PUC

ISSN 0103-9741

Monografias em Ciência da Computação
nº 11/92

Modelo de Implementação de um Terminal Tático Inteligente

Maria Luiza d'Almeida Sanchez
Albrecht Karl von Plehwe

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO
RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22453-900
RIO DE JANEIRO - BRASIL

PUC RIO - DEPARTAMENTO DE INFORMÁTICA ISSN 0103-9741

Monografias em Ciência da Computação, Nº 11/92
Editor: Carlos J. P. Lucena Abril, 1992

Modelo de Implementação de um Terminal Tático Inteligente*

Maria Luiza d'Almeida Sanchez
Albrecht Karl von Plehwe

* Este trabalho foi patrocinado pela Secretaria de Ciência e Tecnologia da Presidência da República Federativa do Brasil.

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
22453-900 — Rio de Janeiro, RJ
Brasil
Tel. +55-21-529 9386 Telex +55-21-31048 Fax +55-21-511 5645
E-mail: rosane@inf.puc-rio.br
techrep@inf.puc-rio.br (for publications only)

Resumo

Este trabalho apresenta o Modelo da Implementação de um Terminal Tático Inteligente. Para tal, faz uso das técnicas que tratam o sistema como um conjunto de serviços executados por um grupo de Subsistemas Autônomos que se comunicam através de troca de mensagens.

Palavras Chave

Terminal Tático Inteligente / Modelo da Implementação / Subsistemas Autônomos / Serviços.

Abstract

This work presents the Implementation Model of an Intelligent Tactical Console. This model considers the system as a group of services executed by Independent Subsystems which interact exchanging messages.

Keywords

Intelligent Tactical Console / Software Design / Independent Subsystems / Services

Sumário

Introdução

Este trabalho apresenta parte do Modelo da Implementação do software do Terminal Tático Inteligente (TTI), atualmente em uso na Marinha do Brasil. A modelagem foi baseada em técnicas e ferramentas de Design Orientado ao Encapsulamento de Dados e à Troca de Mensagens entre Subsistemas Autônomos [41], destinadas ao Design de Sistemas Concorrentes.

Um **Sistema Concorrente** caracteriza-se por conter componentes cooperativos - **Processos**, com execução intercalada no tempo ou até simultânea, caso o computador disponha de mais de um processador. A cooperação entre componentes se dá através de troca de mensagens ou compartilhamento de dados. Para que processos concorrentes e cooperativos, executando a velocidades diferentes, possam interagir em determinados pontos de execução, existem mecanismos de sincronização/comunicação entre eles. Exemplos de Sistemas Concorrentes incluem: sistemas de controle de processos industriais, sistemas de simulação e sistemas operacionais.

Sistemas Concorrentes freqüentemente possuem uma complexidade maior que sistemas seqüenciais, agravada ainda por requisitos rígidos de desempenho e segurança. Isso aumenta o grau de risco do desenvolvimento, as exigências de qualidade dos produtos e o custo.

A necessidade de qualidade excepcional e de redução do risco no desenvolvimento de um Sistema Concorrente, torna aconselhável a utilização de um método que permita o planejamento e a gerência de prazos e custos e que estabeleça a qualidade esperada. Emprega-se técnicas que segmentem o sistema em partes menores, mais compreensíveis e de menor nível de abstração. Sendo descritas com mais precisão e enfatizando pontos anteriormente obscuros, o caminho para implementá-las é mais facilmente previsível. O planejamento do desenvolvimento é efetuado sobre um modelo de ciclo de vida que o organiza em fases de Análise, Design e Implementação.

O método utilizado aqui concentra-se na fase de Design, onde é determinada a estrutura interna e detalhes de processamento do produto de software. Essa fase é dividida em dois níveis: Projeto Básico e Especificação de Subsistemas Básicos. O **Projeto Básico** fornece a visão global da estrutura do sistema em termos de **Subsistemas Básicos** - unidades de trabalho para os membros da equipe de projeto. Cada serviço do sistema só é executado por um único subsistema, agilizando a determinação de erros ou alteração de funcionalidade. Dessa forma, facilita-se a manutenção e a gerência do desenvolvimento. A troca de um equipamento que interage com o sistema, implica apenas em mudanças no subsistema que o trata.

No desenvolvimento de sistemas de grande porte é comum a ocorrência de mudanças na equipe. A utilização desse método permite que um novo membro na equipe obtenha, em pouco tempo, uma visão geral do sistema, a partir do Projeto Básico, e se detenha na Especificação e Design do subsistema autônomo (Subsistema Básico) ao qual foi alocado, não necessitando compreender profundamente

tudo o que já foi feito. Outros aspectos do gerenciamento dessa técnica de Design podem ser vistos em [40].

O capítulo 1 deste trabalho contém uma visão geral das funções do Terminal Tático Inteligente, que resume a Especificação Operativa [42], usada como base para essa implementação e onde se procura definir todos os serviços executados pelo sistema.

No capítulo 2 é feita uma retrospectiva dos métodos empregados no desenvolvimento de Sistemas Concorrentes, contextualizando a técnica de Design descrita em [41] e aqui utilizada. Os conceitos em que se baseia esta técnica são resumidos no capítulo 3.

No capítulo 4, encontra-se um exemplo parcial do Projeto Básico e da Especificação do Subsistema Básico Mov_Acomp_Man, visando explicitar a utilização desta técnica.

No capítulo 5, a conclusão deste trabalho que ressalta as vantagens do emprego do método, comprovadas na prática com a implementação do Terminal Tático Inteligente. Termos e expressões comuns ao ambiente da Marinha são definidos em um glossário. Anexo, uma lista contendo as abreviaturas utilizadas, muitas das quais, normalmente, usadas no âmbito da Marinha e outras criadas para facilitar a denominação dos dados. A bibliografia referenciada, conclui este trabalho.

1 - Visão Geral das Funções do TTI

O TTI - Terminal Tático Inteligente, foi construído para facilitar a operação dos navios de guerra. Ele é um Sistema de Informações Táticas, isto é, capaz de armazenar um conjunto de dados, extraídos dos sensores de bordo, tratando-os e apresentando-os ao operador, de forma organizada, própria para a tomada de decisões.

Durante uma missão de um navio é necessária a constante observação do ambiente que o cerca. A navegação deve ser efetuada com a monitoração de qualquer embarcação que se aproxime do navio bem como de pontos de terra. Para tal, os navios possuem radares de navegação cuja imagem (aqui chamada de Vídeo Bruto) é apresentada ao operador.

Sobre a imagem do Vídeo Bruto, o Terminal Tático Inteligente permite marcar embarcações, pontos de terra e outros, importantes à navegação e à segurança do navio, denominados de Acompanhamentos. Esses Acompanhamentos são apresentados como um conjunto de símbolos gráficos, referenciado como Vídeo Sintético, possibilitando a análise da situação, tanto para navegação como para defesa. Como os símbolos presentes no Vídeo Sintético, na maioria das vezes, correspondem a outras embarcações, eles possuem movimento, e o sistema é responsável por calcular suas posições, a partir de seus rumos e velocidades, atualizando-as na tela periodicamente. Assim mantém-se a consistência das apresentações dos Vídeos Bruto e Sintético. Atuando sobre suas formas de apresentação, existem funções como: mudança de escala, seleção dos acompanhamentos que ficarão visíveis (filtros de apresentação), etc, o que facilita a análise das informações apresentadas.

Os Acompanhamentos são de vários tipos: Acompanhamentos Manuais, Pontos de Referência, o Próprio Navio e outros, correspondentes a forma de obtenção das informações de posição, rumo e velocidade e ao seu emprego para navegação ou avaliação tática.

A observação visual do Vídeo Bruto permite ao operador acompanhar embarcações ou aeronaves, de interesse tático, associadas a pontos de eco do radar - Acompanhamentos Manuais. Esses Acompanhamentos necessitam da intervenção do operador para sua criação, quando é selecionado que eco se deseja acompanhar, e atualização de sua movimentação (rumo e velocidade). O operador deve observá-los e, sempre que o eco radar correspondente no Vídeo Bruto não estiver sob o símbolo equivalente no Vídeo Sintético, deve efetuar uma atualização de posição deste Acompanhamento. A partir dessa atualização, o TTI calcula automaticamente os novos rumo e velocidade.

Acompanhamentos do tipo Ponto de Referência são característicos da navegação e correspondem a pontos fixos de terra, pontos que se movimentam em função da velocidade do vento ou da corrente marítima.

O TTI oferece, como auxílio à navegação, a atualização automática e periódica da posição do navio - Acompanhamento do Próprio Navio. Para tal, está interligado aos sensores que medem velocidade e rumo atuais do navio, odômetro e agulha giroscópica, respectivamente, e calcula a nova posição a intervalos regulares (1/4 seg.).

Os Acompanhamentos possuem um identificador, chamado Número de Acompanhamento, único para o sistema e uma Categoria que os classifica quanto a tipo de embarcação e grau de perigo que representam.

O TTI automatiza várias funções que atuam sobre Acompanhamentos e que eram realizadas manualmente pelos operadores. Essas funções são chamadas de Soluções Táticas e permitem avaliação da segurança de navegação e de posições de combate. Por exemplo, está disponível o cálculo de Pontos de Maior Aproximação que obtém o local geográfico onde ocorrerá a menor distância entre duas embarcações, de acordo com seus rumos e velocidades atuais. Outra Solução Tática é a que permite manter a história das posições ocupadas por Acompanhamentos, isto é, posições anteriores espaçadas a intervalos de tempo regulares (predefinidos ou selecionados pelo operador) são armazenadas e apresentadas no Vídeo Sintético - Pontos Históricos. O TTI possibilita também a recepção e transmissão de informações entre sistemas de um conjunto de navios.

Ele foi projetado para ser operado de forma simples, com uma Interface Usuário-Máquina amigável e, portanto, de rápido aprendizado pela tripulação do navio. Seu console de operação é composto de um monitor de vídeo gráfico onde é apresentado o Vídeo Bruto do radar digitalizado com o Vídeo Sintético sobreposto, um teclado e um cursor gráfico para interação com o operador.

O cursor gráfico, acoplado a uma Esfera de Acompanhamento - EA ("Tracker Ball"), tem sua posição na tela representada por um pequeno círculo (Marca da Esfera de Acompanhamento - MEA), atualizada sempre que houver a movimentação da EA. No entanto, para que a posição atual do cursor seja processada pelo sistema, o operador deve pressionar a tecla - POS EA. Essa tecla está localizada ao lado da EA, acessível pelo polegar da mão que a movimenta.

1.1 - Arquitetura de Hardware

O Hardware do TTI é baseado em um conjunto de computadores do tipo Motorola 68020 e Intel 8086, interligados através de um barramento VME ou via linha serial do tipo RS-232/422. A fig. 1.1 apresenta o Diagrama de Blocos correspondente.

Diagrama de Blocos

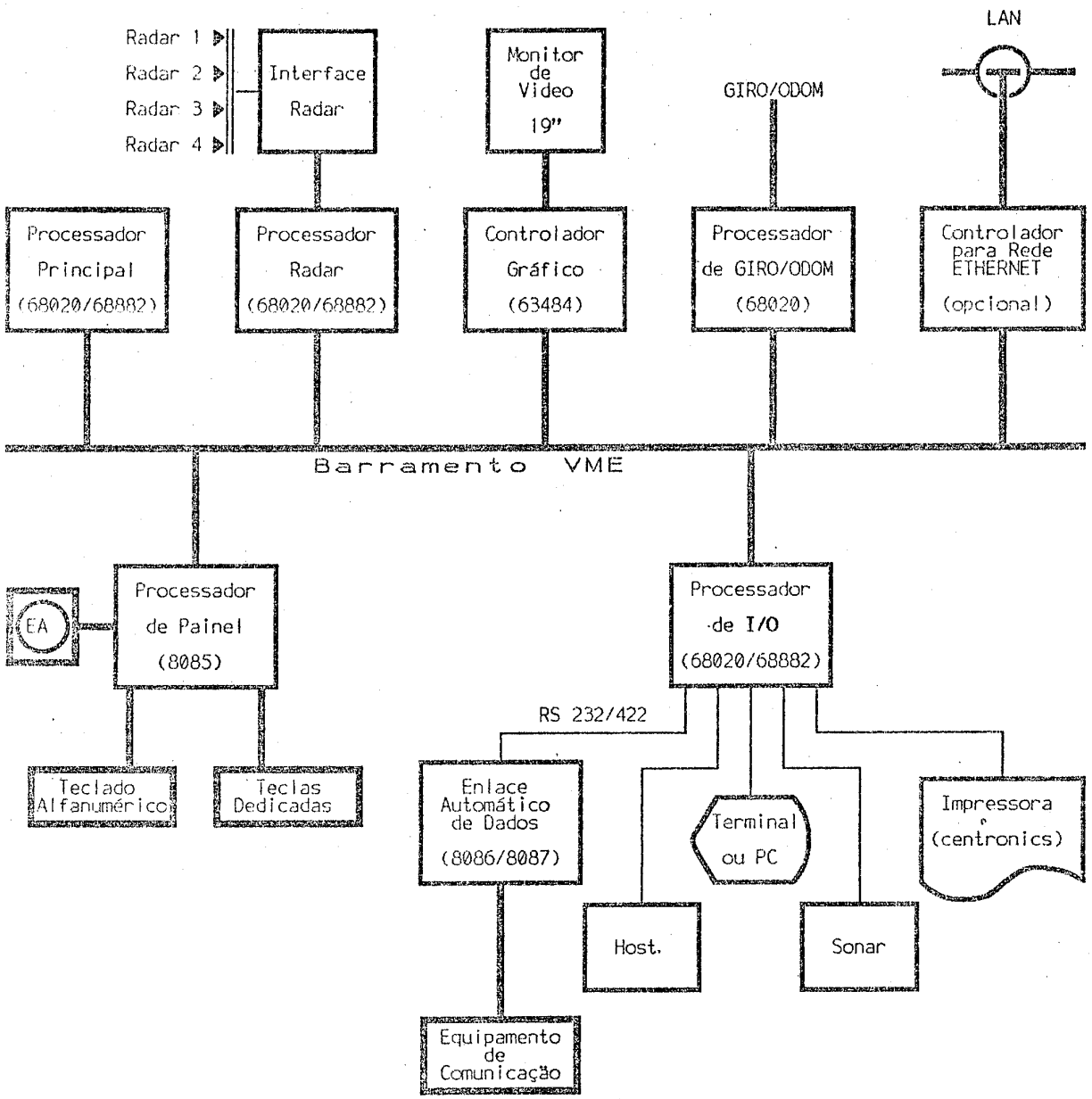


fig 1.1

O Vídeo Bruto é digitalizado no vídeo gráfico através de um sistema integrado de hardware e software específico. Qualquer tecla pressionada é tratada, em primeira instância, pelos processadores de Painel e I/O e posteriormente pelo software do TTI. Caso implique em alterações no Vídeo Bruto, é também processada pelo software do Processador Radar. Esse software não é descrito neste trabalho pois foi fornecido integrado ao hardware. Quanto ao software, objeto deste trabalho, está distribuído entre os processadores:

- **Processador Principal**

Onde são executadas as funções do TTI no que diz respeito a Interface Usuário-Máquina, cálculo de Soluções Táticas, Acompanhamentos Manuais, Pontos de Referência, etc.. Não processa dados diretamente dos sensores ligados ao TTI. Esses dados são colocados disponíveis, após tratamento preliminar, pelo processador específico deste sensor (Processador de GIRO/ODOM ou Enlace Automático de Dados).

- **Processador de GIRO / ODOM**

Responsável pelo tratamento dos dados extraídos da agulha giroscópica e do odômetro, pela validação e filtragem inicial dos valores. Este software é parte do Subsistema Básico Mov_Acomp_Nav.

- **Enlace Automático de Dados**

Controla o equipamento de comunicação e faz o processamento da rede de sistemas táticos. A ele são alocados os serviços do Subsistema Básico Ctr_EAD. No Processador Principal, é alocado um Processo responsável por fazer a conexão entre os dois processadores (EAD e Processador Principal), tratando, inclusive, das diferenças de representação de número existentes entre os do tipo Motorola 68020 e os do tipo Intel 8086/8087. O equipamento de controle do EAD já existia, antes do início do projeto.

2 - Contextualização do Método Utilizado

Durante os últimos anos foram propostas várias técnicas de Design de software. Duas filosofias de desenvolvimento fundamentam grande parte destas técnicas: Orientação a Funções [43,11,45,25] e Orientação a Objetos [4,3,6,2,47]. As duas filosofias distinguem-se pela ênfase que atribuem à modelagem de cada um dos dois componentes principais de um sistema: atividade e informação.

Os chamados **Métodos Estruturados** (Análise, Design e Programação), Orientados a Funções, têm como objetivo principal a identificação das funções que constituem o sistema.

A **Orientação a Objetos** focaliza a descoberta das estruturas de informação necessárias ao sistema. Em seguida, define as operações que atuam sobre cada estrutura - os Serviços prestados pelo sistema. Um **Objeto** é uma estrutura de informação à qual associa-se um conjunto de operações.

Muitos métodos de desenvolvimento de sistemas [45,25,11,43,22,6,18,46,32,39,2,47] baseiam-se em uma ou outra dessas duas filosofias. As propriedades principais que caracterizam os diferentes métodos são:
- a seqüência em que são apresentadas as decisões chaves;
- os princípios que os fundamentam.

Observando-se as primeiras decisões de cada método, pode-se determinar que características de qualidade pretende-se obter no produto: manutenibilidade, reusabilidade, reconfigurabilidade, desempenho, portatilidade, disponibilidade, evolução, custo, segurança (por ex.: tolerância a falhas e faltas). Estas características devem ser estabelecidas na fase de definição dos requisitos de um dado sistema e são determinantes para a escolha da metodologia apropriada ao desenvolvimento. Um único método de design, provavelmente, não será suficientemente abrangente para abordar a diversidade de objetivos no desenvolvimento de vários tipos de sistema. A maior aplicabilidade de um ou outro é obtida a partir dos requisitos de qualidade estabelecidos.

As técnicas de modelagem de Sistemas Concorrentes têm um desafio adicional: a descrição precisa do comportamento do sistema. É fundamental ao entendimento do modelo a especificação dos pontos de sincronismos entre os Processos que implementam o sistema.

Foram elaborados **Ambientes de Desenvolvimento**, hardware e software integrados, com linguagens e primitivas para apoiar o desenvolvimento de sistemas, inclusive o de Sistemas Concorrentes. Eles implementam os conceitos apresentados nos métodos, facilitando e garantindo seu uso correto. Integram as etapas de desenvolvimento visando manter a consistência entre os produtos intermediários gerados em cada uma delas.

Este capítulo contém a contextualização do método descrito em [41] com relação às duas linhas mais populares de Design. Dentro do enfoque de Orientação a Funções, é abordado o Projeto Estrutu-

rado para Sistemas de Tempo Real introduzido primeiramente por Ward e Mellor [45]. No enfoque de Orientação a Objetos são apresentados seus fundamentos, a técnica de Design de Cartões CRC (Class, Responsibilities and Collaboration) [2,47] e os "Objectcharts" [10] usados para modelar o comportamento de Classes de Objetos.

