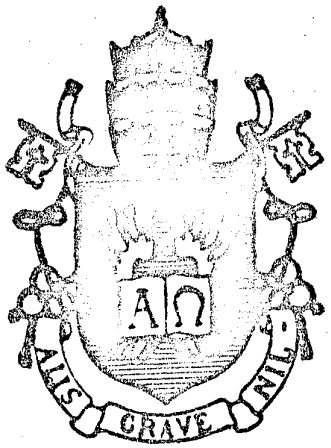



PUC

Série: Monografias em Ciência da Computação,
No. 8/91

# PROJETO ("DESIGN") ORIENTADO A ENCAPSULAMENTO DE DADOS E A TROCA DE MENSAGENS ENTRE SUBSISTEMAS AUTONOMOS

Maria Luiza A. Sanchez
Bruno Maffeo

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO
RUA MARQUÊS DE SÃO VICENTE, 225 - CEP-22453
RIO DE JANEIRO - BRASIL

PUC RIO - DEPARTAMENTO DE INFORMÁTICA

Série: Monografias em Ciência da Computação, No. 8/91
Editor: Carlos J. P. Lucena Maio, 1991

# PROJETO ("DESIGN") ORIENTADO A ENCAPSULAMENTO DE DADOS E A TROCA DE MENSAGENS ENTRE SUBSISTEMAS AUTONOMOS *

Maria Luiza A. Sanchez
Bruno Maffeo

* Apresentado por Bruno Maffeo.

Trabalho patrocinado pela Secretaria de Ciência e Tecnologia da Presidência da República.

**In charge of publications:**

Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC Rio - Departamento de Informática
Rua Marquês de São Vicente, 225 - Gávea
22453 - Rio de Janeiro, RJ
Brasil

Tel.:(021)529-9386 Telex:31078 Fax:(021)511-5645
E-mail:rosane@inf.puc-rio.br

## Resumo

É apresentado um modelo de projeto ("design") de software baseado em "Encapsulamento de Dados e Troca de Mensagens entre Subsistemas Autônomos". Este método, de característica "top-down", visa o domínio da complexidade de sistemas, particionando-os em subsistemas de menor porte, independentes uns dos outros, com características de reusabilidade.

Este modelo facilita a estratégia de Desenvolvimento Incremental, permitindo um menor tempo de resposta por parte das equipes de desenvolvimento, garantindo entrega de versões intermediárias operacionais, o que provoca melhor interação com o usuário / cliente e, em conseqüência, melhor adaptação do sistema a suas reais necessidades.



**Palavras Chaves :**

Modelo de Implementação / Encapsulamento de Dados / Reusabilidade de Subsistemas / Desenvolvimento Incremental / Objetos / Serviços / Portas

**Abstract**

This work presents a model of Software Design based on "Information Hiding and Exchange of Messages between Independent Subsystems". This model is top-down and permits the partitioning of a Complex System in more simple Subsystems. One or more of these Subsystems can be reused by other systems that have to perform similar functions. The Incremental Development Strategy is facilitaded, allowing a lower response time by the development team. This strategy permits the production of operational intermediate versions which may be employed by end users and consequently generates a better interaction between users/clients and developers. In general, the final version tends to better satisfy the end user.

**Keywords :**

Software Design / Information Hiding / Subsystems Reusability / Incremental Development / Objects / Services / Port

Sumário

# 1 - Introdução

Entre a definição abstrata de um sistema sócio-técnico e sua implantação existe uma lacuna muito grande, tanto no nível de abstração e de complexidade, quanto no emprego de tempo e de recursos (materiais e humanos).

O preechimento dessa lacuna constitui, normalmente, uma das etapas mais demoradas do processo de desenvolvimento do sistema, e quanto maior o sistema, maiores serão os problemas nele encontrados. Vários esforços foram feitos no sentido de diminuir esta lacuna, conforme pode ser encontrado em [1 - 11]. Sistemas Sócio-Técnicos de grande porte [12,13] envolvem, na fase de projeto ("design"), um número muito grande de pessoas e recursos. Esses sistemas têm, portanto, custos elevados e alto grau de risco, isto é, implicam investimento grande no desenvolvimento e incerteza quanto ao sucesso de sua implantação.

É importante, portanto, que o processo de desenvolvimento dessa classe de sistemas seja encarado sob nova ótica, que ataque os pontos que são as principais fontes de dificuldades. Entre os objetivos a atingir, decorrentes da supressão dessas dificuldades, podemos citar:

- Garantia de Qualidade Assegurada por Construção
- Redução de Custos (cumprimento de orçamentos planejados)
- Redução de Tempo Gasto na Implementação (cumprimento de cronogramas planejados)

- Otimização de Gerência das Equipes Envolvidas
- Redução do Impacto Devido a Mudanças nas Equipes de Desenvolvimento

A motivação deste trabalho origina-se na percepção desse tipo de questões e seu conteúdo propõe conceitos e técnicas que visam facilitar o processo de desenvolvimento de Sistemas Sócio-Técnicos, em especial sistemas de tempo-real, de grande porte, em ambiente distribuído.

Esses últimos tipos de sistema (tempo-real, grande porte, distribuído) apresentam agravantes quando comparados com sistemas comerciais convencionais, geralmente relacionadas com baixa tolerância referente a requisitos de desempenho e a falhas. Convém acentuar que o domínio adequado dessas características é muito difícil de prever nas fases iniciais do processo de desenvolvimento.

Procuraremos, então, apresentar um método que favoreça a abordagem desses problemas, estabelecendo os requisitos a que o projeto ("design") do sistema deva satisfazer, para que possamos minimizar, nas fases de codificação, depuração e implementação, os problemas mencionados anteriormente.

Basicamente, cinco características de qualidade norteiam o método :

- MODULARIDADE : visando acrescentar novas funções, modificar ou substituir algumas das funções antigas, sem influências relevantes nas demais funções do sistema.

- MANUTENIBILIDADE : a concepção do sistema, bem como a documentação pertinente, deve procurar simplicidade e clareza, facilitando o entendimento por pessoas que não estejam participando do processo de desenvolvimento.

- PORTATILIDADE : o sistema deve ser projetado o mais abstraído possível de qualquer Arquitetura de Hardware bem como do Software Básico associado, de forma a facilitar a migração para outras Arquiteturas de Hardware.

- PREVISIBILIDADE : visando diminuir o risco nas avaliações preliminares de capacidade de processamento da Arquitetura de Hardware/Software Básico escolhida para obedecer os requisitos de desempenho.

- REUSABILIDADE : esse é o ponto mais importante na Redução de Custos e Tempo de Desenvolvimento. Se o projeto ("design") do sistema gerar um conjunto de módulos (software + hardware) que possam ser usados em outro sistema, com o passar do tempo teremos um diminuição progressiva da fase de implementação.

  No entanto, para que esses módulos sejam efetivamente reusáveis, o método de

desenvolvimento tem que estar integrado a uma política de administração do processo de desenvolvimento, bem como a um controle dos módulos já existentes.

Tendo em vista essas características de qualidade, podemos ver o sistema como um conjunto de subsistemas, com funções bastante independentes e, conseqüentemente, com baixo acoplamento. Essa característica facilita a depuração do sistema, já que erros possuirão fontes mais bem definidas. Facilita, também, a incorporação de novas funções ao sistema e a concepção de um documentação modular, em diversos níveis de abstração, visando o entendimento do sistema como um todo ou, mesmo, de partes dele.

Muitos desses subsistemas são identificados em estágios iniciais do processo de desenvolvimento, até durante a fase de construção do Modelo da Essência [14,15], pois devem corresponder a OBJETOS do mundo real.

Subsistemas independentes permitem implementações em paralelo, o que abrevia a apresentação de resultados ao cliente/usuário. Também, facilitam o uso de uma política de desenvolvimento incremental, o que permite entregas parciais do sistema com acréscimo gradativo de funções -Estratégia de Desenvolvimento Incremental [12,16]. Essa estratégia permite uma avaliação parcial do desempenho e aceitação do sistema por parte do cliente/usuário, minimizando os problemas devidos à expectativa e insatisfação relacionadas com a demora do processo de desenvolvimento.

## 2 - Conceitos

Nesta seção, definiremos os conceitos sobre os quais fundamentamos o método e que nos orientam na busca da melhor estrutura para o sistema.

### 2.1 - Objetos

Neste trabalho, o termo OBJETO será usado com o sentido de uma instância de um "Tipo Abstrato de Dado".

Na realidade, trabalhamos com o conceito de "Encapsulamento de Dados" da Engenharia de Software, primordialmente criado para permitir modularidade acoplada a desenvolvimento autônomo de módulos ( "Information Hiding" ) [12].

O Encapsulamento de Dados envolve o conjunto Estrutura de Dados e suas Rotinas de Acesso, isto é, a estrutura de dados é apenas acessivel através dos serviços providos pelas rotinas de acesso. Essa estrutura é, então, definida pelas operações que podem ser executadas sobre ela. Dessa forma, o "Encapsulamento de Dados" provê o OBJETO abstrato de dado [12].

Um Objeto é definido por :

- Um Nome que o identifica univocamente
- Uma Representação que consiste na definição dos dados que ele representa.
- Um conjunto de Operações que definem as funções que podem ser executadas sobre a estrutura de dados.

Os objetos podem ter características :

REATIVAS : tem operações ativadas por outros objetos

ATIVAS : tem existência própria e executa operações independententemente de outros objetos.

Objetos, tal como definidos neste trabalho, permitem uma excelente forma de estruturar sistemas porque encapsulam os conceitos relativos aos procedimentos e aos dados sobre os quais atuam, são unidades autônomas de programação e de distribuição e são as entidades entre as quais a comunicação acontece.

## 2.2 - Sistemas X Serviços

Quando estamos diante de um sistema complexo é importante que usemos uma ferramenta de projeto de forma a dominar a complexidade relacionada com o porte do problema, visando facilitar sua implementação.

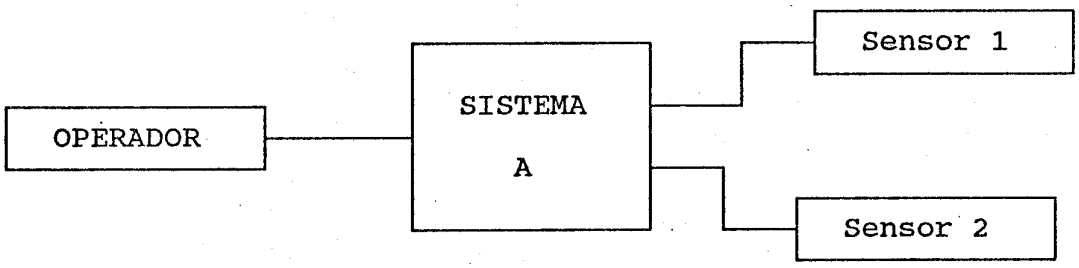
Um sistema pode, em geral, ser segmentado em subsistemas e cada subsistema, por sua vez, pode, eventualmente, ser considerado um sistema. Observa-se, portanto, que o conceito tem uma recursividade intrínseca.

