

Processamento Distribuído Aberto - ODP

Tetsu Gungi
DCA-FEEC-UNICAMP

Agosto, 1995

1 Modelo de Referência ODP

O rápido crescimento do processamento distribuído tem levado a necessidade de uma padronização [7, 20, 22, 1]. O modelo de referência ODP (Open Distributed Processing) cria uma arquitetura que dá suporte à distribuição, interconexão e portabilidade.

O Modelo Básico de Referência do Processamento Distribuído Aberto (RM-ODP), é baseado nos conceitos derivados das pesquisas e desenvolvimentos na área de processamento distribuído.

O RM-ODP consiste de:

- visão geral e guia de uso do modelo de referência [9]: contém a motivação do ODP dando o escopo, justificação e explicação dos conceitos chaves, e algumas linhas gerais da arquitetura ODP. Ele explica como o RM-ODP deve ser entendido e aplicado pelos usuários. Esta parte não é normalizada;
- modelo descriptivo [10]: contém a definição de conceitos, estruturação e notação para uma descrição normalizada de sistemas de processamento distribuído. Este é apenas um nível de detalhe para dar suporte ao modelo prescritivo e para estabelecer requerimentos para novas técnicas de especificação. Esta parte é normalizada;
- modelo prescritivo [11]: contém as especificações das características que qualificam um sistema de processamento distribuído como aberto. Nele estão os requerimentos do modelo de referência ODP. Esta parte é normalizada;
- semântica arquitetônica [12]: contém a formalização do conceito de modelamento ODP definido no modelo descriptivo. A formalização é atingida interpretando cada conceito em termos da construção de diferentes técnicas de descrição formal padronizadas [26]. Esta parte é normalizada.

2 Sistemas ODP

O resultado do modelo de referência ODP possibilita a implementação de sistemas ODP [4] que podem operar consistemente e confiavelmente enquanto está permitindo a distribuição de objetivos, recursos e atividades. Tais sistemas utilizam componentes que podem ser:

- heterogêneos;
- capazes de operação concorrente [17, 29];
- fisicamente e/ou logicamente distribuídos;
- portáveis;
- capazes de cooperar com outro sistema.

Neste contexto, heterogeneidade se refere a:

- equipamentos;
- sistemas operacionais;
- computação (linguagens, banco de dados, etc);
- autoridade (grupos de usuários com diferentes formas de acesso);
- aplicação (integração de planejamento e produção).

3 Organização do Modelo de Referência ODP

O modelo de referência ODP pode ser dividido em categorias. As categorias mais gerais contém as mais específicas. As categorias são:

- o modelo geral de referência (RM-ODP) que define os conceitos e a identificação de funções comuns;
- modelos de referência específicos que cobrem tipos especiais de organizações usando os conceitos e funções do RM-ODP, e definindo detalhes conceituais e funções específicas adicionais, por exemplo *Telecommunication Information Networking Architecture*;
- padrões para a realização de funções comuns e genéricas para um espectro amplo de aplicações, por exemplo *Trader* [8] ou uma linguagem de definição de interface;
- padrões para a realização de funções específicas de aplicações particulares, por exemplo interfaces de chamadas de conexões telefônicas.

O RM-ODP estabelece um estilo e uma base para o projeto e implementação de sistemas, permitindo o uso de componentes, métodos de projeto e representação que já são comuns. O RM-ODP deve:

- prover um modelo consistente, permitindo que a consistência possa ser mantida entre múltiplos padrões desenvolvidos separadamente;
- definir as áreas onde o modelo de referência ODP é necessário, o relacionamento entre os padrões ODP e os outros padrões;

- descrever um conjunto básico de ferramentas a serem usadas na definição do modelo de referência ODP;
- prover um conjunto consistente de conceitos e terminologias comuns.

4 Pontos de Vista do ODP

Ao invés de lidar com toda a complexidade de um sistema distribuído, pode-se ver o sistema de pontos de vista diferentes, cada qual refletindo um conjunto de atributos diferentes. Cada ponto de vista representa um nível de abstração diferente do sistema distribuído original, sem a necessidade de criar um grande modelo que descreva todos estes níveis em detalhes [20].

Especificando um sistema utilizando diferentes pontos de vista permite-se que um sistema grande e complexo possa ser separado em partes gerenciáveis, cada um focalizando assuntos relevantes para membros diferentes da equipe de desenvolvimento [22].

Informalmente, um ponto de vista leva a uma representação do sistema com ênfase sobre um determinado assunto. Mais formalmente, o resultado de uma representação é uma abstração do sistema; isto é, a descrição que reconhece algumas distinções (aqueles relevantes para o nível em questão) e ignora outras (aqueles que não são relevantes para o nível em questão).

O modelo de referência ODP reconhece cinco pontos de vista:

- ponto de vista empresarial, que enfoca as políticas de negócios, políticas de gerenciamento e as regras do usuário humano em relação ao sistema e o ambiente na qual eles interagem: o uso da palavra empresa aqui não implica numa limitação para uma única organização; o modelo construído pode descrever muito bem a interação de um grande número de organizações distintas. Esta visão procura caracterizar os requisitos empresariais;
- ponto de vista de informação, que enfoca o modelamento da informação, provendo uma visão comum consistente cobrindo as fontes da informação, os destinos da informação e o fluxo de informações entre eles. Modelam-se as estruturas e o fluxo de informação que devem governar a sua manipulação;
- ponto de vista computacional, que enfoca os algoritmos e estruturas de dados e com a definição dos requisitos de transparência e distribuição;
- ponto de vista de engenharia, que enfoca os mecanismos de distribuição e a provisão de vários tipos de transparência necessários para dar suporte a distribuição;
- ponto de vista tecnológico, que enfoca os detalhes de componentes e *links* com o qual um sistema distribuído é construído. Deve mostrar qual o *software* e o *hardware* que devem ser utilizados para construir o sistema.

A figura 4 mostra como os pontos de vista do RM-ODP devem ser utilizados nas diversas fases do ciclo de vida de um sistema ODP. Por exemplo o ponto de vista empresarial leva

à análise dos requisitos e políticas do sistema; os pontos de vista de informação e computacional fornecem documentos válidos para a fase de especificação funcional, por exemplo, especificações dos requisitos da informação (sua organização, aquisição, processamento, armazenamento e representação no sistema ODP) e os modelos de interação (acesso a todas as interfaces) e de construção (infraestrutura distribuída para programação de funções). O ponto de vista de engenharia permite a especificação das funções de processamento, armazenamento e comunicação necessárias à implementação do sistema: em particular deverão considerar-se aqui todos os requisitos de transparência de distribuição, de portabilidade e de interconectividade. Finalmente, através do ponto de vista tecnológico, deverão ser formulados requisitos de implementação, identificadas tecnologias a serem usadas, casos de testes, etc [28].

