Luis Manuel Camarinha de Matos

Sistema de programação e controle de estações robóticas

-Uma arquitectura baseada em conhecimento-

Dissertação apresentada para obtenção do Grau de Doutor em Informática pela Universidade Nova de Lisboa, Faculdade de Ciências e Tecnologia.

LISBOA
1989

Janeiro 1989

AGRADECIMENTOS

Várias pessoas e instituições contribuiram para que este trabalho fosse possível.

Em primeiro lugar desejo expressar a minha profunda gratidão ao meu orientador, Prof. A. Steiger Garção, pelas profícuas discussões científicas que sempre proporcionou, pelo frequente questionar das ideias em gestação e pela ajuda no estabelecimento de metas e objectivos. O seu grande optimismo e forte incentivo foram determinantes para vencer os momentos de desânimo, em muito ultrapassando as atribuições de orientação, colocando-se antes no domínio da amizade. Desejo agradecer também os meios humanos e materiais colocados à disposição bem como as oportunidades e incentivos para participação na construção do ambiente laboratorial, coordenação de equipa e elaboração de propostas de projectos, o que possibilitou assim um valioso complemento da minha formação.

Em segundo lugar pretendo agradecer a valiosa colaboração dos colegas e estagiários que, em determinadas fases, tive oportunidade de coordenar, em especial João Moura Pires, Daniel Ferreira, Luis Correia, Albino Barbosa, Hélder Pita, Ricardo Rabelo, Joaquim Batista e Luis Osório, que deram contributos preciosos na implementação de vários componentes da parte experimental deste trabalho. Aos colegas Daniel Ferreira, Fernando Moura Pires e João Moura Pires desejo ainda agradecer a leitura de versões preliminares desta tese bem como os valiosos comentários e sugestões que então fizeram.

A todo o Grupo de Robótica Inteligente (GRI) da UNL sou devedor pelo ambiente criado onde muitas ideias puderam ser discutidas e aferidas bem como pelo esforço colectivo na criação de meios e realização dos diversos projectos de investigação que de alguma forma contribuiram para este trabalho.

A nível de instituições pretendo agradecer:

-Ao Departamento de Informática da FCT-UNL pelas facilidades concedidas, em particular pelo período de dispensa de serviço docente.

-Ao Ministério da Indústria e Energia pelo suporte a nível do projecto UNIROB.

-À Junta Nacional de Investigação Científica e Tecnológica pelo apoio às actividades de investigação do GRI.

-Ao Instituto Nacional de Investigação Científica pelo apoio às actividades do GRI e pela bolsa para doutoramento no país.

-À CEE pelos apoios através do programa ESPRIT.

Finalmente a minha gratidão vai para os meus pais que, apesar das suas dificuldades económicas, tiveram a visão e vontade suficiente para possibilitar o início desta carreira, e para a Tina e para a Rita pelos imensos transtornos familiares que tiveram de suportar durante a realização deste trabalho.

Sumário

Nesta tese faz-se a apresentação e discussão duma proposta de arquitectura para um sistema de programação e controle de estações robóticas para montagem de componentes mecânicos.

É seguida uma aproximação onde se considera o problema numa perspectiva CIM, isto é, em que a ênfase é colocada na integração dos diversos módulos funcionais da estação e na sua inserção num contexto mais geral dum sistema de produção integrada. Partindo dos resultados das fases de concepção de produto, planeamento de processo e especificação da estação, trata-se o problema da geração dum plano executável bem como da monitoração de execução e capacidade de recuperação de situações de excepção. Embora apontando para uma progressiva automatização de funções, a arquitectura concebida é baseada numa filosofia interactiva em que o especialista humano não é dispensado mas deverá antes interactuar com o sistema.

A integração de informação é apontada como elemento determinante e, assim, uma estrutura de sistema de informação, onde se identificam as principais famílias de dados / conhecimento, fluxos de informação, fontes e consumidores, é apresentada. Como estrutura de suporte defende-se um ambiente híbrido, computacionalmente distribuído, envolvendo sistemas de gestão de bases de conhecimento (representação por frames, programação reactiva, orientada por objectos e baseada em regras), gestão de bases de dados relacionais e CAD.

Em termos de resultados experimentais são apresentadas e discutidas implementações parciais demonstrativas dos aspectos determinantes do sistema proposto. São, nomeadamente, tratados os aspectos de construção do ambiente integrado de desenvolvimento (integração de diferentes tecnologias de suporte de informação e módulos funcionais), estabelecimento de modelos (estação, produto, tarefa, plano executável, modelo dinâmico do mundo e informação para recuperação de erros) e protótipos de planeamento, supervisão de execução e recuperação de excepções.

Finalmente apresentam-se as diversas questões em aberto e um plano para futuros desenvolvimentos.

Abstract

In this thesis a programming and control system architecture proposal for a robotic assembly cell is presented and discussed.

The problem is approached in a CIM perspective, meaning this that emphasis is put on the integration of the cell functional modules and their insertion in the more general context of an integrated manufacturing system. Results from the product design, process planning and cell specification phases, are the input for the generation of executable plans and the establishment of an execution monitoring subsystem able to recover from exception conditions. Even though a progressive automation of functions is desired, the conceived architecture is based on an interactive philosophy where the human expert is not put aside but is supposed to cooperate with the system.

Integration of information is pointed out as a determinant factor and, therefore, an information system structure, where the main data / knowledge concepts, information flows, sources and consumers are identified, is presented. As a supporting platform an hybrid, computationally distributed, environment involving knowledge base management systems (frame representation, reactive, object oriented and rule based programming), relational data base management systems and CAD systems, is defended.

In terms of experimental results partial implementations, demonstrating the fundamental aspects of the proposed system, are presented and discussed. Namely, the aspects of establishing an integrated development environment (integration of different information support technologies and functional modules), developing models (cell, product, task, executable plan, world dynamic model and error recovery information), and plan generation, execution supervision and exception handling prototyping, are analyzed.

Finally, several still open questions and a plan for future developments are presented.

NOTA SOBRE TERMINOLOGIA

Na preparação desta tese houve a intenção de utilizar a língua portuguesa para exprimir todos os conceitos e termos. Todavia, nesta área científica, como em muitas outras, a literatura dominante encontra-se em inglês pelo que muitos termos ainda não têm um correspondente em português que seja comummente aceite.

É o caso, por exemplo, de "frame" (enquadramento ?) e "slot" (ranhura ?), no domínio da representação de conhecimento, que não parecem ter ainda uma tradução pacífica, mesmo entre a comunidade da Inteligência Artificial. Uma consulta a colegas brasileiros, sempre tão criativos nestas questões, tão pouco resultou. Deste modo, neste caso e nalguns outros onde os termos ingleses são tradicionalmente usados (hardware, software, on-line, off-line) optou-se por não tentar qualquer tradução, apresentando antes esses termos em itálico.

Outra situação é a da utilização de siglas como CIM, CAD, CAM, FMS, CAPP, etc., que, dada a sua divulgação universal (também sem correspondentes em português pacificamente aceites - PIM ?, PAC ?, FAC ?, SFP ?, PPAC ?) nos pareceu adequado manter.

Relativamente à semântica da terminologia corrente em Robótica, também não se entendeu necessário introduzir qualquer glossário dado existir um bastante amplo preparado pela ISO ("Manipulating industrial robots - vocabulary", Technical Report ISO/TR 8373, Abr 88).

ÍNDICE DE MATÉRIAS

ÍNDICE DE FIGURAS

ÍNDICE DE QUADROS

1. INTRODUÇÃO

1.1 - APRESENTAÇÃO GENÉRICA DO PROBLEMA

1.2 - O ÂMBITO DA TESE

1.1 - APRESENTAÇÃO GENÉRICA DO PROBLEMA

Programação e controle de estações robóticas

A utilização de robôs industriais como componentes dos sistemas de produção tem, de acordo com várias estatísticas [BRA86], [Eng88], vindo a adquirir uma importância crescente não só nos países industrializados, mas também como tendência nalguns países em vias de desenvolvimento. O crescimento do número de instalações tem-se revelado, contudo, mais moderado do que algumas estimativas iniciais fariam prever. De entre as causas para uma tal moderação podem-se referir os elevados custos de instalação de sistemas deste tipo, a que não são estranhos a sua ainda pequena flexibilidade e reduzido grau de automatização do processo de adaptação a novas tarefas. O sucesso e incremento da aplicabilidade de sistemas robotizados passará, em nossa opinião, pela concepção de sistemas flexíveis de programação e controle que permitam reduzir substancialmente os tempos / custos de preparação para cada nova tarefa.

Num contexto de produção industrial, o robô não constitui um elemento isolado mas deve ser encarado como componente de sistemas mais complexos - células ou estações. Nestas poderemos ter, por exemplo, para além de robôs, uma série de elementos periféricos: transportadores (tapetes rolantes) e outros alimentadores, sistemas sensoriais, mesas rotativas, fixadores de peças, ferramentas a usar pelo robô, etc., e até outro tipo de elementos como máquinas de controle numérico.

Numa perspectiva técnica levanta-se a questão não só do controle do robô mas também o da sua interacção - cooperante - com outros componentes da célula. O problema é dificultado pela grande heterogeneidade desses componentes, não só quanto à sua finalidade e complexidade, mas essencialmente no que se refere aos níveis de controle. Alguns elementos estão dotados de controladores fornecendo como interface linguagens com um certo grau de abstração enquanto outros são bastante rudimentares apenas aceitando um conjunto de comandos de baixo nível.

A programação e controle dum tal sistema para a realização duma dada tarefa não é pois, um problema fácil, colocando requisitos em termos de ambientes ainda não presentes nas actuais soluções industriais. Isto é, o objectivo de construir um sistema que permita passar da especificação do produto a fabricar para um programa que controle a estação para a realização da tarefa, constitui uma actividade com múltiplas fases, requerendo contributos de várias áreas / especialistas e suportada por uma variedade de utensílios / tecnologias. Uma questão vital no estabelecimento destes sistemas será, então, o da integração dessas múltiplas vertentes.

Por outro lado, o problema da programação não pode ser desligado do controle de execução já que algumas informações poderão estar disponíveis apenas em tempo de execução e, em tal caso, alguma "programação" de detalhe terá de ser feita nessa altura. Desta forma, numa noção de sistema integrado de programação, estamos englobando a supervisão ou controle de execução.

Revisão de métodos de programação

Numa perspectiva histórica, o robô tem sido, de entre os diversos componentes da estação, o alvo preferido dos esforços de programação. Excluindo os subsistemas percepcionais, este é possivelmente o elemento mais complexo, o que pode justificar tal ênfase.

Os diversos métodos que têm sido propostos para a programação de robôs podem classificar-se segundo várias perspectivas:

i. Fases

•*on-line*

Nesta aproximação, por se usar directamente o robô e o ambiente envolvente real, o resultado obtido na fase de execução será o que se visualiza durante a programação (dependendo da característica de repetibilidade do robô).

Como principais desvantagens podem-se apontar:

-Exigência de disponibilidade do equipamento e das peças / produto durante a fase de programação.

-Falta de segurança.

-Reduzida velocidade de desenvolvimento.

-Ambiente de trabalho "sujo".

•*off-line*

Em oposição, numa aproximação em que o programa é desenvolvido de forma convencional num computador sem conexão - nessa fase - ao robô real, teremos:

-Mais segurança, versatilidade, pouco uso do equipamento / produtos reais e um ambiente de trabalho "limpo".

Como desvantagens:

-Exigência de bons simuladores, o que pode implicar hardware gráfico ainda caro.

Uma afinação final do programa terá de ser feita em *on-line*.

ii. Forma

• *Ensino "guiado" (guiding / by nose / teaching)*

Normalmente processa-se em *on-line* e a "programação" é feita a baixo nível especificando o conjunto de acções elementares através de um teclado de funções (*teach pendant*). Essa sequência é memorizada e depois repetida (*play back*) na fase de produção.

Nalguns casos, a indicação do movimento é feita conduzindo o robô pelo "nariz", ou seja, puxando pelo órgão terminal (garra).

Como principais problemas, têm-se dificuldades na alteração de passos intermédios do programa, estruturas de controle praticamente inexistentes, não modularização e consequente dificuldade na descrição de tarefas complexas.

•textual

Esta é a programação no sentido convencional do termo usando linguagens textuais. Baseia-se em ambientes *off-line* para o desenvolvimento dos programas e o teste pode ser feito, nalguns casos, em simulação com afinação final em *on-line*.

•gráfica

Usando simuladores gráficos pode-se ter uma simulação dos processos de "*guiding*". Poder-se-á também ter um sistema misto gráfico-textual.

iii. Nível

•axial

O controle do robô exerce-se explicitamente a nível dos seus diversos graus de mobilidade, isto é, actuando sobre os motores de cada eixo.

•manipulador

Neste nível, as localizações pretendidas para a extremidade do robô são especificadas pela indicação duma posição, em coordenadas cartesianas, e duma orientação. Isto implica a inclusão de processos de conversão de coordenadas cartesianas em axiais (transformação inversa) e de axiais em cartesianas (transformação directa). Sistemas deste nível também incluem interpoladores permitindo gerar diversos tipos de trajectórias (rectilínea, livre, etc.).

•objecto

Num mais elevado nível de abstracção, introduz-se a noção de objecto (ex. peças a manipular) e o correspondente modelo geométrico. Os movimentos do robô podem agora ser especificados em termos das relações pretendidas entre os objectos. Por exemplo, "mover objecto A tal que a sua face 1 fique coplanar com face 2 de objecto B e ...". Isto implica um sub-sistema de raciocínio geométrico que permita passar das relações entre os objectos para os referenciais cartesianos do nível anterior.

•tarefa

Num sistema de programação a nível tarefa (ou programação implícita) pretende-se especificar a tarefa em termos dos objectivos pretendidos (por exemplo montagem dum dado produto) sem ter de especificar como ela deve ser realizada. Isto implicará uma elevada capacidade de geração automática de planos e de modelação do mundo.

As classificações apresentadas não referem situações mutuamente exclusivas, mas antes perspectivas diferentes. Mesmo numa dada perspectiva, as divisões não são absolutamente rígidas. Poder-se-ão ter sistemas mistos. Por exemplo, a programação por "*guiding* " pode ser útil num contexto de programação textual para fazer o levantamento da cena (referenciais significativos).

Breve historial

Numa primeira fase, em termos cronológicos, não se poderia falar propriamente em programação já que a especificação da tarefa era feita em *on-line*, segundo um processo de "*guiding* " a nível axial. Alguns robôs industriais ainda fornecem (apenas) este nível.

Posteriormente progrediu-se para as chamadas linguagens de nível robô (ou manipulador) - linguagens de programação no sentido convencional do termo, facto a que não é alheia a vulgarização do controle (micro)computorizado. Trata-se genericamente de linguagens procedimentais, frequentemente interpretadas, nalguns casos compiladas. Em relação aos aspectos específicos da robótica, incluem tipicamente:

-O tipo de dados referencial (*coordinate frame*) - posição e orientação - relações entre referenciais e operações (transformações) sobre referenciais.

-Comandos para posicionar o manipulador (MOVEs) em termos de coordenadas cartesianas, podendo incluir especificação de:

.diferentes tipos de trajectórias (livre, rectilínea, circular, lista de pontos / *spline*, ...)

.velocidades, acelerações (nalguns casos), o que implica um controlador comportando modelos cinemático e eventualmente dinâmico do robô.

-Movimentos guardados (*guarded moves*), isto é, condicionados por monitoração de informação sensorial.

-Aquisição de informação sensorial (limitada e a baixo nível).

-Controle da garra; abrir, fechar, com indicação de amplitude, força, etc..

Contudo, grande parte destas linguagens parece ter-se desenvolvido num aparente divórcio em relação aos avanços que entretanto foram surgindo na informática (linguagens de programação, controle em tempo real) e são, deste ponto de vista, bastante simplistas:

-Os mecanismos de controle são normalmente pouco estruturados (muitas vezes a nível do Basic).

-Tipos de dados bastante primitivos e sem possibilidade de definir novos tipos.

-Normalmente sem facilidades de modularização nem de conexão a sub-sistemas externos.

-Os mecanismos para a expressão da concorrência não estão presentes ou são incipientes (com algumas excepções).

Embora inspirados em linguagens convencionais existentes, a maior parte dos casos passou pelo desenvolvimento de sistemas a partir de base. Algumas propostas, contudo, apontaram para a utilização de linguagens convencionais com boas características de modularidade, potencialidades para especificação de novos tipos de dados e programação

concorrente, e aumentá-las com a funcionalidade adicional para o controle do robô. Exemplos: Ada [GinGin82], Concurrent Pascal [SteCam86].

Como se referiu, as primeiras propostas de sistemas de programação estavam essencialmente centralizadas no robô. Assim, dificilmente são contemplados outros componentes da célula. Por exemplo, a linguagem LM contempla múltiplos robôs, mas outros periféricos tem de ser modelados usando o "truque" de os considerar como ferramentas do robô.

Em súmula pode-se dizer que a noção de estação e, consequentemente, a de programação e controle da estação não estão presentes em tais sistemas.

De facto, numa célula podem-se ter diversos tipos de controladores com diferentes níveis de abstracção, desde simples PLCs (*Programmable Logic Controllers*) até interpretadores de linguagens como as descritas. O grau de heterogeneidade pode aumentar também na horizontal se forem usados robôs de fabricantes diferentes - em geral cada um com a sua linguagem. Algumas tentativas de normalização destes controladores têm surgido. É o caso da IRDATA [Rem86] a nível alemão ou da ESL (*Explicit Solution Language*) [JakNas88] no âmbito dum projecto ESPRIT.

Contudo, o problema da programação a nível estação persiste (pelo menos em termos de sistemas flexíveis).

Como se pode deduzir do exposto, os custos de desenvolvimento de programas nestas condições são elevados, a depuração de erros (*debug*) - sempre difícil em sistemas de tempo real - é aqui agudizada, as soluções atingidas são em geral rígidas, para geometrias rigorosas (células altamente estruturadas) e, portanto, só compensatórias para casos de produção em larga escala.

Por outro lado há que contar com especialistas nas diversas linguagens / controladores existentes na célula .

Este nível representa o estado corrente na indústria. Em termos de investigação foram desenvolvidos alguns protótipos parciais de linguagens nível objecto (RAPT [PopAmBel78], [AmbCamCor86], LM-GEO, etc.). Para um quadro sumário comparativo, ver [Nev86].

Quanto ao nível tarefa, após algumas especificações iniciais de possíveis linguagens e algumas tentativas de implementações muito parciais [Loz83], compreendeu-se que o problema era consideravelmente complexo, e esta continua a ser uma importante área de investigação.

Flexibilidade

O desejo de maior flexibilidade nos sistemas de produção e a redução do tempo de programação implicam um progressivo nível de automatização dessa programação.

De facto constata-se que uma ênfase no vector flexibilidade em detrimento do vector productividade está ganhando importância crescente nos sistemas de produção industrial. Daí a popularização de siglas como FMS (*Flexible Manufacturing Systems* - Sistemas Flexíveis de Produção) e FAS (*Flexible Assembly Systems* - Sistemas Flexíveis de Montagem). Tal mudança de enfoque implica um certo deslocamento de objectivos e requisitos bem sintetizado em [Nei87] (tab. 1.1).

Productividade	Flexibilidade
Grandes lotes	Pequenos lotes
Standardização	Variedade
Elevados stocks (peças, produtos)	Pequenos stocks (peças, produtos)
Elevados custos/tempos de estabelecimento	Baixos custos/tempos de estabelecimento
Resposta lenta a:	Resposta rápida a:
-inovação do produto	-inovação do produto
-inovação do processo	-inovação do processo
-desvios na procura	-desvios na procura
Baixa utilização dos equipamentos	Elevada utilização dos equipamentos
...	...

Tab. 1.1 Productividade vs. flexibilidade - caracterização

Adicionalmente são de considerar:

-uma diminuição dos ciclos de vida dos produtos;

-um aumento dos padrões / requisitos de qualidade;

-clientes requerendo tempos de entrega mais curtos e cumprimento rigoroso de datas;

-pequena margem para aumentos de preços.

Esta nova situação / tendência tem implicações a nível hardware e software.

Em termos de equipamentos podem-se referir:

-robôs e máquinas NC multi-ferramenta,

-robôs multi-tarefa em vez de robôs especializados,

-avanços nos sistemas de comunicações,

-esforços de modularização dos fixadores (*fixtures*) ou cooperação multi-robô,

e, em geral, uma diminuição do grau de estruturação da célula, o que passa por um acentuar da utilização de sensores.

A nível do software torna-se imperioso evoluir na direcção duma progressiva automatização - sistemas de programação interactiva semi-automática (nesta fase) - concepção de sistemas com capacidade de tomar decisões *on-line:* planeamento local e recuperação de erros.

A outro nível são de considerar as implicações na própria concepção do produto, referindo-se a premência duma maior interacção entre os sectores de concepção e produção. O surgimento de áreas como "*design for assembly*" ou "*assembly oriented product design* " reflecte tal situação.

O problema num contexto CIM

Como foi referido, a maior parte das aproximações anteriores apresentava grandes limitações ou por se restringirem ao robô (como é o caso das versões mais "industriais") ou por se encontrarem "desfasadas" dos problemas reais (como é o caso de muitas abordagens da Inteligência Artificial baseadas no "mundo dos blocos").

A fim de conseguir uma solução adequada para um problema tão complexo, é necessário entender o processo de produção industrial no que respeita às diversas áreas funcionais e respectivos resultados. A programação duma célula robótica deve ser integrada neste contexto.

Várias funções do processo produtivo têm sido objecto de automatização (pelo menos parcial) dando origem ao termo "ilhas de automatização". Assim, vários sistemas "baseados em computador" (*computer aided*) têm surgido. Por exemplo, CAD (*Computer Aided Design*) para a concepção / projecto do produto, CAPP (*Computer Aided Process Planning*) para o planeamento do processo de fabrico e selecção de componentes da estação, CAM (*Computer Aided Manufacturing*) lidando com a execução do processo de fabrico (usando, por exemplo, máquinas ferramentas de controle numérico). O controle de qualidade também é, nalguns casos, realizado com auxílio do computador e daí os sistemas CAQ (*Computer Aided Quality assurance*). Por outro lado, na zona mais administrativa, desde há muito que a informática tem sido usada na

automatização de funções, podendo introduzir-se o termo CAA (*Computer Aided Administration*).

O desejo de integração de todas essas funções (fig. 1.1) num todo coerente deu origem à expressão Produção Integrada por Computador (CIM - *Computer Integrated Manufacturing*).

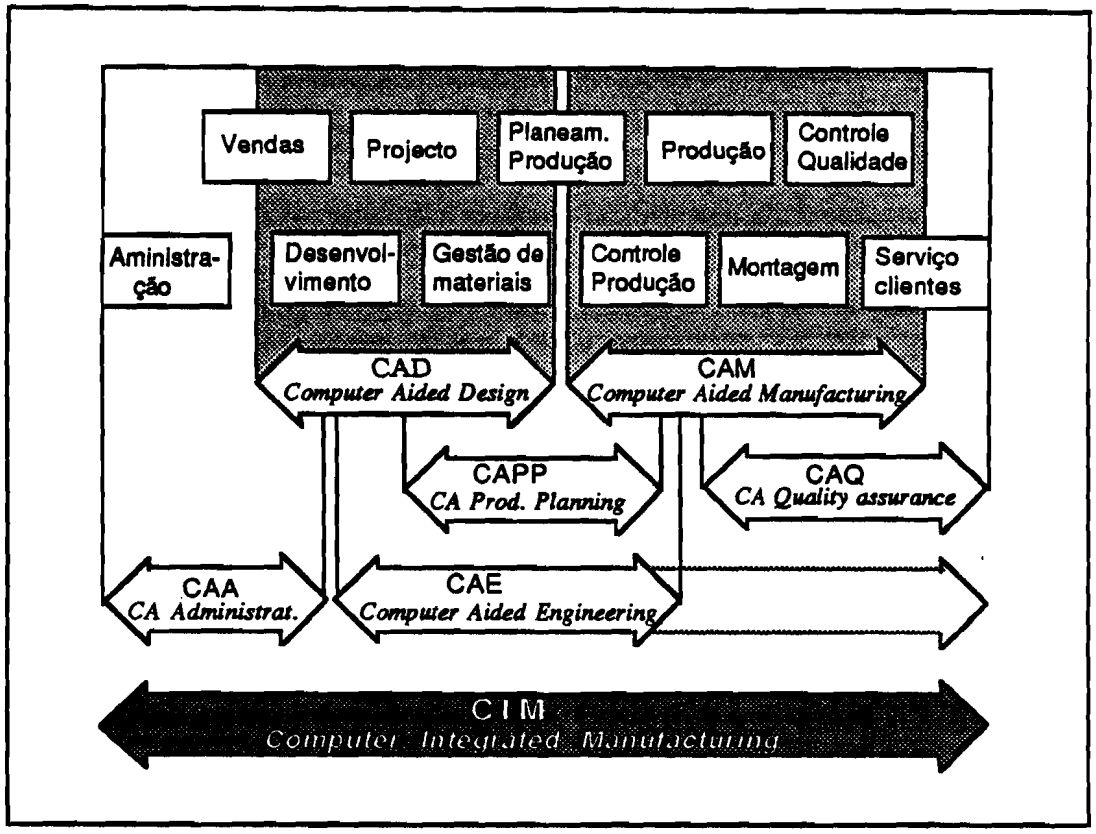


Fig. 1.1 Produção integrada por computador [Nei87]

O CIM é ainda um conceito algo impreciso e apenas implementações muito parciais poderão ser encontradas para certas áreas. Há mesmo quem afirme ainda não existir qualquer implementação CIM a nível mundial. Numa tentativa simplificada de clarificar o conceito poder-se-á dizer que o CIM tenta a integração dos processos de desenvolvimento e concepção, planeamento e fabrico, i.e., tenta suportar sob um sistema computacional integrado, a evolução do produto desde a concepção e projecto inicial até à fabricação e montagem, passando pelo planeamento e incorporando também as funções de gestão, contabilidade, stocks, etc. [Nei87]. Noutro sentido, pode-se dizer que pretende materializar a "ligação" entre "ilhas" previamente separadas. Uma integração de informação é, para este efeito, uma questão fulcral.

Neste contexto, a programação duma estação robótica para a montagem dum dado produto é antecedida por muito trabalho que pode ser considerado "preparatório" para essa tarefa. Por exemplo, concepção do produto (CAD), concepção da célula / selecção de ferramentas, planeamento do processo de fabrico (CAPP), etc.. O sistema de programação deve - segundo a aproximação aqui seguida - ter em atenção os resultados dessas fases e ser integrado nesse processo global.

É de notar que, embora se pretenda uma aproximação integrada ao problema da programação, pelo facto de o CIM ainda não estar bem caracterizado - sendo também uma área de investigação - não é possível fazer uma abordagem perfeitamente descendente (*top down*). Pode-se partir de alguns pressupostos (algumas funções e resultados que parecem óbvios no sistema global) e depois seguir uma análise algo ascendente (*bottom up*), esperando que o estudo da problemática da programação e controle da estação também contribua para a clarificação do conceito de CIM, pelo menos em termos de fluxos de informação.

A nível local - estação - também várias áreas têm tido desenvolvimentos separados: modelação de robôs (geométrica, cinemática, dinâmica), simulação, percepção sensorial, controle e recuperação e erros, programação concorrente, etc., impondo-se não só a referida integração a nível macroscópico (CIM) mas também uma integração de funções e informação a nível local.

Mesmo que as diversas sub-áreas não tenham atingido ainda a maturidade adequada - note-se que elas têm seguido uma evolução separada, de certo modo *bottom up* - é necessário um esforço de estruturação (pelo menos parcialmente) descendente, até como forma de re-orientar esses desenvolvimentos parcelares. De facto, o processo, dada a sua complexidade, exigindo domínio de múltiplas áreas científicas / tecnológicas, deverá ser iterativo: um esforço de integração poderá sugerir alguma redefinição das funções componentes; novos desenvolvimentos desses módulos podem, por sua vez, levar a refinamentos da estrutura global.

1.2 - O ÂMBITO DA TESE

Objectivos gerais

O objectivo deste trabalho consiste em apresentar uma **proposta de arquitectura para um sistema integrado de programação e controle de robôs** no seio de estações robóticas.

Uma das linhas de força da proposta reside na ideia de integração no contexto descrito atrás.

A implementação total dum tal sistema é trabalho para muitos homem/ano, envolvendo forte interacção, e certamente continuará a ser objecto de intensa investigação no futuro. Deste modo, não se pretende apresentar "a solução final" mas tão somente um contributo para a compreensão e o avanço da área. Tão pouco se explorarão exaustivamente todas as zonas de detalhe da estrutura apresentada por tal exceder largamente os limites temporais e os recursos afectos ao trabalho.

Contudo vários trabalhos experimentais foram realizados e publicados permitindo servir como demonstração parcial da consistência das propostas feitas.

Daqui resulta que este não é um trabalho concluído mas sim algo a ter continuidade. Deste modo, os resultados apresentados correspondem ao terminar duma dada fase mas um plano de investigação em continuidade emerge facilmente da apresentação.

Domínio de aplicação

Como área de referência considera-se fundamentalmente o caso de **células de montagem** (*assembly*) de componentes mecânicos. A opção por este domínio de aplicação foi motivada por:

-se tratar de um dos problemas mais gerais, englobando, pelo menos parcialmente, muitas das questões encontradas noutros domínios;

-ser um problema menos estudado, comparativamente às aplicações à soldadura, "*pick and place* ", etc.;

-se tratar de um caso com crescente importância económica.

Todavia, esta opção não invalida que muitas das questões abordadas e soluções propostas sejam generalizáveis para outros domínios.

Organização por capítulos

Neste primeiro capítulo foi feita uma apresentação genérica do problema abordado e uma caracterização dos objectivos perseguidos.

Numa segunda parte faz-se a apresentação da proposta de arquitectura em termos gerais: estrutura de blocos funcional e apresentação do Sistema de Informação como base vital para a integração.

A terceira parte é reservada à discussão de aspectos de detalhe do sistema e apresentação de resultados experimentais. Um divisão em duas grandes áreas é feita:

-Programação genérica, incluindo a estruturação dos componentes de informação considerados como ponto de partida, a geração semi-automática de planos e simulação como apoio à programação interactiva.

-Controle de execução e recuperação de erros, incluindo a estrutura do supervisor multi-agente, monitoração, diagnóstico e recuperação de erros, simulação.

A extensão e complexidade dos tópicos abordados dificultam uma apresentação clara da proposta. Para reduzir tais dificuldades foi tomada a opção de fazer primeiro uma apresentação genérica (cap.2) e só posteriormente uma discussão mais detalhada (cap.3). Contudo, apesar desse esforço, não foi possível eliminar a necessidade de referências cruzadas no texto - certos conceitos referidos num dado ponto apenas podem ser clarificados noutra zona mais à frente - o que dificulta uma leitura sequencial.

No capítulo 4 apontam-se conclusões, limitações e linhas de desenvolvimento / generalização futuras.

Finalmente uma bibliografia completa o trabalho.

Em trabalhos deste tipo é tradicional apresentar um capítulo inicial com o estado da arte no ponto de partida de forma a facilitar a compreensão do que se propõe de novo. Entendemos que esse figurino se ajusta bem a casos em que o trabalho incide sobre um tópico bastante delimitado.

No caso presente, dada a extensão e não especificidade dos temas envolvidos, a estratégia adoptada foi a de tentar uma apresentação distribuída pelo texto nos pontos em que tal se justifique.

Por outro lado, dada a extensão temporal em que decorreu o trabalho - cerca de 4 anos e meio - e ao facto de esta ser uma área onde se começaram a verificar intensos esforços de investigação, o estado da arte tem evoluído bastante. Deste modo, tenta-se fazer uma comparação paralela entre as propostas feitas (nas suas diferentes fases) e a situação em cada momento.

2. MODELO GERAL

2.1. ESTRUTURA FUNCIONAL

2.2. SISTEMA DE INFORMAÇÃO

2.1 - ESTRUTURA FUNCIONAL

2.1.1 CONSIDERAÇÕES GERAIS

Ponto de partida

Como foi enunciado na primeira parte, o sistema de programação e controle deve ser concebido de forma a integrar-se numa filosofia de CIM. Por outras palavras, assume-se que muito do trabalho realizado a nível de concepção do produto (CAD), planeamento do processo (CAPPS) e estabelecimento de células pode ser bastante útil para a actividade de programação se uma adequada integração destes níveis for estabelecida. Muitas aproximações anteriores parecem, contudo, "perder" grande parte da informação presente ou gerada nessas fases.

Tais áreas (projecto e planeamento de sistema) desempenham, de facto, uma actividade preparatória que é de importância vital para a programação. É, então, necessário estabelecer uma adequada forma de "interface" entre o sistema de programação e os outros níveis do sistema CIM.

Para clarificar essa interface é necessário fazer uma análise do comportamento funcional macroscópico de tais níveis, embora dentro das contingências duma ainda não completa clarificação do conceito de CIM. Neste sentido, caracterizemos, ainda que de forma sumária, estas actividades que antecedem a programação e suas inter-relações:

O produto que se pretende produzir (montar) e respectivos componentes é modelado usando um sistema CAD. Adicionalmente, restrições económicas e tecnológicas desejadas ou requeridas são especificadas: limites de custo, qualidade (tolerâncias, ...), quantidades, tempo de fabrico, falhas admissíveis no equipamento, etc..

Para responder a tais especificações há que encontrar uma estação de fabrico adequada - componentes e sua disposição espacial (*layout*). Duas situações podem ocorrer:

i. Pretende-se realizar o processo numa estação concreta já existente. A questão, neste caso, será a de determinar se a tarefa é realizável ou não. Nalguns casos (dependendo do grau de "flexibilidade" da célula) podem-se ter alguns graus de liberdade - por exemplo, o conjunto de ferramentas a ser usado por cada máquina.

ii. O objectivo é conceber uma estação capaz de produzir o produto dentro das restrições dadas. Algumas tentativas para automatizar (parcialmente) esta tarefa constituem assunto para vários trabalhos de investigação corrente.

Em qualquer caso, o problema da especificação / avaliação da célula está estritamente relacionado com o processo de produção adoptado pelo que as duas actividades deverão interactuar. O plano para o processo será desenvolvido pelos engenheiros de produção. Algumas tentativas para construir sistemas interactivos destinados a ajudarem estes engenheiros na tarefa de planeamento de produção têm sido feitos ultimamente [Fur87].

Para avaliar a factibilidade e o desempenho dum dado plano de produção (gama de fabrico), há que simular a sua realização relativamente à estação adoptada.

Esta interacção deveria ser alargada à fase de concepção do produto. Alguma realimentação de informação dos processos tecnológicos poderia levar os engenheiros de concepção a encontrarem esquemas alternativos, ou refinados, melhor adaptados aos processos de produção disponíveis.

Esta necessidade de interacção entre as subáreas de concepção e produção tem sido referida como algo muito importante mas tem sido de difícil implementação. Contudo, alguns tópicos de investigação referidos por "Projecto orientado para montagem" (*Design for assembly*) [Gai87] ou "Projecto orientado para fabricação" estão sendo desenvolvidos.

A fig.2.1.1 [CamSte88] representa, de forma simplificada, as actividades mencionadas e suas principais inter-relações.

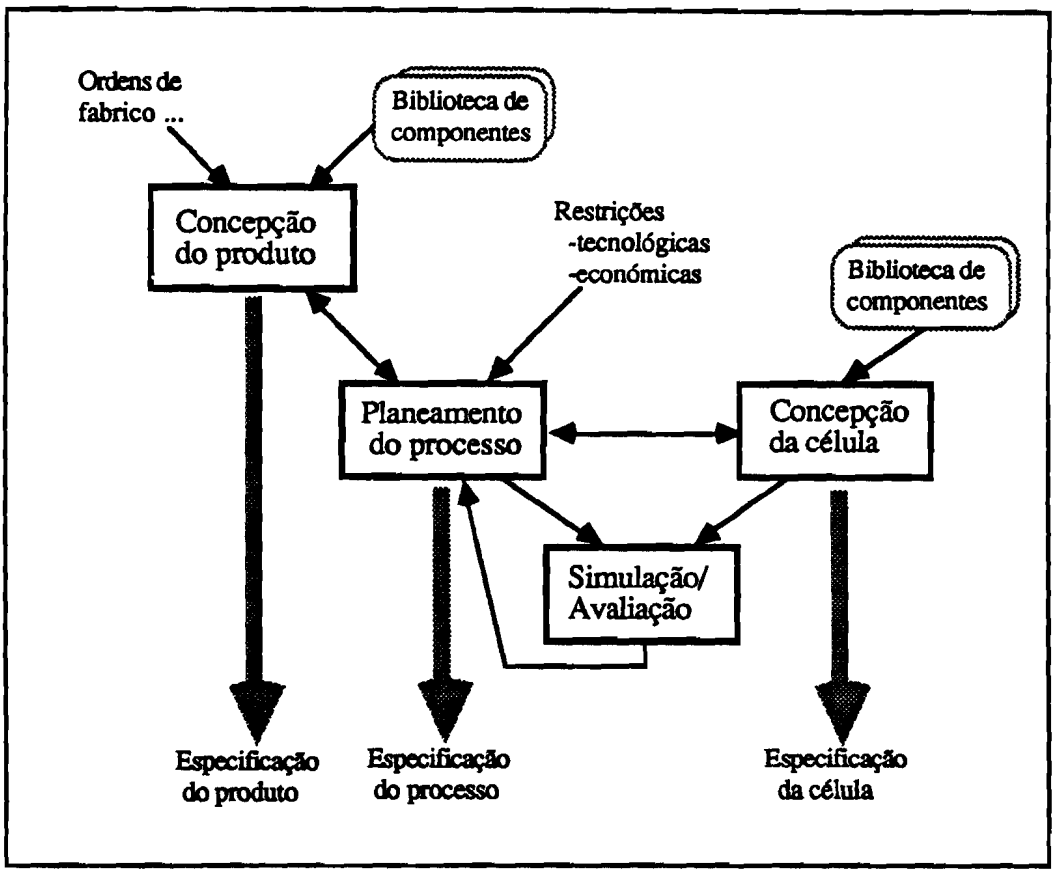


Fig. 2.1.1 Planeamento de sistema

Ainda nos encontramos muito longe dum sistema genérico completamente automatizado para realizar estas tarefas, contudo alguns subsistemas interactivos que têm sido propostos nos últimos anos parecem atestar a consistência desta nossa visão.

Por exemplo, num Projecto ESPRIT (*Operational Control for Robot System Integration into CIM*) [Spu...85], [SpuFurKir87], um conceito de procedimento integrado de planeamento de sistema foi desenvolvido onde se examina a factibilidade técnica dum sistema FAS, determinam os dados básicos, avalia a disposição (*layout*) dos componentes da estação e decide sobre a factibilidade económica. Uma vez tais critérios satisfeitos, o procedimento termina com o detalhe final do sistema. Alguns protótipos interactivos parciais estão sendo desenvolvidos de acordo com tal esquema. Por exemplo, um trabalho desenvolvido na Universidade de Galway, Irlanda [TieBowBro86] fornece ao engenheiro de produção um sistema interactivo que verifica a consistência do plano de processo e faz sugestões quanto às ferramentas a usar para cada subtarefa.

No caso de alguns domínios especializados de menor complexidade, como a montagem de componentes electrónicos em placas de circuito impresso, podem-se

encontrar algumas propostas ambicionando um maior grau de automatização. Como exemplo, pode referir-se um projecto da Universidade de Purdue [IriCha88]. Os maiores esforços em planeamento de processo têm, contudo, sido desenvolvidos na área da fabricação de peças (maquinagem).

Quanto à concepção de estações é de referir os trabalhos na área de Tecnologia de Grupos. A ideia de base consiste em agrupar as peças que requeiram operações similares e as máquinas correspondentes a essas operações. Daqui resultam famílias de produtos e células de máquinas. As vantagens desta aproximação face ao método mais tradicional de agrupar as máquinas por funções (grupos de tornos, por exemplo) incluem:

-Simplificação de fluxo de peças e ferramentas;

-Redução de tempos de "setup" e global de processamento e facilita inventariação do trabalho em curso;

-Maximização da eficiência de concepção e fabrico, já que o projecto e planeamento de processo de peças similares a outras produzidas anteriormente podem ser facilitados.

Várias técnicas e sistemas para auxiliar esta actividade têm sido propostos. Uma visão panorâmica da área pode ser encontrada em [Mac86] e [KusHer87].

A Tecnologia de Grupos tem-se preocupado com os processos de fabrico em geral, talvez com mais ênfase na maquinagem de peças, mas pensamos que algumas das técnicas desenvolvidas podem também ter algum impacto no caso particular do estabelecimento de estações de montagem.

Outros trabalhos mais especificamente ligados à Robótica (IPK/Berlin [SpuFurKir87], por exemplo) assumem uma aproximação interactiva em que a selecção dos equipamentos a incluir na estação é feita pelo operador, a partir duma biblioteca de componentes. Um sistema gráfico permite então estabelecer e visualizar a distribuição espacial dos componentes (*layout*) e verificar a sua consistência (por exemplo, testar se o robô consegue atingir todos os pontos importantes).

De forma manual ou automática, o que importa para a questão em debate é o resultado produzido por este nível. Uma caracterização dos componentes de informação - especificação de produto, processo (gama) de fabrico, e estação - resultantes das fases de concepção e planeamento de processo permitirá definir o objectivo para a actividade de programação, pelo menos do nosso ponto de vista.

Vias para a integração

A integração não é uma questão fácil. Para além de envolver um elevado esforço de engenharia, coloca questões de investigação importantes já que, sem um adequado modelo, não é viável chegar a um sistema consistente [Fik87].

É de notar que os vários subsistemas que se pretendem integrar têm tido uma evolução, em grande medida, de forma isolada e sem um modelo global de suporte. Desta forma, as suas funcionalidades e tipo de interfaces apresentam frequentemente grandes "desvios" em relação às necessidades do sistema integrado de programação e controle.

Não seria preferível começar tudo de novo a partir do zero?

A aproximação adoptada foi a de que num problema com este grau de complexidade se deve seguir uma via iterativa - clarificação gradual de ideias - não se podendo desprezar uma imensa quantidade de trabalho e conhecimento presente nos subsistemas / áreas referidos. A aposta é de que é possível conseguir uma solução sem grandes distorções usando tais resultados e que a experiência então adquirida deverá permitir especificar com mais segurança o que deverão ser tais módulos no futuro.

Por exemplo, um componente importante é um modelador de sólidos (para simulação, raciocínio geométrico - trajectórias sem colisões, etc.). Como ponto de partida podem-se considerar modeladores existentes em sistemas CAD que, por terem sido desenvolvidos com outros objectivos, apresentam várias limitações em relação às necessidades da robótica, mas contudo oferecem algumas funções importantes (edição, representação, cálculo de propriedades, visualização gráfica, etc.).

Estando conscientes de tais condicionantes será possível aproveitar a parte "boa" desses componentes sem grandes "concessões".

A integração passa, então, por:

i-conceber e implementar um modelo global de suporte;

ii-um enorme esforço de compreensão das áreas / subsistemas contribuintes no sentido de identificar qual a sua contribuição para o problema e suas limitações;

iii-esforço de adaptação desses componentes.

Por outro lado, para além duma integração funcional quer macroscópica quer localizada (adaptação de interfaces), o problema residirá fundamentalmente numa integração de informação - adaptação de modelos e formas de representação, fluxos de informação, etc..

Off-line vs. on-line

A programação duma estação robótica, principalmente quando encarada numa perspectiva de automatização, é uma tarefa complexa devido às múltiplas situações que têm de ser previstas. Uma abordagem pouco cuidada pode levar a soluções onde a explosão combinatória das diferentes possibilidades torne inviáveis implementações eficientes de sistemas automáticos. Na origem desta situação está o grau de incertezas com que se tem de lidar: acontecimentos inesperados, e uma certa imprecisão nas localizações e dimensões dos objectos, imprecisão de movimentos do robô, etc..

Um problema a ter em atenção é o da altura em que a informação necessária à tomada de decisões se encontra disponível - em antecipação ou em tempo de execução.

Relativamente a este ponto é de notar que o grau de incerteza aumenta com a diminuição da estruturação do ambiente. A consideração deste último factor aparece como uma consequência do desejo de produzir sistemas com grande flexibilidade de resposta. Em relação ao domínio considerado (montagem), e embora tendo em atenção que se pretende um progressivo aumento do grau de flexibilidade dos sistemas, tem-se um ambiente moderadamente estruturado. Embora se possam ter "surpresas" e factos desconhecidos ou incertos, muita informação está disponível à priori. Numa primeira aproximação, e exceptuando os imprevistos, apenas alguns detalhes são desconhecidos inicialmente. Por exemplo, a orientação e coordenadas exactas duma peça chegando num tapete rolante apenas serão conhecidas em tempo de execução. Mas pode-se ter à priori uma "ideia" razoável da "zona de chegada" já que a localização do tapete na estação é conhecida. Isto permite, por exemplo, planear - em antecipação - um "esqueleto" de trajectória sem colisões para pegar a peça.

Aponta-se, pois, para uma situação intermédia entre dois casos extremos:

-as (ainda) actuais estações perfeitamente estruturadas onde tudo (ou quase) é determinista (posicionamentos completamente determinados) à priori, mas com elevados custos de estabelecimento;

-o caso de um robô móvel deslocando-se num ambiente desconhecido (situação completamente não estruturada).

Por outras palavras, embora apontando para estações flexíveis, pode-se contar com um razoável grau de estruturação do ambiente.

Esta situação permitirá que grande parte das decisões possam ser tomadas em antecipação (*off-line*).

Contudo, é de notar que um sistema onde todas as decisões sejam tomadas à priori é um sistema em malha aberta, incapaz de utilizar eficientemente realimentação de informação sensorial, o que não é adequado já que, como se viu, há aspectos que

permanecem indeterminados até ao tempo de execução. Ainda mais grave, não parece abusivo afirmar-se que qualquer sistema real sem realimentação tem tendência a degenerar.

Por outro lado, uma solução completamente *on-line* não é - de momento - realista já que levanta problemas de eficiência e teste (dificuldades em obter repetibilidade de situações, segurança, etc.). Todavia uma solução com tomada parcial de decisões em tempo de execução permite eliminar a necessidade de prever todas as possíveis situações de excepção, muitas das quais são consequência de incertezas de posicionamento e dimensionamento, o que pode ser obviado pela aquisição de informação sensorial.

Há, então, que decidir que partes do problema terão de ser transferidas para a fase de execução (*on-line*) - altura em que toda a informação estará (potencialmente) disponível - e que partes poderão ser preparadas em antecipação.

Tem-se pois um conflito entre as questões de eficiência temporal e as oportunidades mais adequadas para a tomada de decisões.

De qualquer forma, a questão da eficiência - embora presente - não constitui a nossa preocupação fundamental nesta fase da proposta. Importa primeiro compreender os problemas e conceber modelos adequados. Numa fase posterior fará sentido pensar em optimizações. Também a evolução tecnológica previsível a nível dos equipamentos permitirá ultrapassar parte das questões de ineficiência.

Genericamente pode-se, então, separar a actividade de programação em duas fases:

i. Programação genérica, realizada em *off-line*, em que é produzido um plano ou programa genérico (esqueleto). Uma actividade de simulação permitirá um primeiro teste deste programa antes de passar à execução real.

ii. Planeamento "dinâmico", realizado em tempo de execução (*on-line*), que inclui o detalhar de aspectos do plano genérico e eventual reformulação de partes deste (tratamento de situações de excepção ou erros) face a informação recolhida sensorialmente.

Um adequado equilíbrio nesta divisão e na flexibilidade do plano genérico determinará - em primeira fase - o sucesso do sistema. Por exemplo, uma variável "localização" pode ser determinada por uma operação "localizar objecto" em tempo de execução. Então, uma expressão da flexibilidade do plano genérico é materializada pelas "variáveis" que apenas assumem valores em tempo de execução.

De acordo com esta aproximação e com a natureza multicomponente da estação pode-se pensar na concepção dum sistema executor baseado numa panóplia de agentes tendo a capacidade de complementar - em tempo de execução - a informação em falta e também tomar decisões (dentro de certos limites), i.e., possuindo capacidade de planeamento local.

A capacidade de resolver parte do problema em *off-line* pode ser, contudo, aperfeiçoada se uma adequada simulação de execução e monitoração puder ser realizada nesta fase - simulação dos agentes actuadores e sensoriais.

Programação implícita ou explícita

O objectivo (imediato) das propostas de "linguagens" de programação nível tarefa era o de conseguir um sistema de programação automática ou implícita onde apenas se tinha de especificar qual o objectivo pretendido. De forma automática seria gerado o programa / plano que levaria a estação a atingir tal objectivo.

Após alguns esforços iniciais logo se verificou que este era um objectivo a longo prazo, mesmo quando se considera toda informação resultante da fase preparatória (concepção/planeamento de sistema).

No extremo oposto podem-se encontrar as abordagens tradicionais onde a programação da tarefa é feita de forma explícita pelo programador e expressa num texto (programa). Esta não é uma tarefa fácil dada a quantidade de detalhes a manipular e a complexidade de lidar com posicionamentos num espaço 3D.

Um objectivo realista - mas, de qualquer modo, constituindo um significativo avanço - passa por colocar o alvo para o sistema de programação num ponto intermédio entre estes dois extremos (Fig. 2.1.2).

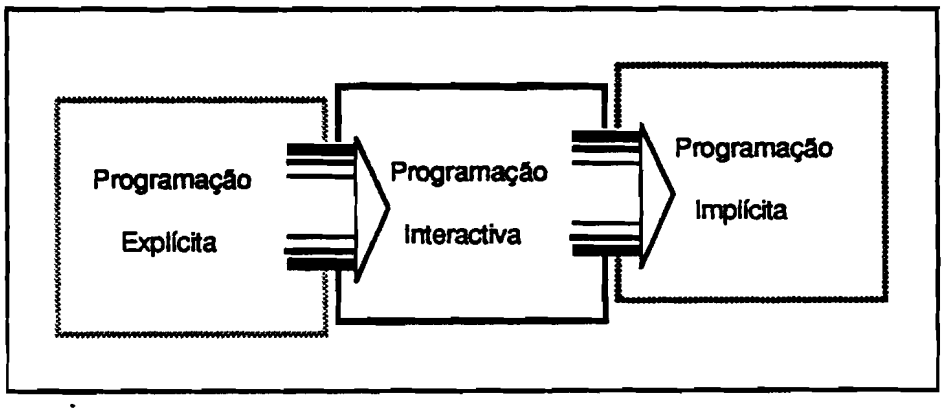


Fig. 2.1.2 Níveis de automatização da programação

Uma solução de programação (interactiva) onde o utilizador vá interativamente cooperando com o sistema na especificação gradual de funções permitirá uma progressiva aproximação do objectivo final.

Para que esta progressão no sentido da automatização possa ocorrer de forma harmoniosa é necessário conceber uma arquitectura para o sistema de programação e controle que, mesmo nesta fase interactiva, contemple já um maior grau de automatização futura.

Nessa arquitectura, os diversos blocos constituintes poderão começar por ser módulos interactivos e progressivamente ser substituídos por módulos "inteligentes" (planeadores especializados, por exemplo).

No contexto dos mais recentes desenvolvimentos tecnológicos, nomeadamente no que se refere aos sistemas gráficos interactivos, parece-nos que esta via é perfeitamente justificada. A disponibilidade de estações gráficas suportando modelos 3D e com diversos meios de interacção (rato, botões rotativos, etc.), permite conceber simulações da estação que sirvam de suporte a uma espécie de ensino guiado de alto nível - onde o operador pode "mostrar" certos aspectos ao sistema (trajectórias esqueleto livres de colisões, detalhe de algumas operações, etc.).

Flexibilidade

Como foi referido, o requisito de flexibilidade constitui uma das principais linhas de orientação para o sistema de programação / controle. Tal flexibilidade e o correspondente desejo de reduzir os tempos de lançamento / alteração de qualquer processo de fabrico (*setup time*), podem ser tentadas por duas vias complementares:

i. automatizando, tanto quanto possível, a actividade de programação;

ii. dotando o sistema de capacidade de monitoração da execução do plano / programa e recuperação de erros.

O problema da detecção e recuperação de erros pode ser considerado a vários níveis, como por exemplo:

-um nível estritamente local, associado à execução de cada acção do plano;

-eventuais níveis intermédios em que a tarefa seja descrita usando operadores abstractos;

-um nível onde seja possível analisar o comportamento global do sistema em execução (pontos de engarrafamento / filas de espera, tempos de fabrico, falhas de equipamento, etc.) e efectuar alguns refinamentos ou optimizações;

-finalmente um nível mais estratégico da malha de controle pode incluir alguma realimentação de informação para o nível do planeamento de sistema possibilitando correcções de actividades nesse nível.

Uma aproximação conceptual segundo este padrão fornece um modelo de referência bastante flexível permitindo alterações / adaptações a diversos níveis. Por exemplo, uma modificação das especificações do produto pode ser encarada como uma alteração a ser "absorvida" ou tratada pela malha de controle mais elevada.

É de referir que muitas aproximações ao problema dos sistemas integrados de programação contemporâneos continuam a basear-se quase exclusivamente em versões de malha aberta. Como exemplos podem referir-se os trabalhos em curso na Universidade de Karlsruhe [FroHoe86], no IPK[SpuFurKir87], na KUKA[WörSta87] e na Renault Automation [Ren87].

Na proposta aqui apresentada, os aspectos de realimentação assumem um carácter absolutamente fundamental.

2.1.2 MODELO DE BLOCOS

Modelo geral de referência

Uma proposta de modelo geral de referência para o sistema de programação e controle [CamSte88], [SteCam88b], tendo em atenção os aspectos referidos atrás, apresenta-se - de forma abreviada - na fig.2.1.3.

Detalhes dos diversos módulos serão apresentados nos capítulos seguintes (3.1 e 3.2). Nesta fase apenas uma breve descrição dos blocos componentes é feita.

•Concepção / planeamento de sistema - este componente representa a fase preparatória desempenhada pelos níveis de concepção do produto, célula e planeamento de processo (e, portanto, fora do âmbito deste trabalho).

•Programação genérica - neste nível será produzido um plano / programa genérico para a realização da tarefa na estação dada.

Começando por ser uma actividade interactiva com forte apoio do especialista humano, deverá caminhar no sentido duma progressiva automatização.

Para além da interacção com o humano, este componente interactuará com outros componentes "especialistas" ou "consultores" (planeadores de trajectórias, por exemplo) que poderão planear em mais detalhe alguns aspectos específicos.

O plano pode ser representado em múltiplos níveis de abstracção. O nível mais baixo representa o programa executável pela estação enquanto o nível mais elevado será o plano de processo.

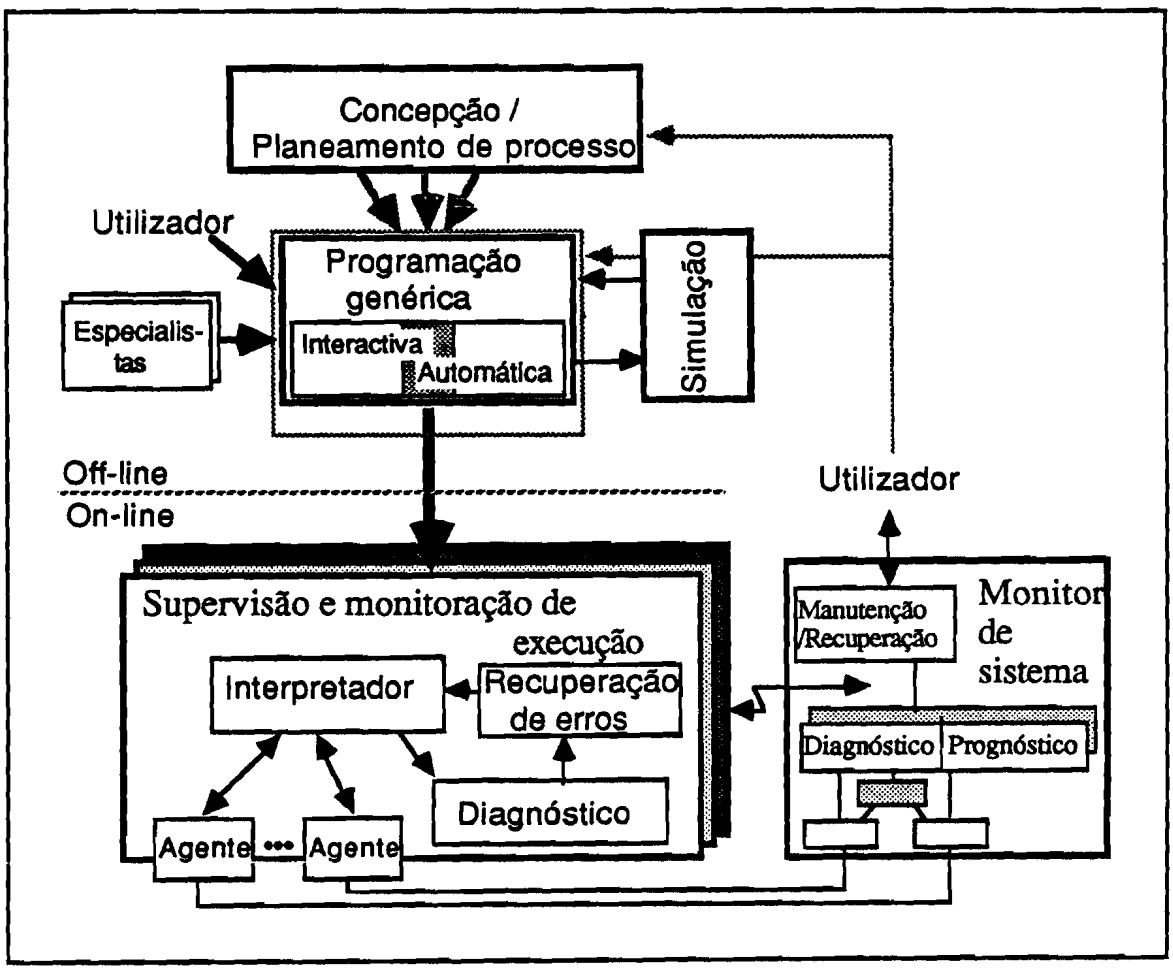


Fig. 2.1.3 Modelo de referência do sistema de programação e controle

•Simulação - módulo que permitirá testar, em *off-line*, o plano produzido. Isto permitirá detectar vários tipos de erros genéricos, por observação visual num sistema gráfico, por exemplo, mas não elimina a necessidade de testes *on-line*. Apenas os reduz.

Serve também como suporte à programação interactiva - através da simulação gráfica da cena pode-se programar por "ensino guiado" de alto nível .

Estas actividades decorrerão fundamentalmente em *off-line*.

•Supervisor de execução - subsistema encarregado da execução com monitoração do plano genérico. Correspondentemente aos diversos níveis de abstracção na representação do plano, ter-se-ão diversos níveis de supervisão de execução.

É constituído por vários submódulos:

•Interpretador - recebe o plano e distribui as acções pelos correspondentes agentes executores, complementa a informação mantendo actualizado internamente um estado do mundo.

•Agentes - são as entidades que executam as acções (constituintes do plano genérico). Estes agentes podem ser actuadores (ex., robô) ou percepcionais (ex., visão).

Considerando um modelo de programação concorrente - os vários agentes poderão, em geral, actuar em paralelo - o que pode levar a uma arquitectura distribuida de módulos cooperantes.

A questão da monitoração pode ser analizada sob duas perspectivas complementares: monitoração de execução e monitoração do comportamento da estação (monitoração de sistema).

•Monitoração de execução - está relacionada com a execução do plano e trata as situações de excepção que eventualmente ocorram. Nesta aproximação admite-se que a fase de programação genérica produz um plano para situações "normais". Isto é, não se considera razoável aumentar a complexidade do plano e a consequente carga computacional apenas para contemplar algumas raras excepções. Assim, cada situação não prevista ou erro de execução será considerado como uma excepção e submetido a tratamento em tempo de execução.

De notar que um sistema com capacidade para tomar decisões em tempo de execução face a situações de excepção (apoiado numa base de regras) é mais flexível que as abordagens de programação tradicional. Num sistema tradicional, embora o fluxo de programa seja decidido em tempo de execução face à informação sensorial, todas as possibilidades tiveram de ser previstas à priori (representadas por uma cadeia de *if's* ou por um *case*).

Na aproximação defendida, o plano genérico é consideravelmente simples, contemplando apenas as situações de funcionamento "normal", o que também facilita a tarefa de geração automática. As situações de desvio em relação a esse funcionamento normal serão tratadas como excepções. Um conjunto de regras permitirá, em tempo de execução, gerar procedimentos de recuperação (planeamento localizado). O carácter modular duma representação de conhecimento por regras facilita a incorporação - numa base incremental - de novo conhecimento sobre estratégias de recuperação (eventualmente por aprendizagem).

Este subsistema inclui:

-Diagnóstico - módulo que permite a detecção e diagnóstico de eventuais excepções, com base em informação sensorial e nos efeitos esperados para a acção em execução. Em cada nível de representação do plano tem de haver conhecimento sobre a tarefa de modo a permitir detectar se os efeitos pretendidos em cada ponto foram atingidos ou não.

-Recuperação - módulo responsável por encontrar um procedimento de recuperação das situações de excepção. Várias estratégias alternativas poderão ser consideradas para cada situação, com base num conjunto de heurísticas.

Quando a recuperação não for possível num nível, o problema é transferido para o nível de abstracção acima (eventualmente para o operador humano).

•Monitoração de sistema - trata da supervisão do comportamento da estação de modo a detectar eventuais falhas ou condições degradantes. Esta monitoração deve ser feita a nível dos componentes mas também a nível de subsistemas (grupos de componentes), visto que os possíveis problemas não dependem exclusivamente de cada componente isolado mas também da forma como esses componentes estão agrupados.

Como principais módulos têm-se:

-Diagnóstico - que permite a detecção e caracterização de situações de falha.

-Prognóstico - Um tópico importante, especialmente relacionado com a perspectiva de monitoração de sistema, é a previsão de potenciais problemas futuros, com base nas

indicações presentes do sistema sensorial e resultados históricos (tendências). O prognóstico permite um controle de qualidade antecipado já que o problema é detectado (suspeitado) num estádio inicial antes de afectar o produto.

-Manutenção / recuperação - módulo responsável por tomar as medidas adequadas a uma situação de falha detectada (ou esperada) no equipamento.

Na formulação mais simples, poderá consistir na geração dum alerta para o operador / responsável pela manutenção.

Pode-se conceber um sistema que, complementarmente, tente algumas medidas de excepção, como a redistribuição da carga de trabalho (se a estação tiver alguma redundância).

Uma situação similar pode ser considerada nos níveis mais abstractos da monitoração de execução, onde se pode pensar numa afinação do desempenho da solução (plano) adoptada.

