

## CAPÍTULO 6

# MODELO PROPOSTO: OPERAÇÃO

---



*Quanto mais quero uma coisa feita, menos a considero como trabalho*

Richard Bach, “Ilusões: As aventuras de um Messias relutante”.

No capítulo anterior foi proposta uma arquitectura holónica para sistemas de produção, de acordo com o que foi estipulado no Capítulo 3 sobre os sistemas de produção do futuro. Deste modo, partindo do conceito holónico e das ferramentas de prototipagem apresentadas no Capítulo 4, cada elemento (*i.e.*, holon) da arquitectura foi caracterizado em termos de objectivos, ciclo de vida e base de conhecimento. Neste capítulo esses holons continuarão a ser objecto de estudo mas agora sob um ponto de vista operacional.

Devido à natureza distribuída da arquitectura apresentada na secção 5.2 torna-se necessário regulamentar a interacção entre os vários holons, especialmente os Holons de Tarefa e de Recursos, na actividade de escalonamento (escolhida como caso de teste para o modelo proposto e respectiva arquitectura). Por esse motivo passar-se-á à especificação de um protocolo de negociação que está preparado para o tratamento de excepções e principalmente para evitar o *Problema de Indecisão*. É também efectuado um estudo da complexidade do protocolo. Em seguida serão apresentados os principais algoritmos de funcionamento dos holons relacionados com o processo de escalonamento de tarefas. Completada a especificação do sistema, será então

apresentado o protótipo *Fabricare*. Seguidamente é efectuada uma comparação entre o trabalho desenvolvido e os trabalhos de referência apresentados no estado da arte.

## 6.1 Introdução

De acordo com o que foi referido nos objectivos deste trabalho, o caso de teste escolhido para validar o modelo computacional e implementar o protótipo foi o de Escalonamento de Tarefas Industriais. Desse modo, convém abordar algumas questões de escalonamento, completando assim o que foi dito na secção 2.4.5 “Escalonamento e Balanceamento” sobre esta actividade de produção.

A Figura 6.1 apresenta um gráfico de Gantt respeitante ao escalonamento de uma ordem de fabrico com quatro operações sequenciais com o intuito de se produzirem três itens de um produto (aqui referenciados por rectângulos com linhas obliquas). Na figura são também visíveis algumas operações já escalonadas em cada recurso (aqui referenciadas por rectângulos ponteados) e a utilização de um mesmo recurso para mais que uma operação da tarefa (M1).

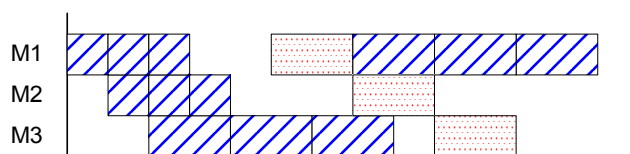


Figura 6.1 – Escalonamento

Este exemplo, e os seguintes, denotam uma situação por ventura simplificada, ao não se considerarem as operações de transporte entre máquinas (ou seja, assume-se que existe sempre transporte disponível de uma máquina para outra, estando a duração desse transporte reduzida a zero), não se considerando também o tempo para o setup da máquina para cada operação a executar (ou seja, assume-se que o setup tem duração zero).

Por outro lado, cada operação é iniciada logo a seguir ao fim da execução da operação antecessora; ou seja, no caso referido em epígrafe, não se espera que se complete o fabrico de cada um dos três produtos para executar a operação seguinte sendo, pelo contrário, cada produto tratado individualmente. Neste caso é então necessário tomar atenção às situações em que a operação que se segue tem uma duração menor que a operação antecessora (Figura 6.2). Conforme se pode observar na Figura 6.2a não é possível iniciar a execução das operações em M3 quando em M2 se terminar de processar um produto. Nestas situações é necessário deixar “buracos” no escalonamento correspondentes à diferença entre as durações das operações em presença, ou, atrasar o início das operações em M3, garantindo dessa forma que todos os produtos estão disponíveis quando M3 iniciar a laboração (Figura 6.2b).

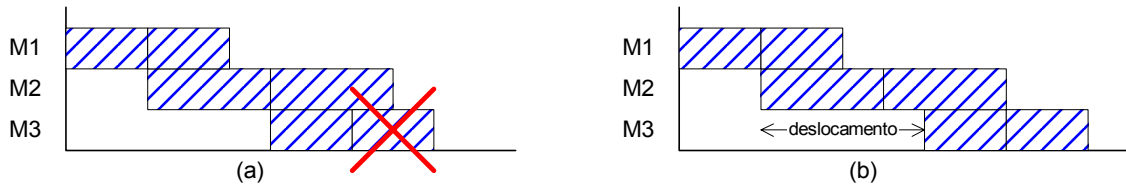


Figura 6.2 – Escalonamento com operações antecessoras de maior duração

O atraso (ou deslocamento) é calculado pela equação (6.1), onde  $\delta(i)$  é uma função que retorna a duração da operação  $i$ .

$$\text{deslocamento} = \delta(i-1) + (\text{quantidade} - 1) \times (\delta(i-1) - \delta(i)) \quad (6.1)$$

Ao atrasar-se as operações em M3, surge então a necessidade da existência de *buffers* nas máquinas. Os *buffers* são espaços de armazenamento à entrada e/ou saída das máquinas onde é possível colocar o material ou produto a processar até que seja utilizado pelo recurso. Estes *buffers* têm obviamente uma limitação física quanto ao espaço de armazenamento mas, não raras vezes, para simplificar a compreensão do processo assumem-se *buffers* de tamanho infinito.

Um outro caso que pode surgir no processo de escalonamento é o da existência de recursos alternativos para a execução de uma mesma operação. Nesta situação, existirão várias combinações possíveis de recursos que podem ser utilizadas na execução da tarefa. A Figura 6.3 mostra uma situação onde uma operação pode ser feita em dois recursos diferentes (M1a e M1b) existindo, neste caso, apenas duas combinações possíveis (M1a – M2 – M3 e M1b – M2 – M3) gerando-se obviamente, escalonamentos diferentes para a tarefa (Figura 6.3a e Figura 6.3b respectivamente).

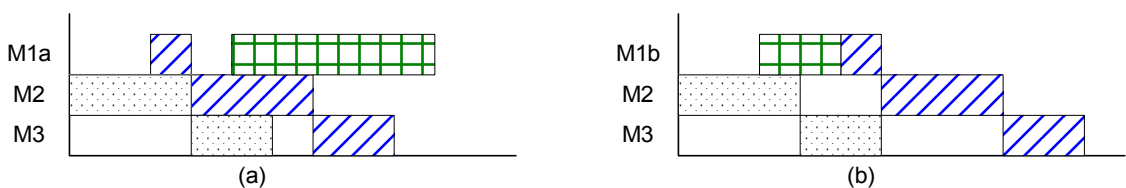


Figura 6.3 – Escalonamento com recursos alternativos

O último caso a considerar é o de poderem ocorrer operações paralelas num plano de produção. Conforme se pode ver na Figura 6.4, a operação *op2* pode ser escalonada em qualquer ponto daquele intervalo de tempo desde que termine antes do início da operação que a sucede; *i.e.*, *op4*. Este tipo de operações é vulgar em tarefas de montagem.

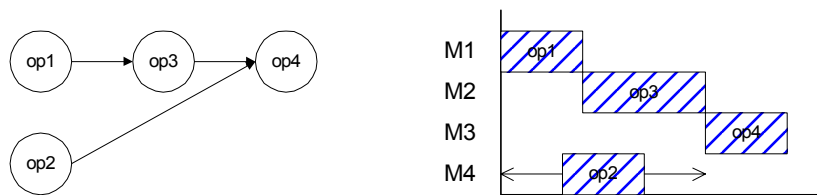


Figura 6.4 – Escalonamento com operações paralelas

## 6.2 Processo de Negociação

Esta secção descreve o processo utilizado para o escalonamento dinâmico de tarefas industriais, através de um protocolo de negociação entre os Holons de Tarefa e de Recurso, baseado no Protocolo de Rede de Contrato [Smith, 1980] [Davis e Smith, 1983]. Este protocolo inclui também uma fase de renegociação para tratamento de excepções e de alteração dinâmica de condições.

Os intervenientes no protocolo, denominado Protocolo de Rede de Contrato com Propagação de Restrições (PRCPR) [Sousa e Ramos, 1998] [Sousa e Ramos, 1999a] [Sousa *et al.*, 1999b], são os Holons de Tarefa e os Holons de Recurso, podendo no entanto ser entendidos como duas abstrações *Recurso* e *Tarefa* à semelhança dos papéis executados pelos agentes no protocolo de rede de contrato [Smith, 1980] [Davis e Smith, 1983]. Uma *Tarefa* representa um trabalho/projecto a ser executado podendo ser dividido em operações concorrentes ou sequenciais. Um *Recurso* é um dispositivo, uma pessoa ou entidade capaz de executar essas operações. Usando este modelo, o mesmo protocolo pode ser usado para escalonamento de ordens de fabrico numa instalação fabril, escalonamento de actividades de um projecto numa empresa virtual, ou gestão de uma cadeia de fornecimento de produtos sendo os fornecedores recursos e as tarefas as compras de produtos.

Para efectuar o escalonamento, o Holon de Tarefa vai negociar com os Holons de Recurso estabelecendo um contrato para a execução de cada operação do plano de produção do produto a que a tarefa se refere. O algoritmo de escalonamento propriamente dito é adaptado de um método centralizado, que considera agendas de recursos e comportamentos de escalonamento [Almeida, 1995] [Ramos *et al.*, 1995]. Cada holon de recurso possui uma agenda de actividades já contratadas e tem que escalonar as novas operações nos espaços livres, de acordo com a janela temporal definida pela tarefa. Devido à existência de restrições temporais de uma operação com as suas operações antecessoras e as suas sucessoras, os holons de recurso irão propagar as suas restrições de forma a eliminar intervalos de tempo que não respeitem a relação de precedência entre operações.

### 6.2.1 Descrição

O protocolo é definido pelos estados de [Sousa e Ramos, 1998] [Sousa e Ramos, 1999a] [Sousa *et al.*, 1999b]: (i) anúncio; (ii) requisição de serviços (iii) influência directa; (iv) influência inversa & oferta; e (v) contratação.

Em relação ao protocolo de rede de contrato original, foram acrescentadas as fases de influência directa e inversa. Estas novas fases correspondem à coordenação de intervalos temporais entre os vários recursos necessários à execução de cada operação da tarefa.

#### 6.2.1.1 Anúncio de Tarefa

A Figura 6.5 (*adaptada de*: [Sousa *et al.*, 1999b]) apresenta as mensagens que são trocadas entre os holons na fase inicial do processo de negociação. Por uma questão de melhor visualização o componente Lançador de Tarefas do Holon de Escalonamento é apresentado em separado.

Em primeiro lugar, o componente *Lançador de Tarefas* do Holon de Escalonamento cria um novo *Holon de Tarefa* e comunica-lhe o que tem que fazer (mensagem *anúncio*). Em seguida, o Holon de Tarefa pede ao *Holon de Planeamento de Processos* um plano de produção para o item a fabricar. Por último, o Holon de Planeamento de Processos responde com o plano seleccionado.

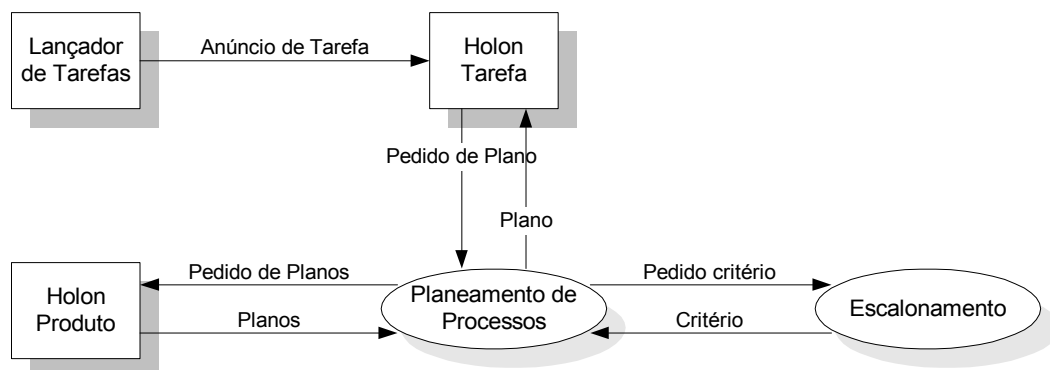


Figura 6.5 – Anúncio de tarefa

O *Holon de Planeamento de Processos* seleccionará de entre os vários planos que foram construídos para o produto, aquele que deve ser utilizado, baseando-se para isso na informação obtida do *Holon de Escalonamento* sobre as condições da instalação fabril (*e.g.*, o melhor plano é aquele que se adequa à minimização do tempo de fabrico, dado que a carga de trabalho está elevada).

### **Mensagem de Anúncio**

O Lançador de Tarefas anuncia ao Holon de Tarefa o que fazer, com uma mensagem do seguinte teor:

*anuncio(TId, NOF, PId, Quant, DataLimite, JanelaTemporal, Attr)*

onde *TId*, *NOF*, *PId*, *Quant*, *DataLimite*, *Attr* e *JanelaTemporal* denotam respectivamente, a identificação atribuída ao holon de tarefa, o número de ordem de fabrico associada àquela tarefa, a identificação do produto a fabricar, a quantidade de produtos a fabricar, a data especificada para conclusão da tarefa, a lista de atributos da tarefa e, o intervalo de tempo em que o escalonamento faz sentido (o limite superior é atribuído com base na data limite da tarefa).

### **Mensagem de Pedido de Plano**

O Holon de Tarefa requisita um plano ao Holon de Planeamento de Processos, enviando uma mensagem com o seguinte teor:

*obter\_plano(PId, Attr)*

onde *PId* e *Attr* denotam respectivamente, a identificação do produto para o qual se requisita um plano e, a lista de atributos/restrições impostas a esse pedido. O termo *Attr* é usado para indicar ao Holon de Planeamento de Processos as características desejáveis (ou indesejáveis) no plano, permitindo assim que o Holon de Planeamento de Processos seleccione um plano de entre os planos alternativos, tendo em conta as necessidades impostas pela tarefa, (e.g., um plano que não utilize determinados recursos que de antemão se sabe estarem completamente ocupados). Algumas das instanciações possíveis para *Attr* podem tomar a forma: *nao\_contem\_operacoes(LOp)* que indica que se pretende um plano que não contempla as operações indicadas por *LOp*; *nao\_contem\_recursos(LRec)* que indica que se pretende um plano que não utilize os recursos indicados por *LRec*; *plano\_usado(PlanId)* que indica que determinado plano já foi usado; e *usar\_criterio(CritId)* que indica qual o critério de optimização que se pretende utilizar.

### **Mensagem de Resposta ao Pedido de Plano**

O Holon de Planeamento de Processos responde ao Holon de Tarefa, enviando uma mensagem do seguinte teor:

*r\_obter\_plano(PId, Attr, plano(Id, Id-Critério, Atributos-Plano, Operações))*

onde *PId* e *Attr* denotam os valores passados na mensagem *obter\_plano* e, o tuplo *plano/4* refere-se ao plano seleccionado para o fabrico do produto de acordo com a especificação dada.

### 6.2.1.2 Requisição de Serviços

Após o anúncio da tarefa, o *Holon de Tarefa* constrói a lista de recursos a contactar para cada operação do plano de fabrico e entra em contacto com esses *Holons de Recurso*, enviando-lhe uma mensagem *pedido* com informação sobre os recursos contactados para as operações a realizar imediatamente a montante e a jusante.

A Figura 6.6 (*adaptada de*: [Sousa *et al.*, 1999b]) apresenta as mensagens trocadas numa situação em que se tem um produto que para ser fabricado necessita de três operações (*op1*, *op2* e *op3*) sequenciais. Neste exemplo considera-se que existem dois recursos (*A1* e *A2*) passíveis de efectuar a operação *op1*.

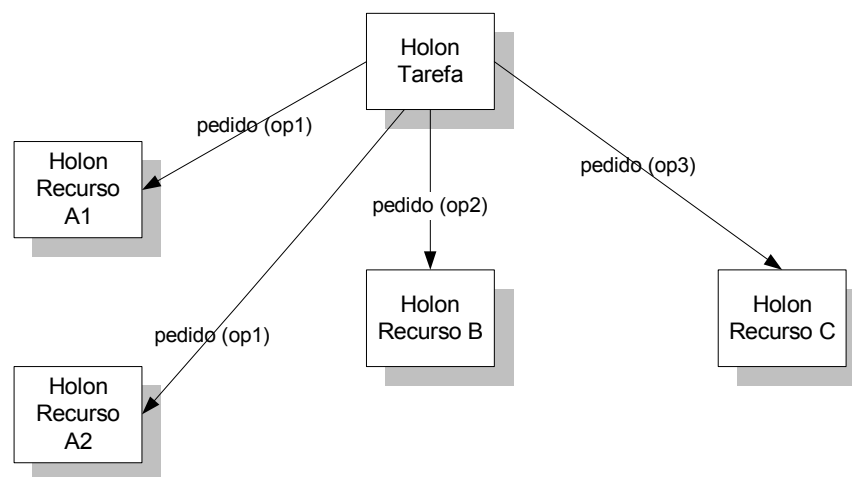


Figura 6.6 – Requisição de recursos

#### Mensagem de Requisição de Serviços

O Holon de Tarefa irá contactar cada Holon de Recurso para cada operação constante do plano com uma mensagem com o seguinte teor:

*pedido(DataLimite, OpId, Quant, JanelaTemporal, Pred, Succ)*

onde *DataLimite*, *OpId*, *Quant*, *JanelaTemporal*, *Pred* e *Succ* denotam respectivamente, a data considerada para conclusão da tarefa, a identificação da operação requerida, a quantidade de produtos a fabricar (*i.e.*, número de vezes que a operação tem que ser repetida), a janela temporal para o escalonamento, a lista de recursos contactados para a realização das operações imediatamente anteriores à que está a ser considerada no plano e, a lista de recursos contactados para as operações imediatamente posteriores a esta. Esta informação sobre antecessores e sucessores é necessária para os recursos poderem coordenar entre si os intervalos de tempo em que faz sentido escalonar cada operação.

### 6.2.1.3 Influência Directa

Esta é a primeira fase de propagação de restrições onde os recursos a montante influenciam os recursos a jusante, para a determinação do limite inferior da janela temporal.

Quando um *Holon de Recurso* recebe uma mensagem *pedido* verifica a sua agenda para as actividades já escalonadas e, constrói uma lista com os intervalos livres, onde é possível escalonar uma nova operação. Em seguida, os Holons de Recurso sem antecessores irão passar a lista de intervalos livres influenciada para os recursos que os sucedem (mensagem *influencia\_directa*) (Figura 6.7, adaptada de: [Sousa *et al.*, 1999b]). Esta lista influenciada é obtida através do deslocamento à direita dos intervalos livres em  $n$  unidades de tempo (correspondente ao tempo necessário para executar a operação requisitada).

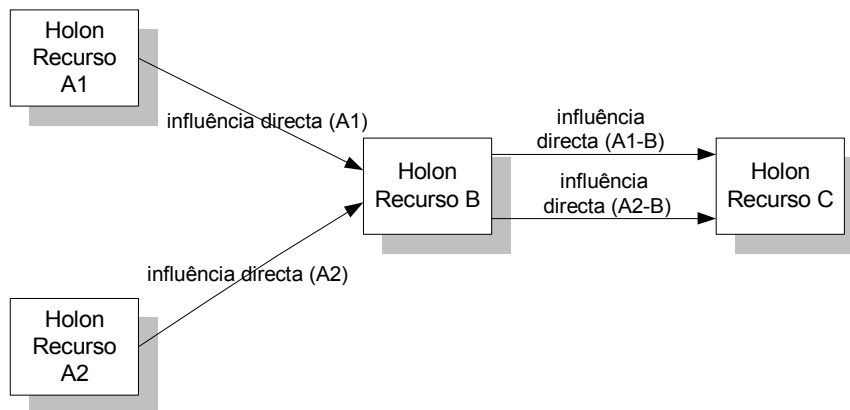


Figura 6.7 – Influência directa

Para cada mensagem *influencia\_directa* que um Holon de Recurso recebe dos recursos antecessores, o Holon de Recurso procederá a uma combinação entre a sua lista de intervalos livres e a lista de intervalos recebida, obtendo assim uma nova lista de intervalos, cujo limite inferior reflecte as restrições impostas por esses recursos. Essa nova lista será influenciada e passada para os recursos sucessores. Ao passar a lista influenciada de intervalos, cada holon passa também o caminho correspondente aos recursos já utilizados. A fase de influência directa termina quando todos os recursos sem sucessores receberem as listas influenciadas dos seus antecessores.

#### Mensagem de Influência Directa

Um Holon de Recurso envia a outro Holon de Recurso seu sucessor a lista influenciada de intervalos livres numa mensagem com o seguinte teor:

*influencia\_directa(TId, OpId, Dur, LIntervalos, Caminho)*

onde *TId*, *OpId*, *Dur*, *LIntervalos* e *Caminho* denotam respectivamente, o identificador do Holon de Tarefa, o identificador da operação requisitada ao Holon de Recurso receptor da



mensagem, a duração da operação requisitada ao holon emissor e, a lista influenciada de intervalos livres desse recurso e a lista de todos os recursos antecessores.

#### 6.2.1.4 Influência Inversa e Proposta

Nesta segunda fase de propagação de restrições, os recursos a jusante influenciam os recursos a montante para a determinação do limite superior (efectivo) da janela temporal. A fase de influência inversa é semelhante à influência directa, mas no sentido oposto.

Após combinarem a sua agenda com a lista de intervalos recebida dos antecessores, os Holons de Recurso sem sucessores irão dar início à fase de influência inversa. Estes holons irão passar a lista de intervalos livres influenciada para os recursos que os precedem (mensagem *influencia\_inversa*) (Figura 6.8, adaptada de: [Sousa *et al.*, 1999b]). Esta lista influenciada é obtida através do deslocamento à esquerda dos intervalos livres em  $n$  unidades de tempo (correspondente à duração da operação requisitada).