Redes de Petri [34,36] são comentadas por constituírem uma linguagem gráfica e formal, capaz de ser empregada na modelagem de sistemas sócio-técnicos de modo geral [16,17,33] e, em particular, por existirem fortes tendências de intensificar seu emprego em Análise e Design de sistemas de controle de processos industriais [15].

Dentre os Ambientes de Desenvolvimento são focalizados o ambiente CONIC [28,29,27,20,30] e sua evolução - REX [26], onde são elaborados conceitos nos quais foi baseada, em grande parte, a metodologia de Design utilizada neste trabalho.

2.1 - Projeto Estruturado para Sistemas de Tempo Real

2.1.1 - Histórico

Esta técnica foi primeiro apresentada em um trabalho de Stevens, Meyers e Constantine [43]. DeMarco [11] suplementou o Projeto Estruturado recomendando que os Diagramas de Fluxo de Dados fossem usados na Análise de Transformação e na Análise de Transação para posterior tradução em Diagramas de Estrutura.

Inicialmente, o enfoque utilizava apenas ferramentas de modelagem - Diagramas de Fluxo de Dados - que privilegiam a representação das funções do sistema. Ferramentas como Diagramas de Entidades e Relacionamentos [8], largamente utilizadas em projetos de Bancos de Dados, passaram a ser empregadas para a modelagem da informação. Ward e Mellor [45] introduziram, nos Métodos Estruturados, a Modelagem de Informação associada à Modelagem das Funções. Adicionaram também construções para desenvolvimento de sistemas de tempo real - fluxos contínuos e de controle -, depósito de eventos, atividades de controle e diagramas de estado e transição, que evidenciam o comportamento do sistema. Maffeo acrescentou construções que possibilitam modelar mais precisamente as interações entre o sistema e o ambiente externo - Esquema Semântico [25] - e a dinâmica dos eventos externos de interesse - Esquema da Dinâmica [23,9] -, o que facilita a caracterização da interação do sistema de tempo real com o ambiente externo, geralmente complexa. Acoplamento e Coesão, critérios introduzidos por Stevens, Meyers e Constantine [43], podem ser usados para avaliar a qualidade de um Design.

2.1.2 - Filosofia de Modelagem

A decomposição do sistema determina as atividades que tratam as entradas, as atividades de transformação dos dados, as atividades de controle e as atividades que tratam as saídas.

O **Projeto Estruturado** é parte de um método de modelagem que cobre todo o ciclo de vida de um sistema e que decompõe sua representação em três modelos, com graus de abstração diferentes, que detalham toda a concepção e implementação do sistema. Existe uma seqüência lógica entre esses modelos mas essa seqüência não define necessariamente a ordem cronológica mais adequada para sua construção. São eles:

- **Modelo da Essência**, onde é representado o resultado da elicitação, análise e especificação dos requisitos funcionais (independentes de qualquer alternativa de implementação) do sistema em questão;
- **Modelo da Implementação**, onde é representado o Design do sistema;
- **Modelo da Automação**, onde é representado o sistema.

As ferramentas de modelagem são semi-formais e primordialmente gráficas, com sintaxe e semântica bem definidas, que facilitam significativamente a comunicação entre pessoas. Essas ferramentas se baseiam principalmente em Diagramas de Fluxo de Dados e sua notação estendida, denominada de ESML (Extended System Modeling Language) [5,24], própria para a modelagem de sistemas de tempo-real, cuja semântica foi parcialmente formalizada em [37].

O Modelo da Essência é segmentado em dois submodelos: **Modelo do Contexto** e **Modelo do Comportamento**.

O Modelo do Contexto é composto de:

- Definição de Sistema, onde são descritos os objetivos e a operação "caixa-preta" do sistema;
- Lista de Eventos Externos, constando da identificação dos eventos externos que traduzem as necessidades das entidades que interfaceiam com o sistema;
- Esquema Transacional, que representa o contexto do sistema, fazendo uso da linguagem ESML;
- Esquema Semântico, constituído de um conjunto de hierarquias semânticas do tipo "é-um" e "é-composto", que apresenta a composição dos fluxos presentes no Esquema Transacional, relacionando-os em transações bem definidas;
- Esquema da Dinâmica, que utiliza a linguagem gráfica dos Diagramas de Estados e Transições e caracteriza a dinâmica envolvendo os eventos externos de interesse para o sistema.

O Modelo do Comportamento é composto de:

- Esquema da Memória Essencial, onde a ferramenta de modelagem é o DER (Diagrama de Entidades e Relacionamentos) e representa toda a memória do sistema (tipos de entidade, correspondentes a depósitos, seus atributos e tipos de relacionamento entre entidades);
- Esquema de Atividades que apresenta, usando ESML, uma hierarquia

de atividades, em uma organização "top-down", até o nível das atividades primitivas, isto é, as diretamente responsáveis pelo atendimento dos eventos externos. A abordagem empregada na construção do Esquema de Atividades é "outside-in";
- Esquema de Atividades Essenciais que apresenta, usando ESML, todas as atividades primitivas em um mesmo diagrama;
- Diagramas de Estados e Transições, um para cada atividade de controle primitiva, correspondente a sua especificação;
- Listas de Pré e Pós Condições, uma para cada atividade operacional primitiva, correspondente a sua especificação.

Complementa os dois modelos, um Dicionário de Dados definindo a informação presente nos esquemas que os constituem.

No que diz respeito a Sistemas Concorrentes, a concorrência fica explícita através do uso de atividades de controle e de seus Diagramas de Estados e Transições empregando comandos (Ativa, Desativa, Habilita, Inabilita) enviados às atividades operacionais.

O Modelo da Essência não contém aspectos de implementação; dito de outra forma, supõe tecnologia ideal. Tomando-o como base, é construído o Modelo da Implementação onde é efetuada a alocação de cada parte do Modelo da Essência a alguma unidade da tecnologia de implementação. As estratégias devem ser selecionadas pela sua habilidade de se aproximar do ideal, isto é, cálculos, simulações ou experimentação devem verificar se, por exemplo, o tempo de resposta de uma atividade está suficiente para sua execução satisfatória.

O Modelo da Implementação é decomposto em três submodelos:
- Modelo de Configuração de Processadores, representando uma rede de processadores, máquinas ou pessoas, que executam parte das funções do sistema, onde a concorrência real é possível pois todos os processadores podem estar ativos ao mesmo tempo;
- Modelo de Configuração de Processos, representando um conjunto de redes de Processos, uma rede por processador. Cada rede do Modelo de Configuração de Processos descreve a distribuição por processador da porção associada do Modelo da Essência. Simula concorrência entre Processos no mesmo processador;
- Modelo de Configuração de Módulos, representando, para cada Processo, o conjunto de Módulos que o implementa. Um Módulo - subrotina/procedimento - é um conjunto de instruções que é ativado como uma única unidade de controle lógico; apenas um está ativo de cada vez . Utiliza-se Diagramas de Estrutura Hierárquica como ferramenta de modelagem, evidenciando os Módulos que compõe um Processo. A transformação das redes em Diagramas de Estrutura se dá através da Análise de Transação e da Análise de Transformação [1].

Este método é independente de um Ambiente de Desenvolvimento específico (linguagens/software básico) e possui um número grande de ferramentas automatizadas (CASE) que dão suporte a seu emprego.

Durante toda a modelagem, são considerados aspectos de **Acoplamento** (medida de associação entre Módulos) e Coesão (medida de associação entre funções de um Módulo) para avaliar a qualidade dos produtos obtidos. Um bom Design deve possuir baixo Acoplamento e alta Coesão.

2.2 - Design Orientado a Objetos

Na abordagem Orientada a Objetos, a decomposição do sistema é baseada no conceito de Objeto, entidade caracterizada pelas ações que sofre e por aquelas que requer de outros Objetos.

2.2.1 - Objetos, Classes e Heranças

Objeto pode ser caracterizado como uma entidade que:
- possui estado;
- sofre ações de sua responsabilidade e requer ações da responsabilidade de outros Objetos;
- é instância de alguma classe, possivelmente anônima;
- é denotado por um nome;
- tem visibilidade restrita de/para outros Objetos;
- pode herdar característica de alguma classe.

Os princípios de Tipos Abstratos de Dados [22] e Encapsulamento de Dados [13], guiam a decomposição do sistema escondendo decisões de Design. Isto é, um Objeto encapsula uma estrutura de dados, cuja implementação é secreta para o resto do sistema. Caso um outro Objeto (Cliente), necessite acessar um valor nessa estrutura - Atributo -, deve pedir um serviço ao Objeto que a encapsula (Servidor). Os Serviços oferecidos por um Objeto constituem o conjunto de operações que atuam sobre a estrutura de dados interna. Objetos são Clientes para alguns Objetos e Servidores para outros.

A comunicação entre Objetos é realizada através de mensagens. A transmissão de uma mensagem indica um pedido de Serviço ao Objeto receptor. A mensagem codifica esse pedido e transporta informação adicional (argumentos) necessária à execução.

A comunicação entre dois Objetos - um como Cliente e o outro como Servidor -, possui as seguintes características:
- O padrão de conexão é de muitos para um;
- O provedor do Serviço (Servidor) é explicitamente mencionado pelo Cliente;
- O nome do Cliente não é conhecido do Servidor;
- O Cliente não necessita conhecer a forma como é executado o Serviço;
- O fluxo de informação pode ser bidirecional.

O comportamento de um Objeto é definido pela Classe à qual pertence Como todos os Objetos são instâncias de uma Classe, o Serviço executado por um Objeto, em resposta à chegada de uma mensagem, é determinado pela Classe do receptor. Todos os Objetos de uma dada Classe executam o mesmo Serviço em resposta a mensagens

similares.

Herança é a relação entre Classes que permite que uma nova Classe seja definida como a especialização de outra. Podem ser organizadas dentro de uma estrutura hierárquica de Heranças. Uma Subclasse pode herdar Atributos e Serviços de alguma Superclasse, uma Classe em nível mais alto na hierarquia.

Na Orientação a Objetos é importante também o conceito de Polimorfismo. Objeto polimorfo é uma entidade que pode armazenar valores de tipos diferentes durante a execução de um dos seus Serviços. O poder do Polimorfismo é permitir que algoritmos de alto nível sejam reusados repetidamente por diferentes abstrações de mais baixo nível. Pode ocorrer como um resultado da relação de Herança, sem que essa relação implique restrições ao seu emprego.

2.2.2 - Filosofia de Design

O segredo de um bom Design Orientado a Objetos é o estabelecimento de quem é responsável (Objeto) pela execução de cada ação. Uma técnica [2,47] utiliza a separação de responsabilidades como base para a definição de Classes. Existe a necessidade de associar responsabilidades a cada Classe e para tal são usados cartões de índice que representam as Classes individuais. Esses cartões são conhecidos como CRC, já que são divididos em três componentes: Classe, Responsabilidade e Colaboração.

Cada Classe no Design será descrita por um cartão CRC separado. Durante esta fase a distinção entre Classes e instâncias é obscura. Cada cartão apresenta o nome da Classe, escolhido conforme as linhas mestras fornecidas por Keller [19]. Imediatamente abaixo, são listadas as Responsabilidades que descrevem o problema a ser resolvido, representando os Serviços prestados pelo Objeto pertencente a esta Classe. O terceiro campo do cartão diz respeito a Colaboradores, já que, poucos Objetos podem fazer seus Serviços sem colaboração de outros. Normalmente, um Objeto possue relação com muitos outros Objetos, como Cliente ou como Servidor. A lista de Colaboradores deve conter todas as Classes provedoras de Serviços necessários para que a Classe sendo descrita cumpra suas Responsabilidades e todas aquelas que requisitam Serviços a esta Classe. Para fazer a conexão entre os dados e os Serviços a eles associados, é adicionado um quarto campo ao cartão CRC, que descreve os dados gerenciados pela Classe que está sendo definida.

Projetando com CRC, progride-se do conhecido para o desconhecido. O Design deve começar com as Classes mais óbvias e as necessárias para tratar o início da aplicação. Os projetistas trabalham simulando cenários de funcionamento esperado do sistema, empregando uma técnica de Design experimental ou de refinamentos sucessivos. Para cada estrutura representativa, as Responsabilidades são examinadas. São geradas especificações mais detalhadas visando descrever o que cada Classe deve fazer e como, identificando-se também quais os dados estritamente necessários a cada uma delas. Resulta na descoberta de novas Responsabilidades, pois uma Classe

pode necessitar dados de outra Classe.

Em um segundo estágio do Design duas relações importantes devem ser detalhadas: a relação "é-um" e a relação "é-parte". A relação "é-um" define a hierarquia de Subclasses - Herança -, enquanto a relação "é-parte" descreve Classes compostas por outras Classes.

Critérios de qualidade são empregados para validação do resultado do Design. Deve-se evitar:
- Classes que fazem modificações diretamente em outras Classes;
- Classes com muita Responsabilidade;
- Classes sem Responsabilidade;
- Classes com Responsabilidade não utilizada;
- Nomes de Classes não significativos de suas Responsabilidades;
- Responsabilidades não correlatas em uma mesma Classe;
- Herança não apropriada, quando a Classe não pode herdar nenhum comportamento apropriado da Superclasse;
- Funcionalidade repetida em uma ou mais Classes ao invés de associada a uma Superclasse.

2.2.3 - Descrição do Comportamento do Sistema

A descrição de um sistema utilizando a filosofia de Orientação a Objetos, deve capturar as instâncias das Classes que constituem o sistema e sua intercomunicação. Podem ser usados Diagramas de Configuração [31], que tentam estabelecer a relação Cliente-Servidor entre Classes. Como segundo passo deve-se especificar o comportamento de cada Classe, o que é normalmente feito através de máquinas de estado com ações imperativas.

"Objectcharts", introduzidos em [10], são diagramas que caracterizam o comportamento de Classes de Objetos como uma máquina de estados com entradas, significando as operações providas pela Classe, e saídas representando as operações que a Classe requer. São "Statecharts" [14] estendidos, onde o efeito das transições de estado nos Atributos é especificado. "Statecharts", por sua vez, estendem a notação de máquinas de estado finitas [12], produzindo uma descrição de estrutura modular e hierárquica; as extensões relevantes são: Hierarquia e Concorrência. Hierarquia permite que transições comuns de estados sejam concentradas (conjuntos de estados). Concorrência é representada pela decomposição do tipo "E" de estados. Os estados representados em um "Statechart" representam os diversos estados em que um Objeto pode estar. As transições são rotuladas com:
- Serviços de mudança de estado oferecidos pela Classe;
- Serviços requeridos a outros Objetos.

O comportamento total do sistemas é descrito por Diagramas de Configuração e o conjunto dos "Objectcharts", um para cada classe presente nesses diagramas.

Em desenvolvimento Orientado a Objetos é apropriado o emprego de Linguagens de Programação Orientadas a Objetos, adequadas à

definição de Classes de Objetos, tais como Ada, C++, SMALLTALK etc.

2.3 - Redes de Petri

Redes de Petri [16,17,38,33,15] podem ser empregadas como ferramenta para modelagem conceitual bem como para modelagem da implementação de sistemas.

Foram inventadas por C. A. Petri, na sua tese de doutorado - Comunicação com Autômata [34] -, objetivando a modelagem de Sistemas Concorrentes.

É representada através de um gráfico, composto de dois tipos de elementos: Lugares (círculos) e Conexões (retângulos) entre Lugares. As Conexões estão ligadas por Ramos (setas) aos círculos representativos dos Lugares conectados. Os Ramos são classificados em:

- Ramos alteradores de entrada/saída: aqueles que designam Alteração de estado dos Lugares de entrada/saida na ocorrência de uma transição representada por uma Conexão que ligue esses Lugares;
- Ramos restauradores de entrada/saída: aqueles que designam "consulta" a Lugares de entrada/saída cujo estado pode habilitar a ocorrência de uma Alteração. Durante esta ocorrência, o estado do Lugar em questão é alterado, sendo restaurada a situação inicial antes do seu final. Isto implica, em caso de Alterações em conflito, a não ocorrência das Alterações concorrentemente;
- Ramos mantenedores de entrada/saída: aqueles que designam consulta a Lugares de entrada/saída cujo estado pode habilitar a ocorrência de uma Alteração. Analogamente aos Ramos restauradores, eles especificam a não alteração do estado do Lugar como conseqüência de uma Alteração. Mas, contrariamente aos Ramos restauradores, o estado do Lugar não é alterado durante a ocorrência de uma Alteração, o que permite seu uso concorrente por várias Alterações.

Sistemas podem ser modelados com redes de Petri de acordo com duas abordagens diferentes:

- Abordagem de Fenômenos, onde o comportamento de qualquer sistema pode ser descrito por um conjunto de processos. Cada processo é descrito em termos de fenômenos classificado de dois tipos: estado (elemento passivo) e transição (elemento ativo). Os fenômenos são históricos, isto é, uma vez acontecidos não se repetem. Portanto, uma rede de fenômenos, mesmo para sistemas simples, pode ter proporção exagerada, até mesmo infinita. Os fenômenos do tipo estado representam estados locais e os fenômenos do tipo transição, transição entre estados locais. Uma rede que modele um processo pode expressar dois tipos de relações entre fenômenos: a seqüência e a concorrência. Dois processos associados a um mesmo sistema expressam uma terceira relação, opção, que incorpora à modelagem a noção de indeterminismo.
- Abordagem Condição/Evento, onde é feita uma abstração que classifica fenômenos equivalentes. Fenômenos do tipo estado são classificados como condição, os do tipo transição como eventos.

As redes condição/evento podem ser de dois tipos:

- Redes Elementares, onde a cada Lugar só pode estar associada uma Marca: a presença da Marca no Lugar significando que vigora a condição por ele modelada. Caso todos os Lugares de entrada de uma Conexão estejam Marcados e todos os Lugares de saída não Marcados, então a Alteração definida pela Conexão pode ocorrer (ao ocorrer, o evento altera a Marcação dos Lugares ligados à Conexão por Ramos alteradores);
- Redes Compactas, onde cada Lugar define um conjunto de Marcas, cada Conexão define um conjunto de Alterações e as regras de evolução das Marcações estão subordinadas a anotações formais em Lugares, Conexões e Ramos.

Quase sempre as redes são complementadas com textos, legendas explicativas, cuja função é facilitar seu entendimento. No entanto, do ponto de vista formal, podem ser compreendidas sem essas legendas.

Redes de Petri possuem regras de execução formais sendo capazes de modelar um sistema de forma precisa. Entretanto, a complexidade das redes resultantes da modelagem ainda constitui uma dificuldade para seu amplo emprego. Existem ferramentas CASE (por exemplo: MetaDesign e Design/CPN da Meta Software Corporation), que visam facilitar a construção de Redes de Petri elementares e compactas e possibilitam sua execução.

2.4 - CONIC/REX (Reconfigurable and Extensive Parallel and Distributed System)

Esta abordagem é baseada em linguagem para construção de aplicações distribuídas, combinando as características de uma Linguagem de Programação com a flexibilidade de um sistema operacional. Configurações flexíveis, modularidade e reusabilidade dos componentes do software são facilitados pela utilização de uma Linguagem de Programação para os Processos ("programming in small") e de uma Linguagem para a Configuração de programas a partir de Módulos predefinidos ("programming in large") [35].