É de notar que estas últimas actividades estão em sobreposição parcial com as tarefas referidas para o nível de planeamento de sistema, mas o raciocínio é agora baseado em dados reais do sistema e não em valores estimados. Tal raciocínio pode levar a um replaneamento de forma a que se atinja a melhor correspondência estação-objectivo.

Existem alguns efeitos cruzados entre as duas abordagens da monitoração. Por exemplo, um erro na execução do plano pode ser a consequência duma deficiência no equipamento e, portanto, a monitoração de execução pode "disparar" uma acção de monitoração de sistema. Semelhantemente, um problema com um subsistema da estação pode ter consequências na execução, implicando uma redistribuição do trabalho (no caso de redundância funcional) ou mesmo falha do plano.

Comum aos dois aspectos da monitoração é a questão da observabilidade. Para que seja possível uma detecção de erro e seu diagnóstico, tem-se como pré-condição a existência dum ambiente sensorial rico. Adicionalmente, qualquer juizo sobre a situação observada tem de ser baseado numa expectativa relacionada com as consequências desejáveis para cada acção ou um modelo de "bom comportamento" de cada máquina / subsistema.

Metodologia

Em sumário, pressupõe-se neste modelo que a montagem dum produto implica, do ponto de vista da programação / controle, as seguintes fases (fig. 2.1.4):

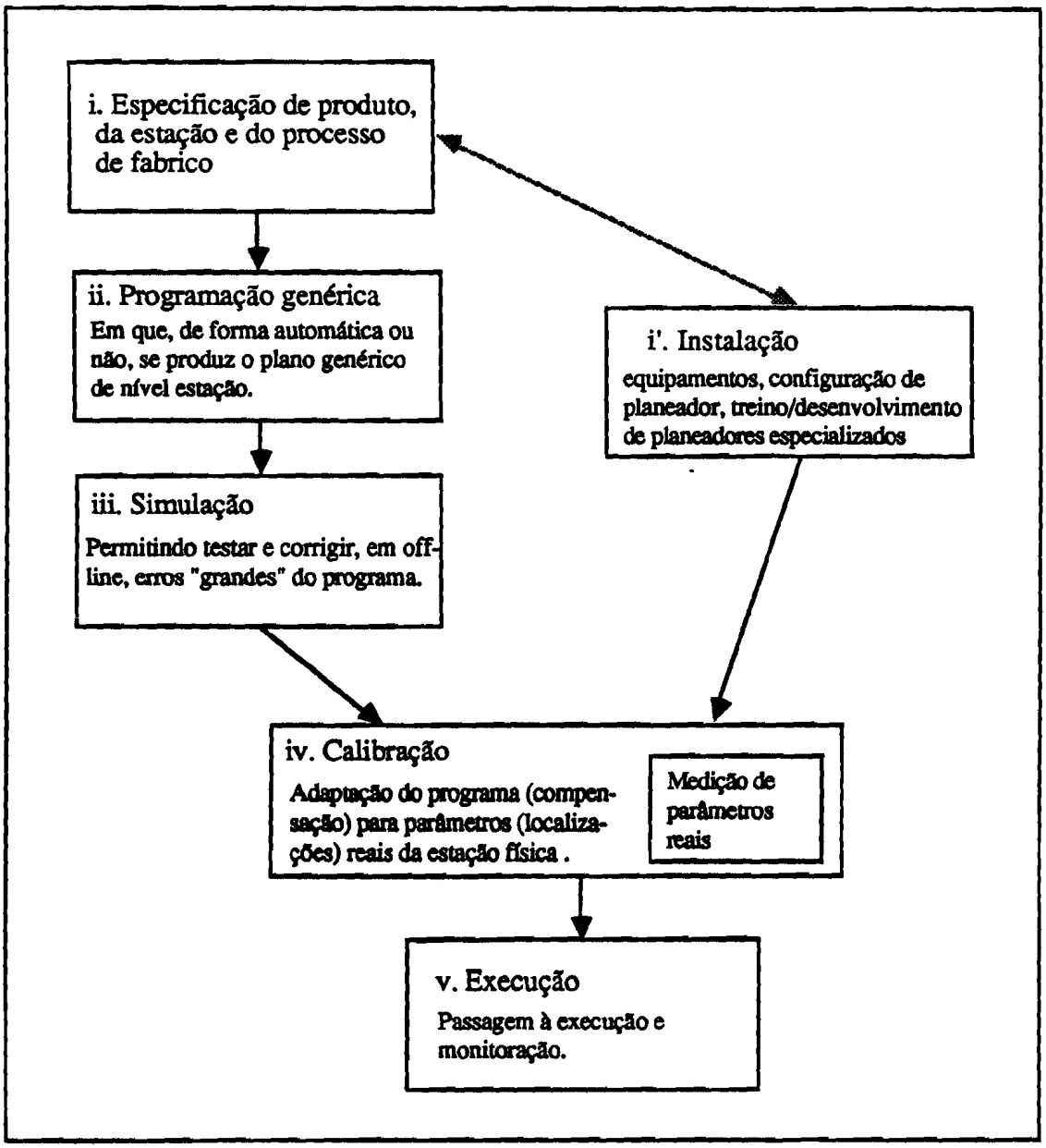


Fig. 2.1.4 Fases para a produção dum produto - perspectiva da programação

Numa situação em que a estação concreta já exista à partida, a calibração pode ser efectuada ao nível dos próprios modelos da estação usados na programação genérica e

simulação. Neste caso o programa gerado em *off-line* estaria de imediato "afinado" para a estação concreta.

2.1.3 EVOLUÇÃO COMPARADA

O modelo descrito atrás teve uma génese que se prolongou por um período temporal de cerca de quatro anos e meio. De forma a poder fazer-se uma comparação entre as propostas apresentadas e o estado da arte nas diversas fases, far-se-à agora uma apresentação da evolução do trabalho durante o período mencionado e sua comparação com outros trabalhos na mesma área.

Aproximações iniciais

Na segunda metade da década de 70 surgiram algumas propostas para sistemas de programação de nível tarefa. Por exemplo, LAMA (1976), AUTOPASS (1977) [Loz83]. Estas propostas, algo prematuras, terão sido derivadas dum certo optimitismo em relação aos primeiros resultados na geração automática de planos no princípio da década. Contudo, todas elas tiveram um relativo fracasso - nenhuma foi concretizada em termos de implementações. Por exemplo, uma das áreas nucleares do sistema pretendido - a geração automática de planos de acções - tem constituido uma importante actividade em Inteligência Artificial, mas as várias propostas iniciais de planeadores limitaram-se a universos muito restritos e idealizados (mundo dos blocos, por exemplo) [Cam87a]. Os problemas do mundo real revelaram-se, contudo, bastante mais complexos que os desse mundo artificial dos blocos.

A complexidade dos problemas encontrados conduziu então a uma concentração de esforços no desenvolvimento de tópicos especializados e considerados de forma isolada, o que ainda hoje tem um peso significativo em muitos centros de investigação. Por exemplo:

-Planeadores especializados de trajectórias sem colisões (*gross motion planning*), de movimentos finos, da acção de pegar numa peça (preensão ?), etc. [Lat84].

-Modelação de robôs (cinemática, dinâmica, etc.).

-A nível sensorial o domínio mais explorado têm sido o da visão [Gev82],[Gev84].

-Noutras áreas "adjacentes", como por exemplo, modelação de sólidos [Req80] também se registaram desenvolvimentos significativos.

No princípio da década de 80 começou a sentir-se a necessidade de integração do que tinham sido esses esforços isolados. Um exemplo significativo é representado pela proposta do sistema TWAIN [LozBro85] que visava:

-Uma primeira integração de alguns resultados de investigação produzidos no MIT, nomeadamente, planeadores especializados de trajectórias, movimentos finos e preensão de peças. Tais módulos deveriam ser integrados através duma filosofia de planeamento genérico baseada numa biblioteca de esqueletos/planos tipo. A abordagem seguida era fundamentalmente em malha aberta fortemente orientada para raciocínio geométrico.

-Fornecer um enquadramento para investigação adicional no domínio de programação a nível tarefa.

Esta proposta ilustra as características típicas presentes noutros trabalhos da época [DilHuc85], [KelBon85], [LauPer85], nomeadamente:

-Integração muito centrada nos desenvolvimentos parcelares anteriores dos centros de investigação envolvidos;

-Pouco gerais, abordagem ascendente (*bottom up*), não integrados num processo mais geral de CIM.

Em 1985 tivemos também oportunidade de fazer uma primeira caracterização da necessidade de integração [SteCam86a] alertando para os riscos de desaproveitamento de muitos esforços especializados devido à multiplicidade de implementações e utensílios usados, ao carácter muitas vezes incompleto dos "produtos" académicos, e à impossibilidade de aceder, de forma fácil, aos diversos utensílios/módulos. Sugeria-se, então, estarem criadas as condições para o desenvolvimento de um ambiente de programação para robótica com as características que permitissem a integração dos desenvolvimentos anteriores num único sistema e simultaneamente fornecesse o suporte para desenvolvimento e teste de novas técnicas de programação.

Numa perspectiva de curto prazo propunha-se uma integração baseada numa aproximação de programação explícita centrada numa linguagem existente (Concurrent Pascal ou Modula). O objectivo imediato era o de, com um investimento mínimo, tornar facilmente acessíveis, num mesmo ambiente, vários componentes (simuladores, CAD, sistemas de processamento de imagem, etc.) que se antevia deverem facilitar a compreensão do problema na perspectiva de evolução para níveis superiores de abstracção. À primeira vista, esta panóplia de componentes parece pouco conveniente dado pertencerem a níveis muito diferentes. Contudo há que considerar as situações de

facto existentes, o que já não deixa muitos graus de liberdade quanto à divisão dum sistema em módulos.

Dificuldades internas fundamentalmente de ordem organizativa da equipa e a própria evolução de perspectivas impediram que esta experiência fosse terminada mas os pontos referidos foram importantes para o marcar de direcções para o trabalho subsequente.

Arquitectura integrada

Uma proposta de arquitectura integrada para a programação de células robóticas feita em 1986 [CamSte86a] [CamSte86b] [SteCam87] já enunciava grande parte das questões da área e apontava várias direcções de solução (fig. 2.1.5).

Comparando as fig.s 2.1.3 e 2.1.5 verifica-se que muitas das ideias de integração já estavam presentes nesta 1primeira proposta, denominadamente em termos macroscópicos.

Como zonas mais deficientes tinham-se:

-Uma ainda pouco clara ligação com os níveis mais gerais do CIM, embora antevendo já as necessidades funcionais dum "Configurador de Estação" ou dum "Especificador de Tarefas".

-Aspectos de monitoração pouco desenvolvidos e só num nível (planeador dinâmico, reparador); o módulo de diagnóstico apenas previa interacção com o operador (casos de emergência) e não estava ligado ao processo de recuperação.

Simulação

A questão da simulação e sua importância como componente do sistema de programação foi também acentuada neste período.

Como principais justificações podem referir-se o possibilitar o teste e eliminação de grande parte dos erros dos programas / planos antes de serem submetidos à estação real. Isto permite reduzir riscos (destruição de componentes, acidentes pessoais, etc.) por erros grosseiros do programa. Por outro lado, faculta a possibilidade de realizar tais testes mesmo antes de se ter instalado a estação ou se dispor das peças componentes a montar.

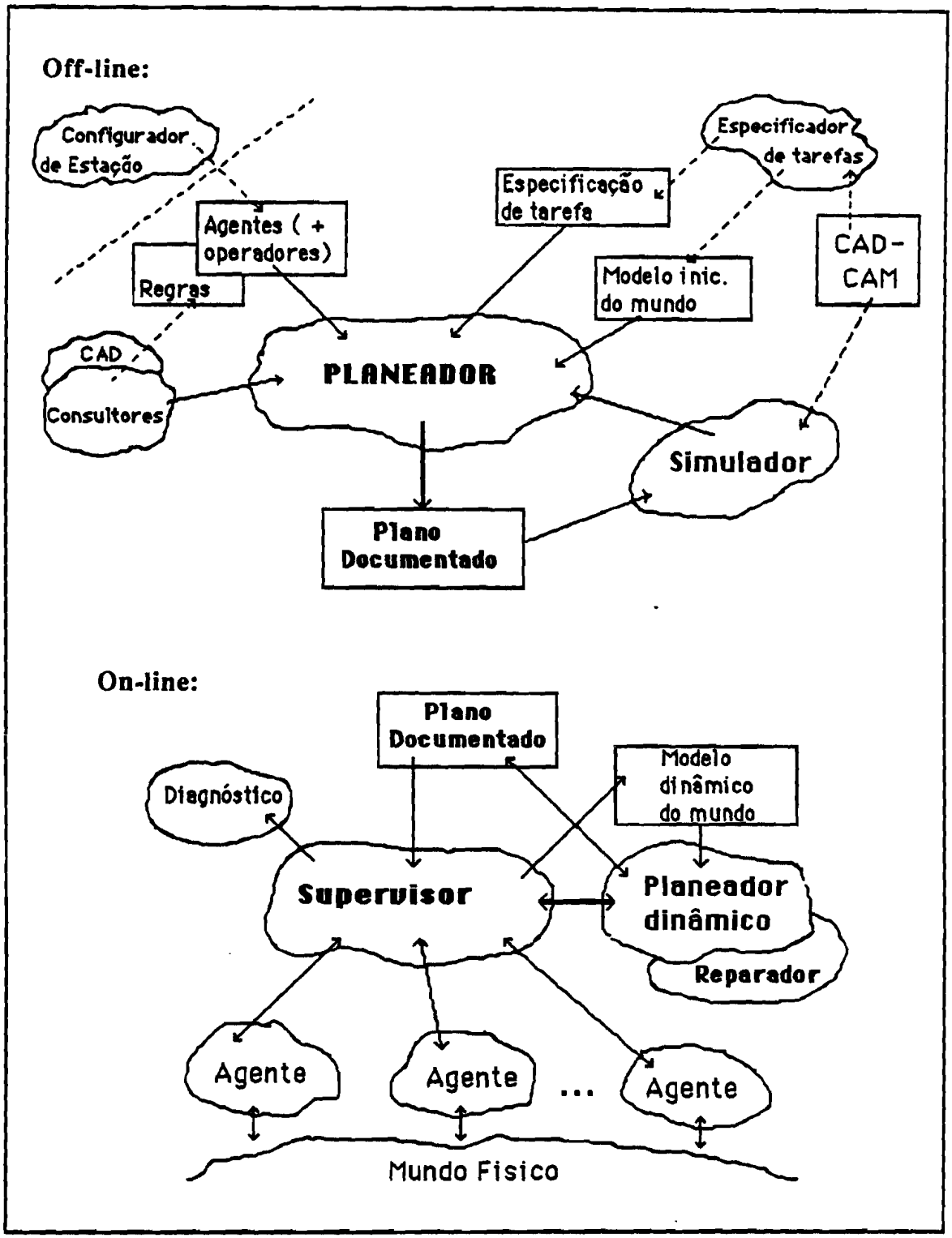


Fig. 2.1.5 Sistema de programação - primeira proposta

Complementarmente à programação *off-line*, a simulação permite que grande parte do trabalho seja preparado sem ter de parar a estação.

De entre os aspectos verificáveis durante a simulação podem referir-se:

-Sequência geral de operações.

-Possibilidade de atingir todas as posições necessárias; eventual experimentação com vários *layouts*, etc.

-Interferências - trajectórias livres de colisões.

-Fluxo de peças, ferramentas, etc.

-Sincronização entre componentes.

-Estimativa de *cycle times*.

Complementarmente e, em especial através da visualização gráfica, uma aplicação da simulação é a do "convencimento" do cliente ou apresentação dos conceitos ao potencial utilizador [AdlRod87]. Ou, noutra perspectiva, como auxiliar à especificação / validação do modelo funcional pretendido.

Uma estrutura então proposta para o simulador é representada pela fig. 2.1.6, ou seja, uma estrutura análoga à do subsistema executor (*on-line*), mas em que os agentes (actuadores ou sensores) estão conectados a um mundo artificial (simulação do mundo real).

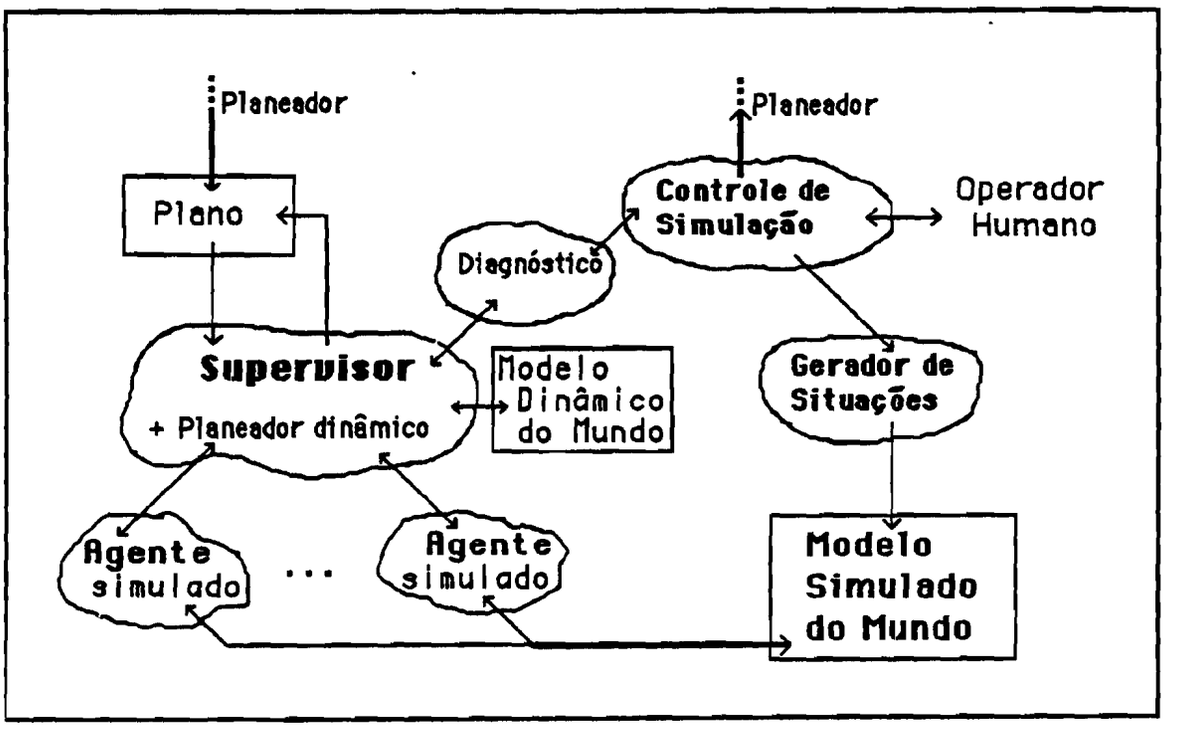


Fig. 2.1.6 Subsistema de simulação

Alguns novos componentes:

• Controle de Simulação: módulo que, num diálogo com o humano, permite estabelecer as condições de simulação. Actua como elemento de controle geral, interactuando com os módulos de diagnóstico de situações de erro e de estabelecimento das situações de teste, e gera pedidos de correcção ou replaneamento.

• O mundo real é agora substituído por um modelo artificial - simulação do mundo real. É neste modelo que se podem definir / estabelecer as diversas hipóteses de cenas sobre as quais se pretende testar a adequação do plano produzido. (Não confundir com o modelo interno do mundo mantido pelo supervisor de execução). "Filtros" podem ser introduzidos simulando as imprecisões dos objectos reais, por exemplo.

• Gerador de situações: módulo que permite, segundo orientação do controle de simulação, estabelecer os cenários simulados, ou seja, "construir" novas cenas hipotéticas que servirão de suporte ao teste do plano em produção. Para além de algumas funcionalidades similares às disponíveis num sistema CAD, deverão ser facultados alguns operadores geradores de "imprecisões" para contemplar possíveis tolerâncias em dimensões, posicionamentos e orientações. Outro aspecto a contemplar é o do dinamismo da cena: Será importante, em termos de teste *off-line* , verificar como o plano se comporta face à coexistência de várias entidades físicas actuando em paralelo.

• Cada agente simulado é baseado num agente real (usado na fase *on-line*): acopla-se a este uma interface para o mundo artificial. A natureza desta interface dependerá obviamente do tipo de agente (actuador ou sensor). Na fig.2.1.7 apresenta-se um detalhe englobando um agente actuador e um sensorial, que poderão ser conectados ao mundo real ou ao mundo artificial (através das adequadas interfaces).

Quando conectados ao mundo real (fase *on-line*), suas ordens são dirigidas aos controladores do respectivo actuador (robô, transportador, alimentadores). Se conectados ao mundo artificial (fase *off-line* : produção /teste do plano), as ordens são captadas pela interface "Simulador de Acção" que produzirá uma modificação (simulação da acção) no mundo simulado. (No cap. 3 ver-se-à como este conceito é facilmente materializável com a noção de contexto.)

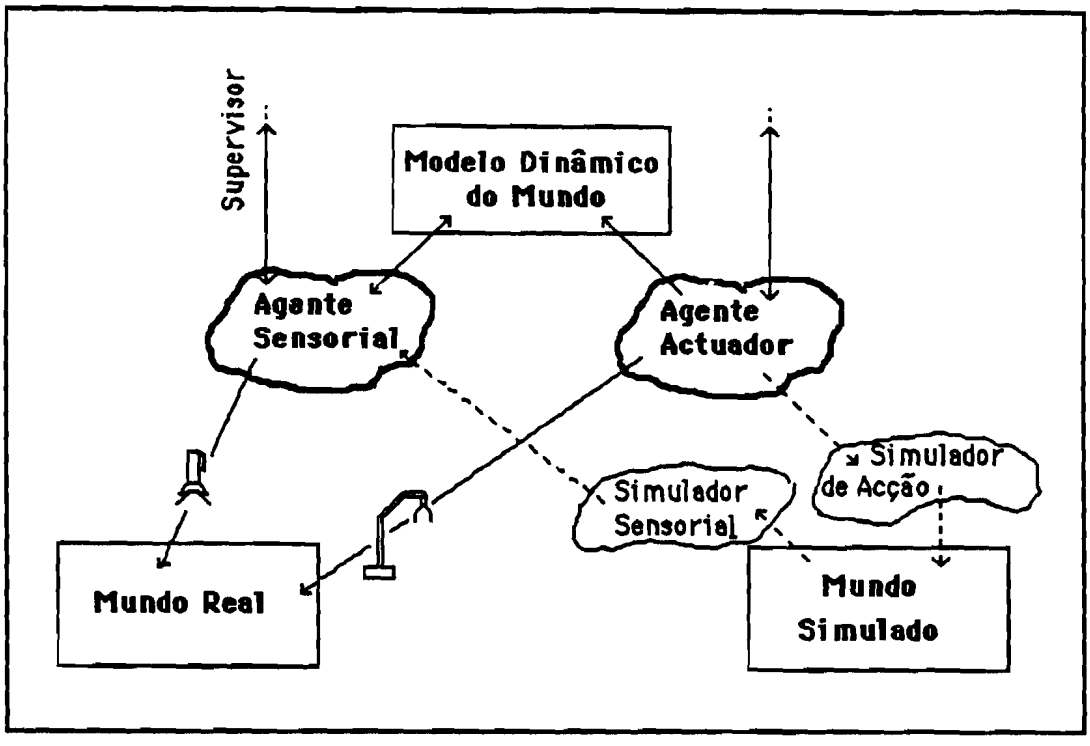


Fig.2.1.7 Conexão de agentes ao mundo real ou simulado

Como é difícil modelar as incertezas do mundo físico, esta simulação da acção poderá ser influenciada por um agente externo manual (operador humano, por exemplo) ou automático (gerador aleatório), de forma a poder introduzir alguns graus de imprecisão nos efeitos das acções.

Para além da simulação dos diversos agentes actuadores da estação (robô, tapete rolante, etc., onde os aspectos geométricos e, eventualmente cinemáticos e dinâmicos, são importantes, e que têm sido tradicionalmente tratados com alguma preferência), considera-se também nesta fase da proposta a simulação dos agentes sensoriais (visão, tacto, ...), um domínio pouco explorado. Esta simulação deverá, a nível lógico, substituir-se no modelo à informação normalmente obtida através dos agentes sensoriais na fase *on-line* .

Um dos principais órgãos sensoriais em qualquer estação robótica será a visão. Para poder simular este órgão é necessário que, a nível do modelo do mundo artificial, se possam modelar cenas 3D. O simulador deverá poder extrair "imagens" dessa cena de quaisquer (ou de vários) pontos de vista, fornecendo como resultado o mesmo tipo de informação que a câmera (fig. 2.1.8). A nível da modelação da cena, para além dos aspectos geométricos, dever-se-ão considerar também elementos (atributos) como luminosidade (e fontes de luz), cor, reflectibilidade, etc, associados aos objectos e que

podem constituir um factor importante na sua identificação. Relativamente a outros sentidos, como o tacto, importa também pensar em possíveis esquemas para a sua simulação.

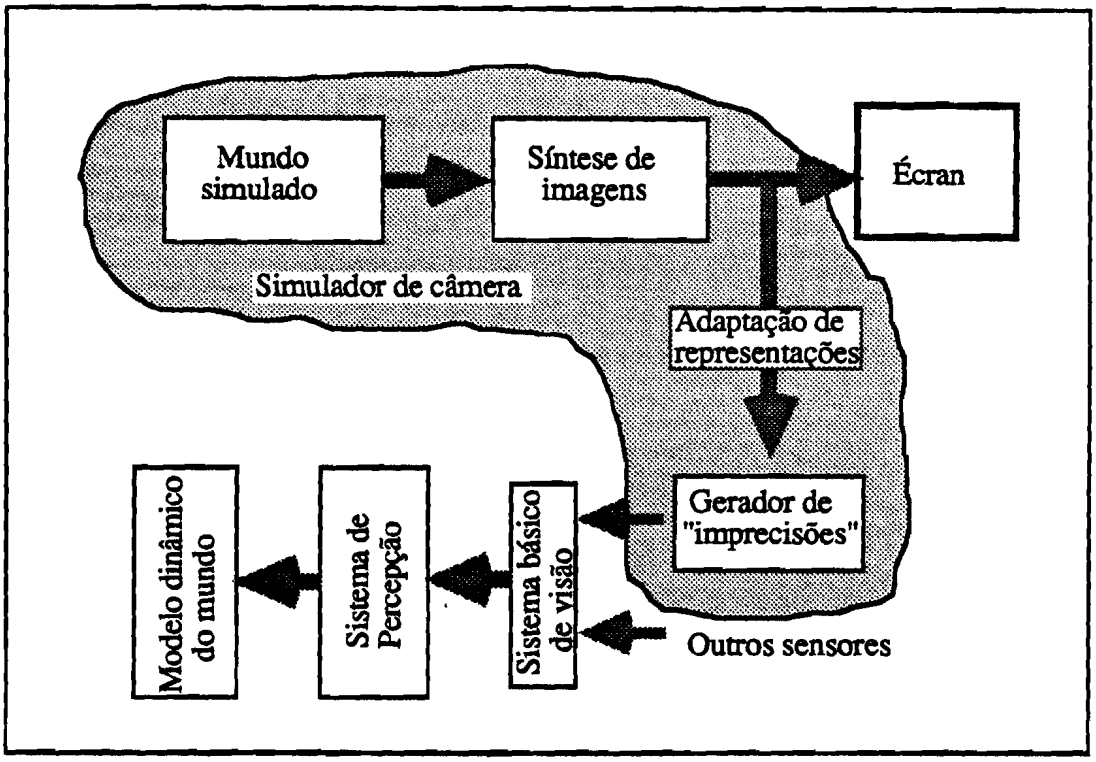


Fig.2.1.8 Simulação de câmera (visão)

Estes aspectos de simulação sensorial são importantes mesmo numa situação de programação automática como suporte ao treino do sistema percepcional.

Alguns trabalhos posteriores, no domínio da percepção baseada em visão, que vieram a ter lugar no Grupo de Robótica ilustraram a consistência destas ideias [SteSanQue87].

Em termos de produtos, surgem nesta fase algumas propostas importantes.

É o caso dos sistemas MRS/McDonnell Douglas [McD85] e ROBCAD [Adl86], [AdlRod87].

Facilidades para a especificação / simulação da estação são aspectos enfatizados nestes sistemas. Assim, um dos módulos permite a modelação geométrica e cinemática dos diversos componentes, indicação de "*layout*", visualização gráfica e verificações de

interferências, capacidade do robô para atingir as diversas posições de trabalho pretendidas, etc..

Para obviar ao problema de multiplicidade das linguagens de diferentes robôs, alguns simuladores usam uma linguagem neutra que, depois, é compilada para as linguagens reais.

Para facilitar a passagem do programa testado nesta "estação" modelo para a estação real, alguns fornecem funções de calibração.

Estes sistemas contemplam um único nível de simulação (um único nível de abstracção), localizado ao nível dos conceitos das linguagens presentes nos actuais controladores de robôs e são, em geral, bastante "fechados" relativamente à possibilidade de os aumentar ou integrar como componentes doutro sistema.

Quanto à simulação sensorial, só muito ao de leve é abordada e a um nível simbólico (o utilizador é solicitado a introduzir os valores).

A nível de investigação, um exemplo significativo é o ROSI desenvolvido na Universidade de Karlsruhe [DilHuc85][Rem86] em torno do modelador de sólidos ROMULUS e da linguagem SRL, e que inclui:

-Modelador e emulador da estação (e seus componentes);

-Módulo de programação textual e gráfica interactiva (*teach in*);

-Módulo de simulação e animação gráfica.

Uma breve panorâmica doutros simuladores pode ser encontrada em [Lau88].

Agentes

Um detalhe dos agentes do sistema executor (quer actuadores, quer percepcionais) é apresentada na fig.2.1.9 [SteCam86b] [CamSte86b] onde se podem considerar os seguintes módulos:

• Interpretador de comandos é o módulo que recebe comandos do supervisor de execução a serem executados pelo agente e que, eventualmente, devolve informação obtida com tal execução (ou indicação de erro).

• Planeador especializado - módulo que detalha uma acção genérica em subacções capazes de serem realizadas pelos componentes "físicos". Por exemplo, o comando "pegar peça" pode ser decomposto em "aproximar", "fechar garra" e "afastar", ou algo mais complexo.

Tal permite assegurar - a este nível - a flexibilidade do plano. Estas acções podem ser de actuação sobre o mundo ou de aquisição de dados sobre o estado desse mundo.

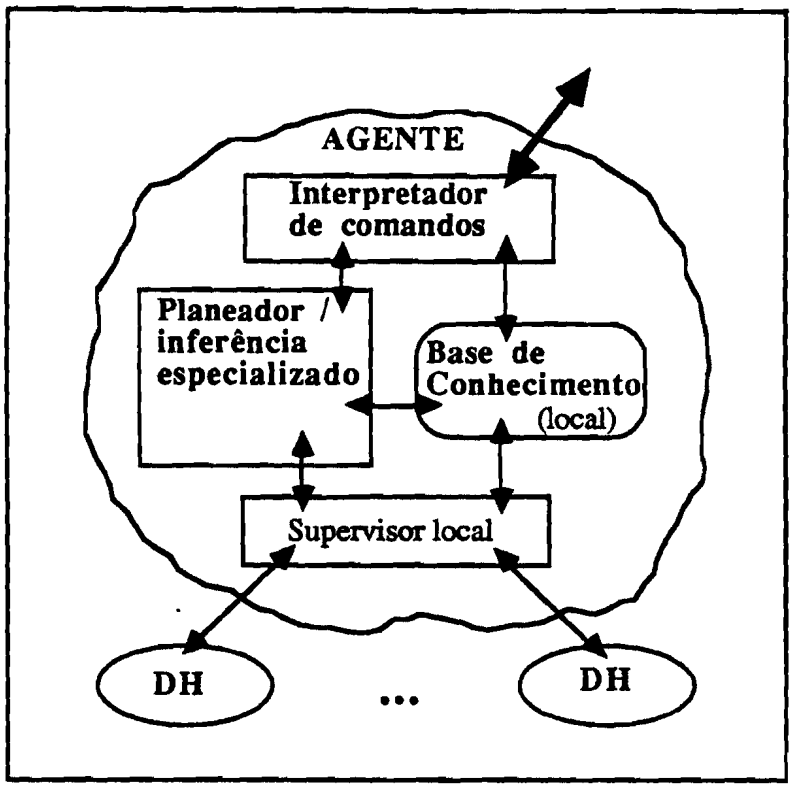


Fig.2.1.9 Estrutura de agente

Neste modelo, os agentes são dotados de capacidade de decisão local representada pela parcela de conhecimento (BC local) que é preparada durante a fase de instalação do sistema. Esta aproximação constitui uma alternativa à necessidade de geração de planos condicionais.

• Base de conhecimento (BC) local suporta o conhecimento especializado do agente. De notar que sendo o "comportamento" do planeador especializado gerido por esta BC, tem-se maior flexibilidade para a alteração das estratégias de planeamento local. Pode-se inclusivamente pensar em bibliotecas de BCs especializadas para diversos tipos de aplicação sendo, na fase de instalação da estação, seleccionada a base adequada.

• Supervisor local - componente que se encarrega da execução das acções detalhadas controlando os componentes físicos (DHs).

• Controladores de dispositivos (*Device handlers*, DHs) - representam abstracções dos componentes físicos. Por exemplo, controlador de robô (interpretador de linguagem nível manipulador) ou sistema básico de visão. Correspondem a entidades físicas bem

determinadas com capacidade de processamento local e, consequentemente, capazes de executarem um conjunto básico de acções primitivas (por exemplo, movimentação do robô ou extracção de características como arestas, áreas, perímetros, etc., num sistema de visão).

Este é o modelo genérico, porém nalguns casos, alguns destes componentes podem ser consideravelmente simplificados. É, contudo, de realçar um certo paralelo entre este modelo e o sistema global, o que permite desde logo apontar para uma possível generalização a um sistema de controle suportado em múltiplos níveis de abstracção.

Aspectos sensoriais

Os aspectos sensoriais e sua integração no sistema global foram objecto dum refinamento [SteCam86b][MouSteCam86] à proposta apresentada inicialmente onde se defende uma aproximação "baseada em conhecimento" para o problema da integração multi-sensorial.

Como ponto de partida tomou-se o problema clássico da identificação de objectos.

A aproximação proposta nesta fase contempla um processo a três fases (fig.2.1.10):

i-"Ensino" ou treino do agente percepcional;
ii-Reconhecimento;
iii-Refinamento baseado em avaliação de resultados.

A primeira fase tem por objectivo a criação da base de conhecimento local do agente. Um dos principais problemas é o da escolha do conjunto de atributos característicos do objecto que devem ser considerados na identificação. Esta selecção deve ser função do conjunto de objectos com que o sistema tem de lidar (na tarefa corrente), das características da célula (que sensores e que características pode extrair com que custo e com que confiança) e de informação sobre os tipos de tarefas. Outros problemas resultam do carácter impreciso da informação sensorial, o que obriga a "treinar" o identificador com vários exemplos em diferentes condições ambientais (diferentes estados de iluminação da cena, por exemplo).

Na segunda fase serão activadas as acções de aquisição de informação sensorial de acordo com o objectivo pretendido (fornecido pelo supervisor de execução) e com a estratégia estipulada na base de conhecimento local.

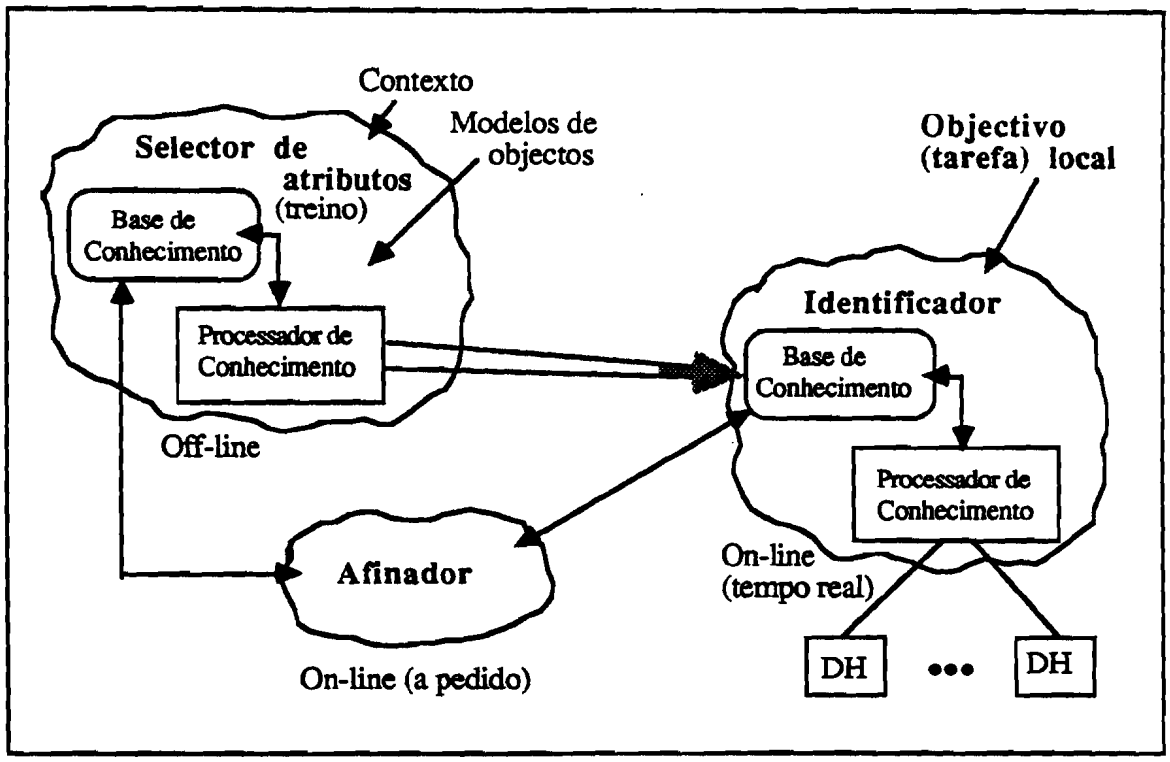


Fig.2.1.10 Agente percepcional

A construção da base de conhecimento local, ou seja, a "compilação" desse conhecimento a partir dos vários elementos referidos acima, deve tender para uma progressiva automatização como forma de aumentar o grau de integração do sistema.

Alguns algoritmos de indução de regras a partir de exemplos têm sido propostos (ID3, por ex.) e apontam algumas direcções para tal automatização [ThoTho86]. Um exemplo é representado pelo Rulemaker - componente do RuleMaster, uma ferramenta para a produção de sistemas periciais [Rad86] - que induz regras automaticamente a partir dum conjunto de exemplos (uma forma de aprendizagem). Os exemplos são fornecidos sob a forma de tabelas de decisão, onde uma das entradas pode ser formada (neste caso) pelo conjunto dos atributos possíveis e a outra pelas conclusões (que objecto) a extrair de cada combinação de atributos. O Rulemaker, ao analisar tais tabelas, tem em conta o poder discriminante dos diversos atributos ao estabelecer a cadeia de "*ifs*", ou seja, usa tal informação para sugerir uma ordem para a extracção de informação sensorial. Também identifica atributos redundantes que não serão incluídos nas regras.

Contudo, algumas experiências com tal utensílio evidenciaram que ele usa uma estratégia demasiado rígida na ordenação de regras baseada exclusivamente no "poder discriminante" dos atributos. Para o reconhecedor é conveniente uma estratégia mais

flexível que nomeadamente tenha em conta a disponibilidade dos sensores, seus custos de extracção e credibilidade (qualidade).

Por exemplo, suponha-se que uma conclusão pode ser atingida por duas vias diferentes: uma baseada num único atributo e outra baseada em vários. Sistemas como o Rulemaker optariam pela primeira via mesmo que o atributo envolvido requeresse um custo de extracção superior à soma dos custos da via alternativa, já que não têm essa informação em consideração.

Noutra perspectiva, especialmente porque o conhecimento é impreciso, a utilização de factores de confiança pode justificar que se tente provar uma conclusão por diferentes vias com a intenção de assegurar uma maior confiança nessa conclusão.

Uma estratégia tendo em atenção alguns destes factores foi apresentada em [HirAraHac87]. Numa primeira fase é determinado o conjunto mínimo de características discriminantes com base no poder discriminante da característica, frequência e tempo de extracção. Uma árvore de reconhecimento é depois construída com base na frequência (esperada) de ocorrência dos objectos.

De qualquer forma, a noção de integração do ambiente sensorial no contexto mais geral do sistema de programação e controle e automatização do carregamento da sua base de conhecimento constituiram inovação da nossa proposta relativamente ao tradicional. Também na apresentação duma aproximação baseada em conhecimento foi uma das primeiras.

Uma proposta contemporânea, da Universidade de Purdue, EUA, apresenta também um plano de trabalho para o desenvolvimento dum sistema de controle duma estação robótica de montagem [KakBoy86] com algum nível de integração. A ênfase desta proposta centra-se fundamentalmente na integração da componente sensorial no sistema de controle *on-line* e no planeamento de movimentos. Deste modo, não são praticamente explorados os outros problemas do sistema de programação nem a sua abordagem num contexto CIM. Representa, contudo, uma das primeiras abordagens ao problema da integração multisensorial com o objectivo específico de suportar a flexibilização da estação. A aproximação seguida é também a dum sistema baseado em conhecimento.

Monitoração

A monitoração de execução constitui um tópico que tem sido relativamente pouco explorado em termos da Robótica Industrial. Considerando instalações concretas verifica-

se que tais aspectos têm estado praticamente ausentes. Por outras palavras, ao nível de abstracção do programa os sistemas instalados funcionam praticamente em malha aberta (embora algumas zonas de monitoração possam existir para certas operações muito específicas).

A procura de sistemas mais flexíveis, com maior grau de autonomia e funcionando em ambientes menos estruturados, leva a que recentemente se tenha vindo a colocar uma certa ênfase no assunto.

A nível dos geradores de planos, algumas experiências de monitoração de execução foram realizadas desde cedo, como no caso do STRIPS/PLANEX [FikHarNil81]. Este trabalho permitiu evidenciar algumas das questões:

-necessidade de conhecer, em tempo de execução, quais os efeitos desejados de forma a ser possível avaliar do sucesso das operações;

-consideração de "surpresas" negativas e positivas, podendo levar a replaneamento ou re-escalonamento das acções.

Contudo, a abordagem realizada era bastante simplista - dirigindo-se no "mundo dos blocos".

De entre as primeiras aproximações ao problema no campo da Robótica, são de referir os trabalhos na Universidade de Minesotta, USA [GinGin83] [Gin86], e na University College of Wales [LeeBarHar83] [HarBarLee86]. Estes trabalhos permitiram colocar o problema num contexto mais realista (por oposição ao mundo dos blocos). Em ambos os casos foi seguida uma aproximação baseada em regras para o diagnóstico e para a recuperação das situações de excepção. Reflectem, contudo, abordagens muito focalizadas sem qualquer integração num sistema global de programação. Por exemplo, no sistema de Gini, parte-se dum programa VAL que tem de ser manualmente complementado com informação suplementar sobre o objectivo de cada segmento de programa.

Em termos do nosso trabalho, a questão da monitoração de execução e recuperação de erros, foi considerada, desde início, como elemento do sistema de programação e controle [CamSte86a], [CamSte86b] (ver fig. 2.1.5), embora na primeira fase ainda de forma pouco elaborada, como já foi referido. Por exemplo, uma das deficiências nessa estrutura era o não haver qualquer ligação entre o módulo de diagnóstico e o subsistema de recuperação.

Em [SteCam88a] e [CamSte88], um refinamento da arquitectura inclui uma análise mais aprofundada destas questões. Embora seguindo uma aproximação similar a [GinGin83] e [LeeBarHar83], no que respeita à utilização duma base de regras para diagnóstico e recuperação, a principal preocupação aqui é com os aspectos de integração: como materializar estas funções no contexto do sistema integrado (em termos de estrutura

funcional e requisitos de informação). Um primeiro protótipo, usando regras OPS para diagnóstico e regras Prolog para recuperação, integrado no sistema global, é apresentado nesta fase. Uma primeira referência à necessidade duma estrutura multi-nível para a monitoração bem como a questão da reconfiguração ou redistribuição de trabalho em caso de falha de componentes da estação, são incluídas na proposta de arquitectura. A previsão de eventuais futuros problemas (prognóstico) é também considerada. Não havia, contudo, ainda uma nítida divisão entre os aspectos de monitoração de execução e monitoração de sistema.

Finalmente, em [SteCam88b], é apresentado um estudo mais detalhado das questões de monitoração, de acordo com o modelo final desta proposta.

Um outro exemplo de trabalho contemporâneo nesta área encontra-se em curso na Universidade de Amsterdam [KuiDui...86],[MeiDui...86]. Também aqui é seguida uma aproximação baseada em regras (Prolog) para diagnóstico e recuperação e uma proposta de classificação das situações de excepção a considerar nos domínios da montagem foi desenvolvida. Como principais limitações do trabalho podem referir-se o facto de considerar exclusivamente o robô (ignorando os outros componentes duma estação) e não ser integrado num sistema global (nenhuma ligação ao sistema de programação, nem conexão aos componentes reais).

Finalmente é de referir que um esforço de integração de ideias derivadas da nossa proposta e dos trabalhos das Universidades de Amsterdam e de Karlsruhe foi iniciado no âmbito dum projecto Esprit [SteCam...88d].

Um aspecto relacionado com a monitoração é o da aquisição e integração de informação multisensorial. Esta questão tem sido muitas vezes encarada em sentido demasiado genérico, com alvos pouco dirigidos. Neste trabalho pretende-se uma integração no contexto do controle de execução em sistemas CIM. Isto inclui identificação / localização de objectos, mas também detecção de estados de erro e antecipação da evolução do sistema. O diagnóstico / prognóstico depende da capacidade de observação do sistema e, portanto, das capacidades sensoriais. A informação adquirida será tratada não apenas em função do seu significado "isolado" mas integrada com os modelos (conhecimento prévio) do sistema, tarefas ou objectivos correntes, expectativas e historial (tendências).

• • •

Feita esta primeira abordagem segundo uma perspectiva dos módulos funcionais do sistema, far-se-à agora uma apresentação dual desta sob a óptica das questões de modelação ou estruturas de informação.

2.2 - SISTEMA DE INFORMAÇÃO

2.2.1 - INTRODUÇÃO

O SI como elemento integrador

A integração de informação tem sido identificada como uma questão fundamental para a realização dum sistema CIM [Nei87], [CamSte87a]. Desta forma, um Sistema de Informação (SI) constituirá o principal elo de integração dos diversos componentes da arquitectura proposta.

No caso da célula robótica, como se descreveu anteriormente, aos diversos blocos funcionais podem ser associadas várias áreas científico-tecnológicas que têm tido desenvolvimentos isolados, conduzindo a resultados nem sempre convergentes. Um esforço de integração de tais blocos passa pelo estabelecimento duma arquitectura de informação e pela identificação dos fluxos e necessidades de dados e conhecimento que se colocam aos diversos níveis.

A posição que aqui se defende é a de que, mais do que uma integração funcional, o problema é fundamentalmente o da integração de informação. O SI suportará não só a integração dos vários módulos do sistema de programação e controle, mas deverá constituir também, o elo de ligação do subsistema estação robótica com outros componentes do sistema CIM. A informação necessária, directa ou indirectamente, ao processo de programação é proveniente de múltiplas fontes (até ao momento com desenvolvimentos de modo isolado). Apesar dos trabalhos de investigação que têm vindo a ser realizados na área de planeamento de sistema, a geração da informação útil para a

programação ainda é feita nesse nível com pequeno grau de automatização. Como consequência directa temos uma incompatibilidade de meios de suporte e de representação que dificultam o processamento subsequente. Por vezes certas especificações são apresentadas em suporte de papel (tendo o humano por destinatário) e, portanto, nada adequadas a uma automatização da programação. Adicionalmente, o processo de transferência pode ter "perdas", isto é, alguma informação gerada em fases separadas e que poderia ser útil para um sistema de programação automática ou interactiva não é explicitamente disponível -porque tais sistemas não anteviam tal objectivo de integração - ainda que essa informação existisse durante as fases preparatórias de especificação da tarefa.

Mesmo quando a informação resulta de um processamento informático, tem-se muitas vezes dificuldade no acesso à sua representação interna. Por exemplo, num sistema CAD nem sempre é fácil ter acesso aos dados representados na respectiva base de dados (modelos geométricos) através de outro programa.

Alguns passos importantes no sentido de ultrapassar esta situação têm sido materializados pelos esforços de padronização de informação CAD. Assim, surgiram propostas como IGES, CAD*I, PDGES, etc. [Byt87],[Sch87]. Contudo, estes padrões visam essencialmente a troca de informação entre sistemas CAD através de ficheiros e, mesmo a esse nível, não resolvem todos os problemas (por exemplo, quando dois CADs usam diferentes formas de modelação de sólidos). A utilização de informação CAD para a programação e controle de estações robóticas faz apelo a certas características que, embora deriváveis dos modelos geométricos, não estão directamente explicitadas em tais modelos nem contempladas nestas propostas de padrões. Por outro lado, a existência de múltiplas funções de processamento geométrico nos sistemas CAD pode sugerir uma ligação interactiva entre tais sistemas e o sistema de programação e controle (não contemplada, pelo menos de forma satisfatória, nas propostas de padrões).

Também a nível de modelação de componentes de estações robóticas, existem actualmente várias bibliotecas de robôs e outros componentes. Contudo os formatos de representação e modelos utilizados não estão normalizados, sendo geralmente estabelecidos para cada caso específico.

O problema é semelhante quando se pensa na especificação do processo de fabrico. Não existem linguagens de especificação padronizadas e genericamente aceites que facilitem a passagem da fase de planeamento de processo para a programação. Uma proposta alemã de "linguagem gráfica" [Vdi82] tem tentado tornar-se um padrão mas não parece ter ainda uma grande aceitação, subsistindo também o problema do suporte informático para uma tal representação.

Por outro lado, a nível do controle de execução, para que sejam possíveis uma efectiva monitoração e capacidade de recuperação de situações de erro é necessário ter disponível informação gerada na fase de programação.

A materialização dum sistema integrado de programação e controle passa, então, pelo estabelecimento duma arquitectura de informação onde se clarifiquem as necessidades e interdependências aos diversos níveis.

Tipos de necessidades

Dos capítulos anteriores facilmente resulta a identificação de várias "famílias" ou tipos de informação a contemplar pelo SI:

-Geométrica - modelo das peças / produto, modelo geométrico do robô, localização dos diferentes componentes na estação, etc..

-Cinemática e dinâmica - modelos dos componentes activos (manipulador, transportador, ...).

-"Tecnológica" - restrições, como esforços máximos admitidos pelas peças, tolerâncias, ferramentas associadas a processos, etc..

-Regras de identificação de objectos, diagnóstico e recuperação de erros.

-Etc..

Por outro lado, um mesmo elemento pode ser visto sob diversas perspectivas. Por exemplo, o robô pode ser caracterizado por:

-modelo geométrico e cinemático / dinâmico;

-lista de operações de alto nível e relações do robô com outros componentes da estação- modelo funcional;

-envelope ou geometria simplificada envolvente (para detecção de colisões);

-carga, repetibilidade, precisão, etc..

Uma vez que os diferentes elementos de informação ou as diferentes "visões" dum mesmo elemento são utilizados ou gerados em alturas diferentes torna-se necessário estabelecer uma relação entre os fluxos de informação e o modelo funcional do sistema de programação e controle.

Por outro lado há duas facetas quanto à "volatilidade" ou taxa de alterabilidade / utilização (também relacionado com tempos de acesso - *on/off-line*) da informação:

-Estática - só esporadicamente modificada (ex., bibliotecas de componentes).

-Dinâmica - modelação do estado dinâmico do mundo (real ou simulado).

2.2.2 - ARQUITECTURA DE INFORMAÇÃO

Uma aproximação à integração de informação no sistema de programação tendo em atenção o modelo funcional é esboçada na fig. 2.2.1 [CamSte87a].

Um primeiro grupo de "blocos" de informação está relacionado com a fase preparatória da programação (planeamento de sistema), ou seja, inclui a especificação da tarefa (solução implícita) e indica também algumas relações com informação de suporte à produção dessa especificação. Num segundo nível têm-se os componentes de suporte e resultados da fase de programação genérica e simulação. Por fim são considerados os elementos de suporte à supervisão de execução e monitoração.

Breve descrição de componentes:

•Especificação do produto - descrição do produto a ser montado e suas peças componentes: geometria, material, restrições, etc..

•Modelo da célula: descrição de todos os componentes da célula (modelos a vários níveis: funcional, geométrico, cinemático), inter-relações entre eles e sua disposição espacial (*layout*).

•Especificação do processo, gama de fabrico ou grafo de montagem: grafo parcialmente ordenado representando uma especificação de alto nível da tarefa a realizar. Esta descrição é feita usando operadores de nível tarefa e não de nível estação. Os nós deste grafo incluem informação adicional como direcção de aproximação, tipo de ferramenta, restrições tecnológicas, etc..

Estes três componentes constituem efectivamente a **especificação da tarefa** para o sistema de programação, ou seja, representam uma **solução implícita** para o problema. Este é o ponto de partida para a programação genérica.

Fazendo a ligação com o modelo funcional, verifica-se que esta informação pode ter como "antecessores" elementos originários de bibliotecas de peças (CAD, onde, por vezes, o problema é mais de re-desenho do que de concepção integral [And87]) e componentes de célula, restrições tecnológicas e económicas (eventualmente BC especializadas) e informação introduzida interactivamente durante a fase de especificação.

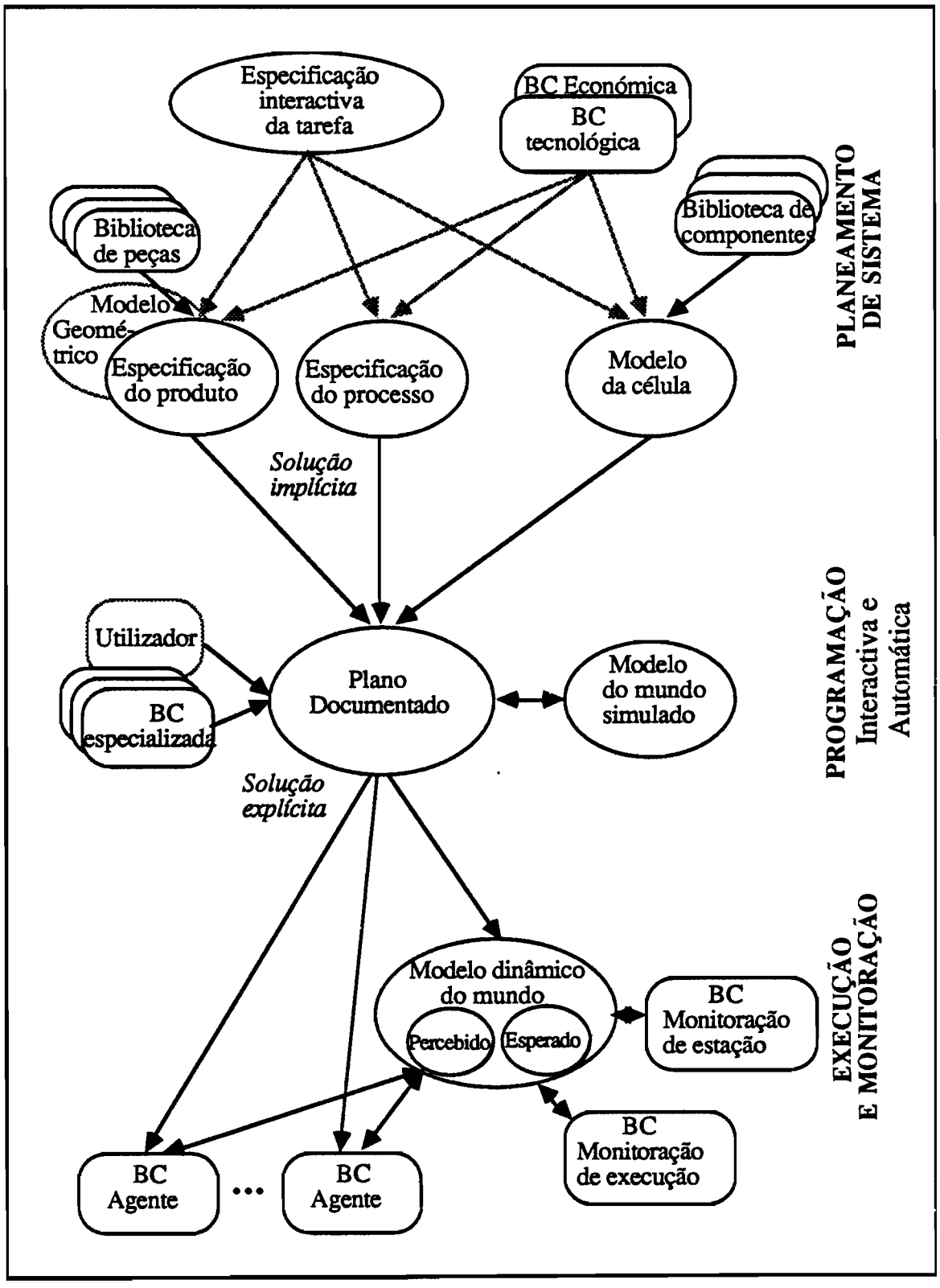


Fig. 2.2.1 Modelo geral do S.I.

•Plano / programa documentado: representa uma especialização da especificação da tarefa, agora a nível dos operadores da célula. É, também, um grafo parcialmente ordenado de operações. Para além das operações e respectivos parâmetros, conterá informação sobre o efeito pretendido (documentação do plano) para monitoração de execução.

•Modelo do mundo (dinâmico e simulado): representa um modelo dos aspectos dinâmicos (quer esperados quer percebidos sensorialmente) da execução do plano.

•BC especializadas - conhecimento de suporte aos especialistas que cooperam com o planeador na elaboração do plano genérico.

•BC monitoração de execução - bases de regras que dirigem os processos de diagnóstico e recuperação de erros de execução.

•BC monitoração de sistema: base de conhecimento de suporte aos níveis de monitoração de sistema (alimentada pela informação sensorial e conhecimento sintetizado de informação histórica sobre o comportamento do sistema).

•BCs dos agentes: conhecimento especializado que suporta o planeamento local e a percepção a nível dos agentes.

2.2.3 - PARADIGMAS E UTENSILIOS DE REPRESENTAÇÃO E PROGRAMAÇÃO

Requisitos de representação

Um aspecto fundamental para a concretização do SI é o da selecção de paradigmas e utensílios de representação de informação. De facto, a capacidade de modelação do mundo e, consequentemente, a "eficiência" do SI como elemento integrador não é independente dos "meios de expressão" usados. De certo modo as ferramentas (ou os paradigmas nelas suportados) "moldam" os processos de raciocínio* .

* *...'a linguagem já não é o reflexo das estruturas sociais, culturais ou psíquicas, torna-se a sua causa ... não serve para designar uma 'realidade' preexistente; é antes ela que nos organiza o mundo que nos rodeia'*
in Moderno Dicionário da Língua Portuguesa, Círculo de Leitores, 1988.

Para efectuar uma análise e a selecção de possíveis candidatos, importa fazer uma caracterização prévia dos requisitos de modelação no contexto da robótica.

A lista seguinte pretende apresentar, de forma resumida, tais características [CamCorBar87]:

a. Potencialidades para a representação de vários tipos de objectos

Mais importante que a representação de grandes quantidades de informação dum dado tipo - situação típica nas áreas convencionais de gestão - é a capacidade de representação de múltiplos tipos com poucas ocorrências de cada.

Em aplicações de engenharia e, em particular de robótica, é necessário poder definir objectos primitivos e complexos bem como diversas relações entre esses objectos. Como exemplo de objectos simples têm-se partes do manipulador, ferramentas terminais, sensores e peças a manipular. Objectos complexos são, por exemplo, robôs, transportadores ou células. Exemplos de relações entre objectos são conexão robô-transportador, robô-peça a montar, etc.. Uma estrutura hierárquica, relacionando os diferentes componentes dos objectos complexos emerge naturalmente mas outros tipos de relações não puramente hierárquicas devem ser contempladas.

Por outro lado, tais objectos terão diferentes classes de atributos - geométricos (modelos 3D), cinemáticos/ dinâmicos, tecnológicos, etc..

Poder-se-à dizer que, neste domínio, se necessitam estruturas mais heterogéneas ou diversificadas que nos sistemas informáticos tradicionais.

b. Aspectos estáticos e dinâmicos

A taxa de modificabilidade requerida não é a mesma para todos os componentes do sistema de informação. Por exemplo, uma biblioteca de componentes tem um carácter muito mais estático que o modelo dinâmico do mundo.

c. Interface com utilizador

Destinando-se este sistema a ser operado por especialistas de processos industriais mas não forçosamente especialistas informáticos, a interface com o humano, em particular na forma de apresentação da informação contida no sistema constitui um aspecto relevante. Será igualmente desejável que o sistema seja capaz de clarificar / esclarecer o utilizador sobre os conceitos que conhece e suas inter-relações.

Facilidades para visualização gráfica dos modelos de informação - estruturas arborescentes representando taxionomias, por exemplo - com possibilidade de inspeccionar (ver em detalhe) os nós da estrutura - constituirá uma boa ajuda, não só na fase de desenvolvimento como na de utilização normal.

A noção de imagem activa, que representa a possibilidade de associar imagens (termómetros, gráficos de barras, medidores, etc.) a objectos do SI de tal forma que a sua visualização gráfica seja automaticamente actualizada sempre que o valor objecto é modificado, constitui outro exemplo de potencial interesse.

É claro que estes aspectos não são requisitos específicos do SI, mas constituem peças essenciais quando este sistema é visto como o centro da integração.

Um outro aspecto relacionado com esta interface é o da existência de vários tipos de utilizadores: conceptores do produto, engenheiros de produção, controladores de processo, etc.. No próprio "interior" do sistema integrado de programação têm-se várias classes de subsistemas que serão "utilizadores" do SI.

Estes diferentes grupos têm requisitos diferentes e, em geral, apenas necessitam de (ou apenas deverão ter acesso a) visões parciais da informação. Deste modo, será interessante dispor de facilidades para definir diferentes submodelos como visões parciais do modelo geral.

d. Informação explícita e implícita

O sistema deverá ser capaz de responder a questões para as quais não tem informação representada mas sim regras que lha permitam inferir a partir doutros dados.

Por outro lado, nem toda a informação é "estável" - alguma depende de situações específicas (relacionada com aspectos estáticos e dinâmicos). O sistema deverá, então, produzir informação dependente do contexto - por exemplo, o centro de gravidade duma montagem articulada depende da configuração geométrica actual.

e. Representação de conhecimento impreciso

Em sistemas destinados a modelar o mundo real, este tópico assume especial importância já que, nestes casos, o conhecimento disponível é frequentemente incompleto e impreciso.