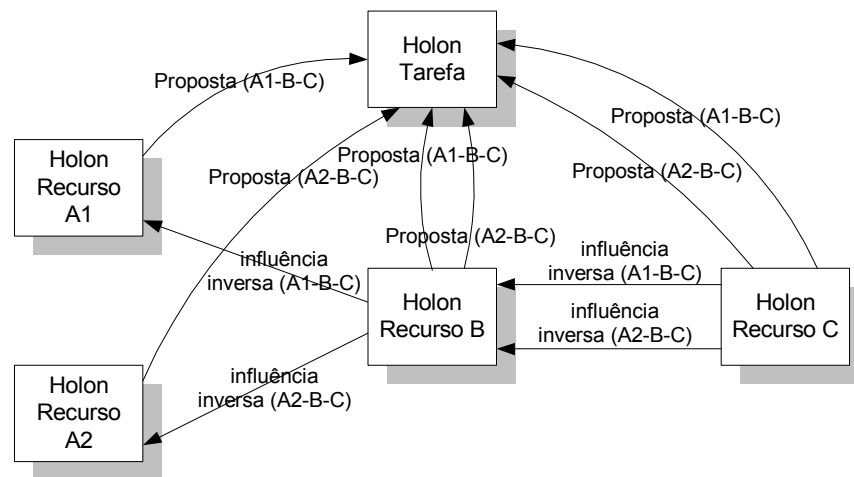


Figura 6.8 – Influência inversa e ofertas

Para cada mensagem *influencia\_inversa* recebida com uma lista de intervalos de tempo possíveis de utilização, cada Holon de Recurso irá combinar essa lista de intervalos com a sua própria lista de forma a determinar o limite inferior (efectivo) de escalonamento, gerando assim a lista final de intervalos livres. Após influenciar essa lista e a passar aos recursos utilizados nas operações precedentes, o Holon de Recurso enviará a proposta de escalonamento ao Holon de Tarefa (mensagem *proposta*). É enviada uma proposta para cada combinação possível de recursos para executar o plano de produção. O envio da lista influenciada aos recursos antecessores leva em consideração o caminho por onde a mensagem chegou (ou seja, uma mensagem que tenha sido originada pela combinação de recursos *A1-B-C* na fase de influência directa irá passar apenas

por esses recursos na fase de influência inversa). Esta fase termina quando todos os recursos sem antecessores receberem as listas influenciadas dos respectivos sucessores.

### **Mensagem de Influência Inversa**

Um Holon de Recurso envia a outro Holon de Recurso seu antecessor a lista influenciada de intervalos livres numa mensagem com o seguinte teor:

*influencia\_inversa(TId, OpId, LIntervalos, Caminho)*

onde *TId*, *OpId*, *LIntervalos* e *Caminho* denotam respectivamente, o identificador do Holon Tarefa, o identificador da operação requisitada ao Holon de Recurso receptor da mensagem, a lista influenciada de intervalos livres desse recurso e, a combinação de recursos que deu origem à solução que esta mensagem de influência inversa materializa.

### **Mensagem de Proposta de Escalonamento**

Quando um Holon de Recursos elabora a lista final de intervalos livres, envia essa proposta ao Holon de Tarefa com uma mensagem com o seguinte teor:

*proposta(OpId, Duração, LIntervalos, Custo)*

onde *OpId*, *Duração*, *LIntervalos* e *Custo* denotam respectivamente, a identificação da operação requisitada, a duração do intervalo de tempo necessário para fabricar a quantidade requisitada de produtos, a lista de intervalos livres onde é possível escalonar e, o custo que este recurso atribui à execução daquela operação.

#### **6.2.1.5 Contratação**

Após receber as propostas de escalonamento de cada um dos Holons de Recurso contactados, o Holon de Tarefa analisa as propostas e escolhe a combinação de recursos a utilizar (caso existam recursos alternativos para uma ou mais operações) e qual o intervalo de tempo mais indicado para a utilização de cada um dos recursos. Após a selecção o Holon de Tarefa envia uma mensagem a cada Holon de Recurso indicando a decisão tomada quanto à proposta recebida (Figura 6.9, *adaptada de*: [Sousa *et al.*, 1999b]).

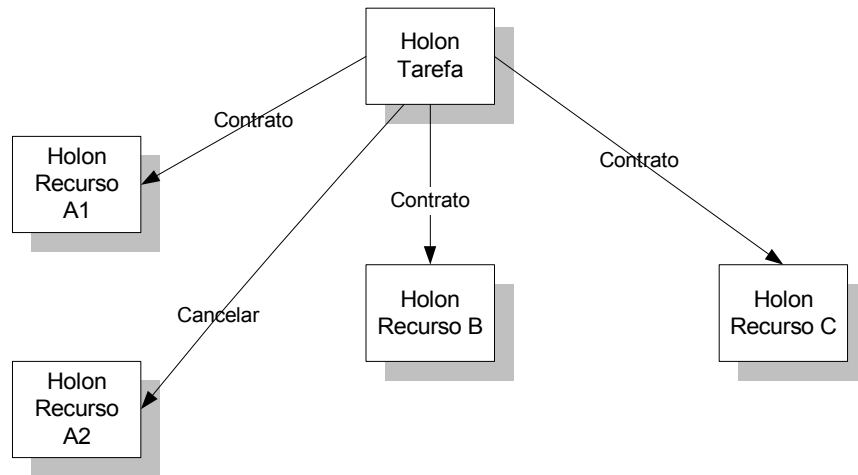


Figura 6.9 – Conclusão da negociação

A selecção da combinação de recursos a usar (bem como do intervalo de tempo a utilizar) é baseada em heurísticas (*e.g.*, maior folga) e no custo total da solução para a tarefa a executar com base no custo de cada proposta recebida e escolhida – equação (6.2).

$$custo-da-solução = \sum_{i=1}^n custo(S(i)) \quad (6.2)$$

onde,  $n$  é o número de operações do plano,  $S(i)$  é uma função que devolve a proposta seleccionada para efectuar a operação  $i$  e  $custo()$  é uma função que devolve o custo de uma proposta.

#### Mensagem de Contratação

O Holon de Tarefa comunica a aceitação da proposta a um Holon de Recurso com uma mensagem com o seguinte teor:

*contrata(OpId, IntervaloSeleccionado, PredSel, SuccSel)*

onde *OpId*, *IntervaloSeleccionado*, *PredSel* e *SuccSel* denotam respectivamente, a identificação da operação anteriormente requisitada, o intervalo seleccionado pela tarefa de entre os intervalos propostos pelo recurso, a lista de recursos seleccionados para as operações antecessoras e, a lista de recursos seleccionados para as operações sucessoras.

#### Mensagem de Cancelamento de Requisição

O Holon de Tarefa comunica a rejeição da proposta de um Holon de Recurso com uma mensagem com o seguinte teor:

*cancela(OpId)*

onde *OpId* denota a operação anteriormente requisitada.

## 6.2.2 Tratamento de Conflitos

Durante a execução simultânea do protocolo por vários Holons de Tarefa pode surgir um conflito quando mais do que uma tarefa requisita a realização de uma operação a um dado recurso em janelas temporais sobrepostas [Sousa e Ramos, 1998] [Sousa e Ramos, 1999a].

Por exemplo, considere-se um cenário em que há duas tarefas a executar,  $T_1$  e  $T_2$ , e dois recursos,  $R_A$  e  $R_B$ , no qual  $T_1$  inicia a negociação de  $op_1$  com  $R_A$  e  $R_B$ . Tanto  $R_A$  como  $R_B$  analisam as suas agendas e encontram um intervalo livre para escalonar  $op_1$  informando  $T_1$  desse intervalo.  $T_1$  analisa então as propostas de  $R_A$  e  $R_B$  para decidir a qual dos recursos atribuir a execução de  $op_1$ . Entretanto,  $R_A$  recebe um pedido de  $T_2$  para uma outra operação ( $op_2$ ). Suponha-se agora que a única maneira de executar  $op_2$  cumprindo o prazo imposto por  $T_2$  passa por utilizar parte do intervalo proposto a  $T_1$ .  $R_A$  tem um *Problema de Indecisão*, pois ainda não recebeu resposta de  $T_1$ , confirmando ou não a utilização do intervalo proposto.

Dois senões podem ocorrer:

1.  $R_A$  considera o intervalo como reservado para a execução de  $op_1$  de  $T_1$  informando  $T_2$  que não pode executar  $op_2$ . Caso  $T_1$  decida contratar  $R_B$ , o recurso  $R_A$  perde ambas as oportunidades de contrato quando seria possível executar  $op_2$ , visto que o intervalo foi rejeitado por  $T_1$ .
2.  $R_A$  considera o intervalo livre e informa  $T_2$  que pode executar  $op_2$ . Caso  $T_1$  e  $T_2$  decidam contratar  $R_A$  o recurso fica comprometido com a execução de duas operações, mas vai falhar a execução de uma delas devido à impossibilidade física de executar duas operações em simultâneo.

Este tipo de problema é denominado Problema de Indecisão podendo ser solucionado de uma forma simples, que passa por acrescentar alguns procedimentos extra à cabeça do protocolo. A Figura 6.10 (*adaptada de*: [Sousa *et al.*, 1999b]) apresenta a nova sequência de mensagens trocadas na fase de anúncio, para contornar o problema de indecisão.

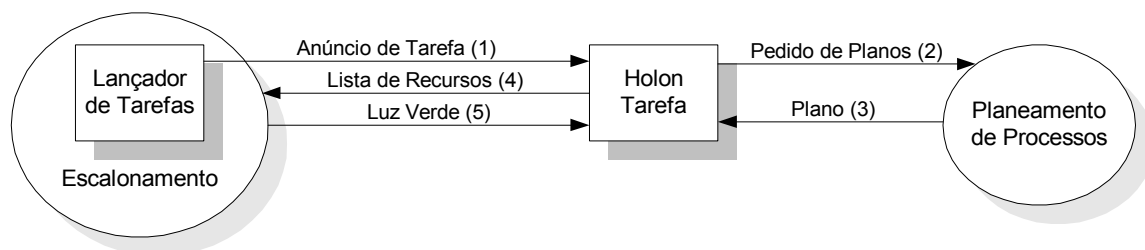


Figura 6.10 – Alteração ao PRCPR: tratamento de conflitos

Os passos (1), (2) e (3) são exactamente os mesmos descritos na secção 6.2.1.1. Após receber o plano de produção a utilizar o *Holon de Tarefa* constrói a lista de recursos a contactar, mas antes de negociar com cada recurso, envia essa lista ao *Holon de Escalonamento* (mensagem *lista\_de\_recurso*s).

Para evitar conflitos o Holon de Escalonamento possui uma lista de recursos em negociação e uma outra lista de tarefas em espera ordenada por prioridade. O Holon de Escalonamento conhece as janelas temporais que estão a ser negociadas para cada recurso, colocando em espera as tarefas que desejam aceder a esses recursos, caso possuam uma janela temporal que se sobreponha às janelas temporais em negociação. Este mecanismo permite a negociação em simultâneo da execução de várias tarefas, apenas impedindo a negociação simultânea de tarefas em janelas temporais sobrepostas. Na prática, este mecanismo é semelhante ao de exclusão mútua na programação concorrente.

A prioridade para a execução de uma tarefa  $T$  é dada pela equação (6.3), baseada no valor total das encomendas que deram origem a essa tarefa ( $\varepsilon_i$ ), bem como na prioridade associada aos clientes dessas encomendas.  $\Delta(v)$  denota uma operação de normalização, aplicada a valores monetários, por forma a que a adição de montantes (monetários) ligados às prioridades associadas à execução de encomendas, faça sentido.

$$prioridade(T) \stackrel{\text{def}}{=} \Delta \left[ \sum_i total(\varepsilon_i) \right] + \Delta \left[ \sum_i prioridade(cliente(\varepsilon_i)) \right] \quad (6.3)$$

Quando não há possibilidade de conflitos o Holon de Escalonamento sinaliza os Holons de Tarefa em espera que podem iniciar a negociação, através de uma mensagem *luz\_verde*.

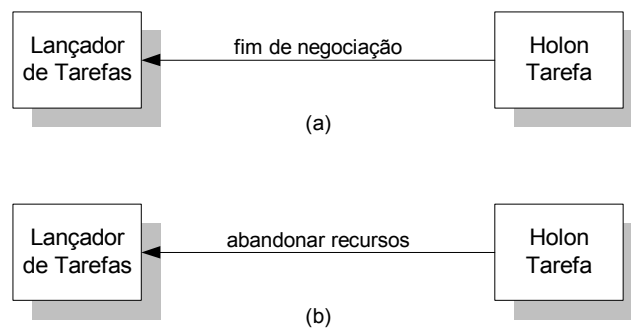


Figura 6.11 – Alterações ao PRCPR: (a) fim de negociação; (b) abandono de recursos

Para actualizar a lista de recursos em negociação o Holon de Tarefa deve indicar ao Holon de Escalonamento, no final da execução do protocolo, que concluiu a negociação (mensagem *fim\_de\_negociação*, Figura 6.11a). tendo em vista a optimização do processo, o Holon de Tarefa pode também durante a execução do protocolo informar o Holon de Escalonamento de qualquer

recurso que afinal já não vai contactar (mensagem *abandonar\_recursos*, Figura 6.11b, *adaptada de* [Sousa e Ramos, 1998]).

#### **Mensagem com a Lista de Recursos**

O Holon de Tarefa avisa o Holon de Escalonamento sobre quais os recursos que vai contactar com uma mensagem com o seguinte teor:

*lista\_de\_recursos(LRecursos, JT)*

onde *LRecursos* denota a lista de Holons de Recurso que esta Tarefa vai contactar e *JT* denota a janela temporal que vai ser negociada.

#### **Mensagem de “Luz Verde”**

O Holon de Escalonamento comunica ao Holon de Tarefa que pode iniciar a negociação com a seguinte mensagem:

*luz\_verde*

#### **Mensagem de Fim de Negociação**

O Holon de Tarefa avisa o Holon de Escalonamento que terminou a negociação com a seguinte mensagem:

*fim\_de\_negociação*

#### **Mensagem de Abandono de Recursos**

O Holon de Tarefa avisa o Holon de Escalonamento que não vai entrar em contacto com determinados recursos usando uma mensagem com o seguinte teor:

*abandonar\_recursos(LRecursos)*

onde *LRecursos* denota a lista de Holons de Recurso que esta Tarefa já não vai contactar.

### **6.2.3 Tratamento de Excepções**

As fases descritas anteriormente correspondem à execução do protocolo para a contratação de serviços entre tarefas e recursos. No entanto, após o estabelecimento do contrato, as condições do sistema podem mudar de tal forma que seja necessário renegociar o escalonamento (*e.g.*, avaria de um recurso, entrada de tarefa prioritária).

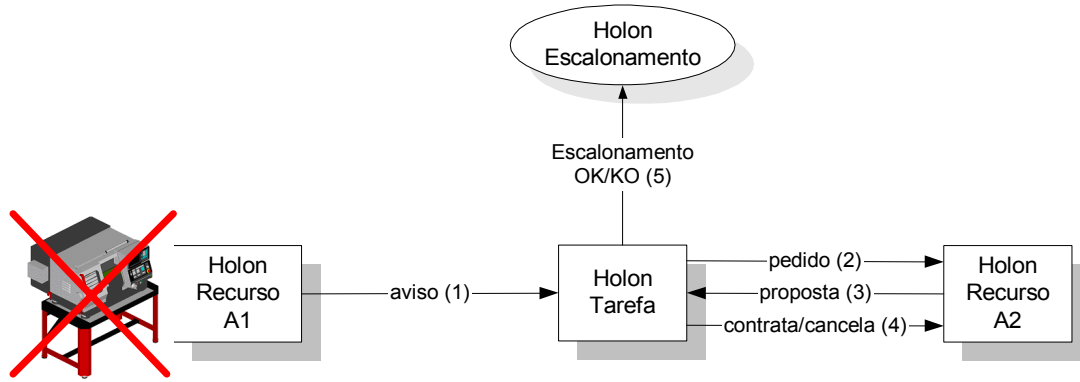


Figura 6.12 – Exemplo de avaria de um recurso

Quando um *Holon de Recurso* detecta, por exemplo, uma avaria que não pode ser recuperada (em tempo útil), informa os *Holons de Tarefa* que o tenham contratado da impossibilidade de cumprir o contrato. Após receber a mensagem de *aviso* do Holon de Recurso, o Holon de Tarefa vai tentar escalonar as operações em causa com outros Holons de Recurso (Figura 6.12, *fonte*: [Sousa e Ramos, 1999a]). Em certas situações, o Holon de Recurso pode não informar os Holons de Tarefa da avaria e tentar recuperá-la, executando as operações contratadas com um pequeno atraso e/ou executando-as de forma a recuperar o tempo perdido (*e.g.*, utilização de uma maior velocidade de operação).

## 6.2.4 Análise da Complexidade do Protocolo

Nesta secção é efectuada uma análise ao protocolo acabado de descrever, do ponto de vista de complexidade do problema (número de soluções), número de mensagens trocadas e tamanho total das mensagens trocadas.

### Complexidade do Problema de Escalonamento

O problema de escalonamento com  $n$  operações em  $m$  máquinas é um problema NP-completo que resulta numa explosão combinatória do espaço de soluções. O protocolo descrito vai gerando as várias combinações possíveis de recursos durante as fases de influência (*i.e.*, o número de planos de produção alternativos tendo em conta os vários recursos capazes de efectuar cada operação).

Tendo em conta que um plano  $\mathcal{P}$  é um grafo constituído por uma sequência de operações ou por várias sequências que ocorrem em paralelo [Ramos *et al.*, 1998] [Rocha, 1999], que pode ser descrito pela seguinte gramática BNF:

$$\begin{aligned}
 PLANO &::= PAR \mid SEQ \\
 PAR &::= par(Lista-SEQ) \\
 SEQ &::= seq(Lista-ARGSEQ)
 \end{aligned}$$

$$\begin{aligned}
 \text{Lista-SEQ} &::= \text{SEQ} \mid \text{SEQ}, \text{Lista-SEQ} \\
 \text{Lista-ARGSEQ} &::= \text{ARGSEQ} \mid \text{ARGSEQ}, \text{Lista-ARGSEQ} \\
 \text{ARGSEQ} &::= \text{Operação} \mid \text{PAR}
 \end{aligned}$$

Sabendo-se que o número de combinações num dos ramos sequenciais é dado pela equação (6.4), e o número de combinações em ramos paralelos dado pela equação (6.5), em que  $n$  é o número de operações no ramo sequencial em questão,  $R(i)$  é uma função que devolve o número de recursos capazes de executar a operação  $i$  daquele ramo,  $k$  é o número de ramos paralelos, e  $\text{ramo}(i)$  é uma função que devolve cada um dos ramos do grafo em questão.

$$\prod_{i=1}^n R(i) \quad (6.4)$$

$$\sum_{i=1}^k \chi(\text{ramo}(i)) \quad (6.5)$$

O número de combinações possíveis, denotado  $\chi(\mathcal{P})$ , é então uma consequência lógica do programa em lógica que a seguir se apresenta, no qual o predicado  $n\_combinações(\text{PlanoBNF}, N)$  denota  $\chi(\text{transformar}(\mathcal{P}))$ , sendo que  $\text{transformar}(\mathcal{P})$  é uma função que constrói uma representação do plano  $\mathcal{P}$  na gramática referida em epígrafe.

$$\begin{aligned}
 n\_combinações(\text{par}(\text{LSeq}), N) &\leftarrow \\
 &\quad n\_combinações\_par(\text{LSeq}, N) \\
 n\_combinações(\text{seq}(\text{LArgSeq}), N) &\leftarrow \\
 &\quad n\_combinações\_seq(\text{LArgSeq}, N) \\
 n\_combinações\_par([], 0) & \\
 n\_combinações\_par([\text{seq}(\text{LArgSeq}) \mid T], N) &\leftarrow \\
 &\quad n\_combinações(\text{seq}(\text{LArgSeq}), X) \wedge \\
 &\quad n\_combinações\_par(T, Z) \wedge \\
 &\quad \text{atribui}(N, X + Z) \\
 n\_combinações\_seq([], 1) & \\
 n\_combinações\_seq([\text{par}(\text{LSeq}) \mid T], N) &\leftarrow \\
 &\quad n\_combinações(\text{par}(\text{LSeq}), X) \wedge \\
 &\quad n\_combinações\_seq(T, Z) \wedge \\
 &\quad \text{atribui}(N, X * Z) \\
 n\_combinações\_seq([Op \mid T], N) &\leftarrow \\
 &\quad n\_rec\_op(Op, X) \wedge \\
 &\quad n\_combinações\_seq(T, Z) \wedge \\
 &\quad \text{atribui}(N, X * Z)
 \end{aligned}$$

A complexidade do problema traduz-se então na análise de complexidade do programa anterior. O melhor caso ocorre quando se tem uma situação onde o plano de produção  $\mathcal{P}$  contempla a execução de uma única operação, sendo portanto  $\chi(\mathcal{P})$  igual ao número de recursos capazes de efectuar essa operação – vide (6.6). Um dos casos particulares ocorre quando só existe



um recurso para a execução de todas as operações do plano, havendo portanto apenas uma combinação possível – *vide* (6.7). No pior dos casos, o plano  $\mathcal{P}$  é dado por uma sequência de  $n$  operações em que cada operação possui um número igual de recursos capazes de a efectuar, sendo por isso o número de combinações possíveis igual a  $r^n$  – *vide* (6.8).

$$n = 1 \Rightarrow \chi(\mathcal{P}) = R(1) \quad (6.6)$$

$$\forall(i) R(i) = 1 \Rightarrow \chi(\mathcal{P}) = 1 \quad (6.7)$$

$$(\forall(i) R(i) = r \wedge r > 1) \Rightarrow \chi(\mathcal{P}) = \prod_{i=1}^n r \Leftrightarrow \chi(\mathcal{P}) = r^n \quad (6.8)$$

A medida de complexidade do problema para o pior caso é dada por  $O(r^n)$ <sup>21</sup> ou, mais genericamente, por  $O(\max(R(i))^n)$ , onde  $\max(R(i))$  denota a função que devolve o valor máximo da função  $R(i)$  aplicada ao plano  $\mathcal{P}$ .

### Complexidade do Protocolo

O número total de mensagens trocadas para uma execução do Protocolo de Rede de Contrato com Propagação de Restrições depende directamente do número de combinações existentes e é dado pela equação (6.9), onde  $n$  é o número de operações do plano e  $\psi(\mathcal{P})$  é o número de mensagens trocadas durante a fase de influência directa ou inversa, dado pela equação (6.10).

$$M_{prepr} = \left( \sum_{i=1}^n R(i) \right) + [2 \times \psi(\mathcal{P})] + [2 \times (\psi(\mathcal{P}) + \chi(\mathcal{P}))] \quad (6.9)$$

$$\psi(\mathcal{P}) = \sum_{i=1}^{n-1} (R(i) \times \chi(\text{truncar}(\mathcal{P}, i)) \times R(\text{succ}(i))) \quad (6.10)$$

O primeiro termo desta adição corresponde ao número de mensagens enviadas para requisição de serviço, o segundo termo corresponde ao número de mensagens trocadas na fase de influência directa e inversa (uma mensagem de influência para cada recurso a aplicar de seguida (sucessor) por cada mensagem de influência recebida de um recursos antecessor) e o último termo corresponde ao número de mensagens de proposta dos Holons de Recurso e resposta do Holon de Tarefa (uma proposta por cada mensagem de influência recebida e uma resposta por cada proposta). A função  $\text{truncar}(\mathcal{P}, i)$  constrói um plano baseado em  $\mathcal{P}$  truncado até à operação  $i$  (exclusive), ou seja, um plano composto por todos os ramos de operações antecessoras da operação  $i$ . No caso de operações sem antecessoras devolve uma sequência vazia (já que

<sup>21</sup> A notação  $O$  permite indicar a complexidade de um algoritmo representado por uma função  $f(n)$ . Mais propriamente,  $f(n) = O(g(n))$  quer dizer que  $c \cdot g(n)$  é um limite superior para  $f(n)$ . Ou seja, existe uma constante  $c$  tal que  $f(n) \leq c \cdot g(n)$  para valores de  $n$  suficientemente grandes [Skiena, 1997].