O ambiente REX [26] - Reconfigurable and Extensible Parallel and Distributed Systems, constitui uma evolução do ambiente CONIC onde maiores facilidades foram incorporadas para agilizar o curso do desenvolvimento. Está sendo desenvolvido para o projeto ESPRIT II, por empresas européias em parceria com pesquisadores de universidades, iniciado em 1989.

O sistema implementado com REX é formado por um conjunto de Instâncias de Componentes interconectados. Componentes são Classes de Objetos que possuem estado e Interfaces bem definidas e especificadas pela Linguagem de Especificação de Interfaces (ISL). A linguagem ISL define os tipos de dados que são transferidos através dos pontos de Serviço - Portas.

A funcionalidade de um Componente pode ser implementada por diversas Linguagens de Programação. ISL fornece uma forma comum de definir tipos entre Componentes, independentemente das características diferentes de cada linguagem.

Instâncias de Componentes executam em paralelo. O conjunto de Componentes de um sistema, junto com suas interconexões, são especificados por uma Linguagem de Configuração - Darwin -, que descreve a estrutura geral do sistema. Permite uma estruturação hierárquica - especificação de Componentes Estruturados. Um Componente Estruturado tem como características:
- Sua Interface (Dados e Portas) é igual à composição das Interfaces dos Componentes que o constituem. Essas Interfaces são especificadas usando-se ISL;
- Pode ser substituído por estes, sem afetar o resto da estrutura do sistema.

Esse tipo de composição pode repetir-se recursivamente em vários níveis; o sistema é o Componente de nível mais alto. É possível também a alteração dinâmica da configuração de Componentes.

Para possibilitar a reusabilidade, Componentes são independentes de contexto e, portanto, especificam internamente sua Interface, por onde são ativados/solicitados Serviços.

As Portas podem ser complexas, como Conjuntos e "Arrays" de Portas. Conjuntos de Portas são equivalentes a estruturas por onde mais de um Serviço diferente pode ser ativado. "Arrays" são usados para facilitar a comunicação com Componentes que possuem mais de uma Instância na configuração do sistema.

As primitivas de comunicação entre Componentes, Call e Accept, funcionam de forma assíncrona. Extensões destas primitivas, como:
- Colocar o Processo em estado de espera pela resposta a um Serviço anteriormente pedido;
- Possibilitar a espera por uma resposta por um tempo ou evento prédeterminado;

permitem a comunicação síncrona e tratamento de falha (por exemplo: mau funcionamento no nó onde executa o Componente que presta o Serviço).

O ambiente REX pretende fornecer ferramentas de Especificação, Design e construção de sistemas distribuídos, integradas ao Ambiente de Desenvolvimento. Ferramentas de Análise e Design serão construídas mantendo-se sua descrição incluída na estrutura do sistema conforme definida com a Linguagem de Configuração.

3 - Metodologia de Design Empregada no TTI

Neste capítulo é apresentada a sistemática utilizada no Design do TTI. Esta sistemática, como um método de engenharia, equivale a princípios e normas nas quais se baseiam a construção do software. Utiliza técnicas de abstração que permitem reduzir a complexidade, escondendo informações ou agregando informações similares [21]. Determina a forma, a ser empregada pelos projetistas para registrar as decisões, facilitando a futura manutenção do sistema.

3.1 - Conceitos de Design

3.1.1 - Características de Qualidade

Quatro métricas foram consideradas relevantes na avaliação da qualidade de sistemas: Modularidade, Manutenibilidade, Portatilidade e Reusabilidade. Esta técnica de Design se fundamenta em obter, por construção, sistemas com alto grau para essas métricas.

A Modularidade mede a capacidade de alteração em um dos subsistemas, que compõem o sistema, com o mínimo de impacto para os outros subsistemas. Alta Modularidade aumenta a clareza do Design e facilita a implementação, teste, documentação e manutenção. Possibilita o acréscimo de novas funções, modificação ou substituição de algumas das antigas, sem influências significativas nas demais funções do sistema.

A Manutenibilidade representa o grau de facilidade com que se mantém o sistema ou um subsistema. Pode ser expressa em termos de outras medidas de qualidade de sistemas como: simplicidade das soluções, clareza das decisões, Modularidade e uma boa documentação (de código fonte e de suporte). Alto grau de Manutenibilidade facilita o entendimento do sistema por pessoas externas à equipe de desenvolvimento, da sua concepção e da documentação pertinente. Reduz problemas decorrentes de mudanças na equipe e agiliza correções de erros.

A Portatilidade indica o poder de transferência do software para outro computador ou ambiente. Alto grau de Portatilidade permite a migração do software, com pequenas alterações, para uma nova Arquitetura de Hardware, com o objetivo de melhor desempenho ou manutenção mais barata.

A Reusabilidade é a medida de quanto um subsistema pode ser usado em outros sistemas. O desenvolvimento de um subsistema reusável empreende um esforço maior do que um não reusável. Seu reuso em outro sistema também implica em esforço adicional para compreender a sua especificação e decisões de Design. Apesar disso, em geral, o emprego de subsistemas reusáveis reduz o tempo de desenvolvimento [35], o que ocasiona também redução de custos.

3.1.2 - Modelo de Implementação Resultante

Para assegurar um alto grau de Modularidade do sistema, este deve ser construído a partir de subsistemas independentes, com serviços e interfaces bem definidos. Para garantir boa Manutenibilidade, além de definir uma política de modularização, deve-se definir uma política de documentação, (padrão de documentos, sintaxe e semântica das linguagens de representação empregadas) bem como uma organização modular desta documentação. A construção de um sistema portátil, requer um projeto o mais abstraído possível de qualquer Arquitetura de Hardware e Software Básico. Para alcançar o objetivo de alta Reusabilidade, cada sistema deve ser construído com base em subsistemas reusáveis. O grau de reusabilidade de um subsistema é função de padrões de implementação (por exemplo, parametrização) e dos tipos de sistemas desenvolvidos por uma empresa. Analisando-os, determina-se os critérios que possibilitam gerar um conjunto de subsistemas reusáveis em outros sistemas.

A estratégia empregada para atingir esses objetivos divide o Modelo de Implementação em duas etapas: o Projeto Básico, que apresenta, em diversos níveis de abstração, a decomposição do sistema até a determinação de um conjunto de subsistemas com características de Subsistemas Básicos, e a Especificação detalhada de cada Subsistema Básico. Com o Projeto Básico é possível entender o funcionamento do sistema como um todo e as opções de Design que levam a um dado particionamento do sistema em subsistemas.

Os Subsistemas Básicos são Unidades de Implementação e Unidades de Reutilização. Cada Subsistema Básico é uma unidade de código fonte desenvolvida e mantida por uma equipe pequena e indivisível quando reutilizada. A separação da documentação do Design do sistema como um todo, da Especificação de cada Subsistema Básico, facilita a divisão do trabalho em equipes, responsáveis por um ou mais Subsistemas Básicos, bem como a identificação dos Subsistemas Básicos já desenvolvidos, que podem ser reutilizados.

A concepção de subsistemas deve refletir um alto grau de Reusabilidade. Estes devem ter apenas conhecimento de si próprio, isto é, não apresentam referências a outros subsistemas. Visando essa independência, utiliza-se a comunicação entre subsistemas por envio/recebimento de mensagens em Portas de dados; as Portas são partes dos subsistemas. A cada subsistema está alocado uma fração dos serviços prestados pelo sistema, muitas vezes concorrentes. Um serviço concorrente está associado a um Processo, de execução periódica ou ativado por um pedido de serviço - recebimento de uma mensagem através de uma Porta de entrada. Essas mensagens são transportadas por Interfaces que conectam Portas de subsistemas.

Subsistemas prontos e testados precisam ser integrados, formando o sistema e, para tal, as Interfaces devem ser implementadas e os Processos que compõem cada Subsistema Básico precisam ser mapeados na Arquitetura de Hardware - Configuração do Sistema. Para respeitar o requisito de alta Portatilidade, a Configuração do Sistema não deve ser incorporada a cada Subsistema Básico e

sim, feita através de uma Linguagem de Configuração com a função de implementar as Interfaces dos subsistemas, verificar a consistência das Portas conectadas (Linguagem de Configuração de Sistemas) e mapear os Processos de cada Subsistema Básico na Arquitetura de Hardware (Linguagem de Configuração de Hardware). A Configuração do Sistema é um módulo de software independente, o que facilita alterações de Arquitetura de Hardware e Software.

3.1.3 - Conceitos de Design - Resumo

- Um sistema é um conjunto de Subsistemas Básicos;
- Cada Subsistema Básico oferece e solicita Serviços através de suas Portas de entrada/saída;
- Essas Portas são conectadas a Portas de outros Subsistemas Básicos através de Interfaces;
- Interfaces são implementadas utilizando-se uma Linguagem de Configuração de Software;
- A cada Porta de entrada de um Subsistema Básico está associado um Processo;
- Os Processos de cada Subsistema Básico são mapeados no Hardware através de uma Linguagem de Configuração de Hardware;

A seguir, são apresentados os conceitos utilizados na construção do Projeto Básico e os resultados esperados no decorrer do Design de um sistema.

3.2 - Projeto Básico

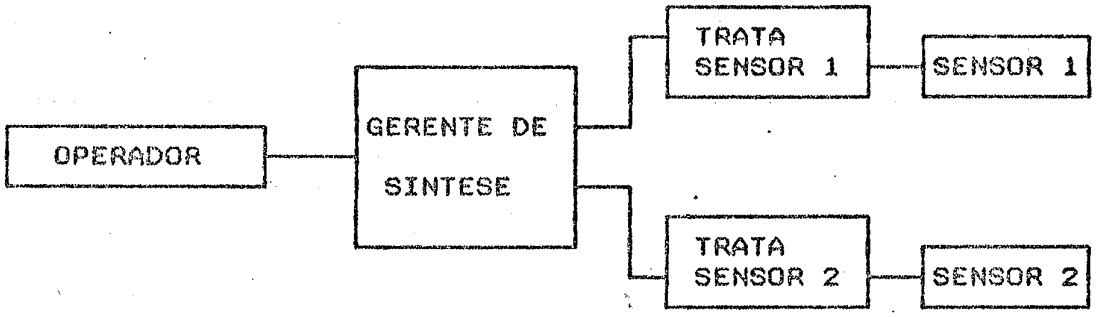
3.2.1 - Decomposição do Sistema

As necessidades do usuário podem ser expressas através dos Serviços a serem oferecidos pelo sistema. Um Serviço é definido como uma operação que atua sobre uma estrutura de dados interna ao sistema. O conjunto de Serviços sobre uma dada estrutura constitui um subsistema usado na construção do sistema. Para um subsistema realizar um Serviço, freqüentemente, necessita requisitar Serviços oferecidos por outros subsistemas. Eles assim podem ser vistos como unidades prestadoras/solicitantes de Serviços, que integrados compõem o sistema.

Os Serviços de cada subsistema podem ser:
- Serviços Externos: são Serviços executados, a partir de um pedido, realizado por outro subsistema ou por entidade externa ao sistema;
- Serviços Internos: são Serviços iniciados a partir de condições limites ou do decorrer de um período de tempo, executados independentemente de pedidos.

Como exemplo, considere um sistema que processa dados provenientes de dois sensores e os apresenta a um operador. Ele teria a seguinte figura de contexto:

Observando-se os Serviços prestados pelo sistema:
- Processar informações do sensor 1;
- Processar informações do sensor 2;
- Sintetizar e apresentar as informações processadas ao operador;
é possível concluir, de forma natural, a seguinte decomposição do sistema com alto grau de Modularidade:



onde os subsistemas TRATA SENSOR 1 e TRATA SENSOR 2 prestam os Serviços de processar informações do sensor 1 e 2, respectivamente. O subsistema GERENTE DE SINTESE, periodicamente, solicita um Serviço a esses subsistemas, que fornece, como resposta, as informações processadas dos dois sensores. Ao receber essas informações executa o Serviço de síntese e apresentação ao operador. O subsistema GERENTE DE SINTESE oferece dois tipos de Serviços: um Serviço Externo para responder a comandos do operador e outro Interno, periódico, ao buscar informações dos sensores e apresentá-las ao operador. Os subsistemas TRATA SENSOR 1 e 2 oferecem um Serviço Interno que processa ciclicamente as informações provenientes dos sensores e um Serviço Externo que atende a solicitação do GERENTE DE SINTESE.

Esta técnica tenta dominar a complexidade de um sistema, segmentando-o em subsistemas mais simples, interrelacionados. A cada subsistema são alocados parte dos Serviços prestados pelo sistema e Serviços necessários ao relacionamento de subsistemas. Em um primeiro estágio, a segmentação de um sistema pode apresentar, ainda, subsistemas complexos. A segmentação deve ser repetida recursivamente, alocando os Serviços a subsistemas, decompondo Serviços complexos em Serviços mais simples e realocando-os a novos subsistemas de nível mais baixo, até que cada um dos subsistemas resultantes possa ser considerado um Subsistema Básico. Na decomposição de um sistema deve-se observar:
- Um Serviço só pode aparecer em um único subsistema. Caso algum

outro necessite desse Serviço, deve pedí-lo ao subsistema ao qual está alocado;
- Os dados são parte integrante dos subsistemas e o acesso é feito através dos Serviços oferecidos pelos subsistemas. Os diversos níveis de decomposição não devem apresentar estruturas de dados compartilhadas entre subsistemas;
- O nível onde aparecem os Subsistemas Básicos é o último nível onde ainda se consegue manter os dados encapsulados.

O conceito aqui empregado é o do Encapsulamento de Dados da Engenharia de Software, primordialmente criado para permitir modularidade acoplada ao desenvolvimento autônomo de módulos ("Information Hiding") [13]. A Estrutura de Dados é especificada pelas operações válidas sobre essa estrutura - Serviços prestados pelo subsistema que a encapsula. Este conceito pode ser empregado nos diversos níveis de decomposição do sistema em subsistemas.

3.2.2 - Interação entre Subsistemas

A interação entre subsistemas, para todos os níveis de decomposição, está baseada nos conceitos de Portas e Interfaces. As Portas podem ser dos seguintes tipos: entrada, saída, entrada com resposta e saída com resposta. Utiliza-se também Portas agregadas, compostas por tipos diferentes de Portas, recurso empregado nos níveis mais altos de decomposição. Uma Porta de saída de um subsistema A deve ser ligada a uma Porta de entrada de um subsistema B. Neste caso, A pede um Serviço a B.

A chegada de uma mensagem a uma Porta de entrada corresponde a um pedido de Serviço. A mensagem transporta os dados necessários à execução do Serviço que requisita. Os dados, equivalentes às respostas resultantes da execução de um Serviço, são carregados por uma mensagem de saída. A comunicação e o sincronismo entre subsistemas é totalmente realizada através de Portas correspondentes a pedidos/respostas a Serviços.

Duas Portas somente podem ser conectadas na definição de uma Interface, caso possuam os mesmos tipos de dados, no sentido inverso de fluxo. A Interface é responsável por transportar o pedido de Serviço de um subsistema A, feito através do envio de mensagem a uma de suas Portas de saída, para a Porta de entrada no subsistema B, conectada a essa Porta de saída.

A toda Porta de entrada (simples ou com resposta associada) de um Subsistema Básico deve estar associado pelo menos uma instância de Processo. Isso devido ao fato de que Serviços independentes prestados por um subsistema são concorrentes. Já várias Portas de saída (simples ou com resposta associada) podem estar acopladas a um único Processo, pois, para executar o Serviço, pode requisitar Serviços prestados por outros Processos.

Serviços ativados através de Portas de entrada simples, em princípio, não causam espera nos Processos que os requisitaram. Ficará a cargo da Interface depositar a mensagem na Porta de

entrada do subsistema responsável pela execução do Serviço, dispondo de um certo enfileiramento de mensagens. O subsistema requisitante ficará em espera apenas em caso de não existir espaço de armazenamento da mensagem, para posterior processamento do Serviço.

Serviços ativados através de uma Porta de entrada com resposta associada, devem ao final de sua execução produzir uma informação de saída correspondente à resposta. Este tipo de ligação assemelha-se ao Remote Procedure Call das Linguagens de Programação. O Processo que requisita o Serviço fica suspenso esperando a resposta, no entanto, a requisição da resposta pode ser explícita. Ocorre da seguinte forma:

- Uma Porta P1 de entrada/saída de um subsistema A deve ser ligada a uma Porta P2 de saída/entrada de um subsistema B.
- O subsistema B pede um Serviço ao subsistema A, colocando uma mensagem na Porta P2, que será entregue na Porta P1 (pelo sistema operacional). O Processo de B que trata a Porta P2 pode continuar executando.
- Ao necessitar da resposta, o Processo de B busca receber uma mensagem na Porta P2. Caso essa mensagem, correspondente a resposta, ainda não esteja disponível, esse Processo é suspenso, esperando por resposta.
- O subsistema A executa o Serviço correspondente a chegada desta mensagem na Porta P1 e coloca a mensagem que transporta a resposta (saída) na Porta P1, que será entregue na Porta P2 de B. Nesse instante, o Processo que trata a Porta P2, caso esteja esperando, volta a estar pronto para executar.
- O processo do subsistema A, acoplado a Porta P1, volta a esperar a chegada de uma mensagem. Enquanto isso não ocorrer, esse Processo está suspenso.

3.3 - Modelagem do Projeto Básico

Para o Projeto Básico é usada uma linguagem diagramática hierárquica (Diagramas de Estrutura - DE), apoiada por uma representação de controle (Diagramas de Comportamento - DC), também hierárquica, que possibilitam a visualização da estrutura e da concorrência entre subsistemas. São modeladas que Operações são executadas pelo sistema, através de seus subsistemas, e que Dados, produzidos por seus subsistemas, trafegam nas mensagens. O Controle é modelado através do uso de DC's, que evidenciam a concorrência entre os Serviços prestados pelos diversos subsistemas que constituem um DE.

O registro das decisões de Design é representado por pares (DE, DC), um para cada nível da estrutura de decomposição, desde a visão do contexto do sistema (neste caso sem DC) até níveis onde só existem Subsistemas Básicos. O uso de Diagrama de Comportamento associado a cada Diagrama de Estrutura permite um domínio eficiente de complexidade, e facilita o entendimento global do comportamento do sistema [31]. O conjunto de DE's e DC's constitui uma estrutura hierárquica, "top-down", de decomposição até o nível de Subsistemas Básicos.

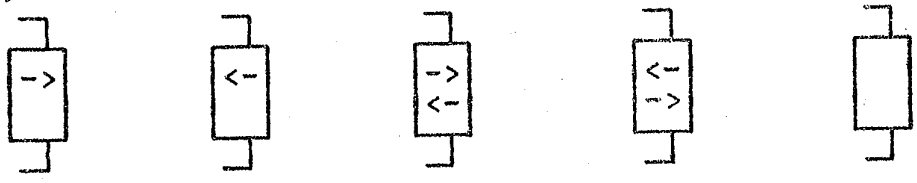
Acrescenta-se a cada DE uma lista de Serviços executados por cada subsistema que o compõe. Para facilitar a visualização da interligação entre Subsistemas Básicos, é incorporado um DE contendo apenas esses subsistemas e suas Interfaces. Para explicitar a estrutura hierárquica de decomposição, complementa-se o Projeto Básico com uma árvore de subsistemas. Um Dicionário de Dados define os dados e pontos de controle presentes nos diagramas, representativos do conteúdo das mensagens e de estados de execução (ver ítem 3.3.2) respectivamente.