É importante, também, a visão que temos do sistema. Este pode ser visto como um conjunto de SERVIÇOS. Um serviço nada mais é do que uma operação específica que atua sobre uma estrutura de dados do sistema . O conjunto de serviços que atuam sobre uma dada estrutura constitui um OBJETO do sistema.

Serviço --> Operação sobre uma Estrutura de Dados

Objeto --> Estrutura de Dados + Serviços

A decomposição do sistema, na visão de serviços, será feita em função destes. Por exemplo, um sistema A (Software e Hardware) que processa dados provenientes de dois sensores e os apresenta a um operador, teria a seguinte figura de contexto:




A idéia de serviços facilita a análise do sistema e chega-se mais naturalmente aos módulos que o compõem.

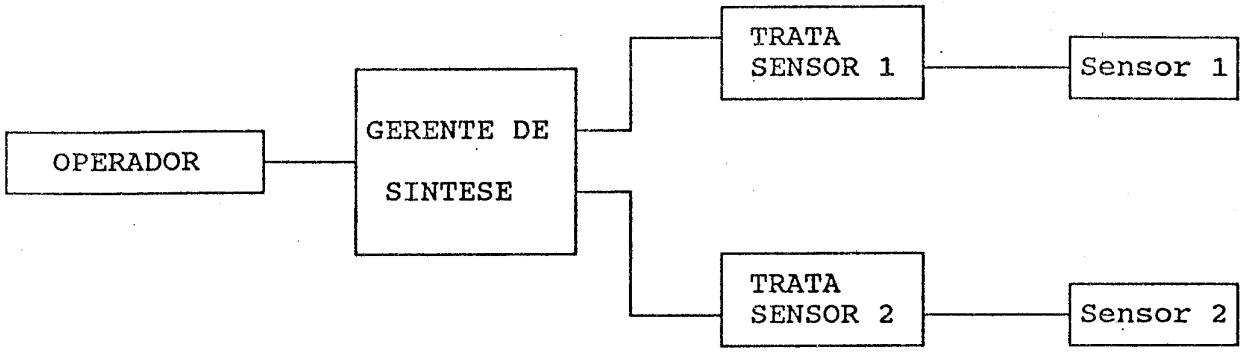
Os serviços do Sistema A são portanto:

1 - Processar informações do sensor 1

2 - Processar informações do sensor 2

3 - Sintetizar e apresentar as informações processadas ao operador.

Teríamos, portanto, de forma bastante natural a seguinte decomposição modular do sistema:




Em um primeiro estágio, a segmentação de um sistema pode apresentar, ainda, subsistemas complexos.

O processo de segmentação deve ser repetido recursivamente, gerando novos subsistemas, até que cada um dos subsistemas resultantes possa ser considerado como uma "unidade de implementação" - SUBSISTEMA BASICO, cujas características serão descritas a seguir.

É importante notar que :

- Os diversos níveis de decomposição que contêm subsistemas não devem apresentar estruturas de dados conectando os subsistemas. Os dados são parte integrante dos subsitemas e o acesso a um dado é feito através dos serviços oferecidos pelos subsistemas.
- Um Subsistema Básico tem como característica importante o fato de oferecer um conjunto coeso de serviços atuando sobre uma estrutura de dados específica. O nível onde aparecem os Subsistemas Básicos é o último nível onde ainda conseguimos manter os dados encapsulados. Deve ser observado que um Subsistema Básico, tendo em vista a definição apresentada na seção 2.1, é um Objeto. Os serviços que compõem um

subsistema básico podem ser:

- Serviços Externos : são serviços executados a partir de um pedido externo de outro subsistema básico.
- Serviços Internos : são serviços executados independentemente de pedidos, iniciados a partir de condições limites ou do decorrer de um período de tempo.

Um subsistema pertencente a um dado subsistema mais abrangente, para completar a execução de seu procedimento, normalmente, precisará pedir serviços a outros subsistemas do mesmo subsistema mais abrangente ou de outro subsistema qualquer. Esse tipo de serviços é aqui denominado como SERVIÇO REQUISITADO.

Muitos dos subsistemas básicos são identificados em etapas iniciais da análise do sistema, por ser bastante nítida a existência daquela função independente das outras funções do sistema (por exemplo, o tratamento dos dados provenientes de um sensor).

Esses subsistemas serão apresentados numa representação "top-down", visando permitir rápida visualização de todo o projeto.

Os Subsistemas Básicos são UNIDADES DE IMPLEMENTAÇÃO que se prestam ao manuseio, projeto ("design"), produção e manutenção de grandes sistemas. São também unidades de encapsulamento de dados, o que implica a não existência de áreas de dados compartilhadas entre subsistemas básicos.

Essa visão de Engenharia de Software permite a divisão

de trabalho em grandes sistemas, além de facilitar o planejamento, a gerência do projeto de desenvolvimento e o processo de integração do sistema.

## 2.3 - Interfaces

Cada sistema comunica-se com suas entidades externas através de uma INTERFACE por onde fluem controle e dados. Por exemplo, quando um operador faz incidir um fluxo de dados no sistema, dois tipos de informação ocorrem simultaneamente:

- a ordem de executar um comando (conteúdo de controle do fluxo de dados)
- os dados de que o sistema necessita para executar e que não existem internamente ao sistema.

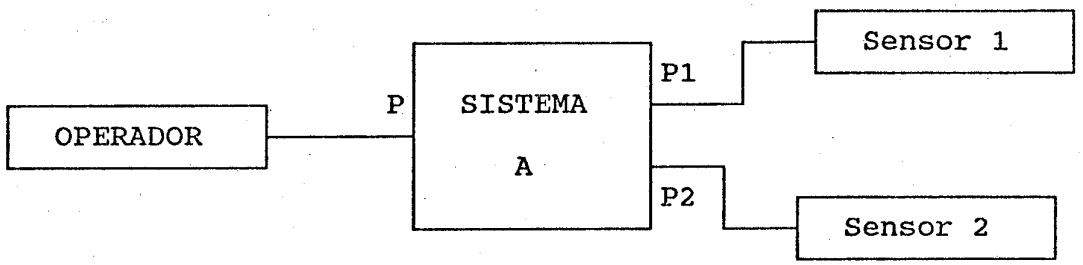
Estendendo esse conceito aos subsistemas e utilizando a recursividade do conceito de sistema, obtemos: os subsistemas se interligam também através de INTERFACES ou, mais restritamente, os subsistemas básicos se interligam através de INTERFACES.

## 2.4 - Portas

Uma das premissas iniciais do método é manter a independência entre subsistemas e, para tal, é fundamental a introdução do conceito de PORTA.

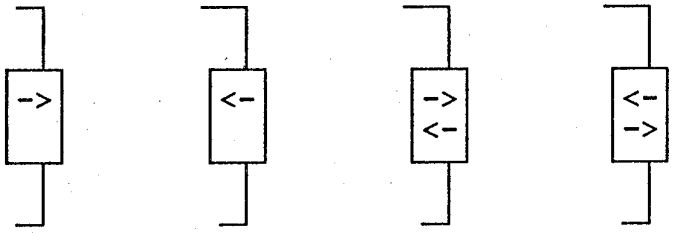
O sistema, bem como os subsistemas que o compõem, devem ter apenas conhecimento de si próprio, isto é, um subsistema não sabe da existência do outro. Nessas condições, cada subsistema deve possuir PORTAS de entrada e saída de dados

que definam sua comunicação com o mundo exterior. Toda definição, projeto ("design") e automação do subsistema deve fazer referência apenas a suas portas e nunca a outros subsistemas com os quais se comunica. Desta forma fica mais fácil o cumprimento do requisito de Reusabilidade.




onde P,P1 e P2: Portas de entrada e/ou saída

As portas podem ser representadas através de figuras mais precisas, que indiquem se estas são de entrada, saída e se têm resposta fluindo no outro sentido. Por exemplo :




Através dessas portas são efetivamente ativados os serviços que o sistema/subsistema presta ou requisita ao mundo exterior.

Para que tenhamos uma definição precisa de um subsistema necessitamos apenas de:

- a(s) entrada(s);

- a(s) saída(s);
- como, a partir da(s) entrada(s), pode-se produzir a(s) saída(s).

Notar que, na definição acima, "como" quer significar uma relação, no sentido matemático do termo, e não exige a especificação de um procedimento.

Todo o projeto interno do subsistema só interage, na realidade, com essas portas. Cada porta tem, intrinsecamente associado, um tipo de variável que a descreve, isto é, uma estrutura, descrita a partir de dados atômicos.

**Porta de Entrada** : é a que canaliza o estímulo para um serviço externo oferecido pelo subsistema

**Porta de Saída** : dessa porta parte um pedido de serviço requisitado a outro subsistema.

É importante salientar que, a qualquer serviço, pode haver resposta associada, isto é, se em uma porta de entrada chegar um pedido de serviço, através dela pode sair uma informação associada à resposta produzida pelo serviço.

As INTERFACES, são, portanto, os instrumentos que conectam PORTAS de SUBSISTEMAS; mais especificamente, conectam PORTAS de SUBSISTEMAS BÁSICOS. Convém observar que duas portas quando conectadas, devem sempre ser do mesmo tipo. Isto é, a porta a, que solicita um serviço cujo pedido a porta b espera receber, deve realizar o pedido utilizando o mesmo padrão ( tipo físico) de dados que caracteriza o fluxo

de informação nesta interface ( conexão entre **a** e **b**). Desta forma, o sincronismo entre subsistemas fica completamente resolvido pela conexão de suas portas de entrada e saída.

## 2.5 - Configuração

Esta é a característica que impõe ao sistema sua Arquitetura de Hardware.

O projeto ("design") do sistema será feito levando-se em conta que este é um conjunto de subsistemas básicos interconectados por Interfaces.

Esses subsistemas básicos são UNIDADES DE IMPLEMENTAÇÃO individuais para projeto ("design"), automação e teste, ou melhor, UNIDADES DE DESENVOLVIMENTO E TESTE.

Mas, subsistemas prontos e testados precisam ser integrados, formando o sistema.

A configuração de um sistema concebido dessa forma é feita através de uma **linguagem de configuração** com a função de definir as Interfaces dos subsistemas, verificar a consistência das portas conectadas e mapear os processos de um dado subsistema básico na Arquitetura de Hardware.

Esse procedimento permite, com certa facilidade, integração gradativa do sistema, viabilizando entrega parcial ao cliente/usuário -Estratégia de Desenvolvimento Incremental- e, portanto, a solução de problemas que ocorrem no desenvolvimento de sistemas de grande porte como, conforme já mencionado, a insatisfação do cliente quanto ao tempo gasto no desenvolvimento.