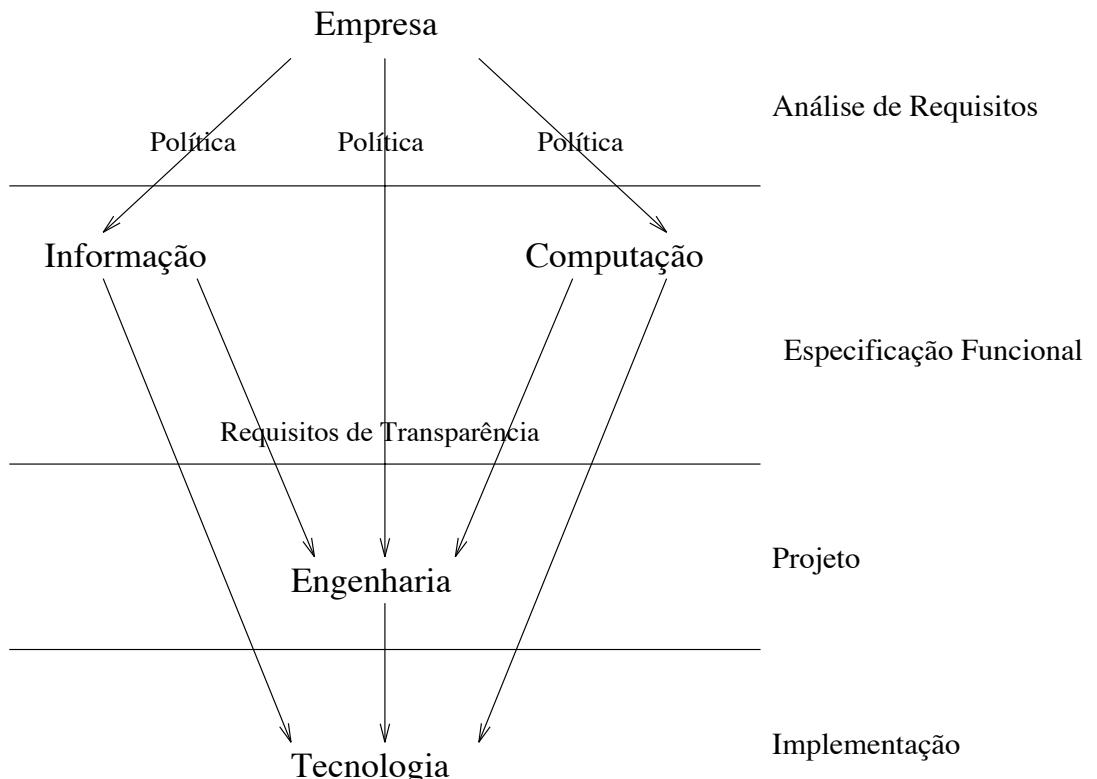


Figura 1: Uso de pontos de vista no ciclo de vida

5 O Modelo Prescritivo

O modelo prescritivo define linguagens para cada ponto de vista e então identifica as funções chave relacionadas com estas linguagens. As definições são expressas em termos

presentes no modelo descritivo [10]. As informações acerca das linguagens empresarial, de informação, computacional e tecnológica foram resumidas enquanto que a linguagem de engenharia foi descrita de uma forma mais completa.

5.1 Linguagem Empresarial

A linguagem empresarial está preocupada com políticas, regras organizacionais e restrições do sistema. Ela deve suportar as decisões e objetivos organizacionais da qual muitas escolhas de projeto irão depender [6, 5]. Por exemplo, políticas de segurança estabelecidas pelo ponto de vista empresarial determinarão muitas das subsequentes escolhas dos mecanismos de segurança.

A linguagem empresarial é expressa em termos da identificação de um conjunto de atividades chaves e as regras envolvidas para executá-las. Estes conceitos são usados para forçar as ações disponíveis em uma dada situação e para expressar obrigações que devem ser cumpridas.

5.2 Linguagem de Informação

A linguagem de informação é expressa em termos dos objetos abstratos que representam a informação manipulada pela empresa. Nesta linguagem, a distribuição do processamento ainda não é visível.

As técnicas aplicadas são provavelmente familiares aos projetistas de sistemas de informação, mas um cuidado particular deve ser tomado para assegurar que a decisão tomada não prejudique na preparação para a distribuição.

5.3 Linguagem Computacional

O ponto de vista computacional lida com o particionamento lógico das funções, quebrando o problema em fluxos de invocação e provedores de serviço. Aqui as regras de interação entre clientes e servidores tornam-se importantes [14].

Existem dois principais aspectos relacionados com a linguagem computacional. O primeiro é o modelo de interação, que introduz os conceitos de invocação e anúncio para representar interações com ou sem resposta. Este modelo estabelece a interpretação da parametrização e mostra a semântica de uma falha para uma interação. Ele define os requerimentos sobre um tipo de interface do sistema e as regras de *matching* entre interfaces que devem ser satisfeitas antes que uma ligação entre eles possa ser estabelecida [15].

A segunda principal parte da linguagem computacional é relacionada com a construção da configuração de objetos [21, 27], e o suporte para a criação de complexas redes de objetos interagentes, estipulando as regras que governam seu estabelecimento e sua desconexão.

Portanto a linguagem computacional define o meio pela qual os objetos podem interagir e estabelece os mecanismos pelos quais eles podem ser criados e depois destruídos, provendo os meios pelas quais configurações de sistemas complexos são construídos. Estas definições formam a base para a identificação das funções genéricas necessárias para dar suporte a especificação computacional, tal como a noção de um “fabricante” de objetos ou de um *trader*[8] para negociar e buscar serviços.

5.4 Linguagem de Engenharia

A linguagem de engenharia é um conjunto de conceitos, regras e estruturas para a especificação de um sistema ODP sobre o ponto de vista de engenharia.

Uma especificação de engenharia é utilizada para:

- descrever a organização de uma forma abstrata possibilitando a execução das funções de um sistema ODP;
- identificar as abstrações necessárias para gerenciar recursos físicos locais e distribuídos do sistema;
- identificar e definir os papéis dos diferentes objetos que dão suporte as funções ODP (por exemplo, funções de transparência);
- identificar os pontos em comum entre os diferentes objetos.

5.4.1 Conceitos Chave

A linguagem de engenharia contém os conceitos do ITU-T Rec.X.902 — ISO/IEC 10746-2 [10] e os que estão definidos abaixo.

