SBA: Controle & Automação, Vol. 1, Nº 4, pp. 322-330

MODELAGEM, ANÁLISE E PROJETO DE SISTEMAS DINÂMICOS INTEGRADOS POR COMPUTADOR

P.A.V. Ferreira, W. Fontanini, A.C. Guerra,
Centro Tecnológico para Informática - CTI
Instituto de Automação
Caixa Postal 6162
13.081 Campinas - SP - Brasil

W.C. Amaral, F.A.C. Gomide
Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica
Caixa Postal 6101
13.081 Campinas - SP - Brasil

RESUMO: Neste artigo são abordados alguns aspectos funcionais, estruturais e metodológicos que norteiam o desenvolvimento de sistemas computacionais modernos para a área de Engenharia de Controle. Procura-se fornecer uma visão abrangente do estágio atual de ambientes de programação do tipo CACE (Computer Aided Control Engineering), focalizando-se as soluções adotadas por diversos autores para aspectos como os de estruturação de dados, interface homem-máquina e utilização de sistemas baseados em conhecimento como ferramenta de suporte ao usuário nas fases de modelagem, análise e projeto de sistemas dinâmicos. No final do artigo são apresentadas as principais características de alguns sistemas CACE atualmente em uso.

Modelling, Analysis and Design of Dynamic Systems Integrated by Computer
ABSTRACT: In this paper, some of the functional, structural and methodological aspects involved in the development of modern computational systems directed to Control Engineering are approached. An attempt of furnishing an overview of the current stage of CACE (Computer Aided Control Engineering) environments is made by focusing solutions adopted by several authors in aspects as data structures, man-machine interfaces and the role played by knowledge-based systems as a suporting tool to the user in tasks of modeling, analysis and design of dynamic systems. At the end of the article are presented the main characteristics of some CACE environments being currently used.

1. INTRODUÇÃO

Estudos recentes (Challenges, 1987) reconhecem atualmente duas principais linhas de desenvolvimento na área de controle automático. A primeira diz respeito à pesquisa básica em controle e está orientada no sentido de se obter avanços significativos tanto a nível teórico quanto de aplicabilidade de técnicas de controle a classes mais amplas de problemas. A segunda linha é referente ao desenvolvimento de ambientes de programação do tipo CACE (Computer Aided Control Engineering) com as finalidades específicas de promover o aumento de produtividade do engenheiro de controle em tarefas de modelagem, análise e projeto e ao mesmo tempo situar o ensino e a pesquisa da teoria de controle dentro de um enfoque moderno e integrador. O presente artigo se restringe à abordagem de alguns aspectos desta segunda linha.

Historicamente é possível situar a utilização de computadores como ferramenta de auxílio ao engenheiro de controle em duas fases distintas. Até aproximadamente fins da década de 60, a produção de software na área de controle estava inteiramente orientada para implementações de segmentos isolados da teoria e/ou para aplicações de parte da teoria a problemas particulares de controle. Exemplo típico desta fase é o conjunto de programas para análise de sistemas lineares listados em (Melsa e Jones, 1970).

A partir do desenvolvimento e difusão de novas metodologias de projeto (engenharia de software, computação gráfica) viabilizadas pelo surgimento de novas arquiteturas de computadores, deu-se início à segunda fase de desenvolvimento de sistemas CACE, onde agora a ênfase seria concentrada em oferecer ao engenheiro de controle um ambiente de programação no qual a teoria ou uma parte significativa da teoria estivesse disponivel numa for-

ma integrada. Concorreu para tanto o fato da solução de um problema particular de controle frequentemente exigir a utilização de mais de um procedimento de projeto e a existência de um grande número de diferentes procedimentos de modelagem, análise e projeto que podem ser empregados na solução de uma ampla gama de problemas de controle. Automatizar o uso destes procedimentos permitindo ao engenheiro se concentrar apenas nos aspectos qualitativos das suas atividades traz benefícios imediatos em termos de aumento de produtividade e qualidade final do trabalho. À segunda fase mencionada deu origem a contribuições importantes de sistemas, como o LAS - Linear Analysis System, Bingulac et alli (1983), cujo desenvolvimento teve início na UNICAMP em meados da década de 70.

Comparativamente a países mais avançados na área, e a menos de iniciativas isoladas, o desenvolvimento de sistemas CACE no Brasil encontra-se hoje ainda no começo da segunda fase, enquanto no exterior já se concretiza uma evolução para sistemas CACE baseados em conhecimento (Taylor e Frederick, 1984). Por outro lado, o fato de se observar atualmente no país o surgimento de diversos grupos de pesquisa atuando em CACE (Murata e Yoneyama, 1986; Farines et alli, 1986) enseja uma revisão de trabalhos desenvolvidos nos últimos anos na área, como forma de tornar disponível a experiência de inúmeros pesquisadores que se dedicaram ao assunto e, ao mesmo tempo, indicar linhas, soluções e metodologias consolidadas e questões que permanecem em aberto em relação ao tema. Entre os trabalhos anteriores que já se preocuparam com estes aspectos, destacam-se Denham (1984) e Gomide e Szajner (1985).

O artigo está organizado da seguinte forma: na proxima seção são apresentados uma conceituação informal de sistemas CACE, a estrutura básica atual e algumas considerações sobre propriedades que estes sistemas devem apresentar. Na seção 3 discutem-se aspectos relacionados com interface homem-máquina, com ênfase em linguagens orientadas para a área de sistemas de controle. Na seção 4, a introdução de técnicas de inteligência artificial (sistemas baseados em conhecimento, especialmente) como ferramenta de suporte ao usuário é comentada e, em seguida, na seção 5, são analisados alguns dos principais sistemas CACE atualmente em uso. Finalmente, a seção 6 apresenta as conclusões do artigo.