Todavia este é ainda um tópico em estudo e poucos são os utensílios que apresentam algumas facilidades para o efeito.

f. Comunicação / interface com outros sistemas

Esta questão está mais relacionada com ferramentas do que com paradigmas mas é de importância fundamental dado o papel integrador que o SI é pressuposto ter.

Deste modo, deverá ser possível interfaciar outros sistemas / linguagens (fontes ou consumidores de informação, tais como sistemas CAD, sistemas sensoriais, controladores de robôs, etc.).

Em geral, implementar estas interfaces envolve algum esforço de adaptação pelo que um factor de decisão na escolha do sistema deverá ser exactamente a facilidade com que essa conexão pode ser feita.

Outros factores tecnológicos adicionais surgem normalmente associados a este problema, especialmente quando se pretende uma conexão *on-line:*

-comunicação dentro do mesmo sistema computacional - por troca de mensagens através de *mailboxes*, por exemplo;

-comunicações num sistema fisicamente distribuído, usando as facilidades de rede eventualmente existentes.

g. Acesso multi-utilizador/ concorrente

Este aspecto é importante num contexto distribuído em que múltiplos subsistemas operam em paralelo e, portanto, originando acessos concorrentes ao SI.

h. Facilidades para aquisição de informação

O sistema será alimentado por diferentes tipos de utilizadores / especialistas, donde resulta a importância deste tópico. Complementarmente são importantes:

a-Validação automática e

b-Modificação dinâmica da estrutura - tratando-se dum domínio ainda não perfeitamente conhecido, o sistema deverá permitir, de forma rápida, alterações do modelo de dados.

i. Capacidade dos utensílios

Inclui as capacidades de armazenamento em memória central e em disco, velocidade de acesso, etc.

j. Disponibilidade e portabilidade

Em que equipamentos correm, custos e "idade" das ferramentas, etc..

"Panorâmica de mercado"

Em termos de utensílios constata-se que os diferentes paradigmas aparecem associados a diferentes tecnologias sem que se tenha atingido uma situação onde um único produto possa satisfazer os diversos parâmetros referidos acima. Isto pode ser comprovado por um levantamento de mercado reportado em [CamFer...87].

No que se refere a sistemas de representação, as principais áreas tecnológicas e respectivas características são:

•Bases de dados - área principalmente representada pelos sistemas relacionais, mas que tem sido tradicionalmente mais orientada para domínios de gestão, apresentando bastantes limitações quanto a aplicações de engenharia [Fer88], [KemWal86] [CamFerMou88].

Contudo algumas facetas importantes são normalmente contempladas, nomeadamente:

-acesso multi-utilizador (controle de concorrência, segurança e protecção dos dados);

-eficiência para tratar grandes volumes de informação (acessos optimizados, gestão de memória secundária);

- diversas "visões" da base.

Adicionalmente, trata-se duma tecnologia mais estabilizada, logo mais padronizada. Embora não exista um padrão de linguagem de acesso, o uso do SQL tem-se vindo a generalizar.

É, porém, um tanto limitada no que respeita a:

-possibilidades de modelação de objectos com estruturas complexas;

-formas de apresentação / interface com utilizador;

-representação de informação implícita.

•Sistemas CAD - especialmente vocacionados para modelação geométrica, incluem a sua própria base de dados especializada e um conjunto de funcionalidades adequadas a certos níveis de raciocínio geométrico.

Embora alguns produtos, como o PADL-2 [HarMar83] ou o ROMULUS [Sha85], apresentem a capacidade de modelar objectos 3D, grande parte dos sistemas CAD são bastante deficientes quanto à representação de sólidos.

Por outro lado, estes sistemas dificilmente se podem aumentar para contemplar outros tipos de informação (para além da geométrica) e não facilitam a expressão de modelos em diferentes níveis de abstracção (para além dos pré-definidos). A situação está, porém, evoluindo e alguns sistemas mais recentes começam a fornecer novas possibilidades de definição de atributos associados aos objectos ou mesmo a possibilidade de conexão a bases de dados externas.

•Bases de conhecimento - tem sido nesta área que mais desenvolvimentos têm sido feitos no campo da modelação / estruturação de informação. Existem disponíveis vários sistemas que combinam uma rica variedade de paradigmas - sistemas híbridos (KEE

[Int86a], Knowledge Craft [Car87], ART [Cla85], NEXPERT [Neu85], etc.). Tipicamente combinam:

- Representações por *frames* , onde é possível estabelecer estruturas hierárquicas ou, mais genericamente, redes. Uma forma especial de inferência - mecanismo de herança - permite que um *frame* herde os atributos do(s) seu(s) antecessor(es) segundo uma dada linha de ascendência muitas vezes representada pelas relações *is-a* ou *instance*. Nalguns casos a herança múltipla (*frame* com vários antecessores) é permitida.

-Programação reactiva (associando "demónios" aos *slots* que são disparados/activados quando determinados acessos a esses *slots* são feitos - valores activos).

-Programação orientada por objectos (um *frame* pode servir como forma de encapsular um conjunto de primitivas de acesso ao objecto, cujas funções / métodos são "guardadas" em *slots*).

-Programação baseada em regras (normalmente suportando inferências por encadeamentos *forward* e *backward*).

Tem-se, então, uma integração de aspectos declarativos e procedimentais.

A associação de demónios ou métodos aos *slots* representa uma forma de conhecimento procedimental permitindo modelar os aspectos dinâmicos dos objectos.

As características destes sistemas são, em certo grau, complementares dos SGBDs:

-Especialmente ricos no que se refere a poder de expressão (definição de novos tipos, com relações complexas entre si), interface com utilizador (visualização gráfica de estruturas - *browsers,* imagens activas, etc.), inferência de informação implícita;

-Apresentando limitações na forma de acesso (mono-utilizador), capacidades (toda a informação suportada em memória central), estabilidade (ainda não se atingiu uma situação estável em termos de nível de conceitos e funcionalidades em tais sistemas, embora os produtos mais significativos apresentem uma razoável proximidade entre si; contudo alguma evolução é previsível nesta área).

Soluções híbridas

Neste contexto, e também atendendo a situações de facto - existência à partida de muita informação industrial em sistemas CAD e BD relacionais - a aproximação aqui defendida é a de que, na fase actual e numa tentativa de equilíbrio entre requisitos conceptuais e a "realidade" existente, o SI deve ser suportado por uma configuração híbrida, integrando as 3 tecnologias referidas. (Em capítulos seguintes far-se-á uma

apresentação concreta.) O sistema de representação de conhecimento actuará, contudo, como elo central.

Relativamente à forma de integração pode pensar-se num acoplamento "forte" ou "fraco" (*tight* ou *loose coupling*) entre os sistemas, consoante a intensidade e frequência pretendidas para as comunicações (comunicações entre módulos software e não a nível do suporte físico).

Uma conexão forte deve permitir que as interacções ocorram a qualquer momento e que cada sistema desempenhe o papel em que é mais especializado. Uma comunicação fortemente interactiva entre os sistemas garante grande disponibilidade das funções de cada um permitindo, deste modo, uma efectiva distribuição funcional de acordo com as "especializações" de cada um.

Numa aproximação de acoplamento fraco, as interacções entre sistemas serão mais esporádicas.

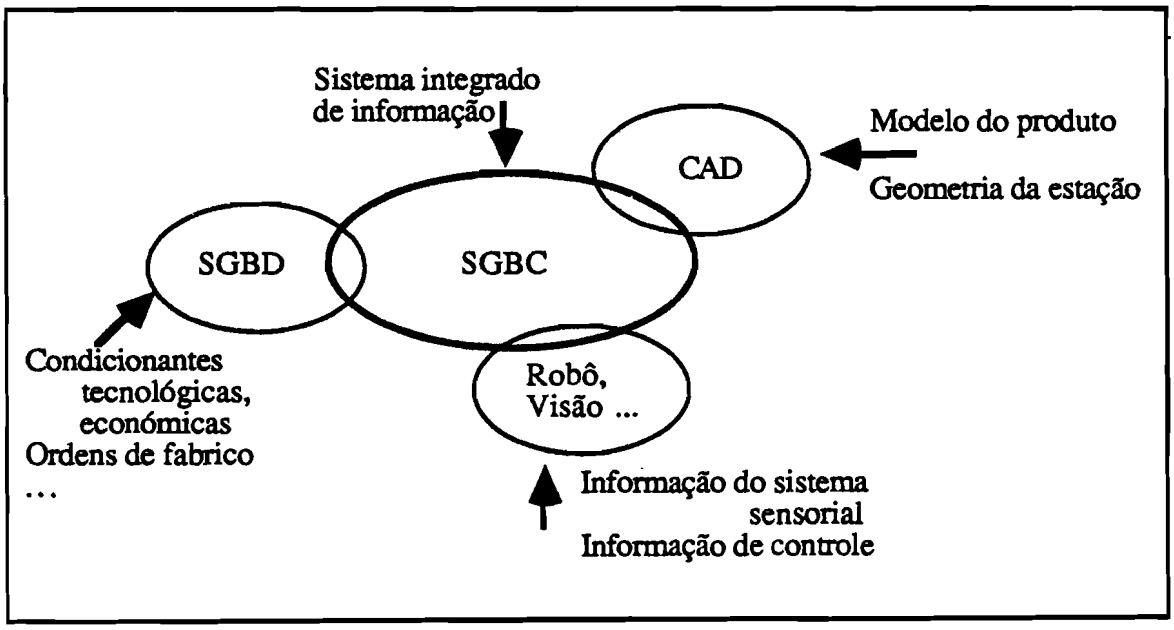


Fig. 2.2.2 Integração de tecnologias de suporte ao SI

A questão da forma de acoplamento coloca-se não só em relação aos sistemas de suporte ao SI mas também em relação à integração no SI dos diferentes módulos funcionais do sistema de programação e controle.

A aproximação seguida foi preferentemente a duma conexão forte, embora nalguns casos também se considerassem acoplamentos mais fracos (caso da ligação CAD - Percepção, na sua fase actual).

Uma conexão forte das várias funções do sistema de programação implica - para facilidade de concretização - que o seu desenvolvimento tenha lugar sobre o mesmo ambiente do SGBC.

2.2.4 - EVOLUÇÃO COMPARADA

Como já foi referido, os primeiros trabalhos em diferentes subáreas da robótica tiveram desenvolvimentos separados. Desta forma, antes de se tentar uma aproximação integrada para o sistema de programação também não se tentou qualquer integração de informação.

Durante os primeiros anos da década de 80, as questões da modelação do mundo foram colocadas em contextos isolados, de acordo com as necessidades de cada subsistema. Como exemplos de tais desenvolvimentos, com contributos importantes para as aproximações de integração posteriores, podem referir-se:

-Modelação de sólidos

Uma das áreas mais pesquisadas, normalmente ligada às necessidades dos sistemas CAD, deu origem a diversas técnicas de modelação 3D: CSG (*Constructive Solid Geometry*), B-Rep (Representação por fronteiras), ... [Req80].

Como exemplo paradigmático deste período, podem-se referir os trabalhos da equipa de Automatização da Produção da Universidade de Rochester, de que resultou o sistema PADL-2 [HarMar83]. Os resultados, em termos de modelação de sólidos, constituiram um marco importante, embora a contribuição para a ligação CAD-CAM (passagem dos modelos de produto para o controle das máquinas NC), que era um dos objectivos, fosse mais modesta.

-Modelação de robôs

Uma área de certo modo complementar (por envolver raciocínio geométrico) é a da modelação do manipulador: modelos geométrico, cinemático e dinâmico. Estes

desenvolvimentos surgiram ligados à construção dos controladores para os equipamentos reais e ao desenvolvimento de emuladores / simuladores gráficos.

[Pau81] faz uma boa apresentação destes aspectos e dos resultados hoje estabilizados.

Os aspectos de controle do robô a baixo nível continuam, porém, a ser objecto duma exagerada atenção (comparativamente com outros domínios) em termos dos investimentos de investigação.

-Modelo do mundo numa perspectiva de IA

A área de geração de planos (como outras áreas da IA) tiveram importantes contributos para a modelação do mundo (estado dos agentes, modelo dos operadores, etc.). Todavia, no caso da geração de planos, o domínio considerado foi frequentemente o mundo dos blocos, portanto bastante limitado [Cam87a].

Uma caracterização determinante do papel que poderia ser desempenhado pelo SI ou arquitectura de conhecimento no sistema de programação foi feita em [CamSte86]. Aî se defende que a estrutura, eficiência e flexibilidade do sistema dependem do conhecimento que este tem. Neste trabalho são identificados alguns dos componentes básicos:

-Modelo da estação (agentes e respectivos operadores);

-Especificação da tarefa e produto;

-Modelo dinâmico do mundo;

-Representação do plano;

e apresentada uma lista de algumas das questões (elementos de informação) por resolver, com especial ênfase nos aspectos de modelação geométrica.

São também referidos a natureza multifacetada do sistema, os diferentes níveis de abstracção, as diferentes visões para os diversos subsistemas e a necessidade duma clarificação dos fluxos de informação.

Ideias iniciais no sentido da integração de ferramentas para suporte do sistema de informação são apresentadas, com destaque para a integração de um modelador de sólidos (PADL-2) e Prolog.

Um maior nível de detalhe nesta caracterização da arquitectura de conhecimento, com especial ênfase no que se refere aos agentes sensoriais, foi conseguido em [SteCam86b]. Aí se introduz a "necessidade" de bases de conhecimento locais aos agentes, se detalha a composição dessas bases - modelos dos sensores, árvores de identificação (para reconhecimento de objectos), etc. - e discute o carácter impreciso da informação a manipular a este nível.

O relacionamento entre estas bases locais e o sistema global é também estabelecido:

-Questões de actualização do modelo dinâmico do mundo;

-O próprio "carregamento" inicial da base de conhecimento local através dum eventual processo de aprendizagem por indução, a partir dos modelos geométricos das peças a reconhecer e do modelo da estação (que sensores existem e que características extraem, com que custos e com que confiança).

A utilização de sistemas de desenvolvimento de bases de conhecimento para a realização de protótipos rápidos é também defendida.

Em [KemWal86] é também feita a defesa da utilização duma BD como elemento centralizador ou integrador em sistemas CIM de um modo geral e, em particular, nas aplicações robóticas. Constatando as limitações dos actuais sistemas de gestão de BD, apresenta uma súmula crítica dos esforços que têm sido desenvolvidos no sentido de alargar as potencialidades de tais sistemas, terminando com uma proposta em desenvolvimento (R^2D^2) numa cooperação entre a Universidade de Karlsruhe e a IBM. Tal proposta apresenta um sistema de gestão de bases de dados baseado no conceito de tipo de dados abstractos (pelo menos a nível de interface com utilizador) onde, a par de bibliotecas de tipos pré-definidos e específicos para certos domínios de aplicação, seja permitido ao utilizador desenvolver outros tipos (incluindo as respectivas operações de acesso) mais específicos de acordo com as necessidades. A proposta parece, contudo, ignorar completamente muitos dos desenvolvimentos contemporâneos, ou mesmo anteriores, emergentes das áreas de representação de conhecimento.

No que respeita ao Grupo de Robótica Inteligente (GRI) da UNL, alguns desenvolvimentos parciais, tendo por orientação geral a noção de integração de informação proveniente de múltiplas fontes, foram iniciados em 1986. Como ponto de partida, procedeu-se à implementação dum pequeno "frame engine" experimental sobre o Prolog [Cam87b], destinado a avaliar da adequabilidade dos paradigmas de representação por "frames", programação reactiva, orientada por objectos e baseada em regras, às necessidades da robótica.

Tendo este sistema por centro de integração, interfaces com modeladores de sólidos (Padl-2) [CamSteBap87][CamSte87b] bem como com os próprios controladores dos elementos activos da estação (robô controlado por LM [CamBar87], [CamCor88], transportador), foram implementadas com sucesso e permitiram comprovar a adequabilidade da aproximação.

Em 1987, a noção de SI como elemento integrador em sistemas robotizados é aceite no consórcio do projecto Esprit 623 [SteCam87c] bem assim como a proposta de arquitectura híbrida. A concretização duma tal aproximação passou a ser a principal actividade do Grupo de Robótica Inteligente da UNL no referido projecto.

Todavia as naturais dificuldades de coordenação de actividades numa equipa multinacional conduziu a que, nesse projecto, a aproximação fosse mais a dum acoplamento fraco do que uma real integração. De facto, cada subsistema, desenvolvido por equipas diferentes, utiliza os seus próprios modelos e fontes de informação, o que conduz a forte redundância e apenas a pequenas trocas de informação entre subsistemas. A nossa contribuição permitiu, porém, apontar direcções para uma maior integração futura [SteCam...88c].

Um primeiro modelo de arquitectura de informação como suporte a um sistema integrado de programação de sistemas robóticos é defendida em [CamSte87a]. Nesta fase é feita a caracterização do ponto de partida para o sistema de programação e, portanto, definidas as trocas de informação entre este e as fases de concepção e planeamento de processo (modelo de produto, processo de montagem e modelo da estação). Uma caracterização dos vários componentes do SI é feita e apresentada a aproximação de desenvolvimento com base num *frame engine*.

Um conjunto de resultados experimentais, embora ainda com pequeno grau de integração, é também apresentado em apoio da proposta incluindo uma 1ª versão da integração CAD-FrameEngine, modelo dinâmico do mundo, modelo dos agentes (aspectos declarativos e operativos), etc..

Uma proposta contemporânea, em vários aspectos coincidente com a aproximação aqui preconizada, é apresentada em [SchDim87]. Aí também se faz a defesa da utilização dum sistema de gestão de Bases de Conhecimento como centro de integração de informação em CIM e se caracterizam algumas das necessidades da integração CAD-CAM.

Como suporte experimental é proposto o sistema KANON, desenvolvido na Universidade Técnica de Berlin. Este é basicamente um sistema híbrido, implementado em Prolog, que apresenta diferentes estruturas de representação: regras, redes semânticas e *frames*, com facetas de programação reactiva e orientada por objectos. Uma interface com bases de dados relacionais, via SQL, permite que o suporte de memória do sistema seja assegurado por essas BDs externas.

Parece, contudo, que o maior esforço foi colocado na construção deste utensílio tendo ainda sido pouco desenvolvidas as questões da sua real utilização em CIM.

Por outro lado, é de notar que neste período se verificou uma grande evolução a nível dos sistemas genéricos de gestão de bases de conhecimento disponíveis no mercado. Especial destaque para os sistemas KEE, Knowledge Craft, ART , surgidos por volta de 1985 e que atingiram em 87 (versão 3) uma certa uniformização de conceitos.

A aproximação entre as tecnologias de bases de dados e de bases de conhecimento representa, há vários anos, uma área onde se tem registado intensa actividade, quer nos aspectos conceptuais, quer do ponto de vista de materializações experimentais. Essa aproximação pode verificar-se em resultado de esforços de extensão de cada uma das áreas ou dum desejo deliberado de integração [BroMyl86][DitDay86].

Como resultado dos esforços conceptuais para a integração BD-BC, começaram a surgir sistemas concretos.

Um exemplo de integração é constituído pelo sistema KEEconnection que tem por objectivo combinar a performance das BDs com o poder de expressão das BCs, através duma conexão entre SGBDR/SQL e o sistema KEE [Int87].

Um protótipo permitindo a integração dos sistemas Knowledge Craft e Rdb, atendendo aos requisitos específicos da robótica, foi também desenvolvido na UNL [SteCam87c][CamFerMou88][CamFer...88].

Um dos aspectos inovadores deste sistema relativamente ao KEEconnection foi a introdução da noção de **carregamento virtual**. A aproximação usual consiste em carregar a memória de trabalho da BC com cópias (que podem ser parciais) da informação contida na BD e então raciocinar sobre essas cópias. Na nossa aproximação, a par desta opção (designada por carregamento real), existe a possibilidade de criar, na BC, imagens virtuais dos tuplos da BD, sem cópia real da informação. Desta forma, os dados residem efectivamente na BD. Quando, do lado da BC, se acede a um item virtual, através dum mecanismo de programação reactiva (demónio associado a esse item) é feito um acesso à BD, permanecendo, no entanto, todo o processo transparente para o utilizador.

A solução de carregamento virtual tem o inconveniente de implicar menor eficiência no acesso aos dados, embora permita recuperar uma faceta importante dos sistemas de gestão de BD: o controle do acesso concorrente (que não é contemplado nos actuais sistemas de BC), uma vez que os acessos são sempre feitos à BD que, assim, manterá imagens actualizadas para todos os eventuais utilizadores.

Tem-se, neste caso, uma interacção forte entre os dois sistemas enquanto que no carregamento real existirá uma conexão fraca.

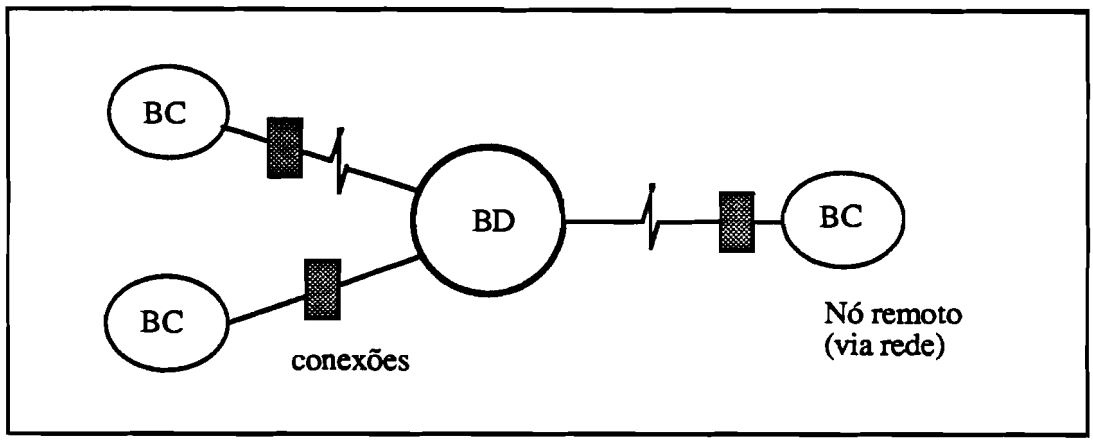


Fig.2.2.3 Multiacesso de SGBC a um SGBD

Outro aspecto contemplado nesta implementação tem a ver com o facto de a BC servir, na aproximação aqui defendida, como elo de integração de informação provinda de múltiplas fontes. Desta forma, os conceitos criados na BC têm uma natureza "aditiva" relativamente à origem dos respectivos atributos. Por exemplo, um conceito não é criado exclusivamente a partir da informação presente na BD mas pode ser complementado com informação CAD.

A evolução para o modelo apresentado na secção 2.2.2 (Arquitectura de informação) é descrita em [CamSte88] e também parcialmente em [SteCam88a]. Nestes trabalhos, a par duma clara localização do problema num contexto CIM, apresentam-se vários refinamentos em relação às fases anteriores da proposta, nomeadamente pelo desenvolvimento da noção de monitoração.

A vertente "demonstração experimental" é também mais desenvolvida pela apresentação detalhada dum sistema integrado de ensaio contemplando exemplos dos principais componentes do SI. Neste sistema integram-se modelos geométricos (CAD), controladores de componentes da célula, módulo de percepção baseado em visão, simulação, processador de referenciais e informação proveniente de BD relacionais.

Detalhes da geração de planos e monitoração de execução são também apresentados integrados neste contexto de SI.

Informação complementar pode ser encontrada em vários relatórios de implementação: [MouCam88] - integração de informação geométrica, [CamCor88] - integração de controladores da célula, [PitCam88] - gestão de referenciais, [CamFer...88] - integração de BD e BC, [Cam89b] - sistema integrado de informação.

3. DETALHE DA PROPOSTA

3.1 - PROGRAMAÇÃO GENÉRICA

3.2 - SISTEMA EXECUTIVO E DE MONITORAÇÃO

Após a apresentação do modelo genérico feita no capítulo anterior, segue-se uma análise mais detalhada dos diversos aspectos e discussão de resultados experimentais.

Numa primeira parte, dedicada à programação genérica, discutem-se os aspectos relativos à construção dum ambiente integrado de desenvolvimento, estabelecimento de modelos (componentes do SI) sobre esse ambiente e a problemática da geração interactiva do plano.

Na segunda parte são discutidos os aspectos relativos ao subsistema de supervisão de execução e monitoração. As estruturas de informação necessárias, a interacção do ambiente de desenvolvimento com os controladores dos componentes físicos da estação são também tratadas.

A divisão nestes dois sub-capítulos não é isenta de problemas pois existem várias interacções e referências cruzadas no texto que dificultam uma leitura sequencial. Por exemplo, alguns aspectos considerados durante a geração do plano genérico (documentação do plano, detalhe de acções e informação não disponível em tempo de planeamento) só serão esclarecidos na parte de execução e monitoração. Estas difculdades são consequência da extensão e complexidade intrínseca da temática abordada, parecendo difícil encontrar uma organização mais adequada para o texto. Espera-se, porém, que a visão geral dada no cap. 2 atenue os mencionados problemas de legibilidade.

Como foi referido no primeiro capítulo, não se faz a apresentação duma implementação completa mas apenas dos aspectos que se julgaram determinantes na arquitectura proposta. Todavia, em paralelo, são levantados vários dos problemas não resolvidos e discutido o seu papel no sistema global, particularmente em termos de interacção com os modelos de informação. Estão neste caso, por exemplo, os aspectos de planeamento especializado, simulação no suporte ao planeamento interactivo e monitoração de sistema.

3.1 - PROGRAMAÇÃO GENÉRICA

3.1.1 - INTRODUÇÃO

Como foi indicado nos capítulos anteriores, na abordagem ao problema da programação da estação assumiu-se a existência duma fase preparatória (planeamento de sistema) e a conveniência de recuperar a informação disponível ou gerada a esse nível. Isto permite colocar o problema no contexto das tendências / realidades dos sistemas de produção industrial e não numa perspectiva demasiado generalista.

Assim, neste capítulo, após uma caracterização da informação de partida e dos objectivos pretendidos - especificação abstracta da tarefa - aborda-se essencialmente o problema da geração do plano ou programa genérico, isto é, produção duma descrição da tarefa a nível dos operadores executáveis pelos agentes da estação.

É seguida uma aproximação interactiva onde algum grau de automatização (geração automática de planos) é complementado por ajuda externa do especialista humano tentando-se estabelecer, de forma realista, quais as tarefas que podem ser facilmente realizadas pelo humano e aquelas onde é mais prioritária a automatização. A abordagem é, contudo, feita sem perder de vista um progressivo incremento da automatização.

Uma apresentação da estruturação de informação e da interacção da actividade de geração do plano com as várias componentes do SI serão feitas em paralelo.

Como suporte ao desenvolvimento e avaliação dos modelos propostos foi fundamental um trabalho de estabelecimento dum ambiente integrado de experimentação, que se apresenta numa primeira secção.

Teste-padrão de Cranfield

A fim de tornar mais clara a metodologia que se propõe serão apresentados exemplos tendo como base o teste-padrão de Cranfield (*Cranfield benchmark*). Este é baseado na produção dum pêndulo por montagem do respectivo conjunto de peças componentes (ver fig. 3.1.1).

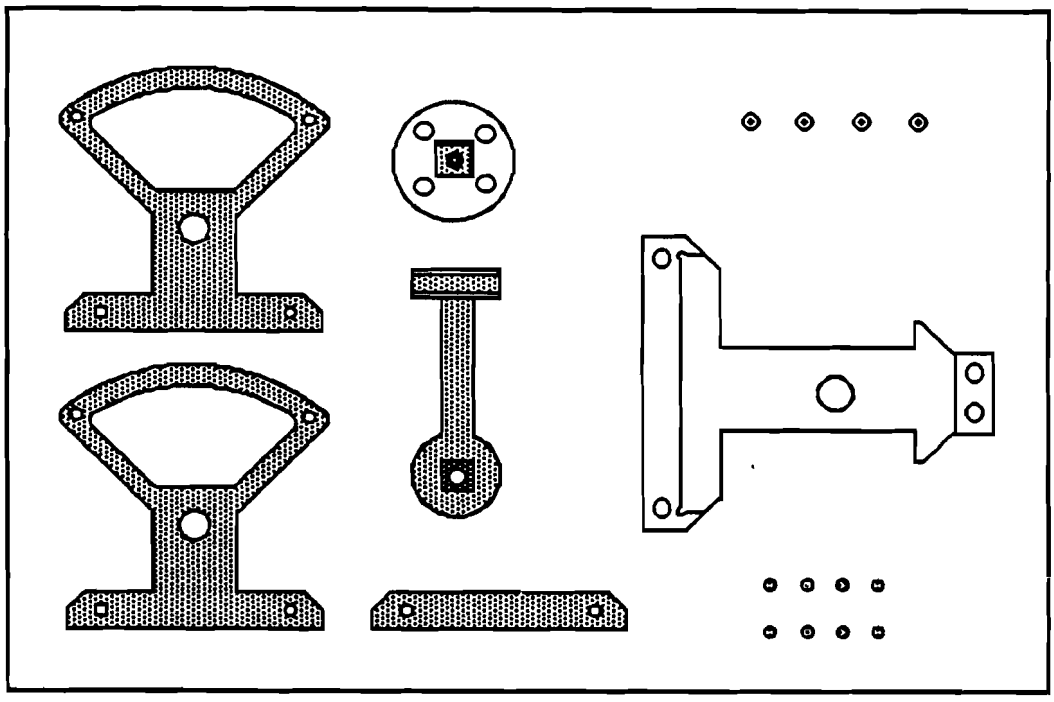


Fig. 3.1.1 Teste-padrão de Cranfield

O teste foi concebido pelo Instituto de Tecnologia de Cranfield [ColPalRat84] no âmbito dum projecto Esprit e destina-se a avaliar a adequabilidade / desempenho de sistemas robóticos em tarefas de montagem e tem vindo a ser usado em vários centros de investigação europeus.

Na sua versão original, as peças vêm fixadas numa paleta que também inclui um fixador (fixture) para suportar a montagem. O processo de montagem segue essencialmente uma filosofia de empilhamento, excepto no que se refere aos 8 pinos que devem ser introduzidos lateralmente. Nesta versão pretende-se essencialmente testar a precisão do robô, a adequação de movimentos / volumes de trabalho e complacência para a introdução de pinos (tolerância apertada).

Não se recorrendo à fixação de peças à palete, o teste também pode ser usado para testar as capacidades de percepção do sistema (visão 2D, por exemplo).

3.1.2 - SISTEMA INTEGRADO DE DESENVOLVIMENTO

Ambiente de desenvolvimento UNL

Um ambiente CIM apresenta tipicamente características de um sistema distribuído e envolve a utilização de ferramentas (equipamentos e software) variadas, colocando-se de forma acentuada, como foi referido, o problema da integração de informação. O modelo conceptual de um sistema de programação e controle para uma estação robótica inserida neste contexto deve atender a tais características. Neste sentido, como plataforma de suporte ao desenvolvimento e avaliação dos distintos aspectos do sistema de programação e controle, a estratégia adoptada consistiu na realização dum ambiente de desenvolvimento integrando múltiplos e diferenciados utensílios. Numa perspectiva de informação, um sistema integrado de programação pode ser visto como sendo constituído por múltiplas fontes e múltiplos consumidores. O actual sistema de desenvolvimento foi concebido de forma a integrar utensílios típicos de suporte a essas fontes / consumidores. Nesta integração utilizam-se soluções de conexão forte (sistemas embebidos) e fraca (*tight / loose coupling*).

Como elemento central à integração foi seleccionado - pelo poder de estruturação e por permitir aliar aspectos declarativos e procedimentais - um sistema híbrido de representação de conhecimento por *frames*, incorporando os paradigmas de programação reactiva, orientada por objectos e baseada em regras.

Numa fase exploratória, e para avaliar a adequabilidade destes paradigmas de representação / programação no contexto da Robótica, foi desenvolvido um "*frame engine*" (FE) experimental sobre Prolog materializando tais aspectos [Cam87b]. Como características básicas do FE podem referir-se:

-Sistema dinâmico - *frames* e *slots* podem ser criados / modificados dinamicamente;

-Herança (hierárquica) de atributos (usando a relação pré-definida *é-um*);

-Programação reactiva, pela associação de predicados (demónios) aos *slots*. Um conjunto limitado de tipos de demónios foi considerado: *if-read, if-needed, if-added, if-replaced;*

-Programação orientada por objectos, onde um objecto é representado por um frame, métodos são suportados por slots e o envio duma mensagem para um objecto é realizado pelo predicado: frame-call-method (Frame, Slot, Parameters);

-Utilitários para visualização da estrutura de *frames* sem necessidade de interpretação da representação interna (fig. 3.1.2); armazenamento e carregamento de ficheiros em formato textual.

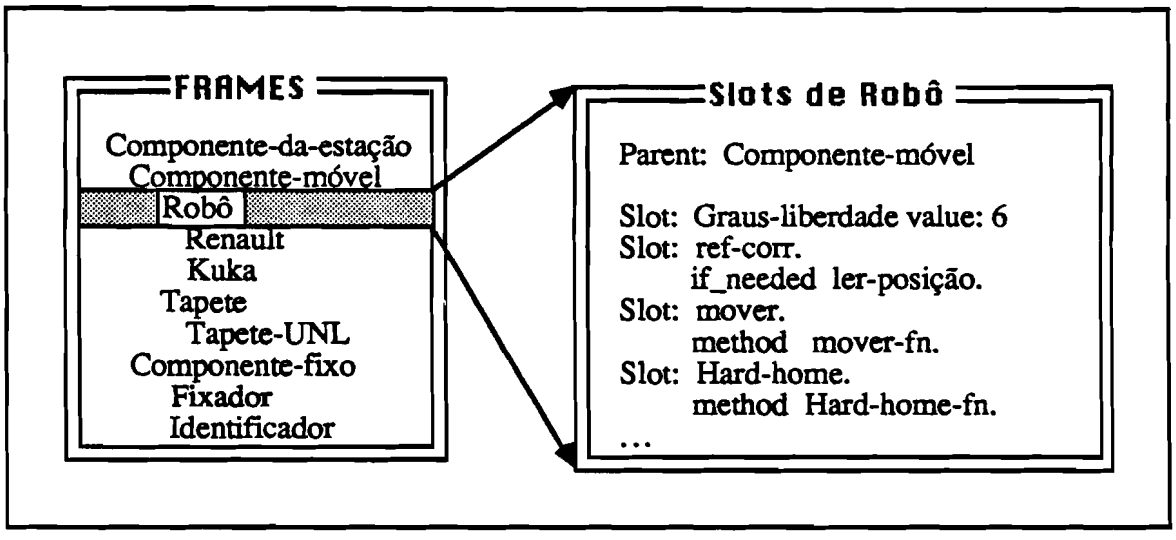


Fig. 3.1.2 Visualização da estrutura de *frames* no *frame engine* / Prolog

Uma primeira versão de sistema de desenvolvimento foi realizada sobre este "*frame engine*" (fig.3.1.3).

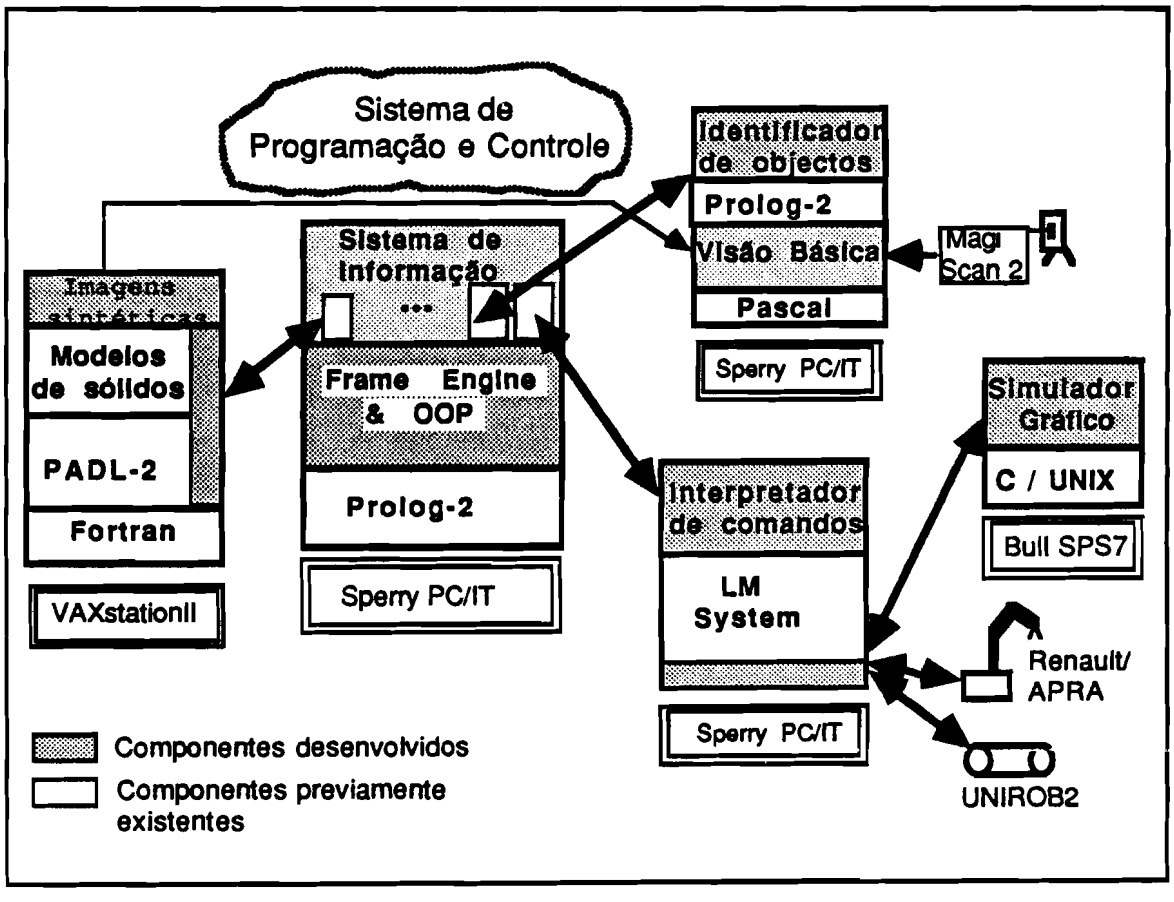


Fig. 3.1.3 Sistema integrado de desenvolvimento - primeira versão.

Numa segunda fase, e em resultado da primeira avaliação, adoptou-se um ambiente mais complexo de desenvolvimento de sistemas baseados em conhecimento e com funcionalidades adicionais denominado Knowledge Craft [Car87]. Algumas facilidades adicionais deste sistema, comparativamente com o anterior, incluem:

-Para além da relação *is-a*, que permite especificar relações entre *frames* e estabelecer hierarquias de conceitos, onde os níveis mais abaixo correspondem a especializações dos níveis acima, suporta a relação *instance* que permite estabelecer as folhas dessa hierarquia. Por outras palavras, enquanto a relação *is-a* é estabelecida entre conceitos ou protótipos a relação *instance* é estabelecida entre um "objecto" concreto e o seu protótipo.

Adicionalmente, o utilizador pode definir novas relações e especificar o respectivo mecanismo de herança: que atributos são herdados através da relação, eventual transformação da informação durante a herança, alcance / visibilidade através duma hierarquia de *frames*, etc..

-Integra estratégias de inferência usando encadeamento de regras "*backward*" (CRL-PROLOG) e "*forward*" (CRL-OPS).

-Em termos de programação reactiva, não limita os demónios a um conjunto pré-estabelecido de tipos, mas fornece mecanismos para especificar o comportamento do demónio: que acções os despoletam, quando actuam (antes ou depois da acção despoletadora) e que efeitos têm.

-Heranças múltiplas, isto é, através das relações poderemos ter, não apenas árvores, mas grafos dirigidos acíclicos.

-Algumas facilidades para a representação e processamento de meta-conhecimento.

-Mecanismo de contextos permitindo definir estruturas hierárquicas de cenários ou caminhos exploratórios alternativos.

-Módulo gráfico, fornecendo um conjunto de facilidades para definição de interfaces com utilizador, com gestão de filas de eventos assíncronos.

-Gestão de filas de espera.

-etc..

O sistema foi desenvolvido em Common Lisp onde se encontra embebido e pode comunicar com outros sistemas através dos mecanismos da linguagem Lisp para I/O e acesso a procedimentos externos.

Em relação a esta segunda fase, o sistema de desenvolvimento foi aumentado com uma integração dum sistema de gestão de bases de dados relacionais e um processador de referenciais.

O desenvolvimento do "*frame engine*" em Prolog permitiu que algum trabalho experimental fosse realizado mesmo antes de se dispor dum utensílio mais versátil (como o Knowledge Craft).

Como efeito "lateral", forneceu um instrumento que, embora pouco sofisticado, pode ser suficiente para certos problemas especializados e susceptível de ser suportado por pequeno poder computacional (que não é o caso de instrumentos mais "pesados"). De facto as características do Knowledge Craft facilitam o desenvolvimento de protótipos mas a materialização de sistemas mais eficientes pode justificar reimplementações parciais específicas.

Outro efeito lateral importante foi o de permitir acelerar o processo de formação da Equipa de Robótica da UNL nestas metodologias.

Vejamos agora em detalhe cada um dos componentes do sistema:

a) Integração CAD - FE

Um sistema CAD é importante para a modelação geométrica quer dos produtos, quer dos outros elementos da estação, pelo que deve ser um dos componentes de suporte ao Sistema de Informação. Do ponto de vista das necessidades do sistema de programação, um sistema CAD fornece essencialmente um editor de modelos, permitindo a sua visualização gráfica, documentação e algumas facilidadades de raciocínio geométrico - características úteis para a simulação e programação interactiva. Nalguns casos incluem facilidades para geração de imagens sintéticas, o que é importante para a simulação da visão.

Por outro lado os sistemas existentes comercialmente são tradicionalmente pouco flexíveis quanto à extensão das facilidades de modelação, isto é, dificilmente permitem aumentar os modelos com informação não geométrica. O próprio acesso às estruturas internas de representação não é normalmente muito linear. Esta situação começa, porém, a mudar com os sistemas mais recentes.

A aproximação seguida é a de que uma conexão interactiva dum sistema CAD com um *frame engine* constitui um auxiliar para a implementação do sistema de informação. Os aspectos de edição de modelos, visualização e algumas facetas de raciocínio geométrico ficarão centrados no sistema CAD, enquanto as restantes facetas dos modelos e raciocínio mais simbólico serão suportados pelo FE.

Uma primeira implementação segundo esta filosofia foi baseada no sistema PADL-2 [HarMar83] e no *frame engine* experimental, conforme se ilustra na fig.3.1.4 [CamSteBat87] [CamSte87b] [MouCam87]. Uma segunda implementação foi baseada no Knowledge Craft.

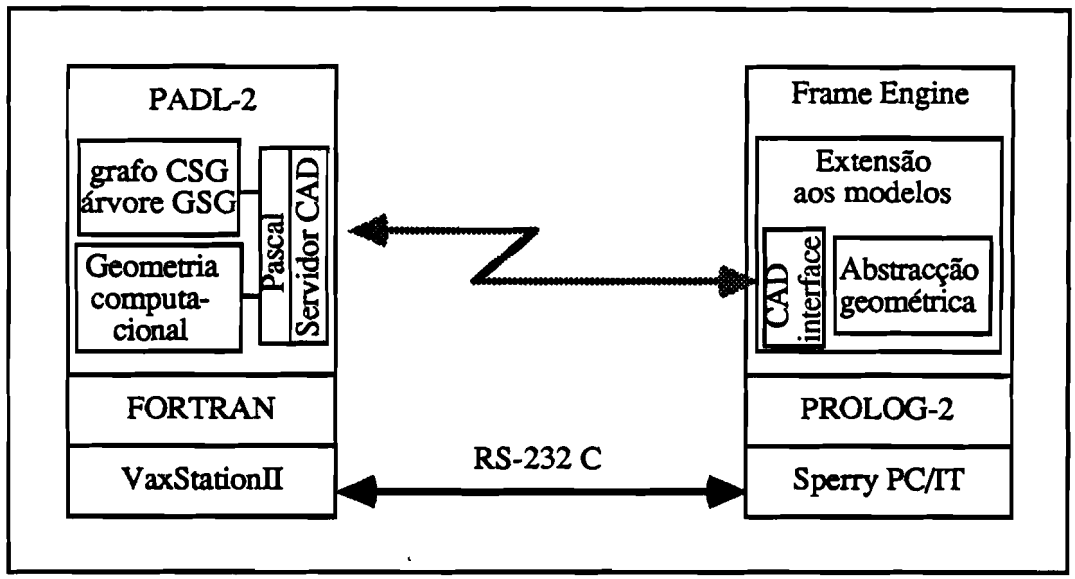


Fig. 3.1.4 Conexão Padl-2 - Frame Engine / Prolog

Abstracções dos modelos geométricos podem ser extraídas do CAD e representadas numa estrutura de frames, que depois será complementada com informação adicional proveniente de outras fontes.

Do lado do CAD houve que desenvolver o componente "Servidor CAD" que acede à informação da base de dados local e a torna disponível, e na forma necessária, quando solicitada pelo *frame engine*.

O diálogo entre os dois sistemas é suportado por um protocolo apoiado numa filosofia mestre / escravo em que o *frame engine* desempenha o papel de mestre.

Do lado do *frame engine*, e tendo já em vista tarefas de montagem, implementou-se um conjunto de conceitos de suporte (CAD Interface), organizados sob a forma duma hierarquia (fig. 3.1.5), onde se incluem:

- **Família**, representando objectos indivisíveis representando protótipos ou peças tipo;
- **Peça**, representando instâncias concretas de peças tipo (correspondendo aos objectos reais);

•**Montagem**, formada por peças e outras (sub)montagens, posicionadas e orientadas no espaço.

O volume definido por cada objecto pode ser representado por diferentes formas [Req80]:

-CSG (*Constructive Solid Geometry*)

-B-rep (*Boundary representation*)

-Envelope (*enclosing box*)

-etc..

Para além destes conceitos, consideram-se ainda as entidades real, texto e referencial, para suportar características dos objectos.

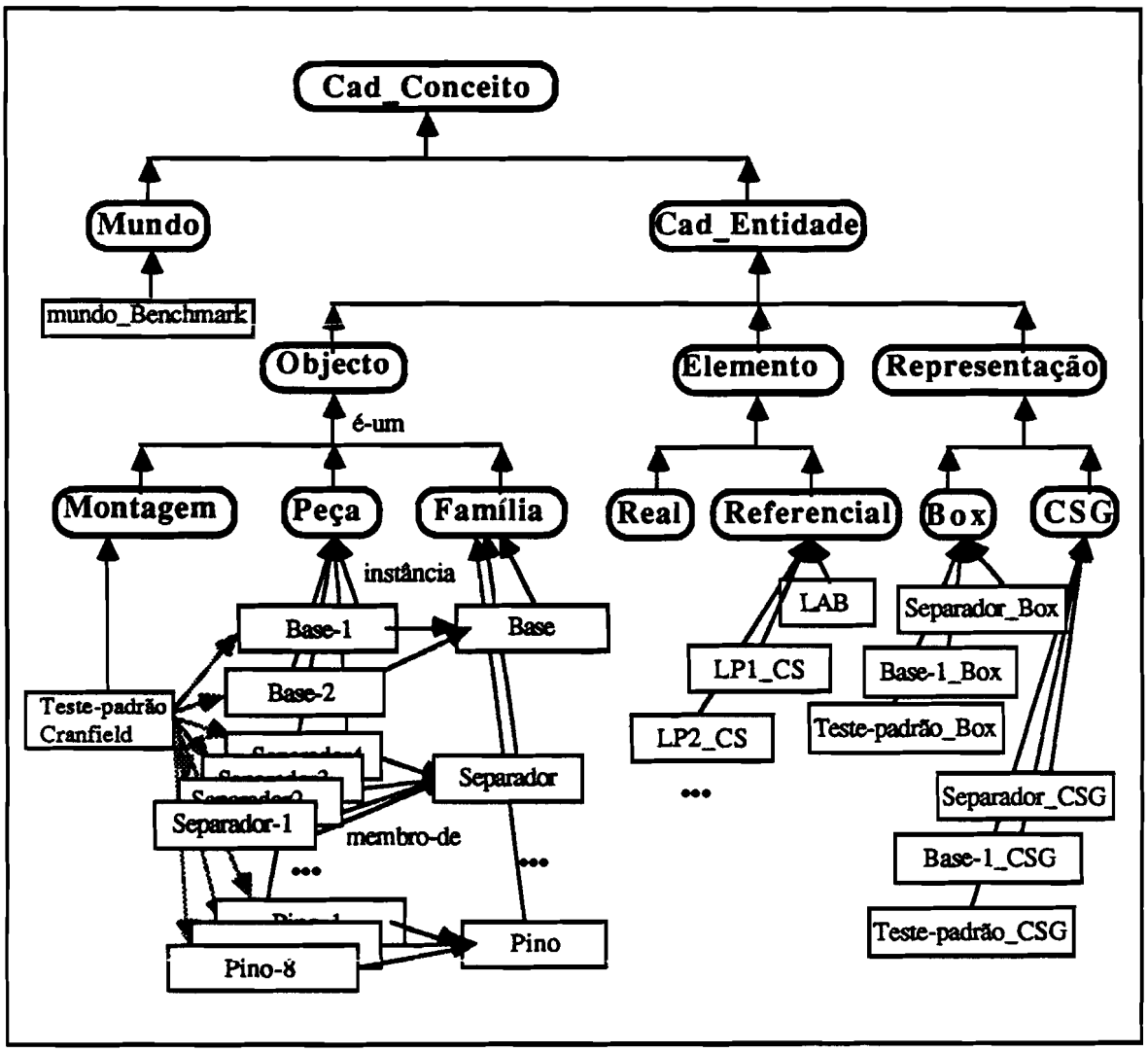


Fig. 3.1.5 Representação em *frames* dum modelo extraído do CAD

Num sistema CAD pode ser suportado o conceito de **mundo** ou contexto representando os objectos envolvidos num dado produto. Uma conexão é, então, estabelecida, de cada vez, para um dado mundo. Quando um mundo é activado, os dados extraídos do CAD vão ser representados como instâncias da referida hierarquia de conceitos, conforme se exemplifica na fig. 3.1.5 para o caso do teste-padrão de Cranfield.

É de notar que nem todas as relações entre instâncias estão representadas na figura. Por exemplo, relações entre peças e referenciais ou representações não estão indicadas.

Uma peça herdará os atributos da respectiva família através da relação "Membro-de". Por exemplo:

```
{{BASE:                          {{BASE-1:
        instância: Família               instância: PEÇA
        ...                              Membro-de: BASE
        Material: Al                     Ref-loc: LP1-CS
        Tolerância: 0.05                 Represent: BASE-1-CSG
        ...                              ...
}}                               }}
```

BASE-1, sendo uma peça da família BASE, herdará propriedades como "Material", "Tolerância", etc..

Como se referiu, a conexão estabelecida é interactiva. Desta forma, alguns atributos podem ser carregados durante a geração inicial da rede de *frames* (carregamento da abstracção geométrica) ou, então, a pedido quando forem necessários (em tempo de execução). Conforme se ilustra na fig. 3.1.6 a programação reactiva suporta adequadamente esta última possibilidade: quando se tenta obter o valor dum atributo não presente num *frame* ou se pretende uma representação dum objecto, um demónio é activado e tenta obter, do sistema CAD, esse valor ou representação. Por outras palavras, a associação de demónios a *slots* permite representar, duma forma transparente, a funcionalidade do *CAD Server*.

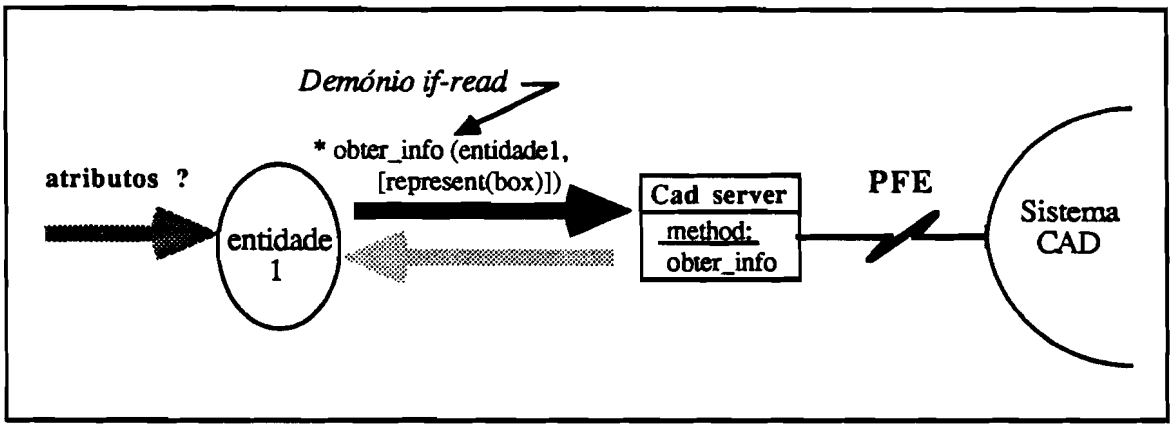


Fig. 3.1.6 Utilização de programação reactiva na interrogação do sistema CAD

Este mecanismo permite, de facto, uma distribuição de funcionalidades pelos componentes sendo, contudo, oferecido ao programador um acesso centralizado e uniforme (i.e., atributos explicitamente representados ou inferidos localmente são acedidos da mesma forma que os gerados remotamente).

Os conceitos que suportam programação reactiva e orientada por objectos são também usados para estruturar toda a parte de interface, como se ilustra na fig. 3.1.7.

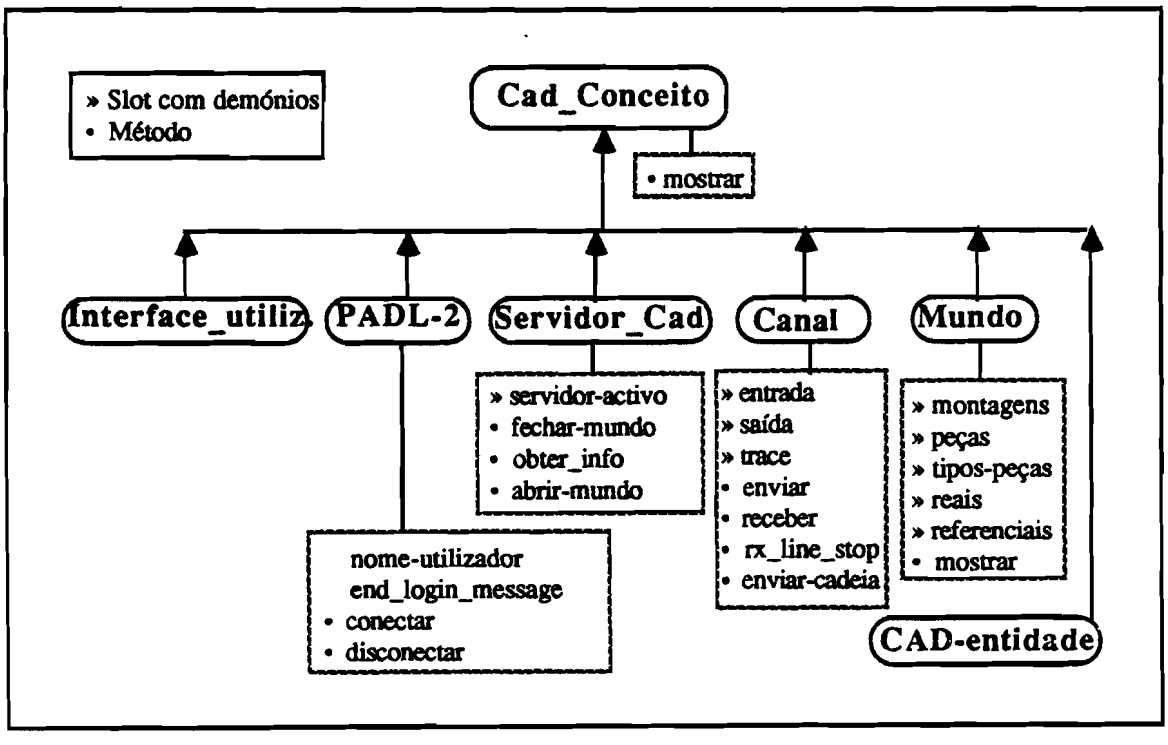


Fig. 3.1.7 Estruturação de conceitos auxiliares à conexão PADL-2 - FE

Note-se, por exemplo, que a linha série de comunicação entre os dois sistemas é representada por um *frame* canal e as primitivas básicas de comunicação por métodos e demónios.

Um aspecto importante nos sistemas CIM é a possibilidade de integrar componentes diferenciados. Por exemplo, pode ser conveniente utilizar diversos CADs através de interfaces similares à anteriormente descrita. Em virtude de ser necessário aceder às representações internas dos modelos CAD, este trabalho está dependente do sistema CAD usado (no caso o PADL-2). A mudança para outro sistema pode implicar um considerável esforço. Desta forma, uma normalização a nível de representações de sólidos seria bastante útil.

Alguns esforços têm sido feitos no sentido de facilitar a transferência de informação entre diferentes sistemas CAD através dum formato neutro. A fig.3.1.8 ilustra as vantagens de utilização de um tal formato.

A norma IGES (Initial Graphics Exchange Specification) é um dos resultados desse esforço, contudo as suas versões preliminares são relativamente pobres em termos de modelação de sólidos [Byt87].

A nível europeu, o projecto Esprit 322 abordou este problema e produziu uma especificação mais poderosa - o CAD*I [Sch86].

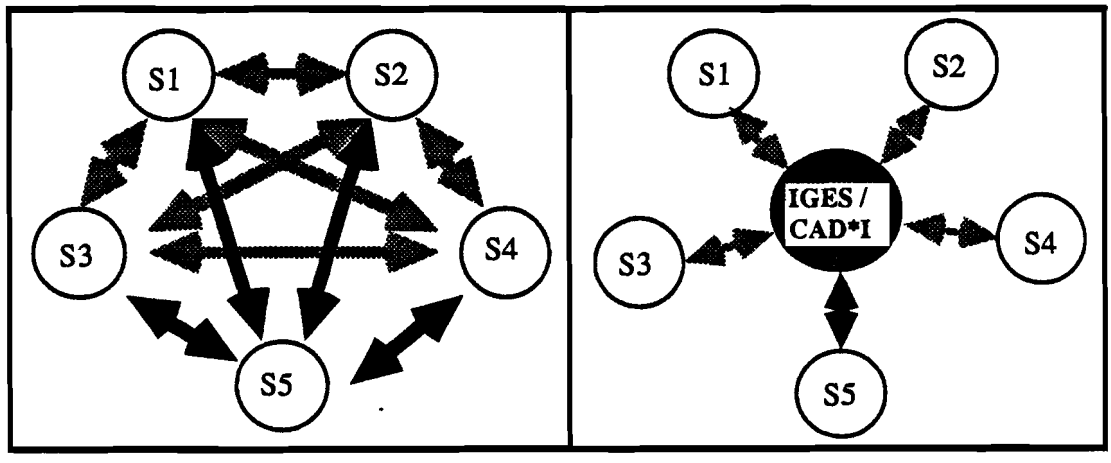


Fig.3.1.8 Vantagens dum padrão para troca de informação entre CADs

Contudo, a ideia base por detrás destes sistemas é a da transferência de informação através de um ficheiro neutro.

A conversão do formato dum CAD particular para o formato neutro é realizada por um pré-processador. A conversão do formato neutro para um CAD específico é feita através dum pós-processador.

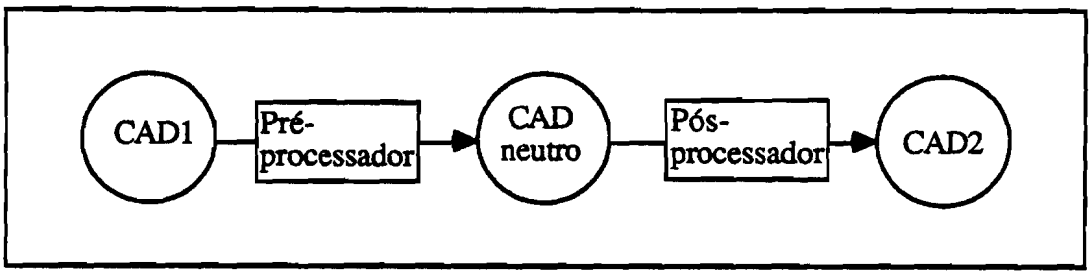


Fig. 3.1.9 Pré- e pós-processadores

Existem diversos pré- e pós-processadores para diferentes CADs, quer para a norma IGES quer para o CAD*I. Contudo estes processadores não são 100% fiáveis. Nem sempre há garantia de sucesso na transferência, especialmente quando os dois sistemas CAD usam conceitos diferentes. Por exemplo, se um sistema representar sólidos em CSG e o outro em B-rep, ambos podem ser convertidos de e para CAD*I (que aceita ambos os sistemas de representação) mas a transferência dum para o outro não é possível (os processadores não fazem conversão de formas de modelação).

Em relação ao domínio deste trabalho subsiste outra dificuldade: é fundamental uma conexão directa e, portanto, a necessidade duma troca interactiva e não via ficheiro.

Por outro lado, estas propostas de normalização estão basicamente orientadas para os modelos geométricos e não para o problema da modelação do processo de montagem. Deste modo, a informação adicional - que, por exemplo, pode ser derivada dos dados CAD - não está normalizada sendo, pois, uma questão em aberto.

A implementação realizada (integrando pré- e pós-processamento) foi, contudo, influenciada pelo CAD*I - conceitos, formato de protocolo - nos aspectos cobertos por aquele sistema.

Detalhes de implementação podem ser encontrados em [CamSteBat87] e [MouCam88].

b) Integração BD- FE

A necessidade de escolher entre a eficiência e a expressividade tem sido uma fonte constante de conflitos na evolução da representação de dados. Por um lado, a tecnologia de bases de dados orientou-se, preferencialmente, para o desenvolvimento de sistemas bastante eficientes de forma a poder suportar aplicações com necessidades intensivas de processamento de dados. A tecnologia de bases de conhecimento, por outro lado, tem dado particular atenção aos problemas de expressividade uma vez que pretende satisfazer aplicações (diagnóstico, planeamento e outras) com necessidades de processar conhecimento sobre pequenos conjuntos de dados.