- núcleo: um objeto que coordena processamento, armazenagem e funções de comunicação para uso por outros objetos de engenharia dentro do nó da qual ele pertence. Um sistema operacional (o *kernel* é um exemplo de um núcleo);
- cápsula: uma configuração de objetos formando uma unidade para o propósito de processamento e armazenagem. Um processo de um sistema operacional é um exemplo de uma cápsula;
- gerenciador da cápsula: um objeto que gerencia os objetos numa cápsula;
- *cluster*: uma configuração de objetos básicos de engenharia formando uma unidade para o propósito de desativação, *checkpoint* [3], recuperação, reativação e migração;
- gerenciador do *cluster*: um objeto que gerencia os objetos num *cluster*;
- *cluster checkpoint* : a criação de uma cópia do *cluster* quando o *cluster* se encontra num estado consistente;
- *checkpointing*: a operação de criar um *cluster checkpoint*;
- desativação: *checkpointing* seguido de uma eliminação de um *cluster*;
- clonagem: estabelecimento de uma cópia de um *cluster* pela instanciação de um *cluster checkpoint*;
- recuperação: clonagem de um *cluster* após a falha ou eliminação de um *cluster*;
- reativação: clonagem de um *cluster* após sua desativação (previsto);

- **objeto básico de engenharia:** um objeto de engenharia que requer o suporte de uma infraestrutura distribuída;
- ***stub*:** um objeto que provê funções de conversão (por exemplo, conversão/desconversão para uma forma canônica de representação de dados para dar suporte ao acesso transparente na interação entre objetos de engenharia);
- ***binder*:** um objeto que mantém a comunicação entre os objetos de engenharia;
- **interceptador:** um objeto que está presente na fronteira entre dois domínios que:
 - executa checagens ou monitora políticas para permitir a interação entre domínios diferentes;
 - executa transformações para mascarar as diferenças na interpretação de dados nos domínios.

Um *gateway* é um exemplo de um interceptador;

- ***protocol*:** um objeto que comunica com outro objeto *protocol* para permitir a interação entre objetos de engenharia (possivelmente em outros *clusters*, cápsulas ou nós);
- **canal:** uma configuração de objetos *stub*, *binder*, *protocol* através da qual uma interação pode ocorrer;
- **interface de comunicação:** uma interface de um objeto *protocol* que está ligada a uma interface de um objeto interceptador ou de um outro objeto *protocol*
- **identificador de interface de engenharia:** um identificador para uma interface de objeto de engenharia [24] que está ligada a um canal;
- **referência da interface de engenharia:** um identificador para uma interface de objeto de engenharia que está disponível para o estabelecimento de uma comunicação¹;
- **domínio administrativo:** um conjunto de objetos cuja segurança, auditoria, monitoramento e outras funções de controle são controladas por uma administração comum;
- **domínio da comunicação:** um conjunto de objetos *protocol* capazes de se interagirem;
- **nó:** uma configuração de objetos formando uma unidade para o propósito de localização no espaço, e que incorpora um conjunto de funções de processamento, armazenagem e comunicação. Um computador e os *softwares* nele presentes correspondem a um nó;
- **migração:** mover um *cluster* para uma cápsula diferente.

¹Obs: Uma referência da interface de engenharia é apenas utilizada para o estabelecimento de uma comunicação, e é distinta do identificador de interface de engenharia que é utilizada para o propósito de interação.

5.4.2 Mapeamento de Linguagem de Computação em Linguagem de Engenharia

O objeto computacional pode ser suportado por funções de engenharia. Isto implica que pode-se fazer um mapeamento de um objeto computacional em funções de engenharia. Portanto para obter-se uma especificação de engenharia utiliza-se dos conceitos e regras de engenharia. O diagrama da figura 5.4.2 ilustra o mapeamento de um objeto computacional em uma configuração de objetos de engenharia, em termos de núcleo, cápsula e cluster.

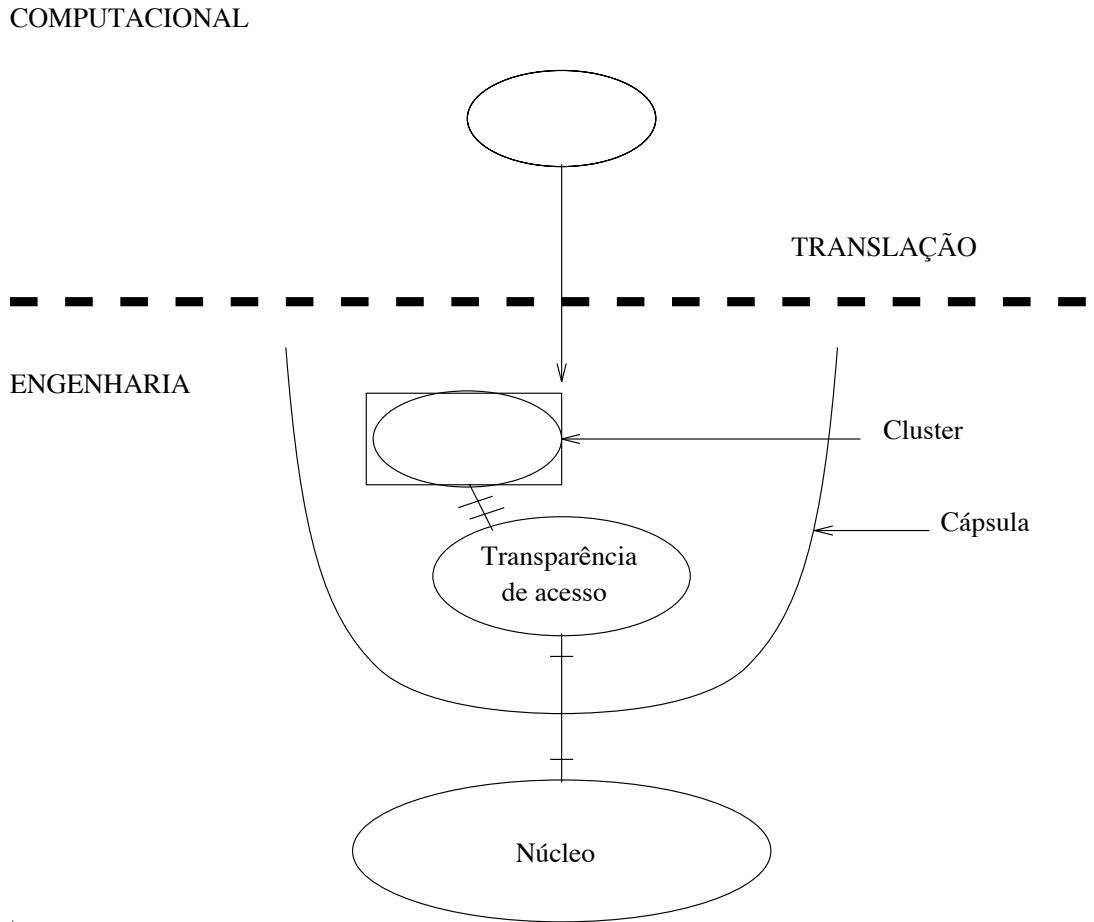


Figura 2: Translação da linguagem computacional para a de engenharia

O refinamento de *templates* computacionais [25, 14] em *templates* de engenharia (formando um *cluster*) corresponde a noção de compilar programas para produzir código objeto.