2. CONCEITUAÇÃO E ESTRUTURA FUNCIONAL DE SISTEMAS CACE

Do ponto de vista conceitual, um sistema CACE pode ser entendido como um ambiente de programação onde uma parte significativa de teoria de controle (por exemplo, análise e projeto no domínio da frequência) se encontra descrita através de algoritmos computacionais que podem ser facilmente utilizados e *combinados* de modo a simplificar tarefas mecânicas e repetitivas de um usuário qualquer do sistema.

Embora existam diferentes implementações práticas deste conceito, entende-se que sistemas CACE modernos são aqueles geralmente constituídos por uma estrutura funcional onde destacam-se (na terminologia dos autores):

i) *Base de Utilidades*: um conjunto de rotinas matemáticas básicas (operações algébricas com polinômios, matrizes, números complexos) as quais suportam o desenvolvimento da *Base de Métodos*;

ii) *Base de Métodos*: um conjunto de procedimentos de modelagem, análise e/ou projeto descritos em termos de algoritmos computacionais que define, juntamente com a *Base de Dados*, o escopo dentro do qual o sistema trabalha;

iii) *Base de Dados*: uma estrutura de dados flexível que garante uma representação interna eficiente do segmento da teoria considerado, facilmente manipulada pelo usuário;

iv) *Interface Homem-Máquina*: meio de comunicação do usuário com o sistema responsável pela entrada e saída de informações, inclusive em forma gráfica, e geralmente associado a uma *linguagem orientada* para controle, através da qual o usuário pode facilmente expressar as ações que devam ser desenvolvidas e informações que devam ser passadas pelo sistema;

v) *Supervisor/Interpretador*: responsável pela coordenação de utilização dos recursos do sistema, particularmente no que se refere ao uso e interpretação dos comandos da linguagem, alocação de memória, manipulação de arquivos, etc.

A Fig. 1 apresenta uma visão simplificada de um sistema CACE estruturado como descrito em i) - v).

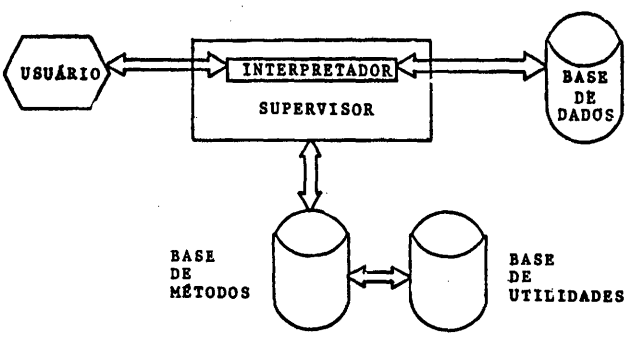

Fig. 2.1: Estrutura simplificada de um sistema CACE

Atenção especial deve ser dedicada à definição da Base de Dados, das características da Interface Homem-Máquina e das funções a serem desempenhadas pelo Supervisor/Interpretador. Embora estes três itens estejam mais diretamente relacionados ao ambiente no qual o usuário desenvolverá o seu trabalho, a sua especificação implica em estabelecer *a priori* limitações e potencialidades do sistema. Em

particular, a definição da Base de Dados, por estar intimamente vinculada à representação interna de modelos, deve refletir de forma flexível e eficiente o escopo dentro do qual o sistema será utilizado. Exemplificando, estruturas de dados cujos elementos básicos são funções racionais representam naturalmente modelos no domínio da frequência, assim como estruturas baseadas em matrizes reais têm a ver com representação de modelos no espaço de estados. Por uma estrutura de dados flexível deve-se portanto entender aquela que permite levar em conta diferentes tipos de representação de modelos.

O desenvolvimento de sistemas CACE de acordo com a estrutura funcional ilustrada na Fig. 2.1, além de fornecer uma visão geral do fluxo de informações no sistema facilmente assimilável pelo usuário, apresenta inúmeras vantagens. Uma vez que cada módulo da Fig. 2.1 possui funções bem definidas, torna-se bastante simples realizar manutenção periódica e eventualmente expandir a faixa de utilização do sistema, incorporando-se, por exemplo, novos métodos à Base de Métodos existente. Neste caso, a estrutura funcional da Fig.2.1 viabiliza a construção progressiva do sistema a partir da definição dos três itens mencionados.

Especificamente com relação à Base de Métodos se colocam algumas exigências as quais visam assegurar um desempenho adequado do sistema. Na implementação numérica de um determinado segmento da teoria deve-se, como regra geral e sempre que possível, selecionar entre os métodos existentes e que se adequam ao problema, aquele que reúne as melhores características em termos de *robustez* (no sentido de tratar eficientemente erros de arredondamento, mau condicionamento numérico, etc.), *aplicabilidade* (no sentido de estabelecer hipóteses fracas para sua utilização) e *eficiência computacional* (medida em termos de quantidade de memória necessária e tempo de processamento).

Considere, como exemplo para discussão destas características, um sistema linear discreto invariante no tempo representado pela equação a diferenças

$$x(k+1) = Ax(k), \qquad x(0) = x_o \tag{1}$$

onde $k = 0,1,\ldots,N$, $A \in R^{nxn}$ é uma matriz real e $x(k) \in R^n$ representa as variáveis de estado do sistema. Deseja-se, a partir do 2º método de Lyapunov, verificar se o sistema autônomo (1) é estável e para tanto utiliza-se a função de Lyapunov usual neste caso

$$V(x(k)) = x(k)'P\,x(k) \tag{2}$$

Impondo-se $\Delta V(x(k)) < 0$, vem

$$\begin{aligned}\Delta V(x(k)) &= V(x(k+1)) - V(x(k)) \\ &= x(k)'[A'PA-P]x(k) < 0 \qquad (3)\\ &\qquad \forall\, x(k) \in R^n\end{aligned}$$

O problema consiste então em, dado $Q = Q' > 0$, verificar se a solução $P = P'$ da equação matricial de Lyapunov

$$P = A'PA + Q \tag{4}$$

é definida positiva.

