o estabelecimento de um conjunto de regras de padronização a serem aplicadas às especificações da rede.

Este conjunto de regras irá normatizar o desenvolvimento dos modelos. Todos modeladores de sistemas serão submetidos ao mesmo padrão de concepção das redes que descrevem a visão. Desta forma, cria-se um estilo de modelagem comum e evita-se as distorções ou divergências de conceitos.

Entre o conjunto de regras de padronização pode, por exemplo, ser questionado:

- O que é importante descrever?
- De que forma representar o objeto a ser descrito (canal ou agência)?
- Qual a terminologia adotada?
- Como identificar unidades funcionais vizinhas a outras visões e como impor validações especiais de combinação sobre estas unidades?

Procedimentos de confronto das especificações contra as regras de padronização podem ser incorporadas a um editor gráfico especializado, como ilustrado na figura 3.3, que irá garantir uma modelagem uniforme.

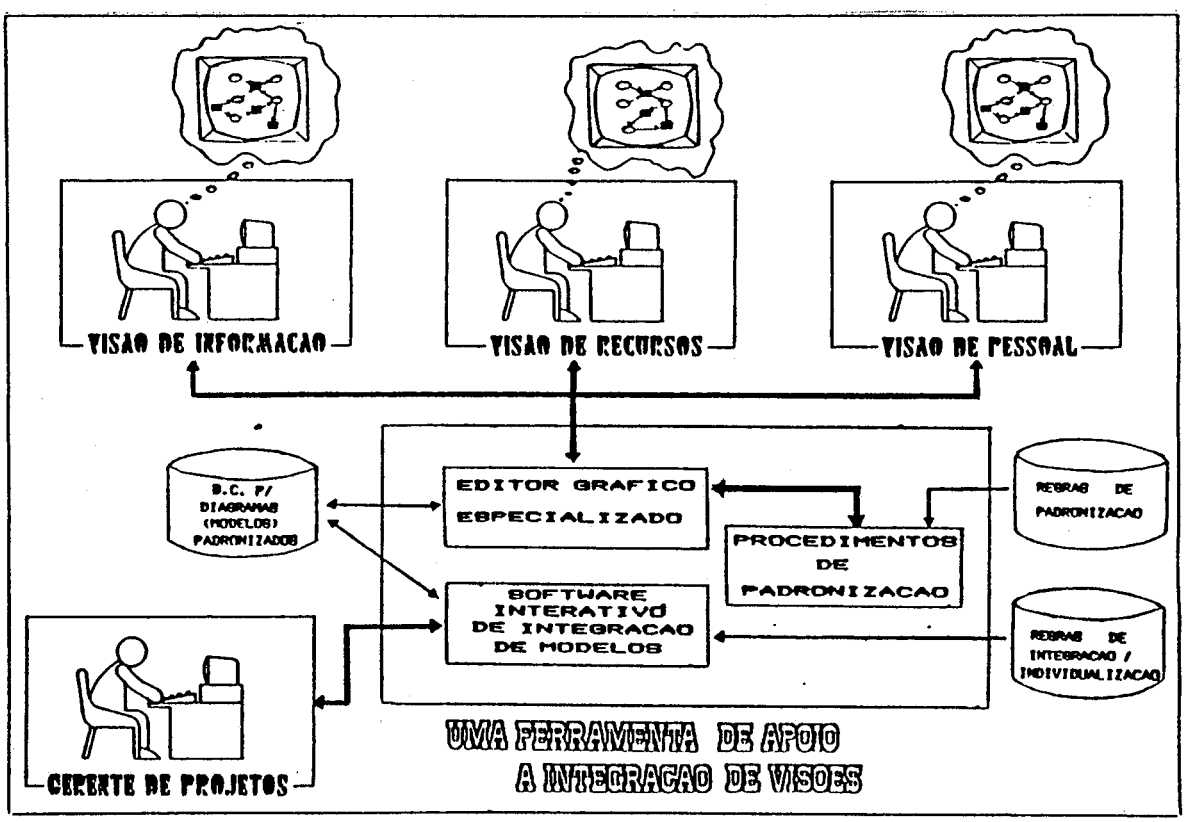


Fig. 3.3 - Uma ferramenta de apoio à integração de visões

Este mesmo editor, após validada a especificação, manterá uma Base de Conhecimentos contendo os modelos padronizados gerados que irão representar as diferentes visões do sistema.

Como ferramenta de apoio à integração, propõe-se a construção de um software interativo que, a partir da B.C. dos modelos, passa a auxiliar o projetista do sistema a integrar as diversas visões. Para tal, irá valer-se de um conjunto de regras de integração que indiquem como são mapeadas as interfaces entre as visões e como é feita a sobreposição das redes parciais.

O software se apoiará neste conjunto de regras mas deverá, por outro lado, ser flexível o suficiente para permitir que o projetista dite novas regras de integração (ajuste da sobreposição das visões). Deve ser capaz de realizar uma integração automática, segundo as regras estabelecidas, deixando para o projetista somente a tarefa de resolver eventuais conflitos de integração ou de interagir em reformulações da integração a partir de regras adicionais de "casamento" das redes.

A proposta desta pesquisa, portanto, é a de analisar todo este ambiente e estabelecer as características básicas da ferramenta de apoio a integração aqui sugerida.

4. CONCLUSÕES

Esta monografia procurou relatar, através de um estudo abrangente, o estado da arte e algumas perspectivas do uso de sistemas especialistas e da tecnologia de IA nos ambientes de desenvolvimento de sistemas de informações.

O presente trabalho foi motivado pela necessidade de avaliar o potencial dos sistemas especialistas quando aplicados a área de engenharia de software. Pretende-se, a partir do resultado obtido deste estudo, detectar alguns pontos de intersecção entre as áreas de inteligência artificial, engenharia de software e banco de dados e linguagens orientados a objetos a fim de que possam ser considerados em pesquisas posteriores no âmbito da construção automática de sistemas.

A fundamentação teórica para a tecnologia de IA foi também alvo de estudos neste trabalho e mostrou que os sistemas especialistas estão despertando interesse nas mais variadas áreas do conhecimento humano. Constatou-se que o conceito de "Shell", introduzido a partir do desenvolvimento de sistemas que utilizam técnicas de IA, aparece como um novo paradigma de construção de sistemas. Através deste gerador de sistemas especialistas pode-se rapidamente desenvolver protótipos mediante a simples indicação, com o auxílio de uma linguagem lógica, das regras e estratégias para seu funcionamento. Este gerador de sistemas pode vir a ser uma ferramenta promissora também dentro da engenharia de software auxiliando na construção de sistemas de apoio ao ciclo de desenvolvimento dos aplicativos.

Um amplo estudo bibliográfico, referente ao uso de sistemas especialistas na engenharia de software, foi realizado e, a partir desta coletânea de publicações, foi possível consolidar um conjunto de idéias que estão sendo consideradas pelos pesquisadores deste tema. Constatou-se

que diferentes fases do ciclo de vida de um sistema podem vir a usufruir de ferramentas de apoio construídas com a tecnologia de IA. Exemplos práticos foram citados e perspectivas promissoras estão sendo lançadas pelos pesquisadores como tendências para um futuro próximo. Sem dúvida nenhuma, este é um campo que ainda pode ser amplamente explorado.

Realizou-se um estudo de caso para uma aplicação de sistemas especialistas numa fase específica do ciclo de vida do software (modelagem de sistemas) onde foi focalizado o aspecto de representação do conhecimento e o processo de aprendizagem.

Com respeito ao processo de aprendizagem foi possível constatar que, apesar dos esforços empreendidos no estabelecimento de mecanismos que permitam uma aprendizagem automática por parte dos sistemas especialistas, a implantação de alguns aspectos mentais, referentes à aprendizado e criatividade, é um dos principais obstáculos enfrentados pelas pesquisas em sistemas especialistas. Neste cenário, Oliveira [OLI 90],em seu estudo sobre a utilização de metaconhecimento em sistemas de aprendizagem simbólica automática, conclui que existe uma necessidade praticamente inevitável da presença de metaconhecimento e/ou metaraciocínio sempre que for desejado uma generalização no mecanismo de aprendizagem ou quando se pretende atribuir-lhe uma maior autonomia. É um assunto que ainda está sendo investigado dentro do núcleo básico da IA e um resultado prático destas pesquisas pode vir a ser incorporado também aos sistemas especialistas aplicados à engenharia de software.