O refinamento de *templates* de engenharia em *templates* de *clusters* (formando uma cápsula) corresponde a noção de unir módulos para formar um programa executável.

O conceito de cápsula corresponde a noção de espaço de endereçamento ou processo na maioria dos sistemas operacionais.

As seguintes correspondências têm sido identificadas entre objetos computacionais e objetos básicos de engenharia:

- um objeto computacional com uma interface corresponde a um conjunto de objetos básicos de engenharia todos num único *cluster* ou apenas a um objeto básico de engenharia;
- um objeto computacional com várias interfaces podem ser executadas numa mesma estação de trabalho ou em múltiplas estações de trabalho.

5.4.3 Transparência

Na passagem da linguagem computacional para a linguagem de engenharia, a preocupação da especificação de estruturas computacionais e declarações das propriedades necessárias para a interação entre objetos é agora relacionada aos mecanismos de engenharia capazes de assegurar estas propriedades. Isto leva a identificação de uma série de transparências (conforme definidas na página 4).

O mecanismo de transparência situa-se entre o usuário e o sistema e possibilita que o usuário peça alguma requisição ao sistema não se preocupando com o comportamento do sistema para atender seu pedido.

O conceito de transparência têm muitas aplicações além do modelamento de um sistema ODP. A transparência é o resultado normal do processo de abstração que é utilizado para especificar qualquer grande sistema, sem necessariamente implicar em ser distribuído. No ODP os requerimentos estão limitados a garantias de várias formas de transparência de distribuição.

As transparências definidas na linguagem de engenharia são:

- transparência de acesso;
- transparência de localização;
- transparência de migração;
- transparência de concorrência (ou transação);
- transparência de recursos;
- transparência de falha;
- transparência de grupo;
- transparência de federação.

A figura 3 mostra o meio pela qual as linguagens computacional e de engenharia estão relacionadas pela introdução de transparências.

5.4.4 Estruturação de Engenharia

Uma especificação de engenharia define a infraestrutura necessária para dar suporte as funções de um sistema ODP, identificando:

- as funções ODP necessárias para o gerenciar distribuição física, comunicação, processamento e armazenamento;
- os papéis dos diferentes objetos que dão suporte as funções ODP (por exemplo o núcleo);
- as ligações entre os diferentes objetos.

Uma especificação de engenharia é expressa em termos:

- da configuração dos objetos de engenharia;
- das atividades que ocorrem dentro dos objetos [13];
- da interação que ocorrem entre os objetos [19, 23, 18].

Uma especificação de engenharia deve seguir regras da linguagem de engenharia. Estas regras são:

- regras de canal;
- regras de estabelecimento de um canal;
- regras de referência de interfaces;
- regras de *cluster*;
- regras de cápsula;
- regras de nó;
- regras de falha.

Regras de canal

Um canal é responsável pela transparência na interação entre objetos de engenharia. Isto inclui:

- interação entre um objeto cliente e um objeto servidor;
- um grupo de objetos interagindo com outro grupo de objetos;
- um fluxo de dados entre múltiplos objetos produtores e objetos consumidores.

Um exemplo de canal está ilustrada na figura 5.4.4.

Um canal é uma configuração de objetos *stubs*, *binders* e *protocols* e interceptadores conectados a um conjunto de objetos de engenharia. Se for criado um canal no mesmo domínio de comunicação não existe a necessidade de um interceptador.

Stubs

Os objetos de engenharia que estão interagindo são ligados a *stubs*. Os *stubs* são responsáveis pela conversão/desconversão de dados durante a interação. Os *stubs* podem controlar e manter registros (por exemplo, para segurança e auditoria). Eles podem interagir com outros objetos fora do canal (por exemplo, uma função de segurança) caso seja necessário.

Um *stub* tem:

- uma interface de apresentação para interação com um objeto de engenharia;
- uma interface de controle para gerenciamento da qualidade de serviço [2, 16].

Binders

Os *binders* em um canal gerenciam a integridade fim-a-fim e a qualidade de serviço do canal. Os *binders* devem prover transparência de relocação quando for necessário, monitoração de falhas de comunicação e reparar conexões quebradas.

Um *binder* tem interfaces de controle que possibilitam:

- mudanças na configuração do canal;
- destruição de todo, ou parte, de um canal.

Protocols

Os objetos *protocol* provêem funções de comunicação. Eles podem interagir com objetos fora do canal para obter informações adicionais. Quando os objetos *protocol* estão em domínios de comunicação diferentes eles requerem um interceptador para que possam se interagir entre si.

Interceptadores

Os interceptadores são responsáveis pela verificação e transformações durante uma interação entre domínios diferentes; portanto eles necessitam ter acesso aos tipos das interfaces dos objetos de engenharia que estão unidas pelo canal em que o interceptador está localizado.

Regras de estabelecimento de um canal

O núcleo estabelece canais configurando adequadamente objetos *stubs*, *binders*, *protocols* e interceptadores. Os canais são estabelecidos em estágios (ilustrado na figura 5.4.4):

1. primeiro, um objeto (E1) inicia a configuração de um canal interagindo com seu núcleo:
 - a interação é parametrizada pelo tipo do canal, um papel (define a qualidade do serviço) e uma interface de um objeto (E2) para ser unida ao canal;

- o núcleo une a interface de E2 com a interface de apresentação de um *stub* (S1), o *stub* com um *binder* (B1) e o *binder* com um *protocol* (P1);
 - o tipo do *stub*, *binder* e *protocol* são selecionados e determinados pelos parâmetros tipo do canal e papel;
 - o resultado da interação é uma referência de uma interface de engenharia (I) para comunicação com outros objetos e a união das interfaces de controle do *binder* (B1) e *stub* (S1) com o objeto (E1).
2. segundo, a referência de interface (I) é comunicada para outro objeto de engenharia (E3), possivelmente em outro *cluster*, cápsula ou nó;
 3. terceiro, o objeto (E3) interage com seu núcleo para ligar o canal:
 - a interação é parametrizada pelo tipo do canal, um papel e uma interface de um objeto (E4) para ser unida ao canal;
 - o núcleo une a interface de E4 com a interface de apresentação de um *stub* (S2), o *stub* com um *binder* (B2) e o *binder* com um *protocol* (P2) e, se for necessário, o *protocol* com um interceptador;
 - o tipo do *stub*, *binder*, *protocol* e se necessário um interceptador são selecionados e determinados pelos parâmetros tipo do canal e papel;
 - é unido o protocol (P1) com o protocol (P2) através de um interceptador se o canal o utilizar;
 - o núcleo une as interfaces de controle do *binder* (B2) e *stub* (S2) com o objeto (E3).