A solução de (4) pode ser obtida, entre outros, pelo método iterativo proposto por Davison e Man (Kitagawa, 1977)

$$P_{2\ell} = (A^{\ell})'P_{\ell}A^{\ell} + P_{\ell}, \quad \ell = 1,2,\ldots \tag{5}$$

ou pelo algoritmo não-iterativo de Kitagawa (1977) baseado em fatorização QR (Wilkinson, 1965). Entretanto, uma análise detalhada dos algoritmos mostrará que

i) o algoritmo não-iterativo é aplicado a partir de uma forma de Hessenberg de A, o que permite reduzir a solução de (4) à solução de uma série de equações algébricas com uma, duas ou quatro incógnitas, resultando num procedimento numericamente mais preciso do que o método iterativo, o qual trabalha com a matriz completa A;

ii) o algoritmo iterativo não pode ser aplicado se A possui autovalores fora do círculo unitário, o que neste caso é uma limitação bastante rígida. O algoritmo não-iterativo não apresenta esta limitação;

iii) o algoritmo iterativo requer entre $30\,n^3$ e $(175/2)n^3$ multiplicações enquanto que o algoritmo baseado em fatorização requer aproximadamente $21\,n^3$ multiplicações. Por outro lado, os requisitos de memória deste último são maiores.

Assim, no compto geral, é fácil decidir pelo algoritmo não-iterativo de Kitagawa, que apresenta uma melhor relação de compromisso entre robustez, aplicabilidade e eficiência computacional no processo de obtenção da solução (4), embora deva-se ressaltar que a sua implementação não é trivial.

De uma maneira geral, algoritmos não-iterativos, por possuirem custos computacionais bem definidos e envolverem hipóteses de aplicabilidade muitas vezes bastante fracas vêm merecendo atenção nos últimos anos, especialmente tendo-se em vista a sua utilização em ambientes CACE. Em Arnold IV e Laub (1984) são discutidas algumas técnicas baseadas em Autovalores Generalizados, empregadas pelos autores na solução de equações algébricas do tipo Riccati, equações estas que surgem naturalmente em alguns problemas de controle e filtragem de sistemas representados no espaço de estados. Detalhes de implementação destas técnicas podem também ser encontrados em Silva Fº (1987).

Embora discutidas num contexto específico, as características mencionadas devem sempre que possível estar presentes em cada rotina da Base de Métodos. Sem dúvida esta tarefa envolve um esforço considerável de pesquisa e não raro acaba exigindo a participação conjugada de especialistas em teoria de controle e análise numérica.

3. INTERFACE HOMEM-MÁQUINA

Os aspectos de interface homem-máquina se revestem de grande importância em sistemas que se propõem oferecer a seus usuários facilidades de manipulação de um volume considerável de informações, métodos de análise, técnicas de projeto, etc.

Na maioria dos sistemas CACE atuais, a interação com o usuário se dá através de uma linguagem orientada, de alto nível, cujas instruções procuram refletir procedimentos comumente utilizados por um engenheiro de controle na solução de um problema ao qual o sistema CACE em questão é adequado.

Uma característica desejável, e também consideravelmente mais difícil de se atingir na prática, é fazer com que esta linguagem orientada contemple diferentes categorias de usuários, desde principiantes até usuários mais experientes na manipulação da linguagem. Alguns sistemas hoje existentes procuram estabelecer um certo compromisso entre estas diferentes categorias incorporando outras formas de interação com o usuário, como por exemplo através de Menus e esquemas do tipo Pergunta-Resposta.

Formalmente, a definição de uma linguagem genérica implica em se considerar três aspectos básicos: sintaxe, semântica e sistema de execução. O primeiro aspecto, a sintaxe, determina as construções válidas, isto é, delimita a forma dos elementos que compõem a linguagem e como estes elementos podem ser combinados. A semântica atribui então um significado às construções válidas, estabelecendo uma relação entre a representação abstrata (a linguagem) e o seu efeito no mundo real (o resultado das operações efetuadas).

No caso de uma linguagem orientada é essencial que a definição sintática e semântica dos elementos da linguagem reflita conceitos e procedimentos básicos da teoria de controle a partir dos quais o usuário possa expressar as ações de modelagem, análise ou projeto que devam ser realizadas. As estruturas de controle de fluxo de execução de um programa escrito com base nesta linguagem não deve diferir essencialmente das estruturas de controle (*se* <condição> *então* <ação-1> *senão* <ação-2>, *enquanto* <condição> faça <ação-1>,...) comuns à maioria das linguagens de alto nível para propósitos gerais e, assim como estas últimas, deve apresentar algumas propriedades básicas como portabilidade, legibilidade, configurabilidade (no sentido de permitir a configuração do sistema para uma aplicação específica) e oferecer recursos para processamento simbólico.

Uma linguagem hipotética com as características acima poderia ser utilizada para, por exemplo, o cálculo da resposta temporal do sistema mostrado na Fig. 3.1 submetido a diferentes tipos de entradas padrão (impulso, degrau, rampa, etc.).