Como recomendação, cabe registrar que todos os temas, superficialmente abordados nesta monografia na seção 3.1.1, podem ser alvos de futuras pesquisas (trabalhos individuais, por exemplo) onde seriam mais profundamente avaliados.

APÊNDICE A - Revisão Bibliográfica Comentada

Bibliografia em destaque para o tema da monografia

Neste apêndice é apresentado uma coletânea, devidamente comentada, dos principais trabalhos publicados que tem como referência, ou tema de estudo, o uso de sistemas especialistas e técnicas de inteligência artificial, principalmente com vistas a aplicações na engenharia de software. Estas publicações aparecem rotuladas com uma identificação completa e acompanhadas de anotações e comentários sobre o tema por elas abordado.

As publicações estão catalogadas em ordem alfabética por autor/data da publicação.

[ALT 90]
Altenkrueger, D.E.
KBMS: Aspects, Theory and Implementation
Information Systems 15
--/1990

Este artigo discute a tecnologia dos sistemas de gerência baseados em conhecimentos (SGBC) no confronto com a tecnologia clássica de banco de dados.

O autor salienta que os sistemas baseados em conhecimentos, juntamente com as técnicas de I.A, obtiveram popularidade em vista do projeto do Computador de Quinta Geração idealizado pelos japoneses.

Salienta que diferentes áreas influenciam os S.G.B.C:

.sistemas distribuídos;
.ambientes e ferramentas de desenvolvimento;
.programação orientada a objetos;

.programação em lógica;
.S.G.B.D;
.sistemas de informações;
.sistemas especialistas;
.representação do conhecimento;

O autor compara a terminologia adotada em SGBC com os termos normalmente usados na área de banco de dados:

.conhecimento .....-> dados
.aquisição de conhecimento ..-> entrada
.representação do conhecimento .....-> estrut. de armaz.
.inferência/dedução .....-> manip. de dados
.unidade de conhecimento ....-> item de dado
."Shell" (casco, embrião), sistema especialista vazio..-> sist. de BD
.SGBC .....-> SGBD

São discutidas as formas de representação do conhecimento nos diferentes paradigmas de programação (procedural, funcional, em lógica, orientada a objetos, orientada a acessos) e os mecanismos de processamento do conhecimento (controle de fluxo, estratégia de pesquisa e raciocínio). Discute, também, as propriedades de ambientes e linguagens de programação que são relevantes para determinadas aplicações.

TEMAS DISCUTIDOS:

- Sistemas baseados em conhecimentos - princípios
- Representação e processamento do conhecimento - mecanismos e quesitos

[ALV 89]
Alvares, L.O., Giraudin, J.P.
A inteligência artificial na modelagem de sistemas de informação
XXII Congresso Nacional de Informática
09/1989

Este trabalho apresenta uma abordagem de sistemas especialistas para a condução da modelagem de sistemas de informação baseada na representação formal de métodos de modelagem.

Alvares e Giraudin salientam que as preocupações básicas atuais na concepção de novas ferramentas são:

- melhorar a ergonomia da ferramenta usando interfaces gráficas ou linguagem natural;
- aumentar a coerência e a completeza através de uma maior formalização.

Para o processo de modelagem, enfatizam que:

- a concepção de sistemas é mais uma arte que uma ciência exigindo algumas qualidades especiais (imaginação, experiência) do analista;
- o resultado da modelagem depende essencialmente do conhecimento do analista;
- a qualidade da modelagem obtida é dependente da experiência e da perspicácia do analista.

Neste sentido os autores propõem uma abordagem "sistema especialista" para diminuir as limitações das ferramentas atuais mediante a utilização de ambientes de resolução de problemas baseados em uma base de conhecimentos de objetivos, tarefas e estratégias de desenvolvimento.

Os conhecimentos da atividade de um analista, especialista em modelagem, são agrupados em:

.conhecimento de análise e concepção;
.conhecimento de um domínio específico da aplicação;
.conhecimento de um método.

Neste trabalho, os autores focalizam apenas os conhecimentos sobre o método onde salientam que trata-se de um tipo de conhecimento que se caracteriza por quatro componentes: modelos, linguagens, "demarche" (sequência de operações) e ferramentas. No artigo é proposto um formalismo de representação dos três primeiros componentes mediante o uso de um subconjunto da linguagem Z.

Alvares e Giraudin apresentam a arquitetura de um sistema especialista para guiar e verificar o trabalho de modelagem de sistemas. Este sistema (sistema de condução) é composto por:

.um sub-sistema de configuração para criar uma base de conhecimentos sobre um método personalizado com funções para definir regras de controle a ativar, modos de ativação, tipo de interfaces e etapas da modelagem;

.um sub-sistema de condução que utiliza a base de conhecimentos para guiar, orientar e verificar uma atividade de especificação.

A base de conhecimentos de definição de um método particular contém:

.conhecimento de controle: regras de gestão associadas a cada modelo, a sequência de modelos e o modo de ativação das regras;

.conhecimento de explicação: para apresentar aos usuários o método particular a ser seguido no projeto.

A arquitetura funcional do sistema de condução foi implementada em Prolog.

TEMAS DICUTIDOS:

- Sistemas especialistas na modelagem de sistemas.

[BAL 83a]
Balzer, R., Cheatham, T.E. Jr. & Green, C.
Software technology in the 1990,s using a new paradigm
Computer
11/1983

Este artigo do Balzer apresenta a proposta de um novo modelo de desenvolvimento de software no qual todos os aspectos não criativos de construção e manutenção de sistemas, normalmente também atribuídos ao homem, são agora transferidos para a máquina.

Enfatiza a necessidade de pesquisas de modelos de software baseados na automatização, em vista do alto custo atribuído ao pessoal comparativamente aos outros custos associados ao desenvolvimento de sistemas.

Balzer inicia o artigo colocando como problemas do atual modelo:

a)Não existência de tecnologia efetiva para gerenciar as atividades do conhecimento que constituem o processo de desenvolvimento de software.

b)Manutenção sendo executada sobre o código fonte (a implementação) ao invés de realizá-la sobre a especificação ocasionando, assim, problemas de:

* compreensão da alteração, visto que diversas pessoas podem produzir alterações segundo critérios próprios.
* documentação da alteração

Ele apresenta o novo paradigma de desenvolvimento de software, baseado na automatização, traçando um paralelo com o modelo comumente adotado:

MODELO ATUAL	MODELO AUTOMATIZADO
.Especificação informal	.Especificação formal
.Prototipação incomum	.Prototipação padrão
.Protótipo criado manualmente	.Especificação é o protótipo
.Código validado contra a intenção	.Protótipo validado contra a intenção
.Protótipo descartado	.Protótipo resulta em implementação
.Implementação manual	.Máquina apoiando a implementação
.Código testado	.Testes reduzidos
.Manutenção no código fonte	.Manutenção na especificação formal
.Decisões de projeto perdidas	.Desenvolvimento automaticamente documentado
.Manutenção por "Patch"	.Manutenção por "Replay" (mesmos proc. do desenvolv.)

Como prerrogativas do novo modelo, Balzer enumera:

.facilidade de manutenção;
.manutenção realizada sobre a especificação;
.o usuário como analista do sistema;

.o processo de implementação rápido, realizável e inexpressivo.

No modelo de software baseado em automatização, especificações formais são passíveis de serem criadas e mantidas pelo usuário final e no processo de manutenção a própria especificação é reimplementada após as devidas revisões.

Balzer propõe, no artigo, o uso de ferramentas de suporte à implementação automática justificando através dos benefícios que estas podem trazer:

.evitar erros de transcrição pois o mapeamento "especificação para implementação" é proporcionada pela ferramenta;
.otimizações são estimuladas visto que, uma vez que as tarefas tradicionais de construção são automatizadas, o esforço humano pode ser direcionado para o estabelecimento de critérios de otimização;
.melhoria na documentação normalmente negligenciada em processos manuais;
.reusabilidade de software mediante o uso de bibliotecas.

O artigo relaciona, também, algumas características desejáveis para a ferramenta de suporte:

.manter histórico de modificações;
.localizar todos métodos para resolução das tarefas em um domínio específico;
.localizar procedimentos ou regras que atendam às especificações e regras de seleção de estruturas de dados;
.produzir requisitos, especificações, código, derivações, etc;

.proporcionar auxílio para depurações;
.localizar pontos do programa afetados por mudanças na especificação;
.localizar testes de programas passíveis de serem realizados para atender às mudanças na especificação;
.monitorar a execução do programa;
.analisar alternativas de implementação.