3.3.1 - Diagramas de Estrutura

Um DE apresenta a forma de decomposição do Sistema em um conjunto de subsistemas, cada qual com uma parte dos Serviços prestado ou requisitado pelo Sistema. Portas, associadas a fluxos de entrada e saída do sistema são também usados por esses subsistemas, restando acrescentar Portas e fluxos de dados necessários ao relacionamento entre os subsistemas que compõem o DE.

Os Diagramas de Estrutura modelam operações e dados e são constituídos pelos seguintes elementos básicos:

- Subsistema, representado 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 que 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. Evidencia a Interface entre subsistemas.
- 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 é possível ter fluxos bidirecionados em Portas de entrada com resposta 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, geralmente presentes em níveis de representação mais abstratos, representam a agregação de portas de tipos diferentes e, portanto, devem ser representadas sem setas.



Porta de saída

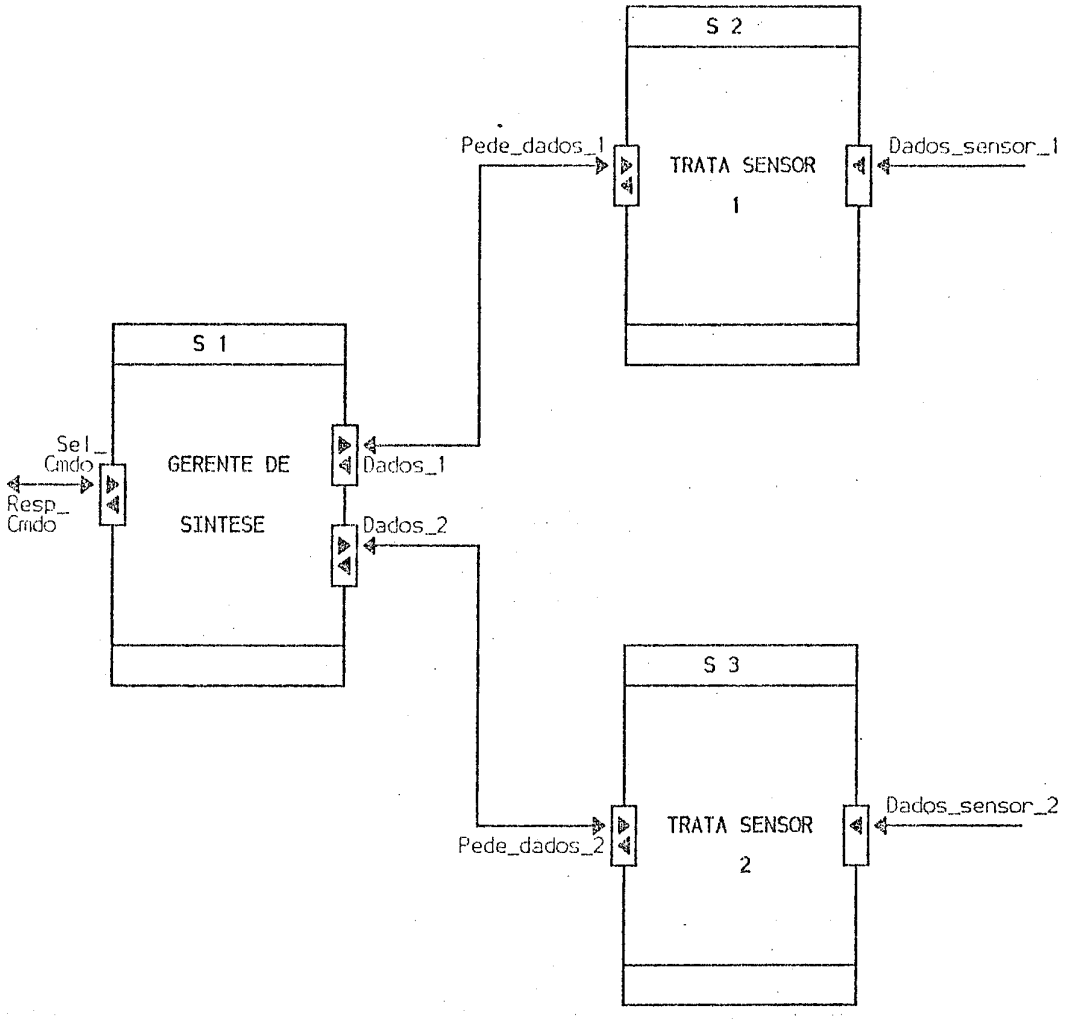
Porta de entrada

Porta de saída c/ resposta

Porta de entrada c/ resposta

Porta agregada c/ diversos tipos de serviço

DE - Exemplo



Comentarios:

- Este DE possui tres subsistemas. Pelo campo superior dos retangulos que os representam, verifica-se que dizem respeito a primeira decomposicao do sistema.
- O Subsistema 1 possui tres Portas, uma de entrada com resposta associada e duas de saida com resposta associada.
- O Subsistema 1 pede a execucao de um Servico pela porta de saida associada ao dado Pede_dados_1. O Processo associado a esse subsistema tem sua execucao suspensa ao tentar receber a resposta Dados_1, quando esta ainda nao estiver disponivel.
- Essa Porta esta conectada a uma Porta de entrada com resposta associada. O Processo correspondente e ativado pela chegada da mensagem que transporta Pede_Dados_1 e, como resultado, retorna a resposta Dados_1.
- Os Subsistemas 2 e 3 possuem uma porta apenas de entrada, ligada a uma entidade externa. Representa que, por pedido ou periodicamente, essa informacao e acessada ativando a execucao de um servico.

3.3.2 - Diagramas de Comportamento

Os Diagramas de Comportamento apresentam os Serviços de cada subsistema e os Pontos de Controle, que estabelecem a precedência/concorrência entre Ações dos subsistemas (execução total ou parcial de um Serviço). Os Pontos de Controle referem-se a um acontecimento num dado instante de tempo que possui a propriedade de ser memorizado, isto é, um estado que prevaleceu em algum instante anterior. Ao ser usado é consumido.

Os DC's modelam operações, dados e controle, através de diagramas com a seguinte simbologia:

- Subsistema/Ação, representado por um círculo, com uma barra horizontal dividindo-o. Na parte superior aparece o nome do subsistema responsável pela Ação, especificada na parte inferior.
- Ponto de Controle, representado por uma Barra.
- Descritor, texto associado a um Ponto de Controle, correspondente a um fluxo de dados ou estado de execução (por exemplo, decorrido tempo).
- Controle representa uma regra de composição/decomposição de Pontos de Controle. Faz uso de conectores análogos aos utilizados em Lógica. São eles:
  - x - análogo a E
  - * - análogo a Ou inclusivo
  - + - análogo a Ou exclusivo

Quando um Ponto de Controle ocorre entre ações de subsistemas diferentes, seu Descritor está relacionado ao um fluxo de dado entre esses subsistemas, significando a ocorrência da mensagem que transporta esse dado. Quando ocorre entre ações diferentes de um mesmo subsistema, seu Descritor equivale a um fluxo de dado ou estado interno a decomposição desse subsistema. Utiliza-se o Descritor Feito para indicar o Ponto de Controle equivalente ao final de uma Ação.

Na composição, dois ou mais Pontos de Controle são ligados por arcos direcionados de entrada ao Ponto de Controle composto. Significa que esse estado só é verdadeiro se aqueles que o compõem ocorrerem segundo a regra de composição. Na decomposição, um Ponto de Controle é ligado a dois ou mais Pontos, nos quais se decompõe segundo suas regras de decomposição, por arcos direcionados de saída. Só é permitida uma regra de composição/decomposição por Ponto de Controle mas é possível o encadeamento dessas regras. Por exemplo, um Ponto de Controle se decompõe em dois, segundo um E (a ocorrência dos dois é simultânea) e um desses se decompõe em outros dois, segundo um OU Exclusivo (ocorrem apenas um dos dois últimos Pontos de Controle resultantes).

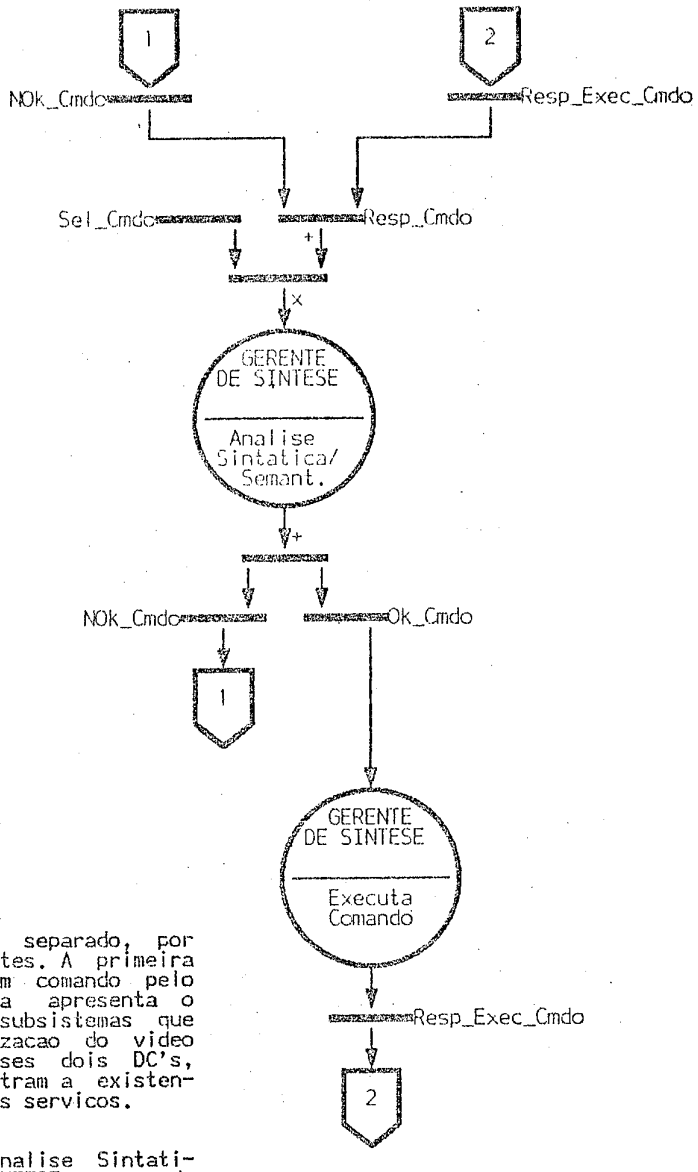
A execução de uma Ação de um subsistema representado no DC só acontece quando o Ponto de Controle de entrada, respeitando sua regra de composição, é verdadeiro. O resultado desta execução dá origem ao Ponto de Controle de saída deste Subsistema/Ação, decomposto, segundo suas regras de decomposição. Essas regras explicitadas nos DC's através dos Controles, fazem parte do código responsável por implementar a Ação dos Subsistemas Básicos que esses

Pontos de Controle ativam/resultam, já que representam o padrão de comportamento do Subsistema Básico.

Apesar de nos DC's existir a mistura de dados, operação e controle em um mesmo diagrama, sua construção e leitura é simplificada já que são segmentados através da hierarquia correspondente aos diversos níveis de particionamento do sistema em subsistemas, apresentados nos DE's.

DC - Exemplo

Tratamento de Comando



Comentarios:

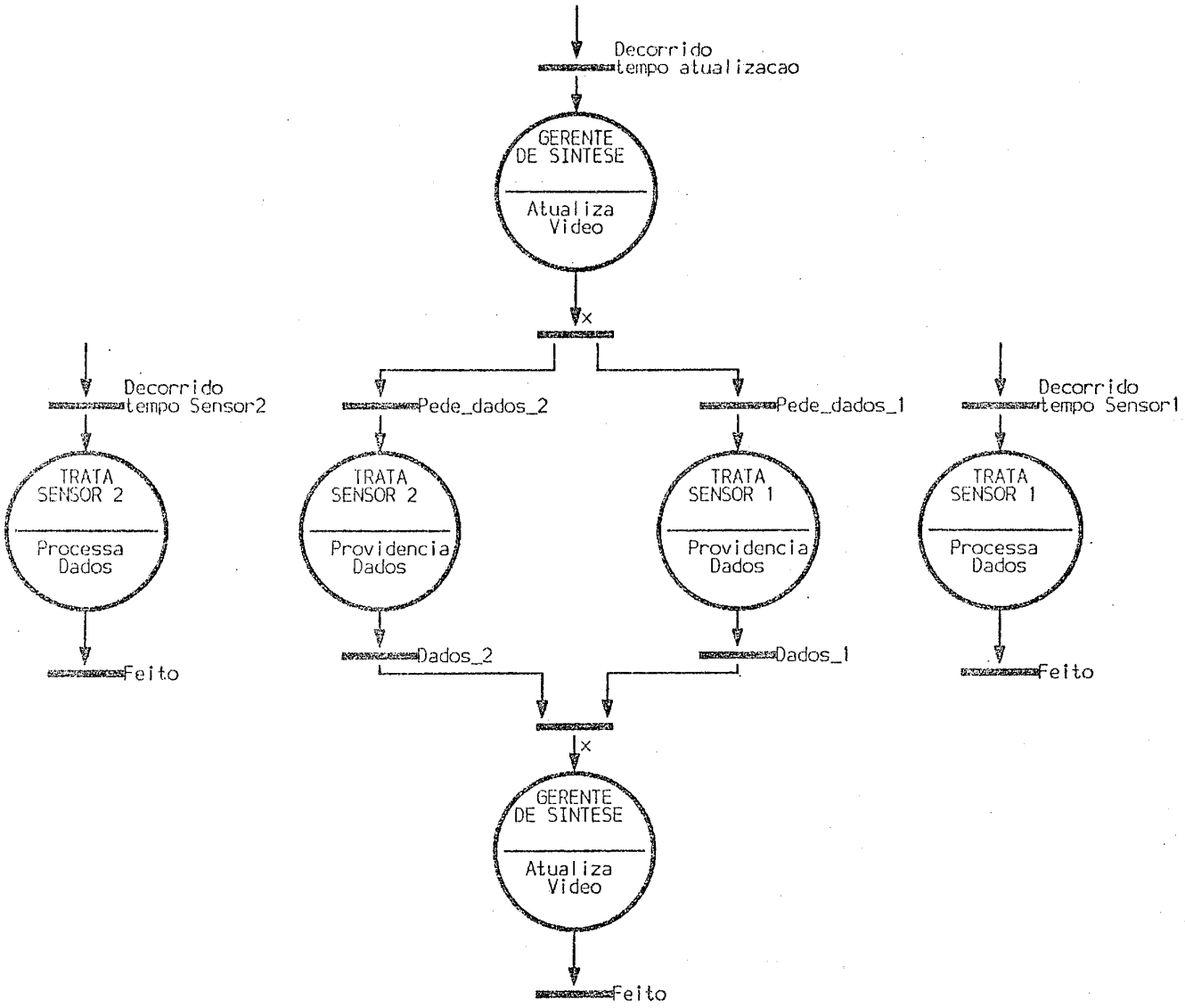
- O DC correspondente ao DE foi separado, por simplificacao, em duas partes. A primeira apresenta o tratamento de um comando pelo GERENTE DE SINTESE. A segunda apresenta o tratamento dos dados pelos subsistemas que tratam sensores e a atualizacao do video pelo GERENTE DE SINTESE. Esses dois DC's, por serem independentes, mostram a existencia de concorrência entre esses servicos.

- Para executar o Serviço de Analise Sintatica/Semantica, o GERENTE DE SINTESE nao pode estar analisando ou executando nenhum outro comando, representado pelo E presente entre o Ponto de Controle Sel_Cmdo e o Ponto de Controle Resp_Cmdo (comando nao aceito ou executado).

- A sequencialidade dos Servicos de Analise Sintatica/Semantica e Executa Comando, ambos do Subsistema Gerente de Sintese, fica explicita pois, um dos Pontos de Controle resultante da execucao do primeiro e condicao para a execucao do segundo.

DC - Exemplo

Relacao entre Subsistemas



Comentarios:

- O servico que fornece dados ao GERENTE DE SINTESE e ativado por pedido desse, representado pelos Pontos de Controle Pede_dados_1 e 2. Esses Pontos ocorrem concorrentemente, representado pelo E no Ponto de Controle de saida do GERENTE DE SINTESE.
- A resposta a esses pedidos podem vir em instantes de tempo diferentes mas, a atualizacao do video so ocorre apos existir dados disponiveis pelos dois sensores, respresentado pelo E no Ponto de Controle de entrada desta acao.
- O subsistema TRATA SENSOR 1 possui pelo menos dois processos. Um que processa dados, periodicamente, obtidos do sensor. Outro que atende aos pedidos de dados do GERENTE DE SINTESE.
- O subsistema GERENTE DE SINTESE possui pelo menos dois processos. Um responsavel pela analise e execucao de comandos, o outro responsavel pela manutencao do video. Sao totalmente independentes, pois nao possuem Pontos de Controle que os sequencialize.

3.4 - Subsistemas Básicos

3.4.1 - Características de um Subsistema Básico

Um Subsistema Básico tem como característica principal o fato de oferecer um conjunto coeso e reduzido de Serviços atuando sobre uma estrutura de dados bem definida. Esses Serviços são logicamente reunidos e possuem uma finalidade específica e, portanto, de rápido entendimento. Isto é, são determinados a partir de áreas de responsabilidade de uma particular solução para o sistema global. Suas conexões - Portas e Interfaces representam como estes Subsistemas Básicos podem trabalhar juntos para implementar essa solução.

A maior parte dos Subsistemas Básicos está em uma das seguintes categorias [7]:

- 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 estruturas de dados: visam manter a filososofia do encapsulamento de dados, bem como obter homogeneidade na segmentação, já que um dispositivo de hardware pode ser considerado uma estrutura de dados encapsulada em uma peça de hardware.
- Responsáveis por funções do sistema: funções relacionadas, não associadas diretamente a objetos físicos ou a estruturas de dados, devem ser reunidas em um Subsistema Básico.
- Responsáveis por relacionamento de funções: estes são os Subsistemas Básicos que coordenam a execução, essa coordenação é o que determina a forma de operação do sistema.

As duas últimas categorias são necessárias para aumentar o número de subsistemas reusáveis. Durante a segmentação, quaisquer aspectos não reusáveis, presentes nos serviços de um subsistema, devem ser encapsulados e associados a um ou mais Subsistemas Básicos não reusáveis. Funções reusáveis que não necessitam ser sempre usados em conjunto, também devem ser separados em Subsistemas Básicos diferentes.

3.4.2 - Interação entre Processos de um Subsistema Básico

Um Subsistema Básico é formado de uma ou mais unidades de execução concorrentes - Processos para um supervisor de tempo real. Na comunicação entre Processos de Subsistemas Básicos diferentes são utilizados os conceitos de Porta e Interface, isto é, Processos possuem Portas e Interfaces que as conectam, para possibilitar a comunicação entre eles. A comunicação e o sincronismo são feitos através da troca de mensagens. No entanto, internamente, conforme os tipos de Serviços executados pelo Subsistema Básico, poderão aparecer áreas de dados compartilhadas e a utilização de recursos tradicionais de programação concorrente como monitores, semáforos e eventos, além da troca menos estruturada de men-

sagens.