$\chi(seq(\emptyset)) = 1$ , elemento neutro da multiplicação), e  $succ(i)$  denota a função que devolve a operação que sucede à operação  $i$ .

A Figura 6.13 apresenta três planos de produção, utilizados como exemplo para análise da complexidade do PRCPR. Na figura as operações são representadas por círculos tendo na parte superior a identificação da operação e, na parte inferior, o número de recursos capazes de efectuar essa operação. Na Tabela 6.1 são apresentados os valores de  $\chi(\mathcal{P})$ ,  $\psi(\mathcal{P})$  e  $M_{prcpr}$  para cada um desses planos.

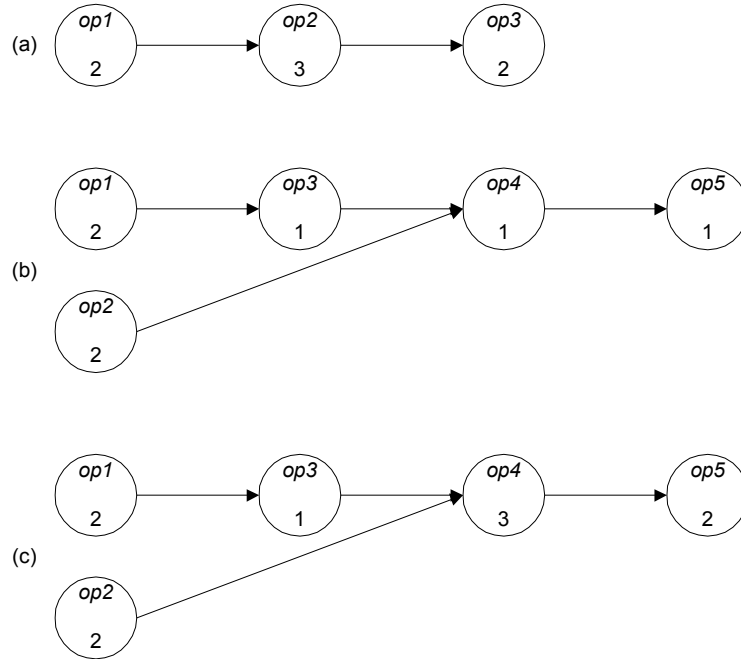


Figura 6.13 – Exemplos de planos para estudo da complexidade do protocolo RCPR

Tabela 6.1 – Valores de  $\chi(\mathcal{P})$  e  $M_{prcpr}$  para os planos exemplo

Plano	Termo	Valor
(a)	$\chi(\mathcal{P})$	$2 \times 3 \times 2 = 12$
	$\psi(\mathcal{P})$	$2 \times 1 \times 3 + 3 \times 2 \times 2 = 18$
	$M_{prcpr}$	$(2+3+2) + 2 \times 18 + 2 \times (18 + 12) = 7+36+60 = \mathbf{103}$
(b)	$\chi(\mathcal{P})$	$[(2 \times 1 \times 1) + (2 \times 1)] \times 1 = 4$
	$\psi(\mathcal{P})$	$2 \times 1 \times 1 + 2 \times 1 \times 1 + 1 \times 2 \times 1 + 1 \times 4 \times 1 = 10$
	$M_{prcpr}$	$(2+2+1+1+1) + 2 \times 10 + 2 \times (10 + 4) = 7+20+28 = \mathbf{55}$

Plano	Termo	Valor
(c)	$\chi(\mathcal{P})$	$[(2 \times 1 \times 3) + (2 \times 3)] \times 2 = 24$
	$\psi(\mathcal{P})$	$2 \times 1 \times 1 + 2 \times 1 \times 3 + 1 \times 2 \times 3 + 3 \times 4 \times 2 = 38$
	$M_{prcpr}$	$(2+2+1+3+2) + 2 \times 38 + 2 \times (38 + 24) = 10+76+124 = \mathbf{210}$

Conforme se pode verificar, o número de mensagens trocadas é elevado, mesmo para planos relativamente simples e com poucos recursos alternativos. Em cenários reais é teoricamente possível executar todas as operações em cada um dos recursos existentes na instalação fabril (devido à polivalência dos recursos), e além disso, os planos de produção consistem normalmente em dezenas ou centenas de operações.

Conforme foi dito anteriormente, o pior caso para a execução do algoritmo ocorre quando se tem um plano sequencial de  $n$  operações com  $r$  recursos para cada operação. Assim sendo,  $M_{prcpr}$  é majorada da seguinte forma:

$$M_{prcpr} \leq n \cdot r + 2 \times 2 \times r^n + 2 \times (2 \cdot r^n + r^n) \Leftrightarrow M_{prcpr} \leq n \cdot r + 10r^n$$

A complexidade do PRCPR (no que toca ao número de mensagens trocadas) para o pior caso é  $O(r^n)$ , pois a componente linear  $n \cdot r$  pode ser ignorada para valores de  $n$  suficientemente grandes, devido à componente exponencial  $r^n$ .

Devido à explosão combinatória do problema e à correspondência do envio de uma mensagem de influência para cada mensagem de influência recebida (o que torna o número de mensagens trocadas directamente relacionado com  $\chi(\mathcal{P})$ ), é lícito considerar como hipótese uma outra abordagem, em que se diminua o número de mensagens trocadas.

#### **Alteração ao Protocolo na Tentativa de Diminuir o Número de Mensagens Trocadas**

A natureza distribuída da arquitectura favorece um protocolo de interacção semelhante ao descrito, onde cada mensagem recebida é processada e dá origem a uma ou mais respostas (granulosidade fina). No entanto, devido à explosão combinatória de soluções para o problema, o protocolo pode ser modificado, de modo a que cada recurso envie apenas uma mensagem com todas as combinações, ao invés de enviar uma mensagem por combinação [Sousa *et al.*, 1999b]. Trata-se então de equacionar uma outra solução, onde se diminui o número de mensagens, mas aumenta-se necessariamente o tamanho das mensagens trocadas (granulosidade grossa).

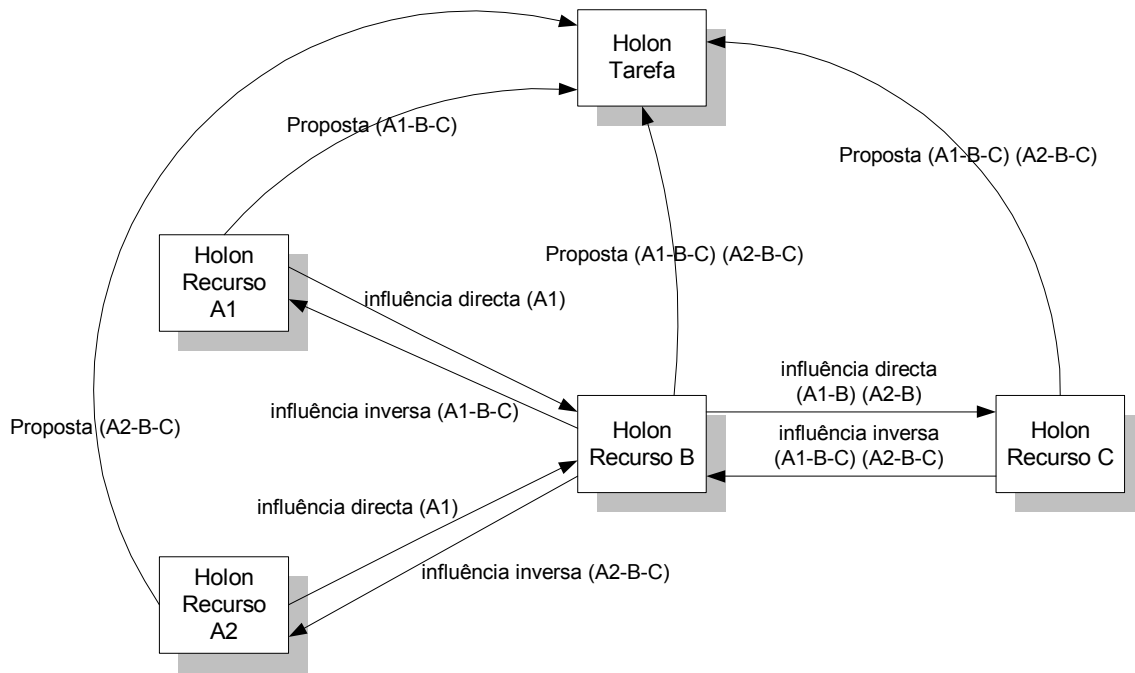


Figura 6.14 – Alterações na fase de influência directa, influência inversa e proposta

A Figura 6.14 apresenta as fases de influência directa, inversa e proposta para esta variante do PRCPR. Conforme se pode observar na figura, apenas é trocada uma mensagem por todas as mensagens recebidas na fase de influência directa. Na fase de influência inversa o número de mensagens enviadas corresponde ao número de mensagens recebidas na fase de influência directa. Além disso, o número de mensagens de proposta é agora apenas de uma por Holon de Recurso.

As mensagens de influência directa, influência inversa e proposta passam a ter os teores apresentados em seguida:

*influencia\_directa*(TId, OpId, LComb)

*influencia\_inversa*(TId, OpId, LComb)

*proposta*(OpId, Duração, LCombProposta)

onde *TId*, *OpId* e *Duração* denotam respectivamente, o identificador do Holon Tarefa, o identificador da operação requisitada ao holon de recurso receptor da mensagem, e a duração do intervalo de tempo necessário para executar as quantidades indicadas de produtos. O atributo *LComb* denota, por seu lado, uma lista de pares (*LIntervalos*, *Caminho*) em que *LIntervalos* é a lista influenciada de intervalos livres para esse recurso e *Caminho* é a lista de todos os recursos antecessores, e o atributo *LCombProposta* denota uma lista de tuplos na forma (*LIntervalos*, *Caminho*, *Custo*) onde *LIntervalos* é a lista de intervalos livres onde é possível escalonar e *Custo* representa o custo que este recurso atribui à execução dessa operação.

### Complexidade da Variante do Protocolo RCPR

O número de mensagens trocadas na execução desta variante do PRCPR é representado por  $M_{prcpr}^2$  e calculado pela equação (6.11).

$$M_{prcpr}^2 = 3 \times \left( \sum_{i=1}^n R(i) \right) + 2 \times \left( \sum_{i=1}^{n-1} (R(i) \times R(succ(i))) \right) \quad (6.11)$$

O número de mensagens de proposta e de resposta é agora igual ao número de requisições (daí o factor de multiplicação três (3) no primeiro termo da equação (6.11)). O número de mensagens nas fases de influência directa e inversa fica neste caso apenas dependente do número de recursos capazes de efectuar cada operação e do número de recursos capazes de efectuar as operações sucessoras.

Na Tabela 6.2 são apresentados os valores de  $M_{prcpr}^2$  para os planos tomados como exemplo e dados pela Figura 6.13. Conforme se pode verificar, o número de mensagens trocadas é agora substancialmente menor.

Tabela 6.2 – Valores de  $M_{prcpr}^2$  para planos exemplo

Plano	$M_{prcpr}^2$
(a)	$3 \times (2+3+2) + 2 \times (2 \times 3 + 3 \times 2) = 21+24 = \mathbf{48}$
(b)	$3 \times (2+2+1+1+1) + 2 \times (2 \times 1 + 2 \times 1 + 1 \times 1 + 1 \times 1) = 21+12 = \mathbf{33}$
(c)	$3 \times (2+2+1+3+2) + 2 \times (2 \times 1 + 2 \times 3 + 1 \times 3 + 3 \times 2) = 30+34 = \mathbf{64}$

A Figura 6.15 apresenta em forma gráfica a evolução do número de mensagens trocadas com o aumento do número de operações de um plano. Os dados correspondem a uma situação em que se tem um plano composto por uma sequência de  $n$  operações, em que cada operação possui  $r$  ( $= 2$ ) recursos capazes de a efectuar<sup>22</sup>. O gráfico utiliza uma escala logarítmica, em que  $M_{prcpr}$  é dado pela linha com marcas circulares,  $M_{prcpr}^2$  pela linha com marcas quadradas e  $\chi(\mathcal{P})$  pela linha mais esbatida.

<sup>22</sup> No Apêndice B é apresentado o modelo matemático e respectiva folha de cálculo utilizada para a geração dos dados deste gráfico e do da Figura 6.16.

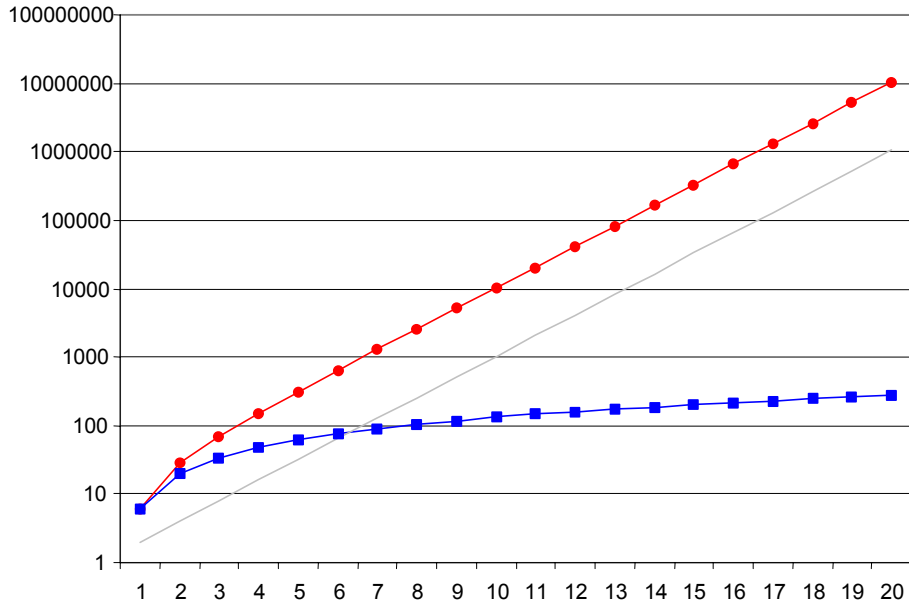


Figura 6.15 – Evolução do número de mensagens com o aumento de operações

Conforme se pode ver  $\chi(\mathcal{P})$  e  $M_{prcpr}$  crescem exponencialmente com o número de operações, enquanto que  $M_{prcpr}^2$  cresce linearmente. Daqui se pode observar que a variante do PRCPR coloca uma carga menor no sistema em termos de número de mensagens trocadas ( $M_{prcpr}^2 \ll M_{prcpr}$ ).

Como exemplo ilustrativo, a razão  $\frac{M_{prcpr}^2}{M_{prcpr}}$  para  $n = 10$  é de 0,07 enquanto que para  $n = 20$  é de  $2.59 \times 10^{-5}$ .

Conforme foi dito anteriormente, o pior caso na utilização do algoritmo ocorre quando se tem um plano sequencial com  $n$  operações e com  $r$  recursos para cada operação. Nesta situação,  $M_{prcpr}^2$  é majorada da seguinte forma:

$$M_{prcpr}^2 \leq 3 \cdot n \cdot r + 2 \times (n-1) \times r^2 \Leftrightarrow M_{prcpr}^2 \leq 3nr + 2nr^2 \Leftrightarrow M_{prcpr}^2 \leq n \cdot (3r + 2r^2)$$

A medida de complexidade da variante do PRCPR é dada por  $O(n)$ , pois não só  $r$  é constante, como é sempre possível encontrar uma constante  $c > (3r + 2r^2)$ .

### Cálculo do Tamanho Total das Mensagens Trocadas

Para comparar as duas abordagens ao PRCPR vai-se então efectuar o cálculo do comprimento das mensagens trocadas num e noutro caso. Para simplificar, assume-se que o tamanho da lista influenciada de intervalos, o caminho e a lista de propostas é sempre igual para qualquer par  $\langle \text{recurso}, \text{operação} \rangle$  (uma situação que pode teoricamente acontecer quando todos os recursos

possuem a agenda de actividades completamente livre). O tamanho da mensagem de influência directa ou inversa é então dado pela equação (6.12) e o tamanho da mensagem de proposta pela equação (6.13), onde  $Cab$ ,  $LdI$ ,  $Cam(n)$  e  $LdP$  denotam respectivamente, o tamanho do cabeçalho da mensagem, o tamanho da lista de intervalos influenciados, o tamanho da lista com a combinação de recursos de uma solução e, o tamanho da lista com os intervalos propostos.

$$T_{inf} = Cab + LdI + Cam(n) \quad (6.12)$$

$$T_{bid} = Cab + LdP \quad (6.13)$$

Para a variante do PRCPR, o comprimento da mensagem de influência directa ou inversa é dado pela equação (6.14) e o da mensagem de proposta pela equação (6.15) onde  $\emptyset$  denota o conjunto vazio.

$$T_{inf}^2(i) = Cab + \chi(trunc(\mathcal{P}, i)) \times (LdI + Cam(n)) \quad (6.14)$$

$$T_{bid}^2(i) = \begin{cases} Cab + \chi(\mathcal{P}) \times LdP & succ(i) \neq \emptyset \\ Cab + \chi(trunc(\mathcal{P}, i)) \times LdP & succ(i) = \emptyset \end{cases} \quad (6.15)$$

O comprimento de todas as mensagens trocadas nas fases de influência inversa e directa e proposta no PRCPR é então dado pela equação (6.16) e, no caso da variante do PRCPR, pela equação (6.17).

$$T_{prcpr|inf+bid|} = 2 \times \psi(\mathcal{P}) \times T_{inf} + (\psi(\mathcal{P}) + \chi(\mathcal{P})) \times T_{bid} \quad (6.16)$$

$$T_{prcpr|inf+bid|}^2 = 2 \times \left( \sum_{i=1}^{n-1} (R(i) \times R(succ(i)) \times T_{inf}^2(i)) \right) + \left( \sum_{i=1}^n R(i) \times T_{bid}^2(i) \right) \quad (6.17)$$

A Figura 6.16 apresenta em modo gráfico uma comparação entre  $T_{prcpr|inf+bid|}$  (barras claras) e  $T_{prcpr|inf+bid|}^2$  (barras escuras). O eixo primário do gráfico (lado esquerdo) tem uma escala logarítmica e denota o comprimento das mensagens, enquanto que o eixo secundário (do lado direito) tem uma escala linear e denota a razão  $\frac{T_{prcpr|inf+bid|}^2}{T_{prcpr|inf+bid|}}$  (linha escura).

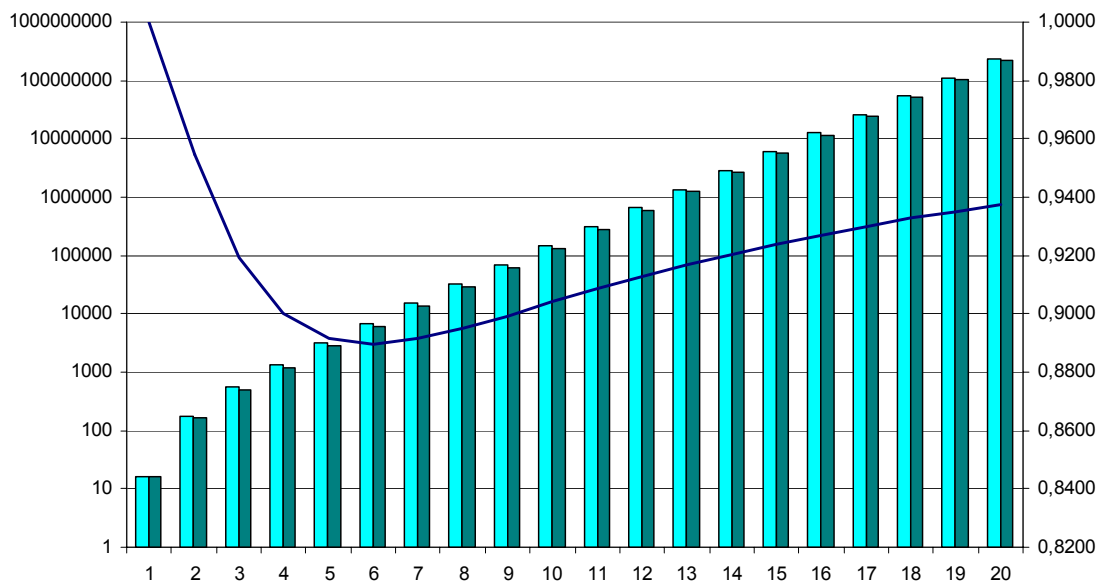


Figura 6.16 – Tamanho total de mensagens

O gráfico corresponde a valores encontrados para um plano de produção sequencial, de  $n$  operações com um número igual de recursos  $r (=2)$  para efectuar cada operação, e considerando-se  $Cab = 2$ ,  $LdI = 10$ ,  $Cam(n) = 2 \cdot (n-1)$  e  $LdP = 6$ .

Daqui se pode verificar que o comprimento das mensagens trocadas na variante do PRCPR é menor que no original ( $T_{prcpr|inf+bid}^2 < T_{prcpr|inf+bid}$ ). Também se verifica que a razão entre os termos

das duas séries é crescente, o que implica que o ganho ( $G = 1 - \frac{T_{prcpr|inf+bid}^2}{T_{prcpr|inf+bid}}$ ) vá diminuindo com

o aumento de  $n$ . Por exemplo, para  $n = 20$ , o valor médio de  $\frac{T_{prcpr|inf+bid}^2}{T_{prcpr|inf+bid}}$  é 0,9195 e o valor médio

de  $G$  é 0,0805; para  $n = 50$ , o valor médio de  $\frac{T_{prcpr|inf+bid}^2}{T_{prcpr|inf+bid}}$  é 0,9426 e o valor médio de  $G$  é 0,0574.

### 6.3 Funcionamento dos Holons

Na secção 5.2 foi apresentada a arquitectura para um sistema de produção de nova geração e na secção 5.3 foram formalmente caracterizados os vários holons da arquitectura do ponto de vista de conhecimento, objectivos e ciclo de vida. Nesta secção, esses holons serão considerados do ponto de vista de procedimentos que a si têm associados. A especificação destes procedimentos terá apenas em conta a função de Escalonamento, escolhida como caso de teste para o protótipo a



desenvolver. Assim sendo, os Holons de Tarefa e de Recurso são descritos com maior detalhe que os restantes.

O arquétipo de holon apresentado na secção 5.2.2 é bastante genérico. Os holons a considerar nas subsecções seguintes denotam especializações desse arquétipo; ou mais concretamente, agentes de *software*, reactivos (orientados por mensagens) com estado interno e com funcionalidades definidas aquando da concepção do sistema.

### 6.3.1 Arquétipo de Holon

Esta secção apresenta funcionalidades comuns a todos os holons, principalmente relacionadas com o processamento de mensagens para gestão das holarquias. No âmbito deste trabalho, apenas são focadas as holarquias pré-definidas na arquitectura.