Regras de referências de interface

A referência de uma interface de engenharia identifica sem ambiguidades no espaço e no tempo as interfaces dos objetos de engenharia. Os objetos de engenharia podem ser realocados porém isto não invalida as referências de interface de engenharia desses objetos.

Os *binders* são responsáveis por manter o mapeamento das referências de interface de engenharia para identificadores de interface de comunicação quando os objetos são relocados, em cooperação com a função de relocação.

Uma referência de uma interface de engenharia contém dados que possibilitam a determinação de:

- um template adequado de um objeto canal para ser selecionado e instaciado, para a interface;
- a localização no espaço e no tempo de alguma interface de comunicação, para a interface.

Regras de cluster

Um *cluster* contém um conjunto de objetos básicos de engenharia, associados a um gerenciador de *cluster*. Cada membro de um *cluster* tem uma interface que dá suporte a

função de gerenciamento do objeto. Cada interface de gerenciamento de objeto deve ser associado ao gerenciador de *cluster*. Um *cluster* é sempre contido numa única cápsula.

O gerenciador de *cluster* é responsável por gerenciar a política para os objetos no *cluster*.

Um *cluster* é responsável por sua própria segurança, mas pode ser assistido por funções de segurança. Ele também é responsável pelo gerenciamento das referências de interface de engenharia para as interfaces dos objetos no *cluster*, mas pode ser assistido pela função de gerenciamento de referências de interface de engenharia.

Um objeto básico de engenharia num *cluster* é sempre unida a:

- seu núcleo, através de uma interface provida pela função de gerenciamento do nó;
- seu gerenciador de *cluster*.

Todas as outras interfaces de um objeto básico de engenharia são:

- ligadas a outro objeto de engenharia no mesmo *cluster*, ou
- ligadas a um canal.

O gerenciador de *cluster* é associado ao gerenciador de cápsula.

A figura 5.4.4 ilustra esta estrutura.

A instanciação de um *cluster* (incluindo a clonagem de um *cluster*) é desempenhada por um gerenciador de cápsula.

Um *template* de *cluster* (incluído *checkpoints* de um *cluster*) especifica a configuração dos objetos no *cluster* e qualquer atividade necessária para instanciá-los e estabelecer suas ligações. Se o *template* é um *checkpoint* de um *cluster*, a instanciação (isto é, clonagem) possibilita que o novo *cluster* aja como um substituto do *cluster* original da qual o *template* do *cluster* foi derivado. Neste caso todas as ligações que eram mantidas pelo *cluster* original devem ser reestabelecidas.

Regras de cápsula

A estrutura de uma cápsula é ilustrada na figura 5.4.4. Uma cápsula consiste de:

- *cluster(s)*;
- gerenciadores de *cluster*, um para cada *cluster* na cápsula;
- objetos *stubs*, *binders* e *protocols* para cada canal que é ligado a uma interface de um objeto básico de engenharia dentro de alguns dos *clusters*;
- um gerenciador de cápsula na qual é ligado cada gerenciador de *cluster* presente na cápsula.

O gerenciador de cápsula é responsável por gerenciar a política para os *clusters* na cápsula. Ele possui uma interface que provê a função de gerenciamento da cápsula.

A cápsula é sempre contida dentro de um único nó. A instanciação de uma cápsula é desempenhada pelo núcleo utilizando um *template* de uma cápsula que especifica a

configuração inicial dos objetos na cápsula. A estrutura de engenharia no sistema ODP deve prover mecanismos para a interação entre *clusters*, gerenciadores de cápsula e o núcleo.

Regras de nó

Um nó consiste de um núcleo e um conjunto de cápsulas. Todos os objetos num nó comportam as mesmas funções de processamento, armazenagem e comunicação. A estrutura de um nó está ilustrada na figura 5.4.4.

O núcleo incorpora a política de gerenciamento do nó. Ele provê uma interface de gerenciamento do nó separada para cada cápsula dentro do nó.

O procedimento de instanciação de um nó está fora do modelo ODP, mas deve resultar em:

- introdução de um núcleo ao nó e associado com ele as funções de processamento, armazenagem e comunicação, incluindo funções que possibilitam a ligação entre interfaces;
- introdução de alguma função de negociação necessária para o processo de instanciação;
- alguns canais necessários como parte da configuração inicial do nó (por exemplo, para objetos de suporte tais como um realocador);
- um conjunto de objetos *protocol* que determina o conjunto inicial de domínios de comunicação na qual o nó pertence.

Regras de falha

Uma falha pode ser localizada num *cluster*, numa cápsula ou num nó.

- uma falha localizada num *cluster* pode ser detectada pelo seu gerenciador de *cluster*;
- uma falha localizada num cápsula pode ser detectada pelo seu gerenciador de cápsula;
- uma falha de um nó pode ser detectada pelos objetos *protocol* dos outros nós da qual ele está interconectado.

5.5 Linguagem Tecnológica

A linguagem tecnológica mostra as possíveis soluções técnicas para os requerimentos da linguagem de engenharia, mostrando o custo e a disponibilidade de *hardware* e/ou *software* que o satisfaçam. O modelo ainda apresenta poucas regras aplicadas a linguagem tecnológica. Trabalhos sobre a linguagem tecnológica ainda estão em andamento.

No capítulo seguinte veremos a descrição da plataforma *multiware*, na qual este trabalho se insere.