Podem ser citadas as seguintes vantagens de uma

Estratégia de Desenvolvimento Incremental:

- Adaptar gradativamente o usuário ao manuseio do sistema;
- Revisão corretiva do planejamento de capacidade do hardware escolhido;
- Apresentar mais cedo resultados operacionais parciais, isto é, o sistema vai sendo entregue através de uma série de versões operando com funcionalidade crescente em direção à funcionalidade total. Esse aspecto é de importância relevante no caso de sistemas de grande porte já que estes tendem a demorar mais tempo na fase de implementação, envolvem custos altos e têm sempre, em conseqüência, maior inquietação do contratante sobre a efetiva capacidade de execução pela equipe de desenvolvimento.

O Sistema é, então, composto por : Subsistemas Básicos, Portas e Interfaces. Os subsistemas básicos possuem um ou mais processos que executam concorrentemente em um ou mais processadores.

Na **Configuração do Sistema**, temos que mapear os processos de cada subsistema básico, suas interfaces e sua execução na Arquitetura de Hardware.

A **Configuração do Sistema** é, assim, constituída de duas etapas:

a) **Configuração do Software**

Nesta etapa são definidas todas as portas e suas conexões, isto é, que Porta de Saída de um subsistema básico está conectada a qual Porta de Entrada de outro subsistema básico. Essas conexões possuem as seguintes formas:

a - Um para um

b - Um para muitos

c - Muitos para um

Nos tipos de conexão a e c, para uma porta de saída, podemos ter resposta associada, o que não ocorre no caso b que funciona como um Broadcast Seletivo. Na definição das conexões entre as portas deve ser verificada a consistência entre os dados que fluem na ligação, isto é, portas, quando conectadas, devem possuir o mesmo tipo de variável.

**b) Configuração de Arquitetura**

Nesta etapa é definida a integração da Arquitetura de Software na Arquitetura de Hardware.

Define-se quais processos pertencem a que processadores. Após uma avaliação das necessidades do sistema, da quantidade de processamento e da resposta requerida, é definida pela equipe de projeto uma Arquitetura de Hardware.

O mapeamento dos subsistemas básicos na Arquitetura de Hardware define o número de encarnações ( número de processos independentes, reentrantes, que compartilham o mesmo código) de cada processo de cada subsistema básico e onde cada uma dessas encarnações vai residir, bem como a prioridade relativa entre os processos. Isso dá origem à Arquitetura de Software.

Deve ser ressaltado que uma reavaliação desta definição é feita durante todo o processo de desenvolvimento.

É importante salientar que todo o processo de

desenvolvimento, pelo fato de nos referirmos a subsistemas básicos, possui uma independência muito grande em relação ao software básico disponível, já que toda a comunicação entre processos ocorrerá através de primitivas básicas - ENVIA / RECEBE - para trânsito de mensagens. Essas primitivas, caso não estejam disponíveis no ambiente em questão, deverão ser implementadas através do uso de semáforos, para que seja possível respeitar a filosofia imposta pelo método. Essas primitivas estarão mais bem detalhadas no item 4.

## 3 - O Processo de Desenvolvimento

Procuraremos, agora, descrever as etapas do desenvolvimento do sistema, sob o enfoque dos conceitos apresentados na seção 2.

Resumindo, temos :

- Um sistema é um conjunto de Subsistemas Básicos;
- Cada subsistema básico oferece e solicita serviços através de suas portas de entrada e saída;
- Estas Portas são conectadas a portas de outros subsistemas básicos através de Interfaces;
- Interfaces são definidas utilizando-se uma Linguagem de Configuração de sistema;
- Os Subsistemas Básicos e seus processos são mapeados no Hardware através de uma Linguagem de Configuração de Hardware;

Portanto, as etapas envolvidas no processo devem ser tais que:

- Permitam a identificação dos Subsistemas Básicos, suas Portas e Interfaces, bem como o tipo de concorrência e sincronismo entre serviços alocados a cada Subsistema Básico - **Projeto Básico;**
- Permitam a definição precisa dos Subsistemas Básicos como unidades independentes de desenvolvimento - **Especificação de Subsistemas Básicos**
- Permitam o projeto e implementação dos Subsistemas Básicos como unidades independentes de desenvolvimento e teste - **Projeto ("Design") de um Subsistema Básico**
- Integrem os diversos Subsistemas Básicos dando origem ao sistema que se deseja implantar - **Configuração do Sistema**

Descreveremos, a seguir, cada uma dessas etapas e os documentos a ser gerados durante o processo de desenvolvimento.

## 3.1 - O Modelo da Implementação

Essa é a etapa de identificação dos subsistemas básicos. Conforme já mencionado anteriormente, a segmentação que conduz aos subsistemas básicos é feita a partir dos serviços do sistema: alocando esses serviços a subsistemas, decompondo serviços complexos em serviços mais simples, realocando estes serviços mais simples a novos subsistemas. Deve ser observado que, um certo serviço só aparece em um subsistema. Caso algum outro necessite deste serviço deve pedí-lo ao subsistema ao qual está alocado.

Será necessária, para essa representação, uma

linguagem diagramática hierárquica, apoiada por uma representação de controle, também hierárquica, que represente a estrutura e a concorrência entre subsistemas.

É importante lembrar que, nesse processo de segmentação em subsistemas, o encapsulamento dos dados deve ser mantido, não existindo portanto, em nenhum nível, o aparecimento de dados compartilhados entre subsistemas básicos.

A comunicação e o sincronismo entre os subsistemas são realizados apenas através do trânsito de mensagens nas Interfaces.

Este processo de segmentação do sistema em subsistemas se repete até encontrarmos cada subsistema básico, com características descritas a seguir:

- um subsistema básico tem a ele alocado um número reduzido de serviços, logicamente reunidos segundo uma finalidade analítica específica e, portanto, de rápido entendimento;
- ele é uma unidade de desenvolvimento e teste, isto é, pode ser realizado por uma equipe pequena, reduzindo os problemas de relacionamento entre os executores da tarefa;
- tem interconexão bem definida com outros subsistemas: Interfaces e Portas;
- preferencialmente, deve ser reusável em outros sistemas.

Cabe ressaltar que aqui é apresentada uma filosofia de particionamento do sistema e das etapas do processo de desenvolvimento. No entanto, não é possível a determinação de um método que chegue à melhor solução ou a uma solução única.

A solução de implementação é obra do projetista, mas os critérios descritos acima e a busca da solução mais simples possível devem nortear o trabalho. Ainda, critérios de avaliação da solução como baixo acoplamento e coesão funcional devem também ser usados como medida da qualidade do trabalho realizado.

Normalmente, os serviços de um subsistema básico são concorrentes sendo os subsistema básicos equivalente a um ou mais Processos de Automação.

No projeto básico chegamos ao conjunto de serviços pertencente a cada Subsistema Básico de uma forma ainda abstrata e sucinta. Entretanto, para que a implementação seja feita por equipes diferentes, cada Subsistema Básico necessita de uma especificação detalhada independente. Isto é, o serviço mencionado no Projeto Básico não estará descrito com o nível de detalhe necessário à implementação. Deve, então, ser delineado o conjunto de serviços de um Subsistema Básico e, a partir disso e do Modelo da Essência do sistema, será feita uma especificação detalhada do Subsistema Básico. Deve ser assegurado o cumprimento das necessidades do usuário e a comunicação via as Portas determinadas no projeto básico. Outro ponto relevante é o da Reusabilidade do Subsistema Básico, de forma a identificar nesse estágio o que deve ser parametrizado para que possa vir a ser usado em outros sistemas.

No entanto, é ainda necessário caracterizar o papel assumido pelos Subsistemas Básicos, de forma a facilitar a

tarefa de particionamento.

Os Subsistemas Básicos devem ser definidos a partir de áreas de responsabilidade de uma particular solução para o sistema global e suas conexões representam como estes Subsistemas Básicos podem trabalhar juntos para implementar essa solução.

A maior parte dos Subsistemas Básicos fazem parte de uma das seguintes categorias [11]:

- **Responsáveis por objetos físicos** (ex.: dispositivos de hardware): normalmente, a um dispositivo de hardware é acoplado um Subsistema Básico, que o controla e o apresenta ao sistema com um conjunto de operações necessárias a seu uso (Dispositivo Inteligente).
- **Responsáveis por estrutura de dados** : visam manter a filososofia de encapsulamento de dados, bem como obter homogeneidade na segmentação, já que um dispositivo de hardware é, efetivamente, uma estrutura de dados encapsulada em uma peça de hardware.
- **Responsáveis por funções do sistema:** cada Subsistema Básico representa uma parte da solução do problema. É importante notar que, como alguns aspectos que caracterizam a solução do problema não são reusáveis, estes aspectos devem ser separados em outro Subsistema Básico respeitando a filosofia da Reusabilidade. Podemos citar como exemplo um sistema com apresentação de seus resultados em vários dispositivos. Os Subsistemas responsáveis pela determinação dos resultados e os responsáveis pela apresentação nos dispositivos são reusáveis. No entanto, deve ser separado

em um subsistema independente o módulo responsável pela identificação do dispositivo onde deve ser apresentado um resultado específico produzido pelo sistema.

- **Responsáveis por relacionamento de funções:** estes são os subsistemas básicos que coordenam a execução, na realidade, a força motriz do sistema.

## 3.2 - Projeto Básico

### 3.2.1 - Ferramentas de Modelagem do Projeto Básico

Outro aspecto de importância para a qualidade do sistema é a qualidade de sua documentação. Vamos descrever aqui as ferramentas usadas na modelagem e a proposta de documentação do projeto básico, com o intuito de que esta seja completa, de fácil entendimento e sua produção não seja motivo de impacto para o desenvolvimento.

Essencialmente, a documentação deve retratar as características de pensamento do projetista e os caminhos que o levaram a uma solução.

A documentação do Projeto Básico deve conter:

- O conjunto de diagramas hierárquicos que representam a estrutura - através de Diagramas de Estrutura (DE) - e o comportamento - atavés de Diagrama de Comportamento (DC) - do sistema e de cada subsistema;
- Para o sistema e cada subsistema, devem ser relacionados os serviços executados pelo subsistema em questão;
- Deve ser elaborado um Dicionário de Dados local de todas as Interfaces e Entidades Externas que aparecem nos diversos

Diagramas de Estrutura;

- Um esquema único, explicitando as Interfaces, que represente a totalidade dos Subsistemas Básicos em um único Diagrama de Estrutura - **Diagrama dos Subsistemas Básicos** - facilitando assim a visualização da Configuração do Sistema;
- Um diagrama que apresente a árvore de decomposição gerada, resumindo a decomposição de todo o sistema nos diversos níveis - **Árvore de Subsistemas;**
- Uma tabela que valide os Subsistemas e serviços associados a cada Subsistema em relação ao Modelo da Essência.