Este artigo do Balzer evidencia o uso de ferramentas que suportem a construção automática de sistemas onde cada vez mais evita-se a necessidade de intervenções humanas no processo.

Em virtude das sofisticadas capacidades que devem ser incorporadas a ferramentas de suporte à construção automática de software e da diversidade de fontes de conhecimentos manipulados (conhecimentos sobre requisitos, especificação, implementação, evolução, validação, análise, etc.), Balzer considera que este assistente de suporte ao novo paradigma tenha que ser um sistema baseado em conhecimento.

TEMAS DISCUTIDOS:

- Paradigma de programação automática;
- Ferramentas de suporte à construção de sistemas;
- Prototipação.

[BAR 79]
Barstow, D.
Knowledge-based program construction
Programming Languages series;6 The computer Science Library --/1979

Este livro do Barstow é uma publicação elaborada a partir da tese de doutorado do autor, em Stanford, e

descreve um projeto experimental de investigação da construção de programas baseada em conhecimentos. A essência da abordagem apresentada envolve a identificação de conceitos e decisões associados ao processo de programação e a sua correspondente codificação em regras individuais. Estas regras são então representadas em um formato próprio para o uso por um sistema de programação automático (sistema PECOS). O sistema PECOS é capaz de contruir implementações concretas para algoritmos abstratos em uma série de domínios, como: programação simbólica elementar (conceitos e classificações básicas quanto à formação de algoritmos), algoritmos de classificação (sort), teoria dos grafos, etc. A base de conhecimentos do PECOS, mantendo informações sobre diferentes técnicas de implementação, lhe permite construir diversas alternativas de implementação para cada algoritmo, em alguns casos, com significativas diferenças nas características.

A idéia básica contida nos experimentos de Barstow é representar toda experiência que os programadores possuem a respeito da atividade de programação na forma de fatos e regras para que possam ser submetidas a uma ferramenta de construção automática dos programas (implementação de algoritmos abstratos). Um algoritmo/programa é expresso em termos de construções simbólicas de alto nível (formalismo) reconhecidas pelo sistema PECOS que faz a tradução, aplicando as regras de transformação, para código Lisp implementável. Existe, portanto, uma transformação da especificação original (linguagem formal do PECOS) para uma especificação Lisp.

No início do livro, o autor apresenta um exemplo de programa aplicativo (algoritmo), mostra como é a especificação no formalismo PECOS e mostra, informalmente, como as regras devem ser aplicadas para um mapeamento para a linguagem Lisp. Nos capítulos 2 a 6 é apresentado o corpo de regras que constituem a codificação do conhecimento de programação sobre muitos aspectos da programação simbólica

elementar. São apresentadas, por exemplo, as regras para representar coleções (listas, arrays,...), mapeamentos, entrada/saída e conversões.

O capítulo 8 apresenta uma linha de raciocínio para justificar a utilização da engenharia do conhecimento no contexto da construção automática de programas/sistemas. O capítulo 9, por sua vez, descreve detalhadamente o sistema PECOS, suas principais características, as regras de representação e as estruturas de controle.

O livro do Barstow aborda a utilização da engenharia do conhecimento e fundamentos da I.A. para a concepção de ferramentas de suporte à construção automática de sistemas. Neste contexto, todos os conhecimentos e técnicas da engenharia de software podem ser representados através de um conjunto de regras e fatos. Como resultado de uma sequência de aplicação destas regras, a partir de fatos conhecidos, obtem-se uma simulação das atividades humanas de programação para estabelecer um ambiente automatizado de desenvolvimento.

TEMAS DISCUTIDOS:

- Ferramentas de construção automática de programas;
- Uso de técnicas de I.A. aplicadas à engenharia de sistemas.

[BAR 85]
Barstow, D.
On convergence toward a database of program transformations
ACM Transactions on Programming Languages and Systems
01/1985

Neste artigo, Barstow coloca em pauta a preocupação com a crescente expansão do número de regras mantidas numa base de conhecimentos e necessárias para

representar diferentes áreas de conhecimenho dentro das tarefas de programação. Na ocasião da publicação do artigo, o sistema PECOS possuia 400 regras para sintetizar a programação simbólica e o subsistema LIBRA continha por volta de 100 regras para estimar custos e estabelecer critérios de seleção de regras [KAN 81] [BAR 79].

O autor comenta que a tendência natural passa agora a ser a busca de uma convergência no conjunto de regras, onde os pontos comuns nas diversas áreas de conhecimentos passem a considerar um elenco compartilhado de regras (integração de regras na base de conhecimentos sobre programação).

Comenta também a questão de como é o processo de elaboração das regras. No caso do PECOS é utilizado um processo pragmático onde técnicas particulares de programação são consideradas, regras contruídas para estas técnicas e então adicionadas a base de conhecimentos. A idéia que o autor coloca é a de definir um padrão para a elaboração das regras o que simplificaria consideravelmente o processo. Outra prática interessante seria a de classificar e agrupar as regras segundo princípios de modularidade reduzindo, desta forma, a redundância e permitindo uma reusabilidade por diversas aplicações.

Conforme Barstow coloca em seu artigo, deve haver uma racionalização do processo de elaboração das regras. Regras estas que, segundo facções que pregam o uso de técnicas de I.A. na construção de sistemas, são a base para um ambiente de programação automática, visto que são elas que sintetizam as tarefas de programação.

TEMAS DISCUTIDOS:

- Sistemas especialistas e o problema da racionalização no uso de regras.

[CAU 88]
Cauvet, C., Proix, C., Rolland, C.
Information systems design: an expert system approach
LNCS 303 - Advances in Database Technology - EDBT 88
03/1988

Neste artigo é descrito um sistema especialista para modelagem conceitual de sistemas de informações, denominado OICSI. A ferramenta visa gerar o esquema conceitual de um sistema a partir da descrição do domínio da aplicação realizada através de construções de um subconjunto da linguagem natural (Francês).

O sistema especialista, inicialmente, faz uma interpretação da descrição feita em linguagem natural levando a uma rede descritiva como primeira versão do esquema conceitual. Como segundo passo, o OICSI usa regras de projeto para completar e transformar a rede descritiva em uma rede normalizada que descreve todos os elementos do esquema.

A base de conhecimentos, que fundamenta o sistema OICSI, inclui uma base de fatos (descreve, em diferentes níveis de abstração, os fatos do domínio da aplicação que devem ser representados em um sistema de informações) e uma base de regras.

A base de fatos possui 3 níveis de fatos:

-Fatos iniciais: sentenças expressas em um subconjunto da lingua francesa;
-Classes de fatos: descritas através de uma rede semântica que expressa entidades, ações e eventos e associações entre eles;
-Elementos do esquema conceitual: descritos através de uma rede semântica que permite, através dos nodos e arcos da rede, estruturar a representação de forma abstrata (agregação,

generalização e associação).

A base de regras inclui 5 tipos de regras:

-Regras de análise (da linguagem natural): tratam da gramática através de regras sintáticas e regras léxicas;
-Regras de interpretação: operam com fatos iniciais e geram classes de fatos;
-Regras de estruturação: operam com classes de fatos e geram elementos do esquema conceitual;
-Regras de validação: permitem eliminar estruturas incorretas ou ambíguas;
-Regras de diálogo: permitem a aquisição de informações complementares e auxiliam o projetista na escolha de estruturas adequadas.

As regras de análise e interpretação são a formalização do conhecimento de abstração de modo que se passe da percepção de fatos concretos para uma descrição em termos de classes.

As regras de estuturação podem ser:

-Regras de estruturação de classes;
-Regras de estruturação de associações entre classes de entidades;
-Regras de estruturação de aspectos dinâmicos de classes de entidades.

As regras de validação garantem a conformidade com as normas do modelo (normas de identificadores, normas de atomicidade de ações, etc.) e testam a adequação da representação escolhida com o fenômeno real (coerência semântica).

TEMAS DISCUTIDOS:

- Ferramenta, baseada em conhecimentos, para modelagem conceitual.