3.4.3 - Modelagem dos Subsistemas Básicos

3.4.3.1 - Especificação Detalhada de um Subsistema Básico

No Projeto Básico determina-se o conjunto de Serviços pertencente a cada Subsistema Básico de uma forma ainda abstrata e sucinta. Entretanto, para que a implementação seja realizada, cada Subsistema Básico necessita de uma Especificação Detalhada independente. Isto é, o conjunto de Serviços mencionado no Projeto Básico não é descrito com o nível de detalhe necessário à implementação. A partir disso e dos documentos definidos na fase de Análise do sistema (por ex.: Modelo da Essência [45,25]) é delineada esta especificação. Deve constar de:

- Descrição dos Serviços, incluindo Serviços Externos (prestados a outros subsistemas) e Serviços Internos ressaltando suas fontes de ativação;
- Serviços Requisitados, Serviços Externos oferecidos por outro subsistema e requisitados por esse;
- Descrição Detalhada dos Dados que constituem sua Interface externa;
- Descrição de Concorrência entre Serviços e suas prioridades relativas;
- Requisitos de Desempenho como o tempo máximo ou médio para execução de um Serviço;
- Parâmetros do Subsistema Básico, isto é, variáveis do sistema passíveis de parametrização;
- Subsídios de Implementação como normas, utilitários ou a escolha de algum algoritmo específico a ser empregado no desenvolvimento.

3.4.3.2 - Design de um Subsistema Básico

A determinação dos Processos é o primeiro passo no Design de um Subsistema Básico. A cada Serviço Interno e a cada Serviço Externo, executado pelo Subsistema Básico, está associado pelo menos um Processo.

Deve-se evitar associar a Serviços Externos diferentes (Portas de entrada diferentes) Serviços que não são efetivamente concorrentes. Por exemplo, Serviços de criam, cancelam ou alteram valores de um estrutura particular de dados não são na sua essência concorrentes pois, a execução de um destes Serviços implica no bloqueio da estrutura de dados, o que serializa esses procedimentos. Tratá-los em conjunto, por uma única porta de entrada, evita dispêndio adicional ("overhead") com a necessidade de introduzir mecanismos de exclusão mútua, durante o Design do Subsistema Básico ao qual estão incorporados.

Um Processo pode necessitar, para sua implementação, de um trabalho ainda grande de Design, já que, apesar de estar associado

a Serviços de fácil e rápido entendimento, tem que executar um conjunto de algoritmos, condições e controles, nem sempre triviais. É usada a decomposição tradicional em Módulos ou rotinas, baseada em decomposição funcional, procurando-se as 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. Para comunicação entre rotinas, já que aqui cada segmento tem natureza seqüencial, é usado o mecanismo tradicional de Call (chamada de sub-rotina).

3.5 - Emprego do Método

3.5.1 - Implantação da Reusabilidade

Após o desenvolvimento de alguns sistemas segundo esta técnica, a empresa disporá de uma biblioteca significativa de Subsistemas Básicos reusáveis. Durante a concepção do Projeto Básico deve ser feita uma pesquisa nessa biblioteca, para determinar os Subsistemas Básicos existentes cujas características, obtidas a partir de suas Especificações, sejam compatíveis com o sistema em desenvolvimento. A decomposição do sistema em subsistemas deve ser feita com esse conhecimento prévio e direcionada a aproveitar esses Subsistemas Básicos. Um sistema pode até vir a ser construído totalmente pela conexão de Subsistemas Básicos existentes, como hoje em dia se constrói um módulo de hardware pela interligação de componentes integrados disponíveis no mercado.

Para a implantação da política de reutilização, é necessário que o método de desenvolvimento esteja integrado a uma política administrativa que:

- garanta a implementação de Subsistemas Básicos reusáveis, analisando-os quanto a métricas de Reusabilidade como as propostas em [35] (por exemplo: ser o mais geral possível, não empregar variáveis globais, ser parametrizável, possuir boa documentação, etc.);
- controle o conjunto dos Subsistemas Básicos já existentes e efetive sua aplicação nos novos sistemas. A identificação de subsistemas que mais se adaptem a um ou outro requisito não é uma tarefa fácil. Um suporte automatizado, com sistema de biblioteca baseado em conhecimento [35] pode ajudar;
- gerencie as diversas versões de Subsistemas Básicos e os sistemas nos quais estão empregadas. Uma correção em um desses subsistemas implica na sinalização às equipes responsáveis pela manutenção dos outros sistemas que o utilizam.

3.5.2 - Ambiente de Desenvolvimento

O uso desta técnica implica em usar software básico e Linguagem de Programação apropriadas. O software básico deve permitir a criação de Processos e fornecer primitivas de Envia e Recebe mensagens entre quaisquer dois Processos. A Linguagem de Programa-

ção deve possuir facilidades de definição de Portas e uma Linguagem de Configuração capaz de implementar Interfaces e mapear Processos na Arquitetura de Hardware, por exemplo o ambiente CONIC [28,29,27]. Ambientes tradicionais de programação empreendem um esforço de adaptação para sua utilização. Essa era a situação durante a implementação do TTI. A descrição de Portas, Interfaces e Processos foi implementada através de uma tabela em um módulo independente de software. Esse módulo recebia as mensagens depositadas em uma Porta de saída (ou de saída com resposta), depositando-a na Porta de entrada (ou de entrada com resposta), correspondente a Interface definida por essas Portas, e sinalizava o Processo associado.

3.5.3 - Prevenção de Deadlocks

Um ponto relevante durante o Design de programas concorrentes não é abordado pelo método - prevenção de Deadlocks.

O particionamento em subsistemas independentes e a divisão dos Serviços para cada subsistema evita a alocação de recursos compartilhados entre Subsistemas Básicos. Mas isto ocorrerá entre Processos de um mesmo Subsistema Básico e deve ser abordado segundo técnicas tradicionais de prevenção de "deadlocks" [7,44].

Ao tratar os Subsistemas Básicos independentemente, reduz-se o universo de trabalho a Sistemas menores (cada Subsistema Básico) o que facilita a trabalho do projetista em prever pontos de Design passíveis de "deadlock". Situações possíveis entre Subsistemas Básicos podem ocorrer, como o exemplo a seguir:

- O subsistema A possui 2 Processos A1 e A2 que compartilham recursos;
- A1 pede um Serviço com resposta associada a um Processo B1 do subsistema B;
- B1 pede um Serviço com resposta associada a um Processo C1 do subsistema C;
- C1 pede um Serviço com resposta associada a A2 que necessita de recurso alocado a A1;

os processos A1, A2, B1 e C1 assim entrarão em deadlock.

É fundamental evitar pedidos de Serviços encadeados que retornem a um dos subsistemas já participantes.

O método apresentado não fornece nenhum auxílio efetivo para identificação de situações como a descrita acima. Dependerá da perícia do projetista o acompanhamento do comportamento do sistema para deteção de fontes de "deadlocks". No entanto, como já mencionado, a característica do particionamento e abstração empregados, resultam em uma estrutura de sistema suficientemente simples, que facilita esse trabalho.

4 - Projeto Básico

A modelagem diz respeito ao software do TTI e evidencia os conceitos descritos no capítulo 3. Relembrando, o Projeto Básico é a etapa de identificação dos Subsistemas Básicos através da segmentação do sistema em subsistemas, a partir dos Serviços apresentados no capítulo 1. Critérios principais empregados: Encapsulamento de Dados, Reusabilidade, Serviços e Portas.

Para evitar que o trabalho ficasse muito extenso, é apresentado somente uma parte do modelo de segmentação. Consta do primeiro nível e do detalhamento do subsistema Interface Usuário Máquina, presente nesse nível.

4.1 - Modelo de Subsistemas

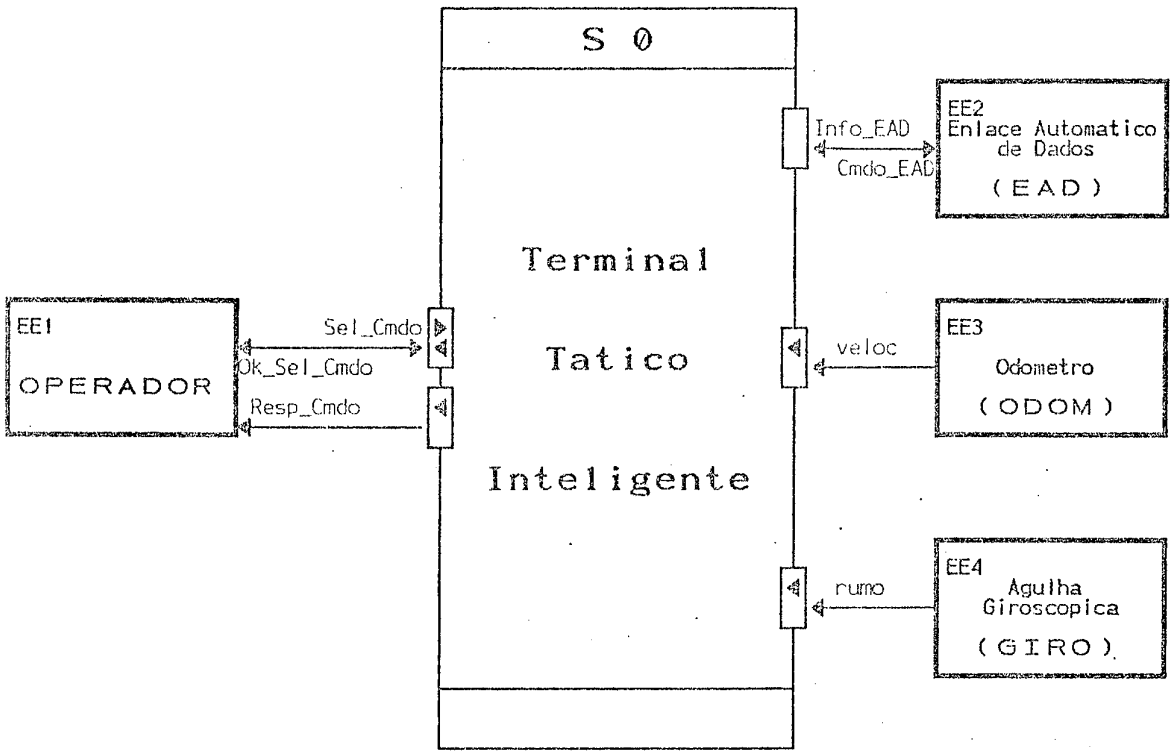
A construção deste modelo explicita o conjunto hierárquico de subsistemas, usados como base de Design para o sistema em questão. É constituido de um conjunto de diagramas hierárquicos que representam o contexto, a estrutura e o comportamento do sistema e de seus subsistemas.

4.1.1 - Contexto

Apresenta a visão do sistema como um todo, suas entidades externas, e a interação entre essas e o sistema. Como ferramenta de representação é utilizado o Diagrama de Estrutura.

No diagrama de contexto, não está presente o radar, já que seu processamento é realizado apenas pelo hardware, para digitalização do vídeo bruto radar, não possuindo interligação direta com o software.

Diagrama de Contexto



S O - Terminal Tático Inteligente
Serviços a Executar

1- Gerenciar a entrada de comandos, interpretando-os.

2- Informar os resultados de execução dos comandos, através de apresentação gráfica e/ou alfanumérica, conforme o caso.

3- Apresentar o quadro tático de forma clara, sintética e constantemente atualizada.

4- Executar os comandos táticos fornecidos pelo operador.
   4.1- Acompanhar Alvos
   4.2- Controlar Áreas Táticas
   4.3- Executar Soluções Táticas

5- Processar informações provenientes dos sensores e controlá-los, quando necessário.

4.1.2 - Decomposição do Sistema

Consta do conjunto de DE's e DC's que representam a decomposição do sistema em subsistemas, em uma organização "top-down", até o nível de Subsistemas Básicos. Cada subsistema é responsável por Serviços, de forma que o conjunto dos Subsistemas Básicos, que constituem a estrutura do sistema, executem todos os Serviços para ele especificados. A modelagem do sistema/subsistema deve constar de:

- Diagrama de Estrutura para este sistema/subsistema;
- Diagrama de Comportamento que especifique a concorrência/seqüencialização entre os Serviços executados pelos subsistemas componentes;
- Lista de Serviços executados por cada subsistema.

Como a finalidade deste documento é evidenciar as técnicas descritas no capítulo 3, foram acrescentados, a cada par (DE,DC), os critérios empregados que justificam a solução de modelagem.

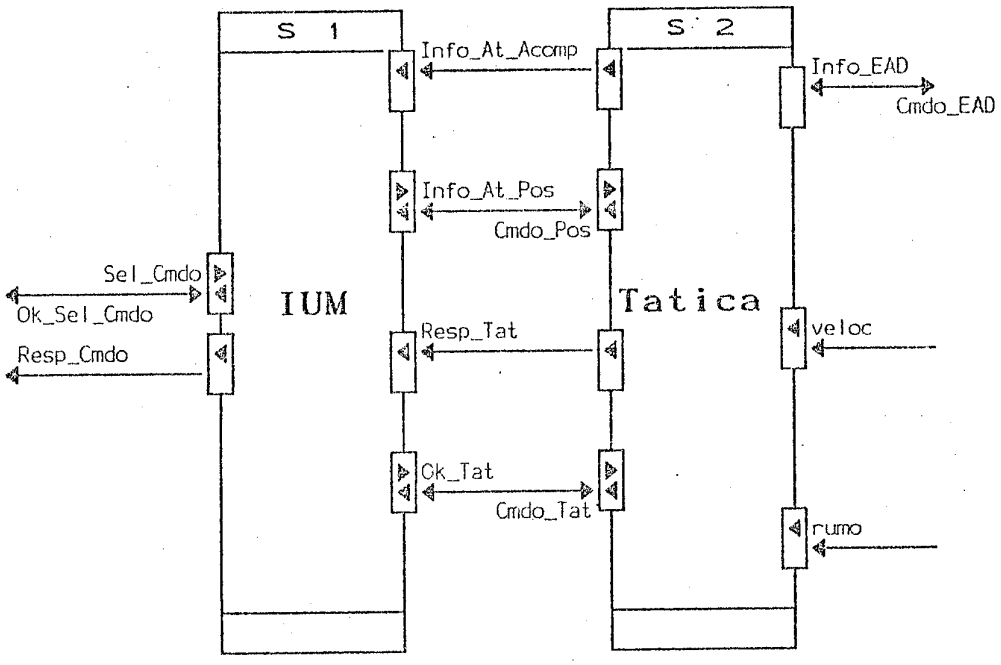
Pode-se notar que as orientações para identificação de Subsistemas Básicos (ítem 3.4.1) são extensíveis a todos os níveis de decomposição e bastante empregadas, isto é, classificam os subsistemas em:

- Responsáveis por objetos físicos;
- Responsáveis por estruturas de dados;
- Responsáveis por funções do sistema;
- Responsáveis por relacionamento de funções.

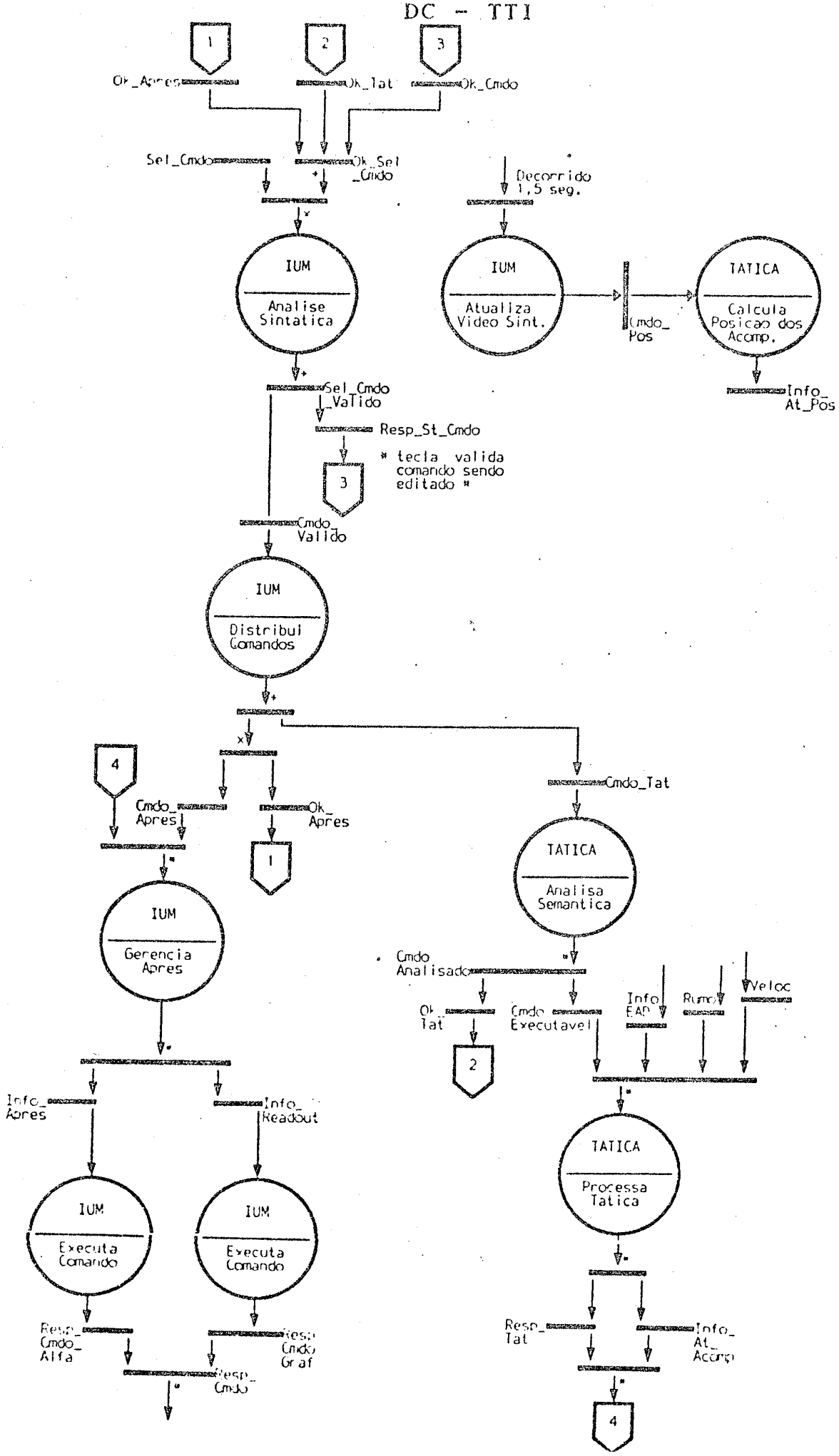
Os requisitos de Encapsulamento de Dados e Reusabilidade guiam as decisões de particionamento. Para determinar quantos subsistemas deixar em cada nível, deve-se tomar com base, em princípio, a reunião, em um único subsistema dos Serviços que normalmente serão usados em conjunto. Conjuntos de Serviços relacionados comuns a diversos sistemas navais, devem ser encapsulados em um subsistema, permitindo sua reutilização por inteiro, mesmo que não seja um Subsistema Básico.

O conceito de Reusabilidade é particular para uma dada empresa. Existe uma gama de sistemas de interesse da empresa que devem ser considerados quando da análise de um subsistema ser ou não reusável.