Na Configuração do Sistema, é feito o mapeamento dos subsistemas, pelo menos de forma preliminar, na Arquitetura de Hardware. Esse mapeamento é preliminar pois, na fase de automação dos Subsistemas Básicos, será feita a reavaliação dessa distribuição. Essa reavaliação é necessária tendo em vista a otimização do desempenho do sistema, já que um Subsistema Básico pode possuir vários serviços concorrentes.

O Projeto Básico é a etapa de identificação dos Subsistemas Básicos. É realizada uma segmentação que conduz aos Subsistemas Básicos feita a partir dos serviços do sistema: esses serviços são alocados a subsistemas, sistematicamente decompondo serviços complexos em serviços mais simples, os quais são realocados a novos subsistemas.

Lembramos que um dado serviço só aparece em um subsistema. Caso algum outro subsistema necessite desse serviço deve pedí-lo ao subsistema ao qual o mesmo esteja

alocado.

A linguagem de representação diagramática hierárquica necessária ao Projeto Básico, será descrita em detalhe (sintático e semântico) nas seções 3.3 e 3.4, e é constituida básicamente dos Diagramas de Estrutura e dos Diagramas de Comportamento.

### 3.2.2 - Organização do Projeto Básico

O Projeto Básico deve ser organizado da seguinte forma:

- **ELEMENTOS DE MODELAGEM DE SUBSISTEMAS BASICOS**
  - **Contexto**
    - Diagrama de Estrutura, representando o Contexto do Sistema;
    - Descrição Textual sucinta, apresentando os diversos serviços que compõem o sistema
  - **Decomposição do Sistema**
    - Apresentar na seguinte ordem, para cada nível de decomposição :
      - Diagrama de Estrutura
      - Diagrama de Comportamento
      - Descrição dos Serviços de cada Subsistema presente no Diagrama de Estrutura

  - **Diagrama dos Subsistemas Básicos**
  - **Árvore de Subsistemas**
  - **Dicionário de Dados**
  - **Tabela de Verificação de Consistência com o Modelo da**

### 3.2.3 - Descrição dos Serviços

Lista numerada que relaciona de forma compacta um serviço realizado pelo sistema. Esse serviço já está detalhado, como funcionalidade do sistema, no Modelo da Essência.

### 3.2.4 - Diagrama dos Subsistemas Básicos

Com a mesma sintaxe e semântica usada nos Diagramas de Estrutura, esse diagrama apresenta apenas os Subsistemas Básicos e a forma como interagem. Apresenta também todos os dados no seu nível mais baixo de decomposição nesse escopo.

### 3.2.5 - Arvore de Subsistemas

A Arvore de Subsistemas proporciona uma visão sucinta da estrutura hierárquica selecionada, facilitando a identificação da decomposição de um dado serviço e auxiliando as atividades manutenção.

Nessa árvore os Subsistemas Básicos são as folhas.

### 3.2.6 - Dicionário de Dados

Conjunto de dados presentes nos diagramas de estrutura e comportamento. Deve ser organizado em ordem alfabética e cada entrada no dicionário deve apresentar :

- Nome;
- Definição : a função do dado na estruturação do sistema;
- Composição : em termos da BNF, descreve, caso

pertinente, de que dados é composto a entrada do dicionário em questão;

- Tipo : no caso de elemento de dado, descrevendo seu domínio (limites inferior ou superior ou conjunto de valores), unidade, precisão e formato, quando pertinentes.

### 3.2.7 - Tabela de Verificação de Consistência

É apresentado o conjunto de relacionamentos entre os eventos evidenciados no Modelo da Essência e os Serviços e Subsistemas Básicos que aparecem no Projeto Básico.

## 3.3 - Diagramas de Estrutura - Sintaxe e Semântica

### 3.3.1 - Introdução

Esta seção aborda o Diagrama de Estrutura (DE), explicitando sua função e apresentando sua sintaxe, semântica e forma de leitura.

Os Diagramas de Estrutura referem-se ao particionamento dos serviços prestados por um sistema em subsistemas que o compõem.

O que temos, em princípio, como modelagem inicial de um sistema, é seu Diagrama de Contexto, isto é, a representação de um conjunto de portas através das quais o sistema comunica-se com o mundo exterior e fluxos de dados de entrada e saída. Para definir completamente o sistema, é necessário associá-lo a um conjunto de serviços que presta a ou requisita de entidades externas.

Um DE modela a forma de decomposição do Sistema em um conjunto de Subsistemas, cada qual com uma parte dos serviços prestado ou requisitado pelo Sistema. Tanto portas, quanto fluxos de entrada e saida associados ao sistema são também usados por esses Subsistemas, restando acrescentar portas e fluxos de dados necessários ao relacionamento entre os Subsistemas que compõem o DE.

Essa decomposição é uma técnica de modelagem que deve, ao ser empregada, respeitar as Leis de Orçamento para Complexidade e da Essência Mínima [14,18].

Essa decomposição tem uma característica descendente ("top-down"), apesar de não coibir o uso de outras técnicas, tais como a estratégia "de fora para dentro" ("outside-in") preconizada em [14,15,18] como estratégia de modelagem de sistems sócio-técnicos.

### 3.3.2 - O que queremos representar?

Um sistema computacional é composto de três conjuntos disjuntos : Operações, Dados e Controle.

Em um DE, duas destas grandezas aparecem de forma clara: Operação e Dados. Conseguimos, assim, modelar que Operações são executadas pelo sistema através de seus Subsistemas, e que Dados são necessários à execução de tarefas do sistema e são produzidos por seus Subsistemas.

O conjunto de DE's gerados para um dado sistema, usando-se a recursividade do conceito de sistema [17,18], constituem uma estrutura hierárquica, "top-down",

apresentando a decomposição do sistema até o nível de Subsistemas Básicos.

O DE também é usado na decomposição de um dado Subsistema Básico, no Modelo da Configuração de Módulos [17], até que sejam determinados todos os módulos com baixa complexidade que constituem o sistema.

### 3.3.3 - Linguagem de Representação

Os Diagramas de Estrutura são constituídos pelos seguintes elementos básicos:

- **Subsistema**- representada por um retângulo com duas barra horizontais superior e inferior contendo, na parte central, o descritor do subsistema que deve resumir o conjunto de serviços por ele realizado, na parte superior, um identificador correspondendo ao nível em que se encontra na estruturação "top-down e, na parte inferior, temos um campo reservado para comentários ou rastreamento e é usado quando o modelador achar necessário.
- **Fluxo de Dados** - representado por uma linha (horizontal ou vertical) com setas, apresentando o nome dos dados que nele fluem sempre próximo à seta que indica o sentido de deslocamento do dado.
- **Porta** - representada por um pequeno retângulo na interface do subsistema contendo setas que indicam o tipo de porta e, portanto, a direção de deslocamento do dado que nela flui. A ordem das setas indica a ordem da incidência e emergência dos dados, já que podemos ter fluxos bidirecionados em

portas de entrada com saída associada ou vice-versa. A primeira seta presente na porta (de cima para baixo, ou da esquerda para a direita) indica o fluxo que ocorre primeiro. Portas agregadas mistas, isto é, uma porta de entrada agregada com uma porta de saída, geralmente presentes em níveis de representação mais abstratos, devem ser representadas sem setas.

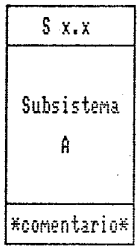
### 3.3.4 - Exemplos de Representação

A seguir apresentamos diversas figuras que ilustram e complementam o que foi exposto na seção precedente, representando todos os elementos de modelagem e as construções descritas.

# DIAGRAMA DE ESTRUTURA

## Sintaxe e Semantica

### - Elementos Basicos de Modelagem

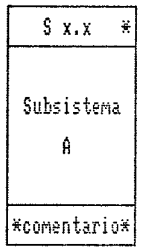



Modela um Subsistema.

No centro, o Descritor do Subsistema, visando identificar facilmente seu conjunto de servicos

Na parte superior, um identificador do seu nivel na construcao hierarquica de Subsistemas

Na parte inferior, um campo destinado a comentario ou rastreamento




O asterisco na parte superior indica que este e um Subsistema Basico.

### PORTAS

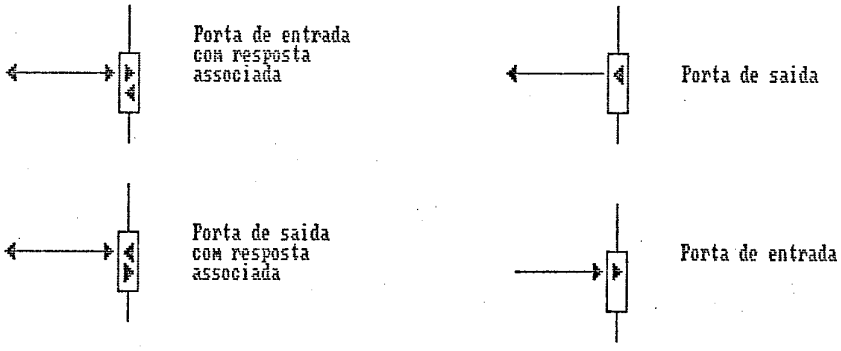



**Figura 3.3.1**

# DIAGRAMA DE ESTRUTURA

## Sintaxe e Semantica

### - Elementos Basicos de Modelagem (Continuacao)



Fluxo unidirecional

Modela uma conexao causal

Ligado somente a portas de entrada ou saida

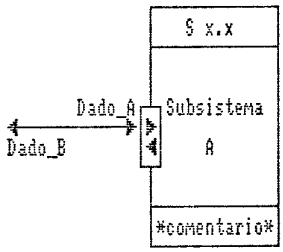


Fluxo bidirecional

Modela conexao causal e sincronismo de comunicacao.

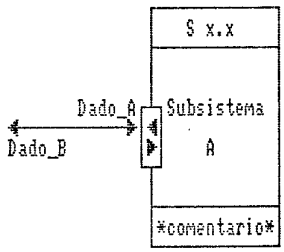
Ligado somente a portas de entrada com resposta associada ou portas de saida com reposta associada.

### - Composicao dos Elementos de Modelagem




O Subsistema A recebe estimulo para executar um pedido de servico atraves de uma porta de entrada com resposta associada.

O pedido do servico contem o dado de entrada DADO_A e a resposta contem o dado de saida DADO_B.




O subsistema A requisita um servico de um outro subsistema atraves de sua porta de saida com resposta associada.

O pedido de servico contem o dado de saida DADO_B e sua resposta contem o dado de entrada DADO_A.

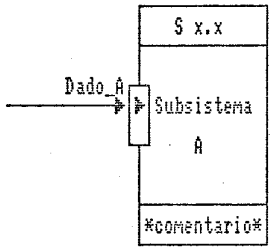
Existe um sincronismo explicito nesta representacao, ja' que o subsistema A pede o servico e fica em estado de espera para receber sua resposta.