[DAY 88]
Dayal, U., Buchmann, A.P., McCarthy, D.R.
Rules are objects too: a knowledge model for an active object-oriented database system
in: [DIT 88], 129-143
09/1988

O trabalho descreve as investigações em um modelo baseado em conhecimento (modelo de dados estendido que inclui construções para representar regras) e orientado a objetos, denominado HIPAC. O ponto central deste ,modelo é o conceito de regras ação-condição-evento (ECA) que generaliza muitos dos mecanismos (asserções, gatilhos, procedimentos de BD e produção de regras) para suporte a funções de um SGBD ativo, como é o caso de funções de controle de integridade, controle de acesso, gerência de dados derivados e inferências.

Os autores argumentam que as regras ação-condição-evento devem ser vistas como objetos de primeira classe em um modelo de dados orientado a objetos.

A composição das regras se constitui de:

.evento: especifica operações do BD, eventos temporais ou sinais de processos arbitrários;
.condição: especifica consultas do BD;
.ação: especifica os programas;
.modos de acoplamento: descreve se os eventos, condições e ações podem ser executados em uma única transação ou em transações separadas;

TEMAS DISCUTIDOS:

- Modelo baseado em conhecimentos para um SGBDOO ativo.

[FIC 85]
Fickas, S.
Automating the transformational development of software
IEEE Transactions on Software Engineering
11/1985

O artigo do Fickas relata os esforços em estender o modelo de implementação transformacional (TI) [BAL 81] de desenvolvimento de software descrevendo um sistema que usa técnicas de IA para automatizar a maior parte das sucessivas transformações de especificações .

Fickas, neste artigo, resume o modelo de desenvolvimento transformacional de software, ou seja, modelo baseado em transformações de especificações, como sendo:

1) inicia-se com uma especificação formal P.
2) um agente S realiza uma transformação T sobre P a fim de produzir uma nova especificação P.
3) o passo 2 é repetido até que se produza uma versão de P em condições de implementação.

Fickas descreve o sistema Glitter (GoaL-directed jITTERed) como agente S para automatizar o processo de transformação de especificações dentro da ideia de modelo TI prosposta por Balzer [BAL 81] e usando como especificação formal P a linguagen Gist. Glitter foi implementado usando a linguagem Hearsay III de construção de sistemas especialistas.

O autor apresenta o sistema e ilustra o problema

da automação da transformação através de exemplo comentado: processo de envio de mensagens através de uma rede.

A tradução interativa da especificação, usando Glitter, inicia a partir de uma especificação escrita na linguagem Gist. No exemplo, aparece ilustrado a formalização dos objetos (representados por um enfoque relacional : tipos e relações), das ações (representadas através de sequências de estados e transições) e das restrições (indicando, por exemplo, conjunto de limitações impostas pelo ambiente e durante as alterações).

O artigo ilustra também, resumidamente, o diálogo interativo do usuário com o Glitter para a formalização de objetivos, estratégias, decisões de projetos e transformações estabelecidas pelo projetista do sistema a fim de automatizar uma significativa parcela das transformações dentro do contexto do modelo TI de desenvolvimento.

TEMAS DISCUTIDOS:

- Sistemas especialistas na engenharia de sistemas;
- Ferramentas de apoio à implementação transformacional;
- Transformações de especificações.

[GOL 86]
Goldberg, A.T.
Knowledge-based programming: A survey of program design and construction techniques
IEEE Transactions on Software Engineering
07/1986

O artigo apresenta uma aplicação de I.A. no desenvolvimento de software, especificamente na construção de programas a partir de uma especificação formal de alto

nível. Discute o desenvolvimento de programas através de um processo de transformação.

O autor considera que uma abordagem promissora para programação baseada em conhecimentos deve considerar uma ferramenta de suporte à automatização do processo de programação [BAL 83a], implementada usando tecnologia de sistemas especialistas da I.A.. Esta ferramenta proporcionaria um aumento na produtividade, permitindo uma rápida prototipação e reduzindo o custo de desenvolvimento e manutenção, este último dentro da idéia apresentada por Balzer [BAL 83a] em utilizar técnicas de reusabilidade e reimplementação (Replay) para alterar somente detalhes da especificação que são afetados pela modificação. No contexto da manutenção, a abrangência de uma alteração depende da granularidade de uma transformação, ou seja, o grau de efeito sobre as estruturas do programa.

Desenvolvimento transformacional normalmente considera linguagens funcionais ou lógicas e prega o uso de linguagens de largo espectro que permitem abranger grande parte das fases do ciclo de vida do sistema.

O ponto central deste artigo do Goldberg é analisar a abordagem transformacional e programação baseada no conhecimento sob a ótica do projeto de algoritmos e otimização de programas, apresentando uma coletânea de técnicas de projeto e construção de programas. Discute, por exemplo, técnicas para seleção de estruturas de dados (Ex.: aplicação de regras de escolha como no projeto PSI [KAN 81]), técnicas de representação procedural de asserções lógicas, técnicas de fusão de loops, técnicas de mapeamento via estruturas de armazenamento/via procedimentos funcionais (Ex.: como mapear uma expressão lambda calculus) e técnicas de projeto de algoritmos.

TEMAS DISCUTIDOS:

- Sistemas especialistas;
- Transformação de especificações e uso de linguagens de largo espectro;
- Prototipação;
- Ferramentas de apoio à transformação de especificações;
- Técnicas de projeto e construção de programas.

[JAR 90]
Jarke, M., Jeusfeld, M., Rose, T.
A software process data model for Knowledge engineering in information systems
Information Systems 15
--/1990

O artigo propõe um modelo de dados para processos de software como base para a engenharia do conhecimento aplicada a sistemas de informações. Este modelo pode ser visto como uma extensão da abordagem E/R em banco de dados com ênfase na orientação a processos, suporte às decisões de projeto e integração de objetos ativos heterogêneos em uma base de conhecimentos sobre processos de software.

Este modelo, ligado ao projeto DAIDA ESPRIT, foi desenvolvido tendo como base:

.definição de como representar e integrar objetos do projeto (o que), decisões do projeto de natureza não determinísticas (por que) e ferramentas de projeto (como);
.exploração de mecanismos de abstração da linguagem de representação do conhecimento CML/Telos no sentido de gerenciar a evolução do projeto de software e do ambiente que envolve este software.

Segundo o autor, a engenharia de sistemas exige muitas decisões de projeto que envolvem conhecimentos sobre requisitos funcionais e não funcionais, sobre projeto físico, estrutural e conceitual, sobre estratégias e linguagens de implementação e, principalmente, sobre o relacionamento entre todos estes níveis de conhecimento. O conhecimento usado para decisões (especialmente para manutenção e reusabilidade) exige a construção e gerência de uma ampla base de conhecimentos. Esta base de conhecimentos representa o domínio de conhecimento do sistema especialista e também as estratégias de solução do problema.

O desenvolvimento do software é visto aqui como um processo da engenharia do conhecimento a ser suportado por um sistema de gerência de base de conhecimentos (SGBC).

Na arquitetura DAIDA, um sistema de informações é visto como uma descrição em diferentes níveis considerando requisitos de análise, projeto e implementação. Estas descrições são feitas por linguagens distintas:

- .requisitos de análise (modelagem da realidade e sistema): pela linguagem CML/Telos (Conceptual Modelling Language);
- .requisitos de projeto (projeto conceitual): pela linguagem Taxis-DL (versão declarativa);
- .requisitos de implementação (programação e BD): pela linguagem DBPL (Database Programming Language).

A linguagem CML é considerada um mecanismo de representação de conhecimentos híbridos que integra redes semânticas (para a base de conhecimentos), regras de produção e frames (agrupando um conjunto de proposições de um objeto e de sua classe). A linguagem combina estruturalmente princípios de orientação a objetos (identidade de objetos, classificação, generalização e agregação) com uma linguagem de asserções predicativas.

Detalhes desta linguagem são amplamente discutidos no artigo, bem como também é ilustrada a formalização, através desta linguagem, do modelo de dados para processos de software.

O artigo descreve também a implementação de um protótipo, denominado Conceptbase (Conceptual Model Base Management Systems) que implementa o "Kernel" da linguagem CML, seu ambiente operacional e o modelo de dados proposto no artigo.