Para além destes factores, os sistemas de bases de dados permitem, em muitos casos, acesso concorrente à informação em ambientes multi-utilizadores, mantendo a integridade e consistência dos dados, possuem características de portabilidade e memória a longo termo elevadas e são o reflexo de uma tecnologia perfeitamente estabilizada e largamente utilizada mesmo para armazenar informação de sistemas CIM. É de notar que, embora não existindo ainda nenhuma padronização, a linguagem de interrogação SQL começa a representar um padrão de facto.

Parece assim razoável pensar em sistemas que tornem possível gerir informação com a eficiência dos sistemas de bases de dados e com as capacidades de expressão dos sistemas de bases de conhecimento. Nesta linha de pensamento, no Grupo de Robótica da UNL foram realizadas algumas experiências para testar e avaliar a viabilidade daquelas ideias [CamFer88][CamFerMou88].

O modelo de arquitectura adoptado tenta aproveitar os pontos fortes das duas tecnologias. Pretendeu-se basicamente conseguir que várias bases de conhecimentos (BC) pudessem aceder concorrentemente aos dados e aos serviços de gestão de um SGBD. Os utilizadores das BCs podem ser locais ou remotos, i.e., podem fazer parte de uma rede através da qual acedem aos recursos centralizados do SGBD.

O protótipo realizado utiliza o SGBD Rdb, correndo sobre uma VaxStation II, e o Knowledge Craft, correndo sobre uma VaxStation 3200. Os dois equipamentos estão interligados pela Ethernet.

Um dos problemas no estabelecimento da conexão entre os dois sistemas é o da correspondência entre as formas de representação. Na figura 3.1.10 está representado um exemplo de correspondência suportado por uma estrutura de *frames* com dois níveis.

O primeiro é constituído por um *frame* "conceito" que incluirá informação (fornecida pelo conversor, que a obtém do dicionário da BD) sobre a relação, ou relações, em que se baseia. Estes *frames* são carregados na memória da BC quando se estabelecer a conexão com a BD através do comando ABRIR-BD.

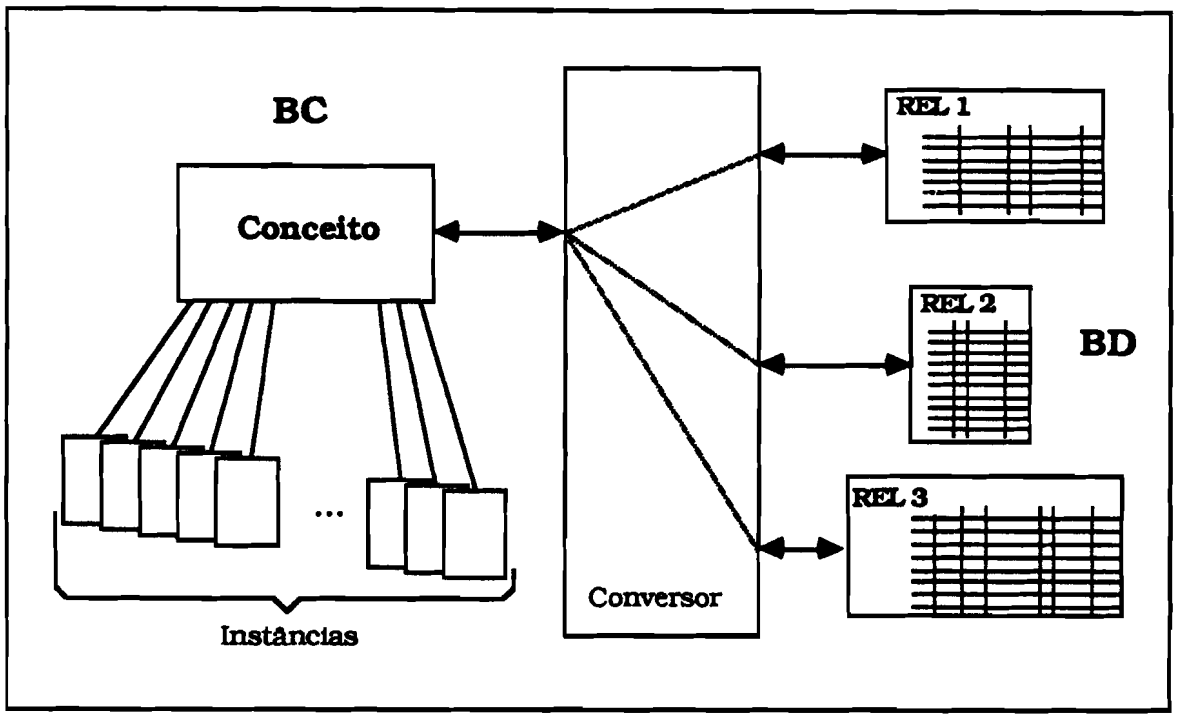


Fig. 3.1.10 Correspondência de estruturas entre BD e FE

Os tuplos seleccionados na BD serão representados na BC como instâncias desse *conceito* (2º nível). Estas instâncias poderão ser "reais" ou "virtuais". No primeiro caso os respectivos *slots* são carregados com os dados lidos na BD quando da criação das instâncias. No segundo caso, os *frames* virtuais permanecem "vazios" e quando se acede a qualquer dos seus *slots* provoca-se, indirectamente - via programação reactiva - um acesso à BD (fig. 3.1.11).

É de notar que a ideia de integração entre BD e BC começa a reflectir-se em produtos comerciais (o KEEconnection [Int87] é um exemplo) que, contudo, apenas oferecem a possibilidade de carregamentos reais.

A noção de *frame* virtual tem como principal vantagem o facilitar a implementação de visões actualizadas dos dados em situações de acesso concorrente e o não carregar a memória de trabalho da BC com duplicação de dados existentes na BD. Como principal desvantagem poderá apontar-se o implicar um maior tempo de resposta aos acessos que se fizerem aos *slots*.

Fornecendo as duas opções - carregamento dos dados ou criação de *frames* virtuais - caberá ao utilizador (programador de aplicações) seleccionar a mais adequada ao problema em causa.

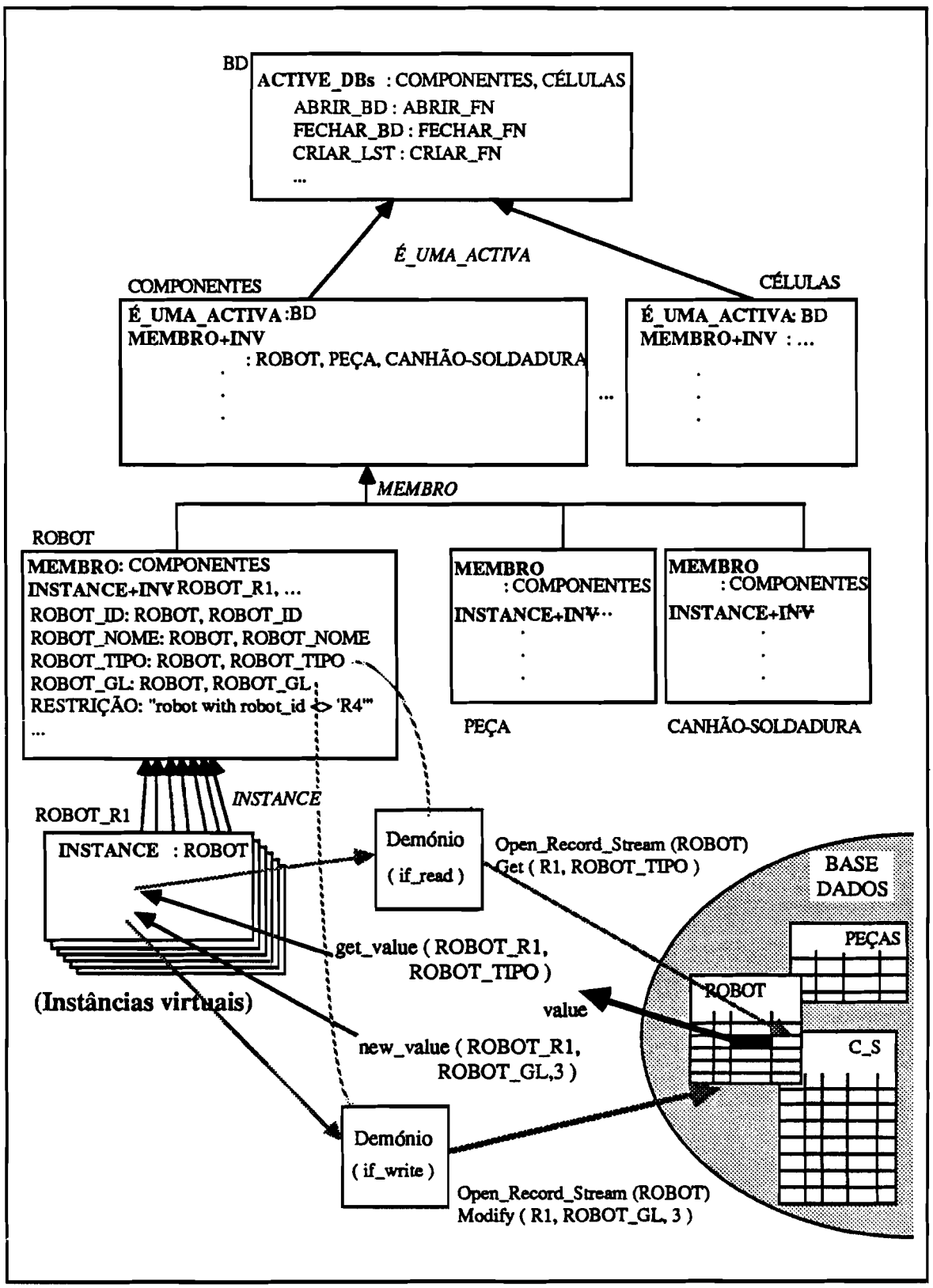


Fig. 3.1.11 *Frames* virtuais na conexão BD - FE

É de notar, contudo, que os problemas de integração dos dois sistemas não se limitam à questão da comunicação. Algumas questões complexas resultam das diferenças em termos de príncipios de estruturação de informação nos dois sistemas. Por exemplo, a uma estrutura linear (tabela) da BD poderá corresponder mais naturalmente uma estrutura hierárquica na BC (representação mais concisa graças ao mecanismo de herança).

Numa primeira fase encarou-se uma aproximação semimanual. Por exemplo, os níveis de estrutura hierárquica são indicados pelo utilizador, enquanto que o "carregamento" das ocorrências de cada nível deve ser automático. No exemplo da fig. 3.1.11, verifica-se que o *frame* BD agrega um conjunto de métodos gerais a utilizar pela conexão. Várias bases de dados podem estar a ser acedidas. Para cada uma existe um *frame* na BC relacionado com o *frame* BD através de "é-uma-activa". Os conceitos baseados numa dada base de dados são relacionados com o *frame* representativo dessa base através da relação "membro".

Por outro lado como a BD não é o único "alimentador" da BC, importa ter uma perspectiva aditiva. Isto é, a estrutura de *conceitos* na BC pode já existir e ser alimentada por múltiplas fontes (note-se que o problema está sendo abordado numa perspectiva de integração).

c) Conexão com elementos da estação

Um dos aspectos importantes no ambiente de desenvolvimento é o da conexão com os elementos físicos da estação. Uma integração, em torno do sistema de *frames*, das facetas funcionais dos controladores desses elementos é discutida no cap. 3.2.3 (Agentes).

Na implementação realizada foram integrados, segundo uma filosofia de conexão interactiva, os controladores do robô e dum tapete rolante e um sistema de percepção baseado em visão, bem assim como um simulador gráfico do robô.

d) Processador de referenciais

O conceito de referencial (posição e orientação) constitui um elemento fundamental nos aspectos de modelação geométrica [AkeBru87]. As operações realizadas pelo robô envolvem essencialmente manipulação e posicionamento de objectos no espaço 3D. As linguagens de manipulação (como o VAL ou o LM) assentam sobre um modo de

representação do universo à base de referenciais cartesianos. O princípio consiste em modelar o utensílio terminal do robô e os objectos do ambiente através dum conjunto de referenciais judiciosamente escolhidos. O programador raciocina, então, em termos de posições e orientações desses referenciais, independentemente da forma dos objectos manipulados e da estrutura mecânica do robô [Lau88]. Referenciais são, pois, usados para modelar o robô e outros componentes da estação bem como os objectos a manipular.

Um objecto, para além do referencial característico ou base que indica a sua localização, pode ter vários outros referenciais associados. (Note-se que este conceito já tinha sido referido a propósito da ligação CAD-FE, ver fig. 3.1.5.)

Pode-se modificar um referencial de localização, por exemplo, através duma operação do tipo

ref * transform

onde ref representa o referencial base e transform exprime a transformação geométrica que permite passar para a nova localização.

Considere-se, por exemplo, a fig. 3.1.12.

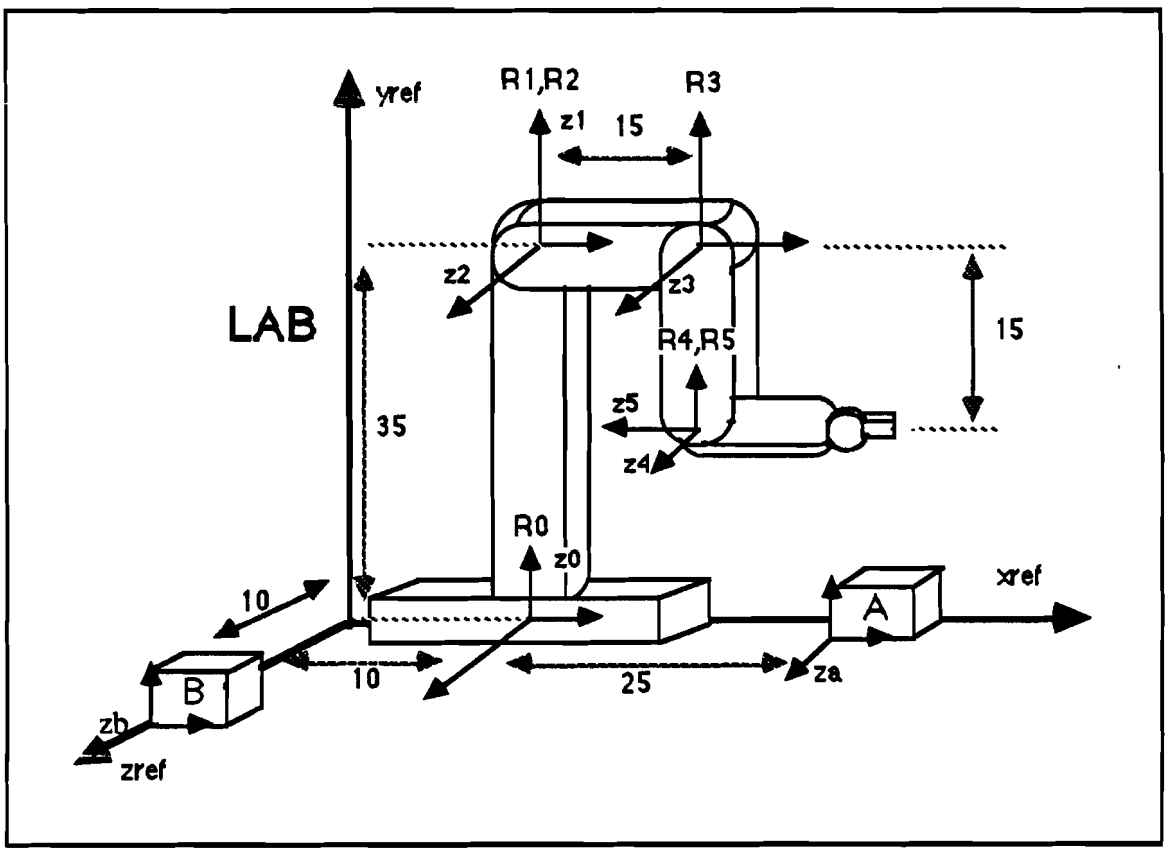


Fig. 3.1.12 Exemplo de uso de referenciais para modelação do mundo

Uma possível solução para a representação dos diversos referenciais envolvidos seria considerá-los definidos em relação ao referencial global da estação - LAB (sistema de referência fixa). Assim, por exemplo, R1 seria representado por uma translação segundo X de 10 unidades, seguido de uma translação segundo Y de 35 unidades e uma rotação segundo Z de - 90 graus. Esta solução é, porém, pouco flexível. Movendo o "antebraço" do robô em torno de Z2 implica actualizar R3, R4 e R5.

Uma outra solução corresponde a utilizar referenciais definidos em termos relativos para os diversos referenciais associados a um componente, isto é, R1 seria definido em relação a Ro, R2 em relação a R1, etc., como se indica na fig. 3.1.13. O referencial de localização duma peça que esteja sendo segura pelo robô também pode ser relativo à garra.

O estabelecimento duma relação entre dois referenciais associados a dois objectos implica ainda a especificação do tipo de ligação entre esses objectos, isto é, conhecimento da estrutura física do ambiente.

Podemos ter ligações com carácter permanente ou temporário. Duas partes do braço dum robô estão ligadas de forma permanente, enquanto que uma peça dum produto pode possuir uma ligação temporária, inicialmente com o referencial LAB, de seguida com a garra do robô e, finalmente, após a sua montagem, uma ligação permanente com a outra peça onde foi montada.

As ligações permanentes podem ainda ser variáveis ou invariáveis, isto é, permitirem ou não movimento dum componente em relação ao outro. Um exemplo de uma ligação permanente variável é a que se encontra estabelecida entre duas partes do braço do robô: embora permanentemente ligadas, a parte descendente pode-se mover em relação à ascendente.

Note-se que as ligações temporárias são, obviamente, variáveis.

A direcção de movimento de uma parte, quando exista, bem como o seu tipo de ligação devem estar expressas no seu referencial característico. Na fig. 3.1.13. mostra-se a estrutura de referenciais correspondente à situação da figura anterior.

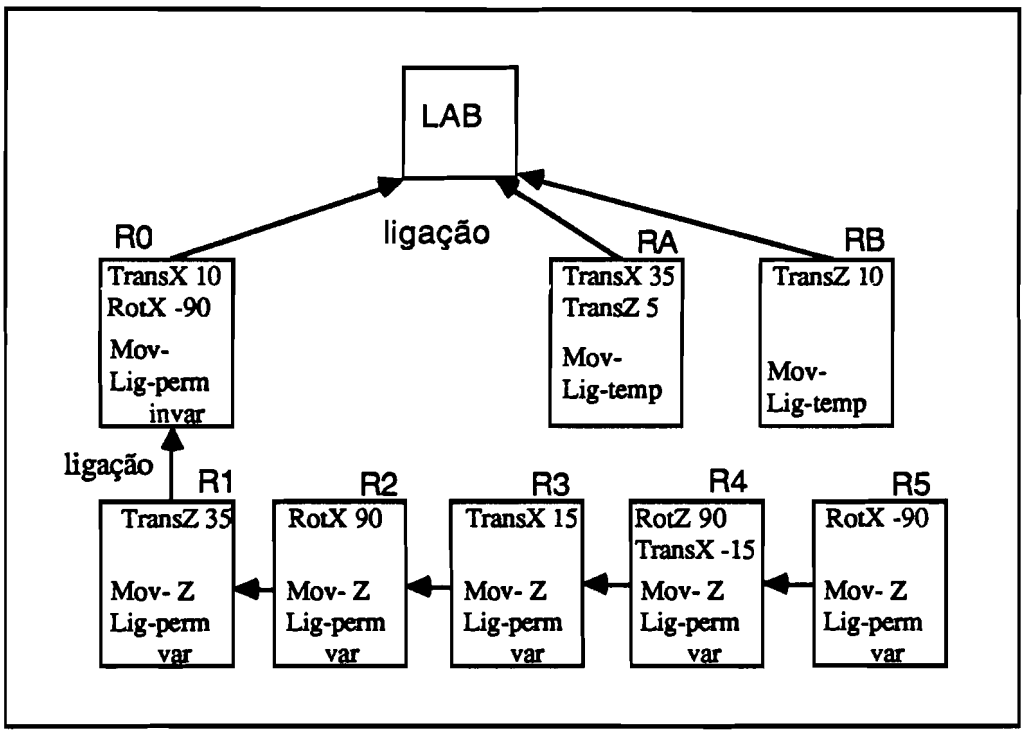


Fig. 3.1.13 Exemplo de estrutura hierárquica de referenciais

Vários dos componentes já referidos (caso do CAD ou do controlador de robôs) incluem o conceito de referencial e têm embebido um processador de referenciais. Em termos do sistema de programação para o nível estação, a mesma necessidade se coloca. Neste sentido, no ambiente de desenvolvimento UNL foi integrado um processador de referenciais desenvolvido sobre Knowledge Craft [PitCam88].

As funções desempenhadas por este módulo incluem:

-Manipulação do conceito de referencial mundo (ou LAB) e referenciais característicos dos objectos.

-Manipulação de relações de ascendência e descendência entre referenciais - estruturas hierárquicas.

-Manipulação de diferentes tipos de ligações.

-Cálculo automático da localização de um referencial em relação a outro qualquer.

-Manter, com coerência e integridade, os valores das relações entre referenciais.

O conceito referencial é implementado por um *frame* onde qualquer alteração é tratada através de programação reactiva:

```
{{Referencial:
    Ligação:
    Tipo-ligação:
    Valor:              ---> demónio  dem-valor (if-write)
    Desc-transf:
    Transformação:      ---> demónio dem-transformação (if-read)
    Mundo:              ---> demónio dem-mundo (if-read)
    Criar-mundo:        cr-mundo-fn     |
    Criar-ref:          cr-ref-fn       |
    Ler-transformação:  ler-transf-fn   |
    Consultar-ref:      cons-ref-fn       ===> métodos
    Alterar-posição:    alt-ref-fn      |
    Nova-ligação:       n-lig-fn        |
    Apagar-ref:         apagar-lig-fn   |
    ...
    }}
```

Os diversos referenciais a manter no sistema serão criados como instâncias deste conceito. É o caso, por exemplo, dos referenciais associados a uma peça, referidos anteriormente na interface CAD - FE.

Como já foi referido, um sistema CIM é, do ponto de vista computacional, um sistema distribuído. No caso duma estação, de acordo com a aproximação aqui seguida, também se recorre a um sistema computacional distribuído, donde resultam problemas de comunicações. No caso das implementações realizadas, usou-se a rede local DECnet/Ethernet ou conexões directas entre equipamentos (RS 232) com protocolos específicos definidos para o efeito.

No caso genérico dum sistema CIM onde se pretende a integração de múltiplas "ilhas de automatização" (estações de maquinagem, montagem, projecto, administrativas, etc.), podem-se dividir as áreas de aplicação em dois grandes grupos: fabricação e escritório (que, contudo não são independentes). Alguns esforços de normalização de protocolos para redes locais que têm vindo a ser desenvolvidos para estes domínios são representados por MAP (*Manufacturing Automation Protocol*) e TOP (*Technical Office Protocol*). Estas propostas pretendem ser o suporte, em termos de protocolos de rede

local, para os futuros sistemas CIM. Uma apresentação detalhada pode ser encontrada em [Cor87].

A adopção de tais normalizações facilitará a interligação dos vários componentes na direcção dum sistema de controle integrado. A nível de cada estação poder-se-ão ter soluções específicas ou usar versões simplificadas do MAP.

3.1.3 - INFORMAÇÃO DE PARTIDA

Fronteira entre planeamento de sistema e programação

Algumas dificuldades na especificação da informação de partida para a programação resultam de não ser muito nítida a fronteira entre esta actividade e a fase preparatória (de planeamento de sistema). Esta "fluidez" da linha de separação é consequência do estado ainda pouco estruturado das áreas de planeamento de processo e concepção de produto e célula.

Neste contexto, as propostas que aqui se apresentam constituem uma tentativa de clarificação do ponto de partida para o sistema de programação, mas também algum contributo para a estruturação das fases anteriores. A este propósito é de referir a aceitação destas propostas como forma de estruturação de actividades na área de "Concepção e planificação da célula" do projecto ibero-americano "Robótica y Automatización Avanzada" [Cyt88].

É claro que o grau de trabalho preparatório que se assume como ponto de partida tem importantes consequências no nível de complexidade das tarefas subsequentes. Estas poderão ser mais ou menos simplificadas dependendo da quantidade de informação de que se dispõe. (Todavia, este não deve ser o critério para a definição do ponto de partida, já que poderia facilmente conduzir a uma transferência de problemas com eventual esvaziamento de responsabilidades do módulo de programação!) Assim, o critério seguido é baseado na interpretação que fazemos do estado da arte e tendências na automatização das zonas de planeamento de sistema, embora tendo sempre em atenção que se deverá tentar recuperar o máximo de informação disponível ou gerada nessa fase. Isto é, pretende-se uma transição entre as duas fases sem perdas de informação que possa ser útil.

Como se referiu anteriormente, a especificação da tarefa para o sistema de programação (solução implícita) deve incluir: modelo do produto (só as partes que interessam à programação), especificação do processo e modelo da estação executora.

Especificação do produto

Os sistemas CAD constituem fontes naturais dos modelos do produto a montar e respectivos componentes, já que são os utensílios usados para a sua concepção.

Estes sistemas suportam fundamentalmente um modelo geométrico dos objectos, sendo pouco flexíveis para a inclusão de outra informação. Em geral limitam-se a suportar, de forma insipiente, alguma "documentação" adicional sob a forma de atributos: material, tolerâncias, cor, etc..

Uma parte dos sistemas existentes foi concebida tendo apenas em vista a função isolada de "editor" geométrico, ou prevendo uma conexão a um sistema CAM com o objectivo de produção de peças em máquinas de controle numérico. Porém, as tarefas de montagem, por a sua automatização ainda ser difícil, não encontram grande suporte, mesmo em termos de informação geométrica, em muitos dos sistemas comerciais. A noção de montagem (*assembly*) está presente em vários sistemas, mas outros items de informação necessários à especificação do processo de montagem (forma de aproximação entre dois componentes, por exemplo) não é contemplada. Como consequência, tornam-se necessários processos auxiliares de "edição" ou geração da informação complementar e também suporte para a sua representação.

Para diferentes actividades do sistema de programação e controle há necessidade de considerar diferentes visões do produto / peça. Por exemplo:

-Programação genérica: interessa uma abstracção em que os objectos sejam representados por um conjunto de referenciais: base (localização), de preensão, de aproximação e afastamento durante operações de preensão, etc. (ver fig. 3.1.18).

-Determinação de trajectórias livres de colisões: interessa um modelo sólido simplificado - caixa ou esfera envolvente, por exemplo, se for suficiente uma trajectória esqueleto sem grandes necessidades de optimização.

-Percepção: interessa um modelo geométrico mais detalhado, por vezes 2D, ou uma abstracção derivada: centro de massa, número de buracos, área, perímetro, eixos principais, etc.. Por outro lado interessa também conhecer a relação entre o referencial usado na localização do objecto pela percepção (normalmente localizado no centro de massa) e o referencial base usado na definição do modelo geométrico.

O modelo representado na fig.3.1.14 está fundamentalmente dirigido para os requisitos da programação genérica.

Em termos de representação, a aproximação defendida passa pelo recurso à conexão CAD-FE (ou SGBC) descrita atrás.

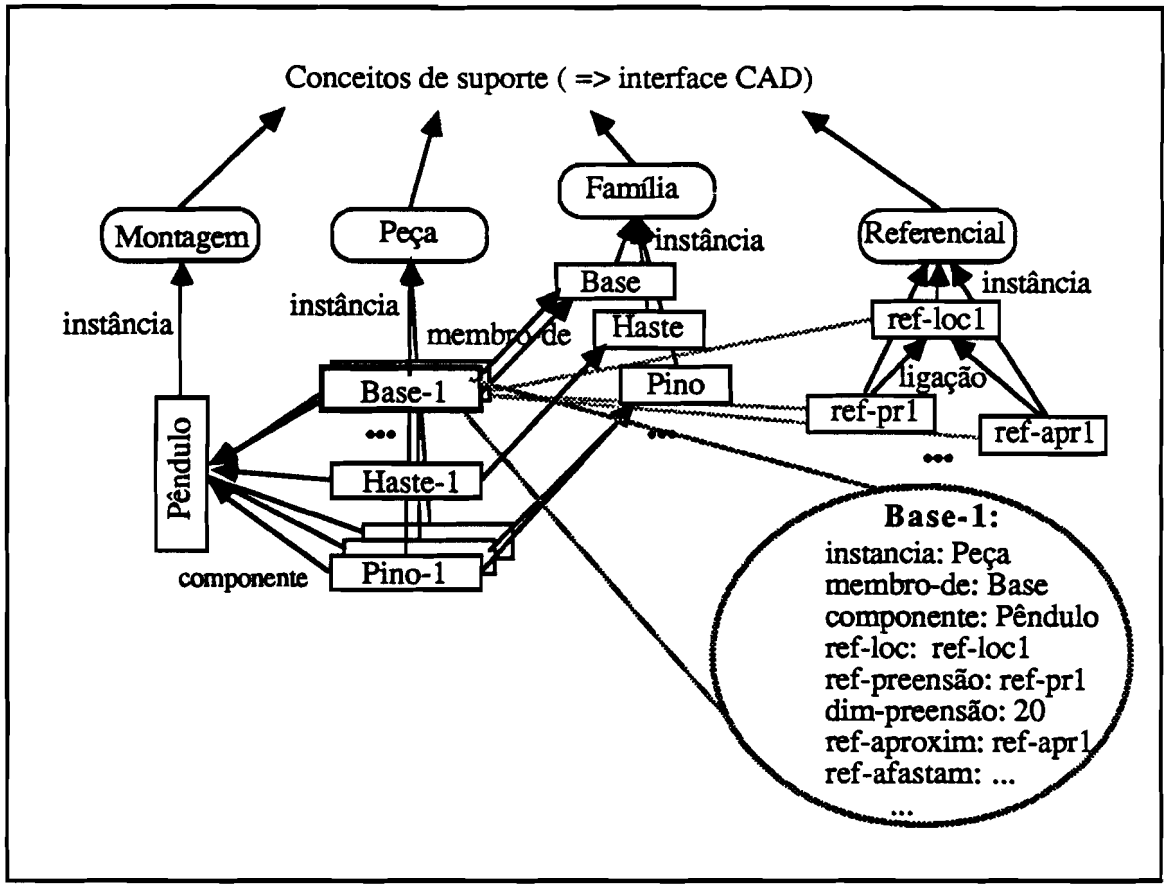


Fig. 3.1.14 Representação do produto

Uma abstracção dos modelos geométricos é materializada no sistema de *frames* através dum conjunto de instâncias dos conceitos de suporte da interface CAD (montagem, família, peça, ...). A informação complementar relativa ao processo é apenas representada no sistema de *frames* e não na BD do CAD.

No exemplo, estão representadas várias relações:

•*instância,* relação pré-definida que permite ligar, por exemplo, um objecto concreto com o conceito de PEÇA. Através desta relação, as propriedades de PEÇA são herdadas por BASE-1.

•*membro-de*, relação que permite ligar uma peça com o seu protótipo ou modelo. O conjunto de membros dum protótipo constitui uma família. Esta é, pois, uma relação específica criada usando as facilidades do Knowledge Craft para a definição de novas relações:

(membro-de
is-a: relation
inclusion: membro-de-inclusion-spec
transitivity: (step membro-de T)
inverse: membro-de+inv
)

(membro-de-inclusion-spec
is-a: inclusion-spec
type: ...
slot-restriction: (not (type is-a relation))
)

Conforme especificado em *inclusion,* através desta relação, apenas são herdados os atributos de família que não sejam relações. De notar que BASE é uma instância de FAMILIA.

•*componente* - serve para ligar ao PÊNDULO todas as peças componentes dessa montagem.

Por outro lado, é de notar que a uma peça estão associados vários referenciais - instâncias do conceito REFERENCIAL. Um deles é o referencial característico da peça (ref-loc). Os restantes são relativos a este, o que se indica pela relação *ligação.* Conforme descrito atrás, o valor de qualquer destes referenciais é automaticamente determinado (pelo processador de referenciais) a partir do seu ascendente.

Os vários items de informação que constituem o modelo do produto, de acordo com as necessidades do processo de montagem, terão origem em diferentes actividades da fase preparatória (fig.3.1.15).

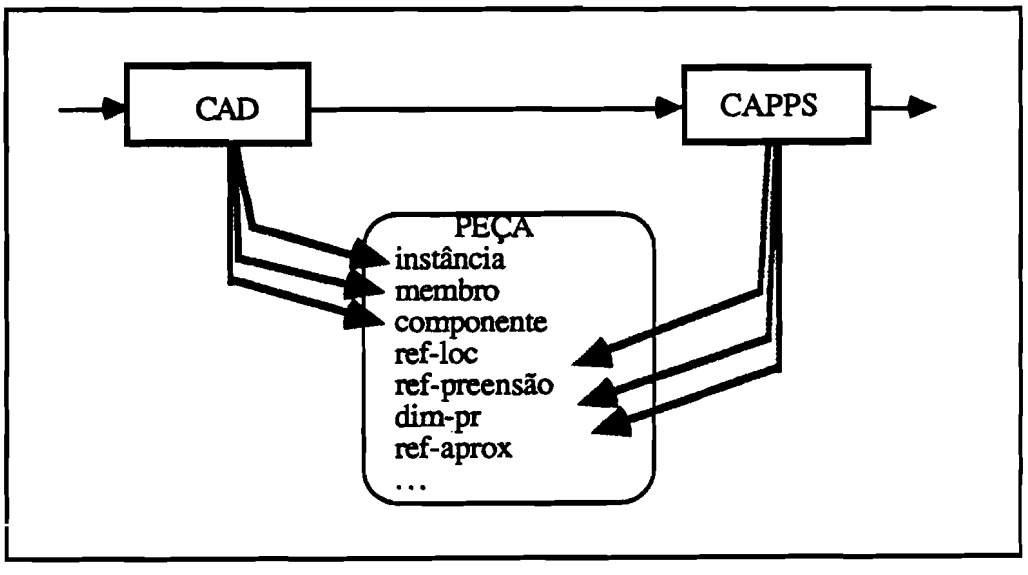


Fig. 3.1.15 Origem dos atributos duma peça

Assim, por exemplo, elementos como estrutura do produto, referenciais de base, etc., resultarão da fase de concepção (CAD), enquanto os referenciais de preensão, aproximação, afastamento, etc., estão mais relacionados com o próprio processo de montagem e, portanto, derivarão da fase de planeamento de processo (CAPPS).

Contudo, devido à não existência de sistemas integrados, esta informação perde-se ou não é adequadamente inserida nos modelos de suporte à programação, situação que se pretende evitar com a arquitectura proposta.

Especificação do processo

Esta pretende ser uma descrição da tarefa em termos abstractos e independente, portanto, dos operadores concretos dos agentes executores a nível da célula.

Embora existam algumas propostas de linguagens nível tarefa, ainda não existe uma "normalização" ou mesmo um consenso sobre os operadores a usar a este nível.

Um esforço inicial de Kondolean, reportado em [NevWhi78], permitiu evidenciar uma lista de operadores abstractos para a descrição de tarefas de montagem. Através de experiências realizadas com alguns produtos típicos (torradeira, bicicleta, motor eléctrico, alternador de automóvel, etc.) foi também possível estabelecer uma tabela de distribuição de frequências para essas operações (Tab. 3.1.1).

Operação	Frequência
Inserção de pino	34.5
Premir e rodar	12.8
Inserção de múltiplos pinos	6.5
Inserção de pino com retenção	5.0
Inserção de parafuso e/ou cavilha	26.8
Ajuste com pressão	7.3
Remover pino de fixação	1.0
Virar posição	2.0
Fixação temporária	1.5
Fixação de chapas metálicas por dobragem	0.5
Remover fixação temporária	1.5
Soldar	0.5

Tab. 3.1.1 Operadores abstractos de montagem de Kondolean

Pelo menos para o grupo de produtos considerado, verificou-se que a inserção de pinos é a operação mais significativa, logo seguida da inserção de parafusos e cavilhas.

Outro aspecto analisado por Kondolean foi o da direcção de aproximação das peças em relação à montagem. Esta direcção é determinada pela localização da peça no produto final, pela forma de "encaixe" e por outras peças que bloqueiem a sua aproximação. Nos items inspeccionados foi observado que (excepto na torradeira) cerca de 60% das peças chegam segundo uma mesma direcção (Z), 20% segundo a direcção oposta (-Z) e 10% segundo um plano perpendicular ao eixo anterior; os restantes 10% chegam segundo outras direcções. Por outras palavras, ele concluiu que estes produtos são essencialmente pilhas do ponto de vista do processo de montagem.

Os resultados deste e doutros trabalhos onde se tem tentado fazer uma caracterização dos diversos problemas da montagem têm consequências não só no planeamento de processo mas também na própria concepção / selecção dos robôs. Por exemplo, uma distribuição de tarefas por grupos de acordo com as direcções privilegiadas permitirá a sua afectação a células especializadas, mas com componentes mais simples (o que nos leva de novo à Tecnologia de Grupos [Mac86] [KusHer87] e também à própria concepção do produto - desenho para montagem).

Do ponto de vista de especifição de processo, embora não se possa falar duma aceitação generaiizada, os operadores apresentados têm constituído uma plataforma para vários trabalhos a nível de planeamento de processo (veja-se, por exemplo [TieBowBro86], [AyrFun89]). No trabalho desenvolvido adoptámos um conjunto de operadores do mesmo nível de abstracção.

A fig.3.1.16 apresenta um exemplo de taxionomia para os operadores abstractos usados.

Uma operação como "Inserção-de-pino" herdará, através da relação é-um (*is-a*), as propriedades dos conceitos de "Inserção" e "Operação-de-montagem".

Alguns dos items que caracterizam uma operação de montagem são, então:

-Componente1, peça a ser montada no fixador ou sobre a peça Componente2.

-Ferramenta, indica o utensílio a usar pelo robô para a realização da operação.

-Tolerância, indica se se trata duma operação de elevada precisão (requerendo movimentos complacentes, por exemplo) ou não.

-Aproximação e afastamento representam, respectivamente, referenciais (trajectórias) de aproximação e afastamento da garra / ferramenta em relação à peça quando da execução da operação.

-ref-m12, ref-m21: referenciais que permitem especificar a situação de montagem de componente 1 no componente 2 (ver fig. 3.1.16).

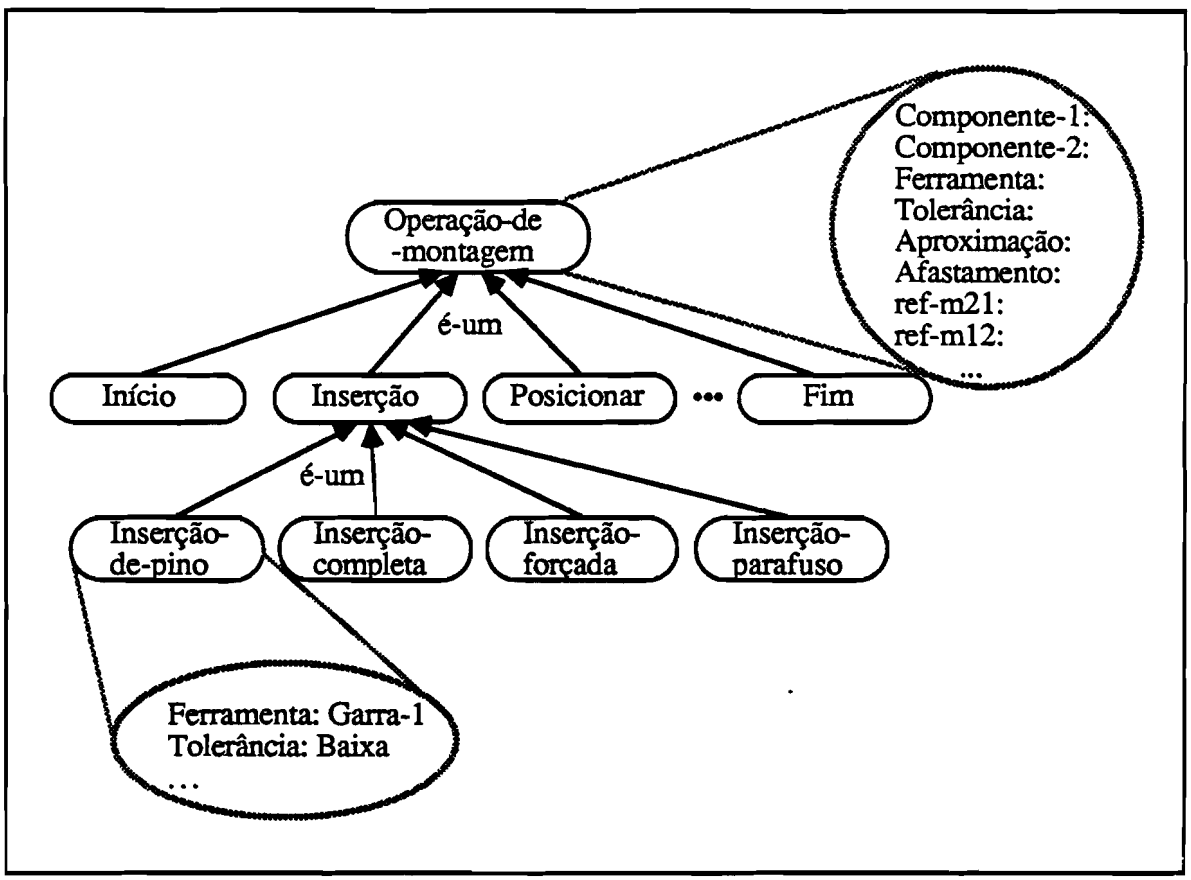


Fig. 3.1.16 Taxionomia de operadores abstractos

A descrição da tarefa - plano de processo ou gama de fabrico - será formada por um conjunto de instâncias destes operadores abstractos (folhas da taxionomia Operação-de-montagem).

Para cada instância podem-se preencher os respectivos *slots* com valores concretos (fig. 3.1.17). Por outro lado, na taxionomia podem-se indicar alguns valores - valores de defeito - que serão herdados pelas instâncias se os correspondentes *slots* não receberem qualquer valor local. Por exemplo, a Ferramenta a usar em OP-NODE-2 (fig. 3.1.17) será, por defeito, a Garra-1 (fig. 3.1.16), mas a Tolerância será normal (definição local) em vez de baixa (valor de defeito).

As precedências entre operadores, no plano de processo, podem ser representadas através dum grafo dirigido (parcialmente ordenado) conforme se exemplifica na fig.3.1.17.

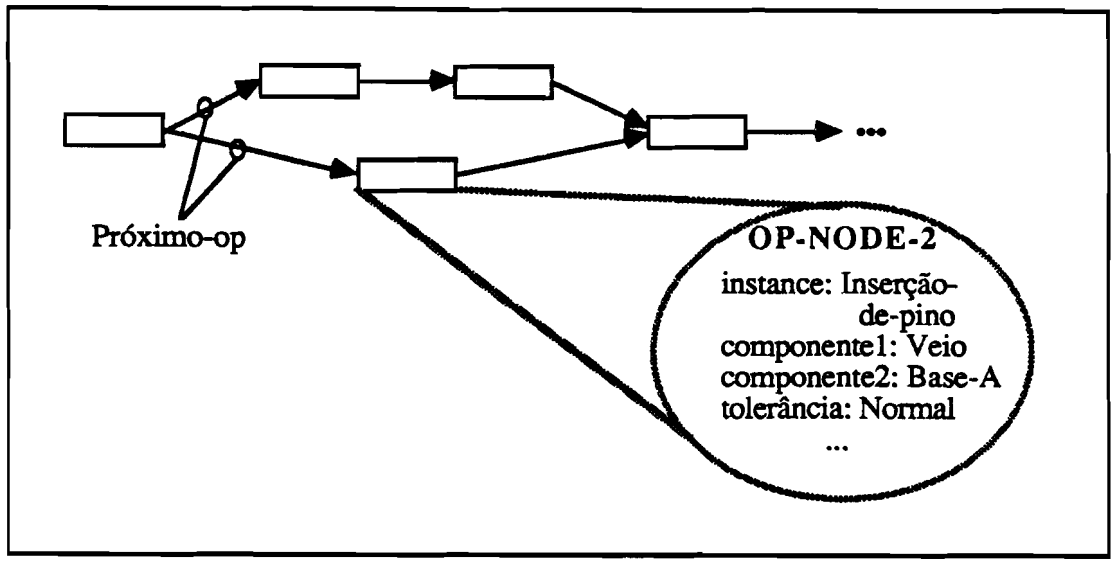


Fig. 3.1.17 Gama de fabrico / plano do processo

As precedências entre operadores e, consequentemente, a delimitação de potenciais paralelismos de execução poderiam, numa primeira análise, ser determinadas apenas por raciocínio geométrico.

Um trabalho nesta direcção tem sido desenvolvido na Universidade de Karlsruhe [Spu...87], em que se tenta determinar o grafo de precedências a partir dos seguintes critérios:

-Uma tarefa de montagem de n componentes necessita de n operações (operadores abstractos).

-A penetração mútua de componentes não é permitida.

-Cada estado intermédio da montagem deve ser estável.

-O risco de (colisões em) uma operação de montagem deve ser o menor possível.

Durante a programação, uma eventual restrição a esse potencial paralelismo de execução seria determinada por uma fase de raciocínio sobre os recursos disponíveis na estação executora.

É, porém, de notar que a fase de planeamento de processo não é completamente independente da estação. A própria concepção do produto pode já ter sido influenciada por uma determinada filosofia de fabrico. Sendo esta interacção entre as áreas de concepção e produção uma situação desejável, é razoável admitir que o grafo de montagem já exprima os paralelismos possíveis (e não todos os potenciais) (escalonamento -*scheduling* - das operações)- questão da recuperação a nível da programação de informação existente ou gerada na fase anterior. Esta é a assunção de que

se parte. Alguns trabalhos de investigação em curso nesta área permitem justificar tal opção. É, por exemplo, o caso dum sistema em desenvolvimento no Departamento de Engenharia Industrial da University College Galway, Ireland [TieBowBro86] em que o resultado do planeamento de processo inclui a ferramenta a usar e indicação da necessidade ou não de um segundo robô.

Como consequência directa ou indirecta desta fase, um importante conjunto de referenciais para a especificação do processo de encaixe de duas peças tem de ser estabelecido como preparação para a programação. A fig.3.1.18 ilustra essa informação e indica também onde tais referenciais são representados (modelo do produto e plano de processo) - uma das possíveis estruturações.

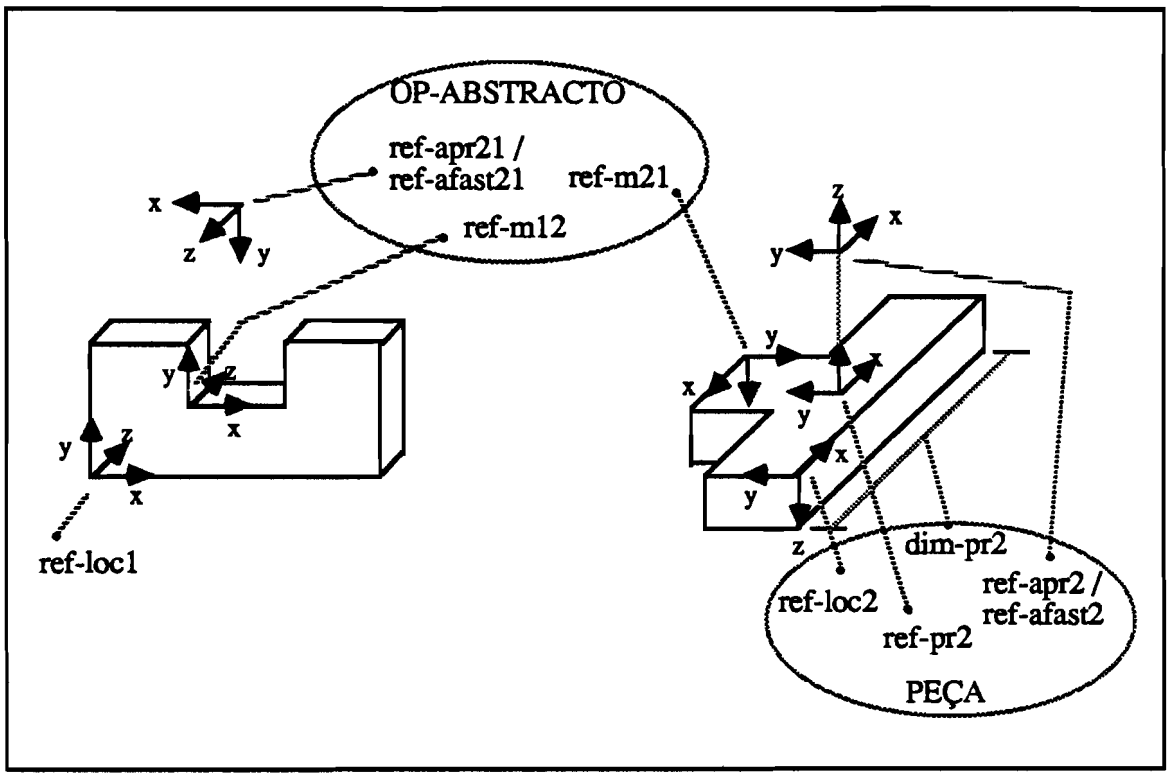


Fig. 3.1.18 Referenciais para a especificação do processo de montagem

Para montar a peça 2 na peça 1:

-ref-m21 e ref-m12, permitem especificar o objectivo: as duas peças estão montadas quando estes dois referenciais ficarem justapostos.

-ref-apr2 - indica o ponto de aproximação e a orientação da garra quando vai pegar a peça 2. A partir deste ponto o deslocamento deve ser com movimento fino, através duma translação segundo o eixo dos Z (no exemplo). De forma semelhante, ref-afast2 indica a

posição e orientação de afastamento da garra após pegar a peça. Por outras palavras, estes referenciais delimitam as fronteiras onde os movimentos são finos, eventualmente complacentes.

-ref-pr2 - indica o ponto para a preensão da peça 2 (*grasp point*). A abertura da garra é dada por dim-pr2. De notar que este é um parâmetro bastante dependente do tipo de ferramenta usada. Uma garra de sucção, por exemplo, não necessitaria de tal parâmetro.

Todos estes referenciais são relativos ao referencial de base da peça - ref-loc2.

-ref-apr21 e ref-afast21 têm um papel similar a ref-apr2 e ref-afast2 no sentido em que delimitam uma zona de movimentos finos, mas referem-se agora ao encaixe da peça 2 na peça 1. Mais genericamente, em vez de um, poder-se-ia ter um conjunto de referenciais (definindo uma trajectória) quando a geometria das peças / montagens parciais força uma aproximação (ou afastamento) segundo uma trajectória complexa.

Numa primeira análise pode-se argumentar que este tipo de especificação é bastante detalhado e até um tanto de baixo nível (próximo do detalhe a usar em linguagens de nível manipulador).

Numa via alternativa, vários sistemas de especificação textual de relações espaciais entre objectos têm sido propostos com a finalidade de facilitar a descrição de tarefas de montagem. Um dos exemplos mais conhecidos é o RAPT [PopAmbBel78] [AmbCamCor86] que usa relações espaciais simbólicas entre características dos objectos (faces, eixos, vértices) tais como AGAINST, COPLANAR, COAXIAL, PARALLEL para descrever as configurações pretendidas. Exemplo (em sintaxe livre):

move OBJECT1 such_that
Face1 of OBJECT1 against Face1 of OBJECT2 and
Face2 of OBJECT1 coplanar Face2 of OBJECT2 ...

Todavia, a passagem duma descrição deste tipo para um programa executável pelo robô, onde os operadores recebem como argumentos referenciais num espaço 3D, não é tarefa fácil, envolvendo um pesado raciocínio geométrico. (Ex.s: RAPT e LM-GEO.)

Uma segunda dificuldade resulta de os sistemas CAD / modeladores de sólidos nem sempre facilitarem tais descrições, já que muitas vezes as características (faces, eixos, vértices) envolvidas são anónimas. Desta forma, uma fase prévia de atribuição de etiquetas teria de ser realizada.

Por outro lado, a evolução registada nos sistemas gráficos, suportando modelos 3D, com elevado grau de interactividade e desempenho, reforça a solução de indicação explícita dos citados referenciais. Embora esta aproximação envolva uma considerável

quantidade de informação detalhada, a sua aquisição pode ser feita de forma interactiva com razoável conforto para o utilizador. No contexto corrente, esta é a aproximação que aqui se defende. Contudo, os desenvolvimentos nas áreas de raciocínio geométrico não deverão ser ignorados e eventuais resultados poderão ser incorporados num sistema interactivo, automatizando certos passos da introdução de dados ou validando outros. Por exemplo, referenciais de aproximação / afastamento podem ser calculados a partir do referencial de preensão e dum conhecimento das características da garra. Só no caso de o utilizador querer alterar estes valores teria de os introduzir interactivamente (apontando no écran).

Modelo da estação

Também relativamente à estação, as necessidades de modelação variam com a actividade pretendida. Assim:

-o planeamento de sistema lida com parâmetros mais "macroscópicos" como: número de graus de liberdade; carga, repetibilidade e precisão, volume de trabalho, etc.;

-para a programação genérica interessam "*layout*" (localização dos componentes), modelo funcional (que operadores executam), envolventes (se calcular trajectórias), relação entre componentes (por exemplo, sensores associados), ferramenta corrente, etc.;

-para a simulação gráfica há que considerar o modelo geométrico e cinemático (emulação, pelo menos parcial, dos controladores), envolventes volumétricas, ...;

-para o controle de execução interessam as conexões aos controladores reais, a capacidade sensorial, as eventuais redundâncias funcionais para redistribuição de tarefas em caso de erro, etc..

Neste capítulo ter-se-ão especialmente em atenção os requisitos da programação genérica.

Os atributos caracterizadores dum componente da estação podem agrupar-se em:

•parte "invariante" - características do componente que, em geral, não variam com o tempo - carga, graus de liberdade, localização na estação (se não for móvel), etc.;

•parte operacional (procedimental) - conjunto de métodos que o componente é capaz de aplicar (operadores ou acções primitivas);

•parte dinâmica - parâmetros que representam a evolução de estado durante a actividade do componente: posição corrente (componentes móveis), estado, valores dos sensores, etc..

As facilidades de hierarquização de *frames* através da relação *is-a* e do mecanismo associado de herança, permitem estabelecer uma taxionomia de componentes de estação e conseguir uma representação bastante sintética (fig.3.1.19).

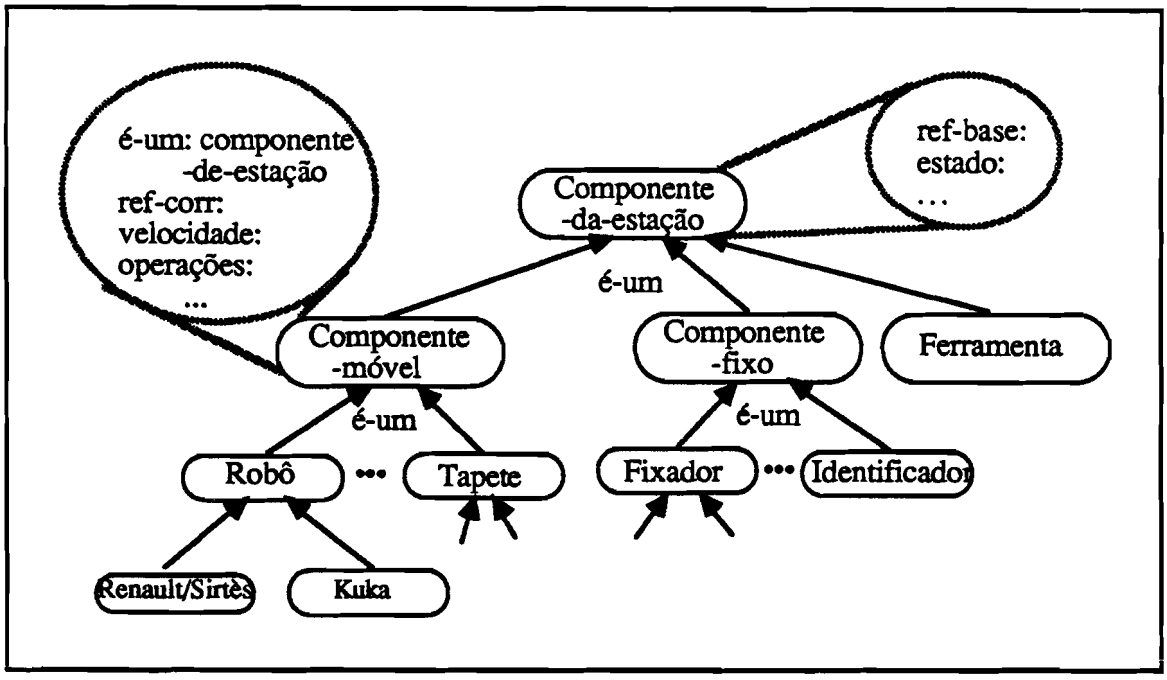


Fig. 3.1.19 Exemplo de taxionomia de componentes da estação

Parâmetros comuns a todos os componentes apenas necessitam ser explicitados no nível mais elevado da hierarquia sendo implicitamente herdados por todos as folhas. A nível do nó "Kuka", por exemplo, tem-se uma especialização particular do conceito de robô que, por sua vez, é uma especialização de componente-móvel.

Um *frame* robô não necessita ter representados, ao mesmo nível, todos os detalhes necessários para as diferentes "visões". Por exemplo, o modelo cinemático pode ser representado por outro *frame* e ligado a robô por uma relação definida para o efeito. O mecanismo de herança permite o acesso a esses atributos sem forçar que eles apareçam todos ao mesmo nível de descrição.

Outra alternativa passa pela utilização de contextos (fig. 3.1.20). A criação dum contexto "filho" pode ser vista como uma cópia (virtual) de todas as estruturas do contexto "pai". Alterações feitas no "filho" não se reflectem no "pai" (a menos que tal seja pretendido.

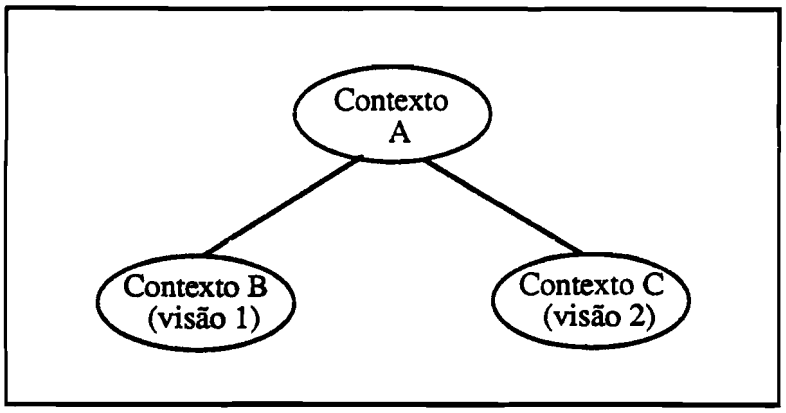


Fig. 3.1.20 Utilização de contextos para suporte de visões diferentes do SI

No exemplo da fig. 3.1.20, os atributos correspondentes a uma dada visão podem ser representados no contexto B, enquanto os duma segunda visão serão modelados no contexto C. O contexto A conterá os aspectos comuns às duas visões.

Uma estação concreta pode ser modelada por um conjunto de instâncias das folhas da taxionomia de componentes de estação (fig. 3.1.21). Nestas instâncias serão preenchidos parâmetros concretos para os componentes reais como, por exemplo, a localização do elemento na estação.

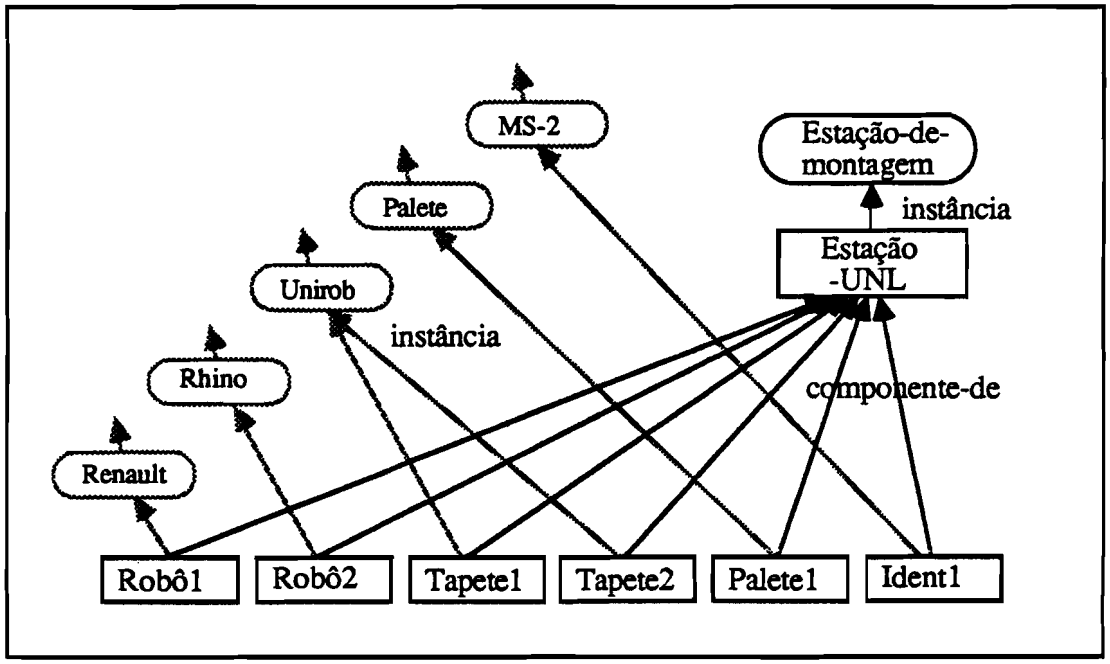


Fig. 3.1.21 Exemplo de estação concreta

Os componentes duma estação não funcionam de forma completamente autónoma. O mecanismo de definição de relações entre *frames* (quando esteja disponível no *frame engine* usado) permite representar essas dependências. Por exemplo, entre o robô e as respectivas ferramentas pode-se estabelecer uma relação *ferramenta-corrente* e sua inversa *utilizador-corrente.*

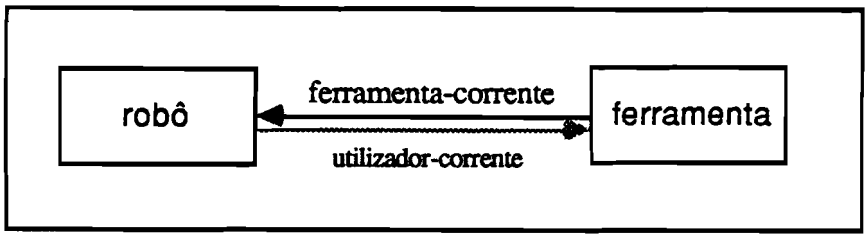


Fig. 3.1.22 Relação entre robô e ferramenta

Nesta relação deve-se ter cardinalidade 1 (o robô só usa uma ferramenta de cada vez) e pode-se definir herança de operações:

{{ferramenta-corrente:
instance: relation
inverse: utilizador-corrente
inclusion: ferram-corr-her ------> {{ferram-corr-her:
instance: inclusion-spec
slot-restriction:
(or pegar posicionar)*
}}
}}

Isto é, o conjunto de operações realizáveis pelo robô é variável com a ferramenta que está usando a cada momento. Exemplo: ao conectar o robô com uma garra possuindo as operações PEGAR e POSICIONAR, o conjunto de operações básicas do robô é enriquecido com estas duas.