Figura 3.3.2

# DIAGRAMA DE ESTRUTURA

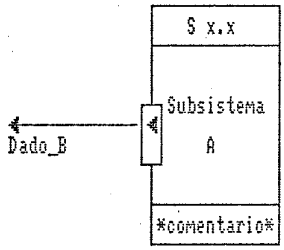
## Sintaxe e Semantica

- Composicao dos Elementos de Modelagem (Continuacao)




O Subsistema A recebe estimulo para executar um pedido de servico atraves de uma porta de entrada. Este servico e' executado produzindo apenas uma resposta interna ao Subsistema A.

O pedido do servico contem o dado de entrada DADO_A.




O subsistema A requisita um servico de um outro subsistema atraves de sua porta de saida.

O pedido de servico contem o dado de saida DADO_B.

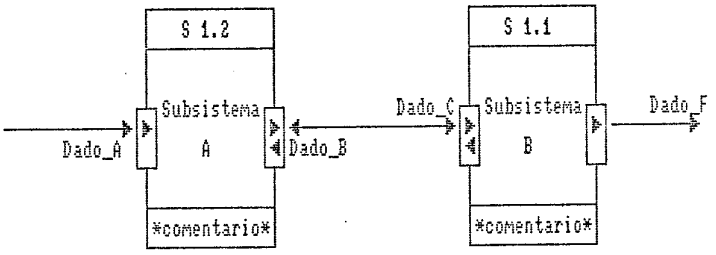
Existe uma representacao de causalidade ja' que a chegada do DADO_B pressupoe o inicio da execucao de um servico sem que haja estado de espera por parte do requisitante do servico.

Figura 3.3.3

# DIAGRAMA DE ESTRUTURA

## Sintaxe e Semantica

- Construcoes Classicas




O Dado_A pede um servico ao Subsistema A, que inicia sua execucao. Mas o Subsistema A precisa, para completa-lo, de um servico disponivel que gera o Dado_B atraves de uma porta de saida com resposta associada.

O Subsistema A pede, enviando o Dado_C, o servico ao Subsistema B usando a porta que o conecta a esse Subsistema, tal como esta' expresso na Linguagem de Configuracao. O Subsistema B inicia a execucao deste servico. Enquanto isso, o servico do Subsistema A que pediu o servico fica em estado de espera.

Quando o Subsistema B produz sua resposta, o Dado_B, o Subsistema A retoma o servico interrompido.

O servico executado pelo Subsistema A possui apenas resposta interna, ja' que nao tem nenhuma saida.

Como o Subsistema A possui uma outra porta de saida, este subsistema oferece um servico, concorrente com o primeiro, estimulado por um valor especifico do Tempo.

O Dado_F e' uma resposta externa produzida pelo Subsistema B. Este dado pode ser gerado, exemplo, por um servico do Subsistema B.

Figura 3.3.4

### 3.3.5 - Regras de Consistência

Conhecidas a sintaxe e a semântica das construções possíveis deve ser levado em conta que, após a construção do conjunto de DE's, deve ser verificado se as seguintes regras de consistência foram seguidas :

- Todos os fluxos de dados presentes em um dado subsistema no nivel x devem aparecer na decomposição desse subsistema no nível x+1. Caso esses fluxos sejam desagregados, essa decomposição deve estar explícita no Dicionário de Dados.
- Todos os serviços de um dado subsistema devem ser mapeados nos subsistemas componentes. Nessa oportunidade, alguns desses serviços podem ser melhor detalhados, com visões internas ao subsistema componente.
- Os Subsistemas Básicos devem possuir um "*", colocado no campo superior da notação, indicando que a eles não mais será associado um DE e sim uma Especificação Detalhada de Serviços.
- Consistências mecânicas tais como :
    - Não existe Subsistema que não produza pelo menos uma informação de saida;
    - Todo subsistema deve ter pelo menos um fluxo de entrada, ou executar serviços a partir da decorrência de um período pré-determinado de tempo.

Além dessas regras de consistência estrita é importante garantir que os DE's não possuem sintomas de "maus DE's".

São eles:

- Número muito grande de interfaces entre dois subsistemas ou número muito grande de subsistemas. Isso diagnostica, provavelmente, a falta de um nível intermediário ou nova divisão de responsabilidade associada aos serviços que o sistema deve executar.
- Repetição do mesmo serviço em mais de um subsistema, o que viola a Lei da Essência Mínima [18].
- Um número muito grande de níveis, o que diagnostica um Projeto Básico muito complexo, sem uma apresentação clara dos Subsistemas Básicos. O uso da recursividade do conceito de sistema permite gerar um Projeto Básico intermediário, no qual Subsistemas Básicos são tratados como novos sistemas e o processo de decomposição é repetido.

## 3.4- Diagramas de Comportamento - Sintaxe e Semântica

### 3.4.1 - Introdução

Este capítulo aborda o Diagrama de Comportamento (DC) [20], explicitando sua função, e apresentando sua sintaxe, semântica e forma de leitura.

Conforme mencionado na seção 3.2, esses diagramas devem ser elaborados em paralelo com os Diagramas de Estrutura (DE), visando explicitar o paralelismo estrito existente entre os diversos Subsistemas que constituem um DE.

Os Diagramas de Comportamento referem-se, portanto, aos Subsistemas presentes no DE, bem como aos dados que fluem entre cada subsistema. Além disso, eles também apresentam os

Serviços de cada subsistema e os Eventos. Eventos modelam pontos de controle de execução.

É importante ressaltar que o conceito de evento aqui usado, além de referir-se a um acontecimento num dado instante de tempo, possui a propriedade de ser memorizado, isto é, pode referir-se a um estado que prevaleceu em algum instante anterior. O evento ao ser usado é consumido.

### 3.4.2 - O que queremos representar?

Conforme já foi ressaltado um sistema computacional é composto de três conjuntos disjuntos : Operações, Dados e Controle.

Em um DE, duas destas grandezas aparecem de forma clara: Operação e Dados. Conseguimos, assim, modelar que Operações são executadas pelo sistema e que Dados são necessários à execução das tarefas do Sistema e são produzidos e usados por seus Subsistemas.

Assim sendo, existe uma dimensão dos Sistemas Computacionais que não foi modelada: o Controle. A alternativa de incorporar funções de controle a um DE ou a Diagramas de Fluxos de Dados (DFD) [17,18], a nosso ver, não consegue transmitir eficientemente o paralelismo eventualmente existente entre funções. Um domínio eficiente de complexidade, que permita o entendimento global do comportamento do sistema, sugere o uso de Diagramas de Comportamento [20].

O uso de Diagramas de Comportamento, associados a

cada DE, visando especificar a interação entre os Subsistemas a cada nível de detalhamento é uma das formas possíveis de modelar Controle.

Os DC's modelam os três conjuntos (operações, dados e controle), através de diagramas com sintaxe e semântica descrita em 3.4.3, combinando:

Subsistema/Ação : Círculo

Evento : Barra

Dados : Descritor de um fluxo de dado associado a um evento

Controle : Modelado através do uso de Composição e Decomposição envolvendo Eventos e Dados, utilizando conectores análogos aos utilizados em Lógica :

x - análogo a E

* - análogo a Ou inclusivo

+ - análogo a Ou exclusivo

Apesar da mistura de dados, operação e controle em um mesmo diagrama, os DC's são correspondentes aos diversos níveis apresentados nos DE's, o que facilita sua leitura e a visão do comportamento do sistema e seus subsistemas nos diveros níveis de particionamento.

### 3.4.3 - Linguagem de Representação

Os Diagramas de Comportamento são constituídos pelos seguintes elementos básicos:

- **Subsistema/Ação** - representado por um círculo dividido por uma barra horizontal dividindo-o, contendo na parte inferior uma ação executada pelo Subsistema e, na parte superior, o descritor do Subsistema responsável por executar essa ação. É importante ressaltar que uma ação representada no DC tem um serviço corrrespondente na descrição de serviços do Subsistema que está sendo modelado.
- **Evento** - representado por uma barra espessa (horizontal ou vertical), apresentando ao seu lado , quando necessário, o descritor do evento ou do fluxo de dados com ocorrência relacionada. Neste último caso, deve haver coincidência com um fluxo de dados presente no DE associado. Se não houver essa coincidência, isso indica que o Evento corresponde a situação interna a um dos subsistemas presente no DE ou a especificação de concorrência ou sincronismo entre os subsistemas. Esses casos não possuem correspondência no DE.
- **Conexão** - representada por um segmento de reta orientado que, se for dirigida de uma Ação para um Evento, modela que a Ação tem como resultado a ocorrência do Evento; pode também ser dirigida de um Evento para uma Ação o que modela a ocorrência do evento como condição necessária para habilitar a execução da Ação.

A ocorrência de um único evento pode não bastar como

condição para estimular uma Ação. Como o resultado de uma ação pode dar origem a mais de um evento, para que possamos representar de forma precisa o comportamento do sistema, é também necessário especificar se existe ou não necessidade de sincronismo dos eventos que iniciam uma ação, bem como, se existe ou não sincronismo nos eventos originários de uma ação.

Usamos para tal o recurso de composição e decomposição de um evento complementando sua representação com o uso de conectores, cujo significado descreveremos a seguir.

Esses conectores indicam como será feita a composição ou decomposição do evento. São operadores que podem ser vistos como análogos a conectores lógicos, já que quando em composição de eventos, o evento composto só ocorrerá, isto é, será verdadeiro, no caso da ocorrência dos eventos que o compõem, respeitando a operação determinada. Esses conectores têm o seguinte significado :

**a - Quanto à Composição**

* - Modela que a ocorrência de um ou mais eventos anteriores, equivale à ocorrência do evento posterior composto. Essa operação é usada para indicar que existe ainda a possibilidade de concorrência na execução dos diversos serviços representados pela Ação conectada ao evento composto.

**+** - Modela que a ocorrência de um dos eventos anteriores equivale à ocorrência do evento posterior composto. Esta operação, usada para indicar que apenas um destes eventos será processado de cada vez pela Ação conectada ao evento composto, modela, portanto, a seqüencialização de um serviço para os diversos eventos que habilitam sua execução.

**x** - Modela que somente a ocorrência de todos os eventos em instantes anteriores de tempo dá origem ao evento posterior composto. Essa operação indica que a Ação conectada ao evento composto só será executada se houver a ocorrência de todos os eventos; portanto, nessas condições, a execução de um serviço só será iniciada a partir do momento em que todos os eventos tiverem ocorrido.

**b - Quanto à Decomposição**

***** - Modela que a ocorrência do evento é equivalente à ocorrência de quaisquer dos eventos posteriores (um ou vários). Esta operação é usada para indicar que a Ação origem conectada ao evento decomposto pode iniciar a execução de várias Ações, do conjunto de ações conectadas aos eventos componentes, sincronamente ou não.

