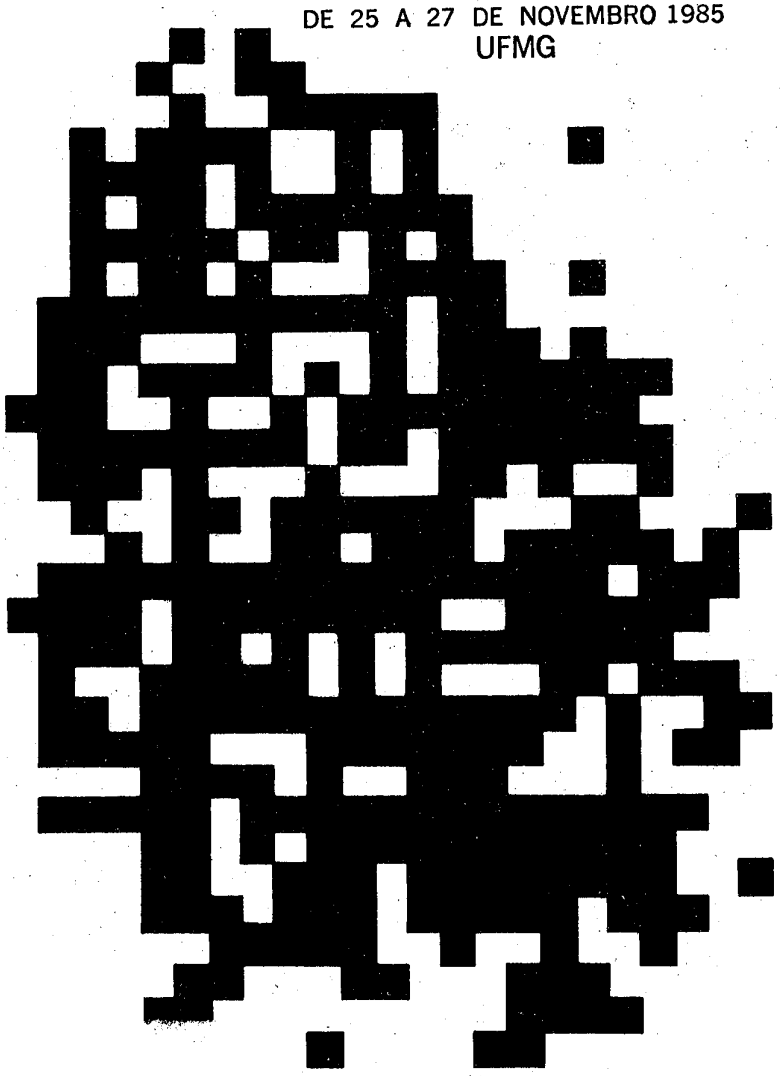
# V SIMPÓSIO

## SOBRE DESENVOLVIMENTO DE SOFTWARE BÁSICO

DE 25 A 27 DE NOVEMBRO 1985
UFMG





# ANAIS

# 5º SIMPÓSIO SOBRE DESENVOLVIMENTO DE SOFTWARE BÁSICO

BELO HORIZONTE 25 A 27 DE NOVEMBRO DE 1985

# A N A I S

PROMOÇÃO: SOCIEDADE BRASILEIRA DE COMPUTAÇÃO - SBC
, COMISSÃO ESPECIAL PARA LINGUAGENS E SISTEMAS DE PROGRAMAÇÃO
UNIVERSIDADE FEDERAL DE MINAS GERAIS - UFMG

PATROCÍNIO: CONSELHO NACIONAL DE DESENVOLVIMENTO CIENTÍFICO E TECNOLÓGICO - CNPq

# UMA LINGUAGEM DE CONFIGURAÇAO PARA O USO DE MODULA-2 PARA PROGRAMAÇAO EM AMBIENTES DISTRIBUIDOS

Lidia Segre *

Michael Stanton #

* COPPE/UFRJ
Programa de Sistemas
Caixa Postal 68511
21945 Rio de Janeiro, RJ

# Departamento de Informàtica
PUC-RJ
Rua Marquês de São Vicente, 225
22453 Rio de Janeiro, RJ

Resumo

Este trabalho apresenta uma proposta de uma linguagem para especificar a configuração estàtica de um programa distribuido a partir de um conjunto de mòdulos escritos em uma extensão da linguagem Modula-2.

## I - Introdução

Sistemas de computação grandes, tais como muitos sistemas de controle de processos, precisam ter uma vida mèdia longa. Estes sistemas estão sujeitos a mudanças de tipo operacional, tais como, por exemplo, falhas, relocação de componentes ou redimensionamento, e a mudanças de tipo evolucionàrio, devido a novos requisitos das aplicações, novas necessidades humanas e novas tecnologias.