Além do tratamento de mensagens para gestão de holarquias, todos os holons possuem uma funcionalidade que lhes permite sinalizar ao Holon de Serviços de Directório que estão em execução, enviando, a intervalos regulares, uma mensagem com o seguinte teor:

*bater\_do\_coracao*

A Figura 6.17 corporiza o algoritmo de processamento de mensagens de gestão de holarquias existente em todos os holons.

```

func_msg_hms( $\gamma$ ,  $\mathcal{M}$ )  $\stackrel{\text{def}}{=}$ 
  seja  $\gamma$  a identificação do holon interlocutor
  seja  $\mathcal{M}$  uma mensagem enviada pelo holon  $\gamma$ 
  seja  $Act$  a identificação da operação a executar na lista de membros
  seja  $\eta$  a identificação de um holon
  seja  $H$  a identificação de uma holarquia
  início
    se  $\mathcal{M} = \text{hms\_adicionar}(\gamma)$  então
       $Membros \leftarrow \text{trata\_adicionar}(\gamma)$ 
       $\text{enviar\_msg}(\gamma, \text{r\_hms\_adicionar}(\gamma, Membros))$ 
    senão se  $\mathcal{M} = \text{hms\_remover}(\gamma)$  então
       $\text{trata\_remover}(\gamma)$ 
    senão se  $\mathcal{M} = \text{hms\_act\_membros}(Act, \eta, H)$  então
       $\text{trata\_act\_membros}(H, Act, \eta)$ 
    fim se
  fim

```

Figura 6.17 – Algoritmo de processamento de mensagens de gestão de holarquias

As mensagens referentes à gestão de holarquias são as seguintes:

- *hms\_adicionar*(*Id*) – indica que um holon pretende adicionar-se a esta holarquia;
- *hms\_remover*(*Id*) – indica que um holon pretende retirar-se desta holarquia;

- *hms\_act\_membros(Id-Holarquia, Operação, Id-Holon)* – indica que uma holarquia sofreu uma alteração (adição ou remoção) na sua lista de membros.

O programa em lógica apresentado em seguida, materializa as actividades do algoritmo anterior:

```

trata_adicionar(Holon, Membros) ←
    todo(Holarquia) ∧
    holarquia(Holarquia, Membros) ∧
    actualiza_holarquia(Holarquia, [Holon | Membros]) ∧
    avisa_todos(hms_act_membros(adicionar, Holon, Holarquia), Membros) ∧
    enviar_msg(Holon, r_hms_adicionar(Holon, Membros, ok))

trata_remove(Holon) ←
    todo(Holarquia) ∧
    holarquia(Holarquia, Membros) ∧
    retira_elemento(Holon, Membros, NovosMembros) ∧
    actualiza_holarquia(Holarquia, NovosMembros) ∧
    avisa_todos(hms_act_membros(remove, Holon, Holarquia), Membros)

trata_act_membros(Holarquia, adicionar, Holon) ←
    parte_de(Holarquia) ∧
    holarquia(Holarquia, Membros) ∧
    actualiza_holarquia(Holarquia, [Holon | Membros])
trata_act_membros(Holarquia, remove, Holon) ←
    parte_de(Holarquia) ∧
    holarquia(Holarquia, Membros) ∧
    retira_elemento(Holon, Membros, NovosMembros) ∧
    actualiza_holarquia(Holarquia, NovosMembros)

```

No programa anterior, o predicado *actualiza\_holarquia* modifica a base de conhecimento do holon referente aos membros de uma dada holarquia e o predicado *avisa\_todos* envia uma mensagem de alteração da constituição da holarquia aos seus membros constituintes. Esses holons actualizarão a informação referente à constituição da holarquia ao processarem a mensagem *hms\_act\_membros*.

A operação dos holons para integração nas holarquias é bastante simplificada, sendo ignorados aspectos como a verificação de identidade do holon que executa o pedido de adesão a uma holarquia e o seu “direito” de pertença a essa holarquia. Nesta implementação todos os holons que desejem ser adicionados ou removidos de uma holarquia são aceites ou retirados, partindo-se do pressuposto que todos os holons fornecem informação válida; *i.e.*, todos os holons cumprem o pressuposto de veracidade. Em situações mais próximas da realidade, seria necessário validar o pedido de adesão, verificando se o holon requerente pode ou deve de facto pertencer à holarquia.

### 6.3.2 Serviço de Directório

O *Holon de Serviços de Directório* é um auxiliar, funcionando como uma base de dados centralizada com informação sobre a identificação de holons e anúncio das suas reais potencialidades [Sousa *et al.*, 1999b].

A Figura 6.18 corporiza o algoritmo para o funcionamento do Holon de Serviços de Directório. Nesta implementação são ignorados aspectos de segurança, tais como verificação da identidade de quem faz o registo e principalmente de quem faz a remoção do registo, admitindo-se sempre que a informação é válida e de confiança (pressuposto de veracidade).

```

func_directorio()  $\stackrel{\text{def}}{=}$ 
  seja  $\gamma$  a identificação do holon interlocutor
  seja  $\mathcal{M}$  uma mensagem enviada pelo holon  $\gamma$ 
  seja  $\mathcal{H}$  a lista de habilidades do holon  $\gamma$ 
  seja  $h$  a identificação de uma habilidade
  seja  $\eta$  a identificação de um holon
  seja  $\mathcal{V}$  o valor de verdade da prova de um axioma
  início
    enquanto verdadeiro fazer
      receber_msg( $\gamma, \mathcal{M}$ )
      se  $\mathcal{M} = \text{registra}(\gamma, \mathcal{H})$  então
        regista( $\gamma, \mathcal{H}$ )
      senão se  $\mathcal{M} = \text{remover\_registo}(\gamma)$  então
        apaga_holon( $\gamma$ )
      senão se  $\mathcal{M} = \text{faz}(\eta, h)$  então
        demo fornece( $\eta, h$ )  $\mapsto \mathcal{V}$ 
        enviar_msg( $\gamma, \text{r\_faz}(\eta, h, \mathcal{V})$ )
      senão se  $\mathcal{M} = \text{o\_que\_faz}(\eta)$  então
        o_que_faz( $\eta, LH$ )
        enviar_msg( $\gamma, \text{r\_o\_que\_faz}(\eta, LH)$ )
      senão se  $\mathcal{M} = \text{quem\_faz}(h)$  então
        quem_faz( $h, LH$ )
        enviar_msg( $\gamma, \text{r\_quem\_faz}(h, LH)$ )
      senão se  $\mathcal{M} = \text{bater\_do\_coracao}$  então
        trata_bater_do_coracao( $\gamma$ )
      senão se  $\mathcal{M} = \text{em\_execucao}$  então
        quem_esta_em_execucao( $LH$ )
        enviar_msg( $\gamma, \text{r\_em\_execucao}(LH)$ )
      fim se
    fim ciclo
  fim

```

Figura 6.18 – Algoritmo de funcionamento do holon de serviços de directório

A interacção com este holon é feita através de um pequeno e simples protocolo de pedido-resposta de acordo com as seguintes produções (mensagens):

- *registra*(*Id*, *ListaFuncionalidades*) – anuncia o potencial de um holon existente no sistema;
- *remover\_registo*(*Id*) – remove informação sobre um holon no sistema;
- *faz*(*Id*, *Funcionalidade*) – permite saber se um holon fornece ou não um determinado serviço;
- *o\_que\_faz*(*Id*) – permite conhecer o potencial de outros holons no sistema;
- *quem\_faz*(*Funcionalidade*) – permite obter a lista de holons do sistema capazes de fornecer determinado serviço;
- *bater\_de\_coracao* – anuncia a disponibilidade de um holon, sinalizando que o holon se encontra em execução;
- *em\_execucao* – permite obter a lista de holons registados no sistema e em execução.

Quando o Holon de Serviços de Directório recebe uma mensagem (*i.e.*, conjunção de termos lógicos ou teoremas) vai tentar obter a partir da sua base de conhecimento o valor de verdade correspondente ao pedido e enviar o resultado ao interlocutor. O programa em lógica dado a seguir materializa esta situação.

```
registra(H, LHab) ←
    ¬servicos(H, _) ∧
    adiciona_holon(H, LHab)
registra(H, LHab) ←
    actualiza_holon(H, LHab)

quem_faz(Hab, LH) ←
    fornecedores(Hab, LH)
quem_faz(Hab, [])

o_que_faz(H, LHab) ←
    servicos(H, LHab)
o_que_faz(H, [])

trata_bater_do_coracao(Holon) ←
    agora(Instante),
    adiciona_actualiza_em_exec(Holon, Instante)

quem_esta_em_execucao(LHols) ←
    agora(Instante),
    todas_as_solucões(H, realmente_vivo(Instante, H), LHols)

realmente_vivo(Actual, H) ←
    esta_vivo(H, UltimoBater) ∧
```

$$\text{atribui}(Z, \text{Actual} - \text{UltimoBater}) \wedge \\ \text{realmente\_vivo\_aux}(Z, H)$$

$$\text{realmente\_vivo\_aux}(Z, \_) \leftarrow \\ Z < 300 \\ \text{realmente\_vivo\_aux}(\_, H) \leftarrow \\ \text{remover\_em\_exec}(H)$$

No programa anterior *registra(H, LHab)* acrescenta ou actualiza a base de conhecimento com informação sobre as operações que um holon pode executar (através dos predicados *adiciona\_holon* e *actualiza\_holon*). O predicado *apaga\_holon* remove da base de conhecimento todos os itens de informação referentes a um dado holon. O predicado *quem\_faz(Hab, LH)* tem como função dar a conhecer a existência na base de conhecimento de axiomas do tipo *fornecedores* para a operação em questão. De modo semelhante, *o\_que\_faz(H, LHab)* dá a conhecer a existência ou não na base de conhecimento de axiomas do tipo *serviços* para o holon em questão. O predicado *trata\_bater\_do\_coracao(Holon)* acrescenta ou actualiza a base de conhecimento do Holon de Serviços de Directório com informação sobre o instante em que um dado holon se sinalizou como estando em execução, usando para tal termos do tipo *esta\_vivo(Holon, Instante)*. Finalmente o predicado *quem\_esta\_vivo(Lholons)* tem como função dar a conhecer todos os holons registados no sistema, que se encontram em execução e sinalizarem esse facto há menos de um determinado tempo (e.g., 300 segundos). Caso esse tempo seja ultrapassado, a informação sobre esse holon estar em execução é removida da base de conhecimento.

### 6.3.3 Holon de Produto

Um *Holon de Produto* denota um item do catálogo de produtos fabricados pela empresa, sendo essencialmente responsável por fornecer informação relativa ao processo de fabrico aos outros holons do sistema [Sousa *et al.*, 1999b].

A interacção com este holon é feita através de um simples protocolo do tipo pedido-resposta, de acordo com as produções (mensagens):

- *obter\_arvore* – permite obter a árvore de composição de cada produto a fabricar;
- *obter\_BOM* – permite obter a lista de material para execução de 1 (um) produto de um dado tipo;
- *obter\_planos(Critério)* – permite obter os planos de produção do produto de acordo com critério(s) preestabelecidos (e.g., critérios de qualidade).

A Figura 6.19 denota o algoritmo que corporiza o funcionamento de um Holon de Produto (apenas estão referidas as funcionalidades necessárias para o Planeamento de Produção e Escalonamento).

```

func_produto ()  $\stackrel{\text{def}}{=}$ 
  seja  $\gamma$  a identificação do holon interlocutor
  seja  $\mathcal{M}$  uma mensagem enviada pelo holon  $\gamma$ 
  seja  $Crit$  um critério de optimização de um plano de produção
  início
    enquanto verdadeiro fazer
      receber_msg( $\gamma$ ,  $\mathcal{M}$ )
      se  $\mathcal{M} = \text{obter\_planos}(Crit)$  então
        obter_planos( $Crit$ ,  $LP$ )
        enviar_msg( $\gamma$ , r_obter_planos( $C$ ,  $LP$ ))
      senão se  $\mathcal{M} = \text{obter\_arvore}$  então
        expande_arvore( $Arv$ )
        enviar_msg( $\gamma$ , r_obter_arvore( $Arv$ ))
      senão se  $\mathcal{M} = \text{obter\_BOM}$  então
        expande_BOM( $BOM$ )
        enviar_msg( $\gamma$ , r_obter_BOM( $BOM$ ))
      fim se
    fim ciclo
  fim

```

Figura 6.19 – Algoritmo de funcionamento de um holon de produto

Quando o Holon de Produto recebe uma mensagem vai tentar obter a partir da sua base de conhecimento o valor de verdade correspondente ao pedido e enviar o resultado ao interlocutor. O programa em lógica dado a seguir materializa esta situação.

```

obtem_planos( $C$ ,  $LP$ )  $\leftarrow$ 
  todas_as_soluções(plano( $Id$ ,  $C$ ,  $A$ ,  $LOp$ ), plano( $Id$ ,  $C$ ,  $A$ ,  $LOp$ ),  $LP$ )

expande_arvore( $Arv$ )  $\leftarrow$ 
  todas_as_soluções( $s(S$ ,  $C$ ,  $Q)$ , composicao( $S$ ,  $C$ ,  $Q$ ),  $L$ )  $\wedge$ 
  trata_arv( $L$ ,  $Arv$ )

trata_arv([], [])
trata_arv([ $s(0$ ,  $C$ ,  $Q)$  |  $T$ ], [componente( $C$ ,  $Q)$  |  $T2$ ])  $\leftarrow$ 
  trata_arv( $T$ ,  $T2$ )
trata_arv([ $s(P$ ,  $0$ ,  $Q)$  |  $T$ ], [produto( $P$ ,  $Q$ ,  $Arv$ ) |  $T2$ ])  $\leftarrow$ 
  enviar_msg(id_srv_dir, quem_faz(produto( $P$ )))  $\wedge$ 
  receber_msg(id_srv_dir, r_quem_faz(produto( $P$ ),  $HP$ ))  $\wedge$ 
  enviar_msg( $HP$ , obter_arvore)  $\wedge$ 
  receber_msg( $HP$ , r_obter_arvore( $Arv$ ))  $\wedge$ 
  trata_arv( $T$ ,  $T2$ )

expande_BOM( $BOM$ )  $\leftarrow$ 
  todas_as_soluções( $s(S$ ,  $C$ ,  $Q)$ , composicao( $S$ ,  $C$ ,  $Q$ ),  $L$ )  $\wedge$ 
  trata_BOM( $L$ ,  $X$ )  $\wedge$ 
  agrega_BOM( $X$ , [],  $BOM$ )

```

```

trata_BOM([], [])
trata_BOM([s(0, C, Q) | T], [s(C, Q) | T2]) ←
    trata_BOM(T, T2)
trata_BOM([s(P, 0, Q) | T], T2) ←
    enviar_msg(id_srv_dir, quem_faz(produto(P))) ∧
    receber_msg(id_srv_dir, r_quem_faz(produto(P), HP)) ∧
    enviar_msg(HP, obter_BOM) ∧
    receber_msg(HP, r_obter_BOM(L1)) ∧
    mult_BOM(L1, Q, L2) ∧
    trata_BOM(T, LTemp) ∧
    juntar(Ltemp, L2, T2)

```

No programa anterior, o predicado *obtem\_planos(C, LP)* permite que se construa uma lista com os planos de produção de um certo produto, que por sua vez obedecem a um determinado critério de optimização. O predicado *expande\_arvore(Arv)* corporiza a árvore de composição desse produto, em que cada nó da árvore é um componente ou uma subárvore de um produto. O predicado *expande\_BOM(BOM)* permite que se obtenha a lista de componentes utilizados no produto e subprodutos com as respectivas quantidades. Estes predicados recorrem a alguns predicados auxiliares, nomeadamente, *mult\_BOM(Lista-Original, Quantidade, List-Final)* que devolve a lista de materiais de um subproduto com as quantidades actualizadas do número de componentes necessários e *agrega\_BOM(Lista-Original, Aux, Lista-Final)* que agrega as quantidades dos elementos correspondentes a um mesmo componente.

#### 6.3.4 Holon de Escalonamento

A operação do *Holon de Escalonamento* descrita nesta secção cinge-se ao tratamento de conflitos, nomeadamente, na resolução do Problema de Indecisão.

A Figura 6.20 denota o algoritmo que materializa o funcionamento do Holon de Escalonamento.

```
func_trata_conflitos()  $\stackrel{\text{def}}{=}$   
  seja  $\gamma$  a identificação do holon interlocutor  
  seja  $\mathcal{M}$  uma mensagem enviada pelo holon  $\gamma$   
  início  
    enquanto verdadeiro fazer  
      receber_msg( $\gamma$ ,  $\mathcal{M}$ )  
      se  $\mathcal{M} = \text{lista\_recursos}(LR, TW)$  então  
        se possível( $LR, TW$ ) então  
          enviar_msg( $\gamma$ , luz_verde)  
          act_em_negociação ( $LR, \gamma, TW$ )  
        senão  
          coloca_em_espera( $\gamma, LR, TW$ )  
        fim se  
      senão se  $\mathcal{M} = \text{fim\_de\_negociacao}$  então  
         $LR \leftarrow \text{obter\_estado}(\gamma)$   
        libertar_tarefas( $\gamma, LR$ )  
      senão se  $\mathcal{M} = \text{abandona\_recursos}(LR)$  então  
        libertar_tarefas( $\gamma, LR$ )  
      fim se  
    fim ciclo  
  fim
```

Figura 6.20 – Algoritmo de funcionamento do holon de escalonamento

Quando o Holon de Escalonamento recebe uma mensagem *lista\_de\_recursos* vai verificar se é *possível* iniciar a negociação com esses recursos e, em caso afirmativo, envia ao Holon de Tarefa a mensagem de “Luz Verde”, actualizando o estado dos recursos em negociação. Em caso negativo, essa tarefa é colocada em fila de espera. Este processo é descrito pelo programa em lógica dado a seguir.

```
possível([], _)  
possível([Rec | T], TW)  $\leftarrow$   
  em_negociacao(Rec, LRecTW)  $\wedge$   
  possível_um_recurso(TW, LRecTW)  $\wedge$   
  possível(T, TW)  
possível([Rec | T], TW)  $\leftarrow$   
   $\neg$ em_negociacao(Rec, _)  $\wedge$   
  possível(T, TW)  
  
possível_um_recurso(_, [])  
possível_um_recurso(TW, [s(Tarefa, RecTW) | T])  $\leftarrow$   
   $\neg$ sobrepoe(TW, RecTW)  $\wedge$   
  possível_um_recurso(TW, T)
```

No programa anterior o predicado *em\_negociacao* verifica se um dado recurso se encontra actualmente em negociação e em caso afirmativo devolve a lista de intervalos em negociação e o predicado *sobrepoe* verifica se dois intervalos de tempo são sobrepostos.



Caso receba informação sobre o fim de negociação ou o abandono de recursos, o Holon de Escalonamento vai actualizar o estado de negociação desses recursos e libertar eventuais tarefas em fila de espera (Figura 6.21).

```

func libertar_tarefas( $\gamma$ , LR)  $\stackrel{\text{def}}{=}$ 
  seja  $\gamma$  a identificação do holon interlocutor
  seja LR o conjunto de recursos libertados
  início
    libertar(LR,  $\gamma$ )
     $\langle t, LRe, TWe \rangle \leftarrow \text{pode\_libertar}()$ 
    enquanto  $t \neq \text{nenhuma}$  fazer
      enviar_msg(t, luz_verde)
      act_em_negociação(LRe, TWe)
       $\langle t, LRe, TWe \rangle \leftarrow \text{pode\_libertar}()$ 
    fim ciclo
  fim

```

Figura 6.21 – Algoritmo de funcionamento do holon de escalonamento: libertar tarefas

Com este procedimento é possível começar-se por *libertar* os recursos associados à tarefa em questão e, de seguida tentar libertar cada uma das tarefas em fila de espera, usando para tal o predicado *pode\_libertar* que permite obter informação associada a uma tarefa em fila de espera que possa ser libertada face aos recursos agora livres, e, o predicado *act\_em\_negociação* que actualiza a lista de recursos em negociação.

### 6.3.5 Holon de Tarefa

Os *Holons de Tarefa* denotam as ordens de fabrico enviadas à instalação fabril para a execução de  $n$  itens de um determinado produto, sendo responsáveis por garantir o seu escalonamento [Sousa e Ramos, 1998] [Sousa *et al.*, 1999b].

Como o objectivo deste tipo de holons é o de efectuar o escalonamento da ordem de fabrico e monitorizar a sua execução, após o arranque, o holon passa, essencialmente, por dois estados (Figura 6.22):

- *em negociação*, onde é efectuado o escalonamento da tarefa através de uma negociação operação a operação com os vários recursos disponíveis;
- *em acompanhamento*, onde o holon vai recebendo informação da execução da tarefa dada pelos recursos utilizados.

A Figura 6.22 apresenta o diagrama de estado para os Holons de Tarefa.

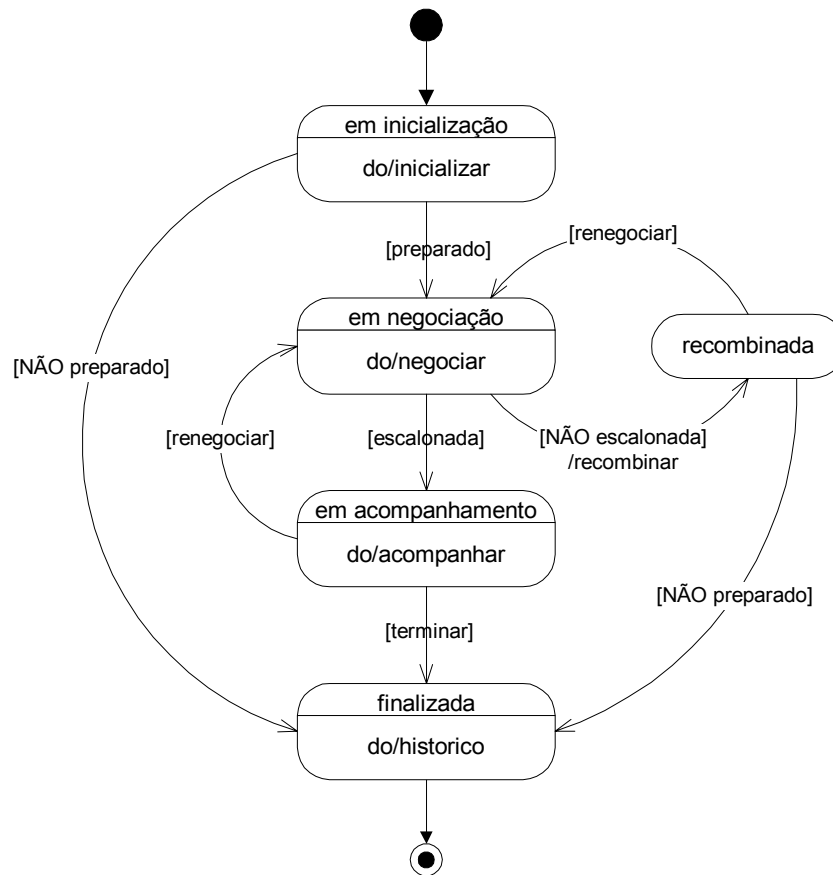


Figura 6.22 – Diagrama de estado para holons de tarefa

O funcionamento geral de um Holon de Tarefa é dado pelo algoritmo da Figura 6.23 que traduz o diagrama de estados apresentado na Figura 6.22.

```

func_tarefa() def
início
  inicializar
  se preparada então
    enquanto ¬terminar fazer
      negociar
      se ¬escalonada então
        recombinar
      senão
        acompanhar
      fim se
    fim ciclo
  fim se
  historico
fim
  
```

Figura 6.23 – Algoritmo de funcionamento de um holon de tarefa

O Holon de Tarefa começa por obter informação sobre o que fazer (algoritmo da Figura 6.24) e em seguida negocia a sua execução com os Holons de Recurso (algoritmo da Figura 6.25), recorrendo para tal a variante do PRCPR apresentada no capítulo anterior. As várias propostas recebidas dos Holons de Recurso são avaliadas e seleccionadas (algoritmo da Figura 6.26) e, caso não seja possível efectuar o escalonamento, o Holon de Tarefa irá recombina os recursos disponíveis para tentar nova negociação (algoritmo da Figura 6.27). Caso seja possível efectuar o escalonamento, o Holon de Tarefa acompanha a sua execução (Figura 6.28) e, no fim, guarda o seu estado na base de dados de histórico.