Neste caso, o usuário poderia expressar-se através da linguagem orientada da seguinte forma:

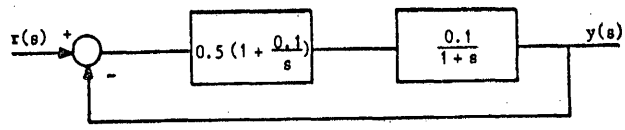

Fig. 3.1: Diagrama de blocos simples

```
r(s) := 1;
c(s) := 0.5*(1. + 0.1 / s^1);
p(s) := 0.1*(1. + s^1);
g(s) := c(s)*g(s)/[1. + c(s)*g(s)];
i := 3;
enquanto  i ≥ 0  faça
início
y(s) := g(s)*r(s);
respt(y(s), tmin, tmax, canal); /* Resposta
     Temporal */
r(s) := r(s)*[1./s^1];
i := i-1
fim
```

Note que este pequeno programa faz uso do procedimento *respt*, acessível ao usuário através da linguagem, permitindo o cálculo da resposta temporal de y(s) entre tmim e tmax cujos resultados são dirigidos para o dispositivo assinalado como *canal*. Com alguns recursos adicionais o usuário poderia definir este programa como uma *macro-instrução* a ser utilizada em alguma fase de análise ou projeto de sistemas de controle no qual devesse ser usada constantemente (Walker et alli, 1984).

Uma característica notável do programa apresentado é o fato de se utilizar uma linguagem orientada bastante natural e com grande capacidade de processamento simbólico. Na prática, estes dois aspectos em geral se contrapõem a uma implementação eficiente da linguagem, pois a interpretação das instruções listadas torna-se muito complexa dado o nível em que estas instruções foram codificadas.

As linguagens orientadas para CACE atuais procuram, em geral com maior ênfase na simplificação do interpretador, levar em conta todos estes fatores. A título de comparação são apresentadas na Tabela 1 as construções sintáticas utilizadas por alguns sistemas CACE para representar a igualdade matricial D = A+BC.

Do ponto de vista do usuário, a sintaxe utilizada pelo LAS pode parecer muito pouco natural, mas ao adotar uma representação do tipo

<entrada> (<operador>) = <saída>

a interpretação de instruções torna-se simples e homogênea. Por exemplo, através do LAS, a solução de um problema linear quadrático, dados os pares (A,B) e (Q,R) representando respectivamente o sistema e o critério de desempenho do problema é obtida com a instru-

TABELA 1 Representação de D = A+B*C em 3 sistemas CACE existentes

SISTEMA	D = A+BC
L.A.S.	B,C(*) = BC A,BC(+) = D
CTRL-C	BC = B*C D = A+BC
DSCAC*	OPMAT D = A+B*C.

ção

$$A,B,Q,R(RIK) = K,H$$

onde RIK é o operador necessário, K é a solução da equação matricial de Riccati e $H = A-B\bar{K}$. A Fig. 3.2 apresenta um pequeno programa escrito na linguagem LAS para projeto de um observador e alocação de polos através de realimentação de estado (Bingulac et alli, 1983).

```
1   (INP) = A,B,C,RR,RI,RRO,RIO
2   A,B,RR,RI(PPL,T) = F,AF
3   A,C,RRO,RIO(OBS,T) = FO,AFO
4   B,F(*) = A12
    :
    :
11  AA(EGV,T) = R1,R2
12  R1,R2,RR,RI(OUT,T,SOLUTION) =
```

Fig. 3.2: Programa escrito em LAS

Considerável esforço vem sendo realizado no sentido de trazer as linguagens de programação para próximo da linguagem com que normalmente o usuário se expressa. Além de proporcionar vantagens evidentes em termos, por exemplo, de legibilidade, uma linguagem natural é rapidamente assimilada pelo usuário, o qual tendo-a dominada, passa a se concentrar apenas em aspectos qualitativos relacionados às funções de engenheiro de controle.

Cabe ainda destacar a importância que interfaces gráficas (entrada e saída de dados em forma gráfica) adquirem na manipulação de ferramentas tão intrinsecamente associadas com este tipo de representação, como as da teoria de controle. Além dos recursos usuais de representação *2-D* (Root-Locus, Nyquist, Plano de Fase, etc.), existem atualmente aplicações de representação *3-D*, como no caso de projeto de controladores no espaço de parâmetros (Ackermann, 1980; Putz e Wozny, 1987). Evidentemente este tipo de representação pressupõe a existência e disponibilidade de equipamentos periféricos apropriados (terminais gráficos, plotters, mouse, touchpad, etc.).

A existência de algum tipo de suporte para utilização do sistema por parte dos usuários também é desejável. Neste contexto pode se incluir um editor associado à linguagem, um editor gráfico de diagramas de bloco que permita a definição de estruturas gerais de controle passíveis de análise através do sistema e possibilidade de acesso on-line (help) a informações sobre a linguagem (sintaxe e semântica de cada instrução), modos de operação do sistema, etc.

Na próxima seção será abordada a utilização de sistemas baseados em conhecimento como ferramenta de suporte à modelagem, análise e projeto de sistemas dinâmicos.

4. SISTEMAS CACE BASEADOS EM CONHECIMENTO

O emprego de técnicas de Inteligência Artificial, notadamente de Sistemas Baseados em Conhecimento (SBC), como suporte metodológico ao usuário em tarefas de modelagem, análise e projeto deve constituir-se nos próximos anos em atividade inerente ao desenvolvimento de sistemas CACE.