+ - Modela que a ocorrência do evento equivale à ocorrência de apenas um dos eventos posteriores. Esta operação é usada para indicar que a Ação origem conectada ao evento decomposto só pode iniciar a execução, do conjunto de ações conectadas aos eventos componentes, de uma Ação de cada vez, não existindo concorrência entre elas.

x - Modela que a ocorrência do evento equivale à ocorrência de todos os eventos posteriores. Esta operação é usada para indicar que a Ação origem conectada a este tipo de evento inicia a execução todas as Ações conectadas aos eventos componentes, sincronamente.

### 3.4.4 - Exemplos de Representação

A seguir apresentamos diversas figuras que ilustram e complementam a exposição da seção precedente, representando todos os elementos de modelagem e as construções descritas.

### 3.4.5 - Regras de Consistência

Tendo em vista a sintaxe e a semântica das construções possíveis e após a construção de um DC, deve ser verificado se as seguintes regras foram seguidas :

- A toda ação está associada um evento que a estimula para iniciar a execução. Este evento pode ser um fluxo de dados presente no DE associado, uma ocorrência de valor específico de Tempo (condição temporal) ou um evento equivalente a composição ou decomposição de outros eventos.
- Após sua execução, toda ação dá origem a um evento, que pode ou não se decompor em outros eventos. Se a execução der origem apenas a uma Resposta Interna, o evento resultante deve apresentar esse descritor.
- Todos os fluxos de dados do DE associado devem aparecer no DC associado.
- Todos os subsistemas do DE , e apenas esses, devem aparecer no DC associado.

## 3.5 - Como Fazer a Especificação de um Subsistema Básico

Conforme descrito anteriormente, para cada Subsistema Básico será feita uma especificação [21] que deve conter:

**1 - Descrição dos Serviços**

**1.1 - Serviços Externos**

- **Entradas** : que o subsistema necessita para possibilitar a execução do serviço;
- **Saídas** : respostas fornecidas pelo serviço;

# DIAGRAMA DE COMPORTAMENTO

## Sintaxe e Semantica

### Elementos Basicos de Modelagem


Modela um servico executado pelo subsistema

Onde, Acao pode ser um servico ou parte dele.



Modela um evento. Ao seu lado pode constar o Descritor de um fluxo de dados presente no DE correspondente ou um "lugar" do subsistema, modelando dado gerado internamente a ele.



Representa uma conexao, modelando ligacao entre dois dos elementos anteriores.
Uma conexao pode unir uma acao a um evento e vice-versa, e um evento a outro evento.

* Conector que modela uma operacao de composicao com valor analogo ao OU inclusivo da logica.

× Conector que modela uma operacao de composicao com valor analogo ao E da logica.

+ Conector que modela uma operacao de composicao com valor analogo ao OU exclusivo da logica.

" " Comentario, muitas vezes necessario para melhor entendimento do diagrama.

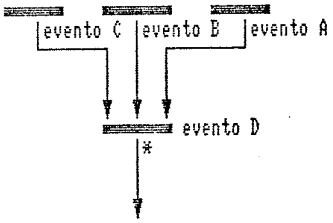
Figura 3.4.1

# DIAGRAMA DE COMPORTAMENTO

## Sintaxe e Semantica

### Composicao e Decomposicao de Elementos de Modelagem

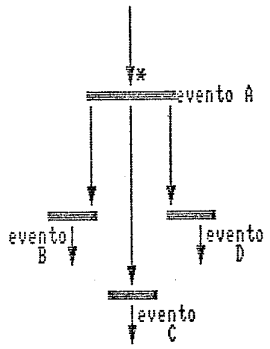
### Composicao de Eventos - OU inclusivo




A ocorrencia de quaisquer dos tres eventos A,B ou C implica ocorrencia do evento D.

Esta notacao significa que pode existir concorrencia no acontecimento dos eventos A,B e C, de forma que o acontecimento de um deles, ou qualquer composicao possivel da' origem ao evento D. Por exemplo, a ocorrencia de A e C, ou a ocorrencia de A, B e C da' origem ao evento D.

### Decomposicao de Eventos - OU inclusivo



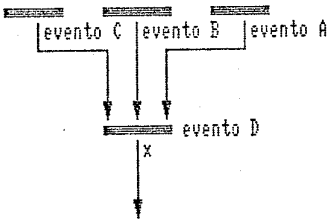

A ocorrencia do evento A implica ocorrencia de um ou mais dos eventos subsequentes, isto e', pode dar origem, por exemplo, ao evento A, ou aos eventos B e A, ou aos eventos A, B e C.

Figura 3.4.2

# DIAGRAMA DE COMPORTAMENTO

## Sintaxe e Semantica

### Composicao de Eventos - E

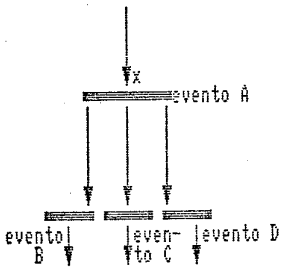



Somente a ocorrencia dos 3 eventos da' origem ao evento D.

Isto implica a existencia de um sincronismo necessario a geracao do evento D.

A ocorrencia de A,B e C nao necessita ser simultanea; nesta notacao existe memoria do que ja' ocorreu e espera pelos outros eventos.

### Decomposicao de Eventos - E




A ocorrencia do evento A implica ocorrencia simultanea dos eventos B, C e D.

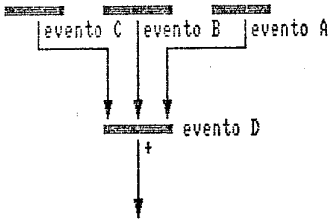
Neste caso o sincronismo e a simultaneidade sao especificadas.

Figura 3.4.3

# DIAGRAMA DE COMPORTAMENTO

## Sintaxe e Semantica

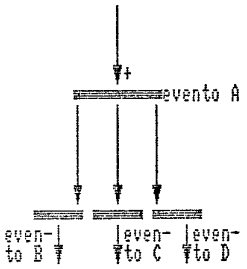
### Composicao de Eventos - OU exclusivo




A ocorrencia de quaisquer dos tres eventos A,B ou C implica a ocorrencia do evento D.

Esta notacao implica que uma ocorrencia de cada vez da' origem ao evento D. Existe, portanto, uma sequencializacao dos eventos.

### Decomposicao de Eventos - OU exclusivo




A ocorrencia do evento A implica ocorrencia de um dos eventos subsequentes.

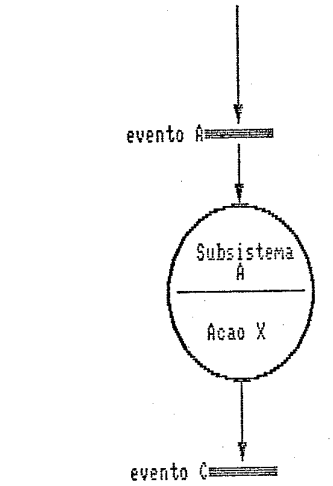
Esta notacao implica que os eventos B, C e D nao acontecem simultaneamente.

Figura 3.4.4

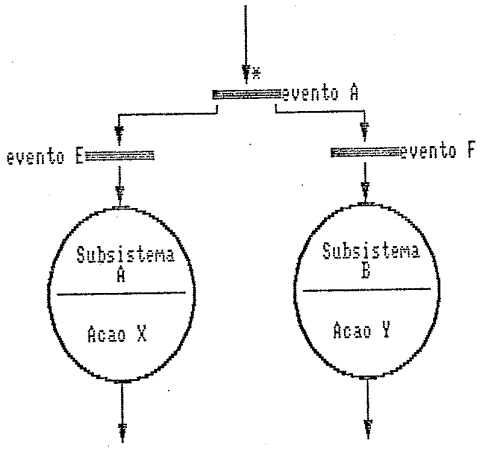
# DIAGRAMA DE COMPORTAMENTO

## Sintaxe e Semantica

### – Ativacao de um Subsistema




A ocorrencia do evento A faz com que o Subsistema A realize a acao X e da' origem ao evento C.



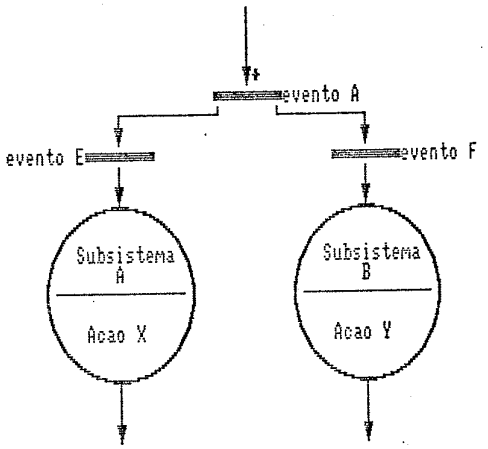

A ocorrencia do evento A pode dar origem aos eventos E, F, E ou E e F, o que ativa A, B ou A e B simultaneos, respectivamente.

**Figura 3.4.5**

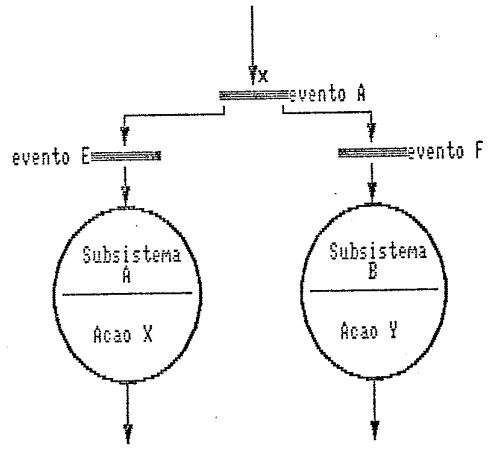
# DIAGRAMA DE COMPORTAMENTO

## Sintaxe e Semantica

### - Ativacao de um Subsistema




A ocorrencia do evento A da' origem a um dos eventos E ou F, que ativam os subsistemas A e B, respectivamente.



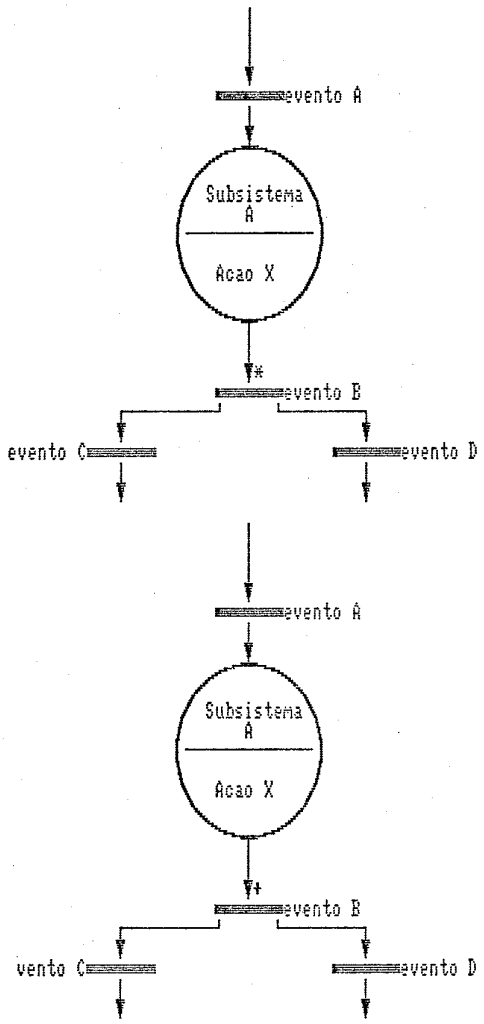

A ocorrencia do evento A da' origem aos eventos E e F, o que ativa os Subsistemas A e B simultaneos, respectivamente.