Por outro lado, o estabelecimento duma relação (temporária) entre os referenciais dos dois componentes, leva a que, alterando um, o outro fique implicitamente alterado - com base na funcionalidade do gestor de referenciais - sem que seja necessário efectuar

* O exemplo aqui representado é um caso particular - não permite indicar a herança de quaisquer operações mas apenas das explicitadas. Todavia, por razões de clareza de exposição não se mostra uma implementação mais complexa. Uma solução mais genérica pode passar por uma representação como a da fig. 3.1.23.

qualquer operação adicional. Um comando "trocar-ferramenta" encarrega-se de estabelecer estas ligações.

Os aspectos operativos são materializados no modelo através dum conjunto de métodos e demónios que representam os operadores realizáveis pelos agentes. Exemplo:

```
{{robô:
        ...
        estado:          ----------> demónio (if-read ) ...
        mover: robô-move-fn  <----- método
        home: robô-home-fn  <-----   "
        ...
}}
```

Por outro lado, o conhecimento desses operadores e seus efeitos está também representado nas regras do planeador, a discutir em 3.1.5.

Outra alternativa para representar as acções executáveis por cada agente está representada na fig. 3.1.23. A relação "operações" permite ligar o agente com os *frames* que descrevem as respectivas acções primitivas.

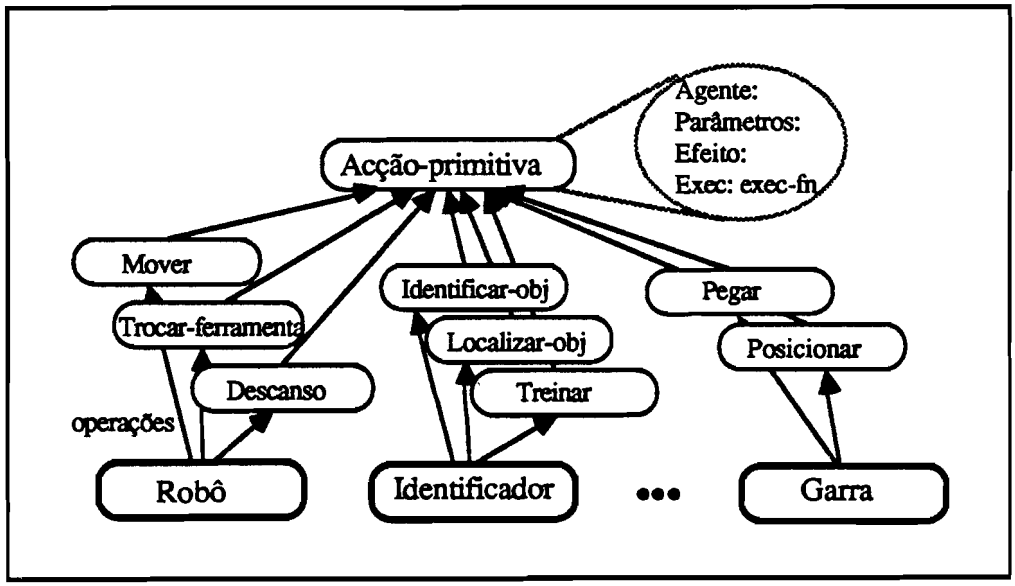


Fig. 3.1.23 Representação das acções executáveis pelos agentes

A lista de componentes e seu posicionamento espacial (*layout*) é um resultado do planeamento de sistema - fase de concepção da estação.

A construção da taxionomia onde efectivamente se têm (directa ou indirectamente) os modelos dos componentes, pode ser feita a partir de informação proveniente de bibliotecas. Alguns fabricantes dispõem de bibliotecas de modelos de componentes suportadas por bases de dados relacionais. A interface BD-FE representa, assim, uma forma de recuperar / aceder a tal informação. Os modelos das "envolventes" são normalmente realizados em sistemas CAD (modeladores de sólidos), existindo também bibliotecas para vários robôs. A ligação CAD-FE permite o acesso a tal informação.

De notar que, como já foi anteriormente referido, as interfaces entre o FE e as "fontes" de informação foram desenvolvidas de forma a contemplar uma perspectiva "aditiva". Isto é, um conceito pode ser instanciado por facetas provenientes do CAD, da BD, etc., e, portanto, cada interface não deve "apagar" e redefinir esse conceito, mas antes adicionar o seu contributo à imagem já existente na BC.

Em síntese, para o modelo da estação, podem-se considerar três aspectos:

-taxionomia de componentes;
-estrutura (elementos constituintes e sua localização);
-modelo funcional
    .operações executáveis pelos componentes
    .relações entre componentes (e entre componentes e peças).

Os aspectos de modelo funcional voltarão a ser discutidos na secção sobre geração de planos.

Relações entre produto e estação

Uma informação vital para a programação e que é também um resultado da fase de planeamento de sistema é a indicação das origens das peças na estação e destino(s) dos produtos.

Quais os alimentadores para cada tipo de peça? Essa informação pode ser representada através do conceito "origem" que estabelece a relação entre a peça e um componente da estação (alimentador) (fig.3.1.24).

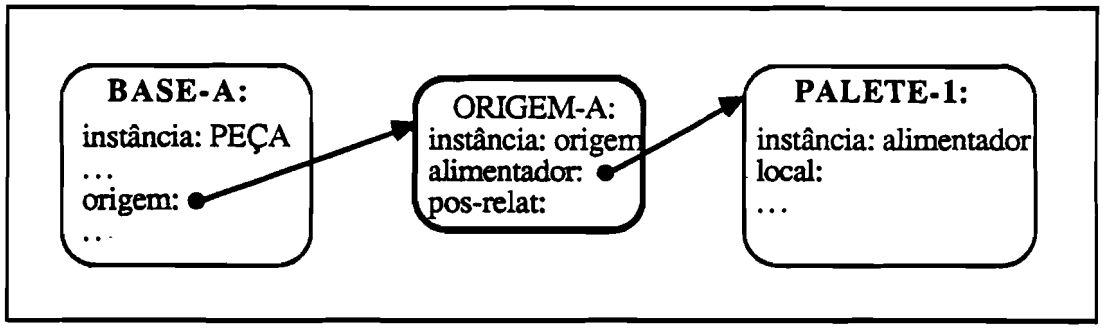


Fig. 3.1.24 Relação entre componente do produto e componente de estação

No caso de a localização da peça ser rigorosamente conhecida à partida (alimentador "bem estruturado"), o *slot* pos-relat é preenchido. No caso dum alimentador menos estruturado (mais flexível) como é o caso dum tapete rolante, o valor *nil* é especificado. Neste último caso, a localização exacta terá de ser determinada em tempo de execução (através do subsistema percepcional).

Porém, pelo facto de se conhecer qual o alimentador, pode-se fazer uma estimativa grosseira o que permite gerar à partida esqueletos de trajectórias para atingir a peça mesmo sem conhecimento completo da posição final.

De forma similar se pode estabelecer um conceito "destino" ligando o produto final (montagem) com, por exemplo, um tapete rolante ou qualquer outro dispositivo de "saída".

De notar que o processador de referenciais desempenha aqui um papel importante no suporte a estas relações.

3.1.4 - PLANO DOCUMENTADO

O principal objectivo da programação genérica é conseguir um programa / plano onde a tarefa seja descrita a um nível em que seja realizável pelos agentes executores.

Estrutura multinível

Em geral, uma tarefa pode ser descrita a diversos níveis de abstracção (ver NOAH [Sac77] e Molgen [Ste81]). Os operadores abstractos dum dado nível representam

objectivos ou tarefas para o nível abaixo. Por outras palavras, cada operador abstracto dum dado nível é expandido numa sequência de vários operadores do nível abaixo. Na base tem-se uma descrição em termos dos operadores concretos (executáveis) da estação.

Se as diversas representações da descrição da tarefa - diversas representações do plano - forem mantidas, é possível conceber um sistema de supervisão de execução e monitoração multinível.

Na experimentação realizada mantiveram-se dois níveis de representação (fig. 3.1.25):

-plano de processo que, como se descreveu, é o resultado da fase preliminar à programação genérica;

-plano genérico, a nível de acções executáveis pela estação.

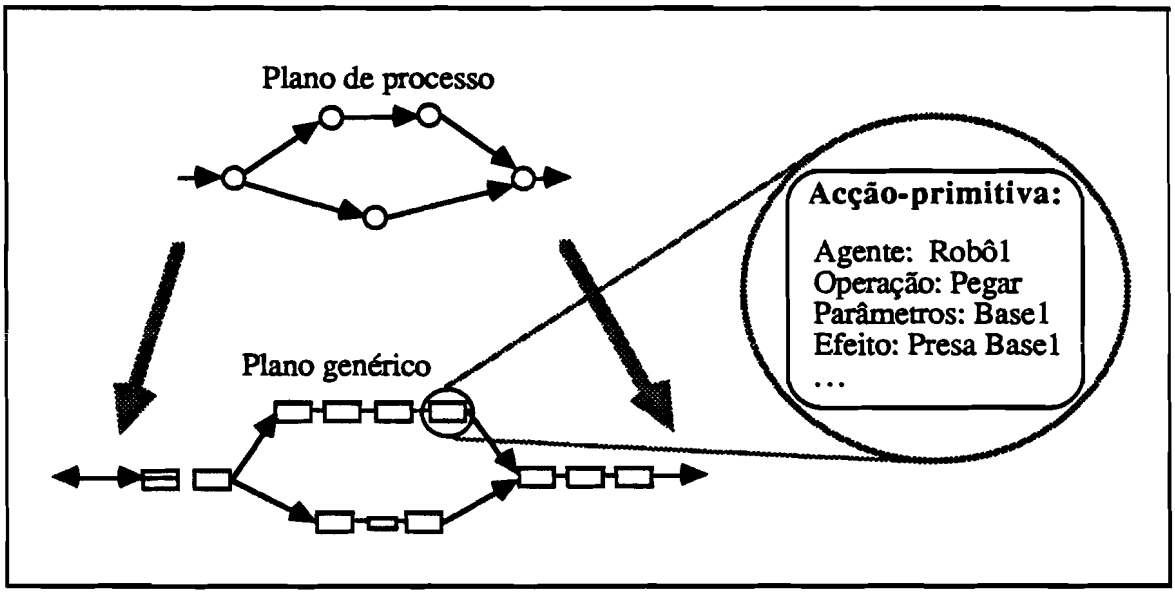


Fig. 3.1.25 Múltiplas representações do plano

A actividade de programação genérica limita-se a transformar a descrição de nível plano de processo numa descrição em termos de "acções primitivas", com recurso aos vários componentes de informação descritas e a apoio externo (utilizador). Contudo, o plano de processo não é usado somente como *input* da programação; ele é mantido para a fase de execução para suportar a monitoração multinível, como se verá adiante.

Em termos de representação, cada um dos níveis pode ser suportado por um grafo dirigido (parcialmente ordenado, reflectindo o paralelismo possível).

Acções primitivas

O plano genérico é, de facto, o resultado duma síntese de informação, obtida em diversos elementos do SI, em que cada nó (acção primitiva) é representável por um frame com os slots:

•Agente - indica qual o destinatário da acção. Ex.: Robô1.

•Operação - operador a realizar pelo agente. Ex.: Pegar.

•Parâmetros - argumentos para o operador indicado. Ex.: Base-1.

•Efeito - informação necessária para a tarefa de monitoração, indica as expectativas após a execução da acção, ou por outras palavras, de entre os possíveis efeitos da acção, quais os que contribuem para a tarefa em curso. Ex.: Presa Base-1.

É, porém, de notar que o facto de se introduzir aqui esta informação não significa que ela seja forçosamente monitorada. A monitoração do cumprimento duma dada expectativa depende das capacidades de observação (sensorial, por ex.) da estação.

Esta inclusão de informação sobre a tarefa no plano deu origem ao termo "plano (genérico) documentado".

Obviamente que estes aspectos podem / devem ser considerados nos diversos níveis de abstracção se se pretender monitoração multinível.

•Prox-op e Prev-op servem para estabelecer os arcos do grafo.

Notar que, como valores destes *slots*, podem ser indicados nomes de outros objectos do SI e, portanto, se pressupõe uma grande integração entre a representação do plano e o restante sistema.

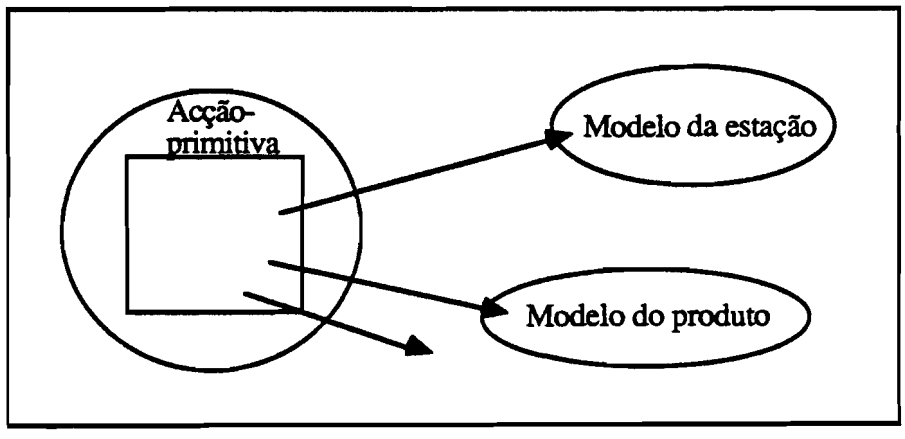


Fig.3.1.26 A representação do plano é integrada no SI

Os operadores usados a nível do plano genérico incluem as acções tipicamente contempladas nos actuais controladores de componentes. Exemplo:

robô:	percepção:
-mover <trajectória>	-identificar-obj <obj>
-posição-de descanso	-localizar-obj <obj> <ref-loc>
-trocar-ferramenta <ferram>	...
...	
garra:	transportador:
-pegar <peça>	-mover-p-frente <npassos>
-posicionar <peça>	-mover-p-trás <npassos>
...	...

Tab. 3.1.2 Operações primitivas

Esta parece-nos uma opção razoável pois corresponde ao estado da arte industrial em termos de linguagens de nível manipulador como VAL II ou LM. Um esforço de padronização deste nível de linguagem (somente para o robô) tem sido desenvolvido no âmbito do projecto Esprit 623, de que resultou a proposta ESL [JakNas88].

No caso da percepção pressupõe-se um subsistema capaz de identificar e localizar objectos na cena com base em informação sensorial. Para casos bem delimitados já existem alguns destes sistemas - baseados em visão 2D, por exemplo, assumindo objectos não sobrepostos e uma família limitada de objectos possíveis. Uma implementação dum tal sistema foi realizada pela Linha de Sistemas Sensoriais do Grupo de Robótica da UNL [SteSanQue87].

É, porém, de notar que as operações indicadas para as ferramentas (garra, no exemplo) não são exactamente as fornecidas habitualmente pelas linguagens de nível manipulador (*open, close*). De facto, acções como pegar e posicionar ainda são operações abstractas no sentido em que são decomponíveis numa sequência de operadores do nível aceite pelo controlador do robô.

A opção tomada resulta do facto de este tipo de operações implicarem normalmente movimentos finos, recurso a forte realimentação de informação sensorial e, portanto, cairem na alçada dum planeamento especializado, local aos agentes, durante a execução.

De notar também que alguma informação necessária para detalhar estas operações já foi providenciada anteriormente: referenciais de aproximação, preensão, afastamento, etc.. Outra terá de ser adquirida em tempo de execução.

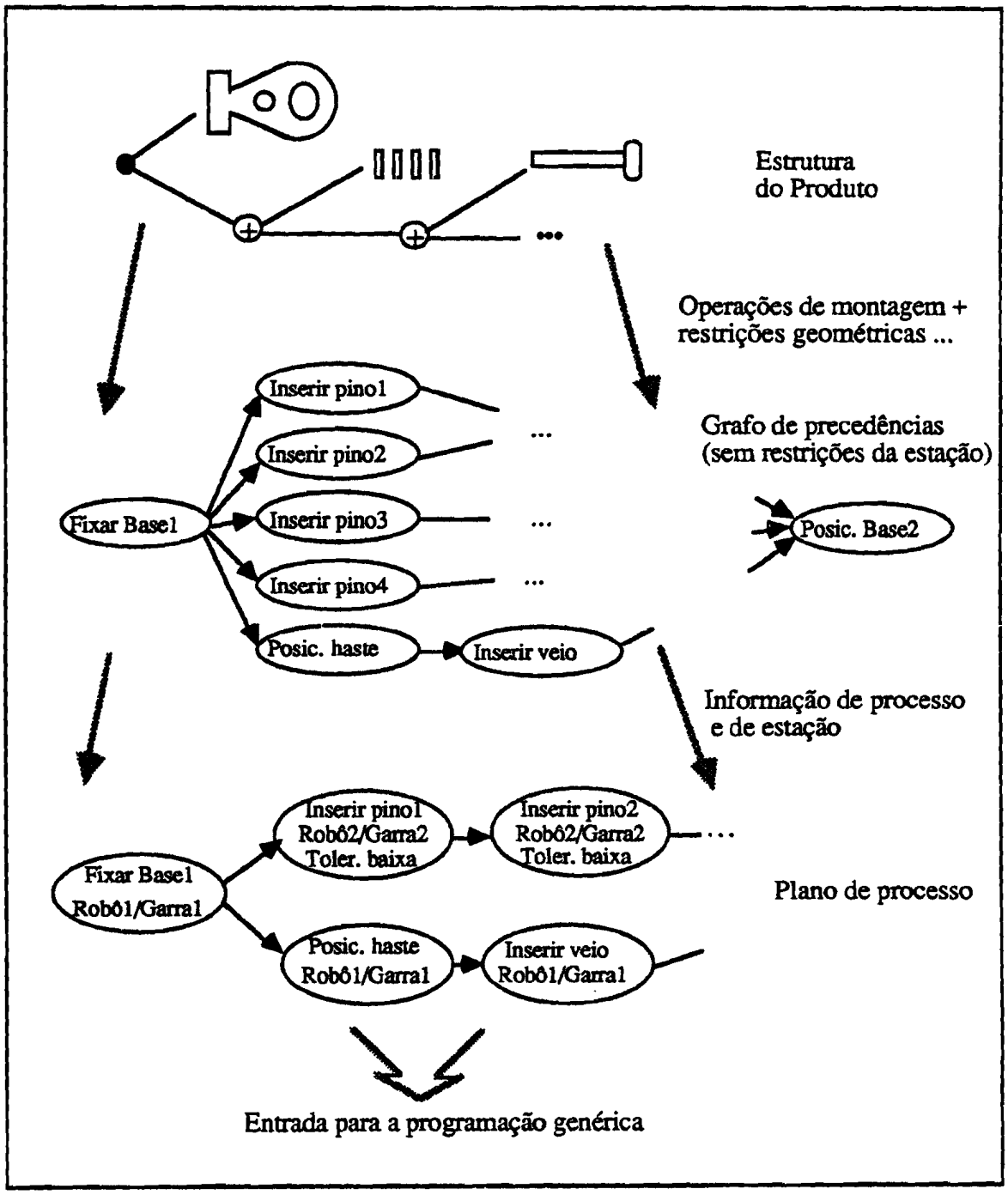


Fig. 3.1.27 Diferentes níveis de especificação da tarefa

Para além dos dois níveis de representação que temos vindo a considerar, outros poderiam ser explicitados.

Na fase de planeamento de sistema, que considerámos fora do contexto deste trabalho, podem-se ter diferentes níveis de representação do plano / descrição da tarefa, correspondentes a diferentes níveis de informação associada. É o que se ilustra na fig.3.1.27.

De notar que esta é apenas uma das possibilidades. A ideia geral é a de que em cada etapa do processo de planeamento se vai adicionar um novo nível de informação, correspondendo-lhe um novo nível de descrição da tarefa. A nossa perspectiva é de que a automatização desta área de actividade passa também por uma aproximação integrada em torno do conceito de sistema de informação, como temos vindo a defender para o sistema de programação. Essa aproximação teria a vantagem adicional de facilitar a "transferência" de informação entre as duas fases reduzindo as "perdas" de informação referidas no capítulo 2.2.

Também, como foi referido, o plano genérico pode ainda ser sujeito a um maior nível de detalhe. É o caso por exemplo, dum operador "pegar" que se pode decompor em:

-abertura da garra, de acordo com dim-pr;

-movimento fino desde o referencial de aproximação até ao referencial de preensão;

-fecho de garra, controlada por dim-pr e informação de sensor de força;

-movimento fino até referencial de afastamento.

Esta expansão será da responsabilidade dum planeador especializado a nível de agente.

A opção por uma representação do plano a múltiplos níveis tem a sua consequência a nível da arquitectura de supervisão e monitoração que também será multinível, como se verá adiante.

3.1.5 - GERAÇÃO DE PLANOS

Breve historial

Os trabalhos de geração automática de planos, na formulação mais ampla de resolvedor genérico de problemas, remontam aos anos 60 (GPS - Newell e Simon 1960; Green 1969).

Durante a década de 70 surgiram vários planeadores que permitiram resolver algumas questões básicas mas para universos simplificados e, portanto, ainda longe duma aplicação generalizada ao mundo real.

Uma possível classificação dos planeadores desta primeira geração, proposta em [CohFei82], distingue quatro aproximações:

i• Não hierárquicos

-Planeadores que apresentam uma única representação para um plano.

-Não distinguem entre as acções que são críticas para o sucesso do plano e as que são simples detalhes.

-Exemplos: STRIPS [Nil80], WarPlan [War74], HACKER [Sus75], INTERPLAN.

ii• Hierárquicos

-Geram uma hierarquia de representações dum plano, em que o nível mais elevado é uma simplificação ou abstracção do plano e o mais baixo constitui o plano detalhado (comparar com programação por refinamentos sucessivos).

-Exemplos: GPS, ABSTRIPS, NOAH [Sac77], MOLGEN [Ste81].

É de notar que ambos os tipos de planeadores geram planos com uma estratégia hierárquica de subobjectivos. É o que acontece, por exemplo, no estabelecimento das pré-condições, visto que as acções têm pré-condições que constituem subproblemas a resolver, os quais, por sua vez, terão as suas pré-condições, e assim sucessivamente. Mas somente os planeadores hierárquicos utilizam uma hierarquia de representações do plano.

iii• Baseados em esqueletos

-Selecciona-se um esqueleto de plano, a partir duma base de esqueletos, que seja aplicável ao problema em causa e, depois, os passos abstractos desse plano são preenchidos (ou instanciados) com operadores do contexto do problema particular.

-Em caso de falha, tenta gerar um novo plano para a situação pretendida. Alguns sistemas (HACKER, por exemplo) tentam generalizar os novos planos e adicioná-los à base de esqueletos (uma forma de aprendizagem !).

-Exemplos: Extended STRIPS / PLANEX [SteHarNil81], HACKER [Sus75].

iv• Com base nas oportunidades

-Usam uma estrutura de controle de "blackboard" ou agenda onde especialistas de planeamento afixam sugestões.

-Cada especialista foi concebido para tomar um tipo particular de decisões. Estes especialistas não funcionam segundo uma ordem pré-estabelecida mas sim quando houver razão ou oportunidade para tal - as decisões de planeamento são tomadas assincronamente.

-Parcelas do plano podem, assim, estar em desenvolvimento independentemente.

-Exemplo: trabalhos de Hayes-Roth [HayHay...79].

Um dos principais alvos da atenção dos primeiros planeadores foi a questão da interacção (negativa) entre objectivos que surge quando o problema a resolver é composto por uma conjunção de subproblemas: a resolução de um subproblema pode destruir a solução derivada para outro. A ênfase colocada na resolução deste problema teve a sua origem nos exemplos do mundo dos blocos e outros universos artificiais que, embora introduzindo problemas simplificados, desviaram a atenção de questões mais importantes do mundo real.

Alguns desses outros problemas que, em certos casos, já tinham tido algumas contribuições nos primeiros planeadores, começaram a constituir linhas de trabalho mais intensas, embora ainda de forma fragmentada, no início dos anos 80. Dentre essas questões podem-se destacar, pela importância que assumem para a Robótica:

-relação do planeamento com a supervisão de execução

-replaneamento ou reparação de erros

-concorrência e interacção entre especialistas

-planeamento interactivo

-raciocínio sobre recursos

-planeamento para aquisição de informação sensorial

-planeamento com conhecimento impreciso

-raciocínio sobre o tempo.

Uma extensa apresentação do estado da arte no domínio da geração automática de planos, com especial incidência na sua aplicação à Robótica, a par de algumas perspectivas pessoais sobre os tópicos indicados, pode ser encontrada em [Cam87a], um trabalho preparatório realizado numa fase inicial desta investigação.

Em termos da Robótica, surgiram algumas propostas de linguagens de nível tarefa, que têm em si implícita a necessidade de geração automática de planos como o LAMA e o AUTOPASS [ver Loz83]. Estas propostas surgiram de forma apenas conceptual e não deram origem a nenhuma implementação que não fosse muito parcial.

Alguns avanços importantes começaram, porém, a surgir a nível de planeadores especializados: geração de trajectórias livres de colisões, encaixe de peças, etc..

Na segunda metade da década de 80, e com base em todos os desenvolvimentos anteriores bem como em avanços nos métodos de representação de conhecimento, começaram a surgir tentativas - ainda limitadas - de utilização de geradores de planos em sistemas robóticos reais. Isto pode ser exemplificado pelos trabalhos de [Zri86] baseados em Prolog, ou pelos projectos em curso na FIAR [Rod87] [GalPezRod88] e na Universidade de Karlsruhe [Spu87] utilizando OPS5.

A aproximação seguida nesta proposta baseia-se, como referido no cap. 2, na utilização duma estrutura de planeamento hierárquico através da combinação dum planeador genérico (interactivo) com planeadores especializados (especialistas). A abordagem é feita na perspectiva duma integração em torno do Sistema de Informação.

A responsabilidade do planeador genérico no contexto definido pela informação de partida e plano documentado, pode sintetizar-se em: expansão de operadores, colecta de informação e optimização. Um protótipo integrado no suporte de informação definido atrás foi desenvolvido em CRL-Prolog (componente Prolog do Knowledge Craft) [Cam88b].

Expansão de operadores

Neste protótipo, cada operador abstracto do plano de processo é expandido num conjunto de acções primitivas. Um conjunto de regras Prolog permite dirigir esta expansão para cada tipo de operador abstracto. Por exemplo (notação CRL-Prolog):

```
(inserção-de-pino ?Pino1 ?Peça2) < (obter ?Pino1)
                                   (aproximação-montagem ?Peça2  ?ref-apr21)
                                   (mover-para ?ref-apr21) (inserir-pino ?Pino1)
```

A interpretação desta regra é praticamente linear:

Planear a inserção do pino ?Pino1 na peça ?Peça2 implica:

-Planear a aquisição, pelo robô, do pino ?Pino1

-Planear o movimento para o referencial de aproximação à peça ?Peça2 (informação obtida do operador abstracto que está correntemente sendo expandido).

-Planear operação fina de inserção.

Utilizando a notação de Edinburgh ter-se-ia:

```
inserção-de-pino (Pino1, Peça2) :- obter(Pino1),
                                   aproximação-montagem(Peça2, Ref-apr21),
                                   mover-para(Ref-apr21),
                                   inserir-pino(Pino1).
```

Outro exemplo:

```
(obter ?P) < (localizar-objecto ?P) (obter-ferramenta)
             (aproximação-preensão ?P ?ref-apr)
             (mover-para ?ref-apr) (pegar ?P)

(localizar-objecto ?P) < (origem-determinada ?P) (...)
(localizar-objecto ?P) < (localizar-por-visão ?P)
```

Isto é, obter uma peça implica:

-Localizar a peça, com eventual recurso ao sistema percepcional se o alimentador não for estruturado (consulta à relação origem)

-Eventual mudança de ferramenta, se a corrente não coincidir com a especificada para o operador abstracto em expansão

-Obter referencial de aproximação para preensão da peça

-Mover para referencial de aproximação da peça a pegar

-Pegar na peça.

Colecta de informação

Como actividade complementar à expansão de operadores, o planeador genérico tem de proceder a uma síntese - colecta e transformação - de informação proveniente de diferentes componentes do SI para preencher os detalhes de cada acção primitiva.

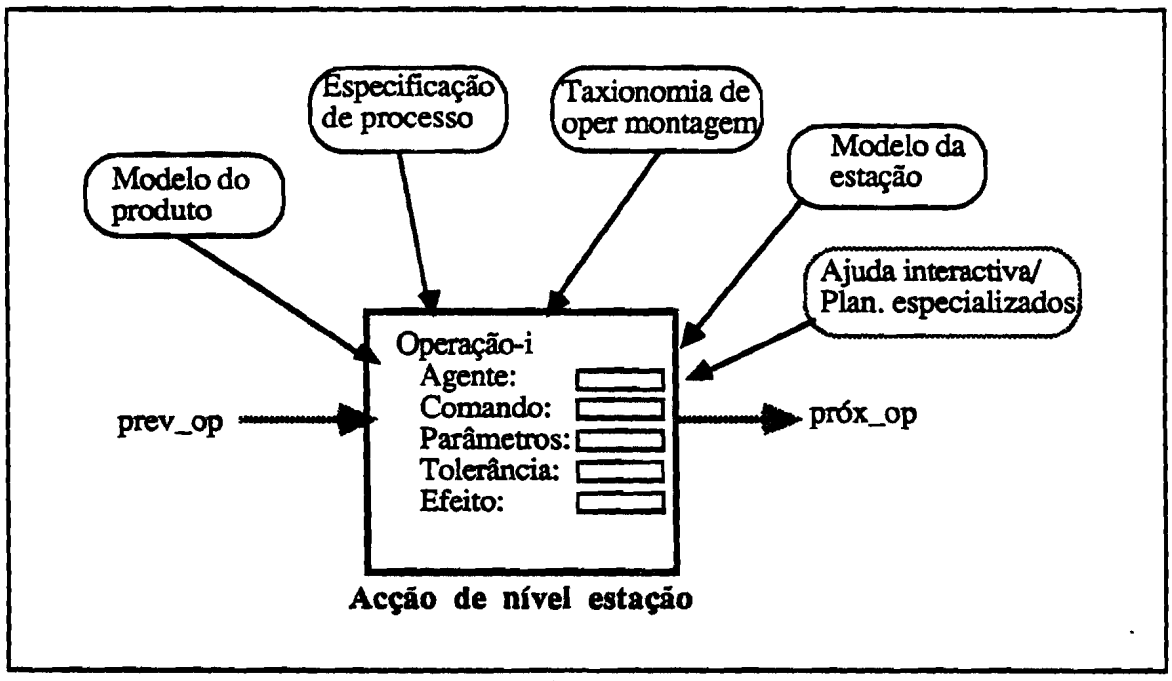


Fig. 3.1.28 Síntese de informação na geração de planos

Em relação a esta questão é de realçar as vantagens da integração entre Prolog e o sistema de frames CRL que o Knowledge Craft faculta, o que simplifica a tarefa. Toda a estrutura de frames declarada no CRL fica automaticamente visível para o Prolog. O seu acesso é garantido por um conjunto de predicados Prolog:

:schema, :slot, :value, :exists, :new, etc. (ANEXO B).

Exemplo: Regra que verifica se a localização de uma peça ?P é conhecida à partida ou não (ver fig. 3.1.29).

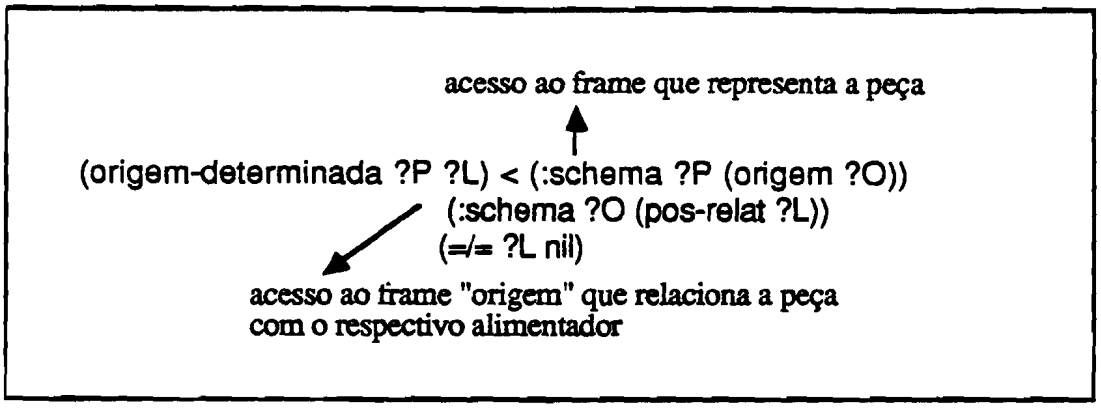


Fig. 3.1.29 Exemplo de integração Prolog - CRL

A propósito deste exemplo, convém lembrar que há informação que não está disponível na fase de planeamento mas somente durante a execução, como pode ser o caso da localização duma peça.

Como resolver esta questão?

É de notar que nos *frames* "acção primitiva" (gerados pelo planeador) não se tem efectivamente explicitada toda a informação a usar na execução, mas antes referências para outros componentes do SI (agentes, peças, ...) de onde os dados serão finalmente colectados na fase de execução.

Assim, por exemplo, em relação à localização duma peça, como não se colocam os valores absolutos dos referenciais no *frame* acção-primitiva mas antes referências para os respectiva representação, em tempo de execução - e após uma acção de localização - o valor efectivo do referencial será obtido, pelo funcionamento do processador de referenciais, a partir do ref-loc da peça (que, na altura, já deverá ser conhecido), fig. 3.1.30.

Neste exemplo, assume-se que o supervisor de execução executou previamente uma operação de localização de BASE-1, acção essa que deverá estar indicada no plano.

Noutra forma de estruturação podia ter-se um demónio *if-read* associado ao *slot* "localização" do objecto que activasse implicitamente a função de localização no agente percepcional.

Como se depreende, está-se pressupondo um sistema executor "bem informado", isto é, com apoio do Sistema de Informação - faceta modelo dinâmico do mundo. Ou seja, pressupõe-se um acesso aos modelos durante a execução (embora a apenas alguns atributos - visão parcial da informação).

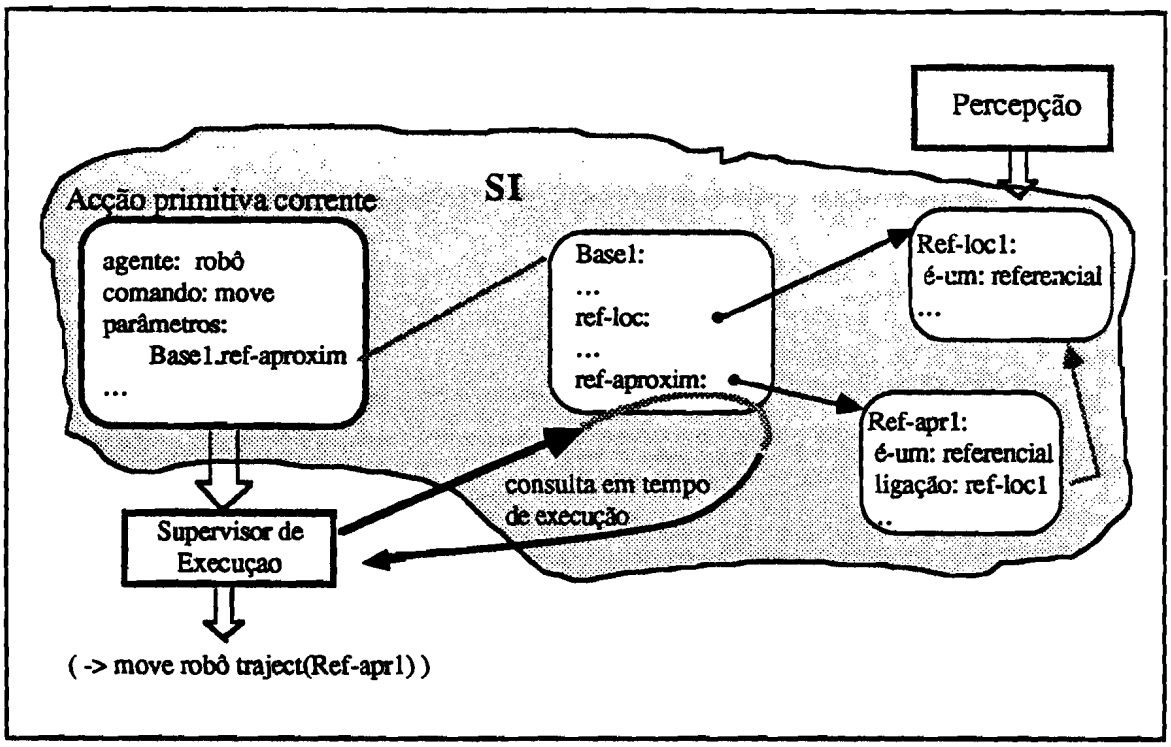


Fig. 3.1.30 Exemplo de acesso, em tempo de execução, a informação eventualmente não disponível durante o planeamento.

A actividade de síntese de informação pode ser totalmente realizada pelo planeador genérico ou por recurso a especialistas quando envolver raciocínios mais elaborados.

Neste último caso, o planeador solicita, de forma interactiva, um serviço aos especialistas para detalhar alguma parte do plano (caso da determinação de trajectórias, por exemplo), fig. 3.1.31.

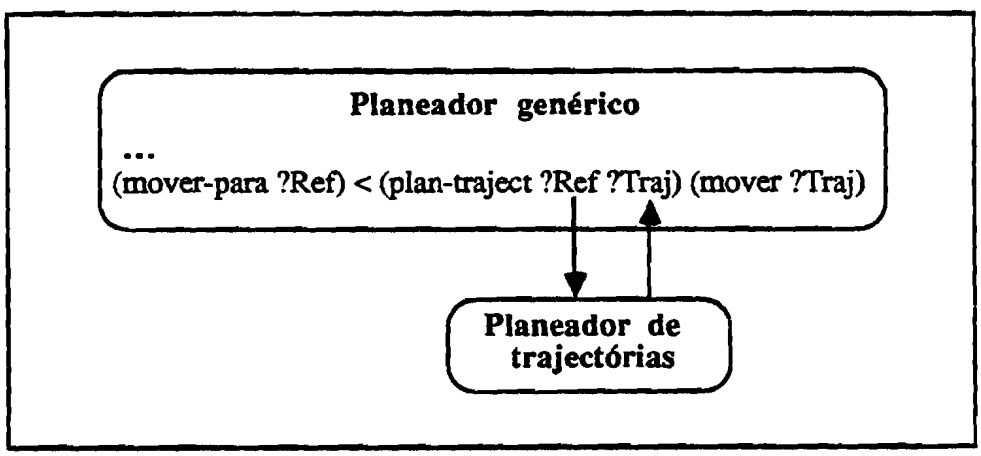


Fig. 3.1.31 Exemplo de interacção entre planeador genérico e especialista

Um dos especialistas a que o planeador pode recorrer é ao humano (planeamento interactivo).

No próximo capítulo analisar-se-ão estas questão mais em detalhe.

Optimização

A expansão de operadores não tem que ser "cega". Alguma optimização pode ser conseguida em função do estado resultante de acções que precederam a fase corrente do plano.

Então, em paralelo com o processo de planeamento, uma simulação de execução é feita de forma a manter actualizado o conhecimento sobre o estado corrente (expectativa) da execução. Com base neste estado corrente podem-se seleccionar diferentes regras de expansão e, assim, evitar uma expansão cega.

Exemplo:

(obter-ferramenta) < (...)
(:schema ?operador-corrente (ferramenta ?ferr))
(:schema robô1 (ferramenta-corrente ?ferr-corr))
(teste-e-troca ?tool ?curr-tool)

(teste-e-troca ?ferr ?ferr) < !
(teste-e-troca ?ferr ?ferr-corr) < (trocar-ferramenta ?ferr)

Só é gerada uma acção de mudança de ferramenta se a acção corrente requerer uma ferramenta diferente da que fora usada na fase anterior. O conhecimento da ferramenta usada na fase anterior é representado pela relação "ferramenta-corrente" entre o robô e essa ferramenta (ver fig. 3.1.21). A simulação duma acção *trocar-ferramenta* durante o planeamento deve manter (ligar / desligar no contexto de planeamento) estas relações.

Discussão de problemas adicionais

De notar que regras como as exemplificadas incorporam em si conhecimento sobre a estação concreta. Neste sentido, este planeador, entendido como um conjunto de regras de expansão, é específico para cada estação. Assim, parte do modelo da estação - no que respeita aos aspectos "operativos" - fica implícito nas regras do planeador e a construção

de tais regras pode, então, ser vista como uma forma de introduzir, no sistema de programação, conhecimento sobre a estação.

Isto constitui uma solução com alguma falta de generalidade. Todavia, se se considerar que uma das direcções de flexibilidade passa pelo incremento da polivalência da estação, o resultado será que não se está "mudando de estação todos os dias". A flexibilidade será materializada, em parte, pela multiplicidade de ferramentas que podem ser usadas por cada agente ou pela generalidade das operações disponíveis nos agentes, e não pela mudança radical dos agentes envolvidos na estação. Nesta perspectiva, a solução apresentada não é tão particular. A fase de instalação duma estação incluirá também a criação das regras para o planeador (programação de sistema).

Um maior grau de independência da estação pode ser conseguido se os operadores (acções primitivas) de cada agente forem representados explicitamente e definida a sua semântica (pré-condições e efeitos, à semelhança do STRIPS [Nil80]). A fig.3.1.32 apresenta um exemplo.

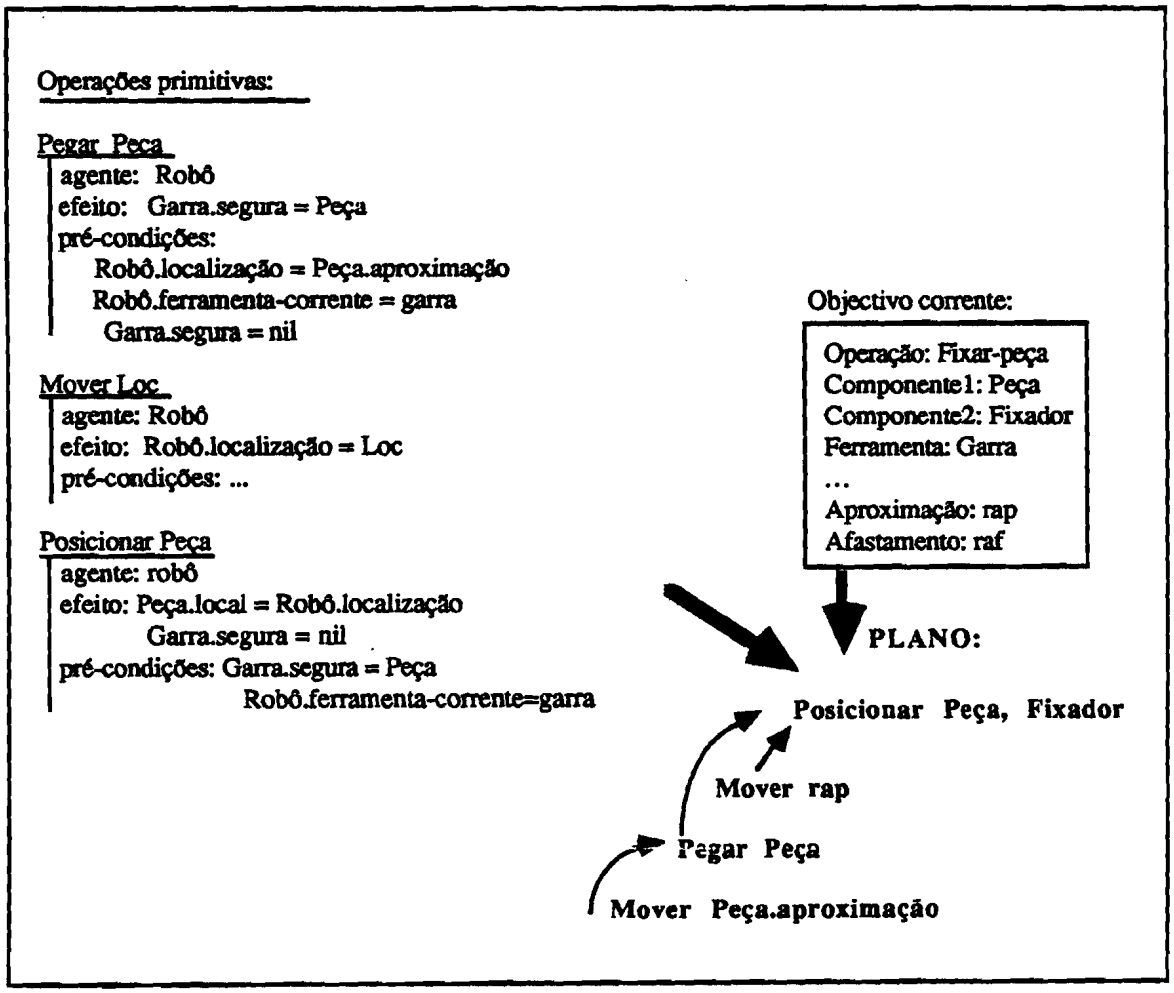


Fig.3.1.32 Outra aproximação para a modelação das operações primitivas

Um planeador como o Warplan poderia ser adaptado para resolver este tipo de problema em termos do esqueleto do plano.

A nível de cada agente ter-se-ia a lista de operações primitivas como no exemplo da fig. 3.1.23. Esta aproximação pode ser mais conveniente para um raciocínio sobre recursos quando da recuperação de erros, já que, de forma declarativa, se tem conhecimento das funções de cada agente sendo, assim, possível procurar agentes alternativos.

Em qualquer dos casos há que modelar a estação do ponto de vista operativo ou funcional e, portanto, configurar o planeador.

O objectivo nesta implementação não foi investir no estudo dos problemas básicos de planeamento mas antes analisar o problema da integração de planeadores num sistema real - identificação de dependências e interacções com outros componentes. Em particular, realçam-se as questões de integração com as diversas componentes do Sistema de Informação. A utilização de especialistas e o recurso ao utilizador (planeamento interactivo) também são importantes neste contexto.

Por outro lado, estas regras de expansão estão completamente dependentes dos operadores abstractos de montagem. Isto é, trata-se dum planeador específico para o domínio de aplicações de montagem e não um resolvedor de problemas genérico, o que parece defensável por critérios de realismo.

Na implementação realizada, o plano produzido é fundamentalmente sequencial - assumindo uma estação com um só robô.

Para estações mais complexas - com vários robôs, por exemplo - o plano pode ser constituído por sequências parcialmente ordenadas, ou seja, vários ramos que se executam em paralelo e a geração de planos nessas condições introduz alguns problemas adicionais: determinação do possível paralelismo (dependente de restrições de tarefa e das potencialidades da estação) e questões de sincronização.

Novamente se coloca a questão do nível onde este problema deve ser abordado - na programação genérica ou no planeamento de processo. Se se pretender que o planeamento de processo tenha alguma independência da estação concreta, fará sentido que, pelo menos em parte, o problema seja abordado a nível da programação genérica. Por outro

lado, o paralelismo possível está dependente do próprio processo de fabrico pelo que a sua análise não pode ser completamente independente da fase anterior à programação.

Na abordagem do NOAH (e planeadores derivados), a expansão de operadores é feita admitindo todo o potencial paralelismo, sem qualquer dependência do sistema executor. Restrições ao paralelismo vindas do próprio processo podem ser expressas nas regras de expansão de operadores abstractos (SOUP procedures). Numa segunda fase, e em consequência dum processo de crítica ao plano produzido para eliminação de situações de interacção negativa, algumas restrições adicionais são introduzidas. Finalmente, novas restrições são estabelecidas em função das capacidades do sistema executor.

Uma outra aproximação resultante duma tese de doutoramento em preparação na Universidade de Karlsruhe [Neg88], começa por fazer uma primeira expansão dos operadores abstractos num nível intermédio baseado nos componentes da estação e, numa segunda fase, realiza a expansão deste nível para os operadores primitivos, mas agora gerando sequências independentes para cada componente (fig. 3.1.33).

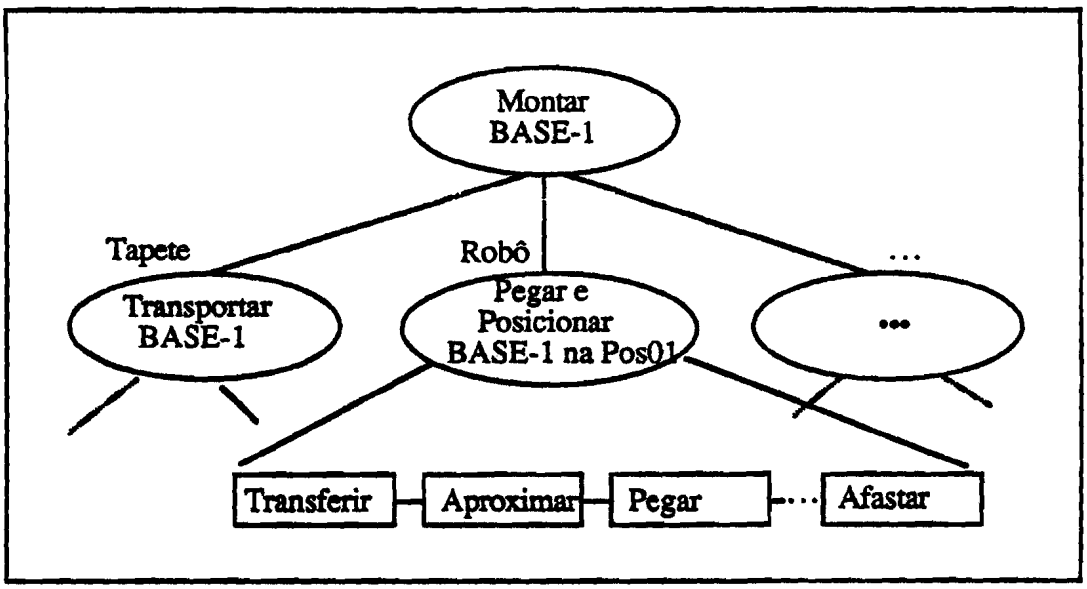


Fig. 3.1.33 Expansão de operadores abstractos, condicionada pelos componentes da estação

Neste processo de planeamento temos, desde o princípio e ao contrário do NOAH, uma dependência da estação específica.

Em relação aos aspectos de sincronização, a ênfase deste trabalho é colocada na concepção da arquitectura de controle da estação onde está sendo explorado um modelo derivado das redes de Petri.

Esta aproximação está mais próxima do modelo que seguimos, já que se está assumindo que uma indicação dos utensílios / componentes a usar para cada subtarefa é implícita no conhecimento do sistema planeador ou indicada pelo planeamento de processo. Um raciocínio semelhante às primeiras etapas do NOAH pode ser útil para a automatização do planeamento de processo.

Cremos que a solução passará por uma abordagem unificada das duas questões (planeamento de processo e genérico) com progressiva automatização de fases intermédias e desaparecimento da necessidade de programação. Ou seja, com um aumento do nível de abstracção em que se especifica a tarefa e redução de necessidade de intervenção do operador humano na passagem dessa especificação para a programação da estação. Estas questões foram, contudo, deixadas para uma segunda fase do trabalho (ver cap. 4).

3.1.6 - INTERACÇÃO COM ESPECIALISTAS

Como foi referido, o planeador genérico pode, na concepção aqui apresentada, recorrer a módulos especialistas que detalhem certos aspectos do plano que está sendo produzido.

Esta forma de conceber o problema propicia uma maior estruturação, pela divisão da tarefa em subproblemas e facilita o processo de automatização gradual. Algumas das funções mais difíceis de automatizar podem ser inicialmente desempenhadas pelo especialista humano e, há medida que se avança na automatização, procede-se à sua integração, sem mudanças drásticas na estrutura geral do sistema.

Que especialistas?

Um exemplo já citado é o do planeamento de trajectórias livres de colisões. Outros exemplos, que estão muito dependentes do "grau de incursão" na área de planeamento de sistema, podem incluir:

-determinação de pontos de preensão

-raciocínio sobre recursos

-obtenção de referenciais de aproximação / afastamento, etc.

-o próprio processador de referenciais pode ser encarado como um especialista.

Adicionalmente podem-se considerar outra zona de planeamento especializado, relacionada com movimentos "finos":

-detalhe de acções de pegar, posicionar, inserir pinos, parafusos, etc.. Este planeamento necessita de informação adquirida durante a fase de execução e, por isso, será discutido na parte 3.2 (supervisão de execução).

Geração de trajectórias

Como se sintetiza em [Cam87a], a maior parte do esforço de concepção de sistemas capazes de gerar trajectórias livres de colisões tem sido feito no domínio dos robôs móveis.

Algumas das técnicas desenvolvidas incluem:

-Métodos do espaço livre - os diversos segmentos da trajectória passam pelo meio dos "corredores" entre obstáculos. Esta estratégia permite a máxima margem de segurança para o robô mas, nalguns casos, pode conduzir a percursos bastante mais longos que o necessário.

-Grafo de vértices - cada obstáculo é expandido (ou "engordado") da dimensão do robô, enquanto este é conceptualmente reduzido a um ponto. É depois construído um grafo de possíveis caminhos com arcos conectando os vértices dos objectos expandidos: se a linha entre 2 vértices não interceptar nenhum dos objectos expandidos, é um segmento candidato e adicionado ao grafo. O grafo é depois pesquisado com um algoritmo de pesquisa padrão para determinar o caminho mais curto. Esta aproximação funciona razoavelmente quando o robô tem uma forma aproximadamente circular.

-Campos de potencial - numa perspectiva figurada, os obstáculos são vistos como montanhas de altura infinita nas zonas de possível colisão mas com ravinas descendo rapidamente até zero. O chão (planície) é inclinado na direcção do destino pretendido. O robô é transformado numa pequena bola que é lançada no chão em direcção ao objectivo. Esta rodará a uma distância prudente dos obstáculos (devido às ravinas) mas não demasiado afastada. Se a bola cai num "vale" fechado é necessário recorrer a um mecanismo de retrocesso e geração de nova tentativa (vedando a via mal sucedida). É de notar que a pista com maior declive pode ser muito mais comprida que outra onde se optasse por uma pequena "subida" inicial - problema de ser uma abordagem estritamente local.

-Grelha regular - o espaço é "coberto" por uma rede de pontos, cada um conectado com os seus vizinhos, de forma a constituir um grafo. A informação guardada em cada nó indica se ele está ou não dentro do obstáculo. O grafo é pesquisado e encontrado o caminho mais curto. Se os nós não tiverem informação binária mas sim um peso indicativo da sua distância ao obstáculo, tem-se uma aproximação discreta do caso anterior.

Cada uma destas estratégias sofre de algumas limitações, pelo que se encontram diversas variantes com o objectivo de resolver alguns casos particulares ou combinações de duas ou mais aproximações.

Relativamente a manipuladores operando no contexto duma estação de montagem, por exemplo, os resultados são muito mais escassos. Os vectores que se têm procurado abordar são:

-evitar colisões

-optimizar percursos, pela redução de custos (energia, tempo).

O problema do planeamento de trajectórias livres de colisões para o manipulador e peça a ser manipulada revelou-se difícil, envolvendo elevados tempos de cálculo mesmo para problemas simples [Bro83]. Desta forma, apenas alguns algoritmos para casos particulares foram desenvolvidos [LozBro85].

Associada à questão de evitar colisões está o problema de conseguir detectá-las. Isso pode ser conseguido, em simulação, por visualização (manualmente) num sistema gráfico ou por raciocínio geométrico. Grande parte dos sistemas modeladores de sólidos têm funções que permitem calcular a intersecção de 2 (ou mais) objectos. Se os componentes da estação estiverem modelados em termos de objectos CAD, estas funções podem ser usadas para calcular interferências na estação. Algumas estratégias podem ser usadas para reduzir os elevados custos desta computação. Uma das vias consiste em utilizar aproximações (esféricas ou envolventes poliédricas simples, por exemplo) dos objectos para facilmente eliminar objectos que estejam bastante afastados e concentrar depois a análise mais detalhada em zonas críticas (ver, por exemplo, [Ren87], [Sto87]).

É também de notar que os robôs industriais não têm normalmente uma envolvente volumétrica muito bem definida ou coincidente com os modelos CAD, dada a cablagem externa associada ao manipulador. Desta forma, também não tem muito ter sentido grandes preocupações com determinação de trajectórias muito rigorosas.

Adicionalmente, é de notar que o facto de muitos produtos serem predominantemente de montagem em pilha, também simplifica o problema das trajectórias para certas aplicações.

Alguns trabalhos partem duma trajectória - previamente definida - e tentam a sua optimização em termos temporais ou energéticos e redução de esforços mecânicos sobre as partes componentes do robô, determinando a adequada distribuição ou perfil de velocidades / acelerações ao longo do percurso (por exemplo, [Ipk87], [AkeBru...87]).

Este tipo de optimizações é importante quando se pretende a produção em larga escala. Pensando, porém, que a tendência introduzida pelos sistemas flexíveis vai no sentido de produção de pequenas quantidades dum mesmo produto [NevWhi78], tais optimizações já não parecem tão relevantes.

Aproximação interactiva

Dada a panorâmica apresentada, parece bastante realista a consideração duma aproximação gráfica interactiva na fase actual. A visualização gráfica da cena 3D numa estação gráfica permitirá ao programador especificar - apontando no écran - um esqueleto de trajectória.

Este esqueleto pode, então, ser "*smoothed* " automaticamente e adicionado de informação como velocidades / acelerações, incorporando, portanto, resultados de subsistemas ou planeadores especializados referidos acima.

Desenvolvimentos no Grupo de Robótica Inteligente da UNL prosseguem esta perspectiva dentro da filosofia de sistema integrado [Cam88a].

Note-se que as linguagens de nível manipulador aceitam, como argumento das operações MOVE, trajectórias complexas. Exemplo (LM):

MOVE via <segm1>, <segm2>, <segm3>, ... to <posição-final>

Esta trajectória é formada por vários segmentos. Para cada um pode-se indicar o modo (livre, rectilíneo, circular) e a velocidade.

A geração de trajectórias está dependente do conhecimento que se tem sobre a localização dos vários objectos na cena. Em estações completamente estruturadas onde tudo é previsível à partida, a geração das trajectórias para os diversos movimentos a realizar pelo robô pode ser feita em *off-line*.

Em situações menos deterministas em que as localizações concretas dos objectos a manipular apenas são conhecidas em tempo de execução, o problema complica-se. Nessa altura não se pode usar um processo interactivo!

Todavia, no caso de estações industriais, embora desejando um progressivo aumento de flexibilidade, ainda se tem um razoável grau de estruturação e, consequentemente, de conhecimento à priori. Assim, por exemplo, embora não seja possível prever a localização exacta duma peça num tapete rolante, pode-se fazer uma estimativa aproximada uma vez que se conhece a posição do transportador na estação. Desta forma, é possível gerar, em *off-line*, trajectórias esqueleto. Em tempo de execução apenas será necessário gerar o troço final, então função da localização efectiva da peça. Mas aí não se têm normalmente problemas de colisões e o movimento pode ser bastante directo.

Outros aspectos de raciocínio geométrico

A aproximação a uma peça e determinação da melhor posição para lhe pegar constitui também uma área de planeamento especializado onde se podem encontrar alguns trabalhos [Loz81], [Lat83], [BasTor...88].

A maior parte das aproximações seguidas compõe-se de duas fases:

-Em primeiro lugar é determinado um conjunto de posições para possível preensão do objecto. Este conjunto poderá ser determinado a partir das propriedades morfológicas do objecto. Por exemplo, volumes cilíndricos ou paralelipédicos serão bons candidatos. Outra aproximação será considerar os pares de arestas, faces ou vértices como possíveis candidatos. Duas faces que não estejam em frente uma da outra não constituirão um par candidato.

-Depois, esse conjunto de posições candidatas é depurado tendo em conta eventuais restrições de acesso. Outros factores a considerar terão a ver com a estabilidade do objecto quando pegado. Para tal são, por vezes, usadas regras heurísticas.

De qualquer forma, estes planeadores são lentos e não estão genericamente acessíveis (dada a natureza quase sempre inacabada dos protótipos académicos).

A via gráfica interactiva constitui uma alternativa. Soluções semiautomáticas podem ser conseguidas combinando a via interactiva com algum raciocínio geométrico realizado pelo sistema. Por exemplo, após o operador ter indicado no écran a "zona" onde a peça deve ser pegada, o sistema pode determinar qual o posicionamento e orientação da garra. Uma solução deste tipo pode ser encontrada nalguns sistemas de programação em desenvolvimento (IPK, Renault Automation, KUKA) [Spu...87] para aplicações de soldadura por pontos - o operador indica o esqueleto da trajectória de soldadura e o sistema determina o posicionamento do canhão de soldadura para cada ponto.

Outras áreas onde se podem encontrar esforços para o desenvolvimento de planeamento especializado incluem:

-Determinação automática de referenciais / trajectórias finais de aproximação - para pegar numa peça ou montá-la na sua posição final [Spu...87] que também pode ser visto como um caso particular de planeamento de trajectórias livres de colisões.

-Determinação da ordem de montagem dos componentes a partir da estrutura do produto - estabelecimento de grafo de precedências (com base em raciocínio geométrico) e eventual serialização devido a restrições da estação (raciocínio sobre recursos), [AbeSakTsu88]. Os resultados nestas áreas ainda são bastante limitados.

A aproximação seguida nesta proposta é a de considerar que estas questões são resolvidas - interactiva ou automaticamente - na fase de planeamento de processo.

3.1.7 - SIMULAÇÃO E INTERACÇÃO GRÁFICA

Funções

Um sistema de simulação gráfica como componente dum sistema integrado de programação desempenha duas funções vitais:

-Suporte ao teste dos programas desenvolvidos em *off-line*

-Suporte ao próprio processo de programação interactiva.

A interacção entre o planeador genérico e o especialista humano, para ser "confortável", deverá ser suportada por um sistema de simulação que permita visualizar graficamente a cena (componentes da estação e peças a manipular) e actuar sobre essa cena. Um tal ambiente propicia uma espécie de "*teach pendant*" evoluído onde o utilizador pode "ensinar" certos aspectos do processo ao sistema.