Embora muitas destas técnicas sejam ainda temas de intensa pesquisa são indiscutíveis os benefícios que a incorporação de algumas ferramentas básicas podem trazer a curto e médio prazo para a área de sistemas de controle. Esquemas simples podem ser introduzidos, e na prática alguns sistemas já o fazem, para fornecer por exemplo estimativas de condições iniciais, intervalos de tempo e amostragem, faixas de frequência e ganho, etc., e num contexto mais específico, selecionar, entre um certo número de algoritmos, qual o mais apropriado a uma certa classe de modelos.

Entretanto, nos exemplos mencionados, o auxílio ao usuário é feito a partir de análises de *modelos formais* (baseados em relações matemáticas) em termos de constantes de tempo, frequências de corte, etc., e não propriamente envolve o conhecimento (muitas vezes puramente empírico) de um especialista em práticas de modelagem, análise e/ou projeto. Lidar eficientemente com *modelos conceituais* (baseados em conhecimento não-quantificável matematicamente) representa atualmente a grande aspiração dos projetistas de sistemas CACE.

Contudo, antes de se partir para desenvolvimentos nesta linha, é fundamental se ter uma compreensão exata dos recursos e limitações tanto deste tipo de abordagem quanto das ferramentas de software necessárias para implementá-la. Um aspecto que deve ser sempre enfatizado é que um sistema CACE baseado em conhecimento não prescinde da atuação de um operador humano, e sim torna possível a sua utilização por usuários com diferentes níveis de conhecimento sobre técnicas de controle. A ênfase no uso de conhecimento deve ser dirigida para atividades nas quais o usuário possui uma capacidade *relativa* de compreensão (dependente de seu nível de conhecimento) sobre as con

(*) Sistema CACE em desenvolvimento no CTI/IA.

sequências de ações e procedimentos de controle no nível estrutural dos modelos, e não para atividades que envolvem decisões, muitas vezes bastante subjetivas. Neste sentido, verificar se as especificações de um determinado projeto são coerentes e realistas, tendo em vista as características da planta a ser controlada e as restrições de projeto/implementação dos controladores, seria uma atividade típica a ser desempenhada por este tipo de sistema.

Em termos práticos, o desenvolvimento de sistemas CACE baseados em conhecimento pressupõe, mais do que a existência de um conjunto coerente de metodologias de análise e projeto, uma abordagem conceitual do segmento da teoria no qual o sistema se propõe a atuar. Neste sentido, deve-se ressaltar as dificuldades fundamentais de representação de conhecimento em termos de software convencional, particularmente quando fatores como heurística e capacidade de inferência estão presentes, e de compatibilização desta representação de conhecimento com os modelos formais existentes para descrição de processo. Na Fig. 4.1 é proposta uma estrutura funcional (Amaral et alli, 1987) que procura levar em conta estas considerações.

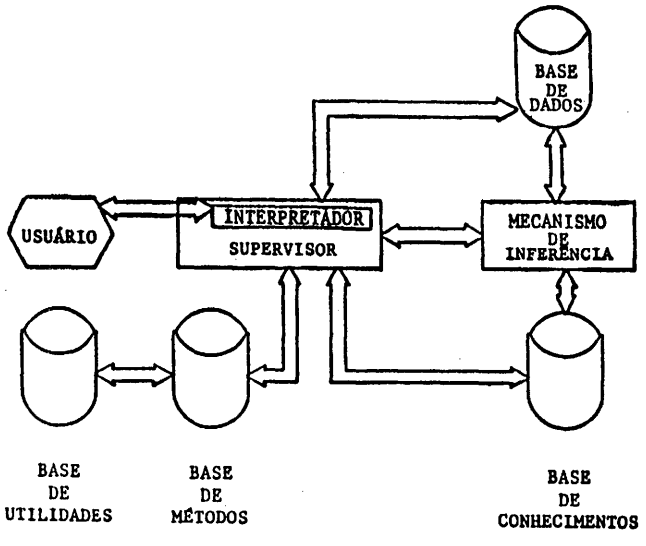

Fig. 4.1: Estrutura do CACE baseado em conhecimento

Em relação à estrutura convencional de sistemas CACE, a Fig. 4.1 apresenta uma diferença fundamental: a existência de uma *Base de Conhecimento* e de um *Mecanismo de Inferência* associado.

A *Base de Conhecimento* pode ser entendida como um conjunto estruturado de práticas e procedimentos heurísticos utilizados usualmente por um especialista em engenharia de controle. Através do *Mecanismo de Inferência* é possível obter conclusões a partir de *fatos* registrados pelo sistema e de práticas descritas na Base de Conhecimento. Frequentemente, o Mecanismo de Inferência possui capacidade de explicação no sentido de que, quando solicitado, pode fornecer ao usuário a linha de raciocínio na qual se baseou para chegar a uma determinada conclusão. Exemplo eloquente desta característica foi recentemente apresentado por Barbara et alli (1986) em termos de um sistema baseado em conhecimento para identificação de sistemas. A forma de interação típica deste sistema com o usuário é descrita a seguir.

o sistema é não estacionário e o estimador não acompanha as variações do sistema

sugere-se

alterar os limites de deteção

Você concorda? (sim/não/por quê?)

por quê?

se o sistema é não-estacionário
e o tempo de estabilização do detetor é maior do que o limite tr
e a média do erro previsto é maior do que limite 1
então é necessário aumentar o valor tr_{max} no módulo de deteção

Em geral, nos protótipos hoje existentes, a Base de Conhecimento é estruturada em termos de *regras*, do tipo *se* <premissa> *então* <conclusão>, *frames* que no contexto da Inteligência Artificial são definidos como *listas de fatos* relacionados a alguma situação, ou uma combinação destas duas estruturas, como a sugerida em Taylor e Frederick (1984). Neste trabalho, os autores propõem *frames* para a definição do problema (características da planta, restrições de projeto/implementação de controladores e especificações de desempenho) e para o *status* da solução corrente (descrito em termos de necessidades observadas e índices de desempenho obtidos). A cada um dos *frames* estão associados conjuntos de regras que têm como função selecionar, inicializar e coordenar o uso de ferramentas de análise e projeto disponíveis nos sistemas CACE convencionais.