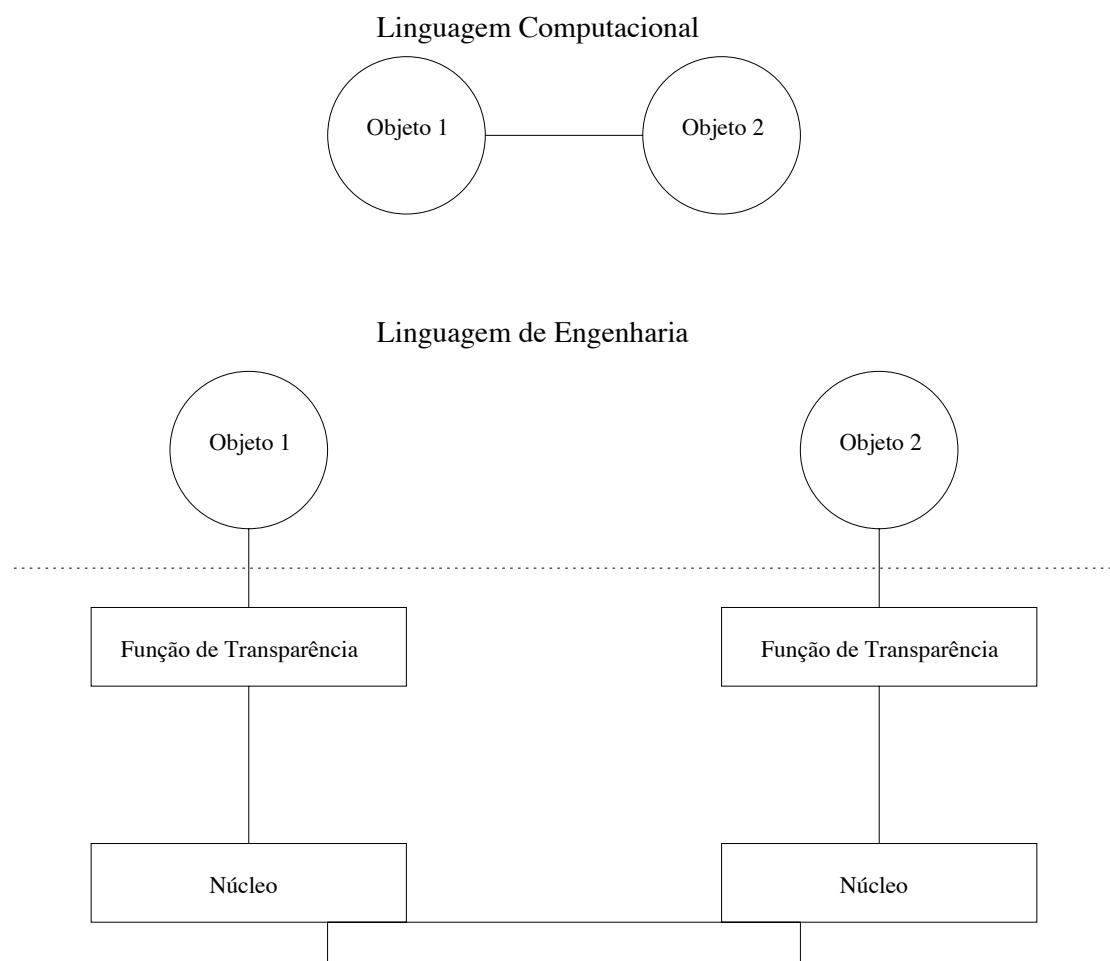


Figura 3: Visão computacional e de engenharia da interação de objetos

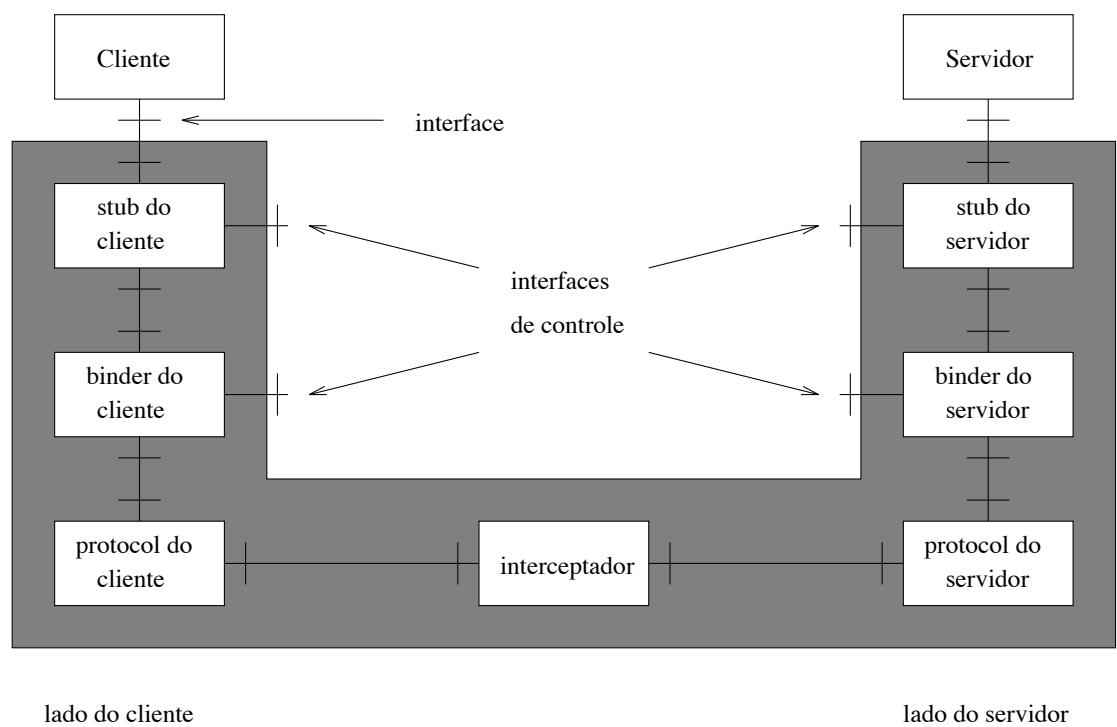


Figura 4: Exemplo de um simples canal cliente/servidor

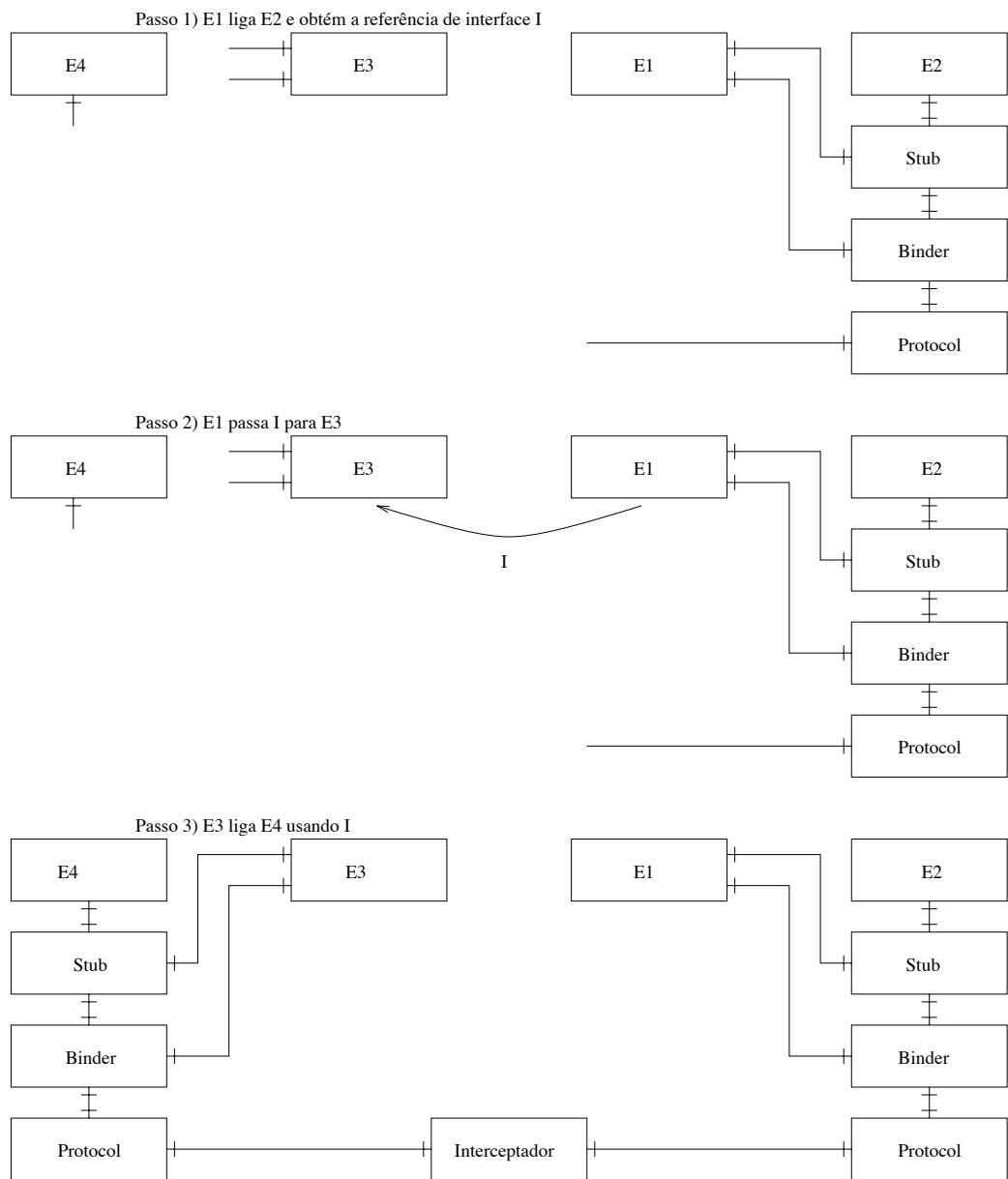
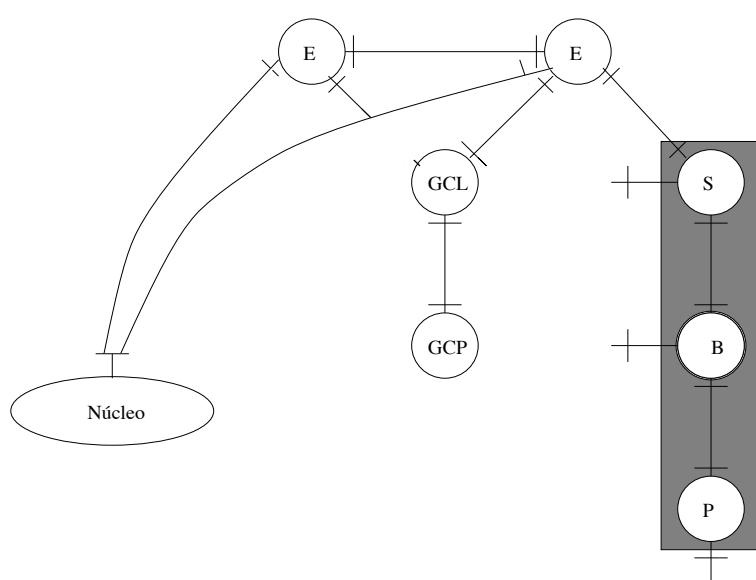