Para além das vantagens habituais da operação com sistemas simulados versus sistemas reais, tem-se a faculdade de poder operar a diferentes níveis de abstracção na fase de ensino. À medida que o grau de automatização cresce, reduz-se o número de detalhes que é necessário "ensinar". Na utilização de *teach pendant* real é difícil escapar ao detalhe dado o baixo nível dos operadores aí usados.

Ainda em relação à função de teste, o sistema pode suportar também algumas tarefas dos especialistas como, por exemplo, detecção de colisões num processo de especificação interactiva de trajectórias. A verificação pode ser visual ou por computação.

Deste modo, o sistema de simulação não é apenas uma forma gráfica de *output* mas também de input permitindo a intervenção externa.

No sentido aqui descrito concebe-se esta faceta de simulação e interacção gráfica como completamente integrada com o restante sistema todavia, a implementação completa dum tal componente não foi considerada para a fase do trabalho que aqui se descreve.

Relação com programação interactiva

O sistema de simulação serve de suporte à interacção entre o planeador genérico e o especialista humano. Será, portanto, através desta via que se podem introduzir trajectórias, posições de preensão, condições de montagem (quais os pontos que têm de ser justapostos), etc.. Um adequado conjunto de funções gráficas - expressas por comandos textuais ou através dum rato - permitirá uma forma de "programação pelo exemplo".

Para esta tarefa contribuem as facilidades oferecidas pelos sistemas gráficos que suportam representações 3D e permitem facilmente a mudança do ponto de observação. No caso do PS390 (Evans & Sutherland), por exemplo, um conjunto de botões de rotação (*dials*) permite rotações e translações em qualquer eixo.

Estas facilidades de mudança do ponto de observação são também vitais para uma verificação visual das trajectórias e detecção de eventuais colisões.

A nível de sistemas comerciais, o ROBCAD [Adl86], [AdlRod87] constitui um dos exemplos mais significativos em termos de estado da arte. Não se trata apenas dum simulador pois pretende constituir um sistema completo para auxiliar a concepção e teste da estação e dos programas a executar sobre ela.

Num aspecto está na direcção que acabámos de defender pois baseia-se fortemente na interacção gráfica para a especificação do programa. Contudo, o princípio subjacente é o da programação explícita, não contemplando qualquer geração automática de planos. Por outro lado, apenas prevê um nível de abstracção o que causa dificuldades na organização dum sistema de controle a nível de estação.

Como principais módulos inclui:

-Edição de modelos geométricos (3D CSG) e cinemáticos de componentes da estação.

-Configuração (*layout*) da estação.

-Programação - descrição explícita (em *off-line*) das tarefas para cada componente da estação usando uma linguagem própria - TDL (*Task Description Language*).

-Simulação e animação - para verificação do programa e da estação:

.avaliação de espaços de trabalho e tempos de ciclo;

.detecção de violações de restrições entre equipamentos ou de interferências;

.determinação de sincronizações;

.verificação de fluxo de peças na estação;

.definição de sensores e sua funcionalidade;

.ajuda no processo de interacção com o "cliente" / utilizador para "venda" duma solução (facilitando a explicação de conceitos ao cliente) ou mesmo para a elaboração das especificações.

-Emissão de desenhos (projecções 2D) e documentação.

-Tradução e carregamento (*download*) do programa simulado para os controladores reais.

-Biblioteca de robôs e outros componentes.

Outro exemplo significativo de sistema com uma filosofia similar é o MRS [McD85?] .

Sendo, por razões de proteccionismo comercial, sistemas fechados, impedem a sua utilização como ponto de partida para novos desenvolvimentos, o que força a muitas duplicações de esforços.

Em termos de protótipos de investigação podem referir-se, como exemplos, o sistema ISR desenvolvido no LIFIA, Grenoble em torno da linguagem LM [Lau88], o ROSI da Universidade de Karlsrhue [DilHuc85] e o sistema da KUKA [WörSta87].

No caso do Grupo de Robótica da UNL, a aproximação em curso passa também pelo desenvolvimento dum sistema de simulação e interacção gráfica, mas integrado na arquitectura global proposta [Cam88a].

Na fig. 3.1.34 representa-se o suporte para o desenvolvimento em curso. Este ambiente comporta:

- um modelador de sólidos (ROMULUS ou PADL-2) - para especificação dos modelos geométricos;
- a estação gráfica da Evans & Sutherland PS 390 - visualização e interacção; e
- o Knowledge Craft - como centro de integração com os restantes módulos do sistema de programação e controle.

Uma interligação destes componentes foi realizada em Pascal.

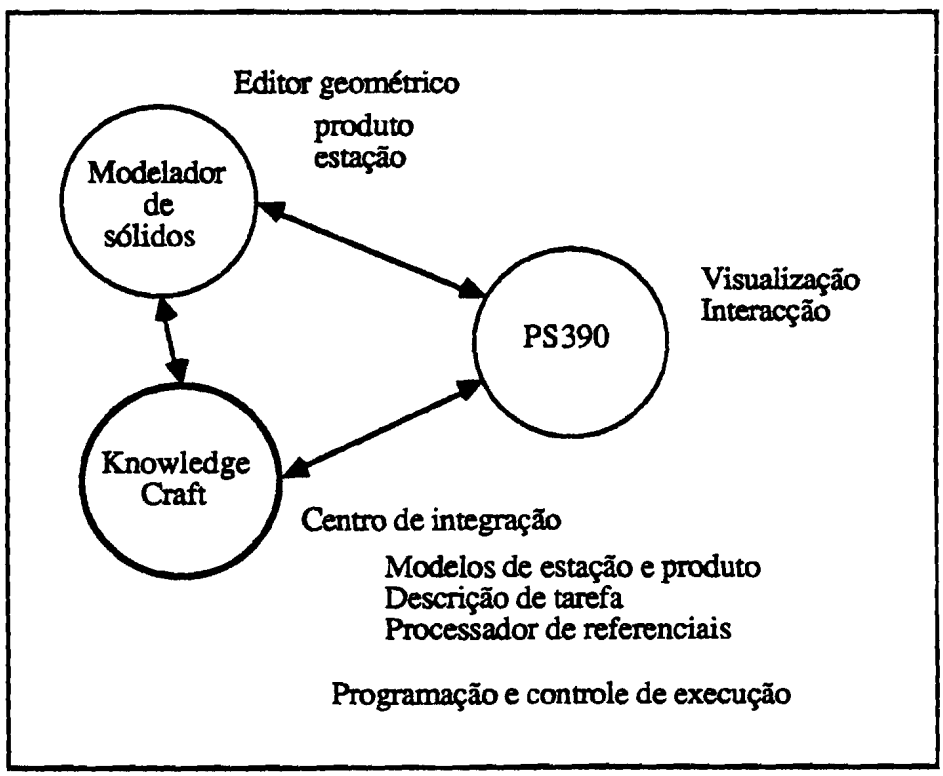


Fig.3.1.34 Suporte para o sistema de simulação e interacção gráfica

Múltiplos níveis

Numa perspectiva de progressiva automatização de funções e no seguimento duma filosofia *top down* para o desenvolvimento de programas, tem sentido conceber um sistema de simulação e interacção a múltiplos níveis de abstracção.

Por exemplo, num dado nível de abstracção poder-se-à querer apenas verificar o fluxo das peças na cena e não as suas trajectórias detalhadas. A informação manipulada a este nível - a que é possível fornecer ao simulador e a que pode ser pedida ao operador - não é informação de detalhe. Deste modo não interessa, para este nível, usar uma simulação próxima da emulação dos componentes reais, já que isso exigiria informação muito mais detalhada.

Uma perspectiva de simulação a múltiplos níveis ajusta-se a uma filosofia de programação interactiva onde se pretende ir automatizando funções de forma progressiva. O grau de simulação e interacção variará de acordo com o nível de automatização de cada função.

No nível próximo da execução real interessa dispor duma simulação capaz de emular o comportamento dos componentes / controladores reais; para níveis mais elevados será mais conveniente dispor de representações gráficas mais abstractas. Por exemplo, diagramas de barras ou medidores angulares, poderão ser mais adequados quando o objectivo é o de verificar fluxos de peças no sistema, desempenho de cada componente, etc.. O conceito de imagem activa, suportando não só uma visão actualizada do objecto a que está associada mas também o *input* assíncrono, constitui, como se verá em 3.2.6, um bom suporte.

Também uma visualização gráfica de redes de Petri poderá ser adequada para representar / verificar os aspectos globais de sincronização / escalonamento.

É, porém, de notar que algumas situações são difíceis de representar / indicar através das imagens da cena manipuladas pelo simulador - normalmente modelo de arame para ser mais rápido.Por exemplo, em certos casos será difícil decidir, por observação visual, se houve colisão ou não. O simulador pode, então, ser enriquecido se incorporar alguns módulos de raciocínio geométrico que permitam auxiliar na verificação de tais situações.

3. 2 - SISTEMA EXECUTIVO E DE MONITORAÇÃO

3.2.1- INTRODUÇÃO

Segundo a perspectiva defendida nesta proposta, deve existir uma forte relação entre a programação / planeamento e a execução visto que algumas partes do plano genérico apenas podem ser detalhadas ou concretizadas em tempo de execução quando a informação necessária fica disponível.

Por outro lado, a recuperação de excepções ou erros implica um replanear de actividades e, portanto, aplicação de princípios semelhantes aos usados durante a programação genérica, mas agora com premissas diferentes. Nesta fase têm-se menos graus de liberdade - apenas se está procurando uma solução de recurso, de carácter transitório, para uma situação de excepção enquanto que, durante a programação genérica, se tentou obter uma solução de aplicação ao caso normal.

Duma forma geral, a arquitectura de execução e monitoração inclui um supervisor (controlador de estação) que recebe o plano e distribui as respectivas acções pelos agentes que se encarregam da execução. Esta distribuição de acções despoletará também o subsistema de monitoração de execução que verificará do sucesso de cada acção e tentará uma recuperação em caso de erro. Em paralelo, poder-se-à ter um monitor de sistema que vigiará as condições de funcionamento da estação. As duas facetas de monitoração, embora distintas, estão relacionadas e uma estrutura que traduza as interdependências deve ser estabelecida.

Todo o processo de execução e monitoração deve ser consistente com a estrutura multinível do plano Algumas situações de excepção podem ser resolvidas ao nível

estritamente local das acções primitivas enquanto outras poderão ter consequências mais gerais implicando uma reformulação do plano noutro nível de abstracção.

A abordagem seguida considera uma arquitectura de execução e monitoração fortemente integrada no sistema de informação, isto é, durante a execução há acesso aos diversos modelos (produto, estação, etc.), embora segundo um "ponto de vista" diferente das fases anteriores. Para a supervisão de execução e diagnóstico de excepções, importam fundamentalmente as facetas dinâmicas - modelo dinâmico do mundo - como por exemplo, a parte operativa dos agentes, informação de estado (posições correntes, valores de sensores), etc. (fig.3.2.1).

Os aspectos dinâmicos - expectativas e reais - são acessíveis / representados por *slots* com eventuais demónios associados. A componente operativa dos agentes é fundamentalmente representada por métodos.

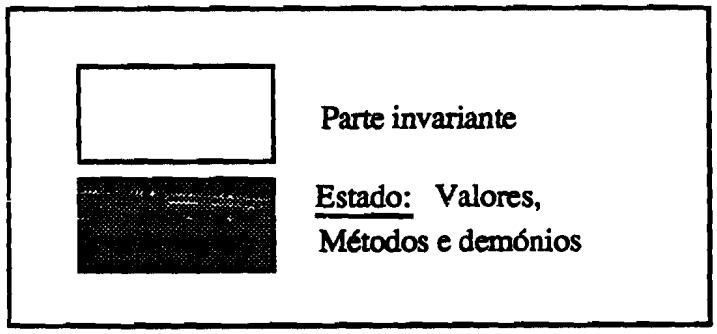


Fig. 3.2.1 Aspectos do modelo dinâmico do mundo

Será, portanto, nesta fase que se lidará com os atributos dinâmicos, se gerirão as relações dinâmicas entre componentes da estação e entre componentes da estação e peças (à semelhança do que é feito na simulação).

A nível dos agentes podem-se ter modelos especializados do mundo, de acordo com as necessidades locais.

Para a recuperação já interessa uma visão mais próxima da usada na programação genérica.

3.2.2- SUPERVISOR DE EXECUÇÃO

O supervisor de execução (controlador de estação) é basicamente um interpretador que executa o plano distribuindo e coordenando trabalho a efectuar pelos agentes (fig.3.2.2).

A interacção entre o supervisor e os agentes foi concebida com base na programação orientada por objectos, através do envio de mensagens.

Por exemplo, a mensagem

(-> 'robô1 ' mover 't1)

é dirigida ao agente robô e provocará a activação do método move com o argumento t1, que identifica uma trajectória.

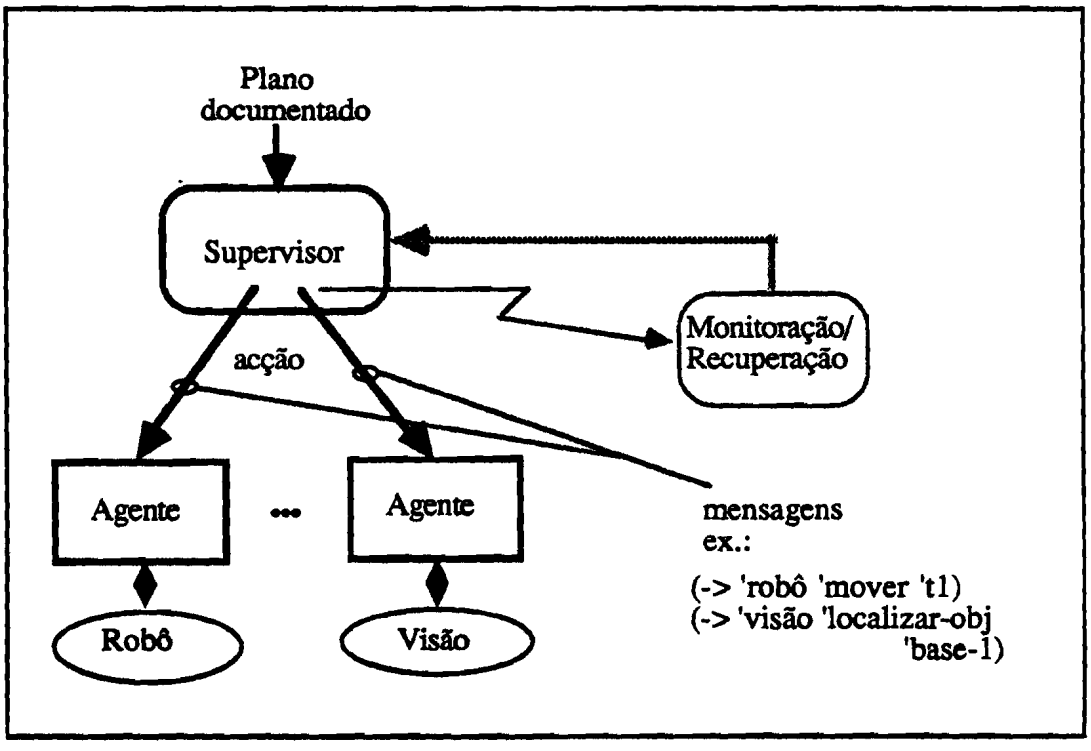


Fig. 3.2.2 Relação entre o supervisor e os agentes executores

A informação para a composição destas mensagens é extraída dos *frames* acção-primitiva do plano.

Eventuais efeitos destas acções vão, implicitamente, reflectir-se numa actualização do modelo dinâmico do mundo. Por exemplo, em consequência da mensagem:

(-> 'visão 'localizar-obj 'Base-1)

dirigida ao agente visão, o *slot* "ref-loc" da peça Base-1 será actualizado com o referencial da localização actual da peça, desde que a operação tenha sucesso.

Se uma acção subsequente tiver necessidade desta informação, já a encontrará acessível. De recordar a forma como se tratou a questão da informação não determinada à priori - o plano apenas contem referências para os *frames* onde a informação deve ser realmente colectada em tempo de execução.

Tem-se, portanto, uma forte interacção entre o sistema de execução e monitoração e o sistema de informação.

A estrutura do interpretador segue a representação multinível do plano.

No caso duma representação do plano a dois níveis (protótipo que tem vindo a ser considerado), no nível de topo seguem-se as operações abstractas (solução implícita). A execução de cada uma é conseguida pela execução da respectiva expansão (com operadores de nível estação - execução real).

No protótipo implementado, os agentes - como reflexo dos executores reais - recebem ordens de nível acção primitiva.

Poder-se-iam considerar agentes abstractos para o tratamento dos níveis abstractos do plano. Todavia, para o caso de dois níveis, isso seria uma complicação desnecessária, pelo que o nível abstracto é apenas tratado pelo supervisor.

3.2.3 - AGENTES

Introdução

As noções de objecto ou tipo de dados abstracto e representação por *frames*, combinando aspectos declarativos (*slots* e relações) e procedimentais (métodos e demónios), fornecem um bom suporte para a modelação dos componentes actuadores e percepcionais da estação robótica - os agentes.

O conjunto de agentes consubstancia, assim, a visão dinâmica do modelo da estação discutido anteriormente.

As operações que um agente é capaz de realizar (acções primitivas) são representadas como métodos e os aspectos de "estado" do agente como *slots*. Alguns destes slots podem ter demónios tipo *if-read* associados que consultarão o elemento físico sempre que forem acedidos para leitura. Uma conexão interactiva entre estes modelos e os controladores dos elementos reais (ou simulados) está aqui implícita.

O conjunto de operadores disponíveis (acções primitivas) é constituído em parte pelas acções disponíveis a nível do controlador do correspondente elemento físico (robô real, por exemplo), mas também por macro-operadores que têm de ser submetidos a um planeamento especializado durante a execução. Desta forma, a noção de agente não é meramente uma interface com os controladores dos elementos reais, mas representa uma abstracção de mais alto nível podendo incluir alguma capacidade de decisão local, conforme se viu no cap. 2.1. Associada a esta capacidade existirá a correspondente base de conhecimento: regras de planeamento ou de identificação de objectos, por exemplo.

Recuperação de controladores

Os componentes duma estação robótica apresentam, em termos dos respectivos controladores, características muito heterogéneas. Nalguns casos suportam linguagens de programação de alto nível, como é tipicamente o caso dos robôs, enquanto noutros casos se tem um controle por PLAs (*Programmable Logic Arrays*) fornecendo unicamente uma interface de mais baixo nível. Podem-se ter também vários componentes com o mesmo nível de controle mas de fabricantes diversos e, normalmente, com linguagens diferentes.

Mesmo no caso das linguagens de nível manipulador se verifica que estas não são adequadas para uma subida em nível de abstracção e, em geral, não suportam facilmente o controle de múltiplos componentes.

Contudo, estas linguagens / controladores incorporam algumas características como:

-modelos geométrico e cinemático (e eventualmente dinâmico) dos robôs;

-interpolação de trajectórias;

-interface com os dispositivos físicos (motores, sensores, etc.), sempre problemática, providenciando um moderado nível de abstracção.

Tais características serão úteis - como camada de suporte - num sistema mais avançado de programação e controle.

Está-se, de facto, procurando um controlador de estação que uniformize e racionalize a forma de acesso aos diversos controladores de componentes. O problema é, então, como integrar no novo ambiente de programação as funcionalidades já presentes nestes controladores?

A aproximação defendida [CamBar87] consiste em estabelecer uma conexão interactiva - através de métodos e demónios - entre tais controladores e o modelo da estação no SI. Pretende-se deste modo ultrapassar as naturais limitações das linguagens dos controladores existentes, que são o seu carácter fechado, pouca flexibilidade e,

eventualmente, a diversidade de linguagens para os diferentes componentes da estação, tentando simultaneamente recuperar as suas vantagens.

Esta abordagem permite recuperar trabalho já realizado ao nível de cada componente da estação - instalação ou desenvolvimento de linguagens próprias - e desenvolver sobre essa base um sistema mais completo de programação da estação, que possibilite a construção de um sistema de controlo integrado dos componentes da estação.

Apresentam-se em seguida alguns exemplos nesta direcção.

Conexão com LM

Um dos trabalhos experimentais realizados consistiu na integração dos aspectos funcionais dum robô controlado por um sistema LM no ambiente de desenvolvimento [CamBar87], [CamCor88].

A linguagem e sua instalação

A linguagem LM [Itm84] foi desenvolvida na Universidade de Grenoble e posteriormente comercializada por uma empresa da especialidade, representando um caso típico das linguagens de nível manipulador.

Como principais características podem referir-se:

-Linguagem de nível manipulador, inspirada no Pascal em termos de mecanismos de controle.

-Implementa os tipos referencial e transformação (relação entre dois referenciais).

-Especificação de movimentos: uma só operação - *move* - mas permitindo indicar vários tipos de movimentos e trajectórias. Um movimento pode ser definido por um conjunto de segmentos. Para cada segmento pode-se indicar a velocidade e optar por trajectória livre, cartesiana (segundo um linha recta), circular, etc.

-Controle de utensílios: operações específicas para a garra (*widen, open, close*) e uma operação genérica para outros periféricos (*operate tool*).

-Funções sensoriais:

.vision - devolve um referencial associado a um objecto reconhecido pelo sistema de visão (caso exista na instalação).

.fx, fy, fz, mx, my, mz - devolvem as componentes, em relação aos eixos dum dado referencial (normalmente situado no punho do robô), da força exercida pelo robô (caso este disponha dum sensor adequado).

-Paralelismo e sincronização: É possível especificar acções a serem realizadas concorrentemente por vários robôs e/ou utensílios, através das opções *nowait* e

immediatly ; a instrução *wait* permite a sincronização. Outro aspecto do paralelismo - entre a execução duma acção e a monitoração de sensores - é implícito nos "comandos guardados" onde se podem especificar condições de paragem / interrupção (via opção *until*).

No trabalho realizado partiu-se duma versão portável do sistema de suporte ao LM. Nesta versão, desenvolvida em Fortran, um programa LM é compilado para uma linguagem intermédia LME e depois interpretado por uma máquina virtual (interpretador de baixo nível). A máquina virtual é formada por duas partes: uma independente da instalação concreta e, portanto, portável; a outra deve ser adaptada para cada robô (fig. 3.2.3).

De notar que uma das propostas para a resolução do problema da multiplicidade de linguagens de robôs consiste na padronização dum nível intermédio - código IRDATA, sem grande aceitação até ao momento - que deveria ser comum a todos os robôs e para o qual cada fabricante construiria o respectivo interpretador [Rem86].

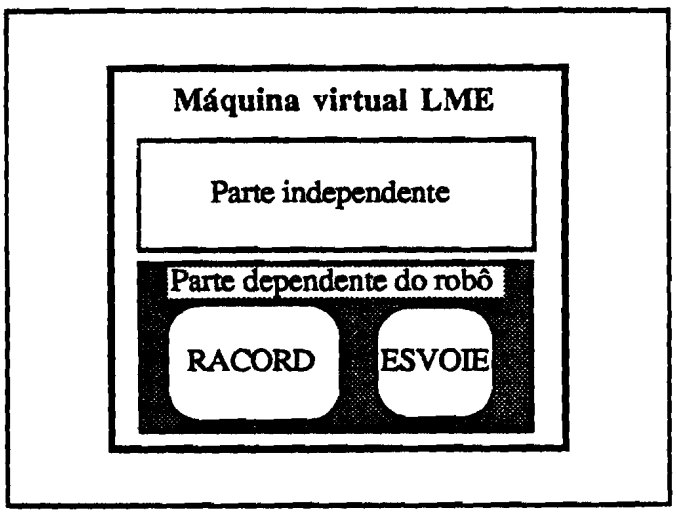


Fig. 3.2.3 Arquitectura da máquina virtual LM

A parte dependente do robô tem de ser escrita para cada tipo de instalação e condicionará o conjunto de funcionalidades disponíveis.

Esta parte é, por sua vez, organizada em vários módulos de que se destacam:

- RACORD - onde se incluem as primitivas necessárias ao controle dos movimentos do robô e operação dos seus utensílios; inclui a implementação do modelo geométrico / cinemático do robô

(metodologia de Denavit-Hartenberg) e, portanto, a resolução do problema das transformações de coordenadas (directa e inversa);

•ESVOIE - constituído pelas primitivas de comunicação com o exterior, incluindo a interface com sensores.

Duas instalações, para os robôs Renault/Sirtés e Rhino, foram realizadas [BarRosCamNev86], tendo por suporte computacional um Sperry PC/IT.

A linguagem LM admite o controle de vários robôs. Quanto ao controle de outros elementos actuadores da estação terá de ser feito pela "redução" do modelo desses elementos a robôs simplificados ou pela sua consideração como ferramentas do robô. Neste último caso, o seu comando é representado pela instrução *operate tool.*

Nas instalações efectuadas foram considerados dois componentes:

-uma mesa rotativa Rhino,

-um tapete rolante (desenvolvido na UNL),

seguindo-se a filosofia de os considerar como ferramentas do robô.

Adicionalmente, e com o objectivo de facilitar a "aquisição" de referenciais significativos na cena real, foi desenvolvida uma rotina (MANUAL) que permite o controle directo do robô através do teclado do computador, simulando o teclado de funções. Esta rotina é invocada pelo RACORD sempre que é executada a instrução MANUAL do LM. A rotina permite o controle de movimentos do robô no espaço cartesiano ou no espaço de articulações. A posição corrente do robô é memorizada por uma variável de estado quando o controle regressa ao programa LM.

Interpretador

Na perspectiva de integração dum tal controlador no sistema de desenvolvimento, não interessam os aspectos de controle de fluxo de programa fornecidos pelo LM mas apenas a parte correspondente aos operadores sobre o robô (e utensílios). Com os paradigmas de programação usados no ambiente de desenvolvimento têm-se mecanismos de controle mais poderosos que os convencionais mecanismos sequenciais, selectivos e repetitivos do LM.

Por outro lado, a conexão deve ser interactiva de forma a que o sistema supervisor possa ter um controle "contínuo" sobre a execução. Por outras palavras, pretende-se uma situação onde uma realimentação de informação acção a acção seja possível e um replaneamento / alteração de comando possa ter lugar.

Algumas outras aproximações para a "recuperação" de controladores usam uma solução de "compilação-carregamento", onde o programa é produzido e simulado num sistema de desenvolvimento *off-line* e depois compilado e carregado no controlador do

robô [AdlRod87], [Spu...85], [SpuFurKir87]. Trata-se duma solução unidireccional - uma vez carregado o programa no controlador, apenas há que esperar que tudo "corra bem". É uma aproximação orientada para sistemas completamente estruturados onde tudo é determinista.

A conexão interactiva é condição obrigatória para permitir monitoração de execução.

De facto, pensando numa supervisão de execução a múltiplos níveis, correspondendo às várias abstracções de representação do plano, uma perspectiva interpretada é necessária a cada nível. Por outras palavras, a monitoração duma acção num dado nível implica a interacção com o nível abaixo onde a expansão dessa acção é executada. O tratamento de uma situação de excepção no nível abstracto implicará uma alteração de acções e, portanto, diferentes expansões a serem executadas no nível abaixo (dinamismo na afectação de tarefas para esse nível).

Uma filosofia onde os níveis de abstracção apenas tivessem servido como forma intermédia (aproximação *top down*) para atingir o nível executável - o único a ser mantido no executor - não permitiria este dinamismo de monitoração / recuperação aos diferentes níveis.

Neste sentido, foi desenvolvido, no topo da máquina LM, um interpretador dum subconjunto da linguagem - os comandos relativos ao robô e ferramentas - como forma de recuperar a desejada funcionalidade. O interpretador, escrito em LM, recebe comandos do controlador de estação através duma linha série e executa as acções correspondentes nos componentes físicos (fig. 3.2.4).

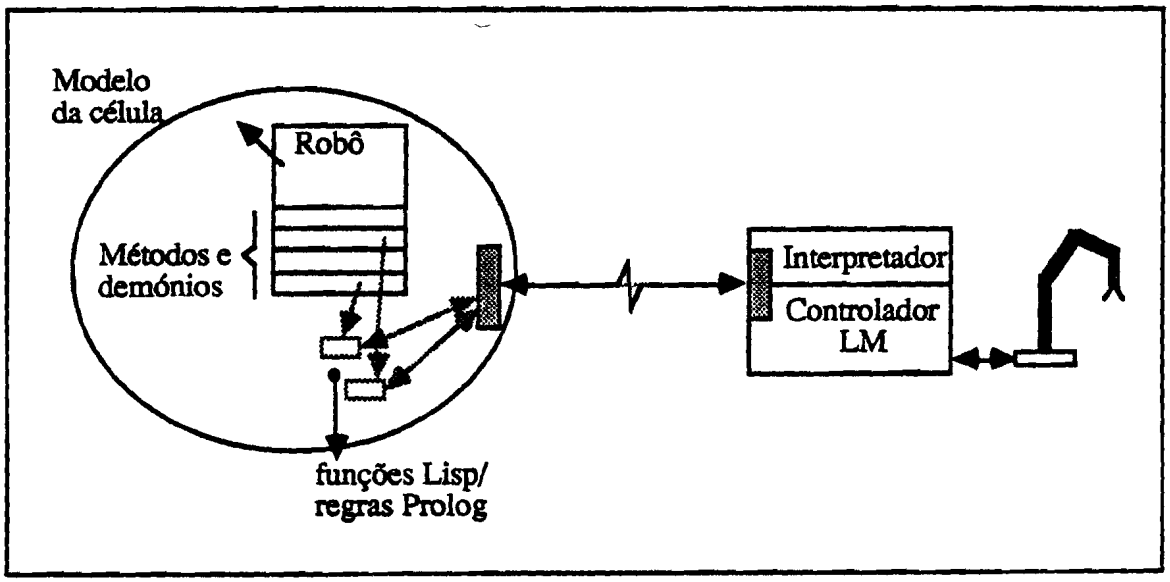


Fig. 3.2.4 Conexão LM - *Frame Engine*

A tab. 3.2.1 mostra a lista de primitivas consideradas na 1ª implementação.

Função	FrameEngine -> LMi		LMi -> FrameEngine
	Código	Parâmetros	Resultado
Interpretador-ligado?	0		Erro
Obter-posição-cartesiana	1		TransfHomog
Obter-estado-robô	2		Estado
Obter-tempo-movimento	3		Tempo
Obter-abertura-garra	4		Abertura
Mover-robô	5	Tipo e TransfH	Erro
Abrir-garra	6		Erro
Fechar-garra	7		Erro
Hard-home	8		Erro
Soft-home	9		Erro
...			
Desligar-interpretador	100		Erro

Tab. 3.2.1 Comandos do interpretador LM

Modelo de *frames*

Do lado do sistema de *frames* , a interface com o interpretador LM foi incluida nos modelos de agentes já descritos, sob a forma de métodos e demónios. Uma primeira implementação foi feita usando o *frame engine* experimental desenvolvido sobre Prolog [CamBar87].

Facilidades adicionais do Knowledge Craft permitiram um posterior refinamento dos modelos, nomeadamente através da possibilidade de definir novos tipos de relações, conforme foi discutido no cap. 3.1.

Numa representação dos operadores através de métodos, a sua activação tem de ser explicitada pelo envio duma mensagem para o agente, conforme se exemplificou em 3.2.2.

Outra alternativa será considerar apenas a programação reactiva. Às diversas acções do componente corresponderiam *slots* cujos valores simbolizariam um efeito desejado.

Assim, por exemplo, um *slot* "posição-corrente" poderia ter associado um demónio do tipo *if-write*. Quando se escrevesse um novo valor nesse *slot*, o demónio seria disparado e desencadearia a execução duma operação "mover" com o fim de colocar o robô de acordo com o valor indicado para o *slot*.

Na solução implementada, adoptou-se uma aproximação mista: aos operadores que sugerem uma acção (movimento, actuação de ferramenta, etc.) associaram-se métodos; em relação a operações de "consulta de estado" (consulta de valores mantidos pelo controlador, ou consulta de sensores) associaram-se demónios.

Na fig. 3.2.5 [CamCor88] ilustra-se um exemplo de utilização de demónios e métodos: um método auxiliar "mostra-estado" destina-se a mostrar, numa janela, o estado corrente do robô. A função que implementa este método consulta o *slot* "estado" o que, por sua vez, dispara um demónio "Ler-estado-dem" que obtém, do controlador LM, a informação desejada .

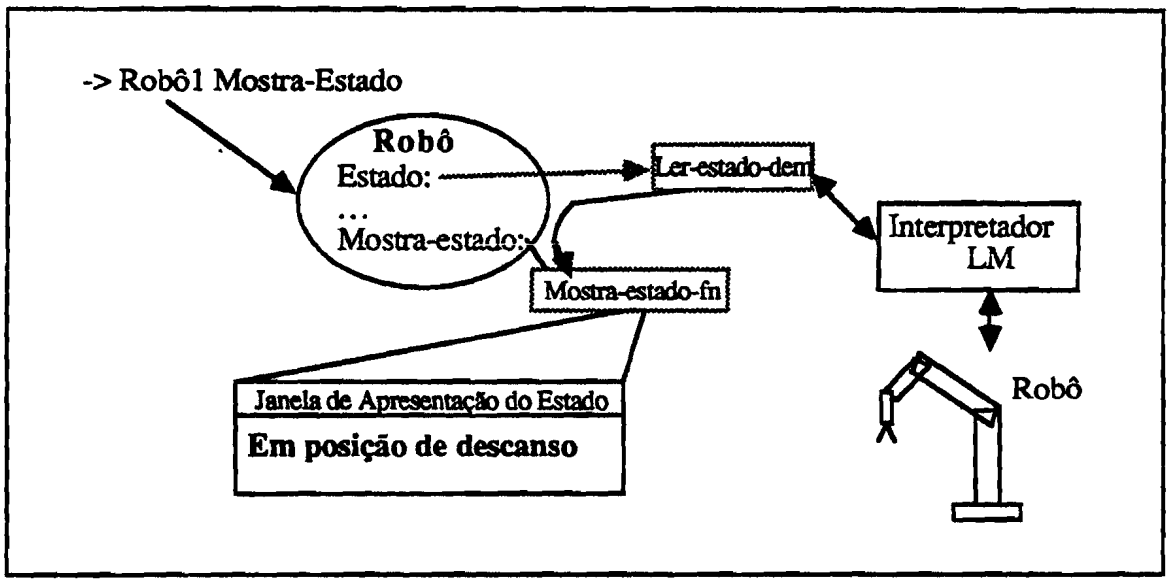


Fig. 3.2.5 Exemplo de utilização de programação reactiva para consulta do estado do robô

Em qualquer das soluções, algo importante a reter é o ter-se conseguido uma uniformização da forma de acesso aos elementos físicos. O mesmo princípio é válido quando os agentes são simulados, mas isso se discutirá no cap. 3.2.6 (Simulação).

É também de referir que os métodos (ou demónios) indicados não têm apenas por missão dialogar com os controladores. Têm também que realizar algumas actividades de

"manutenção" do modelo dinâmico do mundo, quer quanto a efeitos esperados, quer quanto a resultados reais. Por exemplo, ao executar um comando "Trocar-ferramenta", não basta activar o correspondente comando no controlador; é também necessário conectar o modelo do robô com o modelo da nova ferramenta através da relação "ferramenta-corrente" e ligar os respectivos referenciais por uma ligação temporária recorrendo ao gestor de referenciais.

Adicionalmente, como já foi referido e será posteriormente analizado em mais detalhe, um método pode incluir um planeamento especializado local se o comando que ele materializa necessitar de decomposição em operadores mais elementares, função de informação recolhida em tempo de execução.

Percepção

Os aspectos sensoriais podem ser considerados a dois níveis:

-locais aos agentes;

-como agentes por si próprios.

No primeiro caso incluem-se sensores, em geral simples, que fornecem indicações sobre o estado dum componente:

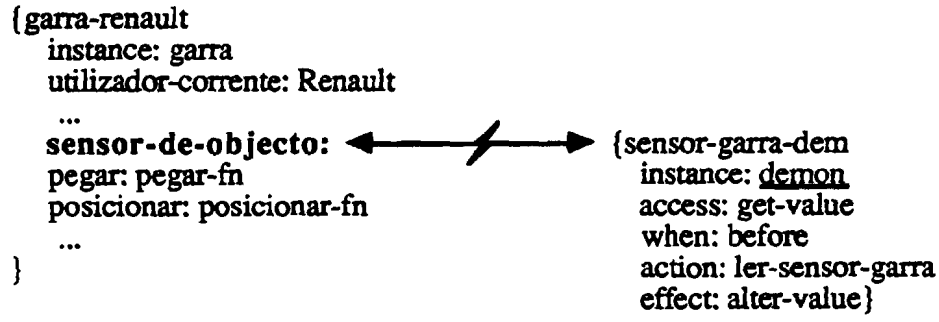
-sensores de força ou de presença / ausência de objecto, associados à garra;

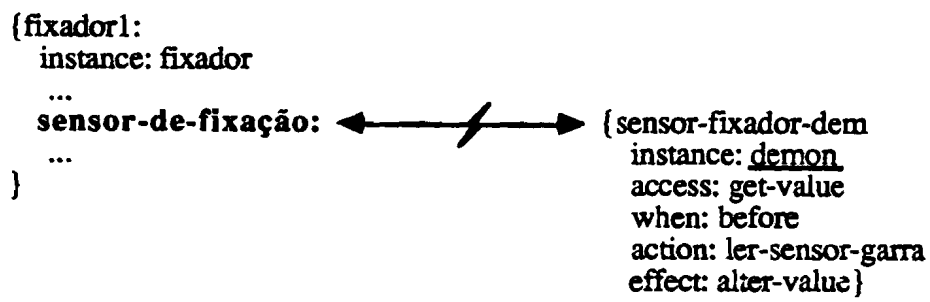
-sensores de posição, associados a componentes móveis (robô, tapete rolante, mesa rotativa);

-etc..

Na aproximação defendida, estes casos são incluídos no próprio modelo dos componentes a que aparecem ligados, em geral como *slots* de estado tendo associados demónios de tipo *if-read*.

Exemplos:





O segundo caso inclui a percepção de alto nível - identificação e localização de objectos, por exemplo.

Os problemas da percepção têm sido alvo de intensa pesquisa, quer no que se refere ao domínio da robótica industrial quer num contexto mais global dos sistemas percepcionais e cognitivos genéricos.

A visão tem sido, indiscutivelmente, o campo onde maiores esforços se têm colocado [Gev82], mas outros sensores, como o tacto, têm vindo a ser objecto dum interesse crescente nos últimos anos. Uma boa panorâmica dos sistemas sensoriais em robótica pode ser encontrada em [Mou86].

Embora ainda continuem a existir imensos problemas nestas áreas, alguns resultados são já acessíveis em termos de produtos comerciais:

-sistemas de processamento de imagem, capazes de extrair um conjunto básico de características (área, perímetro, número de buracos, ...);

-sistemas de visão 2D para identificação de objectos não sobrepostos;

-outros dispositivos sensoriais encapsulados por uma camada de software que trata aspectos básicos de processamento de sinais e fornece uma interface com um grau de abstracção mínimo, através dum conjunto de funções primitivas. Ver [Lor85], por exemplo.

Na década de 80, a ênfase começou a ser colocada na questão da integração multisensorial. Diferentes tipos de sensores permitem diferentes visões do mundo podendo, em princípio, permitir a unificação e complementação de informação originária de múltiplas fontes sensoriais de forma a inferir níveis de informação mais abstractos [MouSteCam86] ou a assegurar redundância que melhore a capacidade de decisão.

A nossa proposta nesta direcção [SteCam86b], [MouSteCam86], já descrita no cap. 2.1 em termos de modelo geral, usa uma aproximação baseada em conhecimento, ou percepção dirigida por modelos, que entendemos ser adequada ao caso da robótica industrial (sistemas moderadamente estruturados e com conhecimento prévio do universo de objectos a identificar / localizar).

Na arquitectura de agente percepcional consideram-se duas fases: i) selecção de características / treino do identificador e ii) reconhecimento (fig. 2.1.10).

A selecção de características que deverão constituir os elementos discriminatórios para a identificação e que, eventualmente, poderão ser provenientes de múltiplos sensores, deverá ser feita em função do conjunto de objectos (peças) com que o sistema terá de lidar, das características sensoriais da estação e de conhecimento sobre o domínio de aplicação (condições de iluminação, ruído, etc.). Para além da selecção de características há que determinar a ordem (ou ordens alternativas) de extracção, função do seu poder discriminatório, custos de extracção e fiabilidade (confiança). Este conhecimento pode ser adquirido de forma automática por aprendizagem a partir dum conjunto preparado de exemplos (treino) com base nos objectos reais ou nos respectivos modelos CAD. Como resultado desta fase pode gerar-se, por exemplo, um conjunto de regras ou árvore de decisão que constituirá a base de conhecimento de suporte à fase de reconhecimento.

Na segunda fase são activadas as acções sensoriais de acordo com o objectivo pretendido (identificação ou localização, por indicação do supervisor a partir do plano genérico) e de acordo com a base de conhecimento específico produzida anteriormente. Por outras palavras, o módulo de reconhecimento será o executor das árvores de decisão implícitas nas regras que constituem a sua BC. Uma árvore de decisão pode ser vista como um plano local - o detalhe dum comando percepcional presente no plano genérico.

Alguns dos custos e factores de disponibilidade usados na selecção / ordenação de características têm uma natureza dinâmica e, portanto, essa tarefa não deve ser integralmente realizada na fase (*off-line*) prévia à execução do reconhecedor. Por exemplo:

-Um sensor (extractor de características) pode estar temporariamente inibido ou em certos casos não ser aconselhável embora seja o preferível noutras situações:

.um componente pode ter avariado

.um sensor pode ter tido o seu campo sensorial obstruído por qualquer outro objecto na cena

.alguns sensores apenas funcionam adequadamente em certas circunstâncias (exemplo: o desempenho dum sonar depende das distâncias).

-Algumas "restrições" podem advir do plano genérico. Por exemplo, a activação da aquisição de algumas características que dependam de sensores suportados pelo manipulador pode "destruir" a posição corrente deste, o que poderia significar a destruição dum objectivo parcial anteriormente atingido (interacção negativa entre subobjectivos).

Desta forma, um terceiro componente no agente percepcional foi considerado - afinador - permitindo a reformulação dinâmica das regras a usar pelo reconhecedor. Assim, a BC do módulo de reconhecimento é preparada em duas fases:

.por treino em *off-line*,

.pelo afinador, baseado em factores dinâmicos.

A intenção de usar um sistema capaz de realizar algum afinamento da BC do reconhecedor, levanta algumas questões sobre a forma de representação a usar. Uma maior flexibilidade parece ser atingível com um modelo de representação mais declarativo - regras *if-then* simples - sendo o encadeamento de regras realizado durante a execução (aproximação "tradicional" em grande parte dos sistemas periciais). Contudo, esta solução não é tão eficiente como a que se poderia conseguir com uma representação mais procedimental - encadeamento à priori de regras simples formando regras mais complexas (uma forma de compilação de árvores de decisão, estratégia adoptada pelo RuleMaster [Rad86]) mas onde a reconfiguração parece mais difícil.

Implementações parciais mas demonstrativas deste modelo foram realizadas pela Linha de Sistemas Sensoriais do Grupo de Robótica da UNL [SteSanQue87], [SteMouSan88]. Este sistema é apenas baseado em visão (2D) e foi desenvolvido primeiro sobre o Intelligence Compiler [Int86b] e depois sobre Prolog / Pascal.

Um dos aspectos importantes do protótipo é o da representação de conhecimento e raciocínio imprecisos (*fuzzy knowledge representation and reasoning*) com funções de pertença materializadas por distribuições Gaussianas normais, determinadas durante a fase de treino.

Um forma de materialização dos aspectos de reformulação dinâmica da base foi através da alteração dinâmica dos factores de confiança.

Um outro exemplo com algumas semelhanças e também baseado numa aproximação por raciocínio impreciso pode ser encontrado em [HirAraHac87].

Questões de integração

A integração dum subsistema percepcional no sistema executivo pode ser feita de forma semelhante à dos outros componentes: num *frame* representando o agente percepcional ter-se-ão *slots* suportando métodos que representam as operações realizáveis por este.

Desta forma, uma integração do identificador de objectos desenvolvido na UNL foi feita [CamCor88], utilizando também numa 1ª fase, o *frame engine* desenvolvido em Prolog (fig. 3.2.6). Uma implementação em Knowledge Craft tem um processo similar.

```
{{Identificador
        é-um: componente-fixo
        ref-base: ref-ident
        identificar-obj: identificar-obj-fn   |
        localizar-obj: localizar-obj-fn        > métodos
        treinar-ident: treinar-ident-fn        |
        ...
}}
```

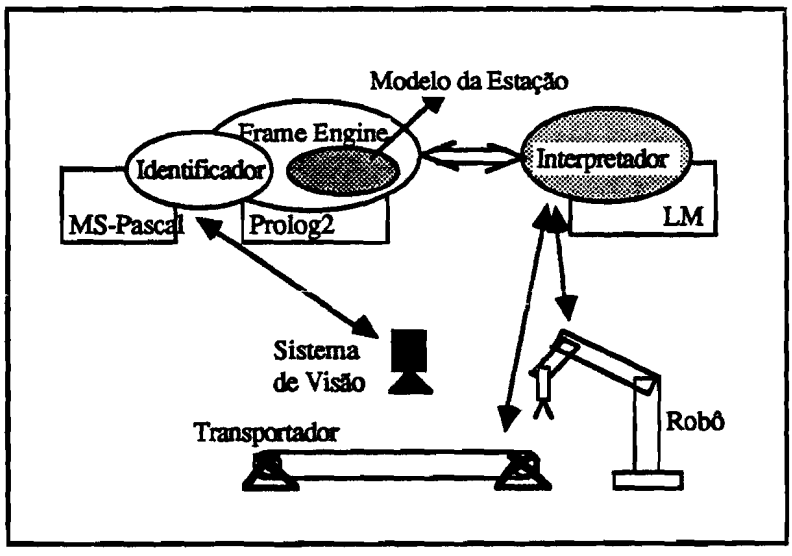


Fig.3.2.6 Integração do sistema identificador de objectos

As primitivas consideradas são:

-identificar-objecto - permitindo identificar ou classificar um objecto no campo de visão;

-localizar-objecto - determinando a localização dum dado objecto, dentro do campo de visão;

-treinar-identificador - correspondente à primeira fase, em que se gera a base de conhecimento específica para um dado universo de objectos.

Novamente o nível de abstracção oferecido pelo conceito de agente está acima do oferecido pelo identificador sòzinho, o que também é consequência do próprio processo de integração. Assim, por exemplo, a operação de "localizar-objecto" oferecida pelo identificador fornece um referencial 2D localizado no centro geométrico da projecção do objecto e com orientação determinada pelo seu eixo principal. Torna-se necessário algum processamento adicional de forma a passar para um referencial 3D. Isto pode ser conseguido com base num conhecimento da localização do alimentador, por exemplo, donde resulta um conhecimento à priori da coordenada z e apenas as coordenadas x,y teriam de ser determinadas pela visão (considerando que não há sobreposição de objectos).

O referencial base do objecto pode, então, ser derivado deste, com recurso ao processador de referenciais, desde que se conheça a respectiva matriz de transformação (que pode ser obtida a partir do modelo geométrico do objecto).

Há ainda um outro problema intermédio: o referencial 2D fornecido pelo identificador é relativo a um referencial localizado num dos cantos do rectângulo de visibilidade da camara. Conhecendo a localização da camara na cena, há então que proceder a uma transformação de coordenadas de forma a obter um referencial relativo ao referencial da estação.

A propósito destas questões de correspondência entre referenciais, é de citar a importância da rotina MANUAL desenvolvida no sistema LM [BarRosCamNev86], que permite obter referenciais de pontos da cena real relativos à base do robô.

Integração "mais profunda"

É de notar que o processo descrito refere apenas uma integração parcial ou uma visão particular dos problemas da integração da percepção numa perspectiva "utilizador".

Um grau mais profundo de integração pode ser tentado se se analisar o problema na perspectiva da própria geração do identificador.

Uma das direcções é a da integração da percepção e dos modelos CAD. Como se viu anteriormente, a base de conhecimento do identificador pode ser parcialmente "alimentada" com informação extraída dos modelos geométricos produzidos no CAD. Na versão actual do identificador UNL, a conexão entre os dois sistemas não passa pelo sistema de informação mas é feita de forma externa. Isto é, uma geração de imagens sintéticas é feita a partir dos modelos CAD e tal informação alimenta directamente o sistema de visão. Informação resultante da intersecção entre um plano de luz (gerado por laser) e objectos 3D pode também ser adequadamente simulada. Mas outros aspectos que poderiam ser derivados de raciocínio geométrico ou de informação adicional sobre o

produto (material, cor,etc.) não estão a ser fornecidos ao identificador pelo SI. A análise das vias para uma maior integração e avaliação da sua adequabilidade constitui pois uma questão em aberto.

Um aspecto complementar e que pode também contribuir para o reforço da integração, tem a ver com os paradigmas de representação de conhecimento usados internamente pelo identificador. Numa primeira versão apenas foram usadas as características básicas do Prolog. Numa segunda versão, a utilização dum conjunto de paradigmas como os usados no SI (*frames*, programação orientada por objectos, reactiva e por regras) facilita o acesso ao SI por parte do identificador.

Um exemplo de trabalho usando tais paradigmas, embora sem a aproximação integrada aqui defendida, pode ser encontrado em [KakBoy...86].

Também na UNL se encontram em curso trabalhos de exploração desta via, quer na visão [SteMouSan88], quer no tacto [MouPim88], que permitem esperar um aumento das facilidades de integração no futuro.

Estas ideias poderiam igualmente generalizar-se para a construção de novos controladores do robô e outros componentes.

Planeadores especializados

Formas de integração

Como já foi referido, algumas das acções primitivas usadas no nível mais baixo do plano genérico constituem ainda macro-operadores que têm de ser decompostos de forma a chegar-se ao nível das operações elementares oferecidas pelos controladores dos componentes da estação.

Esta decomposição, porém, apenas pode ser realizada em tempo de execução do plano, altura em que estará acessível a informação necessária para realizar o detalhe. Estão neste caso todas as operações que envolvem o estabelecimento de contacto com as peças e que requerem movimentos finos (pegar, largar, inserir-pino, etc.) e que devem ser condicionadas por realimentação de informação sensorial (sensores de força, por exemplo).

Na abordagem proposta, tais módulos de planeamento especializado são parte integrante dos agentes. Desta forma, um agente oferece um conjunto de primitivas a usar no nível mais baixo do plano genérico (que se pode considerar de nível objecto, para usar uma terminologia tradicional). Os planeadores locais transformarão tais primitivas na sequência de comandos elementares aceites pelos controladores. Complementarmente, um

modelo dinâmico local do mundo será mantido a nível do agente (não faz sentido "sobrecarregar" o modelo global com detalhes apenas interessantes para o agente).

Assim, a forma de activar tais planeadores é implícita: quando o supervisor de execução activa um método ou demónio do agente pode estar, implicitamente, a desencadear neste um processo interno de planeamento especializado (fig. 3.2.7).

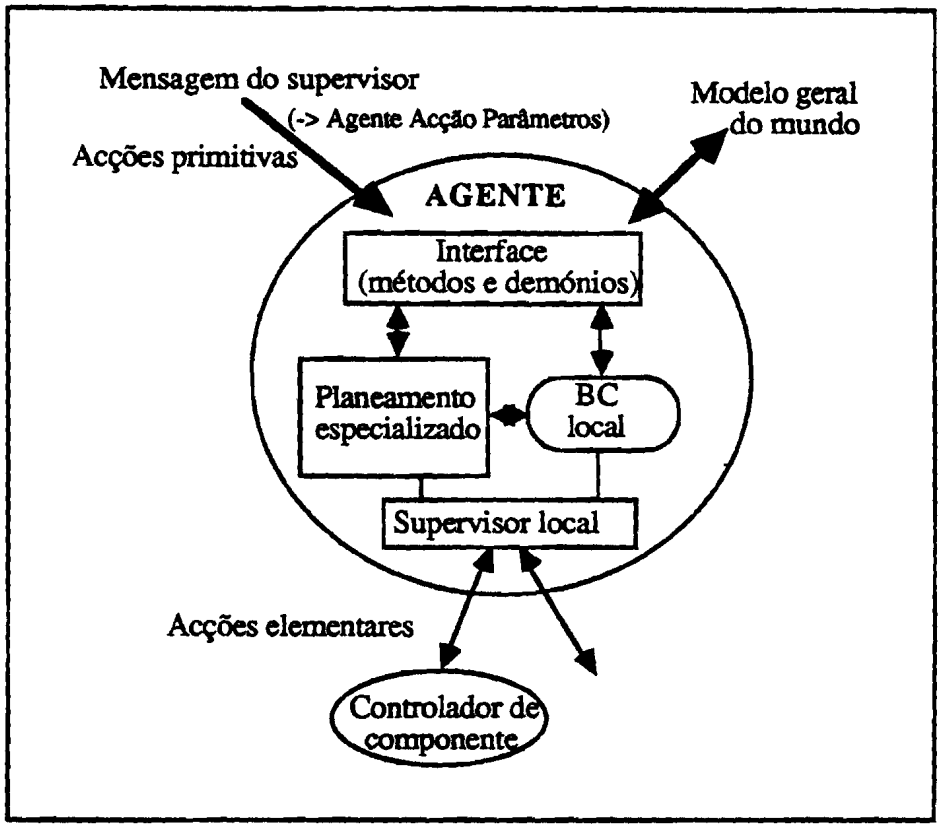


Fig. 3.2.7 Detalhe dum agente

A BC local contém as estratégias de planeamento especializado do agente e o seu modelo particular do mundo. Os aspectos mais gerais do modelo dinâmico do mundo estão no sistema global.

O problema do planeamento especializado é complexo e bastante específico para uma dada estação / domínio de aplicação. Contudo, um conjunto de operações primitivas para aplicações de montagem pode ser esboçado e, portanto, um conjunto de planeadores especializados identificado. Na fig. 3.1.23 dão-se exemplos destas operações.

O nosso trabalho não tem incidido nestes tópicos, contudo é importante fazer uma breve análise do estado da arte e tendências de forma a verificar se a arquitectura proposta é consistente.

A título de exemplo, podem-se considerar o caso da operação "Pegar Base-1" que pode ser decomposta na sequência de acções elementares (rever fig. 3.1.1):

•Aproximar de Base-1 - só depois de conhecer a localização da peça é possível determinar o referencial de aproximação em relação à base do robô.

•Abrir garra

•Movimento fino para referencial de preensão

•Fechar garra - condicionada por sensor de força

•Movimento fino para referencial de afastamento.

Este é apenas um exemplo simplificado.

De notar que uma parte do trabalho é feita em *off-line*, como foi referido em 3.1.6: determinação dos pontos de preensão e posições (trajectórias) relativas de aproximação e afastamento. (Neste exemplo há acesso aos modelos globais do SI.)

Outra tarefa exigindo um planeamento especializado para pequenos movimentos é o encaixe de peças (*parts mating*) - aproximar e tentativamente introduzir uma na outra. No caso do teste-padrão de Cranfield, tem-se inserção de pino em buraco (*peg-in-hole*) e posicionamento de peça sobre outra ou no fixador; noutros exemplos pode-se ter inserção de parafuso, etc..

Note-se que se não existissem incertezas seria fácil atingir a relação geométrica desejada entre as peças. Todavia, devido às imprecisões:

-o deslocamento do manipulador não é absolutamente preciso;

-as próprias peças têm tolerâncias relativamente ao seu modelo e a sua localização também não é conhecida com rigor;

torna-se necessário usar informação sensorial (de posicionamento, força, etc.) para determinar se o objectivo final foi atingido ou não, e mesmo para auxiliar o processo (a trajectória é dinamicamente modificada em função das forças de contacto ou estímulos tácteis que ocorrem durante o movimento). Em caso de insucesso será necessário gerar movimentos finos (*compliant moves*, movimentos complacentes ?) correctivos. Esses movimentos (e eventualmente outras operações) podem ser guiados por um conjunto de regras heurísticas que sugiram, para cada tipo de situação, o procedimento a seguir.

O sistema deverá dispor também de conhecimento sobre o domínio específico de actuação. Por exemplo, qual a força máxima que poderá efectuar na compressão de duas peças para tentar "levá-las ao lugar".

Em muitos dos trabalhos nesta área, a ênfase tem sido posta no raciocínio geométrico. Veja-se, por exemplo, [LozBro85] e [LauPer85].

Este problema é, contudo, bastante dependente das características dos órgãos terminais e ferramentas do robô e, portanto, das evoluções a nível do hardware. Vários esforços têm sido desenvolvidos no sentido de dotar tais dispositivos de capacidades de "complacência" [FuGonLee87], [Sta87]:

-"complacência" passiva, como no exemplo de garra (*remote compliance center*) de Nevin e Whitney [NevWhi78];

-"complacência" activa, suportada por um ciclo de realimentação de informação sensorial. Por exemplo, um dispositivo, colocado no punho do robô, capaz de medir as componentes da força e da torção ao longo de x, y e z poderá ser o suporte para uma estratégia de inserção em que movimentos finos correctores são gerados em resposta aos valores medidos.

De facto as duas perspectivas complementam-se: o raciocínio geométrico para determinar formas de aproximação e condições finais pretendidas; a utilização da "complacência" / controle fino para realizar os ajustes.

Em qualquer caso, a abordagem feita neste trabalho assume que estas questões devem ser resolvidas a nível local.

Dada a dependência dos dispositivos concretos usados, a criação duma biblioteca de procedimentos (ou planeadores) especializados deve fazer parte do processo de instalação da estação, caindo numa zona de "software de sistema".

No cap.3.1 assumiu-se que a selecção das ferramentas a usar para cada subtarefa era feita durante o planeamento de processo. Outra alternativa seria considerar que uma parte dessa tarefa poderia ser realizada por planeadores especializados a nível dos diversos agentes. Assim, caberia ao agente robô, em função do comando recebido e do conjunto de ferramentas disponíveis, decidir qual a ferramenta a usar. Em termos de modelos locais do mundo, implicaria um conhecimento da aplicabilidade de cada ferramenta ou um conhecimento das ferramentas de defeito para cada tipo de tarefa (opção também considerada na implementação realizada).

Porém, se esta selecção for feita não em função de cada comando, mas em função dum subplano, alguma reordenação de acções pode ser efectuada de modo a reduzir as trocas de ferramentas.

Finalmente pode-se referir que o processo de identificação de objectos (agente percepcional) descrito atrás, também pode ser encarado como uma questão de planeamento especializado: planeamento de estratégia de recolha de informação (selecção /

ordenação de características), eventual interacção com outros agentes (por exemplo, quando o sensor está suportado pelo robô), etc..

Desta forma, o agente percepcional também segue o modelo genérico de agente apresentado anteriormente.

Em termos de modelo dinâmico do mundo é de referir que estes planeadores locais podem necessitar manter um modelo mais detalhado do que o necessário para a supervisão global. Contudo julgamos que, novamente, isto deve ser resolvido a nível local sem que se justifique a introdução de complexidade adicional a nível da arquitectura geral.

Concorrência

Um dos aspectos importantes no desenvolvimento dum sistema executivo multiagente é o da capacidade para exprimir a concorrência de processos e sua interacção.

A inexistência de ambientes de programação / representação de conhecimento com as características definidas atrás e ao mesmo tempo suportando a concorrência, constitui uma limitação nesta fase.

Desenvolvimentos como o Delta Prolog [PerMon...85] poderão constituir linhas interessantes a explorar no futuro. Noutra área, o reacender de interesse pelas redes neuronais e as potencialidades de paralelismo real oferecidas pelos "transputers", deverão também ser tidos em conta na perspectiva da concepção de novos sistemas de supervisão e controle.

Contudo muita da evolução nestas áreas têm vindo a acontecer durante o mesmo período temporal do trabalho aqui reportado, pelo que a opção tomada foi de não "misturar" problemas. Por outras palavras, não se julgou razoável explorar a potencialidade dessas vias - laterais em relação ao objectivo deste trabalho - enquanto os primeiros resultados minimamente estabilizados não ficarem aí acessíveis.

Desta forma, a estrutura central do sistema de supervisão desenvolvido tem uma execução essencialmente sequencial, embora algum paralelismo real exista - pelo menos potencialmente - dada a estrutura computacional distribuída de suporte.

Uma vez que os vários controladores dos diversos componentes da estação têm, normalmente, o seu processador associado, pode tirar-se algum "proveito" disso. Para tal, quando o supervisor envia uma mensagem para um agente, não deve ficar bloqueado à espera do fim da correspondente operação, isto é, o método despoletado por essa mensagem apenas inicia a operação no agente mas devolve de imediato o controle ao supervisor. De notar que linguagens como o LM, através da opção *nowait*, fornecem

adequado suporte para um tal funcionamento. A nível do *frame* representativo do agente é necessário introduzir um novo método - *wait* - que permita ao supervisor sincronizar-se com esse agente.

Para estações não muito complexas, esta solução pode ser aceitável, até porque os objectivos deste trabalho estavam mais centrados na análise da estrutura global do sistema. Tendo, porém, sistemas com vários robôs cooperando entre si ou sistemas de monitoração com apertados requisitos de tempo real, já o problema da expressão da concorrência e questões de sincronização assumem um peso mais significativo [Cam83].

Alguns trabalhos de investigação em curso procuram aplicar formalismos derivados das redes de Petri para o desenvolvimento de sistemas de controle de estações com tais características. Um exemplo, apenas a nível de simulação nesta fase, é representado pelo trabalho de Negretto na Universidade de Karlsruhe [Neg88].

3.2.4 - MONITORAÇÃO DE EXECUÇÃO

O problema

Durante a fase de programação / planeamento genérico não é possível antecipar todas as situações que poderão ocorrer durante a execução. Na execução real do plano há que admitir [Cam87a]:

-A possibilidade de as operações do plano poderem não atingir os efeitos desejados (dada a impossibilidade de modelar completamente cada operador e o ambiente envolvente);

-Acontecimentos imprevistos (não estamos num mundo fechado);

-A informação proveniente dos sensores apresenta, em geral, um certo grau de imprecisão;

-Tolerâncias mecânicas (robô, peças) podem introduzir erros à medida que se avança na execução do plano.

Não sendo então praticável a contemplação de todas as situações durante o planeamento, porque

-os modelos do mundo são incompletos, e também porque

-não se justifica uma elevada carga de processamento para contemplar casos raros,

a aproximação seguida é a de produzir um plano para as "situações normais" e dotar o sistema executivo de capacidade para lidar com as excepções quando elas ocorrerem.