TEMAS DISCUTIDOS:

- Modelo de dados para a engenharia do conhecimento;
- Linguagem de modelagem conceitual baseada em conhecimentos;

[KAN 81]
Kant, E., Barstow, D.R.
The refinament paradigm: The interaction of coding and efficiency knowledge in program synthesis
IEEE Transactions on Software Engineering
09/1981

O artigo descreve um modelo de desenvolvimento de sistemas baseado em refinamentos, chamado PSI/SYN, para transformar especificações de alto nível em uma linguagem de implementação. A abordagem, apresentada pelos autores, pressupõe o uso de uma linguagem de especificação de alto nível, de fácil compreensão por parte do usuário e, a partir dela, produzir código executável mediante o uso de um sistema de tradução automático.

Os autores comentam que, considerando que a linguagem fonte é bastante abstrata e muitas possibilidades de implementação são possíveis, deve ser feita a escolha da forma mais adequada para chegar à implementação. O processo

de tradução deve também considerar, em caso de modificações na especificação, critérios de reimplementação onde somente a parcela do programa afetada pela modificação será alterada. É interessante que se mantenha também um histórico das implementações realizadas a partir da especificação e quais decisões foram consideradas.

No modelo PSI/SYN, a especificação original, descrita em termos de conceitos abstratos, é submetida a sucessivas sequências de refinamentos com aplicação de regras de codificação que descrevem as técnicas de implementação (subsistema PECOS) e regras de análise e pesquisa eficientes que determinam o equilíbrio custo/otimização (subsistema LIBRA), onde cada refinamento intermediário incorpora maiores detalhes de implementação que o anterior, até a obtenção do programa executável. Considerando que um conceito abstrato da especificação pode levar a diferentes situações de implementação, a decisão deve ser tomada a partir do caminhamento em uma árvore de refinamentos para localizar a implementação mais adequada. Estes critérios de seleção podem levar a escolha de uma linguagem específica de implementação. Por exemplo, a linguagem Pascal tem tipos de dados "conjuntos" mas não "listas encadeadas", enquanto que, a linguagem Lisp tem "listas encadeadas" mas não "conjuntos". Isto pode conduzir a escolha numa ou outra direção, conforme as necessidades impostas pela especificação original.

O artigo ilustra e descreve o processo de transformação,através de um exemplo de programa de consulta a um B.D. de publicações de histórias, onde detalha as alternativas de refinamentos e possibilidades de representar "palavras-chaves" em diferentes estruturas de dados (listas encadeadas, tabela hash, etc) e a sequência de regras de codificação a serem aplicadas.

O modelo de desenvolvimento de software, aqui apresentado, é uma abordagem baseada em conhecimentos com

aplicação de regras de codificação (para conversão de construções: da especificação para a linguagem de programação),regras de análise (para avaliar as possibilidades eficientes de implemenatação) e regras de pesquisa (para determinar as formas de acesso mais eficientes). Na versão apresentada no artigo, o sistema PSI/SYN é considerado um modelo experimental sem uma avaliação mais profunda de sua aplicabilidade no contexto de sistemas e programas de aplicações reais.

É uma publicação qua analisa o tema "programação automática" sob o enfoque de técnicas de I.A. (representação do conhecimento) aplicadas em um processo de transformações sucessivas de especificações para a obtenção de código executável.

TEMAS DISCUTIDOS:

- Técnicas de I.A. na engenharia de sistemas;
- Transformação de especificações;
- Ferramentas de apoio à transformações.

[KER 90]
Kerschberg, L
Expert Database Systems: Knowledge/data management environments for intelligent information systems
Information Systems 15
--/1990

O artigo coloca em pauta a questão dos sistemas de banco de dados especialistas salientando que as pesquisas nesta área procuram investigar ferramentas e técnicas que tornem os banco de dados agentes "ativos" com capacidades dedutivas e que suportem aplicações de I.A. Esta área de estudos representa a convergência de ferramentas, técnicas e conceitos advindos das áreas de banco de dados, inteligência artificial e programação em

lógica.

Para o ambiente de sistemas de banco de dados especialistas, o autor coloca algumas possibilidades:

- .um sistema especialista sendo fracamente acoplado a um sistema de banco de dados, ou seja, ambos mantém sua própria funcionalidade e se comunicam através de interfaces bem conhecidas. (Ex.: um sistema de I.A recebendo consultas SQL);
- .um S.G.B.D estendido com capacidades de raciocínio para resolver diretamente problemas que envolvem a inferência de conhecimento;
- .um sistema de programação em lógica ou um sistema de I.A de representação do conhecimento incorporando mecanismos de acesso a banco de dados e primitivas de manipulação;
- .uma interface inteligente de usuário para especificação de consultas, otimizações e processamento;
- .um "shell" (embrião,gerador) de sistema de banco de dados especialista fortemente acoplado para especificação, gerência e manipulação de bases de conhecimentos integradas onde dados e conhecimentos podem ser manipulados e atualizados.

No artigo são discutidas, também, as características do modelo de dados, baseado em conhecimentos, denominado KDM (Knowledge Data Model) e de sua linguagem de especificação KDL (Knowledge Data Language). Além disso, apresenta resumidamente as facilidades do protótipo experimental de sistema de banco de dados especialista, denominado KORTEX, onde a representação do conhecimento é baseada em regras.

TEMAS DISCUTIDOS:

- Sistemas de BD especialistas;
- Modelo de dados baseado em conhecimentos.

[KOW 84b]
Kowalsky, R.
AI and software engineering
Datamation
11/1984

Kowalsky indaga, em seu artigo, se a tecnologia de IA poderia realmente ser aplicada na engenharia de software e em particular à fase de análise. Conclui que não só pode como também, em determinados casos, o uso da IA pode resultar em análises eficientes de tal forma que evite, inclusive, a necessidade de gerar novas especificações e programas, ou seja, permite a execução de sistemas a partir da análise.

O autor propõe a utilização da programação em lógica, atualmente em uso nas aplicações da IA, também em aplicações convencionais. Para isto, propõe uma fusão da "visão convencional" com a "visão japonesa" (oriunda da tecnologia dos computadores de quinta geração) criando uma visão alternativa na qual a programação em lógica suporte tanto as novas aplicações (sistemas especialistas, processamento da línguagem natural) como, também, facilitaria a implementação de aplicações tradicionais.

A nova tecnologia de software permite representar o conhecimento explicitamente. O computador usa o conhecimento adquirido para resolver problemas através de raciocínio dedutivo de maneira que simule o raciocínio humano. Desta forma, a programação em lógica pode ser usada como uma análise de requisitos do usuário, passível de ser executada em computador. Assim, pode-se utilizar esta programação baseada em lógica nas fases iniciais do ciclo

de vida do software e os requisitos do usuário podem ser analisados e executados antes mesmo de uma especificação funcional, projeto ou programação.

Pode-se executar a especificação oriunda da análise para verificar se está de acordo com a visão do usuário e, assim, identificar eventuais erros de interpretação já nas fases preliminares do projeto.

As regras podem ser executadas como se fossem procedimentos.

Com a tecnologia de software baseado em lógica, torna-se mais simples e menos oneroso confeccionar protótipos. Isto porque, ao invés de utilizar a linguagem de implementação da versão final (linguagens de terceira ou quarta geração), a proposta é usar linguagens (lógicas) semanticamente equivalentes aos DFDs da análise de sistemas.

Além do uso de técnicas de IA em aplicações que revolucionam o ciclo de vida do software, como a produção de especificações executáveis, o autor propõe também aplicações da tecnologia de IA em:

.ferramentas inteligentes no apoio ao processo tradicional de desenvolvimento de software;
."front-ends" inteligentes;
.bases de conhecimentos para suporte ao processo de software convencional;
.sistemas especialistas que simulem atividades da engenharia de software;
.sistemas de apoio à decisão.

TEMAS DISCUTIDOS:

- Tecnologia de IA na engenharia de software.

[LIN 89]
Lin, C.Y., Levary, R.R.
Computer-aided software development process design
IEEE Trans. Soft. Engineering
09/1989

Este artigo descreve um processo de desenvolvimento de software assistido por computador a partir de uma ferramenta computadorizada inteligente que apoia o projeto de desenvolvimento em termos de planejamento, gerência e controle.