DE / S0 - Terminal Tatico Inteligente



DC - TTI


1
2
3
Ok_Apres
Ok_Tat
Ok_Cmdo
Sel_Cmdo
Ok_Sel_Cmdo
Decorrido 1,5 seg.
IUM
Analise Sintatica
IUM
Atualiza Video Sint.
Cmdo_Pos
TATICA
Calcula Posicao dos Acomp.
Info_At_Pos
Sel_Cmdo_Valido
Resp_St_Cmdo
* tecla valida comando sendo editado *
3
Cmdo_Valido
IUM
Distribui Comandos
4
Cmdo_Apres
Ok_Apres
1
IUM
Gerencia Apres
Info_Apres
Info_Readout
IUM
Executa Comando
IUM
Executa Comando
Resp_Cmdo_Alfa
Resp_Cmdo_Graf
Resp_Cmdo
Cmdo_Tat
TATICA
Analisa Semantica
Cmdo Analisado
Ok_Tat
2
Cmdo Executavel
Info_EAP
Rumo
Veloc
TATICA
Processa Tatica
Resp_Tat
Info_At_Acomp
4

S 1 - Interface Usuário Máquina (IUM)
Serviços a Executar

1- Gerenciar a entrada de comandos, interpretando-os.
   1.1- Permitir a seleção de comandos, através de "softkeys" e Teclas Dedicadas.
   1.2- Coordenar a entrada de parâmetros do comando selecionado, através de tela pré-formatada.
   1.3- Transformar os comandos táticos, sintaticamente corretos, em pedido de Serviço a outros subsistemas.

2- Informar os resultados de execução dos comandos, através de apresentação gráfica e/ou alfanumérica, conforme o caso.

3- Apresentar o quadro tático de forma clara, sintética e constantemente atualizada.

S 2 - Tática
Serviços a Executar

1- Acompanhar Alvos.
   1.1- Criar/Atualizar/Cancelar Acompanhamentos.
   1.2- Manter as informações gerais de todos os acompanhamentos.
   1.3- Informar posição atualizada dos Acompanhamentos.
   1.4- Informar mudanças nos Acompanhamentos que influenciem sua apresentação.

2- Controlar Áreas Táticas.

3- Executar Soluções Táticas.

   3.1- Validar comandos de Solução Tática quanto a existência dos acompanhamentos.
   3.2- Calcular Soluções Táticas.

4- Processar informações provenientes dos sensores e controlá-los, quando necessário.
   4.1- Acompanhar Alvos cujos dados são extraídos dos sensores.
   4.2- Comandar a operação do sensor em funcionamento (por exemplo: qual a taxa de rotação da antena radar)

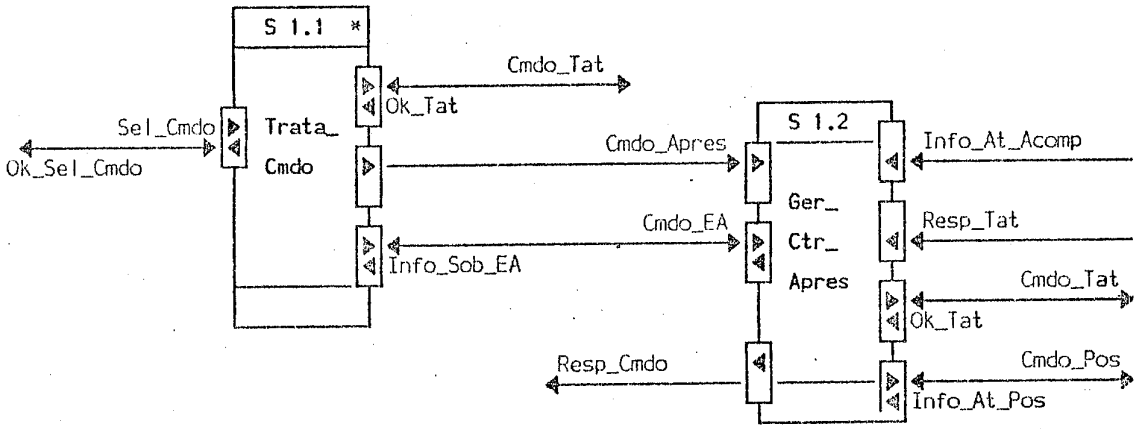
Critérios Empregados na Solução

A primeira decomposição do Sistema levou a dois subsistemas que prestam Serviços bastante distintos. Aqueles alocados a IUM dizem respeito apenas a apresentação de informações ao operador, enquanto que os alocados a Tática coordenam todo o conjunto de funções associadas a processamento de sensores e Soluções Táticas. Um particionamento deste tipo privilegia, principalmente, a Reusa-

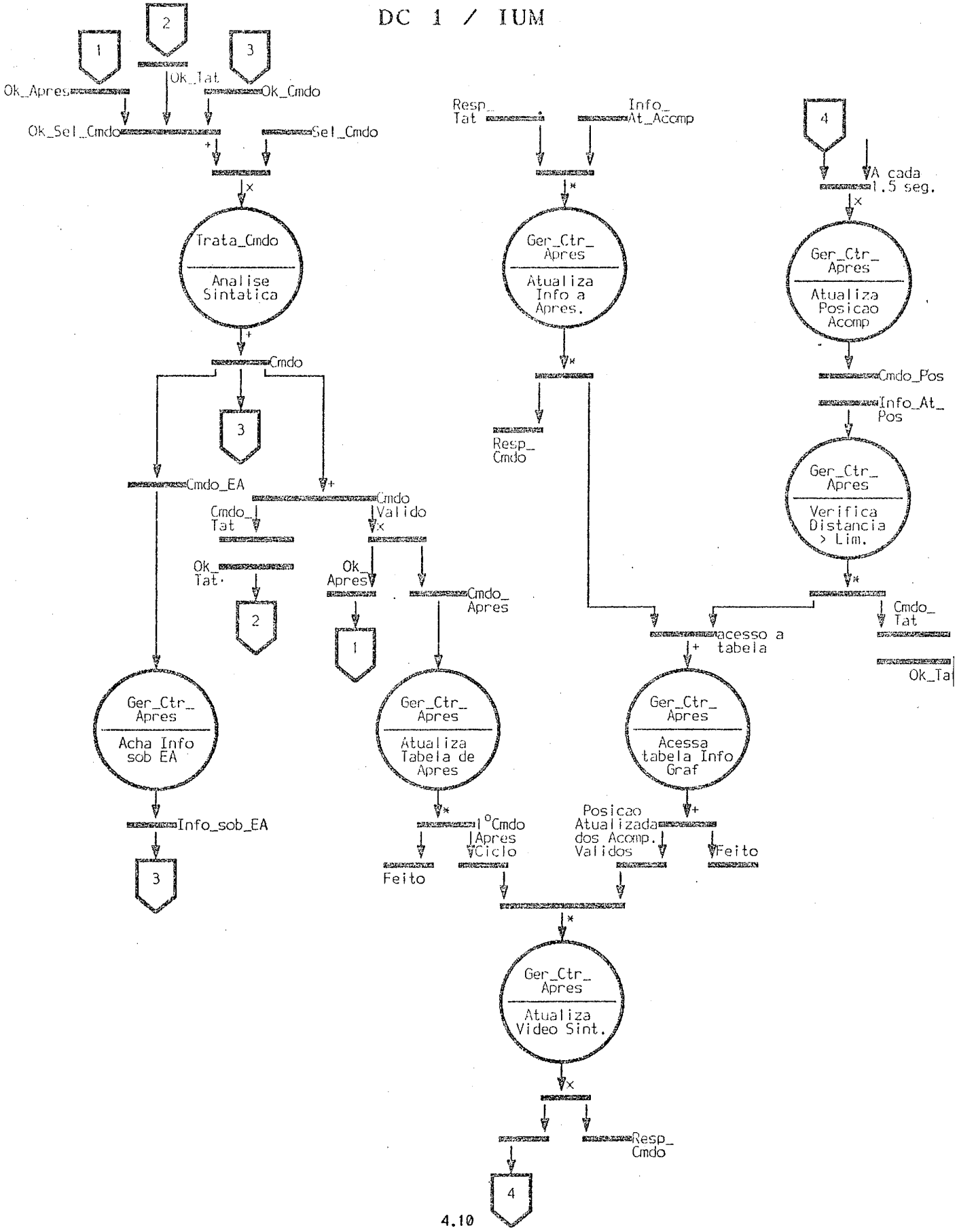
bilidade de subsistemas. Suponha um outro sistema similar ao TTI mas com características diferentes de apresentação de informação ao operador. Por exemplo, ao invés de apresentar o vídeo sintético em um terminal gráfico, apresentaria em um telão ou imprimiria as informações em uma impressora. Como as funções táticas e de processamento de sensores foram isoladas em um subsistema, esta mudança em nada influenciaria esta parte do projeto. Suponha agora, sistemas com finalidades diferentes, por exemplo um sistema de controle de armas, poderá usar basicamente a mesma IUM com pequenas variações de configuração.

Foram utilizadas as regras de identificação de subsistemas - cada um dos subsistemas cuida de um conjunto distinto de objetos físicos. Como o nível de abstração ainda é alto, os subsistemas não são responsáveis por apenas um destes objetos mas sim por um conjunto sem, no entanto, existir interseção entre eles.

DE / S1 - IUM


S 1.1 *
Cmdo_Tat
Ok_Tat
Sel_Cmdo
Ok_Sel_Cmdo
Trata_
Cmdo
Cmdo_Apres
S 1.2
Info_At_Acomp
Ger_
Ctr_
Apres
Cmdo_EA
Info_Sob_EA
Resp_Tat
Cmdo_Tat
Ok_Tat
Resp_Cmdo
Cmdo_Pos
Info_At_Pos

DC 1 / IUM


1
2
3
Ok_Apres
Ok_Tat
Ok_Cmdo
Ok_Sel_Cmdo
Sel_Cmdo
Trata_Cmdo
Analise Sintatica
Cmdo
Cmdo_EA
Cmdo_Tat
Cmdo
Valido
Ok_Tat
Ok_Apres
Cmdo_Apres
Ger_Ctr_Apres
Acha Info sob EA
Info_sob_EA
Ger_Ctr_Apres
Atualiza Tabela de Apres
Feito
Cmdo
Apres
Ciclo
Resp_Tat
Info_At_Comp
Ger_Ctr_Apres
Atualiza Info a Apres.
Resp_Cmdo
4
A cada 1.5 seg.
Ger_Ctr_Apres
Atualiza Posicao Acomp
Cmdo_Pos
Info_At_Pos
Ger_Ctr_Apres
Verifica Distancia > Lim.
Cmdo_Tat
acesso a tabela
Ger_Ctr_Apres
Acessa tabela Info Graf
Posicao Atualizada dos Acomp. Validos
Feito
Ger_Ctr_Apres
Atualiza Video Sint.
Resp_Cmdo

S 1.1 - Trata Comando
Serviços a Executar

1- Permitir a seleção de comandos, através de "softkeys" e Teclas Dedicadas.
   1.1- Caminhar na árvore de comandos, relacionando o nível corrente da árvore com os valores das "softkeys".
   1.2- Tratar Tecla Dedicada, o que leva diretamente a uma folha da árvore de comandos.
   1.3- Ecoar os comandos selecionados bem como o estado corrente na seleção de um comando.
   1.4- Tratar teclas de controle de entrada de comandos.

2- Coordenar a entrada de parâmetros do comando selecionado, através de tela pré-formatada.
   2.1- Apresentar a tela pré-formatada, relacionada a entrada de parâmetros do comando escolhido.
   2.2- Ecoar parâmetros digitados.
   2.3- Validar quanto a: parâmetros obrigatórios, tipos de dados e faixas de validade.
   2.4- Transformar os parâmetros para unidades e coordenadas internas ao sistema.
   2.5- Tratar parâmetros do tipo posição e NA inseridos pela Esfera de Acompanhamento.

3- Transformar os comandos táticos, sintaticamente corretos, em pedido de Serviço a outros subsistemas.

S 1.2 - Ger_Ctr_Apres
Serviços a Executar

1- Informar os resultados de execução dos comandos, através de apresentação gráfica e/ou alfanumérica, conforme o caso.
   1.1- Selecionar a forma de apresentação do resultado de execução de um comando.
   OBS: a) A forma de apresentação pode ser apenas gráfica, apenas alfanumérica ou ambas, em função do tipo de comando.
   b) O resultado de execução de um comando inclui os dados Info_At_Acomp (resultado de Acompanhar um Alvo) e Resp_Tat (resultado de uma Solução Tática).
   c) A maior parte dos comandos implica em monitoração, isto é, após inserido o comando no sistema, o resultado de sua execução é enviado periodicamente ou sempre que ocorrer um evento, por exemplo, mudança de velocidade de um Acompanhamento.
   1.2- Garantir a consistência entre as apresentações gráficas (vídeo sintético) e alfanumérica (READOUT).
   1.3- Manter a apresentação alfanumérica do resultado de uma Solução Tática - READOUT.

2- Apresentar o quadro tático de forma clara, sintética e constantemente atualizada.
   2.1- Executar os comandos de apresentação gráfica para o vídeo sintético.
   2.2- Atualizar periodicamente a posição dos Acompanhamentos no vídeo sintético.

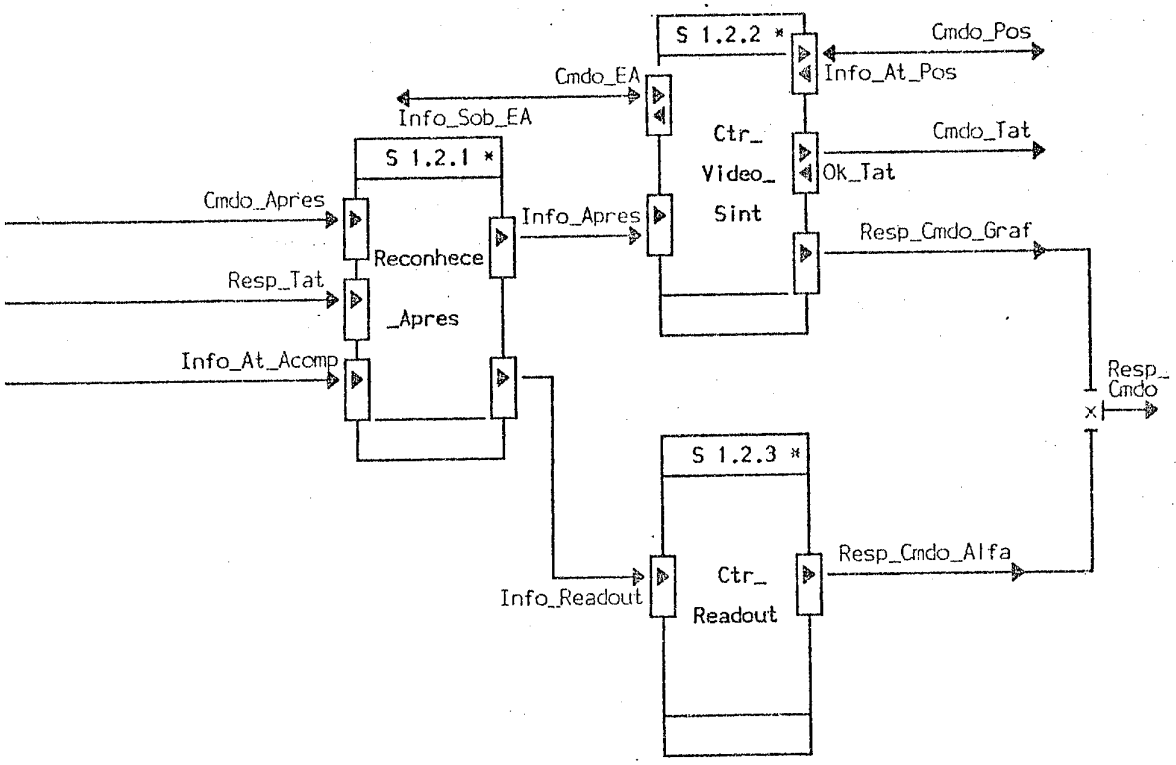
Critérios Empregados na Solução

A IUM foi decomposta em dois subsistemas com Serviços independentes e o critério principal foi a Reusabilidade dos componentes. Pode-se observar que cada subsistema cuida de formas de interação diferentes com o operador.

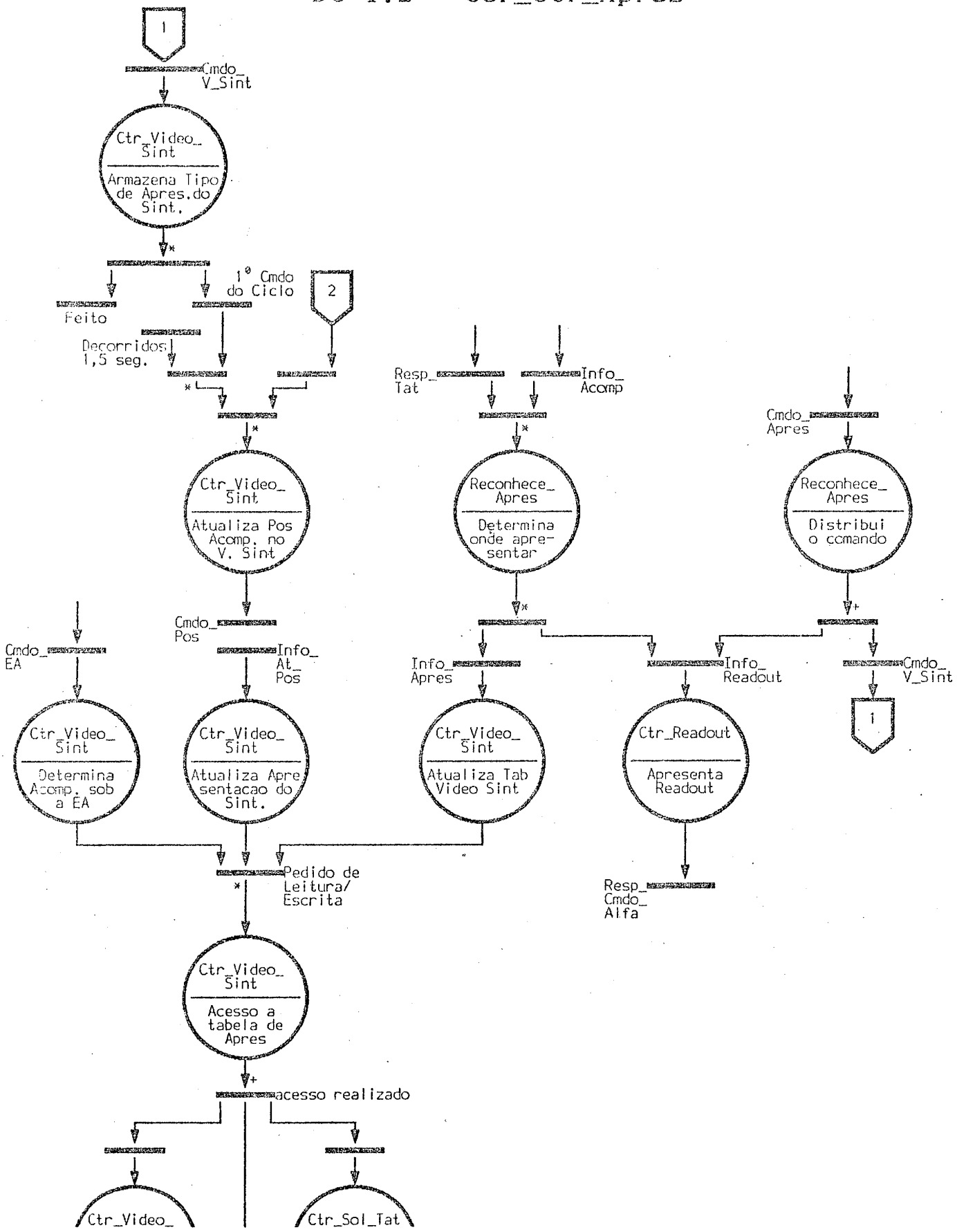
O subsistema Ger Ctr Apres gerencia as apresentações ao operador, no que diz respeito a: vídeo sintético atualizado e Readout.