Embora a implementação destas estruturas possa em geral ser realizada através de linguagens procedurais como Pascal e C é mais comum se lançar mão de linguagens derivadas da área de Inteligência Artificial, como por exemplo Prolog. Neste caso, alguns problemas podem surgir, pois linguagens como Prolog não oferecem atualmente recursos suficientes para manipulação de informações numéricas e precisão de cálculo frequentemente exigidos em aplicações na área de sistemas. Esta característica torna necessária uma interface entre as estruturas já mencionadas, que formam o núcleo da Base de Conhecimento, e os algoritmos computacionais, usualmente codificados em Fortran 77, Pascal ou C, que implementam os métodos numéricos de modelagem, análise e projeto. Um fator que torna atrativo o uso desta interface é a existência de um volume significativo de algoritmos computacionais para a área

de sistemas de controle, cuja qualidade e eficiência vêm sendo testada há vários anos.

Maiores detalhes sobre a utilização de sistemas baseados em conhecimento em CACE podem ser encontrados em Taylor e Frederick (1984), Birdwell et alli (1985) e Manson et alli (1985). O primeiro discute ainda algumas técnicas para implementação de Mecanismos de Inferência no contexto da estrutura de *frames* proposta pelos autores, destacando-se o uso de *forward chaining*, através do qual o Mecanismo avalia um sobconjunto de regras em premissas reconhecidamente verdadeiras, cujas conclusões podem ser incorporadas à Base de Conhecimento e usadas para se chegar às conclusões desejadas, ou ainda o uso de *backward chaining*, através do qual o Mecanismo tenta atingir uma conclusão específica identificando regras que contêm conclusões adequadas, tentando em seguida mostrar a validade das premissas associadas a estas regras.

Finalizando esta seção convém lembrar que as características de uma linguagem orientada, quando associada a um sistema CACE baseado em conhecimento, tornam-se substancialmente mais complexas se se pensar em termos de um sistema completamente integrado. Esta linguagem deve ser capaz de oferecer recursos para manutenção e expansão da Base de Conhecimento (o que pode ser mais simples) e ainda tornar possível a interação com o usuário em termos conceituais. Este último aspecto é ainda tema de discussão no campo da Inteligência Artificial e sua abordagem está além dos objetivos deste trabalho.

5. DESCRIÇÃO DE ALGUNS SISTEMAS CACE

Nesta seção são apresentadas as principais características de alguns sistemas CACE atualmente em uso. Na sua maioria, estes sistemas cobrem apenas parte da teoria de controle, embora esforços venham sendo desenvolvidos no sentido de se obter sistemas cada vez mais gerais.

5.a - *IDPAC* (Identification Package)

O sistema IDPAC é dirigido para análise de sinais (análise espectral, correlação, etc), estimação de parâmetros e identificação de sistemas. Oferece recursos para análise de séries temporais de modelos ARMA e ARIMA e facilidades para manipulação e representação gráfica de dados.

5.b - *CLADP* (Cambridge Linear Analysis and Design Package)

O sistema CLADP tem seu escopo de trabalho voltado para análise e projeto de sistemas multivariáveis no domínio da frequência. Robustez de sistemas lineares multivariáveis pode ser determinada a partir de representação gráfica ou valores singulares. Através do CLADP é possível descrever um sistema como uma série de subsistemas interligados, o qual tanto pode ser contínuo (domínio da Transformada de Laplace) como discreto (domínio da Transformada Z). O CLADP oferece recursos para conversão de domínios.

5.c - *SIMNON* (Non-Linear Simulation)

O sistema SIMNON oferece recursos para simulação de sistemas não-lineares contínuos e discretos e como o CLADP, cada sistema pode ser representado por uma série de subsistemas interconectados.

5.d - *SSDP* (State Space Design Package)

O sistema SSDP é inteiramente voltado para projeto de sistemas multivariáveis no domínio do tempo (espaço de estados). Fornece, entre outros, recursos para projetos de reguladores lineares quadráticos invariantes no tempo, para projetos de filtros do tipo Kalman-Bucy e testes de controlabilidade e observabilidade. Permite ao usuário projetar controladores robustos a incertezas da planta.

Estes quatro sistemas (IDPAC, CLADP, SIMNON e SSDP) foram unificados no sistema *FCACDS* (Federated Computer-Aided Control Design System) como descrito em Spang (1984) com o propósito de se obter um sistema que cobrisse uma boa parte do espectro da engenharia de controle (modelamento experimental, isto é, identificação (IDPAC), análise e projeto nos domínios do tempo (SSDP) e frequência (CLADP), e simulação (SIMNON)). Para tanto, foram definidas uma Base de Dados unificada e mecanismos para a migração de um sistema para outro. O módulo Supervisor do FCACDS se incumbe da instalação de cada subsistema (IDPAC, CLADP, ...) e da conversão de dados entre um subsistema e outro. Uma vez instalado, cada subsistema é operado independentemente através da sua própria linguagem.