Alguns trabalhos preliminares da Inteligência Artificial introduziram esta questão, embora ainda de forma simplificada e sem conexão directa ao mundo físico (não eram aplicados a sistemas robóticos reais).

Uma das primeiras tentativas de integrar um gerador de planos e um supervisor de execução é representado pelo sistema PLANEX [FikHarNil81]. O PLANEX monitora a execução dum plano produzido pelo planeador STRIPS e tem em atenção eventuais "surpresas" positivas e negativas:

-Quando a informação obtida durante a execução do plano mostra que certas partes do mesmo já não necessitam ser executadas (surpresa positiva), o executor deve reconhecer isso e omitir os desnecessários passos.

-Quando a execução duma parte do plano falha (não atinge os objectivos pretendidos), o executor deve reconhecer a falha e dirigir a reexecução dessa parte ou solicitar um replaneamento.

Outro trabalho que introduziu alguns conceitos que, embora desenvolvidos noutro contexto, podem ser aplicáveis nesta zona foi o sistema HACKER [Sus75]. Sussman tenta captar o conceito de *evolução* dum programa no seu processo de geração de planos:

-Um código (plano) já existente em biblioteca é generalizado para novas situações por um processo de depuração (*debugging*) da causa que provocou a sua falha nessas situações (numa execução simulada).

-A proposição de novos planos - quando nenhum dos existentes se adapte - também tenta uma abordagem simplificada à partida que depois irá sendo corrigida.

Estas ideias são extensivamente usadas no HACKER. Por exemplo, a falta de uma pré-condição para a aplicação de um operador que tenha sido seleccionado é também formalizada como "bug" cujo procedimento de correcção vai conduzir à produção do subplano para o atingir dessa pré-condição.

Estas ideias estão relacionadas com o princípio referido de gerar um plano para as situações normais e depois tratar as excepções. Todavia, o objectivo do HACKER é mais um de aprendizagem no sentido em que o plano em produção vai sendo refinado por um processo de crítica. No caso da monitoração de execução, o plano original continuará a aplicar-se após se ter ultrapassado a excepção. Isto é, o tratamento de recuperação tem um efeito transitório, não havendo, portanto aprendizagem. (Todavia, os aspectos de aprendizagem poderão ser importantes para o refinamento do sistema de monitoração, como se refere no cap. 4).

Muitos dos trabalhos posteriores em geração de planos ignoraram quase por completo - em termos de realizações - esta problemática.

Novas evoluções começaram, então, a surgir mas do lado da Robótica, embora de forma ainda pouco integrada. Dois exemplos significativos podem encontrar-se em [GinGin83] e [LeeBarHar83] que, contudo, apenas se centram nos aspectos de detecção e correcção de erros durante a execução, partindo dum plano (programa) previamente construído (por um planeador ou por um programador) mas sem integração entre os subsistemas de programação e monitoração.

Em geral, para a existência dum sistema de monitoração de execução têm-se como pré-condições:

-Existência de expectativas relativamente aos efeitos das diversas etapas do plano, o que deve ser indicado no próprio plano (conhecimento sobre a tarefa -> plano documentado);

- Capacidade de observação dos resultados da execução, normalmente com recurso a meios sensoriais;

-Capacidade de avaliação, isto é, comparação dos resultados esperados com os observados e avaliação/ caracterização dos desvios.

Com base nessa caracterização (diagnóstico) dos desvios ou situações de excepção, deve ser tentado um procedimento de recuperação a partir do conhecimento do estado actual do mundo e dum conjunto de regras de recuperação.

Arquitectura multinível

Como é natural, a arquitectura do subsistema de monitoração de execução deve estar de acordo com a estrutura multinível do plano e do supervisor de execução.

Por outras palavras, a cada nível do plano, deve corresponder uma camada de monitoração, conforme se indica na fig. 3.2.8 para o caso de 2 níveis.

Para cada nível de acções tem-se monitoração e tentativa de recuperação em caso de erro. Caso essa tentativa de recuperação falhe, o nível seguinte é notificado, o que tem por consequência a falha do operador abstracto corrente.

Quando o nível mais elevado é incapaz de recuperar duma situação de excepção, o controle é passado ao operador humano que, então, deverá providenciar directivas a seguir. Por outras palavras, o operador poderá introduzir manualmente um plano de recuperação ou forçar uma paragem do programa.

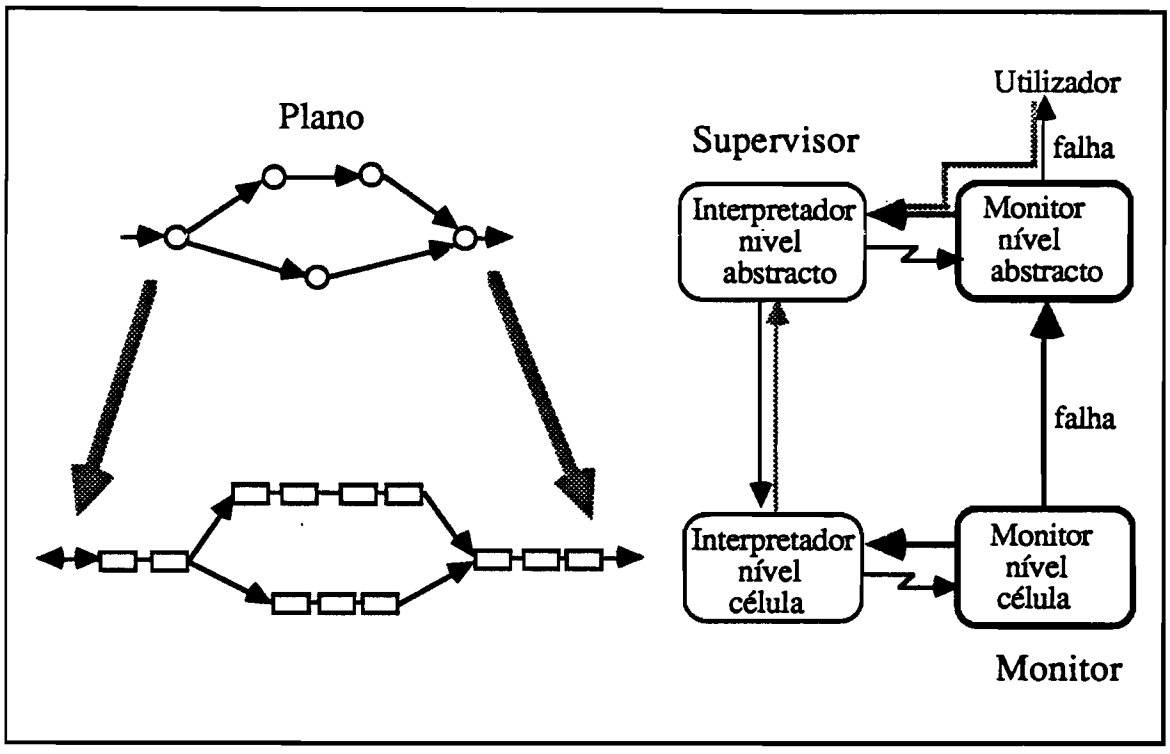


Fig. 3.2.8 Estrutura a dois níveis para monitoração de execução

Para cada nível, o subsistema de monitoração pode ser decomposto nos módulos:

•Recolha de informação - observação, dirigida, dos efeitos da acção não se está pensando num levantamento sensorial completo do estado do mundo mas apenas aquisição de informação pertinente para a avaliação desejada);

•Diagnóstico - detecção de excepções e sua classificação com base nos efeitos esperados e no resultado da observação;

•Recuperação - geração dum subplano para resolver a excepção, de acordo com o diagnóstico.

Na fig.3.2.9 ilustra-se esta estrutura para o nível de abstracção mais baixo (nível dos agentes).

O supervisor, para além de distribuir as acções pelos agentes, torna essas acções acessíveis ao subsistema de monitoração para que a análise do seu resultado seja feita.

O subplano de recuperação é depois submetido ao supervisor que, no caso duma execução bem sucedida, regressa ao plano normal. Em certas situações pode-se ter uma excepção a este funcionamento. Por exemplo, no caso em que uma ferramenta se parta, o plano de recuperação pode ter de ser usado durante algum tempo.

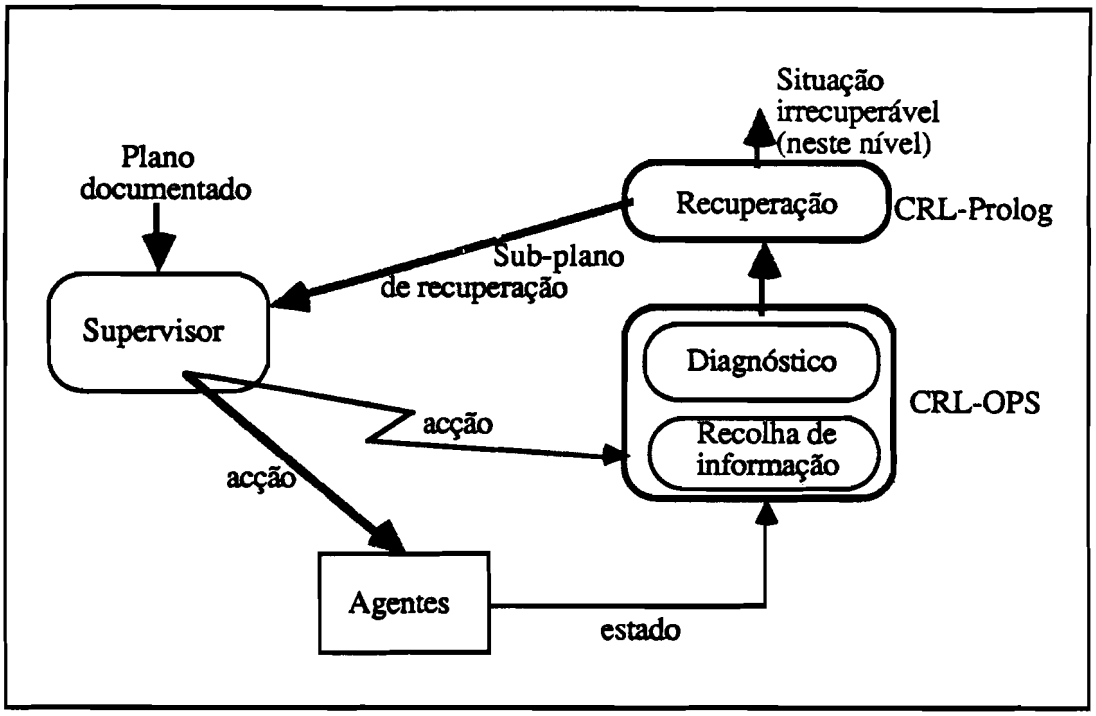


Fig. 3.2.9 Detalhe do subsistema de monitoração

Recolha de informação e diagnóstico

A detecção duma falha pode ser feita de várias formas:

i- Detecção "interna" realizada pelo nível abaixo que devolve o resultado da execução de cada acção (sucesso ou insucesso). No caso da monitoração no nível mais baixo, esta detecção é feita pelos controladores dos componentes da estação que devolvem para cada acção elementar o respectivo resultado (ver tab. 3.2.1).

ii-Detecção por observação complementar, baseada em sensores (reais ou virtuais). Nos níveis de execução abstracta é mais difícil visualizar este aspecto, mas ele pode ser representado por qualquer outra informação adicional que contradiga (ou confirme) uma aparente certeza (dada pelo sucesso do nível abaixo)...

iii. Detecção pela monitoração do comportamento dos componentes da estação, conforme se verá em 3.2.5

Em termos do trabalho experimental realizado, como não se têm facilidades de expressão de paralelismo de execução, as "amostragens"/observações são feitas no final da execução de cada acção (fig.3.2.10).

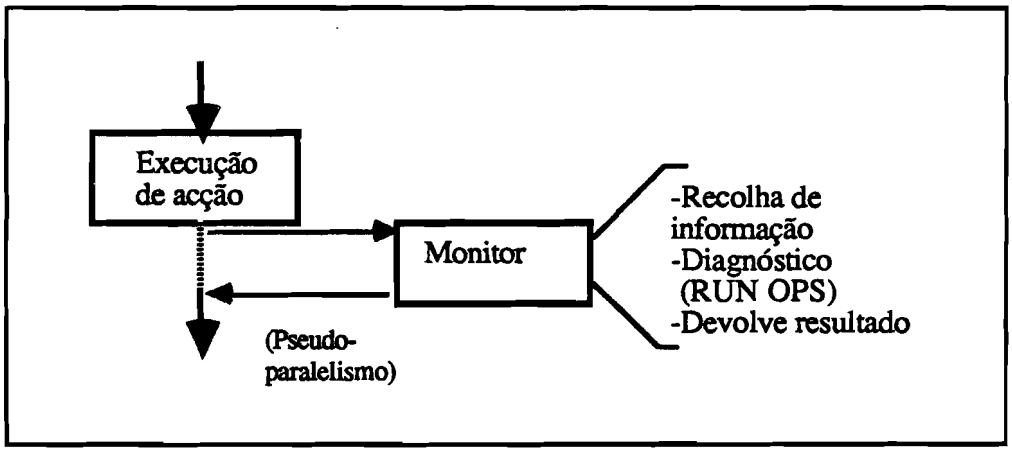


Fig. 3.2.10 Pseudo-paralelismo execução-monitoração

A observação e diagnóstico podem ser dirigidos por um conjunto de regras. Como se está tentando classificar uma situação a partir dos dados conhecidos, pode-se pensar na utilização dum processo de raciocínio por encadeamento para a frente (*forward* ou *data driven inference*). Assim, foi usado o componente CRL-OPS do Knowledge Craft, um derivado do sistema OPS5 [BroFar...85], mas integrado no sistema de representação por *frames*.

A regra seguinte (do tipo **se** <condição> **então** <acção>) ilustra o processo de recolha de informação para o nível mais baixo. Uma breve descrição da sintaxe usada pode ser encontrada no Anexo A.

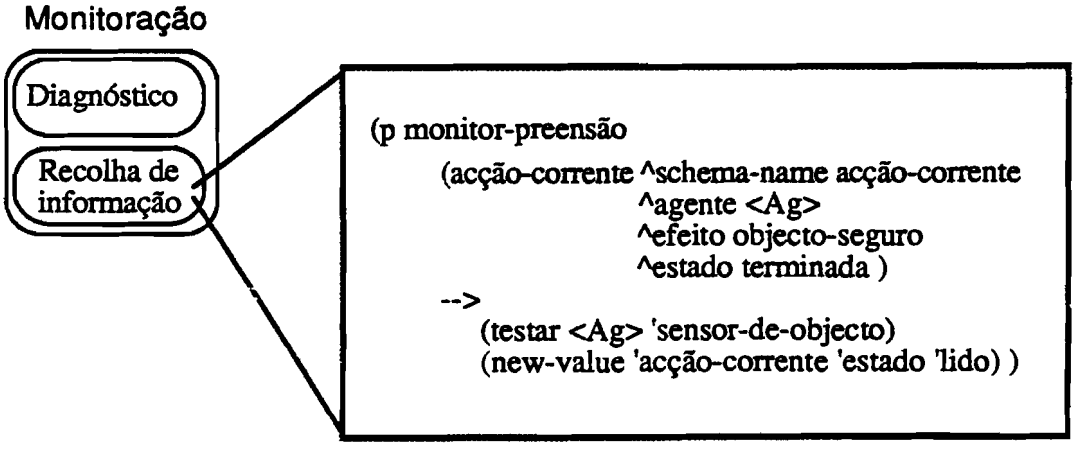


Fig. 3.2.11 Exemplo de regra para recolha de informação sensorial

Isto é:

Se a acção corrente tiver por efeito "objecto seguro" e o seu estado for "terminada" ,
qualquer que seja o agente executor Ag
então
testar o "sensor-de-objecto" desse agente Ag e mudar o estado da acção corrente para "lida".

Como se verifica, a recolha de informação aqui sugerida é dirigida por um objectivo: A regra exemplificada é aplicável quando se pretende verificar o efeito de preensão dum objecto pela garra do robô. Desse modo apenas a informação sensorial adequada (sensor da garra) é adquirida. Não se estão considerando ambientes muito "hostis" e, portanto, não se contempla a exigência de grande "omnisciência" ou capacidade de observação permanente do estado do mundo. No caso dum sistema com elevada redundância sensorial, isto é, onde o conhecimento do estado do mundo fosse baseado na concorrência de múltiplas fontes de informação sensorial, ter-se-ia adicionalmente que tratar a questão da manutenção da coerência dos modelos dinâmicos.

De notar também que o objectivo da acção corrente é explicitado pelo *slot* "efeito" do respectivo *frame* (objecto-seguro, neste caso).

Esta aquisição de informação pode implicar um planeamento local: a função "testar" pode chamar um conjunto de regras (Prolog, por exemplo) que planeiem a recolha de informação.

Este exemplo também permite ilustrar a interface das regras OPS com o CRL:
-para consulta, na parte de <condição> da regra
-para alteração, na parte de <acção>.

A aquisição de informação despoleta, ao garantir as pré-condições das respectivas regras, o processo de diagnóstico/ classificação da situação. No exemplo seguinte apresenta-se uma dessas regras.

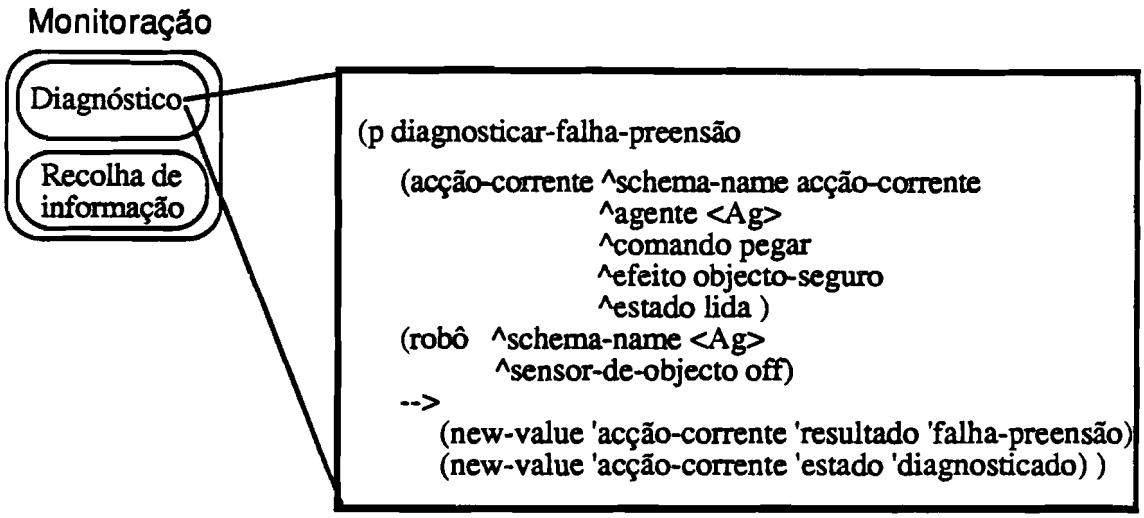


Fig. 3.2.12 Exemplo de regra de diagnóstico

Isto é:

Se a acção corrente, a executar por um agente Ag, consistir no comando"pegar",
tiver por efeito pretendido "objecto seguro" e o seu estado for "lida"
e o agente Ag tiver o valor "off" no seu "sensor-de-objecto"
então
diagnosticar "falha de preensão" como resultado da acção corrente e mudar o seu estado desta para "diagnosticada".

A situação de sucesso será diagnosticada por defeito - se nenhuma outra regra se aplicar face à informação adquirida.

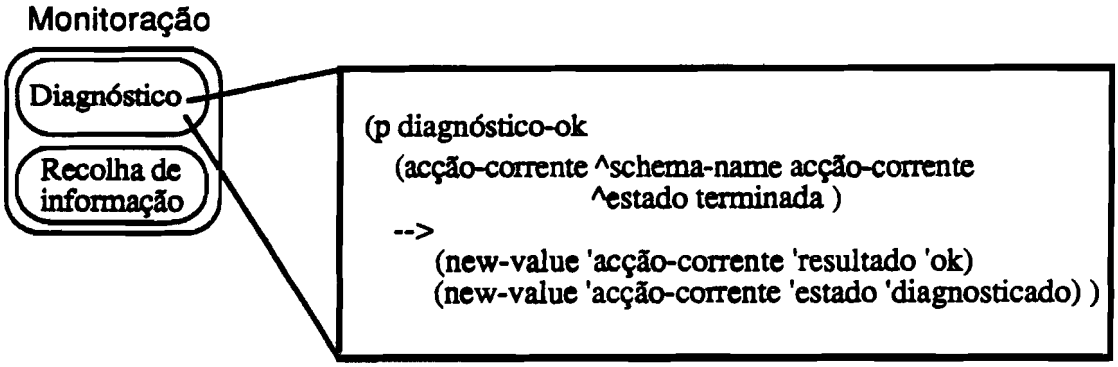


Fig. 3.2.13 Regra que diagnostica sucesso de operação

Isto é:

Se a acção corrente tiver estado = "terminada"

então /*caso nenhuma outra regra seja aplicável*/

diagnosticar resultado "Ok" e modificar estado para "diagnosticada".

Ou seja, um diagnóstico OK significa que o sistema de monitoração - face ao conhecimento de que dispõe - foi incapaz de detectar um erro.

A estratégia de resolução de conflitos do CRL-OPS (ver Anexo A) assegura que esta regra só será executada se não existir nenhuma mais específica em condições de o ser.

Outra questão tem a ver com o facto de os sensores fornecerem informação imprecisa. Deste modo, a interpretação dos valores de observação deveria ser baseada em raciocínio impreciso. Em termos das regras de diagnóstico, em vez dum "pattern matching" relacional simples, podem-se incluir funções booleanas, na parte <condição> da regra, que implementem um critério de decisão adequado.

Na versão corrente do protótipo implementado esta hipótese não foi, contudo, contemplada.

Outro aspecto que se pode relacionar com esta questão é o da combinação de informação redundante. Na implementação realizada considerou-se um contexto onde a capacidade sensorial da estação não é muito redundante.

A estrutura geral da base de regras CRL-OPS pode-se esquematizar da seguinte forma:

(reset ops) ... (strategy mea) ... (schema-literalize robô sensor-de-objecto ...) (schema-literalize .... **L**
(p monitor-preensão ... (p monitor-fixação-peça ... **M**
(p diagnosticar-falha-preensão ... ... **D** (p diagnostico-ok ...

O bloco L - literalizações - constitui uma interface entre o sistema OPS e o sistema CRL (representação por *frames*). Os *frames* representados no sistema não estão automaticamente acessíveis ao OPS mas apenas aqueles que forem tornados "visíveis" através da janela definida pelas asserções *literalize*. Em relação a um *frame* tornado visível, apenas são acessíveis os *slots* indicados na correspondente literalização. Este aspecto do Knowledge Craft deve-se a razões de eficiência do algoritmo de "pattern matching".

Os blocos M e L representam, respectivamente, os conjuntos de regras de monitoração e diagnóstico.

O sistema OPS é activado, conforme fig.3.2.10 , após a execução de cada acção, ou seja, a observação da execução é feita em pontos discretos no tempo. Em princípio, a frequência de verificação dos sensores deve ser de molde a permitir que o sistema pare ou altere as acções em tempo. Isto levanta porém outra questão: a do paralelismo entre execução e observação, o que também coloca requisitos a nível do sistema computacional / modelo de programação.

Algum paralelismo pode ser conseguido "deslocando" parte das tarefas de observação para junto dos agentes / controladores, em vez de centralizar o controle a nível global. É de notar que alguns controladores já fornecem facilidades para controle de execução concorrente. É, por exemplo, o que se reflecte nos "*guarded moves* " das linguagens nível robô.

Neste caso, algumas regras de diagnóstico têm de testar o resultado das operações fornecido pelo controlador ou, em termos mais gerais, cada nível deve analisar o resultado da execução no nível abaixo.

O processo de recolha de informação e diagnóstico também pode ser dirigido por uma base de regras com encadeamento para trás (*backward chaining*).

Exemplos de regras usando o componente CRL-Prolog do Knowledge Craft:

```
(diagnóstico ?Acção falha-preensão) < (:schema <Acção> (comando pegar)
                                                        (efeito objecto-seguro)
                                                        (agente ?Ag)) !
                                      (:schema <Ag> (sensor-de-objecto off))
```

(diagnóstico ?Acção falha-fixação) < (:schema <Acção> (comando fixar)
(efeito peça-fixa))
(:schema fixador1 (sensor-de-fixação off))

Notar de novo a integração entre o Prolog e a representação por *frames*, através do predicado *:schema* (Anexo B).

Nesta versão, a leitura dos sensores é implícita, via programação reactiva, conforme se ilustra na fig. 3.2.14.

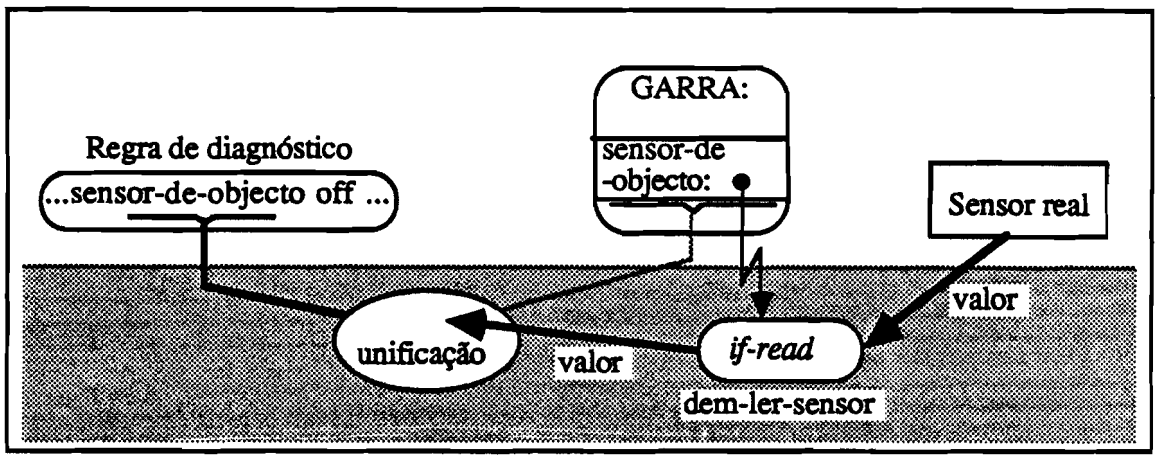


Fig. 3.2.14 Programação reactiva na aquisição de informação sensorial para monitoração

O acesso ao *slot* sensor-de-objecto, implícito na unificação, despoleta um demónio tipo *if-read* associado a esse sensor, que obtém o valor do sensor físico (ou simulado).

Isto não era fácil na versão CRL-OPS porque o algoritmo de unificação aí usado não faz apelo às funções normais do CRL para acesso aos *slots*. O CRL-OPS tem o seu próprio meio de acesso e, portanto não dispara os demónios *if-read*. Desta forma foi necessário usar regras separadas para a recolha de informação, as quais fazem o acesso explícito aos *slots* no lado direito (parte de acção).

As regras Prolog são compiladas para Lisp e o acesso aos *frames* transformado em chamadas às funções CRL. Para o exemplo anterior:

```
(call (schemap <Acção>))
(call (slotp <Acção> 'comando))
(bind (:list pegar (cons :list (get-values <Acção> 'comando)))
(call (slotp <Acção> 'efeito))
(bind (:list objecto-seguro) (cons :list (get-values <Acção> 'efeito)))
(call (slotp <Acção> 'agente))
(bind (:list ?Ag)(cons :list (get-values <Acção> 'agente)) )
(call (schemap <Ag>))
(call (slotp <Ag> 'sensor-de-objecto))
(bind (:list off) (cons :list (get-values <Ag> 'sensor-de-objecto)))
```

A chamada (get-values <Ag> 'sensor-de-objecto) provoca o disparo do demónio *if-read* associado ao *slot* sensor-de-objecto.

Alguns trabalhos contemporâneos sobre esta questão da monitoração e recuperação de erros podem ser encontrados em [Gin86], [HarBarLee86], e [MeiDui...86].

Todos estes trabalhos seguem uma aproximação baseada em regras (monitoração e recuperação). Contudo estas abordagens têm seguido um figurino muito pouco integrado. Isto é, o problema não tem sido analisado no contexto dum sistema integrado de programação e controle.

Por exemplo, no sistema de M. Gini, parte-se dum programa VAL. Esse programa tem de ser manualmente acrescentado, pelo programador, de algumas asserções especificando o conhecimento da tarefa de forma a que o sistema de monitoração possa julgar do sucesso ou insucesso da operação.

Outra diferença entre tais aproximações e esta proposta está na estrutura hierárquica - elas apenas contemplam um nível, o que, aliás, é consequência natural da utilização dum único nível para a descrição da tarefa (VAL, por exemplo).

Estes trabalhos têm, porém, o mérito de terem avançado alguns contributos na caracterização das situações típicas de excepção e proposição de algumas abordagens para a recuperação. Por exemplo, do trabalho da Universidade de Amsterdam [Mei86], [Mei88], resultou uma classificação de situações de excepção para estações de montagem, que se apresenta na tabela 3.2.2.

Grupos de excepções:	
Colisão	-Colisão com objecto desconhecido
	-Colisão com objecto desconhecido na posição P
	-Colisão com objecto (conhecido) O na posição P
Erro com objecto	-Objecto perdido em posição desconhecida
	-Objecto perdido na posição P
	-Deslocação de objecto na garra
Objectivo não atingido	-objectivo não atingido em tempo
Obstrução por força	-Obstrução devida a objecto desconhecido
	-Obstrução devida a objecto desconhecido na posição P
	-Obstrução devida a objecto O na posição P
Erro na garra	-Garra contendo objecto desconhecido
	-Garra contendo objecto O

Tab. 3.2.2 Exemplo de classificação de excepções

Esta tabela é aqui apresentada a título de exemplo, contudo algumas questões podem ser colocadas. Em primeiro lugar está longe de ser completa. Em segundo lugar põe-se a questão de representar ou não um conjunto de situações "realistas" no sentido em que a estação tenha capacidade para as detectar.

Recuperação

Após a detecção e diagnóstico (classificação) duma situação de excepção o controle é passado ao módulo de recuperação que tentará encontrar um método para ultrapassar a crise.

Na implementação efectuada, um módulo tendo por objectivo a geração de planos de recuperação, foi desenvolvido em CRL-Prolog. Esse planeamento é baseado num conjunto de regras onde se podem indicar várias estratégias alternativas para o tratamento de cada excepção.

Exemplo:

```
(recuperar ?Erro) < (seleccionar-op-recuperação)
                        (plan-recuper ?Erro)   ;gera plano de recuperação
                        (exec-op op-recuperação)   ;passa controle ao supervisor
                                                    para execução desse plano

;Regras de recuperação

(plan-recuper falha-preensão) < (:schema ...)   ;estratégia 1
                                (localizar-por-visão ?Peça)
                                (aproximação-preensão ?Peça ?ref-apr)
                                (mover-para ?ref-apr)
                                (pegar ?Peça)
(plan-recuper falha-preensão) < ...                    ;estratégia 2
...
```

Para a geração do plano de recuperação faz-se apelo a conhecimento usado pelo planeador genérico. No exemplo, a estratégia 1 para a falha-preensão utiliza as regras "localizar-por-visão", "mover-para" e "pegar" que também foram usadas durante a geração do plano inicial.

Como se verifica, a questão da recuperação está estritamente ligada à capacidade de classificar situações e, logo, à capacidade de observação. Ou seja, o sistema deverá dispor de regras de recuperação para cada tipo de excepção detectável. A criação das bases de regras tem de ser feita na "instalação do sistema" por um "programador de sistemas", visto depender da estação concreta.

A representação dos métodos de recuperação sob a forma de regras permite um sistema incremental onde é fácil adicionar novo conhecimento sem alterações na arquitectura básica, o que é conhecido como uma das vantagens da modularidade dos sistemas de regras e se revela bastante útil durante a fase de investigação de modelos.

Se todas as estratégias aplicáveis ao caso corrente falharem, o problema é passado para o próximo nível de monitoração. Na implementação corrente, que apenas considera 2 níveis, o controle é passado para o operador humano que deve introduzir um comando manual (eventualmente suspender a execução).

Para além da recuperação podia-se ter replaneamento (quando aquela falha) o que equivale a recuperar no nível acima. Isto é mais complicado pois o "*backtrack* " já não funciona (o planeamento foi feito em *off-line* !).

3.2.5 - MONITORAÇÃO DE SISTEMA

O assegurar do "bom funcionamento" dos diversos componentes da estação é uma das condições para o sucesso da execução das tarefas. Deste modo, a autonomia e flexibilidade do sistema passa, também, por uma capacidade de monitoração do comportamento da estação e seus componentes. Embora tal monitoração não esteja dependente da tarefa, como era o caso da monitoração de execução, pode ter consequências sobre a execução dessa tarefa. Este é, porém, um assunto bastante vasto e fora do âmbito deste trabalho pelo que apenas se fazem algumas considerações sobre a sua integração na arquitectura.

Implicações no modelo da estação

Um modelo de "bom funcionamento" e caracterização das correspondentes condições de monitoração têm de ser especificados para cada elemento da estação. Uma das questões importantes consiste, então, em determinar que informação complementar deve ser incluída no modelo da estação de forma a representar o seu estado de "bom funcionamento".

Adicionalmente há que garantir a necessária capacidade sensorial. Só é possível monitorar o comportamento dos equipamentos se houver capacidade para observar os seus parâmetros característicos (variáveis seleccionadas para caracterizar o estado do equipamento). Por exemplo, observação de:

-Consumo de energia; este parâmetro está também relacionado com a tarefa em execução, visto depender do esforço solicitado.

-Quebra de utensílio - broca, peça de corte em geral, chave de parafusos, etc.

-Vibrações.

-Temperatura.

-Erro operativo - quando o próprio controlador assinala erro ao executar uma operação (o que é detectado pelo monitor de execução).

Como representar esta informação? Um ponto de partida simplificado pode ser a indicação de alguns valores limite (ou condições de alarme).

Em caso de desvio em relação a tais limites, um processo de diagnóstico caracterizará o problema e um procedimento de recuperação ou manutenção pode ser activado.

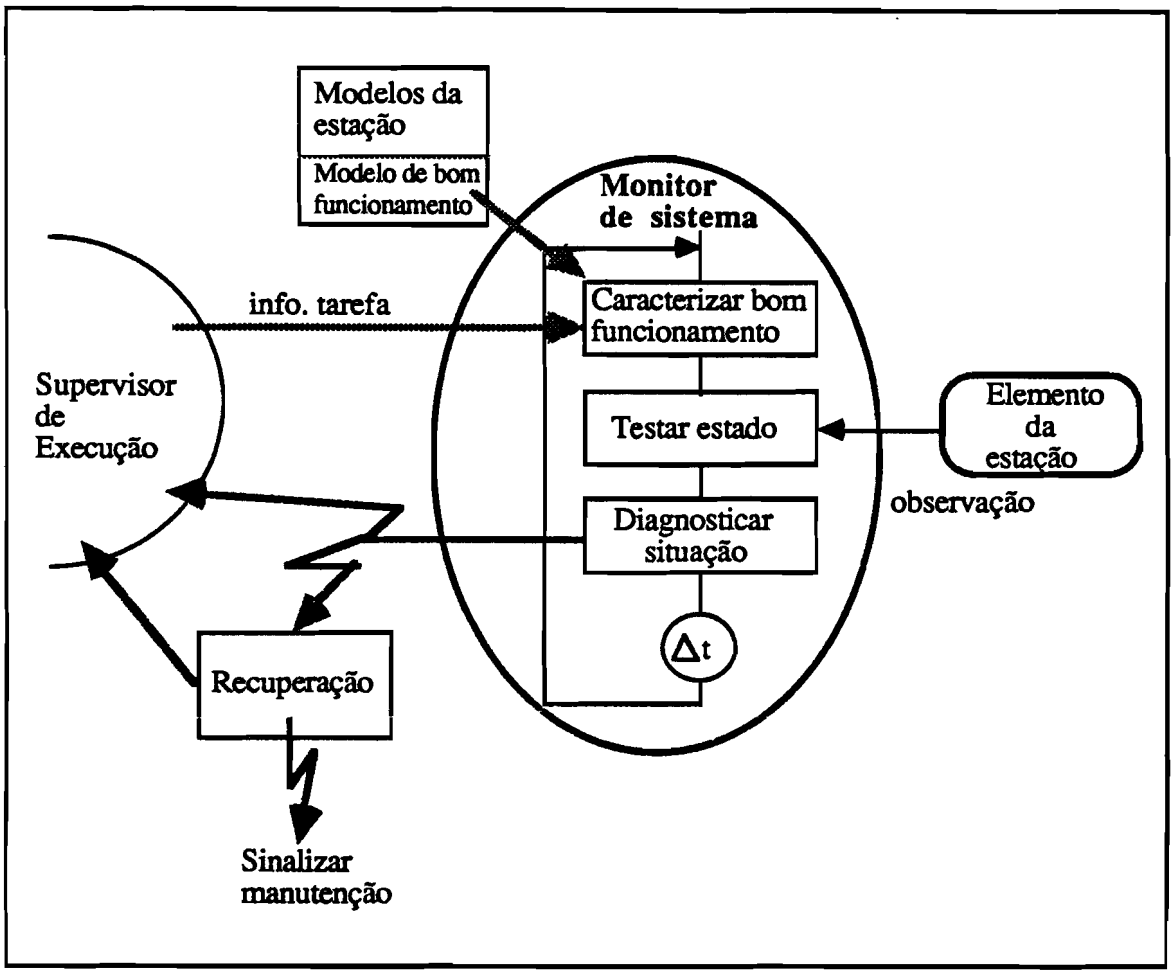


Fig. 3.2.15 Monitoração de sistema e relação com supervisor de execução

Isto terá efeitos na execução da tarefa. Todas as subtarefas afectas a um componente avariado têm de ser replaneadas ou reescalonadas (redistribuição da carga de trabalho se possível). Duas aproximações podem ser consideradas: ou soluções alternativas (em termos de máquinas e ferramentas) foram geradas durante o planeamento de sistema, ou o subsistema de recuperação tem de ter acesso ao conhecimento necessário para raciocínio sobre recursos (por exemplo, alguns conceitos usados na área de tecnologia de grupo).

No campo da Inteligência Artificial muitos sistemas de diagnóstico têm sido construídos [Mil87]. Contudo a quase totalidade desses sistemas requer a intervenção

dum operador humano para a introdução de dados. Um passo importante, e que constitui já uma tendência emergente, é o da conexão dos sistemas de diagnóstico com os instrumentos de teste e medida. Em suporte desta direcção tem-se o facto de a instrumentação computorizada ou o uso de computadores para recolha e registo de dados se ter tornado comum. Complementarmente tem-se também o surgimento, em pequenos equipamentos, de ferramentas para o desenvolvimento de sistemas baseados em conhecimento.

Como consequência duma tal conexão *on-line*, dois aspectos assumem especial importância:

i. Tempo real. Porque os valores dos parâmetros observados mudam ao longo do tempo, há que tentar garantir que os valores em uso, num dado momento, não estão obsoletos e, por outro lado, evitar que uma frequência muito elevada das observações se torne um factor de ineficiência do sistema. A determinação e o assegurar dos intervalos de observação adequados para cada caso é, pois, um ponto crítico.

Por outro lado, em geral, interessa ter uma noção das tendências de evolução dos valores no tempo e não apenas conhecimento dos valores actuais. Coloca-se, pois, uma questão de encontrar representações adequadas para a "história" dos parâmetros observados.

ii. Informação imprecisa. A informação fornecida pelos sistemas sensoriais é geralmente imprecisa. Por outro lado, o tipo de raciocínio usado em diagnóstico é frequentemente de tipo qualitativo, usando conceitos "fluídos" (*fuzzy*). Por exemplo, pode-se pretender saber o valor dum parâmetro em termos de "elevado", "normal" ou "baixo". Há, então, que proceder a uma "classificação" das medições efectuadas em termos destes valores qualitativos, o que passa pelo estabelecimento de adequadas funções de pertença (*fuzzy sets* e *membership functions*, ver [Zad83], [GraJon88]). Correspondentemente, o sistema de inferência a usar nestes casos deverá suportar raciocínio impreciso.

Várias estratégias têm sido desenvolvidas para a detecção dos componentes em falta e caracterização dessa falta. Uma classificação possível dos vários métodos [PasAnt88] [O'haBlaCon87], distingue:

i.Aproximações baseadas em conhecimento superficial (*shallow knowledge*), onde as associações entre sintomas e as situações de mau funcionamento são feitas de forma empírica, usando regras para representar o conhecimento empírico dum especialista em diagnóstico.

ii. Aproximações baseadas em conhecimento profundo (*deep knowledge*), em que as leis físicas e os princípios subjacentes ao sistema em observação, isto é, o

comportamento de cada componente e as relações entre subcomponentes, são modelados na base de conhecimento.

iii. Aproximações mistas, combinando as duas anteriores.

Implicações na arquitectura

Em termos da arquitectura global, uma forte relação (ou mesmo sobreposição) entre os módulos de recuperação e planeamento (planeamento de sistema e programação interactiva) pode ser identificada, nomeadamente no que se refere a aspectos de raciocínio sobre recursos. Contudo, deve voltar a referir-se que, embora abordem alguns problemas comuns, há algumas diferenças importantes. Os módulos de planeamento têm mais graus de liberdade (amplitude mais vasta) e são supostos produzir a solução "normal" que, em princípio, é optimizada. Os módulos de recuperação lidam com situações de excepção e, portanto, tomam decisões para um horizonte temporal mais curto (soluções transitórias ou de recurso). Os módulos de recuperação podem usar um subconjunto dos mecanismos de raciocínio e conhecimento disponíveis para os módulos de planeamento mas, devido à diferença de amplitudes faz sentido considerar uma separação funcional em diferentes blocos.

Também aqui se poderá ter uma aproximação hierárquica: para além da monitoração de componentes é necessário conhecer quais são as relações entre componentes (organização da estação) e monitorar agregados de componentes. Por vezes, uma falha num componente leva a que todo o sistema fique inoperacional. Por outro lado, cada componente pode, por si, funcionar bem (tanto quanto se consegue detectar) e, no entanto, o conjunto falhar.

Isto deve reflectir-se no modelo da estação - há que estabelecer as dependências funcionais entre componentes e conectar tais representações com os modelos de bom comportamento que se venham a estabelecer.

Prognóstico

Um tópico importante, especialmente relacionado com a perspectiva de monitoração de sistema, é a capacidade de previsão de futuros problemas com base em indicações do sistema sensorial e resultados históricos (prognóstico). Por exemplo, informação acerca de vibrações, consumo de energia, temperatura ou estado da peça de corte numa máquina CN poderão fornecer uma pista para a detecção duma situação de degradação progressiva.

Medidas correctivas baseadas em heurísticas poderão ser tomadas de forma a minimizar ou atrasar tais problemas.

O prognóstico permite um controle de qualidade antecipativo visto que o problema é detectado (suspeita) num estádio inicial antes de deterioração do produto.

Poucos são ainda os resultados nesta área, mas alguns trabalhos estão em curso. Exemplos ligados ao problema da manutenção preventiva de equipamentos podem ser encontrados em [O'haBlaCon87] e [MilMaj87].

No Grupo de Robótica da UNL este assunto constitui parte da nossa participação num projecto ESPRIT II - CIM-PLATO [SteCamFer88a].

3.2.6 - SIMULAÇÃO

Para além do papel no auxílio à programação gráfica interactiva discutido no cap. 3.1.7, a principal missão da simulação é a de possibilitar um teste preliminar - que se deseja tão completo quanto possível - dos programas antes de passar à execução na estação real.

Múltiplos níveis

Na perspectiva duma representação do plano a múltiplos níveis de abstracção tem sentido que a simulação possa também ser feita a múltiplos níveis.

No nível mais baixo deve-se ter mais "realismo" nos modelos da estação e produto simulados, enquanto nos níveis mais elevados bastarão representações mais abstractas e, portanto, requerendo menos detalhe.

Por outras palavras, simular um plano de nível de abstracção elevado num simulador que apenas fornece um nível de representação da estação e produto bastante realista, não tem muito sentido, até porque nesse nível do plano não se dispõe da informação de detalhe necessária para activar tal simulador. Para os níveis abstractos faz sentido uma simulação a um nível mais simbólico.

Em termos de desenvolvimentos, o nível mais baixo, onde se pretende uma emulação tão próxima quanto possível dos controladores da estação real, tem sido o mais enfatizado. Vários protótipos deste nível têm sido produzidos [Lau88], [WörSta87], [DilHuc85]. Em termos de mercado também alguns sistemas bastante sofisticados são oferecidos: ROBCAD [Adl86], MRS [McD85].

Um dos problemas com que se debatem tais simuladores é o da multiplicidade de linguagens (e de controladores / componentes) de células, em consequência da falta de standards. Vários simuladores apresentam-se "ligados" a uma linguagem específica. Exemplos: simulador da KUKA [WörSta87] e ROSI [DilHuc85] - linguagem SRL, ISR / LIFIA, Grenoble - linguagem LM [Lau88], EMULA / IBM - linguagem VAL. A solução oferecida pelo ROBCAD consiste na utilização duma linguagem interna - TDL (*Task Description Language*) - em que se deverá programar a tarefa e que será interpretada pelo simulador. Após o teste do programa, procede-se a uma tradução para a linguagem concreta do robô e a um carregamento no respectivo controlador.

É de voltar a referir uma diferença básica entre esta aproximação e a que se defende na presente proposta. No ROBCAD, tal como noutros simuladores, não existe o conceito de supervisão hierárquica ou de controlador de estação e, portanto, após a tradução o programa é "entregue" aos controladores dos componentes esperando-se que tudo "corra bem". Na nossa aproximação defende-se uma interacção entre os controladores dos componentes e o nível acima (controlador de estação) de forma a permitir monitorar a execução, isto é, os controladores não são "carregados" com programas para execução "independente", mas recebem comando a comando de forma interactiva.

Outro problema com a simulação resulta de a realidade apresentar sempre discrepâncias em relação aos modelos: posicionamentos, tolerâncias, precisões de movimentos, etc.. Mesmo dois robôs do mesmo fabricante e modelo apresentam algumas diferenças de comportamento. Então, antes da passagem do programa testado / desenvolvido com base na simulação / interacção gráfica para a estação real, é necessário proceder a uma **calibração / ajustamento** com base nos dados dessa estação. Alguns simuladores integram módulos, como o ADJUST no MRS, que ajudam a fazer uma adaptação, de forma automática, dos parâmetros usados na simulação (modelo da estação) para os valores reais da estação concreta.

Sistemas em desenvolvimento no IPK, Berlin e na Renault Automation [Spu...87] também tratam este problema. O próprio robô é usado como instrumento de medida para fazer o levantamento dos valores reais. As leituras efectuadas servem depois para compensar (calibrar) automaticamente os programas desenvolvidos em *off-line*. De notar que se trata de pequenos ajustes e, por isso, a tarefa pode ser automatizada. Se os desvios fossem muito grandes já teriam outras implicações (por exemplo, exigindo replaneamento de trajectórias).

Em relação a níveis mais abstractos, para além de alguns protótipos de investigação, não existe ainda qualquer normalização nem produtos comerciais.

Integração dos múltiplos níveis num sistema de programação é algo que desconhecemos e constitui uma das linhas de trabalho na UNL já referida [Cam88a].

Relação com sistema executivo

Como se defendeu no cap. 2.1, a integração da simulação no sistema de programação / controle deve passar pela utilização da mesma arquitectura e, até onde possível, dos mesmos módulos de supervisão de execução.

Apenas a conexão aos agentes reais é "desviada" para os agentes simulados.

No protótipo desenvolvido recorreu-se à noção de contexto do Knowledge Craft para conseguir uma fácil integração das perspectivas de execução real e simulada (fig.3.2.13).

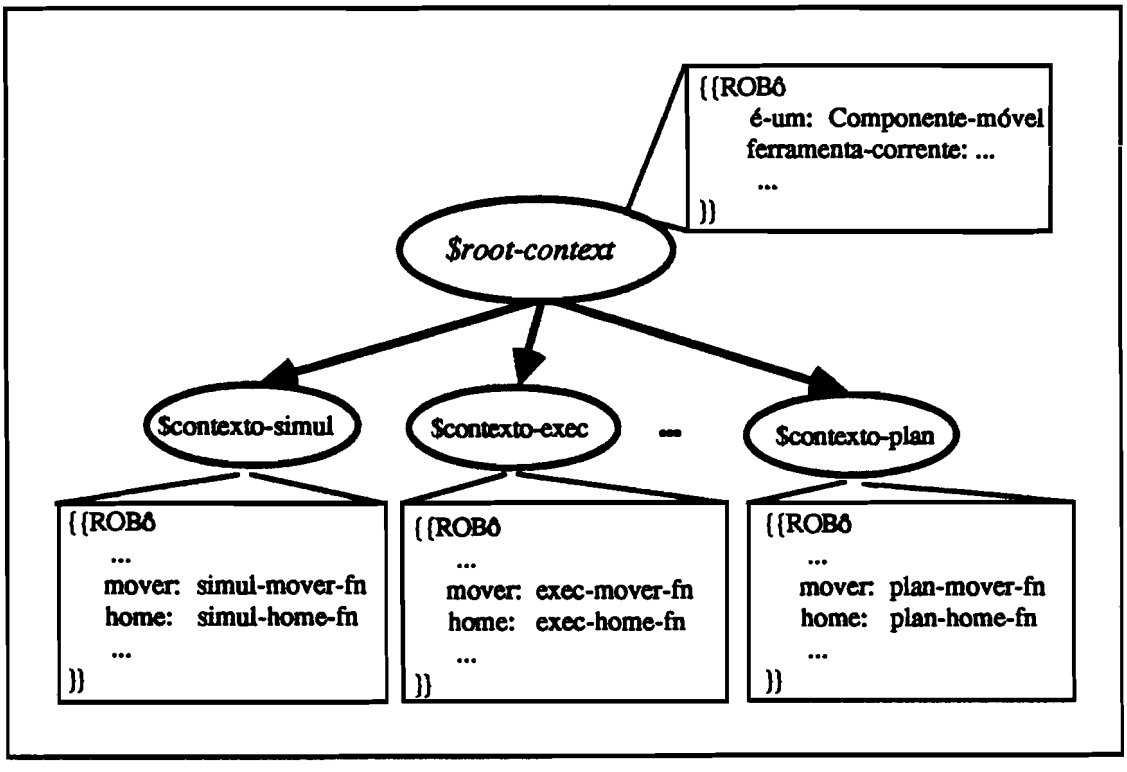


Fig. 3.2.16 Utilização de contextos para suporte do modelo dinâmico do mundo durante as fases de planeamento, simulação e execução.

Nesta abordagem, uma simples comutação de contexto corrente permite a conexão aos elementos simulados ou reais.

Ex.: (assert-context '$contexto-simul)

A fig. 3.2.17 mostra um exemplo de execução simulada (a nível simbólico).

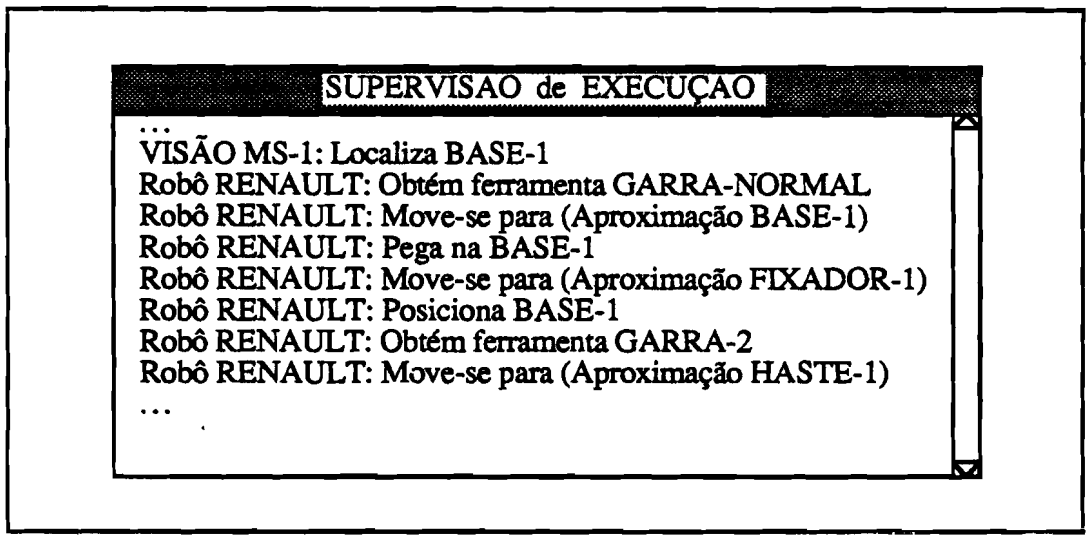


Fig. 3.2.17 Extracto da execução (simulada) do plano genérico para o teste-padrão de Cranfield

A integração da simulação no sistema pode, em certos casos, ser feita a um nível de abstracção mais baixo quando os controladores são decomponíveis em camadas. Na fig. 3.2.18 tem-se um exemplo em que se toma partido da arquitectura modular do controlador LM.

Neste caso, o simulador aceita uma linguagem de baixo nível com controle a nível das juntas, tal como o robô físico. Desta forma, o controlador LM e, portanto, o modelo do robô, é partilhado quer pelo robô real quer pelo robô simulado.

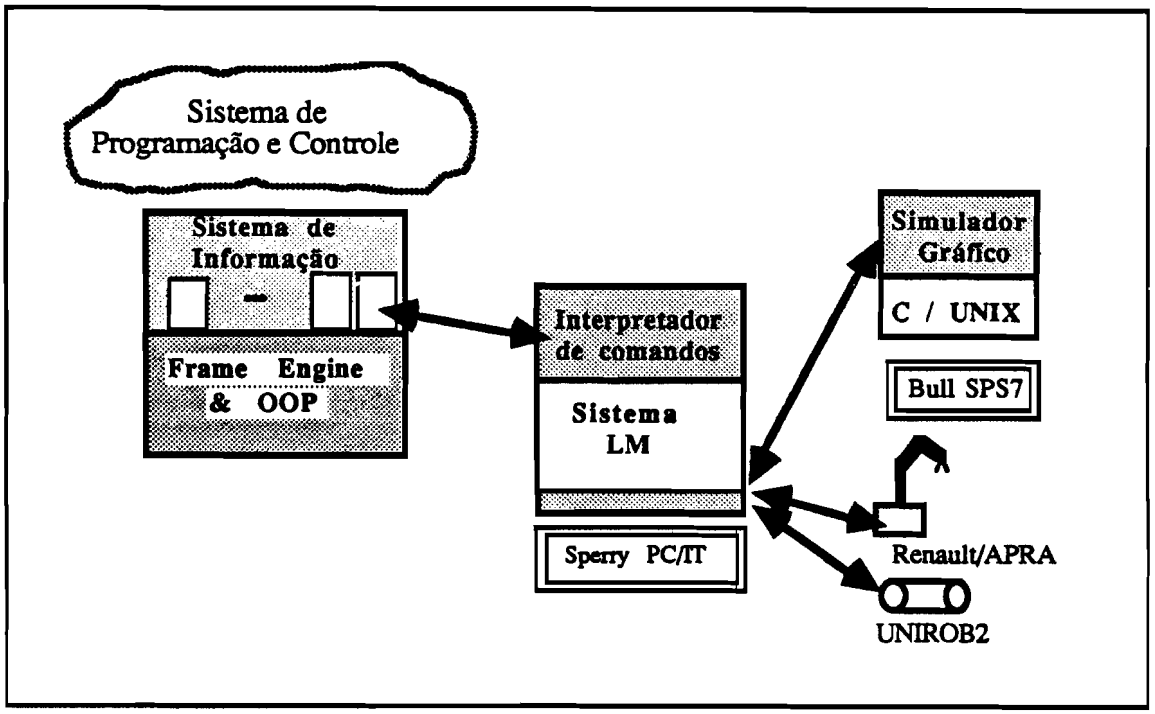


Fig. 3.2.18 Integração da simulação gráfica ao nível mais baixo do controlador LM

Simulação Sensorial

A simulação sensorial constitui um dos elementos importantes para o teste dos programas numa arquitectura fortemente baseada em realimentação.

Um primeiro nível, sugerido em [CamSte86a] e explorado em [SteSanQue87], consiste em gerar, a partir dos modelos CAD, informação equivalente à adquirida pelos sensores físicos - geração de imagens sintéticas, por exemplo. Essa informação é depois passada aos módulos de processamento usados com os sensores reais.

A um nível mais abstracto pode-se simular a entrada de informação já depois do seu processamento básico. Sendo a aquisição de informação sensorial realizada via programação reactiva, como se descreveu atrás, através do mecanismo de contextos pode-se ter um contexto de simulação em que a interrogação é feita ao operador e não directamente ao sensor (fig.3.2.16).

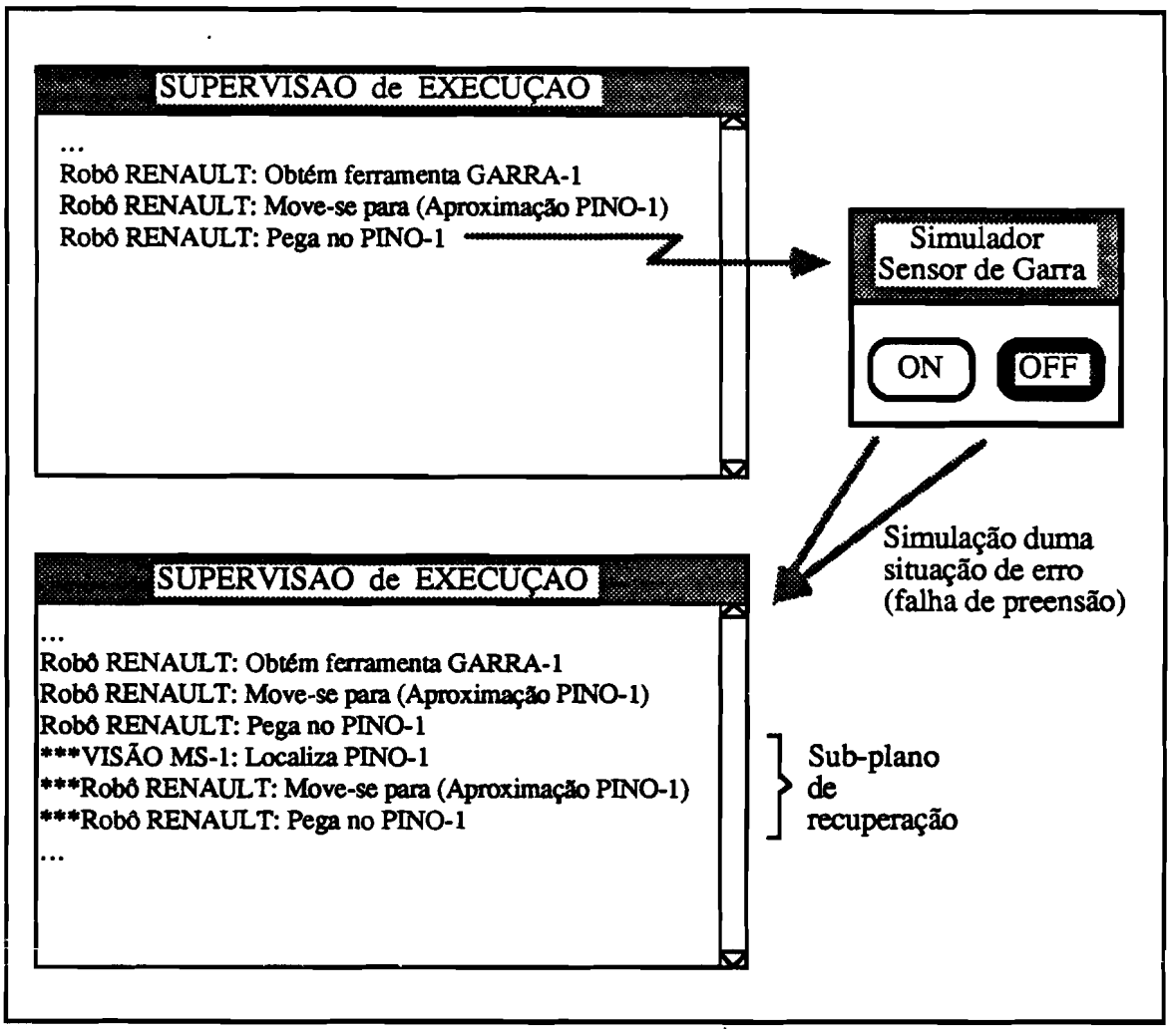


Fig. 3.2.19 Visualização dum exemplo de monitoração de execução com entrada simulada de informação sensorial

Algumas facilidades adicionais para a introdução assíncrona de tal informação podem ser conseguidas com o mecanismo de imagens activas, como se verá adiante.

Imagens activas

O conceito de imagem activa representa a possibilidade de associar gráficos a elementos de informação, simbolizando indicadores como alarmes, mostradores discretos, lineares (tipo "termómetro") ou angulares, etc., de tal forma que, sempre que o valor do objecto for alterado a correspondente representação gráfica é automaticamente actualizada.

Este mecanismo afigura-se interessante para o estabelecimento da interface com o humano nos níveis de simulação mais abstractos e também como complemento nos outros níveis.

Por exemplo, num nível de abstracção em que se pretenda simular a interacção entre agentes e respectivas condições de sincronização, fluxos de peças, etc., tal pode ser visualizado através duma representação gráfica mais abstracta (diagramas de barras, "lâmpadas", por exemplo).

Como complemento à simulação do nível mais baixo, pode servir para criar painéis de controle que representem os valores dos sensores simulados e, assim, permitir monitoração visual do estado de execução e do sistema.

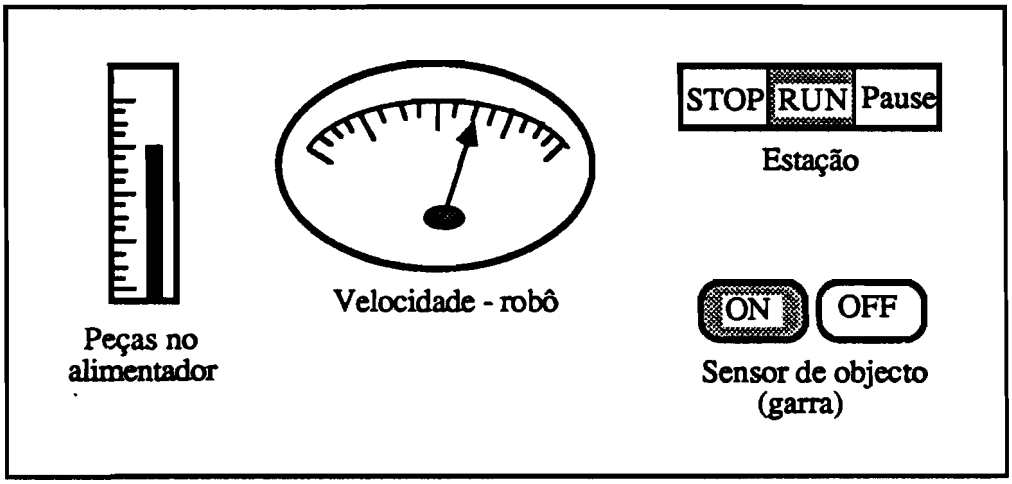


Fig. 3.2.20 Exemplos de utilização de imagens activas

É de notar que esta utilização para a construção de painéis de controle pode ser usada também na execução real.