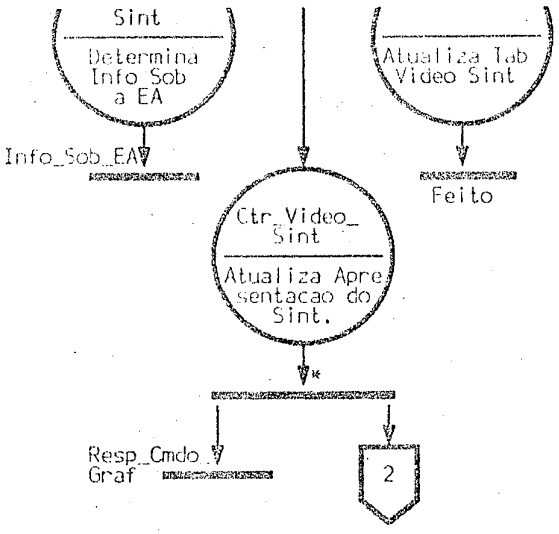
O outro subsistema, Trata Cmdo, coordena a entrada de comandos. Para efetuar este tratamento, é responsável pelos objetos físicos: teclado e EA, que permitem a seleção de comandos, e pela fração relativa a apresentação de informações associadas a comandos no vídeo gráfico. Apesar do terminal gráfico ser um objeto físico já tratado pelo subsistema Ger Ctr Apres, esta opção foi utilizada pois, por motivos de hardware, a interface com o terminal gráfico, no que diz respeito à informações alfanuméricas, é diferente da interface no que diz respeito a informações gráficas. Foi implementada uma biblioteca, que complementou o ambiente de desenvolvimento, com procedimentos do tipo "Read" e "Write", como as disponíveis para ambientes convencionais e utilizada pelos dois subsistemas. A segmentação nesse caso, privilegiou o encapsulamento da função tratamento de comando em um Subsistema Básico pois, todos os serviços prestados por esse subsistema necessitam estar juntos quando reutilizados.

DE / S 1.2 - Ger_Ctr_Apres



DC 1.2 - Ger_Ctr_Apres




Sint
Determina Info Sob a EA
Info_Sob_EA
Atualiza Tab Video Sint
Feito
Ctr_Video_Sint
Atualiza Apresentacao do Sint.
Resp_Cmdo Graf
2

S 1.2.1 - Reconhece_Apres
Serviços a Executar

1- Selecionar a forma de apresentação do resultado de execução de um comando.
Por exemplo, resposta a um PMA deve ser da seguinte forma: apresentação gráfica no vídeo sintético e apresentação alfanumérica na área dedicada à resposta de comandos.

2- Garantir a consistência entre as apresentações gráficas (vídeo sintético) e alfanumérica (READOUT).
Indicações de alterações nos Acompanhamentos sendo apresentados devem ocasionar, caso seja necessário, uma alteração na apresentação do Vídeo Sintético e no Readout.

S 1.2.2 - Ctr_Video_Sint
Serviços a Executar

1- Executar os comandos de apresentação gráfica para o vídeo sintético.
1.1- Escala.
1.2- Centrar/Descentrar.
1.3- Movimento Verdadeiro/Relativo.
1.4- Filtros de Apresentação.

2- Movimentar e atualizar periodicamente a posição dos Acompanhamentos no vídeo sintético.

S 1.2.3 - Ctr_Readout
Serviços a Executar

1- Manter a apresentação alfanumérica do resultado de uma Solução Tática - READOUT.
1.1- Apresentar READOUT conforme formato predefinido.
1.2- Fazer a transformação das coordenadas/unidades internas em coordenadas/unidades de apresentação dos dados da solução a ser apresentada.

Critérios Empregados na Solução

Os subsistemas, Ctr Video Sint e Ctr Readout, controlam o tratamento de apenas um objeto físico, o terminal gráfico. Cada um deles gerencia por formas de interação diferentes com o operador e, portanto, foram decompostos em dois subsistemas. Do lado esquerdo da tela existe uma região totalmente dedicada a apresentação gráfica e independente da outra região. Do lado direito mantém apresentações alfanuméricas dos resultados de execução de comandos. Esses subsistemas podem ser classificados como subsistemas responsáveis por funções independentes e também por estruturas de dados independentes. O subsistema Ctr Video Sint encapsula uma estrutura de dados que corresponde ao vídeo sintético sendo apresen-

tado. O subsistema Ctr_Readout corresponde a estrutura de dados que trata formatos de Readout e contém o Readout corrente.

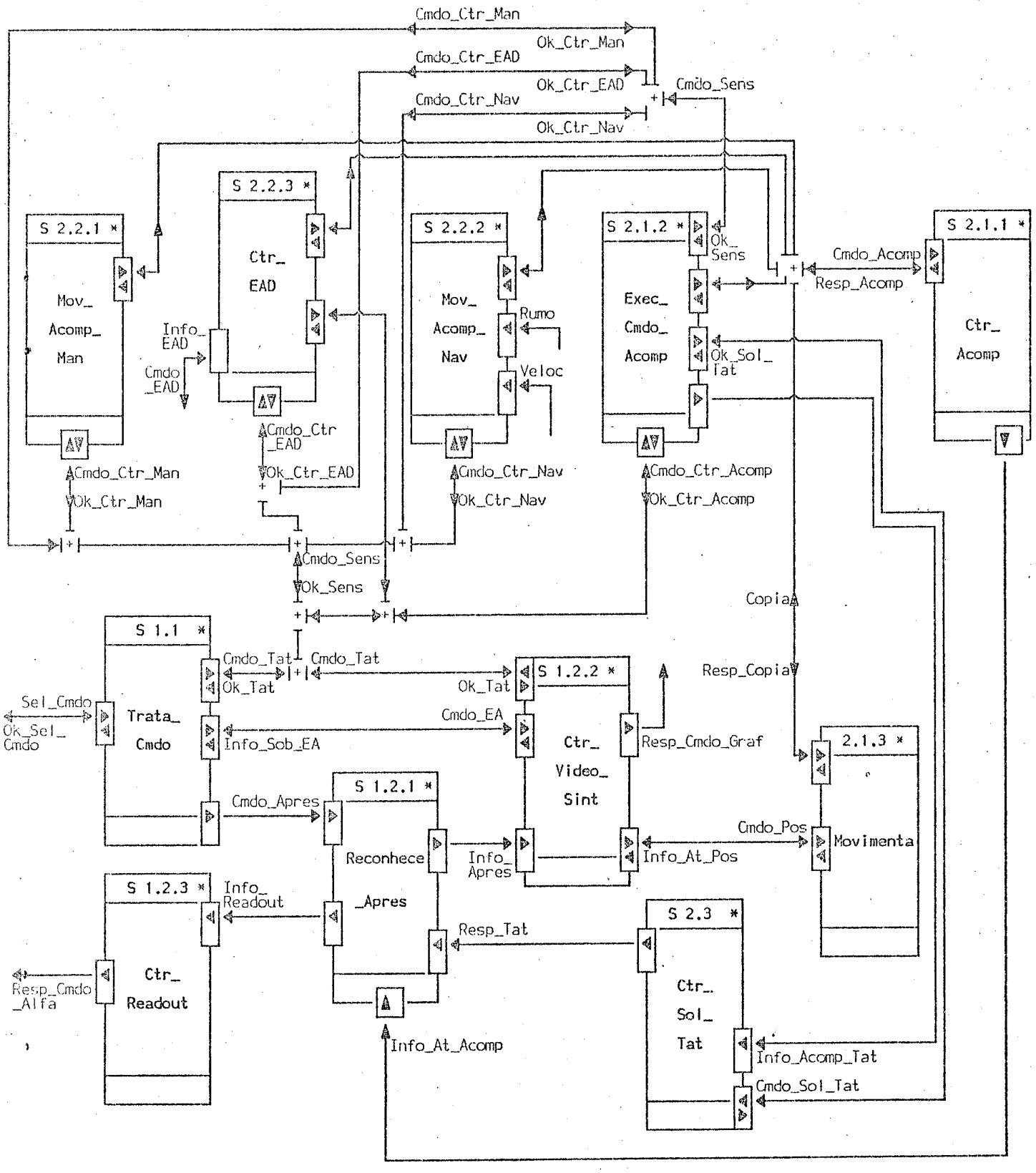
O subsistema Reconhece_Apres cuida do relacionamento de funções. Foi separado para que se isolasse uma parte não reusável das outras reusáveis - Ctr_Video_Sint e Ctr_Readout, para outros sistemas navais. Reconhece_Apres é tipicamente não reusável, pelo menos sem alteração explícita do seu projeto/código, já que sua função é separar o que deve ser apresentado em Readout ou no Vídeo Sintético, decisões extremamente relacionadas a Especificação Operativa deste sistema.

O subsistema Ger_Ctr_Apres jamais poderia ser visto como um Subsistema Básico porque existe dentro dele esta fração não reusável (Reconhece_Apres) e, portanto, uma nova decomposição se fazia necessária de acordo com as normas para identificação de Subsistemas Básicos.

4.1.3 - Diagrama dos Subsistemas Básicos

Apresenta todos os Subsistemas Básicos em um único diagrama. Consta de um DE com esses subsistemas e sua interação, no nível mais baixo de decomposição neste escopo - Interfaces entre Subsistemas Básicos. Esta reunião em um diagrama global, facilita a integração de Subsistema Básicos e a implementação do Módulo responsável pela Configuração do Sistema.

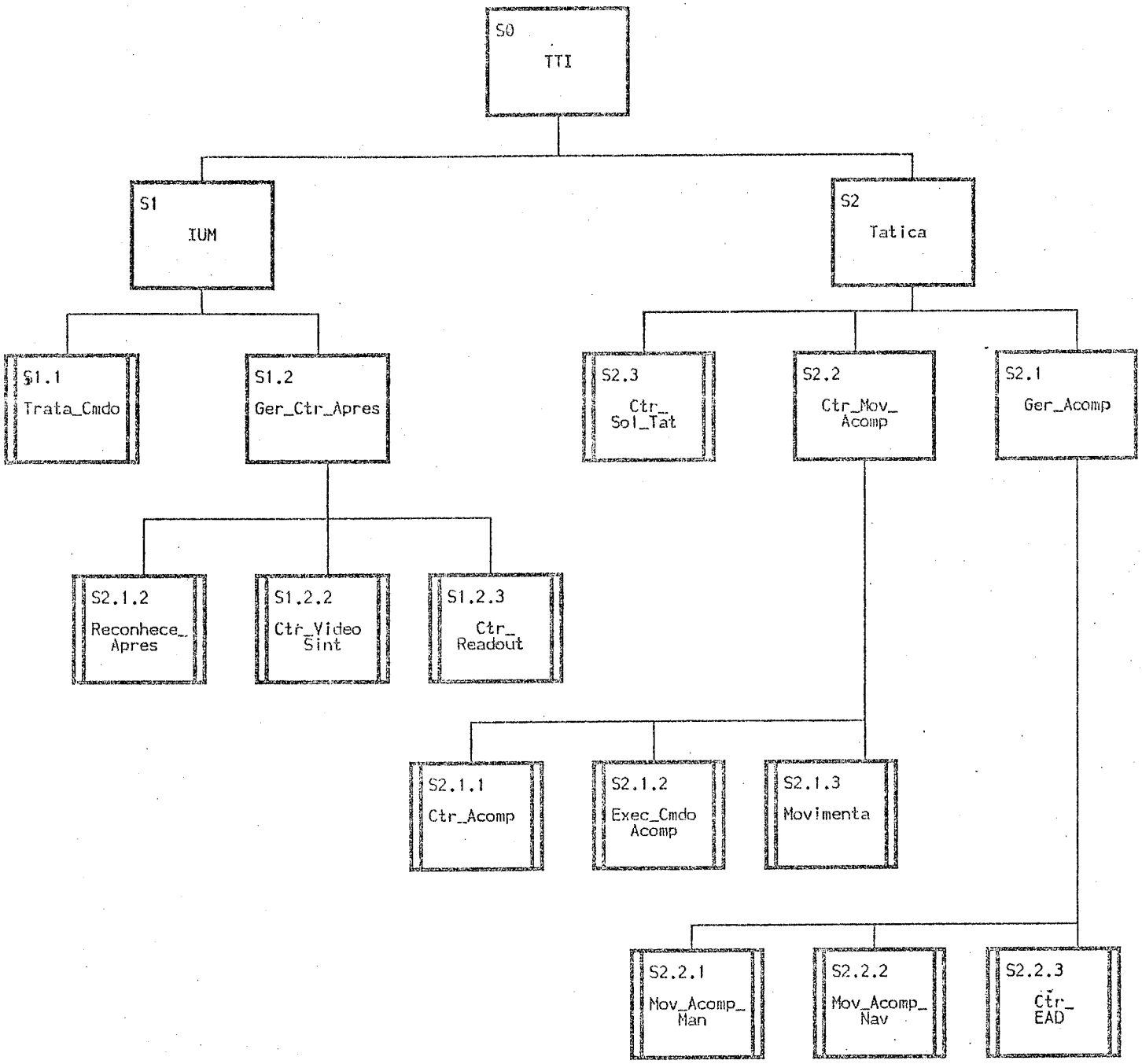
Diagrama dos Subsistemas Basicos



4.1.4 - Árvore de Subsistemas

Proporciona um visão sucinta da estrutura hierárquica selecionada, facilitando a identificação de decomposição de um dado Serviço e o acesso, de uma forma geral, aos documentos que detalham a implementação do sistema.

Arvore de Subsistemas



onde :

Subsistema

Subsistema Basico

4.3 - Dicionário de Dados

É constituido de uma entrada por dado que flui nas Interfaces, entidade externa e Eventos presentes nos DE's e DC's. Apresenta todos os dados que detalham a solução do problema, bem como, aqueles usados recursivamente nas definições.

As entradas são organizadas em ordem alfabética, possuindo em cada uma delas:

- Nome: Apresenta o nome do dado que está sendo definido;
- Definição: Fornece o significado deste dado para o sistema;
- Composição: Apenas para dados compostos pela agregação de outros dados, utiliza linguagem padrão BNF para especificar a forma de composição destes dados.
  + --> E
  | --> OU
  [ ] --> Opcional
  { } --> Conjunto
- Tipo: Apenas para dados elementares, contém informação do formato de armazenamento do dado (Inteiro, Real, etc.), precisão, unidade e domínio.

4.4 - Especificação dos Subsistemas Básicos

Neste ítem será apresentada, a título de exemplo, a especificação de um Subsistema Básico de forma a completar a exposição conceitual do Modelo da Implementação do Terminal Tático Inteligente, foi escolhido o subsistema Mov_Acomp_Man. A apresentação da especificação de todos os Subsistemas Básicos seria excessivamente exaustiva e foge ao escopo do presente trabalho. A especificação deste subsistema também será parcial. Ítens com descrição longa serão reduzidos para o mínimo necessário a exemplificação.

4.4.1 - Mov_Acomp_Man

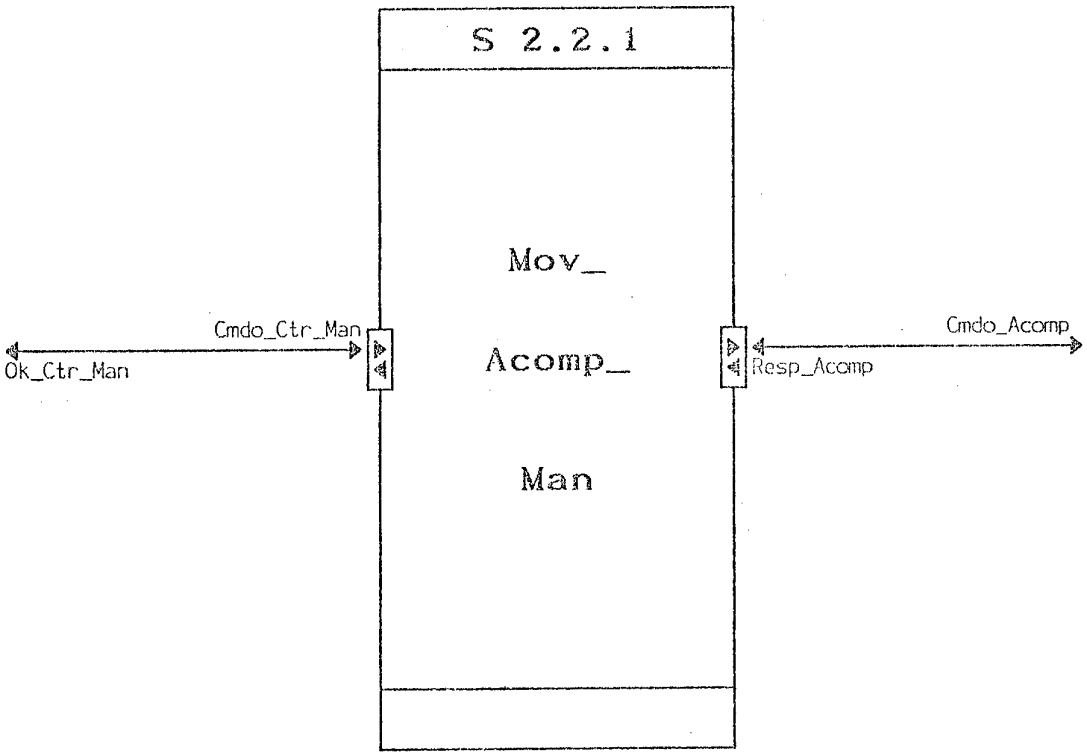
O objetivo deste subsistema é manter acompanhamento de alvos cujas informações de posição são fornecidas pelo operador, através da EA. É capaz de armazenar, estimar e informar dados de Acompanhamentos Manuais.

Após a iniciação de um Acompanhamento, o módulo é responsável por todas as suas informações cinemáticas. Deve, portanto, manter uma estrutura de armazenamento com todos os dados relevantes aos Acompanhamentos sendo efetuados.

4.4.2 - Contexto

Consta do DE correspondente a visão do Subsistema Básico, como um todo, e das Interfaces, através das quais se comunica com outros subsistemas ou Entidades Externas, prestando ou pedindo Serviços.

Diagrama de Contexto de Subsistema Basico



4.4.3 - Descrição de Serviços

4.4.3.1 - Serviços Externos

A seguir, será descrito o procedimento para apenas um Serviço Externo executado pelo subsistema.

- Controle de Acompanhamento Manuais

Entradas : Cmdo_Ctr_Man

Saídas : Ok_Ctr_Man.