Figura 3.4.6

# DIAGRAMA DE COMPORTAMENTO

## Sintaxe e Semantica

### - Geracao de Eventos e sua Decomposicao




A ocorrencia do evento A ativa a acao X do Subsistema A. Esta acao da' origem ao evento B, que equivale aos eventos C e/ou D.

A ocorrencia de C e D pode ou nao ser simultanea.

A ocorrencia do evento A ativa a acao X do Subsistema A. Esta acao da' origem ao evento B, que equivale aos eventos C ou D.

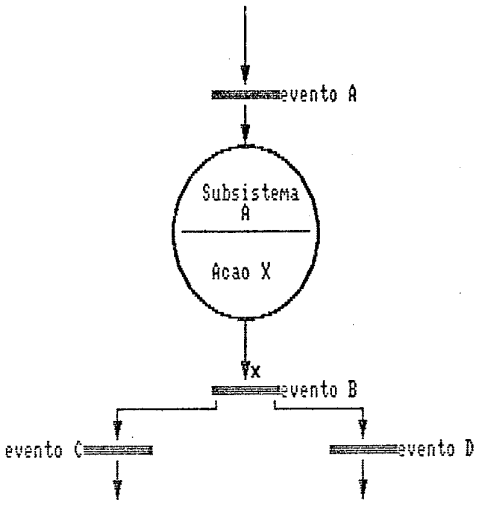
A ocorrencia de C e D nao e' concorrente.

Figura 3.4.7

# DIAGRAMA DE COMPORTAMENTO

## Sintaxe e Semantica

- Geracao de Eventos e sua Decomposicao (Continuacao)

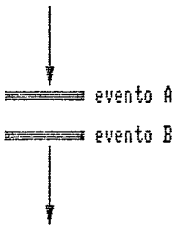



A ocorrencia do evento A, ativa a acao X do Subsistema A. Esta acao da' origem ao evento B, que equivale aos eventos C e D.

A ocorrencia dos eventos C e D e' simultanea.

- Quando a ocorrencia de um evento e' sinalizada para outro Subsistema ou para uma Entidade Externa.


- Quando a ocorrencia de um evento e' sinalizada para outro Subsistema e a essa sinalizacao corresponde uma resposta associada.




- Conectores para indicacao de continuacao, visando melhorar o entendimento do diagrama.


Conector com um numero para que se possa identificar a outra ponta da conexao.

Figura 3.4.8

# DIAGRAMA DE COMPORTAMENTO

## Sintaxe e Semantica

- Acesso a Area de Dados Compartilhada

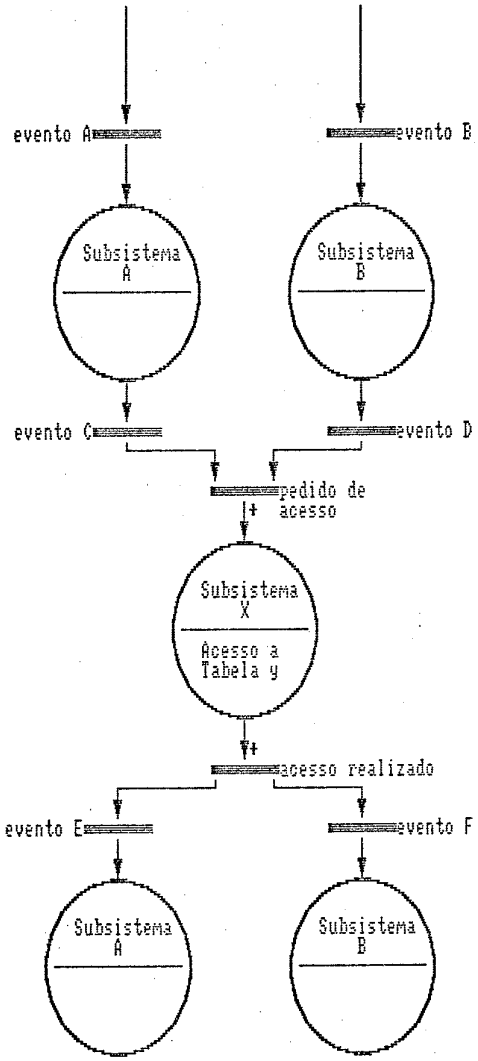



Figura 3.4.9

- **Descrição** : ainda numa forma análoga à de relações matemáticas (linguagem de representação não-procedimental), contendo a descrição de todas as condições de contorno, isto é, restrições impostas pelo ambiente externo ao Subsistema, tais como, pontos de descontinuidade de função, variações máximas previstas.

1.2 - **Serviços Internos**

- **Fonte de Ativação** : como não possui uma solicitação através de um pedido de serviço, deve apresentar a condição que ativa esse serviço (por exemplo, uma condição temporal), sob a forma de evento.
- **Saídas** : listar dados de saída originadas na execução dos serviços
- **Descrição** : ainda numa forma análoga à de relações matemáticas (linguagem de representação não-procedimental), contendo a descrição de todas as condições de contorno, isto é, restrições impostas pelo ambiente externo ao Subsistema.

1.3 - **Serviços Requisitados** : Serviços que são pedidos a outros subsistemas.

- **Saídas** : a ser geradas para pedir o serviço a outro subsistema.
- **Respostas** : que o subsistema espera receber.
- **Descrição** : ainda numa forma análoga a de

relações matemáticas (linguagem de representação não procedimental), contendo a descrição de todas as condições de contorno que devem ser obedecidas pelo prestador do serviço.

2 - **Descrição Detalhada dos Dados** (Dicionário de Dados)

Aqui os dados são detalhados usando BNF ou tipos PASCAL. Devem ser mencionados todos os dados apresentados nos serviços externos, internos e requisitados. Nesse nível, para cada dado apresentado como Entrada ou Saida na seção anterior, devem ser especificados detalhes tais como: estrutura do dado (por exemplo: array, record) se o dado for composto; unidade, precisão, formato e domínio de valores se o dado for elementar.

3 - **Descrição de Concorrência**

- Concorrência de Serviços
- Prioridades relativas entre os serviços

Neste item devem ser detalhados todos os serviços, ou grupo de serviços, cuja execução não pressupõe, devido à funcionalidade, a execução anterior de um outro serviço do Subsistema Básico a que pertencem. Determina também os grupos de serviço que não devem concorrer na execução. Isto não significa que esses serviços são seqüenciais e sim que devem ser executados um de cada vez. A seqüencialidade já deve ter sido apresentada como condição de contorno na descrição detalhada do serviço.

4 - **Requisitos de Desempenho** (quando for o caso)

- Tempo máximo para execução de um serviço;
- Tempo médio para execução de um serviço.

Esses tempos devem sempre ser fornecidos em função de carga de processamento máxima e média.

5 - **Parâmetros do Subsistema Básico**

Neste item devem ser detalhados todas as variáveis do sistema passíveis de parametrização; por exemplo, números máximos para armazenamento de um dado, alocação de memória etc..

A especificação do Subsistema Básico, como já é um documento de implementação, deve possuir também um ítem, Subsídios de Implementação, onde serão registradas normas e utilitários a ser utilizados no desenvolvimento. Serão, ainda, especificados a linguagem de programação, o ambiente de desenvolvimento ( por exemplo: compilador, emulador), o software básico, bem como os procedimentos especiais na manipulação de dados, como a escolha de algum algoritmo específico.

### 3.6 - Projeto ("Design") de um Subsistema Básico

Um subsistema básico é formado de um ou mais PROCESSOS DE AUTOMAÇÃO, conforme já mencionado anteriormente.

Os processos de automação são as unidades de execução concorrentes - PROCESSOS para um supervisor de tempo real.

Portanto, o primeiro passo no "design" do Subsistema Básico é a identificação dos processos que o compõem.

Após a determinação dos processos, passamos para outra

etapa, que é a da determinação da estrutura destes processos. Essa determinação nos levará aos módulos ou rotinas que compõem cada processo.

Essas duas etapas são distintas tanto em execução, como nas ferramentas usadas para documentação e, portanto, vamos descrevê-las separadamente.

### 3.6.1 - Determinando os Processos de um Subsistema Básico

Para tal, temos que fazer uso das seguintes informações:

- O conjunto de serviços e seu detalhamento, constantes da Especificação do Subsistema Básico.
- A concorrência entre esses serviços, também constante do mesmo documento.

A partir destes dados, devem ser seguidos os seguintes passos:

- A cada grupo de serviços especificados como não-concorrentes, deve ser associado um PROCESSO.
- Deve, então, ser construido um DIAGRAMA DE ESTRUTURA onde apareçam os relacionamentos entre os diversos processos. É bom ressaltar que nesse diagrama podem aparecer estruturas de dados compartilhadas.
- Caso o número de processos seja grande e esse fato dê origem a um diagrama de estrutura complexo, para melhorar a documentação e o entendimento do problema, deverá ser feita uma agregação de processos, com serviços afins, construindo-se uma estrutura hierárquica, visando tornar mais visível a segmentação do Subsistema Básico em

processos.

- A cada DIAGRAMA DE ESTRUTURA deve corresponder um DIAGRAMA DE COMPORTAMENTO, evidenciando as concorrências existentes, na mesma forma do PROJETO BASICO.
- A cada par diagrama de estrutura/diagrama de comportamento, deve ser associado um texto contendo o mapeamento dos serviços do Subsistema Básico a cada processo ou o grupamento de processos neles presente. Este texto deve conter também a descrição dos fluxos de dados que aparecerem nos diversos diagramas. Estes devem ser descritos de acordo com sua composição e significado.

É importante ressaltar que até o nível de processos são válidos os conceitos de PORTA e INTERFACE. Além disso toda comunicação e sincronismo são feitos através das primitivas ENVIA/RECEBE. Acessos a estruturas de dados compartilhadas não são necessárias em processos concorrentes, já que não trariam nenhum benefício adicional, podendo até introduzir um dispêndio adicional ("overhead") com a necessidade de introduzir exclusão mútua.

### 3.6.2 - Determinando a Estrutura de um Processo

Um processo pode necessitar, para sua implementação, de um trabalho ainda grande de projeto, já que, apesar de estar associado a serviços de fácil e rápido entendimento, tem que executar um conjunto grande de algoritmos, condições e controles.