Os autores apresentam um sistema especialista para gerência e acompanhamento de projetos de software denominado HESS (Hybrid Expert Simulation System). Esta ferramenta é composta por:

.SLICS (Simulador): visa simular o ciclo de desenvolvimento de software no sentido de experimentar o modelo em confronto com ações de gerência, policiamentos e procedimentos; testar o impacto de pressupostos e fatores ambientais; prever consequências das ações de gerência; examinar sensibilidade de processos a fatores internos e externos;

.IES (Input Expert System): visa validar a compatibilidade dos componentes no vetor de entrada inserido no SLICS;

.OES (Output Expert System): visa fornecer as recomendações sobre características para o processo de desenvolvimento ou sobre modificações no processo;

.KBMS (sistema de gerência da base de conhecimentos): mantem registro do vetor de entrada do simulador e as respectivas

recomendações resultantes.

O artigo apresenta algumas regras de produção utilizadas no IES e no OES.

TEMAS DISCUTIDOS:

- Sistema especialista na gerência e acompanhamento de projetos de software.

[MEL 90]
Mellist, W.
TWAICE: a knowledge engineering tool
Information Systems 15
--/1990

Mellist, em seu artigo, discorre sobre um "shell" para sistemas especialistas, denominado Twaice.

O autor inicia sua explanação apresentando os principais problemas normalmente encontrados nos "shells" de sistemas especialistas. Cita, por exemplo:

a) formalismo limitado para representação do conhecimento: são frequentes formalismos como regras, fatos, redes semânticas e lógica. Existem restrições quanto a múltiplos formalismos e sua respectiva integração;

b) construção da estratégia de solução do problema: muitas vezes difícil de compreender ou então incompatível com o "shell" escolhido agravado pela necessidade de certos sistemas complexos necessitarem de mais de uma estratégia;

c) integrabilidade limitada: necessidade de integrar o sistema especialista com ambientes convencionais de software e outras aplicações não tradicionais como sistemas CAD;

d) dificuldades no suporte à detecção de inconsistências: em vista do grande volume de regras para aplicações que utilizam grandes bases de conhecimentos;

e) capacidade de explanação limitada: muitas vezes aquém das espectativas do usuário;

f) dificuldades no suporte ao casamento de estilos na engenharia do conhecimento: ligado a generalização ou especialização das aplicações dificultando a tarefa de casar os estilos.

Discutindo algumas características da ferramenta Twaice, o autor procurou mostrar as alternativas encontradas para os problemas acima mencionados. Por exemplo:

. formalismos alternativos (tabelas, fórmulas aritméticas, procedimentos, programas Prolog, regras usando comparação de atributos, variáveis e quantificadores, etc);
. ferramentas de suporte à depuração;
. mecanismos de explanação mais legíveis através de um gerador de linguagem natural;
. integração com ambientes de computação através de chamadas a subrotinas escritas em C e, a partir dessas, a programas Fortran, Pascal, etc;
. mecanismos de verificação de consistência;
. modelagem de diferentes estratégias de solução do problema mediante especificação do comportamento desejado (método) usando para isto o Prolog.

Todas estas características são reunidas em uma arquitetura de um ambiente integrado de engenharia do conhecimento que o autor ilustra, em seu artigo, através de um conjunto de ferramentas (editor de regras, editor de exemplos, editor de taxonomia, ferramenta de abstração, etc.).

TEMAS DISCUTIDOS:

- "Shell" para sistemas especialistas;
- Ferramentas para engenharia do conhecimento.

[PUN 88]
Puncello, P.P., et alii
ASPIS: A knowldege-based CASE environment
IEEE Software
03/1988

Neste artigo, os autores apresentam o ASPIS (Application Software Prototype Implementation System), como uma ferramenta de suporte a análise, onde combinam técnicas de engenharia de sistemas e I.A..

Esta ferramenta visa suprir a lacuna entre a fase de análise e a fase de projeto. Na metodologia apresentada, a fase de análise considera os conceitos da análise estruturada para definição das funções e diagramas E/R para representar os dados. Estas especificações informais são então traduzidas para uma linguagem formal RSL (Reasoning Support Logic). Os axiomas da RSL são transformados em código prolog.

A ferramenta apresentada suporta somente as fases preliminares do ciclo de desenvolvimento.

TEMAS DISCUTIDOS:

- Técnicas de I.A. na engenharia de sistemas;
- Transformação de especificação informal para formal;
- Ferramentas de apoio à transformação.

[RID 86]
Riddle, W.E., Williams, L.G.
Software environments workshop report
ACM SIGSOFT
01/1986

Este relatório discute os assuntos tratados no workshop sobre ambientes de software realizado de 12-15 de novembro de 1985 em Boulder,Colorado-USA. O objetivo do workshop foi o de analisar os problemas e o estado da arte nos ambientes de software. Os grupos de estudos procuraram avaliar as mudanças, previstas para o período 1985-1995, nas diversas áreas da engenharia de software.

Dentre os tópicos discutidos, foram ressaltados:

.ferramentas;
.distribuição;
.suporte a banco de dados;
.extensibilidade;
.integração;
.prototipação;
.interfaces humanas e
.inteligência artificial.

Para a tecnologia de inteligência artificial, os avanços previstos para os anos 90 se concentram nos aspectos de representação do conhecimento e modelos de projeto. Pelas previsões, a representação do conhecimento irá prestar assistência a técnicas de programação e a análise e especificação do problema. Já modelos de projeto

aperfeiçoados irão proporcionar aos usuários assistência em ambientes de desenvolvimento.

A inteligência artificial tem desenvolvido técnicas para resolução semi-automática de problemas em: representação do conhecimento, inferência, comunicação em linguagem natural, planejamento, projeto e modelagem. Uma tendência é a de utilizar estas técnicas em ambientes de desenvolvimento de software liberando o usuário de grande parte das tarefas realizadas manualmente.

Os conferencistas relacionam, como aplicações da IA nesta área:

.para modelos de projetos:
-registro interativo, durante o desenvolvimento, de decisões humanas de projeto;
-formalização do processo de projeto;
-análise e representação de metodologias e estratégias de projeto.

.para representação do conhecimento:
-captura de informações sobre requisitos;
-linguagens de especificação formal;
-representação do conhecimento de programação.

TEMAS DISCUTIDOS:

- Problemas e estado da arte nos ambientes de software.

[SHE 87]
Shemer, I.
Systems analysis: a systemic analysis of a conceptual model
Communications of the ACM
06/1987

Shemer comenta o processo de desenvolvimento de sistemas sob o espectro da análise de sistemas e questiona como a experiência das atividades de um analista pode ser representada. Conclui que as técnicas e experiências empregadas no processo de desenvolvimento de sistemas podem ser simuladas através da especificação de uma base de conhecimentos sobre análise de sistemas, regras de inferência e um mecanismo de inferência para empregar estas regras.

O autor menciona o uso de um protótipo, denominado SYS-AIDE (System Analysis Expert Aide), como assistente inteligente do processo de análise de sistemas proposto no artigo. Não apresenta, no entanto, maiores detalhes desta ferramenta.

TEMAS DISCUTIDOS:

- Sistema especialista na análise de sistemas.

[STE 85]
Stephens, M., Whitehead, K.
The analist - a workstation for analysis and design
IEEE
--/1985

O artigo descreve um protótipo, denominado Analist, para suporte as fases de análise e projeto de sistemas utilizando, para tal, sistemas especialistas e técnicas de IA. O objetivo básico foi o da investigação do uso da abordagem baseada em conhecimentos na geração de produtos comerciais.

O protótipo foi desenvolvido usando o Pascal para tratamento de gráficos e janelas e o Prolog para expressar regras para o método e para armazenar e recuperar

informações sobre a aplicação.

O usuário pode adicionar novas regras para uma eventual mudança no método de análise e projeto. A explícita representação de regras e fatos (base de conhecimentos) torna possível adicionar métodos que atendam diferentes fases ou aspectos do ciclo de vida do software.

Utiliza, também, uma interface orientada a objetos, apontando objetos e especificando as operações. É uma interface gráfica semelhante ao estilo usado no Macintosh e Apple Lisa (visualização de objetos na tela, uso do mouse, janelas, ...).

As técnicas do método são implementadas como objetos e, cada qual, consiste de regras de sintaxe (codificadas em Pascal e não acessáveis pelo usuário) e regras de semântica (implementadas em Prolog e podem ser mostradas ao usuário).

Como pesquisas futuras, o artigo coloca:

.processador de linguagem natural;
.estratégias para solução de problema;
.aprendizado (machine learning).