A Figura 6.24 descreve a fase de inicialização de um Holon de Tarefa.

```

tarefa_inicializar()  $\stackrel{\text{def}}{=}$ 
  seja HE a identificação do holon de escalonamento
  seja P um plano de produção de um produto
  seja R o conjunto de recursos necessários para efectuar o plano P
  início
    registar
    receber_msg(HE, anuncio(TId, NOF, PId, Qt, DataLimite, Attr))
    trata_anuncio(TId, NOF, PId, Qt, DataLimite, Attr)
    aderir_holarquias_pre_definidas
    P ← obter_plano(PId,  $\emptyset$ )
    R ← obter_lista_recursos(P)
    se R ≠  $\emptyset$  então
      inserir preparada
    senão
      inserir  $\neg$ preparada
    fim se
  fim

```

Figura 6.24 – Algoritmo de funcionamento de um holon de tarefa: inicialização

Após a criação da ordem de fabrico e respectivo Holon de Tarefa pelo Holon de Escalonamento, o Holon de Tarefa recebe a mensagem *anuncio(TId, NOF, PId, Qt, DataLimite, Attr)* que especifica o que há a fazer. Após processar essa mensagem o holon regista-se no serviço de directório e adere às holarquias pré-definidas a que pertence; *i.e.*, a holarquia de escalonamento. Em seguida, obtém a informação necessária ao seu funcionamento, ou seja, o plano de produção e os recursos passíveis de serem utilizados em cada operação do plano. As várias funcionalidades presentes no algoritmo são dadas pelo programa em lógica a seguir.

```

trata_anuncio(TId, NOF, PId, Qt, DataLimite, Attr) ←
  inserir(tarefa(TId, NOF, PId, Qt, DataLimite)) ∧
  adiciona_attr(Attr)

adiciona_attr([])
adiciona_attr([a(P, V) | Tj]) ←
  inserir(atributo(P, V) ∧
  adiciona_attr(Tj)

```

```
obter_plano(PId, Attr) ←  
    enviar_msg(id_plan_processos, obter_plano(PId, Attr)) ∧  
    receber_msg(id_plan_processos, r_obter_plano(PId, Attr, Plano)) ∧  
    inserir(Plano)  
  
obter_lista_recursos(Operações-Plano, LRec) ←  
    constroi_LR(Plano, LRec) ∧  
    validar_LR(LRec)  
obter_lista_recursos(_, [])  
  
constroi_LR([], [])  
constroi_LR([nodo(Id, Op, rec(H), _, _, _) | Plano], [(Op, [H]) | LR]) ←  
    constroi_LR(Plano, LR)  
constroi_LR([nodo(Id, Op, oper, _, _, _) | Plano], [(Op, LH) | LR]) ←  
    enviar_msg(id_srv_dir, quem_faz(operacao(Op))) ∧  
    receber_msg(id_srv_dir, r_quem_faz(operacao(Op), LH)) ∧  
    constroi_LR(Plano, LR)
```

No programa anterior *trata\_anuncio* processa a mensagem de *anuncio* com informação de carácter geral sobre a tarefa. A actividade *obter\_plano* contacta o Holon de Planeamento de Processos para a obtenção do plano de produção do produto em questão. Finalmente, a actividade *obter\_lista\_recursos* constrói a lista de recursos a contactar para cada uma das operações do plano.

O algoritmo que corporiza o funcionamento dos Holons de Tarefa para a fase de negociação é apresentado na Figura 6.25.

```

 tarefa_negociar() def
  seja  $\Lambda$  o conjunto de recursos contactados para cada operação do plano  $\mathcal{P}$ 
  seja  $\Theta$  o conjunto de propostas recebidas
  seja  $HE$  a identificação do holon de escalonamento
  seja  $\mathcal{R}$  o conjunto de recursos necessários para efectuar o plano  $\mathcal{P}$ 
  seja  $op$  uma operação do plano  $\mathcal{P}$ 
  seja  $\mathcal{P}$  o plano do produto a executar
  seja  $\mathcal{T}$  a lista de operações contratadas por este holon
  início
     $\Lambda \leftarrow \emptyset$ 
     $\Theta \leftarrow \emptyset$ 
    demo tarefa( $TId$ ,  $NOF$ ,  $PId$ ,  $Qt$ ,  $DataLimite$ )
    enviar_msg( $HE$ , lista_recursos( $\mathcal{R}$ ,  $JT$ ))
    receber_msg( $HE$ , luz_verde)
    para cada  $\langle op, LH \rangle \in \mathcal{R}$  fazer
       $Pred \leftarrow antecessores(op, \mathcal{P}, \mathcal{R})$ 
       $Succ \leftarrow sucessores(op, \mathcal{P}, \mathcal{R})$ 
      para cada  $h \in LH$  fazer
        enviar_msg( $h$ , pedido( $TId$ ,  $DataLimite$ ,  $op$ ,  $Qt$ ,  $JT$ ,  $Pred$ ,  $Succ$ ))
         $\Lambda \leftarrow \Lambda \cup \{ \text{contactado}(h, op) \}$ 
      fim ciclo
    fim ciclo
    para cada  $\text{contactado}(op, LH) \in \Lambda$  fazer
      para cada  $h \in LH$  fazer
        receber_msg( $h$ , proposta( $op$ ,  $Dur$ ,  $LComb$ ))
         $\Theta \leftarrow \Theta \cup \{ \text{prop}(h, op, Dur, LComb) \}$ 
      fim ciclo
    fim ciclo
     $\Phi \leftarrow \text{avaliar_ofertas}(\Theta)$ 
     $\mathcal{T} \leftarrow \text{enviar_respostas}(\Lambda, \Phi)$ 
    enviar_msg( $HE$ , fim_de_negociacao)
  fim

```

Figura 6.25 – Algoritmo de funcionamento de holon de tarefa: negociação

O Holon de Tarefa começa por entrar em contacto com cada um dos recursos capazes de efectuar cada uma das operações do plano, requisitando o serviço pretendido. Em seguida, o holon aguarda pelas propostas de cada Holon de Recurso contactado (uma proposta por cada pedido enviado, já que utiliza a variante do PRCPR). Após a recepção das propostas procede à sua avaliação e selecção, terminando com o envio de contratos para os Holons seleccionados para a execução de cada operação.

O envio das mensagens *pedido* aos Holons de Recurso exige o conhecimento dos holons contactados para as operações antecessoras e sucessoras. Essa informação é conseguida através do seguinte programa em lógica que define as extensões dos predicados *antecessores* e *sucessores* referenciados no algoritmo da Figura 6.25.

```

    antecessores(Op, Plano, LRec, LRecPred) ←
        no_plano(Op, Plano, nodo(I, Op, _, _, P, _)) ∧
        nos_operacoes(Plano, P, LOpPred) ∧
        quais_recursos(LRec, LOpPred, LRecPred)

    sucessores(Op, Plano, LRec, LRecSucc) ←
        no_plano(Op, Plano, nodo(I, Op, _, _, _, S)) ∧
        nos_operacoes(Plano, S, LOpSucc) ∧
        quais_recursos(LRec, LOpSucc, LRecPred)
    
```

Em que o predicado *no\_plano*(*Op*, *Plano*, *Nodo*) implementa a pesquisa de um nó do plano correspondente a uma operação, *nos\_operacoes*(*Plano*, *ListaNos*, *ListaOperações*) devolve a lista de operações correspondentes a uma lista de nós do plano e, *quais\_recursos*(*LRec*, *LOp*, *LRecOp*) devolve a lista de recursos capazes de efectuar cada uma das operações de *LOp*, usando a informação recolhida na inicialização do holon (*LRec*).

Uma das actividades mais importantes dos Holons de Tarefa prende-se com a avaliação das propostas recebidas e escolha dos Holons de Recurso. A Figura 6.26 descreve o algoritmo que corporiza a actividade *avaliar\_ofertas*.

```

    tarefa_avaliar_ofertas( $\Theta$ )  $\mapsto \Phi \stackrel{\text{def}}{=}$ 
        seja  $\Theta$  o conjunto de propostas recebidas
        seja  $\Phi$  o conjunto de propostas avaliadas
        seja  $\kappa$  um critério de avaliação de propostas
        início
             $\Phi \leftarrow \emptyset$ 
             $\Gamma \leftarrow \text{preprocessar}(\Theta)$ 
            enquanto  $\Phi = \emptyset \wedge (\kappa \leftarrow \text{critério\_de\_selecção}()) \neq \text{cancelar}$  fazer
                 $\Phi \leftarrow \text{escolhe-}\kappa(\Gamma)$ 
            fim ciclo
            se  $\kappa = \text{cancelar}$  então
                 $\Phi \leftarrow \emptyset$ 
            senão
                inserir escalonada
            fim se
        fim
    
```

Figura 6.26 – Algoritmo de funcionamento de holon de tarefa: avaliação de propostas

Esta função implementa uma forma de meta-selecção, permitindo que se efectuem escolhas das propostas com base em múltiplos critérios fornecidos por *critério\_de\_selecção*. Caso não seja possível encontrar uma solução segundo um determinado critério é tentado outro até estes se esgotarem.

Antes de efectuar a escolha é feito um pré-processamento ao conjunto de mensagens de proposta recebidas dos Holons de Recurso. Assim, a actividade *preprocessar* constrói uma estrutura de dados mais apropriada para a avaliação das propostas, que consiste numa lista de

produções do tipo  $alt(Caminho, CustoTotal, LProp)$ , onde  $LProp$  denota um elemento na forma  $prop(Rec, Op, Dur, Lista-Intervalos, Custo)$ . Desta lista são também eliminadas as soluções impossíveis, ou seja, aquelas para as quais pelo menos um dos recursos não tem intervalos de tempo livre.

De entre os critérios de selecção passíveis de serem utilizados na escolha das propostas, três são materializados a partir do programa em lógica dado a seguir: (i) primeira solução válida; (ii) solução com menor custo; e (iii) solução com maior folga temporal até à data limite de conclusão da tarefa.

```

escolhe-primeiro([alt(Path, C, LProp) | _], ListaSel) ←
    atribui(LProp, ListaSel)

escolhe-menor_custo(ListaPreProc, ListaSel) ←
    menor_custo(ListaPreProc, alt(Path, CT, LProp)) ∧
    atribui(LProp, ListaSel)

escolhe-maior_folga(ListaPreProc, ListaSel) ←
    limite_superior(TW, F) ∧
    calc_folga(ListaPreProc, ListaPreProcFolga) ∧
    maior_folga(ListaPreProcFolga, alt2(Path, Custo, Folga, LProp)) ∧
    atribui(LProp, ListaSel)

atribui([], [])
atribui([prop(Rec, Op, D, [H | _], C) | T], [sel(Rec, Op, D, H, C) | LT]) ←
    atribui(T, LT)

```

Neste programa recorre-se a outros predicados, tais como o predicado *menor\_custo*, que devolve a proposta com menor custo; *limite\_superior* que devolve o limite superior do intervalo no que respeita às temporais para a tarefa em causa; *calc\_folga* que constrói uma estrutura de dados paralela à lista pré-processada, incluindo o valor da folga de cada proposta e *maior\_folga* que à semelhança de *maior\_custo* devolve a proposta com maior folga.

Através do predicado *atribui* escolhe-se o primeiro intervalo livre em cada proposta, utilizando-se assim uma filosofia de escalonamento “o mais cedo possível”. No entanto, seria possível fazer a escolha dos intervalos utilizando uma via alternativa, a de “o mais tarde possível”, que assim nos aproximaria do conceito JIT (*Just In Time*).

Após a selecção de propostas o Holon de Tarefa envia o resultado da avaliação que se fez das propostas aos diversos Holons de Recurso já contactados, enviando uma mensagem de contrato para todos os Holons de Recurso seleccionados e, uma mensagem de cancelamento para os restantes.

```

enviar_respostas([], _)
enviar_respostas([contactado(RecId, Op) | T], LSel) ←
    membro(sel(RecId, Op, Dur, JT, Custo), LSel) ∧

```

```
antecessores_sel(LSel, Op, PredSel) ∧  
sucessores_sel(LSel, Op, SuccSel) ∧  
enviar_msg(RecId, contrata(Op, JT, PredSel, SuccSel)) ∧  
enviar_respostas(T, LSel)  
enviar_respostas([contactado(RecId, Op) | T], LSel) ←  
enviar_msg(RecId, cancelar(Op)) ∧  
enviar_respostas(T, LSel)
```

Caso o escalonamento da tarefa não seja possível, o Holon de Tarefa tenta uma nova negociação utilizando um plano alternativo (Figura 6.27). Para isso é enviada informação extra ao Holon de Planeamento de Processos sobre os planos já utilizados.

```
tarefa_recombinar()  $\stackrel{\text{def}}{=}$   
seja  $\Pi$  o conjunto de planos já utilizados  
seja  $\mathcal{P}$  um plano de produção de um produto  
seja  $\mathcal{R}$  o conjunto de recursos necessários para efectuar o plano  $\mathcal{P}$   
início  
   $\Pi \leftarrow \Pi \cup \mathcal{P}$   
   $\mathcal{P} \leftarrow \text{obter\_plano}(Pid, \Pi)$   
   $\mathcal{R} \leftarrow \text{obter\_lista\_recursos}(\mathcal{P})$   
  se  $\mathcal{R} \neq \emptyset$  então  
    inserir renegociar  
  senão  
    inserir  $\neg$ preparada  
  fim se  
fim
```

Figura 6.27 – Algoritmo de funcionamento de um holon de tarefa: recombinar

Na Figura 6.28 é apresentado o diagrama de estado do holon para a operação *acompanhar*. Esta actividade prende-se com o acompanhamento da operação de fabrico dos produtos constantes na ordem de fabrico. A ligação com o equipamento produtivo sai fora do âmbito deste trabalho, pelo que esta actividade é especificada embora não seja implementada.



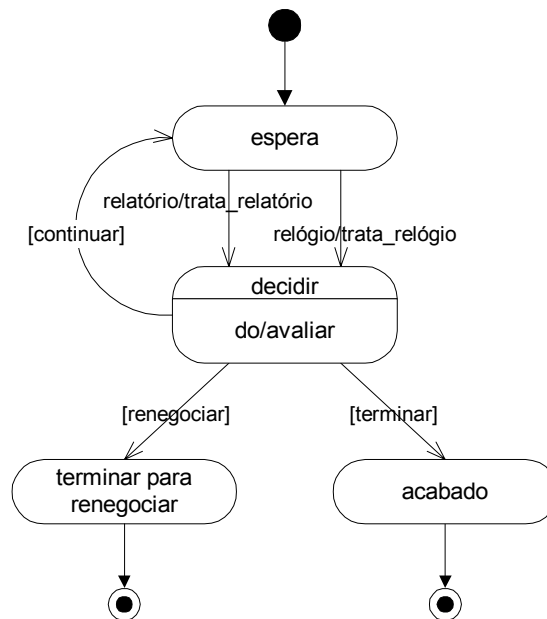


Figura 6.28 – Diagrama de estado para holons de tarefa: acompanhamento

O evento *relatório* está ligado à recepção de uma mensagem de um Holon de Recurso com informação sobre a execução de uma dada tarefa. O evento *relógio* denota um certo intervalo de tempo durante o qual não foi recebida nenhuma informação sobre o estado da execução da tarefa. A actividade *trata\_relógio* envia a todos os Holons de Recurso que nada comunicaram sobre o estado em que se encontra a execução da tarefa, uma mensagem pedindo essa informação. A actividade *trata\_relatório* actualiza a informação sobre a execução da tarefa, com base na informação recebida dos Holons de Recurso. A actividade *avaliar* verifica o estado de execução da tarefa comparando o estado actual com o que foi planeado, e age em conformidade: termina quando a tarefa for executada ou cancelada, ou renegoceia caso a execução da tarefa não esteja a ser cumprida de acordo com o contrato efectuado com os Holons de Recurso.

### 6.3.6 Holon de Recurso

Um *Holon de Recurso* denota o estado actual de um recurso (físico) da instalação fabril (*e.g.*, estado de funcionamento, actividades a efectuar) e é responsável pelo escalonamento de operações requisitadas pelos Holons de Tarefa [Sousa e Ramos, 1998] [Sousa *et al.*, 1999b].

As actividades contratadas por um Holon de Recurso estão representadas na sua agenda. Essa agenda pode ser visualizada como um gráfico de Gantt onde os rectângulos representam as operações contratadas a esse recurso. Para efectuar o escalonamento, o holon necessita de manipular a agenda, podendo efectuar sobre ela cinco operações: (i) *cálculo dos intervalos livres*;

(ii) *limite inferior*; (iii) *limite superior*; (iv) *influência directa*; (v) *influência inversa*. Estas operações são descritas em seguida.

O predicado *calc\_intervalos\_livres*(*AgOcupada*, *Inicio*, *Fim*, *Duração*, *AgLivre*) permite-nos obter a lista de intervalos livres do Holon de Recurso com uma determinada *Duração* mínima e atendendo à janela temporal limitada por *Inicio* e *Fim*. Esta lista pode também ser visualizada como um gráfico de Gantt, em que os rectângulos denotam tempos livres.

A Figura 6.29 dá uma caracterização pictórica desta operação para uma determinada agenda de actividades, considerando que a tarefa a executar ocupa *duas* (2) unidades de tempo, e ocorre entre os limites *a* e *b*.

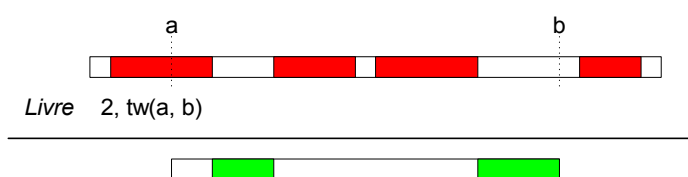


Figura 6.29 – Agenda de um recurso e agenda livre

Na Figura 6.29 a primeira barra refere-se à agenda de actividades contratadas pelo holon (rectângulos escuros), ou seja, a sua agenda de intervalos ocupados. A agenda resultante tem os intervalos livres (rectângulos claros) entre os limites *a* e *b* com uma duração superior ao número de unidades de tempo necessárias (no exemplo em causa têm-se *duas*).

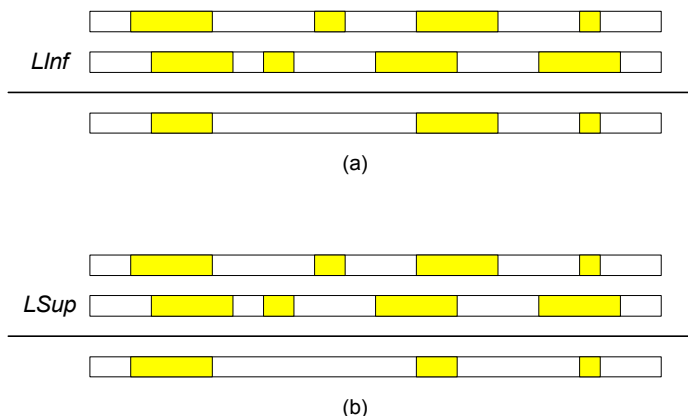


Figura 6.30 – Limitação de agendas: (a) inferior; (b) superior

A limitação de duas agendas permite criar uma nova agenda, cujos intervalos correspondem às zonas ocupadas em ambas as agendas (Figura 6.30), mas cujo limite inferior ou superior obedeça às restrições da segunda agenda. Esta operação é efectuada a partir das extensões dos predicados *limitacao\_inferior*(*ListaA*, *ListaB*, *ListaResultado*) e *limitacao\_superior*(*ListaA*, *ListaB*, *ListaResultado*).

As duas últimas operações sobre as agendas são a influência directa (Figura 6.31a) e a influência inversa (Figura 6.31b), que passam, respectivamente, por uma deslocação à direita ou à esquerda dos limites inferiores e superiores de cada intervalo da agenda.

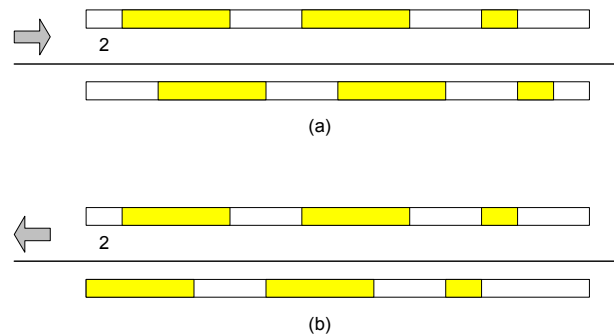


Figura 6.31 – Influência de agendas: (a) directa; (b) inversa

Estas operações são realizadas com recurso às extensões dos predicados *influencia\_directa(ListaA, Duração, ListaResultado)* e *influencia\_inversa(ListaA, Duração, ListaResultado)*.

Conforme o que foi referido na secção 5.3.4, os Holons de Recurso estão divididos a dois níveis: *software* e *hardware*. A Figura 6.32 apresenta o diagrama de estados para a componente de *software* de um Holon de Recurso.