5.e - *LAS* (Linear Analysis System, Bingulac et alli (1983))

Um dos pioneiros na área de CACE, o sistema LAS fornece recursos para análise e projeto de sistemas lineares multivariáveis tanto no domínio do tempo como da frequência e para análise de sinais. O LAS divide seus comandos em 4 categorias: (i) *entrada/saída*, para definição de arrays e seleção dos dispositivos de entrada e saída de dados; (ii) *álgebra linear*, para manipulação de métodos e procedimentos da álgebra linear, tais como operações com matrizes complexas ou polinomiais, cálculo de formas de Jordan e matrizes inversas generalizadas, etc.; (iii) *sistemas*, para solução de problemas específicos da teoria de controle, tais como alocação de polos por realimentação de estado ou saída, solução da equação matricial de Lyapunov, etc.; (iv) *controle de programas*, que consiste basicamente de operadores do tipo *GOTO* e *IF* usados para controle de fluxo de programas.

5.f - *CONSYD* (Control System Design, Holt et alli (1987))

O sistema CONSYD oferece, além dos métodos convencionais de análise e projeto de sistemas lineares multivariáveis e análise de sinais, facilidades para sua utilização em processos químicos podendo levar em consideração, por exemplo, atrasos de transporte e não linearidades definidas pelo usuário. Embora permita a análise no espaço de estados, maior ênfase é atribuída à sua utilização no domínio da frequência e assim como os demais sis-

temas mencionados também pode ser utilizado para fins de ensino e pesquisa.

5.g - *DSCAC* (Desenvolvimento de Sistemas de Controle Auxiliado por Computador, DSCAC (1984))

O sistema DSCAC vem sendo desenvolvido desde 1984 no Instituto de Automação do CTI com o apoio de especialistas da UNICAMP. Em termos de abrangência, o DSCAC é similar ao *FCACDS*, mas com a vantagem de haver sido concebido como um ambiente integrado, com Base de Dados e Interface Homem-Máquina únicas para todos os seus módulos. O sistema DSCAC possui uma linguagem interpretada natural com razoável capacidade para executar operações simbólicas e recursos gráficos apropriados, características que tornam a interação homem-máquina bastante amigável. O sistema leva também em conta diferentes categorias de usuários, desde principiantes até usuários experientes na manipulação de suas ferramentas, permitindo consultas *on-line* sobre características da linguagem (sintaxe, semântica, faixa de variação de parâmetros, etc.). A Fig. 5.1 mostra os módulos do sistema DSCAC que estão sendo atualmente implementados e a forma com que estes módulos estão relacionados.

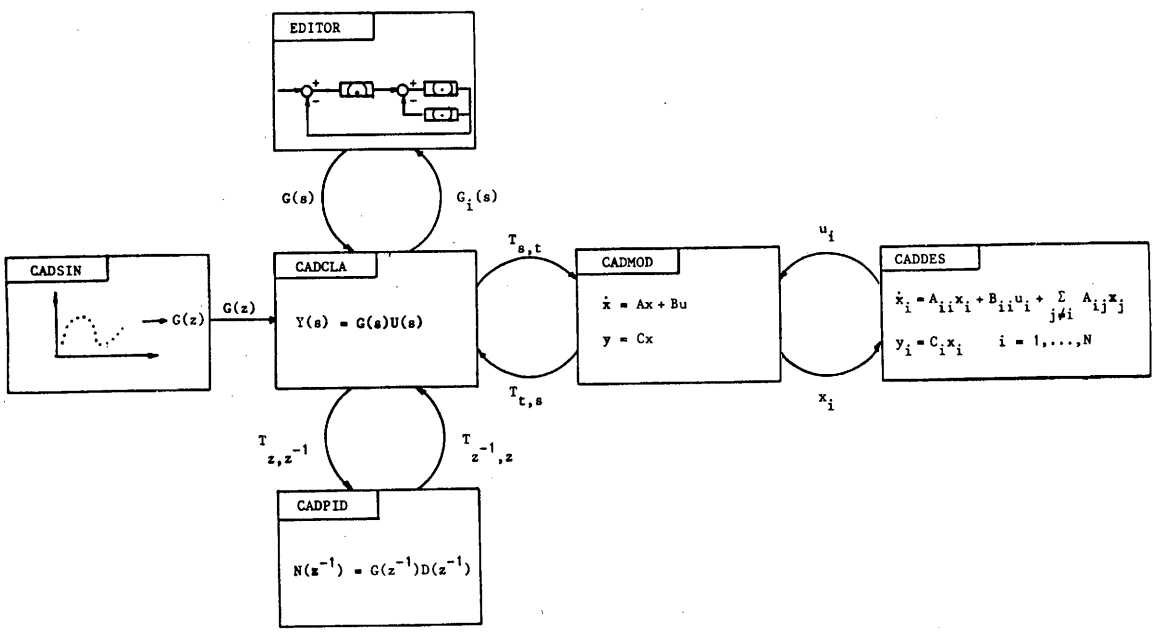

Fig. 5.1: Estrutura atual do sistema DSCAC

Uma descrição resumida de cada módulo da Fig. 5.1 é dada a seguir (quando necessárias, as transformações de domínio T são realizadas facilmente).

CADSIN: módulo para aquisição, tratamento e análise de sinais (filtragem, análise espectral, etc.) estimação de parâmetros e identificação de sistemas (métodos de mínimos quadrados, máxima verossimilhança, etc.).

CADCLA: módulo para análise e projeto de sistemas lineares multivariáveis pelos métodos clássicos (Root-Locus, Bode, Nyquist, Nichols, etc.). Permite tratar tanto sistemas contínuos quanto discretos.

CADMOD: módulo para análise e projeto de sistemas lineares multivariáveis pela teoria moderna (regulador linear quadrático gaussiano, estabilidade através do 2º método de Lyapunov, etc.). Permite tratar tanto sistemas contínuos quanto discretos.

CADDES: módulo para análise e síntese de controladores descentralizados usando técnicas de multiprogramação/multiprocessamento (métodos de Soma Ponderada e Sistema de Comparação, Decomposição Temporal e Espacial, etc.).