TEMAS DISCUTIDOS:

- Sistema especialista para suporte à análise e projeto de sistemas.

[SYM 88]
Symonds, A.J.
Creating a software-engineering Knowledge base
IEEE Software
03/1988

Symonds propõe, em seu artigo, combinar os conceitos relacionados a engenharia de sistemas com os da engenharia do conhecimento (I.A.) para gerar um poderoso ambiente CASE. Segundo ele, engenharia de sistemas permite organizar, a nível conceitual, o conhecimento relativo ao projeto do sistema através de abstrações de dados e funções. A engenharia do conhecimento, por sua vez, fornece as estruturas de dados para representar estes conhecimentos.

O autor apresenta, no artigo, o projeto CASE/MVS da IBM como produto que incorpora as noções anteriormente comentadas para desenvolver o sistema operacional MVS/XA.

Considerando que o modelo tradicional de desenvolvimento de sistemas, conhecido como modelo cascata, apresenta deficiências, Symonds propõe, para o CASE/MVS, uma abordagem que considere a engenharia de sistemas baseada em conhecimento.

O artigo comenta, também, os trabalhos de Balzer [BAL 83a] em direção a um modelo de desenvolvimento que permita um suporte automático à implementação ao qual denomina modelo operacional.

As técnicas de representação baseadas em conhecimento são oriundas das pesquisas de I.A. e as ferramentas que suportam uma base de conhecimentos possuem os seguintes componentes:

- uma representação de fatos usados para descrever informações relevantes ao domínio do problema;
- um conjunto de regras de produção que permitam estabelecer conclusões a partir dos fatos;
- estratégias de raciocínio que especificam como problemas podem ser resolvidos pela

aplicação sucessiva das regras de produção.

Symonds ilustra como os fatos da engenharia de sistemas são armazenados pelo CASE/MVS em uma base de conhecimentos (conjunto de frames compostos de uma lista de propriedades) e como é obtida a capacidade de raciocínio automático (seleção e aplicação de regras).

Para construção da base de conhecimentos é usado um gerenciador de banco de dados relacional.

A arquitetura proposta para o CASE/MVS é composta basicamente de 4 módulos, acessáveis via janelas. São eles:

.gerenciamento das especificações fonte;
.criação e análise dos modelos estáticos (especificações executáveis);
.análise dinâmica (interpretação dos modelos estáticos e emulação semântica do comportamento do hardware do 370/XA);
.geração de código.

O CASE/MVS é um projeto em Prolog que procura implementar o modelo operacional de construção de sistemas, proposto por Balzer, como um sistema baseado em conhecimentos, para construir uma aplicação específica (o S.O. MVS/XA). As tecnologias que fundamentam este projeto são: banco de dados relacionais e raciocínio automático (respostas a asserções a partir de regras que permitem extrair conclusões).

TEMAS DISCUTIDOS:

- Técnicas de I.A. na engenharia de sistemas;
- Ferramentas CASE para integrar ambientes.

[WOH 88]
Wohed, R.
Diagnosis of conceptual schemas
SYSLAB - University of Stockholm
03/1988

O relatório técnico descreve um protótipo de um sistema especialista para diagnose de esquemas conceituais. O protótipo usa um editor gráfico (RAMATIC) para modelar o esquema conceitual em uma linguagem de especificação (SIMOL) semelhante ao modelo E/R.

O editor armazena a especificação do esquema em um banco de dados (CS5) baseado em modelo de dados binários. Uma interface especial é então usada para traduzir o esquema do modelo de dados binário para predicados Prolog.

Os predicados Prolog são utilizados como entrada para o sistema especialista de diagnose do esquema. A saída deste sistema especialista é um relatório de erros e um conjunto de propostas de incremento potencial do esquema.

A capacidade de propor incrementos no esquem conceitual exige que o sistema de diagnose conheça as características de um "Bom" esquema conceitual. Para isto, a base de conhecimentos se fundamenta em:

.regras de sintaxe do Simol;
.regras de semântica do Simol;
.conhecimento sobre o domínio (regras de domínio) corrente da aplicação originado a partir do uso do sistema (domínio do sistema especialista);
.conhecimento sobre a metodologia (regras de qualidade).

A representação do conhecimento é feito através de regras de produção (condição/conclusão). A sintaxe para

estas regras é a seguinte:

"<string>" Id <ident> **Rule If** <condição>
Then <conclusão>

O artigo ilustra a base de conhecimentos do sistema mostrando um conjunto de regras utilizadas para cada tipo (sintaxe, semântica, domínio, qualidade).

Um exemplo de execução do sistema especialista de diagnose também é mostrado no artigo.

TEMAS DISCUTIDOS:

- Ferramenta, baseada em conhecimentos, para diagnose de esquemas conceituais.

REFERÊNCIAS BIBLIOGRÁFICAS

[AHL 90a] AHLERT, H.: Estudo de metodologias e ferramentas que envolvam suporte à construção automática de sistemas, TI. n. 188 - CPGCC/UFRGS, mai 1990.

[AHL 90b] AHLERT, H.: Estudo sobre banco de dados e linguagens orientados a objetos, Exame de Qualificação I - CPGCC/UFRGS, ago 1990.

[ALT 90] ALTENKRUEGER, D.E.: KBMS: Aspects, theory and implementation, Information System 15, nro. 1, 1990, pp. 1-7.

[ALV 89] ALVARES, L.O., GIRAUDIN, J.P.: A inteligência artificial na modelagem de sistemas de informações, Anais XXII Congresso Nacional de Informática - SUCESU 89, 18-22 set 1989, pp. 386-394.

[BAL 81] BALZER, R., Transformational implementation: An example, IEEE Trans. Software Eng., vol. SE-7, pp. 3-14, Jan. 1981.

[BAL 83a] BALZER, R., CHEATHAM, T.E. Jr. & GREEN, C.: Software technology in the 1990,s using a new paradigm. Computer, vol 16, no. 11 pp. 39-45 (1983)

[BAL 85a] BALZER, R.: A 15 years perspective in automatic programming, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, vol. 11, no.11, pp 1257-1267 (1985)

[BAR 79] BARSTOW, D.: Knowledge based Program Construction. Elsevier North-Holland, New York, Programming Languages series;6, The Computer Science library, (1979).

[BAR 85] BARSTOW, D.: On convergence toward a database of program transformations, ACM Trans.Program.Lang. and Systems, vol.7, no. 1, 1985.

[BIB 84] BIBEL, W.: Artificial intelligence in Europe, in: [BIB 85] e Proc. Int. Conf. on Artificial Intelligence: Methodology, systems, applications (AIMSA 84), Varna, Bulgaria, 17-20 sep 1984.

[BIB 85] BIBEL, W., PETKOFF, B.: Artificial Intelligence: methodology, systems, applications, Elsevier Science Publishers B.V., North-Holland, 1985.

[BOE 88] BOEHM, B.W.: A spiral model of software development and enhancement. Computer, Los Alamitos, 21(5): 61-72, may 1988.

[CAS 87] CASANOVA, M.A., GIORNO, F.A.C., FURTADO, A.L.: Programação em Lógica e a Linguagem Prolog, 1987 Editora Edgard Blucher LTDA.

[CAU 88] CAUVET, C., PROIX, C., ROLLAND, C.: Information systems design: an expert system approach, LNCS 303 - Advances in Database Technology - EDBT 88, Springer-Verlag, mar. 1988.

[CLO 84] CLOCKSIN, W.F., MELLISH, C.S.: Programming in Prolog, 1984, Springer-Verlag.

[COS 86] COSTA, A.C.R.: Sobre os fundamentos da inteligência artificial, Relatório de pesquisa - CPGCC/UFRGS, Porto Alegre, 1986.

[DAY 88] DAYAL, U., BUCHMANN, A.P., MCCARTHY, D.R.: Rules are objects too: a knowledge model for an active object-oriented database system, in: [DIT 88], 129-143.

[FIC 85] FICKAS, S.,: Automating the transformational development of software, IEEE Transactions on Software Engineering, vol.11, no.11, pp 1268-1277, (1985)

[FOR 86] FORSYTH, R.: Expert systems: principles and case studies, Chapman and Hall Ltd, New York, 1986.