Uma funcionalidade adicional é a possibilidade de utilizar as imagens activas como dispositivo de *input*. Apontando com o rato num dado valor, significa - se tal for permitido - que se pretende alterar o valor conforme o indicado. Esta é, por exemplo, uma forma de introduzir simulação sensorial ou de simular eventos externos. Nalguns sistemas esta faculdade admite assincronismo nos *inputs* (o sistema gere as filas de espera).

Uma implementação bastante flexível de imagens activas é fornecida pelo sistema KEE [Int86]. Na UNL foram realizadas algumas versões experimentais sobre o *frame*

engine Prolog e sobre o Knowledge Craft [CamCor...88] de forma a avaliar as potencialidades do mecanismo na robótica.

A programação reactiva revelou-se um adequado suporte às imagens activas sendo o gestor da imagem um caso particular de demónio (fig. 3.2.21).

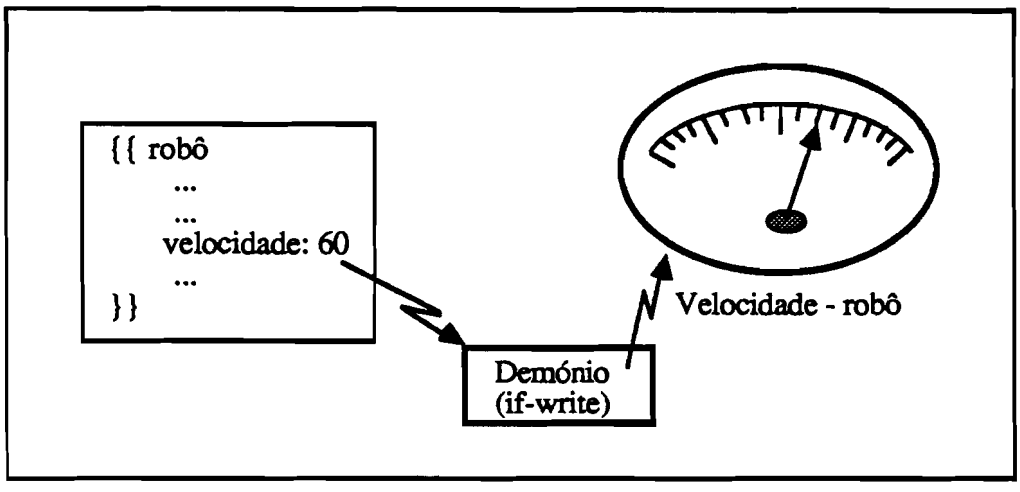


Fig. 3.2.21 Programação reactiva na implementação de imagens activas

Numa segunda fase passou-se à utilização do Graphpak, um pacote gráfico desenvolvido sobre o Knowledge Craft, surgido em meados de 1988 [Car88] e que inclui uma biblioteca de "aparelhos medidores / mostradores" e respectivas funões de entrada / saída. Sobre este sistema implementou-se o conceito de imagem activa, utilizando a programação reactiva de acordo com o exemplo da fig. 3.2.21. Na versão actual do Graphpak apenas é contemplado o *input* síncrono, o que é uma limitação para a construção de "painéis de controle" com múltiplos instrumentos. Deste modo foi também necessário re-implementar o método de *input* para permitir entrada assíncrona de dados [Cam89].

•••

Conclui-se assim a apresentação detalhada da proposta e discussão da experimentação realizada bem como de pontos em aberto. Como foi apontado por diversas vezes no texto, existem várias direcções que devem ser prosseguidas numa segunda fase. O próximo capítulo fará uma apresentação sucinta de tais linhas a desenvolver.

4. CONCLUSÕES E TRABALHO FUTURO

4.1- SÍNTESE DE RESULTADOS

4.2- TRABALHO FUTURO

4.1- SÍNTESE DE RESULTADOS

De acordo com o enunciado no capítulo 1, foi feita a apresentação e justificação duma proposta de arquitectura para um sistema de programação e controle de estações robóticas de montagem, primeiro numa perspectiva de modelo geral e depois numa abordagem mais detalhada.

Este trabalho não pretendia "resolver todos os problemas" do domínio seleccionado mas tão somente avançar alguns contributos no sentido de propor modelos que permitissem melhorar a compreensão e aumentar o grau de automatização das tarefas a realizar.

De entre os vários contributos / resultados podem-se salientar os seguintes:

i. *Arquitectura integrada*

Foi desenvolvida uma proposta de arquitectura integrada multinível para um sistema de programação e controle (SPC) de estações robóticas de montagem, tendo por directrizes principais uma ênfase nos aspectos de flexibilidade e a consideração do problema num contexto CIM.

Complementarmente produziram-se algumas contribuições para uma clarificação de interfaces entre diversas subáreas dum sistema CIM mais directamente relacionadas com a programação da estação: concepção do produto, planeamento de processo e concepção / selecção da estação / ferramentas.

ii. *Sistema de Informação como elemento integrador*

O Sistema de Informação (SI) foi identificado e fortemente sugerido como devendo ser a peça básica para a integração dos diferentes componentes do sistema de programação e controle e, também, como suporte da integração deste no contexto mais geral dum sistema de produção integrada por computador.

Uma estrutura de SI, identificando os seus componentes e fluxos de informação, foi estabelecida, bem assim como a sua relação com a arquitectura funcional do SPC.

A abordagem adoptada seguiu uma aproximação baseada em conhecimento o que, neste domínio, também constituiu uma das primeiras propostas em termos de sistema global.

iii. *Demonstração de factibilidade*

A realização de vários protótipos relativamente a "pontos" determinantes da proposta permitiu evidenciar a factibilidade e consistência geral da mesma, bem assim como a produção de alguns contributos originais para alguns tópicos mais específicos. De entre estes, são de referir:

-Implementação dum ambiente integrado de desenvolvimento, suportado por:

.uma integração interactiva ou fortemente ligada de múltiplas fontes de informação (CAD, BD, BC)

.integração de controladores dos componentes reais da estação ou respectiva simulação

.subsistema de processamento de referenciais

e utilizando múltiplos paradigmas de programação / representação de conhecimento: "frames", programação reactiva, orientada por objectos e baseada em regras.

-Geração de planos segundo uma filosofia hierárquica e no contexto integrado.

-Implementação de um protótipo parcial baseado numa concepção de arquitectura multinível de supervisão de execução / monitoração com capacidade de detecção e recuperação de situações de excepção.

É, todavia, de referir que não se procurou a implementação dum sistema completo, por tal estar fora do âmbito deste trabalho dados os enormes recursos humanos e materiais que requer, não só em termos de investigação como também de engenharia / desenvolvimento. No entanto, através de implementações seleccionadas, procurou demonstrar-se a factibilidade dos elementos críticos preponderantes.

iv. *Publicações*

O trabalho desenvolvido permitiu também submeter, de modo regular, à comunidade científica nacional e internacional - através de simpósios, conferências e revistas - as propostas e resultados produzidos, conforme documenta a bibliografia apresentada.

v. *Abertura de novos caminhos*

O trabalho realizado, pela sua natureza, evidenciou múltiplas pistas para continuação da investigação, o que constitui também um resultado importante.

vi. *Contributos para o ensino*

A actividade desenvolvida teve também os seus reflexos no ensino da Robótica na UNL. Houve assim contribuições para o estabelecimento dum plano de introdução destas matérias na Licenciatura em Engenharia Informática [SteCamMou87]. Na fase de implementação desse plano houve especial contribuição nas Cadeiras de Complementos de Robótica [CamMou...88] e Projecto de Robótica, onde foi possível estabelecer vários trabalhos experimentais / projectos baseados nos desenvolvimentos desta proposta. Complementarmente houve também contributos para a formação de diversos estagiários no Grupo de Robótica Inteligente da UNL / UNINOVA.

4.2- TRABALHO FUTURO

Generalizações

Várias das linhas de possível trabalho futuro têm a ver com a generalização dos aspectos abordados. Nalguns casos apenas foram lançadas pistas ou enunciadas questões. Noutros resolveram-se problemas dentro de condicionantes restritivas.

Embora não pretendendo fazer um levantamento exaustivo das possíveis áreas de investigação em Robótica / CIM, apresentam-se algumas linhas que podem ser derivadas directamente do trabalho atrás descrito e que parecem bastante promissoras ou correspondem mesmo a novas actividades já iniciadas ou de algum modo planeadas.

i. Graus de automatização

Algum esforço deve ser colocado na generalização dos protótipos no sentido de progressivamente reduzir a necessidade de interacção com o especialista humano, ou de alargar a sua zona de aplicabilidade. Em certos tópicos alguns aprofundamentos a nível conceptual são também necessários. Por exemplo, há que:

-Generalizar a análise dos problemas duma estação multiagente relativamente ao planeamento e controle de actividades concorrentes. É de notar que, embora o problema tenha sido colocado no contexto duma estação, os casos estudados foram fundamentalmente centrados em situações dum só robô como elemento central.

-Aprofundar as questões relativas a raciocínio sobre recursos, que estão relacionadas com o ponto anterior e que, nos desenvolvimentos realizados, foram, em grande parte, deixadas para o nível de planeamento de sistema.

-Generalizar a abordagem das questões de monitoração de execução, particularmente no que se refere às estratégias de recuperação.

-Introduzir em mais profundidade as questões da modelação e raciocínio com conhecimento impreciso.

-Prosseguir os trabalhos sobre simulação e especificação gráfica interactiva de tarefas.

Obviamente que estes esforços deverão ter em atenção os desenvolvimentos que estão ocorrendo, em paralelo, noutros centros de investigação ou noutras linhas de acção do próprio Grupo de Robótica da UNL, particularmente no que se refere ao domínio dos subsistemas percepcionais e de planeamento especializado.

ii. Prognóstico

A questão da monitoração do comportamento da estação foi só superficialmente abordada no trabalho a pretexto de estabelecer algumas interacções com a monitoração de execução.

Embora, como então se referiu, muitos trabalhos tenham vindo a ser desenvolvidos no campo do diagnóstico de avarias, a conexão *on-line* de tais sistemas levanta questões ainda não abordadas em profundidade. A detecção antecipada de futuros problemas constitui um importante campo de investigação. Importa aqui compreender melhor os modelos de diagnóstico, antecipação e seu relacionamento com a arquitectura geral do sistema de programação e controle. Por exemplo, a questão da representação do tempo terá de ser analisada visto que o prognóstico não é baseado apenas na observação corrente de certos parâmetros mas também na tendência da evolução desses parâmetros.

Por outro lado, os aspectos de raciocínio sobre informação imprecisa e incompleta parecem aqui determinantes, devendo os seus impactos no Sistema de Informação ser analisados.

iii. Domínio

A análise efectuada foi dirigida para estações de montagem. Outra possível extensão ao trabalho consiste em avaliar e adaptar os modelos desenvolvidos para outros domínios de aplicação.

Alguns dos problemas são, como se referiu no início, semelhantes mas alguns aspectos específicos implicarão desenvolvimentos mais aprofundados em certas direcções. Por exemplo, um dos problemas a ter mais desenvolvimento na soldadura é a definição de trajectórias. Por outro lado, o conhecimento de carácter tecnológico é aí determinante para o planeamento do processo.

Esta generalização deve incluir o controle de estações com outro tipo de equipamentos. Por exemplo, estações de maquinagem onde o robô pode interactuar com máquinas de controle numérico. A ligação CAD-CAM já tem sido explorada no sentido de permitir passar de forma quase automática do modelo da peça para o controle das máquinas que a fabricarão, mas num contexto simples e num só sentido. Importará agora uma nova abordagem no contexto de estações mais complexas e flexíveis, no que respeita a integração de informação, controle de execução e monitoração.

A dimensão do domínio de investigação exige meios e disponibilidades importantes. Durante o período que decorreu foi possível perspectivar e criar condições para o futuro. Assim, a nossa participação nos Projectos CIM-PLATO e B-LEARN do programa Esprit fornece um enquadramento positivo para várias destas generalizações [SteCamFer88].

O objectivo mais global é o de caminhar para a produção de sistemas geradores de subsistemas CIM. Nesta direcção, a nossa contribuição passa pela concepção e desenvolvimento de algumas ferramentas genéricas para:

-desenvolvimento de sistemas de informação em sistemas flexíveis de fabricação (FMS/FAS - *Flexible Manufacturing Systems / Flexible Assembly Systems*), considerando estações complexas multiagente incluindo robôs, máquinas CNC e outros periféricos;

-aquisição e representação de conhecimento impreciso a vários níveis do sistema de produção e incluindo o suporte a actividades de prognóstico.

Grande parte da abordagem realizada foi feita de forma informal. Tal parecia inevitável nesta fase dada a grande complexidade do problema e o seu ainda limitado conhecimento. Numa primeira fase tornou-se necessário compreender as questões envolvidas. Após mais algum avanço será importante proceder a uma tentativa de formalização de forma a consolidar o conhecimento adquirido e partir duma síntese para novas etapas.

Aprendizagem

No trabalho desenvolvido foi posta alguma ênfase na concepção de sistemas flexíveis, com capacidade de adaptação e resolução de situações de excepção. Isto é particularmente patente nos aspectos de monitoração. Todavia, o comportamento de tais

sistemas é determinado pelo conhecimento implantado à partida, não se podendo falar em evolução das suas capacidades de decisão. Por outras palavras, o sistema de monitoração apresentado não é capaz de aprender com as suas experiências. Quando no futuro for confrontado com situações análogas, não revelará qualquer melhoria na sua capacidade de resolução da crise [SteCam88].

A monitoração relaciona-se com a adaptação a situações variáveis ou desconhecidas à priori, enquanto a aprendizagem pretende suportar uma melhoria ou refinamento automático do comportamento do sistema baseado na experiência adquirida. A monitoração pode suportar a flexibilidade mas não suporta uma progressiva melhoria da capacidade de decisão do sistema. A introdução de mecanismos de aprendizagem permitirá passar dum sistema adaptativo para um sistema evolutivo.

É, porém, de notar que existe algo em comum nos dois mecanismos. Similarmente à monitoração, a aprendizagem baseia-se na realimentação. Após uma decisão ter sido tomada, há que fazer uma observação dos resultados da sua aplicação seguida dum criticismo ou avaliação. De certo modo, um processo de aprendizagem pode ser visto como uma espécie de meta-monitoração dum sistema de tomada de decisões baseado em conhecimento.

A questão da aprendizagem tem sido, desde há muito, um dos tópicos de investigação fundamental. Nos últimos anos tem-se assistido a um renovado interesse pelo assunto, motivado pela esperança de que métodos genéricos de aprendizagem possam permitir a automatização da construção de sistemas fortemente baseados em conhecimento. Uma extensa panorâmica dos esforços desenvolvidos e resultados atingidos, normalmente em situações artificiais ou mundos simplificados, pode-se encontrar em [Mic...85] e [Mic...87]. Importa agora fazer um esforço de introdução e generalização de métodos para o caso de sistemas autónomos flexíveis operando em condições reais.

Há múltiplas possibilidades para a aplicação de mecanismos de aprendizagem num sistema robótico de molde a melhorar a sua eficiência e aumentar o seu grau de autonomia. Em princípio, processos de aprendizagem podem ser associados a cada bloco de decisão do sistema de programação e controle. Encontra-se, também, em aberto a questão da combinação de múltiplas estratégias de aprendizagem num mesmo sistema, o que representa um desafio do ponto de vista da compreensão dos mecanismos básicos a utilizar. Contudo a complexidade da questão sugere que apenas procedimentos semi-automáticos de aprendizagem (com ajuda humana) devam ser considerados numa primeira fase.

Em termos "temporais", a aprendizagem pode ser aplicada em dois estádios:

i-Para o carregamento inicial da base de conhecimento, por ensino. Uma das possibilidades passa pela indução de regras a partir dum conjunto preparado de exemplos. Há alguns algoritmos bem conhecidos para este tipo de indução [ThoTho86], [GraJon88], como o ID3 que aprende árvores de decisão eficientes para a classificação de objectos, mas apenas alguns consideram a questão da informação imprecisa, o que é determinante na robótica.

ii-Para o refinamento da base de conhecimento por observação e avaliação em tempo de execução. Esta é a tarefa mais complexa, mas é a faceta que materializa de facto os aspectos evolutivos.

A aquisição de destreza pode também ser útil na resolução de tarefas para as quais não há uma solução ou método explícito. Este é o caso de algumas operações complacentes, requerendo movimentos finos com forte realimentação de informação sensorial. A aquisição e representação do conhecimento comportamental deverá permitir a obtenção de descrições abstractas dessa destreza (não só aquisição mas também explicitação do conhecimento adquirido). A aprendizagem pode ser a base para um "ensino guiado" avançado (e não apenas um processo de memorização e repetição como nos métodos tradicionais de ensino por teclado de funções).

Em linhas gerais, o enquadramento para esta direcção de investigação encontra-se detalhado numa proposta de projecto de investigação (B-LEARN - *Acquisition of behavioral knowledge for autonomous systems operating in real environments*) no âmbito do programa ESPRIT Basic Research Actions, e onde o Grupo de Robótica Inteligente da UNL desempenha o papel de proponente coordenador [SteCam....88b].

A nossa actividade incidirá essencialmente sobre a introdução de mecanismos de aprendizagem no subsistema de monitoração. Para além dos objectivos gerais já enunciados em termos da procura de sistemas evolutivos, é de considerar que uma compreensão mais alargada dos mecanismos de monitoração e aprendizagem poderá conduzir a modelos mais genéricos e mais integradores para o sistema de programação e controle.

Planeamento de sistema

Como se deriva dos capítulos anteriores não é fácil, porventura nem aconselhável, estabelecer uma clara separação entre os níveis de programação genérica e de planeamento de processo.

Este último tem estado fora dos limites definidos para o trabalho aqui apresentado, contudo parece claro que saltos significativos no processo de automatização passarão pelo

grau de automatização e integração que se for conseguindo introduzir nas diversas actividades de planeamento de processo e concepção / selecção de estação.

Como se referiu em 3.1.4 (Plano documentado) (ver fig. 3.1.24 ???) uma tarefa pode ser especificada a múltiplos níveis de abstracção. A passagem dum nível para outro pode ser feita de forma explícita / interactiva ou automática. No trabalho descrito partiu-se dum dado nível, que se designou por solução implícita e referiu como o resultado do planeamento de sistema, e propôs-se uma solução no sentido descendente. Uma importante direcção de investigação no seguimento dos trabalhos realizados será, então, prosseguir na direcção da automatização dos níveis acima. A divisão feita não é a única e vários trabalhos têm sido desenvolvidos, quer da perspectiva da programação, quer do planeamento de sistema. A divisão em níveis facilitou a abordagem do problema, mas a certa altura é necessário voltar a encarar a questão num sentido mais geral - de certo modo um regresso à "utopia" dos primeiros sistemas de nível tarefa em termos de formulação do problema global.

Embora esta seja uma área onde muito trabalho está sendo realizado, parece que uma abordagem integrada na perspectiva que foi defendida se afigura bastante promissora comparativamente com outros trabalhos mais sectoriais.

Note-se, por exemplo, a importância que os avanços no nível do planeamento de processo têm para um incremento das capacidades de recuperação do sistema em caso de excepção. Noutra zona serão também de considerar contributos da área de programação gráfica interactiva quer para o planeamento de processo quer para a própria concepção do produto.

A nossa participação no projecto ibero-americano de "Automatización y Robótica Avanzada / CYTED", a par do projecto CIM-PLATO, fornecem o enquadramento para este desenvolvimento.

•••

Um requisito importante para a viabilidade da execução dos trabalhos esboçados é a existência duma equipa com competências em áreas diversificadas complementares e com grande capacidade de interacção. Está-se apontando para uma zona de forte interdisciplinaridade em que o ponto de maior potencial se encontra exactamente na possibilidade de interacção entre várias áreas de investigação.

A obtenção de tal objectivo passa por uma capacidade de gestão de equipa onde seja pretendido trabalho cooperante e não por sectores isolados. Embora isto constitua um requisito bastante difícil, a forma como os trabalhos descritos puderam ocorrer no Grupo

de Robótica Inteligente constitui uma experiência positiva e um estímulo para a sua continuação e aperfeiçoamento. O facto de o Grupo ter assegurado a sua participação em vários projectos internacionais fornece uma base sólida de meios, contactos e competição para um trabalho estimulante.

REFERÊNCIAS

[AbeSakTsu88]

N. Abe; S. Sako; S. Tsuji - High-level Task Specification for Robot
International Symposium and Exposition on Robotics (ISER / 19th ISIR), Sydney, Australia, 6-10 Nov. 1988.

[Adl86]

A. Adler - Robotics Workcell design, simulation and off-line programming
IEEE Int. Conference on Systems, Man and Cybernetics, Atlanta 1986.

[AdlRod87]

A. Adler; J. Rodriguez - CAD / Simulation of flexible manufacturing systems
TECNOMATIX GmbH, 1987.

[AkeBru87]

L. van Aken; H. van Brussel - A structured geometric database in an off-line robot programming system
Robotica (Cambridge University Press), Vol.5, pp333-339,1987.

[AkeBru...87]

L. van Aken; H. van Brussel; J. de Schutter; P. Simkens; F. de Meester - LOLA (Leuven Off-line LAnguage) - An enhanced manipulator level off-line robot programming system
in *Off-line programming of industrial robots*, A. Ston; J.F. McWaters (Ed.s), *Elsevier Science Publishers* (North-Holland) / IFIP 1987.

[AmbCamCor86]

A.P. Ambler; S.A. Cameron; D.F. Corner - Augmenting the RAPT Robot Language
Proceedings of *NATO Advanced Research Workshop on Languages for sensor-based control in Robotics*, Castelvecchio Pascoli, Italy, Sep 1-5, 1986. Editado pela *Springer-Verlag, Nato ASI Series N. 29*, 1987.

[And87]

R. Anderl - Application of Artificial Intelligence methods in Computer Aided Design Systems
Esprit / CIM Europe SIG workshop on AI methods and tools in CIM, Athens 28-30 Jan. 1987.

[Apa87]

J.N. Aparício - Concorrência na Programação em Lógica
Provas de aptidão pedagógica e capacidade científica, Universidade Nova de Lisboa, Departamento de Informática, Dez. 1987.

[AyrFun89]

R.U. Ayres; J.L. Funk - The role of machine sensing in CIM
Robotics & Computer Integrated Manufacturing, vol.5,n.1,1989.

[BarRosCamNev86]

A. Barbosa; P. Rosado; L.M. Camarinha Matos; José Afonso Neves - Installation of LM language on RHINO and Renault/Sirtés robots
Relatório UNL-DI 65/86; UNIROB RT-L2-57-86; Dez. 86.

[BasTor...88]

L. Basañez; C. Torras; A. Sanfeliu; J. Ilari - Automatic Cell Programming and Monitoring through the Cooperative Interplay of Operation Specialists
2nd International Symposium on Robotics and Manufacturing Research, Education and Applications, Albuquerque, USA, 16-18 Nov. 1988.

[Bra86]

British Robot Association - Robot Facts 1986
Dec 86.

[Bro83]

R.A. Brooks - Planning collision-free motions for pick-and-place operations
The International Journal of Robotics Research, vol.2, n.4, Winter 1983.

[BroFar..85]

L. Brownston; R. Farrel; E. Kant; N. Martin - Programming Expert Systems in OPS5 - An introduction to rule-based programming
Addison Wesley, 1985.

[BroMyl86]

M. L. Brodie; J. Mylopoulos (Ed.s) - On Knowledge Base Management Systems - Integrating AI and DB Technologies
Springer-Verlag, 1986.

[Byt87]

Byte - IGES: One answer to the problems of CAD database exchange
Byte, Jun 87.

[Cam83]

L.M. Camarinha Matos - Concorrência em Microcomputadores
Provas de aptidão pedagógica e capacidade científica, Universidade Nova de Lisboa, Departamento de Informática, Mar. 1983.
Editado como *Manual de Ensino* pelos *Serviços Gráficos da UNL*, Nov. 1983.

[Cam85]

L.M. Camarinha Matos - Sistema de Programação de Estações Robóticas - Proposta de Trabalho
Relatório UNL-DI 28/85; UNIROB RT-L2-14-85, Nov 1985.

[Cam86]

L.M. Camarinha Matos - Agentes e tarefas num sistema de programação de robôs - memorando interno
Relatório UNL-DI 24/86; UNIROB MI-L2-18-86, Mai 1986.

[Cam87a]

L.M. Camarinha-Matos - Plan Generation in Robotics - state of the art and perspectives
Relatório UNL-DI 18/85; UNIROB RT-L2-06-85, Jun 1985.
Revista ROBOTICS (North-Holland) vol.3, n. 3&4, 1987.

[Cam87b]

L.M. Camarinha Matos - Plano para o desenvolvimento dum "Frame Engine" em Prolog
Relatório UNL-DI 19/87; GR DI-DA-29-87; Fev-Abr 87.

[Cam88]

L.M. Camarinha Matos - Especificação gráfica interactiva de tarefas de montagem - plano de trabalho
Relatório UNL-DI 32/88; GR RT-DA-39-88; Jul 88.

[Cam89a]

L.M. Camarinha Matos - Imagens activas - Trabalho nº 5 de Complementos de Robótica, UNL, Jan.89.

[Cam89b]

L.M. Camarinha Matos - Information integration for the High Level Interpreter
Esprit 623 working paper EP-UNL-01.89/1, Jan. 89.

[CamBar87]

L. Camarinha; A. Barbosa - An experiment with object oriented programming for robot control

Relatório UNL-DI 22/87; GR-E623 RT-DA-32-87, Jul 87.

Documento ESPRIT 623 EP-UNL-07.87/1.

[CamCor88]

L.M. Camarinha Matos; L. Correia - Programação orientada por objectos no controle de estações robóticas

5º Congresso Português de Informática, Lisboa, Maio 1988.

[CamCorBar87]

L. Camarinha; L. Correia; A. Barbosa - CIM Information System: Rules for tool selection

Relatório UNL-DI 6/87; GR-E623 RT-DA-03-87; Jan. 87

Documento ESPRIT 623: EP-UNL-01.87/1.

[CamCor...88]

L.M. Camarinha Matos; L. Correia; J.M. Pires; D. Ferreira; J. Afonso - Complementos de Robótica - Notas de apoio

Relatório UNL-DI 04/88; GR RT-DA-03-88; Jan 88.

[CamFerMou88]

L.M. Camarinha Matos; D. Ferreira; J. Moura Pires - Integração de Sistemas de Gestão de Dados e Conhecimento para Modelação em Robótica

5º Congresso Português de Informática, Lisboa, Maio 1988.

[CamFerr...87]

L. Camarinha; D. Ferreira; J. Moura Pires; L. Correia; A. Barbosa - Market Survey on Data and Knowledge management tools - Comparative Analysis

Relatório UNL-DI 10/87; GR-E623 RT-DA-12-87, Fev. 87.

Documento ESPRIT 623: EP-UNL-02.87/1, Technical Meeting Universidade Nova de Lisboa, Mar 87.

[CamFer...88]

L.M. Camarinha-Matos; D. Ferreira; R. Rabelo; H.P. Pita - Knowledge Craft - Rdb Integration - a prototype contribution to the CIM IS -

Relatório UNL-DI 07/88; GR-E623 RT-DA-08-88, Fev 88.

Documento ESPRIT 623 EP-UNL-06.88/1.

[CamSte86a]

L.M. Camarinha-Matos; A. Steiger-Garção - Robotic Cell Programming: A Knowledge-based Approach
Relatório UNL-DI 9/86; UNIROB RT-L2-07-86, Mar 1986.
Proceedings of *8th IASTED / AFCET International Symposium on Robotics and Artificial Intelligence*, Toulouse 18-20 Jun 86.

[CamSte86b]

L.M. Camarinha Matos; A. Steiger Garção - Programação de Estações Robóticas a nível tarefa
Relatório UNL-DI 3/86; UNIROB RT-L2-02-86, Fev 1986.
Actas do *4º Congresso Português de Informática*, Lisboa 23-27 Junho 1986.

[CamSte87a]

L.M. Camarinha-Matos; A. Steiger-Garção - An Information System Architecture for Robot Cell Programming
Nato Advanced Study Institute on CIM: Current Status and Challenges, Istanbul, Turkey, Ago 31- Set 12, 1987 (Springer-Verlag, Nato ASI Series, 1988).

[CamSte87b]

L. M. Camarinha Matos; A. Steiger Garção - CAD no contexto duma estação robótica de montagem
1.as Jornadas Nacionais de Projecto, Planeamento e Produção Assistidos por Computador, Ordem dos Engenheiros, Lisboa, 9-11 Dez 1987.

[CamSte88]

LM Camarinha-Matos; A Steiger-Garção - An Integrated Robot Cell Programming System
International Symposium and Exposition on Robotics (ISER / 19th ISIR), Sydney, Australia, 6-10 Nov. 1988.

[CamSteBap87]

L. Camarinha; A. Steiger; J. Baptista - Integrating CAD in the CIM Information System- Some preliminary results
Relatório UNL-DI 21/87; GR-E623 RT-DA-31-87, Jun 87.
Documento ESPRIT 623 EP-UNL-06.87/1; Technical Meeting Universidade Politecnica Madrid, 29-30 Jun 87.

[Car87]

Carnegie Group Inc. - Knowledge Craft User's Manual
Mai 88.

[Car88]

Carnegie Group Inc. - Simpak / Graphpak
Fev 88.

[Cla85]

B.D. Clayton - ART Programming Primer
Inference Corporation, Apr 85.

[CohFei82]

P.R. Cohen; E.A. Feigenbaum (Ed.s) - Planning and Problem Solving
in *The Handbook of AI*, vol. 3 (cap. XV), 1982.

[ColPalRat84]

K. Collins; A.J. Palmer; K. Rathmill - The development of an European Benchmark for the Comparison of Assembly Robot Programming Systems
Cranfield Institute of Technology, ITTF/2374/84.

[Cor86]

L.M.P. Correia - Sistemas de comunicação em Robótica Distribuída
Provas de Aptidão Pedagógica e Capacidade Científica, Universidade Nova de Lisboa - FCT, Departamento de Informática, Dez 1986.

[Cyt88]

Cyted - Proyecto Robotica y Automatización Avanzada - informe semestral
Santiago Chile, Nov 88.

[DilHuc85]

R. Dillmann; M. Huck - Intelligent Simulation of Robot Application
Proceedings of Symposium on Robot Control '85 (SYROCO'85), Barcelona, 6-8 Nov 1988.

[DitDay86]

K. Dittrich; U. Dayal - Proceedings of the 1986 International workshop on Object-Oriented Database Systems
IEEE Computer Society, Set. 1986.

[Dou86?]

McDonnell Douglas Automation Company - Join us on the frontier of factory automation - Robotics. Folheto de especificações, 1986 (?).

[Eng88]

J.F. Engelberger - Domesticating the industrial robot
International Symposium and Exposition on Robotics (ISER / 19th ISIR), Sydney, Australia, 6-10 Nov. 1988.

[Fer87]

D.M. Ferreira - Bases de Dados em Engenharia: Aplicação à Robótica
Provas de Aptidão Pedagógica e Capacidade Científica, Universidade Nova de Lisboa - FCT, Departamento de Informática, Dez 1987.

[FerMouCam87]

D. Ferreira; J. Moura-Pires; L. Camarinha - Evaluation of Information System development tools - update 1
Relatório UNL-DI 32/87; GR-E623 RT-DA-43-87, Ago 87.
Documento ESPRIT 623 EP-UNL-08.87/1.

[Fik87]

R. Fikes (entrevista) - Fikes puts AI research to the test
Intellinews, vol.3,n.4, Jul. 1987.

[FikHarNil81]

R. Fikes; P. Hart; N. Nilsson - Learning and Executing Generalized Robot Plans
in "Readings in AI", B.L. Webber, Nils J. Nilsson (Eds.), *Tioga Publ. Comp.*, 1981.

[FroHoe86]

B. Frommherz; K. Hoermann - A concept for a robot action planning system
Proceedings of *NATO Advanced Research Workshop on Languages for sensor-based control in Robotics*, Castelvecchio Pascoli, Italy, Sep 1-5, 1986. Editado pela *Springer-Verlag, Nato ASI Series N. 29*, 1987.

[FuGonLee87]

K.S. Fu; R.C. Gonzalez; C.S.G. Lee - Robotics: Control, Sensing, Vision, and Intelligence
McGraw-Hill International Ed., Industrial Engineering Series, 1987.

[Fur87]

B. Furth - Automated Process Planning
Nato Advanced Study Institute on CIM: Current Status and Challenges, Istanbul, Turkey, Ago 31- Set 12, 1987 (Springer-Verlag, Nato ASI Series, 1988).

[Gai87]

A. Gairola - Design for Assembly: A challenge for expert systems
in "Artificial Intelligence in Manufacturing", T. Bernold (ed.), *Elsevier Science Publishers*, 1987.

[GalPazRod88]

R. Galleni; A. Pezzinga; F. Rodighiero - Task level robot programming: an industrial approach
Proceedings of the *2nd International Symposium on Robot Manipulators: Modelling, Control and Education*, Albuquerque, USA, 16-18 Nov 1988.

[GanTho87]

M.V. Gandhi; B.S. Thompson - Automated Design of Modular Fixtures for Flexible Manufacturing Systems
Journal of Manufacturing Systems (SME), vol.5, N. 4, 1987.

[Gev82]

W.B. Gevarter - An overview of Computer Vision
National Bureau of Standards, Washington DC,USA, NBSIR-2582,Sep. 82.

[Gev84]

W.B. Gevarter - An overview of Artificial Intelligence and Robotics - Part A: The core ingredients
National Bureau of Standards, Washington DC,USA, NBSIR 83-2687, Jan. 84.

[Gin86]

M. Gini - Symbolic and qualitative reasoning for error recovery in robot programs
Proceedings of *NATO Advanced Research Workshop on Languages for sensor-based control in Robotics*, Castelvecchio Pascoli, Italy, Sep 1-5, 1986. Editado pela *Springer-Verlag, Nato ASI Series N. 29*, 1987.

[GinGin82]

G. Gini; M. Gini - Ada: a language for robot programming?
Computers in Industry, vol.3, n.4, 1982.

[GinGin83]

M. Gini; G. Gini - Towards Automatic Error Recovery in Robot Programs
Proceedings of IJCAI, 1983.

[GraJon88]

I. Graham; P.L. Jones - Expert Systems - Knowledge, Uncertainty and Decision
Chapman and Hall Computing, London, New York, 1988.

[HarBarLee86]

N.W. Hardy; D.P. Barnes; M.H. Lee - Declarative Sensor Knowledge in a Robot Monitoring System
Proceedings of *NATO Advanced Research Workshop on Languages for sensor-based control in Robotics*, Castelvecchio Pascoli, Italy, Sep 1-5, 1986. Editado pela *Springer-Verlag, Nato ASI Series N. 29*, 1987.

[HarMar83]

E.E. Hartquist; H.A. Marisa - Padl-2 Users Manual
University of Rochester; Production Automation Project, UM-10/1.2, May 83.

[HayHay...79]

B. Hayes-Roth; F. Hayes-Roth; S. Rosenschein; S. Cammarata - Modeling planning as an incremental opportunistic process
Proceedings of IJCAI 79.

[HirAraHac87]

K. Hirota; Y. Arai; S. Hachisu - Fuzzy robot vision and fuzzy controlled robot
Nato Advanced Study Institute on CIM: Current Status and Challenges, Istanbul, Turkey, Ago 31- Set 12, 1987 (Springer-Verlag, Nato ASI Series, 1988).

[Int86a]

IntelliCorp - KEE Software Development System
User's Manual, Jul 88.

[Int86b]

IntelligenceWare, Inc. - Intelligence Compiler
Users Manual, 1986.

[Int87]

IntelliCorp - KEEconnection: A Bridge between Databases and Knowledge bases. 1987.

[Ipk87]

IPK, Berlin - Optimal motion planning procedure - IPK deliverable description
in *Esprit 623 Description of Deliverables*, Review meeting, Milano, Italy, Abr.1987.

[Itm84]

Itmi - LM Reference Manual, version 2.1
ITMI, Meylan, France, 1984.

[JakNas88]

W. Jakob; H. Nase - ESL - Refined definition of the Explicit Solution Language
Esprit 623 working paper IP-PSI-02.88/1, Fev 88.

[KakBoy...86]

A.C. Kak; K.L. Boyer; C.H. Chen; R.J. Safranek; H.S. Yang - A Knowledge-based Robotic Assembly Cell
IEEE Expert, Spring 86.

[KelBon85]

R.B. Kelley; S. Bonner - Understanding, uncertainty and robot task execution
Proceedings of Symposium on Robot Control '85 (SYROCO'85), Barcelona, 6-8 Nov 1988.

[KemWalLoc86]

A. Kemper; M. Wallrath; P.C. Lockemann - Database Support for Robotics Applications
Proceedings of *NATO Advanced Research Workshop on Languages for sensor-based control in Robotics*, Castelvecchio Pascoli, Italy, Sep 1-5, 1986. Editado pela *Springer-Verlag, Nato ASI Series N. 29*, 1987.

[KuiDui..86]

E.A. Kuijpers; W. Duinker; L.O. Hertzberger; G.R. Meijer; F. Tuynman - Handling uncertainties and inaccuracies in rules for multi-sensor robot systems
Proceedings of *NATO Advanced Research Workshop on Languages for sensor-based control in Robotics*, Castelvecchio Pascoli, Italy, Sep 1-5, 1986. Editado pela *Springer-Verlag, Nato ASI Series N. 29*, 1987.

[KusHer87]

A. Kusiak; S. S. Heragu - Group Technology - State-of-the-art
Computers in Industry (North-Holland), vol.9, pp 83-91, 1987.

[Lat84]

J.-C. Latombe - Automatic Synthesis of Robot Programs from CAD Specifications
NATO ASI on Robotics and AI, Castelvecchio Pascoli, Italy, 26 Jun- 8 Jul, 1983, publicado em *Nato ASI Séries, Serie F*, Vol. 11, Springer-Verlag, 1984.

[Lau88]

C. Laugier - Les apports respectifs des langages symboliques et de la cao en programmation des robots
Robotica (Cambridge University Press), vol.6, pp 243-253, 1988.

[LauPer85]

C. Laugier; J. Pertin-Troccaz - SHARP: A System for Automatic Programming of Manipulation Robots
3rd International Symposium on Robotics Research, Gouvieux (France), 7-11 Out. 1985.

[LeeBarHar83]

M.H. Lee; D.P. Barnes; N.W. Hardy - Knowledge Based Error Recovery in Industrial Robots
Proceedings of IJCAI, 1983.

[Lor85]

Lord - Force / Torque wrist sensing systems (technical sheet)
Lord Industrial Automation, 1985.

[Loz83]

T. Lozano-Pérez - Robot Programming
Proceedings of the IEEE, vol71, n.7, Jul 83.

[LozBro85]

T. Lozano-Pérez; R.A. Brooks - An Approach to Automatic Robot Programming. MIT, AI Memo N. 42, Abr 85.

[Mac86]

V.A.C. Machado - Tecnologia de Grupo - Uma filosofia de produção
Relatório UNIROB RT-L4-51-86.
Provas de aptidão pedagógica e capacidade científica, UNL/FCT, Dez. 1986.

[Mei86]

G.R. Meijer - Functionality and specification of the High Level Interpreter
Esprit 623 working paper EP-UVA-09.86/1, Amsterdam.

[Mei88]

G.R. Meijer - Specification of Active Component HLI
in *Esprit 623 6th Interim Report*, Feb 88.

[MeiDui...86]

G.R. Meijer; W. Duinker; L.O. Hertzberger; E.A. Kuijpers; F. Tuijnman - Functionality and specification of the high level interpreter
Esprit 623 working paper EP-UVA-9.86/1, Sep 86.

[MicCarMit83]

R.S. Michalski; J. Carbonell; T. Mitchell - Machine Learning. An Artificial Intelligence Approach, Vol.1,
Tioga Publishing Comp., 1983.

[MicCarMit85]

R.S. Michalski; J. Carbonell; T. Mitchell - Machine Learning. An Artificial Intelligence Approach, Vol.2, *Morgan Kaufman*, 1985.

[Mil87]

R. Milne - Artificial Intelligence for on-line diagnosis
Esprit / CIM Europe SIG workshop on AI methods and tools in CIM, Athens 28-30 Jan. 1987.

[MilMaj87]

V. Milacic'; V. Majstorovic' - The future of computerized maintenance
Proceedings of the IFIP WG 5.3 Working Conference on Diagnostic and Preventive Maintenance Strategies in Manufacturing Systems, Beograd, Yugoslavia, 1987 (a ser publicado por Elsevier Science Publishers).

[Min81]

M. Minsky - A Framework for Representing Knowledge
in *Mind Design*, J. Haugeland (Ed.), MIT Press, 1981.

[Mou86]

F.J. Moura-Pires - Sistemas sensoriais em robótica
Provas de Aptidão Pedagógica e Capacidade Científica, Universidade Nova de Lisboa - FCT, Departamento de Informática, Dez 1986.

[MouCam88]

J. Moura Pires; L.M. Camarinha Matos - Integration of CAD information in the Information System - a refined version
Relatório UNL-DI 09/88; GR-E623 RT-DA-10-88, Jan 88.
Documento ESPRIT 623 EP-UNL-01.88/1.

[MouPim88]

F.J. Moura-Pires; J.P. Pimentão - Sistemas autónomos: uma proposta de modelação do sistema sensorial
Universidade Nova de Lisboa, Grupo de Robótica, Jul 88.

[MouSteCam86]

F.J. Moura-Pires; A. Steiger-Garção; L.M. Camarinha-Matos - An Architecture for Sensor Integration and Interpretation
Proceedings of *MECO'86*, Taormina, Italy, 3-5 Set 86.

[Neg88]

U. Negretto - A functional process model for flexible assembly cells
University of Karlsruhe, Germany, 1988.

[Nei87]

Gerard Neipp - Artificial Intelligence and its impact on industry - A new dimension within the framework of computer-integrated manufacturing (CIM)
Esprit / CIM Europe SIG workshop on AI methods and tools in CIM, Athens 28-30 Jan. 1987.

[Neu85]

Neuron Data - NEXPERT, 1985.

[Nev86]

J.A.C. Neves - Programação de robôs - linguagens e simulação
Provas de aptidão pedagógica e capacidade científica, Universidade Nova de Lisboa, Departamento de Informática, Dez. 1986.

[NevWhi78]

J.L. Nevins; D.E. Whitney - Computer-controlled Assembly
Scientific American, Fev 1978.

[Nil80]

N. Nilsson - Principles of Artificial Intelligence
Tioga Publishing Comp., 1980.

[O'haBlaCon87]

G.M.P. O'Hare; W.J. Black; G.V. Conro - Predictive maintenance: a new paradigm for diagnostic expert systems
University of Manchester, Institute of Science and Technology, 1987.

[Oli84]

E. Oliveira - Caracterização e perspectivação da robótica inteligente
Prova complementar de doutoramento, Universidade Nova de Lisboa, Departamento de Informática, Jul. 1984.

[PasAnt88]

K.M. Passino; P.J. Antsaklis - Fault detection and identification in an intelligent restructurable controller
Journal of Intelligent and Robotic Systems, vol.1, n.2, 1988.

[Pau81]

R.P. Paul - Robot Manipulator: Mathematics, Programming and Control
MIT Press, 1981.

[PerMon..85]

L.M. Pereira; L. Monteiro; J. Cunha; J. Aparício - Delta Prolog: A distributed backtracking extension with events. UNL, Nov 85.

[PitCam88]

H. J. Pinheiro Pita; L.M. Camarinha Matos - Gestor de referenciais - Uma implementação sobre Knowledge Craft
Relatório UNL-DI 33/88; GR RT-DA-36-88; Jul 88.

[PopAmbBel78]

R.J. Popplestone; A.P. Ambler; I.M. Bellos - RAPT: A Language for describing assemblies
Industrial Robot, vol. 5, n.3, 1978.

[Rad86]

Radian Corp. - RuleMaster - Software Tool for Building Expert Systems
Reference Manual, 1986.

[Rem86]

U. Rembold - Programming of Industrial Robots - Today and in the Future
Proceedings of *NATO Advanced Research Workshop on Languages for sensor-based control in Robotics*, Castelvecchio Pascoli, Italy, Sep 1-5, 1986. Editado pela *Springer-Verlag, Nato ASI Series N. 29*, 1987.

[RemSoe87]

U. Rembold; T. Soetadji - The Development of an Autonomous Assembly Robot
Robotics & Computer-Integrated Manufacturing, vol. 3, n.1, 1987.

[Ren87]

Renault Automation - Collision detection of the Renault programming system
in *Esprit 623 5th Interim Report*, 1987.

[Ren88]

Renault Automation - Demonstrator B
in *Esprit 623 7th Interim Report*, Jul 88.

[Req80a]

A. Requicha - Representation of Rigid Solid Objects
in *Computer Aided Design - modeling, systems engineering, CAD-systems*, *Springer-Verlag*, Lecture Notes in Computer Science 89, Set. 1980.

[Rod87]

F. Rodighiero - Progress in the implementation of the Action Sequence Planner and of the Gross Motion Planner
Esprit 623 working paper IP-FIAR-1.87/1, Jan 87.

[Sac77]

E. Sacerdoti - A Structure for Plans and Behavior
Elsevier Computer Science Library, North-Holland, 1977.

[Sch86]

E.G. Schlechtendahl (ed.) - Specification of a CAD*I Neutral File for Solids
Springer-Verlag, 1986.

[Sch87]

R.D. Schraft - The industrial Robots in a Flexible Manufacturing System: State of Art and Prospects
in "Artificial Intelligence in Manufacturing", T. Bernold (ed.), *Elsevier Science Publishers*, 1987.

[SchKar87]

H.-J. Schneider; D. Karagiannis - Intelligent Knowledge Bases in CAD Environments: The hybrid system KANON
Nato Advanced Study Institute on CIM: Current Status and Challenges, Istanbul, Turkey, Ago 31- Set 12, 1987 (Springer-Verlag, Nato ASI Series, 1988).

[Sha85]

Shape Data Ltd. - The Romulus Solid Modelling System
V 6.0 User's Reference Manual version 1, Set 85.

[SpuFurKir87]

G. Spur; I. Furgac; U. Kirchhoff - Robot System Integration into Computer-Integrated Manufacturing
Robotics & Computer-Integrated Manufacturing, vol.3,n.1, 1987.

[Spu...85]

G. Spur et al. - Operational Control for Robot System Integration into CIM
ESPRIT 623 2nd. Interim Report, Berlin, IPK, 1986.

[Spu...87]

G. Spur et al. - Operational Control for Robot System Integration into CIM
ESPRIT 623 5th. Interim Report, Berlin, IPK, 1987.

[Sta87]

A. C. Staugaard - Robotics and AI - An introduction to applied machine intelligence
Prentice Hall, Inc., 1987.

[Ste81]

M. Stefik - Planning and meta-planning (MOLGEN: Part 2)
in *Readings in AI, Tioga Publ. Company*, 1981.

[SteCam85]

A. Steiger-Garção; L.M. Camarinha-Matos - Proposal for joining the ESPRIT Project Nr. 623
Relatório UNL-DI 34/85; UNIROB RT-L2-15-85, Dec 1985.
Proposta submetida (e aprovada pel)à CEE in ESPRIT 623 Minutes and Working Papers e Technical Annex.

[SteCam86a]

A. Steiger-Garção; L.M. Camarinha-Matos - Concurrent Pascal as a robot level language - a suggestion
Relatório UNL-DI 22/85; UNIROB RT-L2-10-85, Jul 1985.
Revista *ROBOTICA (Cambridge University Press*, U.K.), vol.4, n.4, Oct-Dec 86.

[SteCam86b]

A. Steiger-Garção; L.M. Camarinha-Matos - A Knowledge Based Approach for Multisensorial Integration
Relatório UNL-DI 31/86; UNIROB RT-L2-22-86, Jul 86.
Proceedings of *NATO Advanced Research Workshop on Languages for sensor-based control in Robotics*, Castelvecchio Pascoli, Italy, Sep 1-5, 1986. Editado pela *Springer-Verlag, Nato ASI Series N. 29*, 1987.

[SteCam87a]

A. Steiger; L. Camarinha - Esprit 623: Work Report and Preliminary Results - Contribution to the 4th Interim Report
UNL-DI 5/87; GR-E623 RP-DA-04/87, Jan 87.
in ESPRIT 623: 4th Interim Report.

[SteCam87b]

A. Steiger-Garção; L.M. Camarinha-Matos - A Conceptual Structure for a Robot Station Programming System
Relatório UNL-DI 1/86; UNIROB RT-L2-01-86, Jan 1986.
Revista *ROBOTICS - International Journal (North-Holland) vol.3, n.2, Jun1987.*

[SteCam87c]

A. Steiger; L. Camarinha - ESPRIT 623: UNL's contribution to the 5th Interim Report
Relatório UNL-DI 31/87; GR-E623 RP-DA-42-87, Ago 87.
in ESPRIT 623: 5th Interim Report.

[SteCam88a]

A. Steiger Garção; L.M. Camarinha Matos - Uma perspectiva integrada para a programação de células robóticas
5º Congresso Português de Informática, Lisboa, Maio 1988.

[SteCam88b]

A. Steiger-Garção; L.M. Camarinha-Matos - An Integrated Architecture for Robot Cell Programming and Monitoring
2nd International Symposium on Robotics and Manufacturing Research, Education and Applications, Albuquerque, USA, 16-18 Nov. 1988.

[SteCamFer88a]

A. Steiger Garção; L.M. Camarinha Matos; D. Ferreira - CIMPEP: UNL's contribution to an Esprit II Project proposal
Relatório UNL-DI 11/88; GR-E623 RT-DA-12-88, Fev 88.
in *CIM-PLATO, proposta submetida (e aprovada pel)à CEE - Esprit II.*

[SteCamFer88b]

A. Steiger-Garção; L.M. Camarinha-Matos; D.M. Ferreira - Demonstrator A - Work program for the module: Information System Intelligent Interface
Documento Esprit 623 EP-UNL-07.88/1 Jul 88.

[SteCamMou86]

A. Steiger-Garção; L.M. Camarinha-Matos; F.J. Moura-Pires - Education in Robotics: Guidelines for a Curriculum
Proceedings of the *1st International Symposium on Robot Manipulators: Modelling, Control and Education*, Albuquerque, USA, 12-14 Nov 1986.

[SteCamMou88]

A. Steiger Garção; L.M. Camarinha Matos; J. Moura Pires; D. Ferreira - ESPRIT 623: UNL's contribution to the 6th Interim Report
Relatório UNL-DI 07/88; GR-E623 RT-DA-08-88, Fev 88.
in ESPRIT 623 6th Interim Report.

[SteCam...88a]

A. Steiger; L. Camarinha; J. M. Pires; L. Correia; J.S. Afonso -Integrated Development Environment for Robot Cell Programming - Prototype description
Relatório UNL-DI 08/88; GR-E623 RT-DA-09-88, Jan 88.
Documento ESPRIT 623 EP-UNL-01.88/2

[SteCam...88b]

A. Steiger-Garção; L.M. Camarinha-Matos; L. Basañez; C. Torras; R. Niepold; R. Dillmann; L. Saitta et al. - B-LEARN: Acquisition of Behavioral Knowledge for Autonomous Systems Operating in Real Environments
Proposta de projecto submetido ao programa Esprit Basic Research Action, Jun 88.

[SteCam...88c]

A. Steiger-Garção; L.M. Camarinha-Matos; D.M. Ferreira; J. Moura-Pires - Demonstrator A - Work program for the module: Information System Intelligent Interface
Relatório UNL-DI 34/88; GR E623 RT-DA-40-88
Documento Esprit 623 EP-UNL-07.88/1 ; Jul 88.

[SteCam...88d]

A. Steiger-Garção; L.M. Camarinha-Matos; D.M. Ferreira; J. Moura-Pires - Demonstrator C - Work program for the modules: Information System and Information Based Integration
Relatório UNL-DI 35/88; GR E623 RT-DA-41-88
Documento Esprit 623 EP-UNL-07.88/2 ; Jul 88.

[SteMouSan88]

A. Steiger-Garção; F. Moura-Pires; J. Santos-Afonso - Object Identification - a frame-oriented approach
NATO ARW on Highly Redundant Sensing in Robotic Systems, Il Ciocco, Italy, May 88.

[SteSanQue87]

A. Steiger-Garção; J. Santos-Afonso; C. Queirós - Object Identification and Automatic Learning (A vision,CAD and AI based approach)
NATO Advanced Research Workshop on Real Time Object Environment measurement and classification, Maratea, Italy, Set 1988.

[Sto87]

R.K. Stobart - Collision detection for offline programming of robots
in *Off-line programming of industrial robots*, A. Ston; J.F. McWaters (Ed.s), *Elsevier Science Publishers* (North-Holland) / IFIP 1987.

[Sus75]

G. Sussman - A computer model of skill acquisition
Elsevier Computer Science Library, North-Holland, 1975.

[ThoTho86]

B. Thompson; W. Thompson - Finding Rules in Data
Byte, Nov. 86.

[TieBowBro86]

K. Tierney; R. Bowden; J. Browne - A prototype expert system for technological planning in robot based flexible assembly systems
University College Galway, Ireland. *Proceedings of the FMS Conference*, Univ. of Ann Arbor, Michigan, Aug 12-15, 1986.

[ThoTor88]

F. Thomas; C. Torras - Constraint-based inference of assembly configurations
Proceedings of the IEEE Conf. on Robotics and Automation, Philadelfia, USA, Apr 88.

[Vdi82]

Verein Deutscher Ingenieure - Montage-und Handhabungstechnik Handhabungsfunktionen, Handhabungseinrichtungen, Begriffe, Definitionen, Symbole, *VDI 2860*, Okt 82.

[VolWol...86]

R.A. Volz; J. Wolter; J. Barber; R. Desai; A.C. Woo - Automatic Determination of Gripping Positions

Proceedings of *NATO Advanced Research Workshop on Languages for sensor-based control in Robotics*, Castelvecchio Pascoli, Italy, Sep 1-5, 1986. Editado pela *Springer-Verlag, Nato ASI Series N. 29*, 1987.

[War74]

D. Warren - WARPLAN: A system for generating plans

Dep. Computational Logic, Memo 76, University of Edinburgh, Jun 74.

[WörSta87]

H. Wörn; G. Stark - Robot applications supported by CAD simulation

Robotics & Computer-Integrated Manufacturing, vol.3,n.1,1987.

[Zad83]

L.A. Zadeh - A computational approach to fuzzy quantifiers in natural languages

Comp. & Mathematics with Applications, vol 9, n.1, 1983.

[Zri86]

T. Zrimec - Application of logic programming to task-level programming of robots

Proceedings of *8th IASTED / AFCET International Symposium on Robotics and Artificial Intelligence*, Toulouse 18-20 Jun 86.

ANEXOS

Com o objectivo de facilitar a compreensão de alguns exemplos inseridos no texto, faz-se uma breve apresentação dos sistemas de regras OPS e Prolog usados no Knowledge Craft.

ANEXO A: Regras CRL-OPS

O formato geral destas regras é:

```
(p <nome-da-regra>
      (<elemento-condição>)
      [ [-] (<elemento-condição>) ]*
-->
      [(<acção>)]* )
```

(equivalente a if <condição> then <acção> numa notação mais usual).

Um <elemento-condição> consiste no identificador dum *frame* representando um protótipo ou classe de elementos (robô, por exemplo) e num conjunto de testes sobre *slots* desse *frame*. Exemplo:

```
(p procura-robô-s-5                    ;Regra disparável se existir
      (robô ^tipo scara                ;um robô do tipo scara e com
            ^graus-de-liberdade >= 5)  ;um nº de graus de liberdade >= 5
-->
      ...)
```

Os operadores relacionais a usar para teste sobre slots podem ser:
= (por defeito), >, >=, <, <=, <>, <=> (do mesmo tipo), predicado definido pelo utilizador.

O operador "-" precedendo um <elemento-condição> representa um negação. Por exemplo:

```
(p aceitar-na-falta-de-melhor          ;Esta regra é disparável caso
      (garra ^princípio sucção)        ;exista uma garra de sucção
    - (garra ^princípio magnético)     ;desde que não exista nenhuma
 -->                                   ;magnética
      ...)
```

Outro exemplo mais complexo:

```
(p procura-robô-especial               ;As variáveis R e F são afectadas
      (robô ^shema-name <R>            ;com, respectivamente, o nome e
            ^fabricante <F>            ;o fabricante dum robô com
            ^Graus-de-liberdade << 5 6 >> ;5 ou 6 graus de liberdade e
            ^carga { >20 <50 })        ;carga maior que 20 e menor que 50
-->
      ...)
```

Uma <acção> pode ser a chamada de qualquer função Lisp, incluindo as funções do Knowledge Craft.

Algoritmo de encadeamento de regras:

i•• Reconhece regras instanciadas

{Regras}

---> {Conflict Set}

{Frames}

ii•• Resolução do Conflict Set -selecciona a regra dominante desse conjunto

{Conflict Set} ---> Uma regra

iii•• Acção - executa a parte de acção da regra seleccionada

iv•• Repetir

Onde:

-Instância duma regra: A regra mais o conjunto de frames unificáveis com a parte de condição e que, portanto, possibilita o disparo da regra.

-*Conflict set:* conjunto de todas as instanciações correntes.

Para a selecção da regra dominante são utilizados os seguintes "filtros":

1• Refracção - As instâncias apenas podem ser executadas uma vez.

2• Recência - Dá preferência aos frames mais recentes.

3• Especificidade - Dá preferência às regras mais específicas.

A especificidade duma regra é dada por um parâmetro que é a soma de

-nº de elementos-condição positivos;

-nº de testes nesses elementos;

-nº de constantes nesses elementos.

Caso subsistam várias regras candidatas no final da aplicação destes filtros, é escolhida uma de forma aleatória.

Para mais detalhes consultar "Knowledge Craft CRL-OPS" in Knowledge Craft Users Manual, Carnegie Group Inc., May 88.

ANEXO B: CRL-Prolog

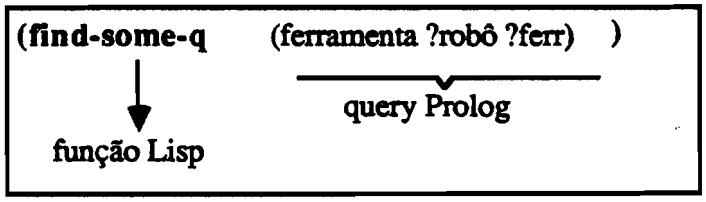
Este subsistema do Knowledge Craft é muito influenciado pelo ambiente "Lispiano" circundante e, portanto, usa uma sintaxe bastante diferente da usual notação de Edinburgh. Numa apresentação comparada tem-se:

	CRL-Prolog	Edinburgh Prolog
Factos:	((graus-liberdade Renault 5))	graus-liberdade(renault, 5).
	((ferramenta(Renault, Garra1))	ferramenta(renault, garra1).
Regras:	((bom-robô ?robô ?G ?F)	bom-robô(Robô, G, F)
	<	:-
	(graus-liberdade ?robô ?G)	graus-liberdade(Robô, G),
	(ferramenta ?robô ?F))	ferramenta(Robô, F).

Interface com Common Lisp: Um conjunto de funções e predicados permite a integração do Prolog com os restantes componentes do Knowledge Craft.

Lisp => Prolog:

1.



...procura a primeira solução

Resposta: (((?robô . Renault) (?ferr . Garra1)))

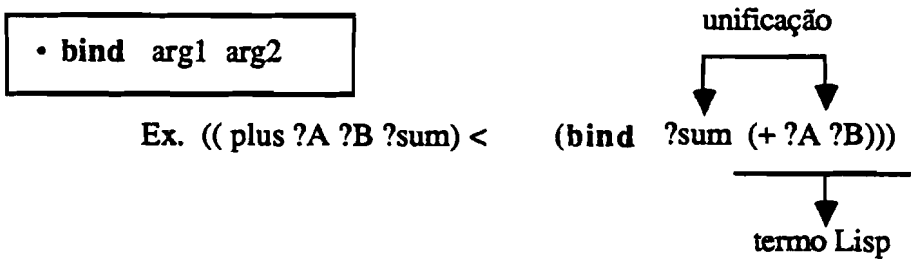
2.

...procura todas as soluções

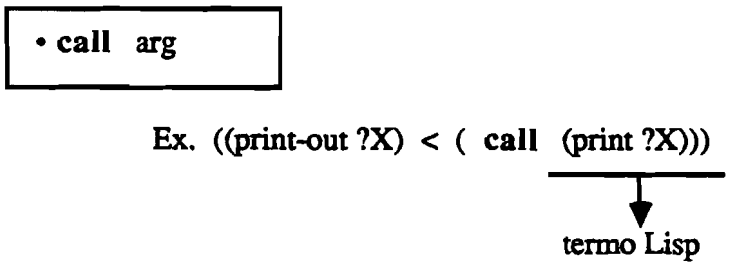
Resposta: (((?robô . Renault) (?ferr . Garra1))
((?robô . Rhino) (?ferr . Garra2))
...)

Prolog => Lisp:

3.



4.



Predicados pré-definidos para acesso a frames:

O acesso ao subsistema de frames a partir do Prolog poderá ser feito através dos predicados *bind* e *call*. Contudo uma forma mais simples está disponível através dum conjunto de predicados específicos:

Nível	Operações	Por defeito
:schema	:exists :create :delete :print	:exists
:slot	:exists :create :add :delete	operação do nível anterior
:values	:exists :create :add :delete	operação do nível anterior

Nota: *schema*, em terminologia Knowledge Craft, significa *frame*

Exemplos:

(:schema ?R (operações (:values mover fechar abrir)))

...tem sucesso se o frame ?R tiver um slot operações com os valores mover, abrir e fechar.

(:schema RENAULT (ferr-corr (:values :create GARRA-1)))

...se existir um frame Renault com um slot ferr-corr, então o valor Garra-1 é afectado a esse slot; senão o predicado falha.

Para mais detalhes consultar "Knowledge Craft CRL-PROLOG" in Knowledge Craft Users Manual, Carnegie Group Inc., May 88.