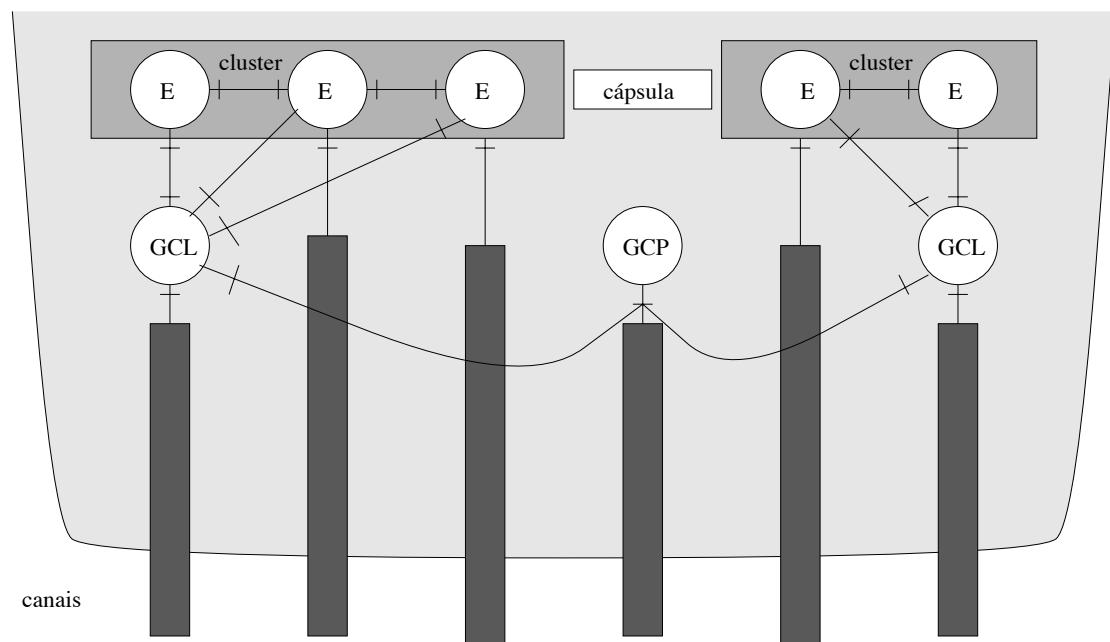


Figura 5: Ilustração das regras de estabelecimento de um canal



E: objeto básico de engenharia
 S: stub
 B: binder
 P: protocol
 GCL: gerenciador de cluster
 GCP: gerenciador de cápsula

Figura 6: Exemplo de uma estrutura que dá suporte a objetos básicos de engenharia