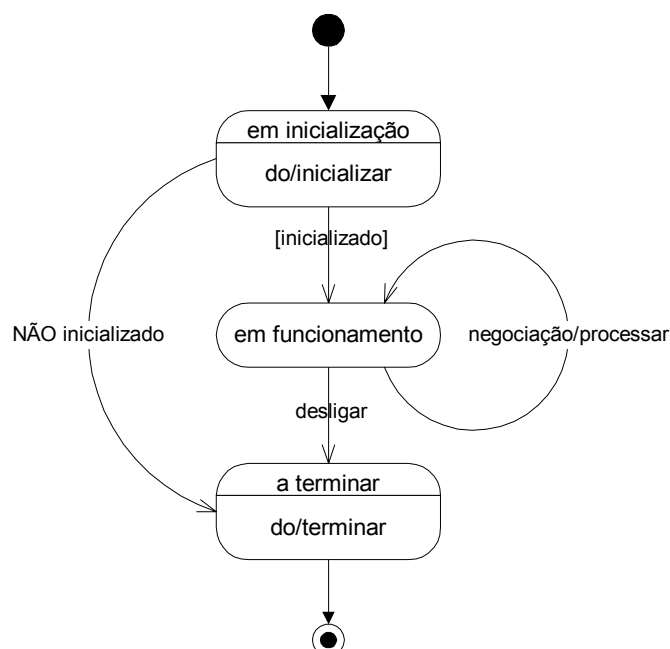


Figura 6.32 – Diagrama de estado para holons de recurso: planeamento

No momento da *inicialização*, o Holon de Recurso constrói a sua lista inicial de actividades, regista-se no serviço de directório e adere às holarquias pré-definidas a que pertence (*i.e.*, as holarquias de planeamento de processos, planeamento da produção e escalonamento). Em seguida, o holon *processa* os pedidos de *negociação* que lhe chegam dos Holons de Tarefa e dos Holons de Recurso, por forma a efectuar o escalonamento das operações inerentes à execução de uma dada tarefa (de acordo com o algoritmo apresentado na Figura 6.33).

```

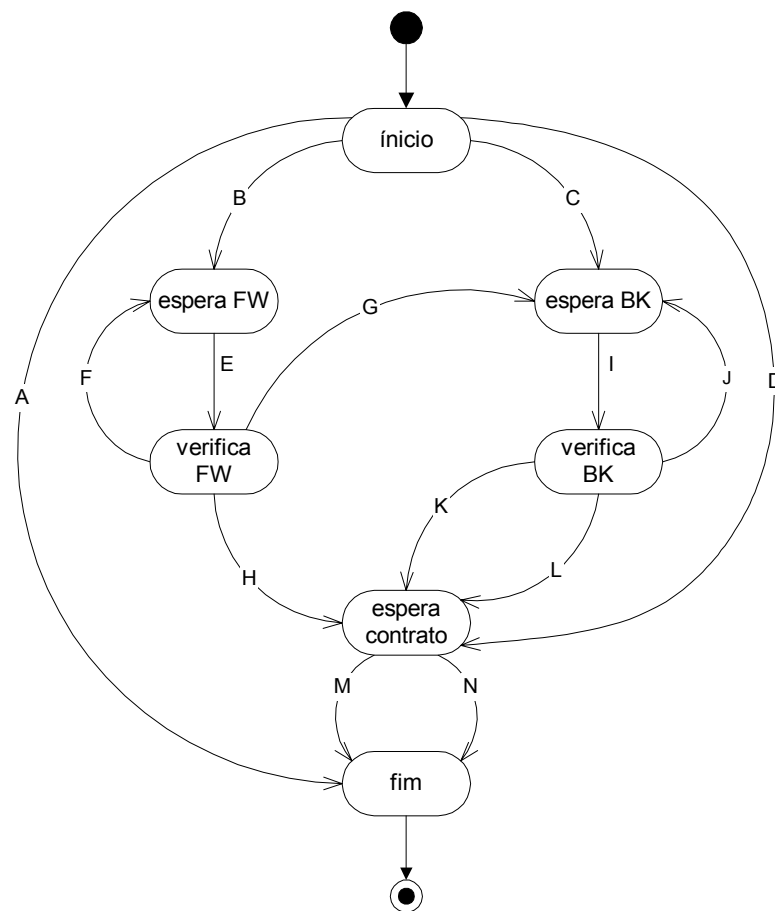
func_recurso_processar() def
  seja  $\gamma$  a identificação de um holon consumidor
  seja  $\mathcal{M}$  uma mensagem recebida
  seja  $op$  uma operação a executar
  seja  $\mathcal{H}$  a lista de funcionalidades do holon
  seja  $\mathcal{A}$  o conjunto de tarefas contratadas pelo holon
  início
    enquanto  $\neg$ desligar fazer
      receber_msg( $\gamma$ ,  $\mathcal{M}$ )
      se  $\mathcal{M} = \text{pedido}(TId, DtLimite, op, Qt, tw(I, F), Pred, Succ)$  então
        se  $op \in \mathcal{H}$  então
          duracao_da_operacao( $op, Qt, DurI, DurQt$ )
           $Ag \leftarrow \text{calc\_intervalos\_livres}(\mathcal{A}, I, F, DurQt)$ 
          cria_estado( $TId, DtLimite, op, Qt, DurI, DurQt, tw(I, F),$ 
             $Pred, Succ$ )
          trata_pedido( $op, Qt, DurI, DurQt, Pred, Succ, Ag$ )
        senão
          enviar_msg( $\gamma$ , proposta( $op, 0, \emptyset$ ))
        fim se
      senão se  $\mathcal{M} = \text{influencia\_directa}(TId, op, DurI, LComb)$  então
        obter_estado( $TId, op, espera\_fw(Ag, CombAg, Pred)$ )
         $op\_pred\_maior(DurI, TId, op, LComb, LComb2),$ 
         $combinacoes(Ag, \{fw(\gamma, Op, Lcomb2)\}, LoI)$ 
        retira_elemento( $\gamma, Pred, Pred\_Falta$ )
        trata_inf_directa( $TId, op, Ag, \{LoI\} \cup CombAg, Pred\_Falta$ )
      senão se  $\mathcal{M} = \text{influencia\_inversa}(TId, op, LComb)$  então
        obter_estado( $TId, op, espera\_bk(CombAg, Succ)$ )
         $combinacoes\_bk(CombAg, LComb, NovaCombAg)$ 
        retira_elemento( $\gamma, Succ, Succ\_Falta$ )
        trata_inf_inversa( $TId, op, NovaCombAg, Succ\_Falta$ )
      senão se  $\mathcal{M} = \text{contrata}(op, IntervaloSel, SelPred, SelSucc)$  então
        obter_estado( $\gamma, op, Qt, \_, DurQt, DtLimite, \_, \_, \_$ ),
         $\langle i, f \rangle \leftarrow \text{escalonar}(IntervaloSel, op, DurQt)$ 
         $\mathcal{A} \leftarrow \mathcal{A} \cup \{ \text{actividade}(\gamma, op, Qt, i, f, DataLimite, por\_fazer,$ 
           $SelPred, SelSucc) \}$ 
        destruir_estado( $\gamma, op$ )
      senão se  $\mathcal{M} = \text{cancelar}(op)$  então
        destruir_estado( $\gamma, op$ )
      fim se
    fim ciclo
  fim

```

Figura 6.33 – Algoritmo de planeamento de holons de recurso

Após a recepção de um pedido, o Holon de Recurso actualiza a sua agenda de intervalos livres, e vê onde poderá escalonar a nova operação. De seguida, corre uma máquina de estados para a execução do PRCPR (dada pela Figura 6.34). Todo o funcionamento do Holon de Recurso rege-se pela execução desta máquina de estado para cada processo de negociação em que participa. Quando, finalmente, obtiver a lista final de intervalos livres para um pedido, o Holon de Recurso envia essa proposta ao Holon de Tarefa, esperando pela aceitação ou recusa da proposta.

A Figura 6.34 denota o diagrama de estados do Protocolo de Rede de Contrato com Propagação de Restrições (PRCPR) apresentado na secção 6.2 e executado pelos Holons de Recurso.



*Figura 6.34 – Protocolo RCPR nos holons de recurso*

Na Tabela 6.3 são apresentadas as transições desta máquina de estados, indicando-se as condições ou eventos que as viabilizam, bem como as acções a executar pelo Holon de Recurso em cada transição de estado.

Tabela 6.3 – Transições na máquina de estados do PRCPR nos holons de recurso

Transição	[Condição] Evento	Acção
<b>A</b>	[ $Ag = \emptyset$ ]	Fazer proposta vazia
<b>B</b>	[ $Ag \neq \emptyset \wedge Pred \neq \emptyset$ ]	–
<b>C</b>	[ $Ag \neq \emptyset \wedge Pred = \emptyset \wedge Succ \neq \emptyset$ ]	Enviar <i>influencia_directa</i> aos sucessores
<b>D</b>	[ $Ag \neq \emptyset \wedge Pred = \emptyset \wedge Succ = \emptyset$ ]	Fazer proposta
<b>E</b>	<i>influencia_directa</i>	Limitar a agenda actual com a lista influenciada de intervalos recebida dos antecessores, gerando uma combinação para cada possibilidade de combinação de recursos
<b>F</b>	[ $falta(Pred) \neq \emptyset$ ]	–
<b>G</b>	[ $falta(Pred) = \emptyset \wedge Succ \neq \emptyset$ ]	Enviar <i>influencia_directa</i> aos sucessores
<b>H</b>	[ $falta(Pred) = \emptyset \wedge Succ = \emptyset$ ]	Enviar <i>influencia_inversa</i> aos antecessores e fazer proposta
<b>I</b>	<i>influencia_inversa</i>	Limitar cada combinação actual de recursos com a lista influenciada de intervalos recebida dos sucessores para a mesma combinação de recursos
<b>J</b>	[ $falta(Succ) \neq \emptyset$ ]	–
<b>K</b>	[ $falta(Succ) = \emptyset \wedge Pred = \emptyset$ ]	Fazer proposta
<b>L</b>	[ $falta(Succ) = \emptyset \wedge Pred \neq \emptyset$ ]	Enviar <i>influencia_inversa</i> aos antecessores e fazer proposta
<b>M</b>	<i>contrata</i>	Escalonar na agenda de actividades
<b>N</b>	<i>Cancela</i>	–

O funcionamento da máquina de estados é concretizado através das extensões dos predicados *cria\_estado*, *obter\_estado*, *act\_estado* e *destruir\_estado*. Em que, *cria\_estado* inicializa a máquina de estados e guarda alguma informação respeitante à negociação em questão; *obter\_estado/3* permite conhecer qual o estado actual e *obter\_estado/9* permite obter a informação

para uma dada negociação; *act\_estado* transfere a máquina de estados para um novo estado e *destruir\_estado* destrói toda a informação referente a uma negociação.

No algoritmo da Figura 6.33 a partir da invocação do predicado *combinacoes* geram-se as combinações de intervalos passíveis de utilização na execução da tarefa, de acordo com os recursos disponíveis. O predicado *combinacoes\_bk* realiza a operação de limitação das listas influenciadas de intervalos livres entretanto recebidas com cada uma das combinações possíveis. A partir da extensão do predicado *op\_pred\_maior* trata-se o caso em que a operação antecessora tem uma duração maior que a operação requisitada ao recurso, pelo que é necessário obter uma nova lista influenciada de intervalos. Finalmente, o predicado *escalonar* determina o limite superior do intervalo em que a operação é escalonada tendo em conta a sua duração.

As acções a executar pelo Holon de Recurso para cada uma das mensagens recebidas são agora definidas através do programa em lógica apresentado em seguida.

```

trata_pedido(Tarefa, Op, _, _, _, []) ←
    enviar_msg(Tarefa, proposta(Op, 0, []))

trata_pedido(Tarefa, Op, _, DurQt, [], [], Ag_Livre) ←
    faz_oferta(Tarefa, Op, DurQt, Ag_Livre)

trata_pedido(Tarefa, Op, Dur1, _, [], Succ, Ag_Livre) ←
    influencia_directa(Ag_Livre, Dur1, Ag_Linha) ∧
    nome(EsteHolon) ∧
    enviar_fw_todos(Tarefa, [inf(Ag_Linha, [EsteHolon])], Succ) ∧
    act_estado(Tarefa, Op, espera_bk(Ag_Livre, Succ))

trata_pedido(Tarefa, Op, _, _, Pred, Succ, Ag_Livre) ←
    act_estado(Tarefa, Op, espera_fw(Ag_Livre, [], Pred))

faz_oferta(Tarefa, Op, D, Ag) ←
    enviar_msg(Tarefa, proposta(Op, D, Ag)) ∧
    act_estado(Tarefa, Op, espera_contrato(Ag))

```

O recurso à extensão do predicado *trata\_pedido* ocorre quando o Holon de Recursos recebe uma mensagem *pedido*. A primeira instância do predicado materializa a situação em que o holon não possui intervalos livres para efectuar a operação. A segunda cláusula denota uma situação em que a operação requisitada é a única operação constante do plano (não tem antecessores nem sucessores). A terceira cláusula corporiza a situação em que o holon não tem antecessores, dando assim início à fase de influência directa. A quarta e última instância materializa a situação em que o holon tem antecessores, tendo por isso que esperar pelas mensagens de influência directa.

Através da invocação do predicado *faz\_oferta* uma mensagem de proposta é enviada ao Holon de Tarefa e, por seu lado, através da invocação de *enviar\_fw\_todos* é enviada uma mensagem de influência directa a todos os recursos sucessores. Por recurso a *trata\_inf\_dir*

processam-se as mensagens de influência directa recebidas pelo holon. Estes acontecimentos são descritos pelo programa em lógica dado a seguir.

```

trata_inf_directa(Tarefa, Op, Ag, LComb, []) ←
    nivelar(LComb, LCombAg) ∧
    obter_estado(Tarefa, Op, _, Dur1, DurQt, _, _, Pred, Succ) ∧
    trata_inf_dir_aux(Tarefa, Op, LCombAg, Dur1, DurQt, Pred, Succ)

trata_inf_directa(Tarefa, Op, Ag, LComb, Pred_Falta) ←
    act_estado(Tarefa, Op, espera_fw(Ag, LComb, Pred_Falta))

trata_inf_dir_aux(Tarefa, Op, LCombFinalAg, Dur1, DurQt, Pred, []) ←
    enviar_bk_filtro(Tarefa, Op, Dur1, LCombFinalAg, Pred) ∧
    faz_oferta(Tarefa, Op, DurQt, LCombFinalAg)

trata_inf_dir_aux(Tarefa, Op, LCombAg, Dur1, _, Pred, Succ) ←
    influencia_directa(LCombAg, Dur1, LoI) ∧
    enviar_fw_todos(Tarefa, LoI, Succ) ∧
    act_estado(Tarefa, Op, espera_bk(LCombAg, Succ))
    
```

A primeira instância do predicado *trata\_inf\_directa* representa a situação em que todos os recursos antecessores já enviaram as mensagens de influência directa, tendo por isso o holon que iniciar a fase de influência inversa caso não tenha sucessores (primeira instância de *trata\_inf\_dir\_aux*); ou então continuar a fase de influência directa caso possua sucessores (segunda instância de *trata\_inf\_dir\_aux*). A segunda instância do predicado *trata\_inf\_dir* mantém o recurso à espera das restantes mensagens de influência directa dos antecessores em falta.

Por via da invocação do predicado *enviar\_bk\_filtro* envia-se uma mensagem de influência inversa a todos os recursos antecessores, tendo em conta o caminho que gerou cada combinação de intervalos livres. Internamente *enviar\_bk\_filtro* vai gerar uma agenda influenciada de acordo com a duração da operação deste recurso e a operação requisitada a cada recurso antecessor. A extensão do predicado *enviar\_bk\_filtro* é dada pelas produções (ou axiomas):

```

enviar_bk_filtro(_, _, _, _, [])
enviar_bk_filtro(Tarefa, MinhaOp, Dur1, LCombFinalAg, [Rec:Op | T]) ←
    dur_op_pred(Tarefa, MinhaOp, Rec, Op, DurOpPred) ∧
    max(Dur1, DurOpPred, Dur) ∧
    influencia_inversa(LCombFinalAg, Dur, LoI) ∧
    filtrar_bk(LoI, R, LBK) ∧
    enviar_msg(R, influencia_inversa(Task, Op, LBK)) ∧
    enviar_bk_filtro(Tarefa, MinhaOp, Dur1, LCombFinalAg, T)
    
```

a extensão do predicado *trata\_inf\_inv* é utilizada para o processamento das mensagens de influência inversa recebidas pelo holon, sendo dada pelas produções:

```

trata_inf_inversa(Tarefa, Op, LCombFinalAg, []) ←
    obter_estado(Tarefa, Op, _, Dur1, DurQt, _, _, Pred, _) ∧
    
```



---


$$trata\_inf\_inv\_aux(Tarefa, Op, LCombFinalAg, Pred, DurI, DurQt)$$

$$trata\_inf\_inversa(Tarefa, Op, LCombAg, Succ\_Falta) \leftarrow \\ act\_estado(Tarefa, Op, espera\_bk(LCombAg, Succ\_Falta))$$

$$trata\_inf\_inv\_aux(Tarefa, Op, LCombFinalAg, [], \_, DurQt) \leftarrow \\ faz\_oferta(Tarefa, Op, DurQt, LCombFinalAg)$$

$$trata\_inf\_inv\_aux(Tarefa, Op, LCombFinalAg, Pred, DurI, DurQt) \leftarrow \\ enviar\_bk\_filtro(Tarefa, Op, DurI, LCombFinalAg, Pred) \wedge \\ faz\_oferta(Tarefa, Op, DurQt, LCombFinalAg)$$

A primeira instância do predicado *trata\_inf\_inv* corporiza a situação em que todos os recursos sucessores já enviaram as mensagens de influência inversa, tendo por isso o holon que terminar esta fase caso não tenha antecessores (primeira instância de *trata\_inf\_inv\_aux*); ou então continuar a fase de influência inversa caso possua antecessores (segunda instância de *trata\_inf\_inv\_aux*). A invocação da segunda instância do predicado *trata\_inf\_inv* mantém o recurso à espera das restantes mensagens de influência inversa dos sucessores em falta.

Conforme o que foi referido anteriormente (secção 5.3.4) os Holons de Recurso estão divididos em dois níveis: *software* e *hardware*. A componente de *software* acabou de ser apresentada, quanto à componente de *hardware*, embora a ligação com o equipamento fabril saia fora do âmbito deste trabalho, esta pode ser traduzida pelo diagrama de estados da Figura 6.35.

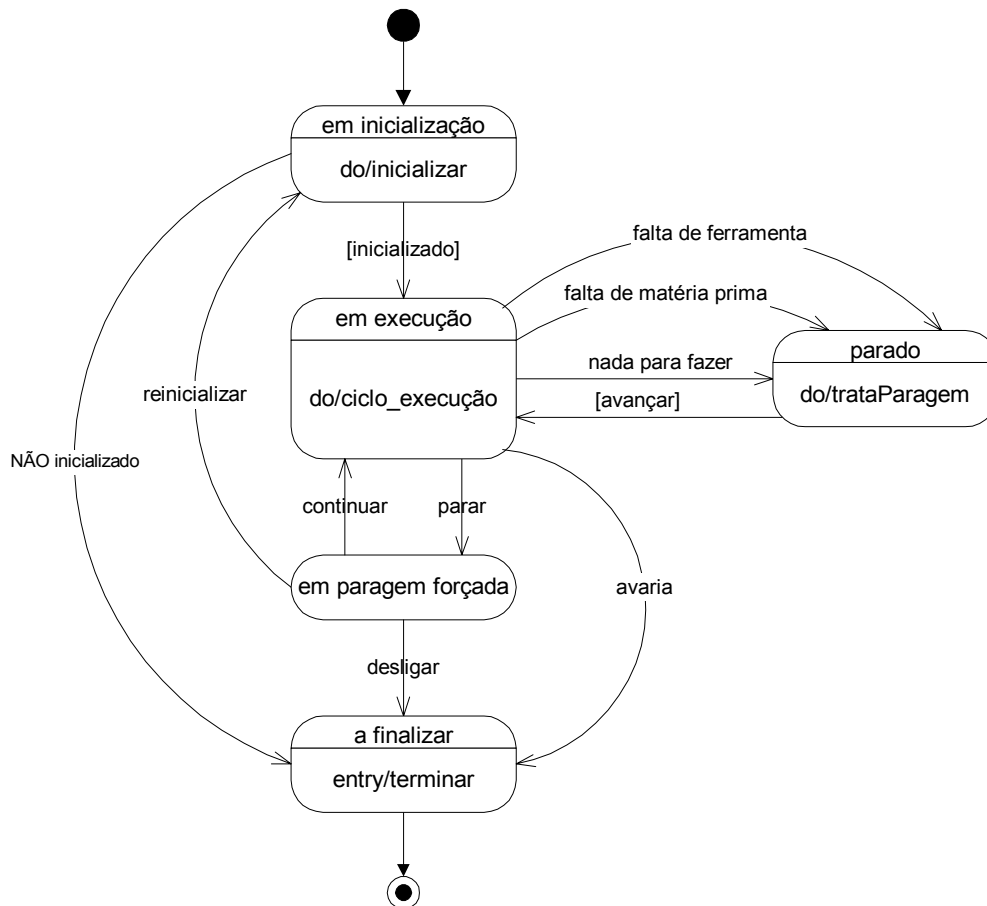


Figura 6.35 – Diagrama de estado para holons de recurso: execução

O evento *reinicializar* corporiza a operação de accionamento do botão de reinicialização do equipamento. O evento *parar*, por sua vez, denota o accionamento do botão de paragem de emergência, enquanto que o evento *continuar* indicia a anulação da operação de paragem de emergência (normalmente pressionando novamente o botão de paragem ou rodando-o). Os eventos *nada para fazer*, *falta de matéria prima* e *falta de ferramenta* denotam, respectivamente, as situações onde não há actividades a executar pelo recurso, não há matéria prima ou componentes necessários à continuação da operação em curso, e não há uma ferramenta indispensável à execução da operação em curso. O evento *desligar* refere-se à pressão do botão de desligar o equipamento. A actividade principal do recurso decorre no estado *em execução*, onde o recurso aguarda comandos enviados pelo dispositivo controlador do equipamento (e.g., carregar programa NC), executando-os de seguida.

### 6.3.7 Relação com o Método Original de Escalonamento

Conforme foi referido anteriormente, o caso de teste considerado para validar este trabalho foi a actividade de escalonamento tendo sido apresentado um protocolo de negociação e os algoritmos de funcionamento dos holons, tendo em vista o escalonamento de tarefas industriais.

O procedimento de escalonamento utilizado neste trabalho, resulta de uma abordagem *sui generis*, porventura distribuída, a um método original apresentado em [Almeida, 1995] e [Ramos *et al.*, 1995] para escalonamento dinâmico de tarefas industriais sujeitas a prazos de entrega.

O método original pode ser caracterizado pela aceitação da seguinte axiomática:

- utilização de uma *abordagem centralizada* com um único processo computacional, conhecendo todos os dados e efectuando todos os cálculos;
- a existência de *um único recurso capaz de efectuar cada operação*, não existindo por isso alternativas para a execução de cada operação;
- a *utilização de comportamentos* [Almeida, 1995] que definem um esquema fixo de escalonamento para as diversas operações do plano de produção, ou seja, o intervalo de tempo entre duas operações sucessivas do plano é fixo;
- não considera tempos de *setup* das máquinas, nem de transporte entre estações de trabalho;
- utilização de um *operador de intersecção* de agendas de intervalos para as fases de influência directa e inversa;
- *sugestão de intervalos de escalonamento* para um operador humano ou sistema de apoio à decisão.

A Figura 6.36a apresenta um comportamento de escalonamento para o fabrico de um produto podendo ver-se na Figura 6.36b a utilização desse comportamento para o escalonamento de uma tarefa com quantidade 3. Na Figura 6.36c pode ver-se um outro comportamento para a execução de três produtos do mesmo tipo considerando a existência de *buffers*.