Para processos é usada a decomposição tradicional em módulos ou rotinas, numa hierarquia de subordinação. Essa decomposiçãço é baseada em uma decomposição funcional, procurando rotinas que possam ser vistas como "caixas pretas", concentrando-se funções similares em uma mesma rotina e evitando-se o uso de variáveis globais. A decomposição funcional "top-down" é ainda um bom método para determinação das rotinas que compõem o processo.

A documentação desta decomposição deverá ser feita através de uma árvore apresentando a hierarquia das diversas rotinas. Caso o número de rotinas seja muito grande, a árvore poderá ser apresentada em níveis, isto é, apresentamos a árvore até o segundo nível de decomposição, por exemplo, e cada uma das rotinas deste nível terá sua árvore correspondente apresentada separadamente.

Devem ser documentadas também as estruturas de dados compartilhadas. Essa descrição deve apresentar seu significado, composição e os controles necessários à integridade dos dados, mas devemos ressaltar que os procedimentos que compartilham essas estruturas não são concorrentes.

A cada rotina deve estar associada uma definição formal da função a ela designada, bem como de seus parâmetros de entrada e saída. Esta definição deve ser feita através de um linguagem precisa - Linguagem Estruturada.

De qualquer forma é importante ressaltar que esta definição não deve jamais descer a níveis de detalhe

necessários às linguagens de programação. O que desejamos é descrever precisamente o que será feito naquela rotina. No entanto, se nesta descrição são colocados níveis de detalhe semelhante a código, teremos uma documentação final de difícil manutenção, e que não apresenta de forma clara e simples a função de cada rotina do processo que está sendo estruturado.

Para comunicação entre rotinas não mantivemos os mecanismos de envia/recebe devidos ao dispêndio adicional introduzido nos tempos de processamento e à falta de concorrência, já que aqui cada segmento tem natureza seqüencial. Os mecanismos de CALL (chamada de sub-rotina) são mais próprios para um processamento seqüencial e mais eficientes.

## 3.7 - Configuração do Sistema

Essa etapa, determina o mapeamento dos subsistemas básicos na Arquitetura de Hardware, como já antecipamos no item 2.6. Mas, é importante detalhar a forma como essa arquitetura é integrada ao software.

É gerada uma unidade independente de compilação, que faz uso da linguagem de configuração e que será denominada UNIDADE DE CONFIGURAÇAO. Nessa etapa será efetivamente configurado o sistema- Configuração do Software e Configuração de Arquitetura, fazendo-se uso da linguagem de configuração, cujas primitivas serão descritas no item 4.

## 4 - Primitivas de Suporte ao Desenvolvimento

O método apresentado visa a geração de um software com alto grau de independência em relação ao hardware e ao software básico.

No entanto, essa independência só é efetivamente possível, fazendo uso de uma camada entre o software básico e o software de aplicação. Essa camada tem que ser definida na forma mais portátil possível, mas sempre será necessário um trabalho de adaptação em casos de mudança do software básico disponível. Mudanças de configuração de hardware implicam primeiramente em mudanças na configuração de arquitetura do sistema.

Definiremos, então, primitivas com sintaxe padrão, e que abranja as necessidades do método.

Temos duas classes de primitivas :

- **Primitivas de Comunicação** : implementam a comunicação entre processos e Subsistemas Básicos.

- **Primitivas de Configuração** : implementam a linguagem de configuração.

- **PRIMITIVAS DE COMUNICAÇÃO**

---> Envia (Porta_a,End_msg_env)

**Porta_a** : identificação da porta que é da forma (Nome da Porta + Nome do Processo + Nome do Subsistema Básico)

**End_msg_env** : endereço com o conteúdo da mensagem a enviar

Essa primitiva, através das informações de conexão entre portas, acha em que porta deve ser depositada a mensagem. Caso a porta seja do tipo que tenha resposta associada, deve ser armazenado o "remetente" da mensagem, para que seja possível o envio da resposta. Isto é, quando é feito um envio através de uma porta de entrada, a mensagem deve ser depositada no seu "remetente".

---> Recebe (Porta_a, End_msg_rec)

**Porta_a** : identificação da porta que é da forma (Nome da Porta + Nome do Processo + Nome do Subsistema básico)

**End_msg_env** : endereço onde será armazenado o conteúdo da mensagem a receber

Essa primitiva busca na área de mensagens associada a essa porta a primeira da fila para entregar ao processo que deseja consumi-la. Caso não tenha mensagem disponível, o processo deve entrar em estado de espera.

- **PRIMITIVAS DE CONFIGURAÇÃO**

---> Define_Porta ( Porta_a, Tipo, Tam_msg, Tam_fila)

**Porta_a** : identificação da porta que é da forma (Nome da Porta + Nome do Processo + Nome do Subsistema básico)

**Tipo** : Entrada | Saída | Entrada com Resposta | Saída com resposta

**Tam_msg** : Número de bytes do tipo que descreve a porta.

**Tam_fila** : Tamanho da fila de mensagens disponível para aquela porta.

A cada porta é associada uma lista de mensagens, circular, onde serão depositadas as mensagens enviadas.

O tamanho dessa lista é função de cada porta, pois depende da relação produtor/ consumidor.

OBS : A definição de porta fica melhor implementada através de um pré-compilador que determinaria o número de bytes, bem como a consistência das conexões.

---> Conecta (Porta_a, Porta_b);

**Porta_a** : Porta de saída de um dado processo.Subsistema Básico

**Porta_b** : Porta de entrada de um dado processo.Subsistema Básico.

Essa primitiva vai armazenar a conexão de duas portas, permitindo resolver a chamada da primitiva Envia, pois é mencionado apenas a porta de saída do processo que a chamou.

---> Define_Processo ( Nome_processo, Nó, ...)

Essa primitiva visa definir um processo, em um dado elemento da arquitetura de hardware (Nó), com todos os elementos necessários, tais como, informação de ciclo, encarnações, etc...

---> Ativa_Processo (Nome_processo)

Torna o processo uma unidade em execução.

É importante ressaltar que essa linguagem de configuração dá origem a uma estrutura de dados com as informações de configuração. Essa estrutura fará parte da camada situada entre o software básico e o software da aplicação. Essa camada, tal como o software básico, tem que ser multi-cpu para uma arquitetura distribuída.

# 5 - Conclusão

O método aqui apresentado visa facilitar o desenvolvimento de sistemas distribuídos de grande porte e de tempo_real.

Muitos dos pontos aqui colocados, como a Estratégia de Desenvolvimento Incremental, facilitam bastante o desenvolvimento desse tipo de sistemas.

Apesar da necessidade de sempre ser feita a camada entre o software básico e o software da aplicação, de forma a tornar a aplicação mais portátil para hardware e software básico, esse trabalho é desprezível em relação ao tempo gasto no desenvolvimento de um sistema desse porte.

A documentação é gerada em conjunto com o desenvolvimento, retratando de forma clara o pensamento do projetista, e permitindo fácil acesso ao conhecimento de pontos específicos do sistema - os Subsistemas Básicos.

Como os Subsistemas Básicos são independentes, podemos, com o passar do tempo, e com uma boa organização do ambiente de trabalho, dispor de uma biblioteca de Subsistemas Básicos significativa. Isso nos levaria a mudar de comportamento no desenvolvimento de um sistema.

Após as fases de construção do Modelo da Essência e do Projeto Básico, seria feita uma busca na Biblioteca de Subsistemas Básicos existente e grande parte dos subsistemas necessários ao projeto em andamento já estariam disponíveis, sendo necessário apenas parametrizá-los corretamente.

Teríamos, portanto, desenvolvimento mais rápido e mais eficiente, já que erros internos a subsistemas amplamente usados e, portanto, testados, seriam em número muito pequeno.

## 6 - Bibliografia


[1] - Constructing Distributed Systems in CONIC

Jeff Magee, Jeff Kramer,Morris Sloman

16 de março de 1987 - Imperial College Research Report -Doc 87/4

[2] - The CONIC Support Environement for Distributed Systems

Jeff Magee, Jeff Kramer, Morris Sloaman

Agosto de 1986 - NATO ASI series

[3] - Flexible Communication Structure for Distributed Embebed Systems

Jeff Magge, Jeff Kramer, Morris Sloman

julho de 1986 - IEEE proceedings vol 133, Pt. E, no 4

[4] - A_HAND - Ambiente de Desenvolvimento de Software, baseado em hierarquias de abstração em níveis diferenciados.

Rogério Drummond, Hans Liebenberg

abril 1987 - Depto de Ciências de Computação - UNICAMP

[5] - CONIC : An Integrated Approach to Distributed Computer Control Systems

Jeff Magee, Jeff Kramer, Morris Sloman

jan 1983 - IEEE Proc. Vol 130,Pt. E, no 1

[6] - Language Support for Changeable Large Real Time Systems

Ivar Jacobson

OOPSLA 86 Proceedings

[7] - The Architecture of an Integrated Local Network

Paul J. Leach, Paul H. Levine, Bryan P. Douros, James A. Hamilton, David L. Nelson, Bernard L. Stumpf

Outubro de 1983 - IEEE

[8] - Distribution and Abstract Types in Emerald

Andrew Black, Norman Hutchinson, Eric Jul, Henry Levy and Larry Carter

Jan 1987 - IEEE - Transactions on Software Engineerings, vol SE 13, no 1

[9] - Object Structure in the EMERALD System

Andrew Black, Norman Hutchinson, Eric Jul, Henry Levy

OOPSLA 86 Proceedings

[10] Distributed Systems and Computer Networks

Morris Sloman and Jeff Kramer

1987 - Prentice Hall

[11] - Concurrent Program Structure

David Bustard, John Eldore, Jim Welsh

1988 - Prentice Hall

[12] - Software Engineering Concepts

Richard E. Fairley

1985 - McGraw - Hill

[13] - Software Engineering - A Practitioner's Approach

R. S. Pressman

1988 - McGraw - Hill

[14] - Essential Systems Analysis

S.McMenamin e J. Palmer

1984 - Yourdon Press

[15] - System Development Without Pain

P. T. Ward

1984 - Yourdon Press

[16] - No Silver Bullet

F. P. Brooks

1987 - Computer pg 10

[17] - "Uma Perspectiva Sistêmica sobre a Engenharia de Software"

G. Richter & B. Maffeo

Revista de Informática Teórica e Aplicada Vol1, No 1, 1989

[18] - "Engenharia de Software e Especificação de Sistemas"

B. Maffeo

(livro em edição pela editora CAMPUS)

[19] Structured Development for Real-Time Systems

Paul T. Ward & Stephen J. Mellor

1985 - Yourdon Press

[20] Manual de Normas do Centro de Apoio a Programação

Publicação Interna da Diretoria de Armamento e Comunicações da Marinha do Brasil

[21] Plano de Documentação do SITAN

Publicação Interna da Diretoria de Armamento e Comunicações da Marinha do Brasil