E: objeto básico de engenharia

GCL: gerenciador de cluster

GCP: gerenciador de cápsula

Figura 7: Exemplo de uma estrutura de uma cápsula

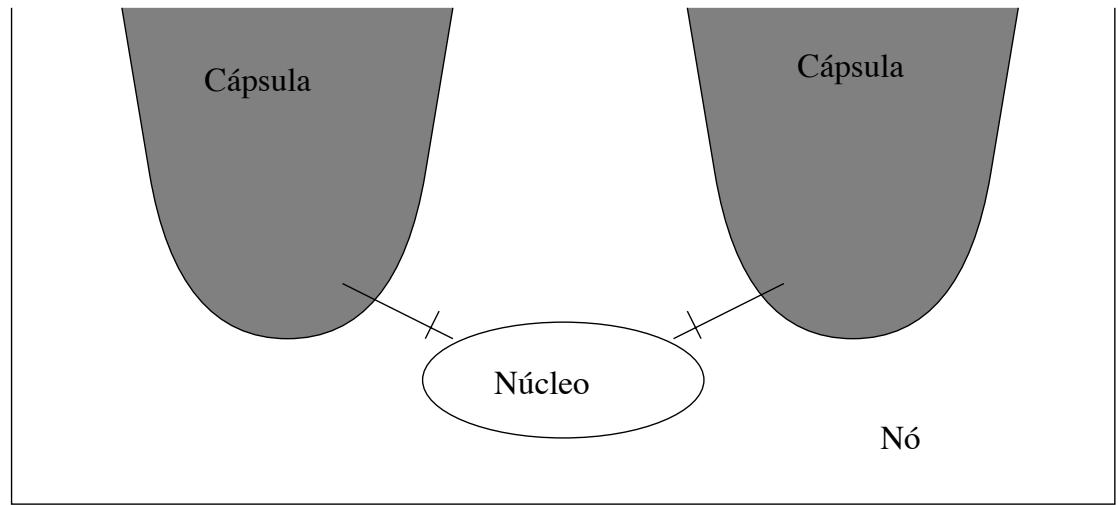


Figura 8: Exemplo de uma estrutura de um nó

Referências

- [1] Bowen, Dyfed *Open distributed processing Computer Networks and ISDN Systems* 23 (1991) 195-201 North-Holland
- [2] Campbell, Andrew; Coulson, Geoff; García, Francisco; Hutchison, David e Leopond, Helmut *Integrated Quality od Service for Multimedia Communications* IEEE INFO-COM 93, San Francisco, March 1993
- [3] Cockshot, W. P.; Atkinson, M. O. e Chisholm, K. J. *Persistent Object Management System* Software-Practice and Experience, Vol. 14, 49-71 (1984)
- [4] Davies, N.; Blair, G. S.; Friday, A.; Cross, A. D. e Raven, P. F. *Mobile Open Systems Technologies for the Utilities Industries* IEE Colloquium on CSCW Issues for Mobile and Remote Workers, London, U.K., 16th March 1993.
- [5] Domville, I. *The Distributed Application Environment - an Architecture Based on Enterprise Requirements* Elsevier Science Publishers B.V. (North-Holland) 1992 IFIP
- [6] Griethuysen, J. J. van *Enterprise modelling, a necessary basis for modern information systems* Elsevier Science Publishers B.V. (North-Holland) 1992 IFIP
- [7] Herbert, Andrew *The Challenge of ODP* Elsevier Science Publishers B.V. (North-Holland) 1992 IFIP
- [8] ISO/IEC JTC1/SC21/N7074 *Working Document on Topic 9.1 : ODP Trader* June 1992
- [9] ITU-T X.901 — ISO/IEC JTC1/SC21/WG7/N885 10746-1 *Basic Reference Model ODP - Part 1 : Overview and Guide to Use* Meeting of Torino (Italy), 3-11 November 1993
- [10] ITU-T X.902 — ISO/IEC DIS 10746-2 *Basic Reference Model ODP - Part 2 : Descriptive Model*
- [11] ITU-T X.903 — ISO/IEC DIS(E) 10746 *Basic Reference Model ODP - Part 2 : Prescriptive Model* 1994 Genebra Suiça
- [12] ITU-T X.904 — ISO/IEC SC21/N7056 10746-4 *Basic Reference Model ODP - Part 4 : Architectural Semantics*
- [13] Kilov, Haim *Precise specification of behaviour in object-oriented standardization activities* Computer Standards & Interfaces 15 (1993) 275-285 North-Holland
- [14] Kock, Frank *A first approach for an ODP - Description Language from the Computational Point of View (ODP-DLcomp)* Relatório Técnico GMD Fokus Berlin, Alemanha.

- [15] Kong, Qinzheng e Berry, Andrew *A General Resource Discovery System for Open Distributed Processing* Relatório Técnico do Projeto DSTC, Universidade de Queensland, Austrália, 4072
- [16] Kurose, Jim *Open Issues and Challenges in Providing Quality of Services Guarantees in High-Speed Networks* 3rd International Workshop on Network and Operating System Support for Digital Audio and Video, Nov. 12-13, 1992, San Diego, CA.
- [17] Lamport, Leslie *A Simple Approach to Specifying Concurrent Systems* Communication of the ACM / January 1989 Volume 32 Number 1
- [18] Lento, Luiz Otávio Botelho; Madeira, Edmundo Roberto Mauro *Um Esquema para Acessar Objetos em Ambientes Distribuídos* XIII Simpósio Brasileiro de Redes de Computadores, Abril de 1995
- [19] Leopold, Helmut; Coulson, Geoff; Ansah, Kwaku Frimpong; Hutchison, David e Singer, Nikolaus *The evolving relationship between OSI and ODP in the new communications environment* Relatório Técnico, ED.02-V1 / 3BY 00212 4001 UPZZA Departamento de Computação - Universidade de Lancaster May 7, 1993
- [20] Linington, P. F. *Open Distributed Processing* Elsevier Science Publishers B.V. (North-Holland) 1992 IFIP
- [21] Nicol, John R.; Wilkes, C. Thomas e Manola, Frank A. *Object Orientation in Heterogeneous Distributed Computing Systems* Computer June 1993
- [22] Raymond, K. A. *Reference Model of Open Distributed Processing: a Tutorial* Proceedings of the IFIP TC6/WG6.1 International Conference on Open Distributed Processing Berlim - Alemanha
- [23] Robinson, David *Remote Procedure Call: a stepping stone towards ODP* Computer Networks and ISDN Systems 23 (1991) 191-194 North-Holland
- [24] Rudkin, S. *Modelling object interfaces for open distributed processing* BT Technol Journal Vol. 10 No. 2 April 1992
- [25] Rudkin, S. *Templates, types and classes in open distributed processing* BT Technol Journal Vol. 11 No. 3 July 1993
- [26] Stocks, Phil; Raymond, Kerry; Carrington, David e Lister, Andrew *Modelling open distributed systems in Z* Computer Communications, Vol. 15, No. 2, March 1992
- [27] Taylor, Calvin J. *Object-oriented concepts for distributed systems* Computer Standards & Interfaces 15 (1993) 167-170 North-Holland
- [28] Tschammer, Volker; Mendes, Manuel J.; Souza, Wanderley L.; Madeira, Edmundo R. M. e Loyolla, Waldomiro P. *Processamento Distribuído Aberto e o Modelo RM-ODP / ISO XI* Simpósio Brasileiro de Redes de Computadores, UNICAMP, Campinas-SP, 10-13 de maio de 1993

- [29] Vários autores e artigos *Concurrent Object-Oriented Programming* Communication of the ACM / September 1993 Volume 36 Number 9