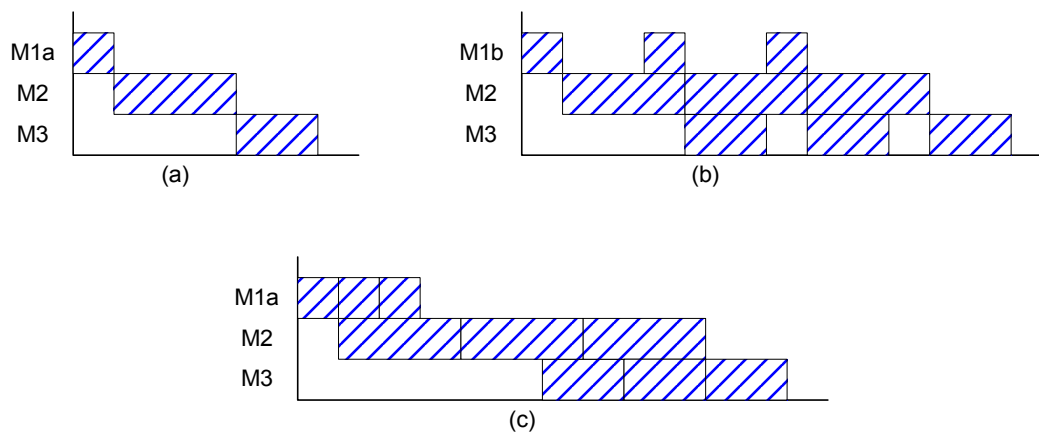


Figura 6.36 – Comportamentos de escalonamento

Conforme se pode ver através do que se expõe na Figura 6.36, o método original dependia da definição *a priori* dos comportamentos para o número exacto de produtos a fabricar, considerando o *buffer* com o comprimento necessário para evitar buracos no escalonamento.

O método de escalonamento utilizado neste trabalho pode ser dado pela axiomática:

- utilização de uma *abordagem distribuída* com vários processos computacionais (*i.e.*, os holons) efectuando parcelas do cálculo e conhecendo apenas parte dos dados do problema;
- utilização de *custos por operação por recurso* que contribui com mais um parâmetro para obtenção de soluções diferentes;
- possibilidade de existência de *múltiplos recursos para a execução de cada operação*, criando por isso várias alternativas para o escalonamento de uma tarefa tendo em conta as agendas de actividades de cada recurso e o tempo e o custo de execução de cada operação nos vários recursos alternativos;
- possibilidade de utilização de *um mesmo recurso em mais que uma operação* do plano;
- *não utilização de comportamentos*, o que permite maior flexibilidade no tempo entre operações, mas implica a existência de *buffers* (assumidos de tamanho infinito);
- não consideração de tempos de *setup* das máquinas nem de transporte entre estações de trabalho;
- utilização de um *operador para o cálculo dos limites inferior e superior* de agendas de intervalos para as fases de influência directa e inversa;
- *escolha de intervalos de escalonamento* com base em algum critério de optimização previamente definido.

Em relação ao método original podem considerar-se duas grandes melhorias, a existência de recursos alternativos para cada operação e a não utilização de comportamentos de escalonamento. A primeira, embora aumente a complexidade do problema, gerando nalguns casos uma explosão combinatória do espaço de soluções, aproxima-se mais da realidade industrial, pois cada vez mais as máquinas poderão executar um maior número de operações e cada vez mais será maior a sobreposição de funcionalidades (*i.e.*, as operações possíveis de efectuar) entre recursos. A segunda melhoria permite evitar o passo prévio de geração de comportamentos evitando o aparecimento de buracos no escalonamento em situações semelhantes à da Figura 6.36b. Adicionalmente, o intervalo de tempo entre operações sucessivas não é fixo o que permite uma maior flexibilidade no escalonamento e a utilização de intervalos de tempo que nunca seriam possíveis com o método original. Por outro lado, neste método assume-se sempre a existência de *buffers* de tamanho infinito. No método original, os comportamentos permitiam modelar a existência de *buffers* de qualquer tamanho (mas com a rigidez no escalonamento que lhes está associada).

Por fim, a maior diferença com o método original prende-se com a abordagem utilizada. Ao passar de uma abordagem centralizada para uma descentralizada tem-se vantagens e desvantagens. As principais vantagens relacionam-se com uma maior aproximação da realidade em termos de entidades modeladas e a diminuição de pontos de falha, já que o procedimento de escalonamento se encontra replicado em cada unidade recurso e tarefa. As principais desvantagens prendem-se com a velocidade de execução (principalmente devido à necessidade de comunicações).

No Apêndice A são apresentados os resultados de várias experiências efectuadas com o protótipo desenvolvido neste trabalho em comparação com os resultados do método original. Além dessas experiências são também apresentadas novas experiências para demonstrar a utilização de recursos alternativos.

## 6.4 Sistema Desenvolvido

Esta secção descreve o sistema informático concebido como protótipo para sustentação dos conceitos e arquitectura apresentados na tese. O protótipo, denominado *Fabricare* [Sousa *et al.*, 2000a], consiste numa série de aplicações informáticas para cada um dos diversos holons da arquitectura, bem como em alguns programas de suporte, tendo por objectivo o escalonamento de tarefas industriais.

### 6.4.1 Introdução

No Capítulo 5 foi apresentada uma arquitectura holónica para sistemas de produção e cada um dos holons constituintes dessa arquitectura foi especificado em termos de objectivos, ciclo de vida e, base de conhecimento, tendo sido em seguida descrito o seu modo de funcionamento nas secções anteriores deste capítulo.

A partir do exposto nesses capítulos partiu-se então para o desenvolvimento de um sistema computacional ao qual se deu o nome de *Fabricare* (Figura 6.37). O termo *Fabricare* é a palavra em Latim com o significado de fabricar. O sistema *Fabricare* é composto por várias aplicações e foi desenvolvido utilizando a linguagem de programação em lógica SICStus PROLOG para a implementação dos agentes do sistema e o Visual Basic para a construção da interface com o utilizador. Cada aplicação será apresentada com maior detalhe nas subsecções seguintes. A aplicação *Configuration Designer* permite definir os recursos existentes na instalação fabril e até certo ponto a disposição desses recursos na instalação fabril. Esta aplicação gera uma descrição do sistema que é utilizada pela aplicação *Deploy* para colocar em funcionamento os *Holons de Recursos*. O *Control Panel* funciona como interface para o sistema permitindo a criação de *Holons de Tarefa* através do *Lançador de Tarefas* que recolhe informação do utilizador sobre a tarefa a executar. A informação sobre os planos de produção de cada produto é colocada numa base de conhecimento usando a aplicação *Product Builder*.

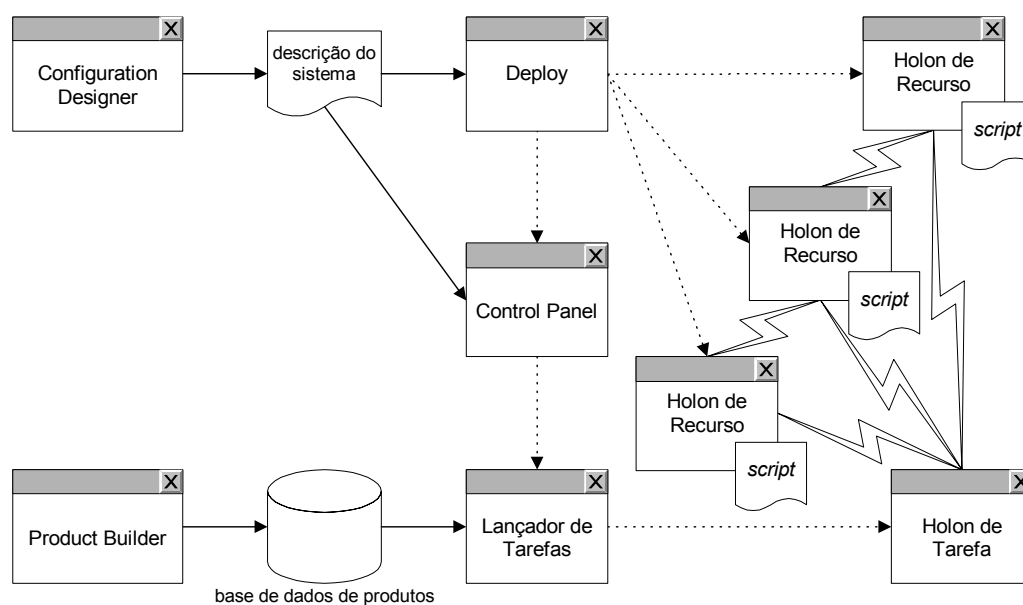


Figura 6.37 – Vista geral do protótipo *Fabricare*

A metodologia de trabalho associada à utilização do sistema é dada pela Figura 6.38.

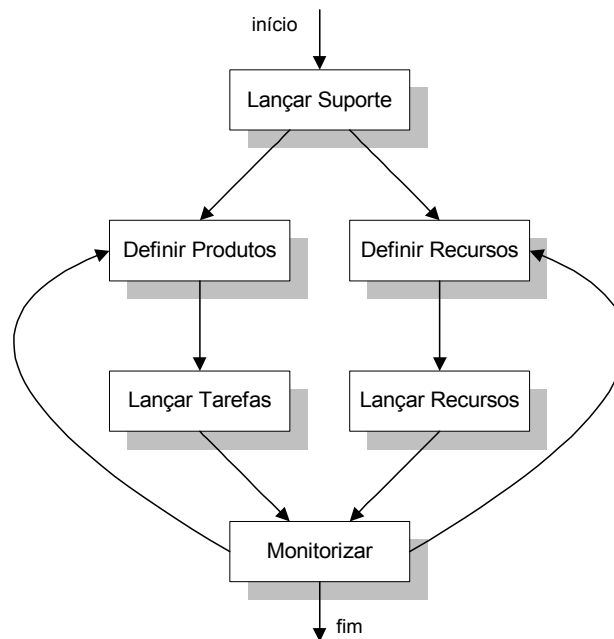


Figura 6.38 – Metodologia Fabricare

Deve começar-se por executar os mecanismos de suporte ao funcionamento do sistema (*e.g.*, o Holon de Serviços de Directório), utilizando para isso a aplicação *Deploy*. Em seguida, é possível definir os planos de produção dos produtos (através do *Product Builder*) ou a configuração dos recursos (através do *Configuration Designer*). Uma vez definida a informação sobre os produtos e recursos, é então possível colocar em execução os Holons de Recurso desejados, usando novamente a aplicação *Deploy*, e criar *Holons de Tarefa* usando o programa *Lançador de Tarefas*. Nesse momento o sistema está em funcionamento, podendo ser monitorizado (através do *Control Panel*). Eventualmente, a informação dos recursos e dos produtos pode ser alterada e o processo repetido.

#### 6.4.2 Núcleo de Holons

Os principais componentes do sistema são obviamente os Holons de Recurso e de Tarefa, no entanto, para o seu funcionamento necessitam de alguns holons de suporte.

Se estes holons de suporte são fixos e pré-definidos, o sistema é bastante dinâmico no que toca ao número de holons de recurso e principalmente de tarefa. Os primeiros estão dependentes do ficheiro de configuração gerado pelo *Configuration Designer*, enquanto que os segundos estão dependentes da interacção com o utilizador que vai criar novas tarefas.

##### Holons de Suporte

O elemento de suporte às comunicações do sistema é o servidor de “quadro negro” (Figura 6.39a) que potencia a interacção entre os diversos holons existentes no sistema. Para tal, é

utilizada a tecnologia de coordenação Linda [Carriero e Gelernter, 1989a] [Carriero e Gelernter, 1989b], descrita em termos de primitivas do SICStus PROLOG, que providencia um espaço de memória partilhado (quadro negro), onde os holons escrevem e lêem as suas mensagens através dos predicados utilitários *enviar\_msg* e *receber\_msg*. Internamente, estes predicados utilizam uma estrutura na forma *fabricare\_msg*(*Emissor*, *Destinatário*, *Conteúdo*) para representar cada mensagem lançada no sistema, em que o termo *Conteúdo* denota a mensagem propriamente dita; e.g., *lista\_recursos([id\_recurso\_1, id\_recurso\_2], tw(10, 20))*. Embora as comunicações sejam efectuados via uma estrutura de “quadro negro”, de um ponto de vista conceptual cada holon emissor está a colocar uma mensagem na caixa de correio (fila de mensagens) do holon destinatário e cada holon receptor lê as mensagens que estão actualmente na sua caixa de correio.



Figura 6.39 – Interface do protótipo Fabricare: holons de suporte

Além do servidor de comunicações, há que considerar mais três holons auxiliares:

- o *Holon de Serviço de Directório* (Figura 6.39b) que implementa o serviço de directório do sistema descrito nas secções 5.3.1, 5.4.2.2 e 6.3.1;
- o *Holon de Planeamento de Processos* (Figura 6.39c) para fornecimento dos planos de produção dos produtos aos Holons de Tarefa, respondendo às mensagens *obter\_plano*, conforme o que é descrito na secção 6.2.1.1. Este holon implementa também as funcionalidades de um Holon de Produto (descritas na secção 6.3.3); e
- o *Holon de Escalonamento* (Figura 6.39d) para o tratamento de conflitos entre tarefas a negociar intervalos de tempo sobrepostos para um mesmo conjunto de recursos,



conforme o que foi descrito na secção 6.2.2) e utiliza, também, os algoritmos descritos na secção 6.3.4.

Todos estes holons são programas de consola, ou seja, sem interface gráfica com o utilizador, já que a sua interacção é feita com outros holons no sistema e não com o utilizador (Figura 6.39).

### Holons de Tarefa

O programa de *Holon de Tarefa* (Figura 6.40) implementa as funcionalidades de um Holon de Tarefa de acordo com a descrição apresentada nas secções 5.3.3, 5.4.2.4 e 6.3.5.

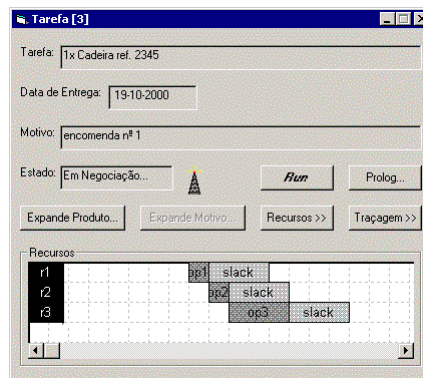


Figura 6.40 – Interface do protótipo Fabricare: holons de tarefa

Para cada tarefa criada o programa é executado, sendo a informação respeitante a cada tarefa passada ao holon que a representa, através da linha de comando do sistema operativo (correspondente à mensagem de anúncio do protocolo RCPR). No entanto, este programa é aberto no sentido em que permite a utilização de um *script* escrito em linguagem de programação em lógica PROLOG, para a definição do mecanismo de selecção de propostas a utilizar, podendo assim ser definidos novos critérios e novos mecanismos.

O seguinte programa em lógica é um exemplo de *script* para um Holon de Tarefa que define a ordem pela qual os critérios de selecção devem ser utilizados, definindo além disso, um novo critério (a nível de exemplo, pois não faz sentido usar um critério que maximize o custo de produção):

```

escolhe_criterio(maior_custo).
escolhe_criterio(primeira).
escolhe_criterio(maior_folga).
escolhe_criterio(menor_custo).

escolhe_k(maior_custo, ListaPreProc, ListaSel) :-
    maior_custo(ListaPreProc, alt(Path, Custo, LProp)),
    atribui(LProp, ListaSel).

maior_custo([S], S).
```

```

maior_custo([alt(Path, Custo, LProp)|T], alt(Path2, Custo2, LProp2)) :-
    maior_custo(T, alt(Path2, Custo2, LProp2)),
    Custo2 >= Custo.
maior_custo([S|T], S).

```

O gráfico de Gantt que é apresentado na Figura 6.40 corresponde ao escalonamento das operações da tarefa, representando cada linha um dos recursos contratados e os blocos escuros os intervalos escalonados para cada operação. Os blocos mais claros denotam as folgas que cada operação possuía no que respeita ao recurso no momento do escalonamento. É possível obter informação sobre cada uma das operações, nomeadamente o seu início, a sua duração e qual o recurso contratado. É ainda possível obter informação sobre o produto em execução por esta tarefa, nomeadamente a visualização gráfica do seu plano de produção.

### Holons de Recurso

O programa de *Holon de Recurso* (Figura 6.41) implementa as funcionalidades de um Holon de Recurso de acordo com a descrição efectuada nas secções 5.3.4, 5.4.2.5 e 6.3.6.

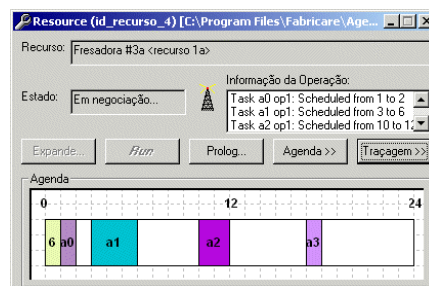


Figura 6.41 – Interface do protótipo Fabricare: holons de recurso

O gráfico de Gantt constante da Figura 6.41 corresponde à agenda de actividades do Holon de Recurso, sendo possível obter informação sobre cada uma das suas actividades (instante de início, duração, operação requisitada e tarefa contratante).

Cada Holon de Recurso é composto por um núcleo contendo as funcionalidades comuns a todos os holons e um *script* que contém as especificidades de cada Holon de Recurso individual.

O programa em lógica dado a seguir corporiza o *script* para um Holon de Recurso, informação esta que consta da base de conhecimento do holon:

```

tipo('holon-recurso').
recurso(1, 'Fresadora #3 <recurso 1>', data(2000, 11, 07), []).

habilidade(op1, 'Furar ângulo', 1, 100, 'prog-op1-r1.cnc', []).
habilidade(op24, 'Furar eixo Z', 1, 250, 'prog-op24-r1.cnc', []).

actividade(a0, op1, 1, 1, 1, data(2000, 11, 07), algum_estado, [], []).

```

Tanto o programa de Holon de Recurso como o de Holon de Tarefa possuem funcionalidades que permitem ao utilizador aceder a uma imagem da execução do programa, apresentando os eventos e os dados mais relevantes. Uma outra característica comum aos dois programas está na possibilidade de se aceder directamente ao interpretador PROLOG, utilizando a base de conhecimento do holon em questão.

### 6.4.3 Ferramentas de Exploração

Embora os principais componentes do sistema sejam de facto os Holons de Tarefa e de Recurso, foi desenvolvido um conjunto adicional de programas que facilitam a exploração do sistema, e que são apresentados der seguida.

#### *Deploy*

O programa *Deploy* (Figura 6.42) permite desencadear os mecanismos de suporte ao funcionamento do sistema, ou seja, o “quadro negro” para serviços de comunicações, o Holon de Serviços de Directório, o Holon de Planeamento de Processos e o Holon de Escalonamento.

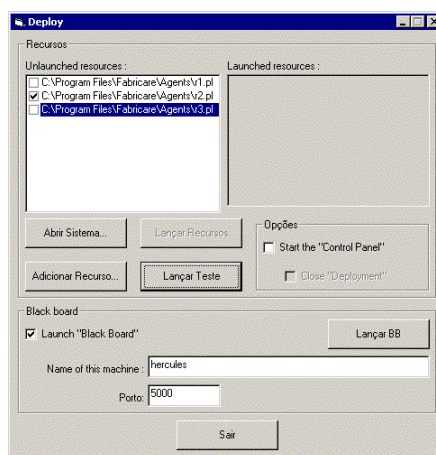


Figura 6.42 – Interface do protótipo Fabricare: aplicação “Deploy”

Adicionalmente, este programa permite também que se coloque em execução os Holons de Recurso distribuídos pelas diversas máquinas da rede informática. Para tal, um ficheiro de descrição do sistema criado pelo programa *Configuration Designer* é interpretado de forma a visualizar os recursos disponíveis. O utilizador tem então a possibilidade de escolher os recursos que pretende colocar em execução na máquina a que acede.

#### **Lançador de Tarefas**

O programa *Lançador de Tarefas* (Figura 6.43) lança os Holons de Tarefa. Este programa recolhe informação do utilizador acerca da tarefa a criar (e.g., tipo de produto, quantidade a produzir, data de entrega, cliente) e cria dinamicamente um Holon de Tarefa para a sua execução.

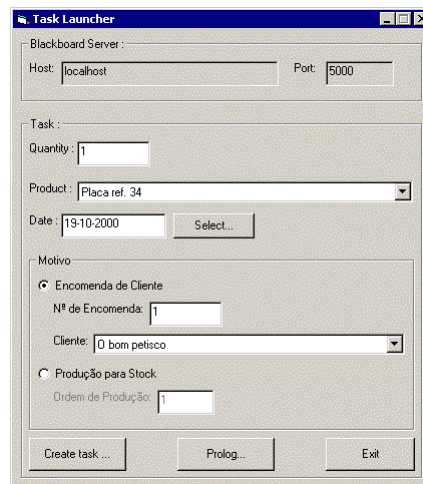


Figura 6.43 – Interface do protótipo Fabricare: aplicação “Lançador de Tarefas”

Este programa recolhe informação acerca dos produtos a fabricar e dos clientes da empresa numa base de dados do sistema.

#### 6.4.4 Programas Adicionais

Para além do núcleo de holons e dos programas de exploração, o sistema possui também três aplicações que de forma gráfica simplificam algumas tarefas a executar por parte do utilizador.

##### *Configuration Designer*

A aplicação *Configuration Designer* (Figura 6.44) permite definir, graficamente, a topologia dos recursos na instalação fabril.

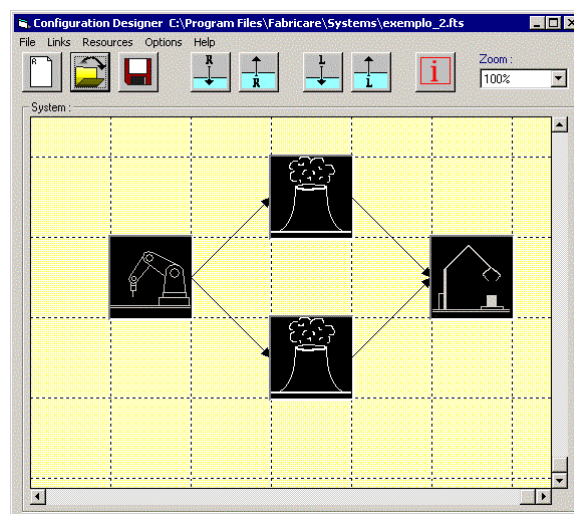


Figura 6.44 – Interface do protótipo Fabricare: aplicação “Configuration Designer”

Usando esta aplicação é possível definir, até certo ponto, a disposição dos recursos na instalação fabril e, indicar as ligações entre recursos (*e.g.*, robôs e passadeiras rolantes). Para cada recurso define-se também qual o *script* de Holon de Recurso a considerar.

### **Control Panel**

A aplicação *Control Panel* (Figura 6.45) funciona como interface para a operação do sistema, permitindo monitorizar o funcionamento dos vários holons e oferecendo uma vista global do sistema ao utilizador. Adicionalmente, esta aplicação permite, ainda, criar tarefas através da invocação da aplicação *Lançador de Tarefas*.