Descrição :

Ao dado Cmdo_Ctr_Man está associado um Serviço a ser executado, que diz respeito à iniciação, alteração ou cancelamento de um Acompanhamento Manual. O tipo de Serviço é identificado através de um campo com código no dado de entrada. Para cada pedido de Serviço recebido, é realizada uma validação quanto à possibilidade de execução pelo subsistema e a resposta é enviada no dado Ok_Ctr_Acomp; sua execução implica em mudanças de conteúdo da estrutura interna de armazenamento.

a - Iniciar um Acompanhamento Manual

É executado a partir do recebimento do dado Cmdo_Ctr_Acomp, contendo as seguintes informações:
- Código de Iniciação de um Acompanhamento Manual;
- NA (pode assumir valor prédefinido);
- CAT (pode assumir valor prédefinido);
- Posição;

Ao receber o comando de Iniciar um Acompanhamento Manual, é alocada uma entrada na estrutura de armazenamento interna ao subsistema e enviado um pedido de Serviço, através do dado Cmdo_Acomp, contendo a identificação que será usada para esse Acompanhamento Manual (ID-Acomp_Man), no âmbito do subsistema, pedindo Iniciação de um Acompanhamento. Retornará, como resultado deste Serviço Requisitado, o dado Resp_Acomp com uma das seguintes informações:
- Já existe um Acompanhamento no sistema com o mesmo Número de Acompanhamento proposto;
- Não existe nenhum Acompanhamento no sistema com este número portanto, seu NA já foi alocado e sua identificação interna é ID_Acomp.

É então enviado uma resposta através do dado Ok_Ctr_Man com os valores:
- Comando inválido ou
- Comando aceito

para os casos acima mencionados, respectivamente.

O Acompanhamento é iniciado com as informações constantes no dado Cmdo_Ctr_Acomp e com valores de rumo e velocidade iguais a zero. Estes valores são calculados automaticamente pelo subsistema a cada pedido de atualização de posição do Acompanhamento, após a primeira atualização.

O número máximo de Acompanhamentos Manuais deve ser configurável para cada geração do subsistema, conforme as necessidades do sistema onde será utilizado.

4.4.3.2 - Serviços Internos

Não Aplicável

4.4.3.3 - Serviços Requisitados

a- Apres dos Acompanhamentos Manuais

Saídas : Cmdo_Acomp

Entradas : Resp_Acomp

Descrição :

Esse Serviço é requisitado na criação, atualização e cancelamento dos Acompanhamentos Manuais, para informar a outro subsistema o estado dos existentes no subsistema Mov_Acomp_Man.

4.4.4 - Descrição Detalhada dos Dados

- Tipos Globais

Os tipos devem ser definidos em uma linguagem similar a Pascal, definindo-os precisamente o formato dos dados que fluirão nas portas deste subsistema.Por exemplo:

```
Tipo_ID_Acomp  :  Inteiro;

/* Modela o acesso direto a tabela de armazenamento de  Acompanhamentos */

Tipo_Posição : Record;
      Pos_X : real;
      Pos_Y : real;

/*  Modela o valor da posição de um Acompanhamento em  coordenadas cartesianas e referencial inercial */
```

4.4.5- Descrição de Concorrências

Todo Serviço é iniciado através do dado Cmdo_Ctr_Acomp, e os três Serviços que o compõem não são concorrentes.

4.4.6 - Requisitos de Desempenho

O tempo máximo para execução de qualquer um dos Serviços é de 2 seg.

4.4.7 - Parâmetros do Subsistema Básico

Esse subsistema deve ter como parâmetro de geração o número máximo de Acompanhamentos Manuais suportados pelo subsistema.

4.4.8 - Subsídios de Implementação

Descrição, por exemplo, do algoritmo para cálculo de Rumo e Velocidade do Acompanhamento Manual.

5 - Conclusão

Nesse trabalho é possível observar a proposta de Modelo de Implementação apresentado em [41]. É importante notar que o sistema modelado possui uma complexidade razoável, e que, apesar disso, pode ser descrito com relativa simplicidade e precisão necessária a sua implementação.

As características do método e da documentação, que segmentam o curso do Design em dois níveis, possibilitam o uso de técnicas gerenciais e o controle de mudanças nas Especificações e Interfaces dos Subsistemas Básicos. Facilita-se o trabalho em equipe já que cada Subsistema Básico fica sob a responsabilidade de um ou dois membros da equipe, permitindo o trabalho independente mas com Interfaces bem definidas. A gerência de mudanças (por exemplo: alteração na Interface de um Subsistema Básico) deve ser feita pelo Gerente do Projeto, de posse do Projeto Básico que integra todo o sistema de uma forma simplificada, para avaliar as conseqüências em outros subsistemas bem como, tomar as providências necessárias (por exemplo: comandar a alteração nos tipos de dados associados a Porta de um outro subsistema que possui uma Interface alterada). Essa gerência eficiente transforma a etapa de integração em livre de problemas.

A Especificação Detalhada dos Subsistemas Básicos, feita em uma segunda etapa do curso do Design, permite a definição posterior de algumas características do sistema, o que beneficia o desenvolvimento pois, raramente, são conhecidos todos os detalhes no início de um projeto, principalmente quanto a interligação com sensores. Aplica-se desta forma uma política de Desenvolvimento Incremental que resulta em um sistema que melhor se adapta às necessidades do usuário.

No desenvolvimento do TTI, estas técnicas foram testadas e o resultado obtido considerado satisfatório. Permitiu facilmente o acréscimo de novas funções sem que isto implicasse em alterações de porte no que já havia sido desenvolvido. A taxa de erros detectados, após sua implantação, foi pequena e praticamente sem erros de especificação.

Glossário

Nome: Acompanhamento

Definição: Observação da trajetória de um alvo, armazenando suas posições e calculando rumo e velocidade.

Nome: Categoria

Definição: Composta das Categorias Primária e Secundária

Nome: Categoria Primária

Definição: Identifica o grau de hostilidade de um Acompanhamento, isto é, define um alvo como amigo, inimigo ou desconhecido.

Nome: Categoria Secundária

Definição: Informação referente a um Acompanhamento que o identifica com relação ao tipo de alvo, isto é, se de superfície, submarino ou aéreo, acrescido de detalhes, quando possível, de seu tipo; por exemplo, se o navio de superfície é uma Fragata. Esta informação é representada por até duas letras, conforme norma estabelecida.

Nome: Eco Radar

Definição: Corresponde à reflexão da onda eletromagnética, na superfície de um corpo, emitida pelo equipamento radar para uma dada posição do espaço.

Nome: Enlace Automático de Dados (EAD)

Definição: Hardware que, ligado ao equipamento de rádio do navio, permite a comunicação entre sistemas automáticos de dados de dois ou mais navios.

Nome: Esfera de Acompanhamento

Definição: Acoplada a um cursor gráfico, similar a "joystick" ou "mouse", também conhecida como "Tracker-Ball". Tem como dispositivo uma esfera que pode girar em qualquer direção mas de posição fixa no console. A movimentação dessa esfera reflete na movimentação do cursor gráfico.

Nome: Especificação Operativa

Definição: Documento de especificação de um sistema onde procura-se evidenciar todos os serviços executados pelo sistema, as entradas necessárias para executar um determinado serviço, as respostas por ele produzidas, a definição de termos, as condi-

ções do funcionamento e qualquer outra informação relevante para o entendimento do sistema, por parte da equipe responsável pela sua implementação.

Nome: Filtros de Apresentação

Definição: Conjunto de comandos que selecionam o que deve ser apresentado no vídeo sintético. Isto facilita o uso do mesmo terminal para diversas aplicações, tais como acompanhar alvos aéreos ou de superfície.

Nome: GIRO

Definição: Agulha Giroscópica. Usada para determinar o rumo do navio em relação ao Norte Verdadeiro.

Nome: Interceptação

Definição: Solução Tática com a finalidade de determinar os dados cinemáticos de um alvo visando atingir a mesma posição que outro.

Nome: IFF

Definição: Equipamento que permite a identificação do grau de periculosidade de um alvo - "Identification Friend or Fold"

Nome: Linhas de Marcação

Definição: Equivalem a Acompanhamentos dos quais só se conhece sua marcação e, portanto, representada por uma semi-reta, com origem no ponto onde foi detetado o alvo, na marcação conhecida.

Nome: Marcação

Definição: Angulo em relação ao Norte Verdadeiro.

Nome: Movimento Relativo

Definição: Apresentação das informações do radar com referencial no Próprio Navio.

Nome: Movimento Verdadeiro

Definição: Apresentação das informações do radar com referencial na Terra.

Nome: NA

Definição: Abreviatura para Número de Acompanhamento.

Nome: Número de Acompanhamento

Definição: Conjunto de quatro dígitos octais que identificam univocamente um Acompanhamento.

Nome: ODOM

Definição: O mesmo que Odômetro.

Nome: Odômetro

Definição: Equipamento capaz de medir velocidade através da rotação dos eixos do motor.
/* mede também distância, que não é usada */

Nome: Ponto de Maior Aproximação (PMA)

Definição: Solução Tática que indica o ponto mais próximo na trajetória de dois alvos.

Nome: Quadro Tático

Definição: O mesmo que Panorama Tático.

Nome: READOUT

Definição: Região do vídeo gráfico de apresentação alfanumérica contendo o resultado de um comando do operador.

Nome: Repetidora

Definição: Local onde é apresentada uma imagem de vídeo proveniente do Radar.

Nome: Softkeys

Definição: Forma de implementar um esquema de menus onde cada opção do menu é escolhida por uma tecla associada. Isto é, a escolha de uma opão do menu, é feita pressionando-se a tecla associada. No vídeo, são apresentados, em região específica, os valores correntes de cada opção de menu.

Nome: Vídeo Bruto

Definição: Informações dos ecos provenientes do radar em forma de vídeo.

Nome: Vídeo Sintético

Definição: Apresentação gráfica contendo as informações do panorama tático. É constituida de simbolos específicos sobrepostos ao vídeo bruto.

Lista de Abreviaturas

Acomp	-	Acompanhamento
Alfa	-	Alfanumérico(a)
Apres	-	Apresentação
At	-	Atualização
Cat	-	Categoria
Cmdo	-	Comando
Ctr	-	Controle
DC	-	Diagrama de Comportamento
DE	-	Diagrama de Estrutura
EA	-	Esfera de Acompanhamento
EAD	-	Enlace Automático de Dados
Exec	-	Executa
Ger	-	Gerencia
GIRO	-	Agulha Giroscópica
Graf	-	Gráfico
Info	-	Informação
IFF	-	Equipamento de "Identification Friend or Fold"
IUM	-	Interface Usuário-Máquina
Lim	-	Limite
Man	-	Manual
Mov	-	Movimenta
NA	-	Número de Acompanhamento
Nav	-	Navio
NAVSAT	-	Equipamento de Navegação por Satélite
ODOM	-	Odômetro
Phist	-	Pontos Históricos
PMA	-	Ponto de Maior Aproximação

Pos	-	Posição
Rad	-	Radar
Resp	-	Resposta
Sel	-	Seleção
Sens	-	Sensor(es)
Sint	-	Sintético
Sol	-	Solução
St	-	Status
Tat	-	Tático(a)
TTI	-	Terminal Tático Inteligente
Velcc	-	Velocidade

OBS: Os nomes dos dados e dos subsistemas foram construídos fazendo-se uso de abreviaturas, para evitar a sobrecarga das figuras. Foram utilizados "_" no lugar de preposições. Por exemplo :

- Info_At_Acomp deve ser lido como Informações Atualizadas de Acompanhamento;
- Ctr_Vídeo_Sint deve ser lido como Controle de Vídeo Sintético.

Agradecimentos

Os autores agradecem a colaboração de Bruno Maffeo na confecção e revisão do capítulo 2.

Bibliografia

[1] André A.A. e Maffeo B.; Projeto ("Design") na Área de Métodos Estruturados: Transformando Redes em Hierarquias; Monografias em Ciência da Computação, Departamento de Informática, PUC-Rio, 7/91

[2] Bech K. e Cummingham W.; A Laboratory for Teaching Object-Oriented Thinking; Proceedings of the 1989 OOPSLA - Conference on Object Oriented Programming Systems, Languages and Applications, reprinted in SIGPLAN Notices

[3] Booch G.; Object-Oriented Design with Applications; Redwood City, CA, Benjamim Cummings, 1991

[4] Booch G.; Object-Oriented Development; IEEE Transactions on Software Engineering, Vol. SE-12, fevereiro de 1986

[5] Bruyn W., Jensen R., Keskar D. e Ward P.T.; ESML: An Extended System Modelling Language Based on Data Flow Diagram; ACM Sigsoft, Software Engineering Notes, 13(1), janeiro de 1988

[6] Budd T.; An Introduction to Object Oriented Programming; Addison Wesley, 1991

[7] Bustard D., Eldore J. e Welsh J.; Concurrent Program Structure; Prentice Hall, 1988

[8] Chen P.P.S.; The entity-relationship model - toward a unified view of data; ACM Transactions on Database Systems, 1(1): 9-36, 1976

[9] Clemente K.; Modelagem de Sistemas Sócio-Técnicos, Estudo de Caso de um Piloto Automático para Automóvel; Tese de Mestrado, Departamento de Engenharia Elétrica, PUC-Rio, abril de 1992

[10] Coleman D., Hayes F. e Bear S.; Introduction to Objectcharts or How to use Statecharts in Object-Oriented Design; IEEE Transactions on Software Engineering, Vol. 18, nº 1, janeiro de 1992

[11] DeMarco T.; Structured Analysis and System Specification; Yourdon Press, 1978

[12] Denning P.J., Dennis J.B. e Qualitz J.E.; Machine

Languages and Computation; Englewood Cliffs, N.J. Prentice-Hall, 1978

[13] Fairley R.E.; Software Engineering Concepts; MacGraw-Hill, 1985

[14] Harel D.; Statecharts: a visual formalism for complex systems; Sci Computer Program, Vol. 8, pp 231-274, 1987

[15] Henderson W. e Taylor P.G.; Embedded Processes in Stochastic Petri Nets; IEEE Transactions on Software Engineering, Vol. 17, nº 2, fevereiro de 1991

[16] Heuser C.A. e Richter G.; Application and Theory of Petri Nets; 13th International Conference Sheffield, UK, 22-26, Proceedings, junho de 1992

[17] Heuser C.A.; Modelagem Conceitual de Sistemas - Redes de Petri; Publicação da V EBAI

[18] Jackson M.A.; Principles of Program Design; Academic Press, New York, 1975

[19] Keller D.; A Guide to Natural Naming; SIGPLAN Notices, 25(5): 95-102, maio de 1990

[20] Kramer J. e Sloman M.; Distributed Systems and Computer Networks; Prentice Hall, 1987

[21] Liskov B. e Guttag J.; Abstraction and Specification in Program Development; MacGraw-Hill, 1986

[22] Liskov B. e Zilles S.; Specification Techniques for Data Abstraction; IEEE Transactions on Software Engineering, Vol. SE-12, fevereiro de 1986

[23] Maffeo B. e Ritto A.C.A.; Definindo o Problema a ser Tratado por um Sistema Computacional - Modelo do Contexto; Monografias em Ciência da Computação, Departamento de Informática, PUC-Rio, 11/91

[24] Maffeo B.; ESML: Uma Revisão de Apresentação, Estrutura, Notação e Conteúdo; Monografias em Ciência da Computação, Departamento de Informática, PUC-Rio, 1/91

[25] Maffeo B.; Engenharia de Software e Especificação de Sistemas; Editora CAMPUS, 1992

[26] Magee J., Kramer J. e Sloman M.; An Overview of the REX Software Architecture; IEEE Computer Society Workshop on Future Trends of Distributed Computing Systems, Cairo, outubro de 1990

[27] Magee J., Kramer J. e Sloman M.; CONIC : An Integrated Approach to Distributed Computer Control Systems; IEEE

Proc. Vol. 130, Pt. E, nº 1, janeiro de 1983

[28] Magee J., Kramer J. e Sloman M.; Constructing Distributed Systems in CONIC; Imperial College Research Report -Doc 87/4, 16 de março de 1987

[29] Magee J., Kramer J. e Sloman M.; The CONIC Support Environment for Distributed Systems; NATO ASI series, agosto de 1986

[30] Magge J., Kramer J. e Sloman M.; Constructing Distributed Systems in Conic; IEEE Transactions on Software Engineering, Vol. SE-15, junho de 1989

[31] Manual de Normas do Centro de Apoio a Programação, Publicação Interna da Diretoria de Armamento e Comunicações da Marinha do Brasil

[32] Orr K.T.; Structured Systems Development; Yourdon Press; 1977

[33] Peterson J.L.; Petri Net Theory and the Modeling of Systems; Englewood Cliffs, N.J., Prentice Hall, 1981

[34] Petri C.A.; Kommunikation mit Automaten; PhD thesis, Technische Hochschule Darmstadt, 1961

[35] Ramamoorthy C.V., Garg V. e Prakash A.; Programming in Large; IEEE Transactions on Software Engineering, Vol. SE-12, fevereiro de 1986

[36] Reisig W.; Petri Nets - An Introduction; Springer-Verlag, Berlin Heidelberg, 1985

[37] Richter G. e Maffeo B.; Towards a rigorous interpretation of ESML - Extended System Modeling Language; aceito para publicação em IEEE Transactions on Software Engineering, 1992

[38] Richter G.; An Introduction to Office Modellings with Informal and Formal Nets; GMD, Informatik Spektrum (1983) 6:210-220 e (1984) 7:28-40

[39] Ross D.; Structured Analysis (SA): A Language for Comunicating Ideas; Transactions on Software Engineering, Vol. SE-3, nº 1, janeiro de 1977

[40] Sanchez M.L.A. e Maffeo B.; Gerência de Projeto ("Design") Orientado a Encapsulamento de Dados e a Troca de Mensagens entre Subsistemas Autônomos; Monografias em Ciência da Computação, Departamento de Informática, PUC- Rio; 1991

[41] Sanchez M.L.A. e Maffeo B.; Projeto ("Design") Orientado a Encapsulamento de Dados e a Troca de Mensagens entre

Subsistemas Autônomos; Monografias em Ciência da Computação, Departamento de Informática, PUC-Rio; 1991

[42] Sanchez M.L.A.; Especificação Operativa de um Terminal Tático Inteligente; Publicação Interna do Instituto de Pesquisas da Marinha do Brasil; 1989

[43] Stevens W., Meyers G. e Constantine L.; Structured Design; IBM System Journal, Vol. 13, nº 12, Maio 1974, pp 115-139

[44] Tanenbaum A.S.; Operating Systems: Design and Implementation; Englewood Cliff, N.J., Prentice Hall, 1987

[45] Ward P.T. e Mellor S.J.; Structured Development for Real-Time Systems; Yourdon Press, 1985

[46] Warnier J.D.; Lógica de Construção de Sistemas; tradução de Luiz Paulo Bastos Abrahão; editora CAMPUS/DATAMEC, 1984

[47] Wirfs-Brock R. e Wilkerson B.; Object Oriented Design: A Responsibility Driven Approach; Proceedings of the 1989 OOPSLA - Conference on Object-Oriented Programming Systems, Languages and Applications, reprinted in SIGPLAN