[GEV 87] GEVARTER, W.B.: The nature and evaluation of commercial expert system building tools, Computer, may 1987, pp. 24-41

[GOL 86] GOLDBERG, A.T.: Knowledge-based programming: A survey of program design and construction techniques. IEEE Trans. Software Eng., vol. SE-12. no. 7, pp. 752-768, Jul 1986.

[HEU 88] HEUSER, C.A. Modelagem de Sistemas com Redes de Petri. Porto Alegre, CPGCC da UFRGS, 1988. (Notas de Aula).

[JAR 90] JARKE, M., JEUSFELD, M., ROSE, T.: A software process data model for knowledge engineering in information systems, Information Systems 15, no. 1, 1990, pp. 85-116.

[KAN 81] KANT, E. and BARSTOW, D.R.: The refinement paradigm: The interaction of coding and efficiency knowledge in program synthesis, IEEE Trans. Software Engineering, vol. SE-7, no. 5, Sept. 1981, pp. 458-471.

[KER 90] KERSCHBERG, L.: Expert Database Systems: Knowledge/data management environments for intelligent information systems, Information Systems 15, No.1, 1990, pp. 151-160

[KOW 84a] KOWALSKI, R.: Software engineering and artificial intelligence in new generation computing, in: Fifty Generation Computer System, North-Holland Publishing Co., 1984, pp. 39-49.

[KOW 84b] KOWALSKY, R.: AI and software engineering, Datamation, nov.1, 1984, pp. 92-102

[LIN 89] LIN, C.Y., LEVARY, R.R.: Computer-aided software development process design, IEEE Trans. Soft. Engineering,vol 15, no. 9, sep 1989.

[MAR 83] MARTIN, J.: Manifesto: Presente e futuro da informática, Compucenter Ltda., 1983

[MEL 90] MELLIST, W.: Twaice: a knowledge engineering tool, Information Systems 15, no.1, 1990, pp. 137-150.

[OLI 90] OLIVEIRA, F.M.: Um estudo sobre metaconhecimento e aprendizagem automática, Trabalho individual TI-184, CPGCC/UFRGS, Porto Alegre, jul 1990.

[PUN 88] PUNCELLO, P.P., et alii : ASPIS: A knowledge-based CASE environment. IEEE Software, pp. 58-65, Mar 1988.

[RIC 85] RICHTER, G., VOSS. K. Toward a Comprehensive Office Model Integrating Information and Resources

[RID 86] RIDDLE, W.E., WILLIAMS, L.G.: Software environments workshop report, ACM SIGSOFT - Software Engineering Notes, vol 11, no. 1, jan. 1986.

[SAN 89] SANTOS, A.P.L., GONçALVES, C.A.: Fatores que influenciam na construção de Bases de Conhecimento, Anais XXII Congresso Nacional de Informática - SUCESU 89, 18-22 set 1989, pp. 440-447.

[SCH 85] SCHNEIDER, H.M., MARQUES, M.L., COHN, P.G.: A inteligência artificial e suas aplicações na automação, SEI/Centro Tecnológico para Informática - Instituto de Automação, jul. 1985.

[SHE 87] SHEMER, I.: Systems analysis: a systemic analysis of a conceptual model, Communications of the ACM, vol 30, no. 6, jun. 1987.

[STE 85] STEPHENS, M., WHITEHEAD, K.: The analist - a workstation for analysis and design, IEEE, 1985, pp. 364-369

[STE 86] STERLING, L., SHAPIRO, E.: The Art of Prolog, 1986, The MIT Press.

[SYM 88] SYMONDS, A.J.: Creating a software engineering knowledge base. IEEE Software, pp 11-16, Mar 1988

[WOH 88] WOHED, R.: Diagnosis of conceptual shemas, SYSLAB - University of Stockholm - Report no 56, mar. 1988.

[VIC 89] VICCARI, R.M.: Ferramentas para Inteligência Artificial, (Anotaçãoes de Aula), 1989.

[VIC 90] VICCARI, R.M.: Inteligência artificial: representação do conhecimento, X Congresso SBC, 22-27 jul 1990, Vitória-ES.

[VOS 86] VOSS. K. Nets in Office Automation, Advanced Course on Petri Nets, Bad Honnef 8-19 sept 86, paper n 6.

Relatórios de Pesquisa

RP-154: "Sistemas Especialistas para a Engenharia de Software", Abril 1991.
H. AHLERT.

RP-153: "Estudo Comparativo e Taxonomia de Ferramentas de Suporte à Construção de Sistemas, Abril, 1991.
H. AHLERT.

RP-152: "IMP-MAC - Emulador de Impressora Padrão Apple", Abril, 1991.
C. DE ROSE; R.F. WEBER.

RP-151: "Em direção a um modelo para representação de aplicações de escritórios baseadas em documentos", Abril, 1991.
D.B.A. RUIZ.

RP-150: "Implementação de Sistemas de Gerência de Banco de Dados", Abril, 1991.
D.B.A. RUIZ.

RP-149: "Integração de Ferramentas no Sistema AMPLO: Crítica e Proposta de Extensões", março, 1991.
F.R. WAGNER.

RP-148: "Interface de Entrada para Teclado e Mouse", março, 1991.
J.M. DE SÁ.

RP-147: "Servidores - Guia do Usuário - Edição 1", janeiro, 1991.
A.R. TREVISAN, C. LEYEN, G. CAVALHEIRO, J.F.L. SCHRAMM, L.G. FERNANDES, P. FERNANDES, R.M. BARRETO, R. TEODOROWITSCH

RP-146: "Biblioteca de Células TRANCA regras ECP15/1", janeiro, 1991.
C. CRUSIUS, L. FICHMAN, M. KINDEL, C. MARCON, R. REIS

RP-145: "Manual do Usuário do Projeto TRANCA; v 1.0", janeiro 1991.
F.G. MORAES; M. LUBASZEWSKI, R.A.L. REIS

RP-144: "Manual do Sistema TRAMO Projeto TRANCA versão 1.0", janeiro 1991.
M.A. SOTILLE; C.E.S. SOUZA; M.G.R. ARAUJO; M. LUBASZEWSKI; R.A.L. REIS

RP-143: "PILCHA: Projeto e Implementação de um Sistema Digital Discreto dedicado ao controle de acesso direto a memória", janeiro 1991.
F. AZEREDO; L. ROISENBERG; D.A.C. BARONE

RP-142: "Ambiente para Estudo de Fractais - Relatório de Projeto", janeiro 1991.
S.D. OLABARRIAGA; F.S. MONTENEGRO

RP-141: "Técnicas para Compilação em Paralelo", janeiro 1991.
L. C. GARCIA

RP-140: "Interval Solution and Safe Starting region for nonlinear systems", janeiro 1991.
M.A. CAMPOS

RP-139: "Programação de Máquinas Pipeline Vetoriais: Compiladores, Linguagens e Estado-da-Arte, janeiro 1991.
R.M. BARRETO.

RP-138: "TAURA, Um ASIC para Terminal de Video", janeiro 1991.
J.L.A. GUNTZEL; L.O.S. FREIRE; R.P. RIBAS, D.A.C. BARONE.

RP-137: "Integrating a VHDL Dialect into the AMPLO Design Framework", dezembro 1990.
F.R. WAGNER

RP-136: "O Ambiente de Execução do Experimento em Programação Diversitária", novembro 1990.
R. CANANI; S.P. ZANI

RP-135: "Teste Prático do Circuito MPC e o seu Ambiente Técnico", novembro, 1990.
L. ROISENBERG; T.V. WAGNER; D.A.C. BARONE

RP-134: "SÍNTESE AUTOMÁTICA DE PARTES OPERATIVAS - Ferramentas resultantes da implementação do protótipo do sintetizador automático de partes operativas (SAPO), agosto 1990.
C. De ROSE; J.L.S. JÚNIOR

RP-133: "Preliminar Thoughts on Agents and their Development", novembro 1990.
A.C.R. COSTA

RP-132: "Towards a Complete Conceptual model: Petri Nets and Entity-Relationship", agosto 1990.
C.A. HEUSER; E.M. PERES

RP-131: "PC como Terminal do ED680 (Uso e Implementação)", agosto 1990.
S.D. OLABARRIAGA

RP-130: "Ferramenta para Apoio à Análise de Requisitos e Modelagem de Sistemas de Banco de Dados", julho 1990.
M.H. YAMAGUTI; S. LOH; J.M.V. CASTILHO