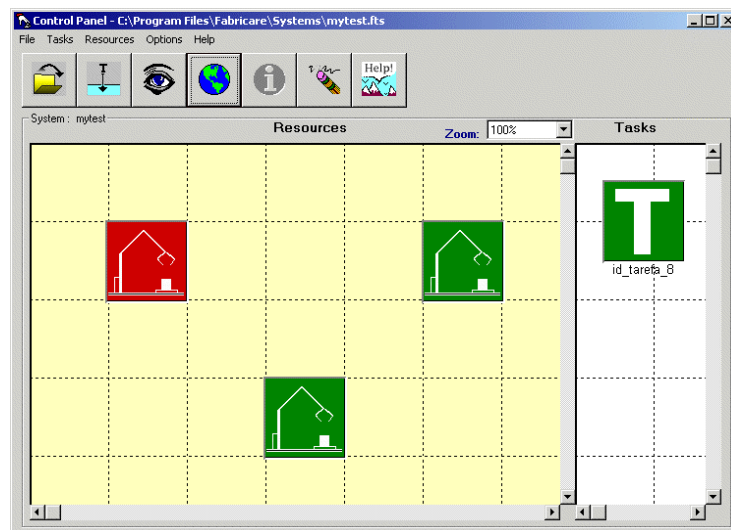


Figura 6.45 – Interface do protótipo Fabricare: aplicação “Control Panel”

No seu estado actual, permite visualizar o ficheiro de configuração, e verificar que holons de Recurso e de Tarefa se encontram em execução, questionando o Holon de Serviços de Directório, através da invocação do predicado *holons\_em\_execução*, definido em termos das produções:

$$\begin{aligned} \text{holons\_em\_execução(ListaHolons)} \leftarrow \\ \text{enviar\_msg(id\_srv\_dir, em\_execução)} \wedge \\ \text{receber\_msg(id\_srv\_dir, r\_em\_execução(LHolons))} \end{aligned}$$

Uma das principais funções desta aplicação é a de fornecer uma visão global do sistema, independentemente do posto de rede onde as aplicações de Holon de Recurso se encontram a correr (Figura 6.45).

É possível observar graficamente quais os holons existentes no sistema, bem como o seu estado (activo ou desactivo), sendo também possível, questionar cada holon sobre informação respeitante à sua agenda de actividades (Figura 6.46 para os Holons de Recurso e Figura 6.47 para os Holons de Tarefa).

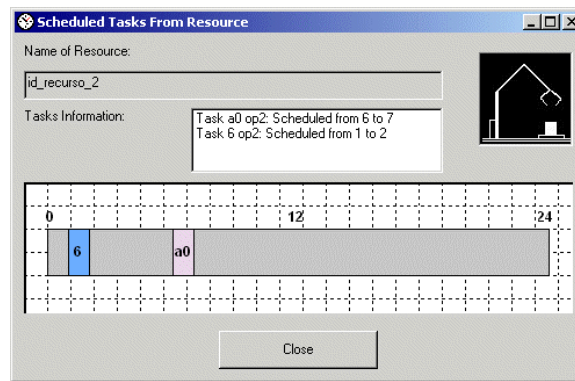


Figura 6.46 – Interface do protótipo Fabricare: escalonamento de um Holon de Recurso

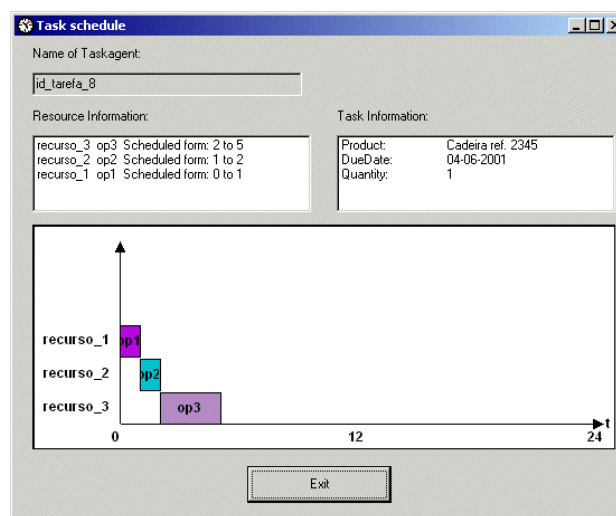


Figura 6.47 – Interface do protótipo Fabricare: escalonamento de um Holon deTarefa

### Product Builder

A aplicação *Product Builder* (Figura 6.48) permite definir graficamente os planos de produção de cada produto. Esta aplicação foi criada para facilitar a criação e a edição de tais planos evitando assim o manuseamento directo das extensões dos predicados que definem um plano na base de conhecimento de um Holon de Produto.



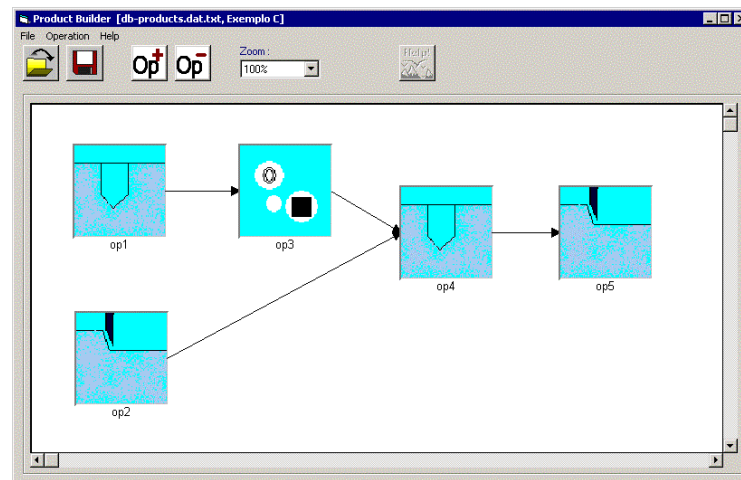


Figura 6.48 – Interface do protótipo *Fabricare*: aplicação “Product Builder”

Em condições ideais esta aplicação seria substituída pelo Holon de Planeamento de Processos que utilizaria os Holons de Produtos e de Recursos para gerar os planos.

#### 6.4.5 Comentários ao Desenvolvimento do Protótipo *Fabricare*

Essencialmente os comentários ao desenvolvimento do protótipo prendem-se com a escolha das linguagens de programação utilizadas, Visual Basic (VB) e SICStus PROLOG (SP), e com o mecanismo de comunicação (quadro negro).

Por um lado, a utilização de PROLOG como linguagem de programação permitiu reduzir o tempo de desenvolvimento, quando comparado com outras linguagens de programação (*e.g.*, C++), devido principalmente à natureza declarativa da linguagem.

Por outro lado, a utilização de uma linguagem de programação em lógica facilitou a prototipagem, ou seja, a transição de uma especificação formal do problema (*i.e.*, dos holons) para a implementação propriamente dita. Adicionalmente, a representação de informação incompleta foi também facilitada pela utilização da programação em lógica.

Um outro factor em consideração tem a ver com a escolha da linguagem de programação em lógica PROLOG, mais concretamente, do SICStus PROLOG, dado que apresenta uma biblioteca de coordenação de eventos baseada na tecnologia Linda, que permite uma fácil implementação de quadros negros, simplificando as comunicações entre holons (no entanto, um “quadro negro” pode gerar um grave gargalo de eficiência pois introduz um elemento centralizador no sistema [Nwana *et al.*, 1996b]). As mensagens trocadas entre holons são constituídas por extensões de predicados, enquanto que, por exemplo, numa solução C++, as mensagens seriam objectos C++, implicando por isso a implementação de todo o código para a construção da mensagem, envio e

recepção pela rede, tendo especial cuidado com a reconstrução dinâmica das mensagens do lado receptor. Todas essas tarefas foram simplificadas pela biblioteca Linda.

A combinação do Visual Basic (VB) com o SICStus PROLOG (SP) permitiu a criação das interfaces gráficas com o utilizador, desenvolvidas em VB e a utilização do núcleo dos programas em PROLOG. O desenvolvimento de interfaces em VB é extremamente simples, e criou uma divisão clara entre procedimentos (manipulação de dados) e apresentação (visualização de dados).

A combinação VB/SP não se apresentou no entanto sem alguns problemas. Um dos principais foi a falta de ligação bidireccional entre o VB e o SP. Por outras palavras, é possível executar predicados PROLOG a partir do VB, mas não o contrário. A invocação de procedimentos VB a partir do PROLOG teria sido útil no que toca à actualização da interface com o utilizador. Adicionalmente, devido à não existência de um mecanismo de sinalização de existência de mensagens em espera no “quadro negro”, essa verificação teve que ser originada a partir do VB, o que implicou a existência de um ciclo temporizado para verificar a existência de mensagens, e processa-las caso existam e proceder à actualização da interface, reduzindo assim o desempenho dos holons.

Por outro lado, devido à natureza distribuída da aplicação, os testes no terreno tornaram-se difíceis de efectuar. Este facto aliado ao facto de ser impossível a invocação do *debugger* PROLOG a partir do VB tornou bastante difícil a tarefa de teste e verificação do funcionamento dos holons.

## 6.5 Comparação com Trabalhos Relacionados

O trabalho aqui desenvolvido foi apresentado no Capítulo 5 e em secções anteriores a esta neste capítulo, sendo agora possível efectuar uma comparação com os trabalhos de referência citados na secção 4.5.1.

É possível então, passar a uma comparação com os trabalhos de referência (Tabela 6.4, tendo por base a Tabela 4.2) segundo quatro perspectivas:

- âmbito e estrutura;
- socialização;
- informação incompleta;
- procedimento de escalonamento.



Tabela 6.4 – Comparação entre o sistema *Fabricare* e trabalhos relacionados

	Âmbito	Entidades modeladas	Socialização	Informação Incompleta	Procedimento Escalonamento
<b>AARIA</b>	Escalonamento e gestão da cadeia de fornecimento	Partes, Recursos e Processos Unitários	Protocolo de rede de contrato	n.a.	Algoritmo DESK efectuado pelos agentes Recurso e Parte
<b>HMS “testbed”</b>	Escalonamento e Controlo	Recursos, Escalonador	Holarquia; troca de mensagens entre holons	n.a.	Método centralizado efectuado pelo holon escalonador
<b>Gou &amp; Luh</b>	Escalonamento	Máquinas, Células, Peças, Produtos, Coordenadores	Holarquia; troca de mensagens entre holons	n.a.	Relaxação Lagrangiana, efectuada pelos holons de Máquina/Célula e Peça/Produto
<b>PROSA (HoMuCS)</b>	Arquitectura de referência vocacionada para Integração empresarial (Controlo)	Produtos, Recursos, Ordens, Auxiliares (escalonador)	Holarquia; favorece a troca directa de mensagens (pedido-resposta)	n.a.	Método centralizado efectuado pelo holon escalonador
<b><i>Fabricare</i></b>	Escalonamento (integração empresarial)	Tarefas, Recursos e Produtos (fornecedores, clientes, compras e vendas)	Protocolo de rede de contrato com propagação de restrições	Representação de informação incompleta, identificação de casos e alguma manipulação	Adaptação do método de Almeida (1995) efectuado pelos Holons de Tarefa e de Recurso

Nas subsecções seguintes são elaborados alguns comentários sobre esta comparação de sistemas. Uma nota no que se refere ao trabalho PROSA, conforme foi referido, PROSA é uma arquitectura de referência e não um sistema. Por esse motivo é considerado ao longo das subsecções seguintes um sistema designado HoMuCS [Langer, 1999], que foi implementado de acordo com as orientações PROSA. Esse sistema aplica-se ao escalonamento e controlo de células de fabrico.

### 6.5.1 Âmbito e Estrutura (Entidades Modeladas)

No que se refere às áreas de aplicação dos vários trabalhos que são objecto de apreciação, é de realçar a escolha dos processos de escalonamento e controlo como área de teste. É também de notar a orientação para o controlo dos trabalhos oriundos de grupos com fortes raízes em Engenharia Mecânica (HMS “testbed” e HoMuCS/PROSA), ao passo que o escalonamento é o domínio preferido dos grupos com raízes nas Ciências da Computação (AARIA, Gou & Luh, *Fabricare*).

O protótipo *Fabricare* tem obviamente o seu âmbito de acção limitado ao Escalonamento, no entanto, a arquitectura proposta na secção 5.2 tem um âmbito mais abrangente, podendo ser classificada como de “integração empresarial”. Adicionalmente, embora as entidades modeladas no protótipo sejam apenas Tarefas, Recursos e Produtos, na arquitectura são também identificadas e especificadas entidades para Fornecedores, Clientes, Compras e Vendas.

Uma das principais diferenças nos trabalhos de investigação na área de produção em geral e do escalonamento em particular, são as entidades escolhidas para modelar na arquitectura [Bongaerts *et al.*, 1996]. Conforme já foi referido anteriormente, PROSA [van Brussel *et al.*, 1998] [Wyns, 1999] é uma taxonomia de Sistemas Holónicos de Produção, sendo considerada por vários investigadores na área como uma arquitectura de referência. Por esse motivo, a comparação entre as diferentes arquitecturas já referenciadas vai ser efectuada tendo em conta os quatro tipos de holons identificados na taxonomia PROSA: (i) Produtos; (ii) Recursos; (iii) Ordens de Fabrico; e (iv) Auxiliares.

O sistema AARIA encaixa parcialmente nesta taxonomia, visto modelar recursos e produtos (através dos agentes Parte). No entanto, as ordens de fabrico não são modeladas como agentes, mas sim como mensagens trocadas entre o utilizador (cliente) e o sistema. Adicionalmente, AARIA modela uma dimensão extra não existente na taxonomia PROSA, ao modelar os processos como agentes.

O sistema HMS “testbed” é um produto oriundo do mesmo grupo de investigação que desenvolveu a taxonomia PROSA, tendo inclusive estado na sua génese. O sistema HoMuCS é um produto de um grupo estreitamente relacionado com o grupo PROSA que implementou um

sistema seguindo essa mesma arquitectura. Por esse motivo ambos os trabalhos (*i.e.*, HMS “testbed” e HoMUCS) se relacionam imenso com PROSA, sendo que o sistema HoMuCS adere totalmente a esta taxonomia.

A arquitectura de Gou & Luh embora não relacionada com PROSA, encaixa perfeitamente na taxonomia dada, visto que modela produtos como holons Parte (que também representam estágios intermédios de fabrico), recursos como holons Máquina e Célula (na arquitectura PROSA não existe esta distinção hierárquica, sendo uma célula considerada como um recurso) e ordens de fabrico como holons de Produto. Adicionalmente, existem holons auxiliares, como, por exemplo o Holon Coordenador de Célula.

A arquitectura proposta neste trabalho pode também ser classificada de acordo com a taxonomia PROSA, visto que as quatro classes de holons são modeladas na arquitectura. Os Holons de Recurso e de Produto possuem o mesmo nome que as classes de PROSA, e a classe Ordem de Fabrico é representada pelos Holons de Tarefa. Adicionalmente, o Holon de Serviços de Directório é um exemplo de um holon auxiliar.

Um outro aspecto relativo à arquitectura de cada um dos sistemas referenciados prende-se com a existência de elementos centralizadores. O sistema HMS “testbed” utiliza um holon para escalonamento e como tal tem aí uma função centralizada. Gou & Luh utilizam holons coordenadores de célula que reúnem informação e participam activamente no cálculo do escalonamento. O sistema AARIA reclama a inexistência de elementos centralizadores, no entanto, na representação gráfica do sistema (Figura 4.6), existe um componente “Gestor” que não é explicado pelos autores.

O sistema *Fabricare* possui um holon que centraliza, actualmente, uma função, a gestão de conflitos. Esse holon (Holon de Escalonamento) serializa as negociações e todos os Holons de Tarefa necessitam de comunicar com ele, no entanto, a sua participação é diminuta e, apenas no início do processo de negociação. Por outro lado, a arquitectura proposta assenta numa abordagem completamente distribuída ao contrário de PROSA que favorece elementos centralizadores (só os recursos, ordens de fabrico e produtos são distribuídos por especificação da arquitectura).

### 6.5.2 Socialização

No que toca a mecanismos de interacção entre holons, são seguidas duas abordagens distintas, orientando-se uma mais para a negociação, e a outra para a simples troca de mensagens, com instruções de comando ou pedido de informação.

O sistema HMS “testbed” utiliza uma filosofia de pedido-resposta entre o holon de controlo e o holon de escalonamento. O holon de controlo pede ao holon de escalonamento que efectue um

plano para o escalonamento das ordens de fabrico existentes no sistema e em seguida executa esse plano. Quando não é possível efectuar as diferentes actividades (*i.e.*, operações) de acordo com o escalonamento sugerido, o holon de controlo decide o que executar requisitando ao holon de escalonamento um novo escalonamento. A execução do plano é efectuada mediante a troca de mensagens de controlo (*i.e.*, *master/slave*) entre o holon de controlo e os holons de recurso. PROSA e HoMUCS seguem uma filosofia idêntica.

Ao contrário do que foi referido em epígrafe, os sistemas AARIA, Gou & Luh e *Fabricare* privilegiam a negociação. AARIA possui uma estrutura plana, onde cada agente age em parceria, não existindo agentes hierarquicamente distintos. A interacção entre os agentes baseia-se no Protocolo de Rede de Contrato. Gou & Luh introduzem uma estrutura de decomposição (devido à natureza holónica da arquitectura) onde cada holon trabalha em parceria com os outros holons dentro da mesma holarquia mas não comunica com holons fora dessa holarquia. A comunicação consiste na troca de resultados parciais do problema de relaxação Lagrangiana.

O sistema *Fabricare* baseia-se também no Protocolo de Rede de Contrato para a negociação de operações entre os recursos e as tarefas. No entanto, devido à necessidade de cooperação (e coordenação) entre os vários recursos intervenientes na execução de um plano de produção, o PRC original é modificado por forma a incluir duas novas fases, onde os recursos trocam informação entre si, para coordenar os seus processos de escalonamentos. Todos os holons de recurso e de tarefa são vistos como pares no processo de negociação.

### 6.5.3 Tratamento de Informação Incompleta

Nenhum dos trabalhos de referência (PROSA/HoMUCS), HMS “testbed”, AARIA e Gou & Lu) aborda a problemática da informação incompleta. No entanto, o grupo de investigação do projecto AARIA desenvolveu um outro trabalho relativo a gestão de cadeias de fornecimento (DASch [Parunak, 1998b]) onde modela a incerteza nos canais de distribuição (nomeadamente no tempo de percurso).

Ao contrário dos trabalhos tomados para comparação (e mesmo outros trabalhos referidos no “estado da arte”), o sistema *Fabricare* permite o tratamento e representação de informação incompleta, utilizando para isso a notação descrita na secção 5.4.1. Na secção 5.4.2 foram identificados casos de informação incompleta nos sistemas de produção (mais concretamente, na base de conhecimento dos holons constituintes da arquitectura proposta), casos esses, que foram implementados nos holons do protótipo *Fabricare*, ou seja, nos Holons de Recurso, Holons de Tarefa e Holons de Produto.

Neste trabalho, contudo, essa informação não é utilizada na sua plenitude, nos processos de operação dos holons, sendo apenas utilizada pelo Holon de Serviços de Directório relativamente ao conhecimento sobre as funcionalidades de outros holons.

#### 6.5.4 Procedimento de Escalonamento

O método de escalonamento utilizado nos sistemas referenciados, é bastante distinto, quer no tipo de escalonamento, quer na distribuição dos procedimentos pelos diversos elementos constituintes da arquitectura.

O sistema HMS “testbed” utiliza um holon de escalonamento centralizado em colaboração com um holon de controlo. Também PROSA e HoMUCS favorecem a utilização de métodos centralizados para o escalonamento, normalmente através de holons auxiliares, que efectuem um escalonamento usando uma qualquer técnica bem conhecida e fornecem esse escalonamento a cada holon de ordem de fabrico.

No sistema AARIA utiliza-se um método bastante flexível que não impõe um momento de arranque fixo para as tarefas, baseando-se antes no acordo feito com o cliente, para decidir a ordem de execução das várias tarefas da forma mais eficiente (do ponto de vista do recurso). O acordo com o cliente passa pela escolha de um ponto numa função de custo/prazo-de-entrega, por utilização do procedimento DESK [Sauter e Parunak, 1999], baseada em janelas temporais e na diminuição de custos para prazos de entrega de produtos mais alargados. Cada agente de recurso efectua o escalonamento local das operações requisitadas.

Gou & Luh utilizam um processo fundamentalmente matemático, tentando minimizar um conjunto de funções através de programação dinâmica com relaxação Lagrangiana. Em primeiro lugar cada célula calcula o seu escalonamento (propagando resultados entre os holons de máquina existentes na célula) e em seguida o escalonamento da fábrica é calculado usando os resultados das células.

Conforme o que já foi descrito anteriormente ao longo deste capítulo, o procedimento de escalonamento utilizado no sistema *Fabricare* é uma adaptação de um método centralizado. O escalonamento baseia-se no Protocolo de Rede de Contrato, sendo efectuado pelos vários recursos intervenientes na obtenção de um produto (requisitado por uma tarefa) através da propagação de restrições para determinar as janelas temporais efectivas de cada recurso.

## 6.6 Resumo do Capítulo

Devido à natureza distribuída da arquitectura apresentada na secção 5.2 torna-se necessário regulamentar a interacção entre os vários holons, especialmente os Holons de Tarefa e de Recursos para a actividade de escalonamento (escolhida como caso de teste para o modelo conceptual e arquitectura). Por esse motivo, na secção 6.2 foi apresentado o protocolo de negociação que rege o processo de contratação de serviços entre Tarefas e Recursos. Esse protocolo está preparado para o tratamento de excepções e, principalmente para evitar o *Problema de Indecisão*. Em seguida procedeu-se a uma análise da complexidade do protocolo no que toca ao número de soluções encontradas, ao número de mensagens trocadas e ao tamanho total das mensagens trocadas.

Tendo sido especificado o protocolo, foi então descrito o funcionamento de cada holon. Os principais algoritmos para a operação dos holons directamente relacionados com o escalonamento (*Holon de Tarefa*, *Holon de Recurso*) e de outros holons necessários (*Holon de Serviço de Directórios*, *Holon de Produto* e *Holon de Escalonamento*) foram então apresentados e explicados, estabelecendo-se a ponte entre as actividades descritas no protocolo para cada um dos holons e os seus procedimentos internos.

De seguida apresentou-se o protótipo de sistema que foi implementado, seguindo as descrições efectuadas quer no Capítulo 5 quer no Capítulo 6. Este protótipo denominado *Fabricare* é composto por um conjunto de aplicações que incluem os Holons de Tarefa e de Recurso, bem como os holons auxiliares (Serviço de Directório, Planeamento de Processos, Escalonamento) e, algumas ferramentas para a exploração do sistema, permitindo colocar em execução os Holons de Recurso e a criação de tarefas. Nesta secção foram também feitos alguns comentários às diferenças entre o método original de escalonamento e o método utilizado neste trabalho, bem como alguns comentários ao desenvolvimento do protótipo.

Finalmente, foi apresentado uma comparação entre o trabalho desenvolvido e os trabalhos de referência mencionados no “Estado da Arte”. Essa comparação foi efectuada segundo quatro linhas de interesse, de acordo como os objectivos enunciados, em termos de estrutura, socialização das entidades, tratamento de informação incompleta e procedimento de escalonamento ■