CADPID: módulo para análise e projeto de controladores digitais com estrutura PID (controladores PID clássicos, PID's auto-ajustáveis, PID's baseados em reconhecimento de padrões, etc.).

EDITOR: módulo que permite a edição gráfica e análise simbólica de estruturas gerais de controle.

Convém destacar que alguns destes módulos (em particular, o CADCLA e o CADPID) se encontram atualmente em fase operacional e assim como vários produtos intermediários do projeto DSCAC, vem sendo frequentemente utilizados em projetos que o IA/CTI desenvolve em comparação com a indústria nacional.

6. CONCLUSÕES

Neste trabalho procurou-se fornecer uma visão geral sobre a estrutura, requisitos e potencialidades de ambientes de programação do tipo CACE (Computer Aided Control Engineering). Embora tenha havido a preocupação de se levantar os principais pontos de discussão sobre a natureza e o papel a ser desempenhado por sistemas CACE, muitos outros pontos igualmente importantes poderiam ainda ser abordados, especialmente em termos de Interface Homem-Máquina e o uso de técnicas de Inteligência Artificial, cuja abordagem neste trabalho procurou enfatizar as características mais importantes e está longe de esgotar o assunto. Deixaram de ser descritos um número razoável de importantes sistemas com o Matrix$_x$, MATLAB, CTRL-C e ainda por limitações de escopo de trabalho, não foi possível focalizar o uso de sistemas CACE em aplicações industriais, que embora seja uma atividade recente, vem crescendo a cada ano e começa a se fazer presente no país.

7. BIBLIOGRAFIA

Ackermann, J., (1980). "Parameter Space Design of Robust Control Systems". *IEEE Trans. on Automatic Control*, Vol. Ac-25.

Amaral, W.C.; Bingulac, S.P., Ferreira, P. A.V.; Fontanini, W. & Gomide F.A.C., (1988). "A Knowledge Based Environment for Computer-Aided Control Engineering". Trabalho submetido ao *4th IFAC International Symposium on CADCS*, Beijing.

Arnold III, W.F. & Laub, A.J., (1984). "Generalized Eigenvalue Algorithm and Software for Algebraic Riccati Equations". *Proceedings of the IEEE.*

Bárbara, A.S.; Fontanini, W.; Amaral, W.C. & Gomide, F., (1986). "Um Supervisor Inteligente para Identificação de Sistemas". *6º Congresso Brasileiro de Automática*, Belo Horizonte.

Bingulac, S.P.; Gluhajid, N. & Milovanovic, A., (1983). *Computer-Aided Design of Control Systems Using LAS Language*, Automatika, Vol. 24.

Birdwell, J.D. et alli, (1985). "Expert System Techniques in a Computer Based Control System Analysis and Design Environment". *3rd IFAC/IFIP Int. Symp. CADCE'85*, Copenhagen.

"Challenges to Control: A Collective View". *IEEE Trans. on Automatic Control*, Vol. AC-32, 1987.

Denham, M.J., (1984)."Design Issues for CACSD System". *Proceedings of the IEEE.*

"DSCAC - Computer-Aided Control System Design". DSCAC Project. *Documento Técnico do IA/CTI nº 07/84*, Campinas.

Farines, J.M.; Savi, V.M. & Bruciapaglia, A. H., (1986). "Projeto Assistido por Computador para Sistemas de Controle: Um Pacote Interativo". *6º Congresso Brasileiro de Automática*, Belo Horizonte.

Gomide, F.A.C. & Szajner, J., (1985). "Análise, Síntese e Projeto de Sistemas de Controle Auxiliado por Computador". *Anais do 2º CONAI*, São Paulo.

Holt et alli, (1987). *Integrated Software Computer-Aided Control System Design and Analysis*. Comp. Chem. Engineering, vol. 11.

Kitagawa, G., (1977). "An Algorithm for Solving the Matrix Equation X=FXF'+S". *Int. Journal of Control*, Vol. 25.

Mansom, M.; Rimvail, M. & Shanfelberger, W., (1985). "Computer-Aided Design of Control Systems, an Integrated Approach". *3rd IFAC/IFIP Int. Symp. CADCE'85*, Copenhagen.

Melsa, I.J. & Jones, S.K., (1970). *Computer Programs for Computational Assistance in the Study of Linear Systems*. McGraw-Hill.

Murata, L.K.C. & Yoneyawa, T., (1986). "Programas Utilitários para Projetos de Sistemas Lineares Multivariáveis". *6º Congresso Brasileiro de Automática*, Belo Horizonte.

Putz, P. & Wozny, N.J., (1987). "A New Computer Graphics Approach to Parameter Space Design of Control Systems". *IEEE Trans. on Automatic Control*, Vol. AC-32.

Silva Fº, O.S., (1987). "Um Algoritmo para Solução Numérica da Equação Algébrica de Riccati - Casos Contínuo e Discreto - Baseado no Cálculo dos Autovalores Generalizados", *Projeto Técnico CTI/IA RT-PDC-001-005/003*, Campinas.

Spang III, H.A., (1984). "The Federated Computer-Aided Control Design System". *Proceedings of the IEEE.*

Taylor, J.H. & Frederick, D.K., (1984). "An Expert System Architecture for Computer-Aided Control Engineering". *Proceedings of the IEEE.*

Walker, R.A.; Shah, S.C. & Gupta, N.K.,(1984). "Computer-Aided Engineering (CAE) for System Analysis". *Proceedings of the IEEE.*

Wilkinson, J.H.: *The Algebraic Eigenvalue Problem*. Oxford University Press.