Para facilitar a construção e modificação de sistemas de software grandes è necessàrio decompor o sistema em componentes que possam ser programados, compilados e testados separadamente. Para poder especificar como serão integrados os componentes para formar o programa, precisamos ter ferramentas, usualmente na forma de uma linguagem de programação em larga escala (LPL),

geralmente distinta da linguagem de programação em pequena escala (LPP) usada para escrever os componentes [De Renner 1976].

Nos sistemas centralizados, a construção modular de programas significa que um programa é montado a partir de um conjunto de módulos desenvolvidos e compilados separadamente. A especificação dos módulos que compõem um programa, e de suas ligações, é chamada configuração, e linguagens como C/Mesa [Mitchell 1979] foram definidas para formalizar esta tarefa.

Para sistemas distribuidos, a configuração precisa incluir também a localização física dos módulos nas diferentes estações. Outra diferença entre sistemas centralizados e distribuidos é que nestes últimos a configuração do hardware poderá ser alterada dinamicamente, obrigando portanto uma reconfiguração do software. A linguagem Conic [Sloman 1984] se destina a configuração de software em sistemas distribuidos.

Portanto num sistema distribuido definimos através da LPL a configuração do programa, que consiste da especificação de:

- os módulos que comporão o programa;
- as ligações de comunicação entre os componentes;
- a localização dos componentes.

Modula-2 é uma linguagem adequada como LPP em sistemas centralizados, já que permite o desenvolvimento e compilação em separado dos módulos, e define adequadamente as interfaces entre os diferentes módulos através de mecanismos de importação e exportação de listas de declarações procedurais, os quais são requisitos necessários para poder configurar sistemas grandes.

E possivel definir extensões a Modula-2, tais como as propostas em [Segre 1985] em relação aos conceitos de paralelismo e confiabilidade, que a tornam adequada para o uso como uma LPP em sistemas distribuidos.

Pode-se identificar dois esquemas de descrição de especificação: o estático e o dinâmico. No estático, supõe-se que, a partir do início da execução de um programa, permanecem inalterados os detalhes da sua configuração. O esquema dinâmico, por outro lado, permite que a configuração seja modificada após o início da execução do programa, e ainda é objeto de estudos pelos autores. Apresentamos aqui, então, apenas o esquema de configuração estática, que pode ser representado pela figura 1:

A partir de um conjunto de módulos já compilados e de uma especificação de configuração o Construtor cria unidades de carga formadas por grupos de módulos para cada uma das estações do sistema distribuido. Este Construtor é análogo ao ligador dos sistemas de programação sequencial.

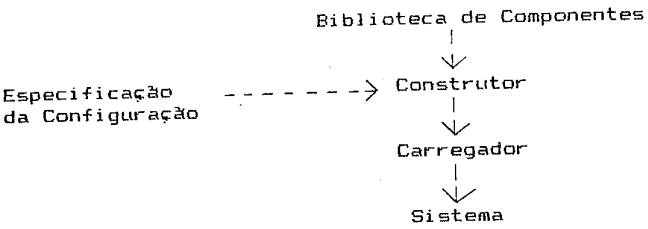



Figura 1: Processo de Configuração Estática

As extensões a Modula-2 apresentadas em [Segre 1985] junto com a linguagem de configuração proposta aqui servirão de base para a construção de um ambiente de programação distribuida em Modula-2.

II - Descrição da linguagem de configuração

A primeira extensão a ser introduzida para poder definir uma configuração de um sistema distribuido é o conceito de tipo de módulo, que em Modula-2 não existe. Este conceito é necessário para poder criar várias instâncias de um mesmo módulo em estações distintas ou até na mesma estação.

Outros conceitos utilizados na configuração são os de estação lógica e estação física. Uma estação lógica é composta por um conjunto de módulos interligados que formam uma unidade lógica a ser carregada numa estação física. Numa estação física podem estar carregadas uma ou mais estações lógicas. A linguagem de configuração proposta permite descrever configurações hierárquicas; isto significa que ela define configurações compostas por módulos programados em Modula-2 ou por (sub-) configurações já definidas anteriormente. Tanto os módulos quanto as configurações são tipos, a partir dos quais podem ser criadas várias instâncias.

Descreveremos a seguir a especificação da linguagem de configuração que desejamos implementar para montar o ambiente distribuido baseado na linguagem Modula-2. A sintaxe e a semântica da linguagem de configuração serão descritas de maneira informal e no final será apresentado um exemplo que ajudará a ilustrar seu uso e alcance.

A sintaxe da definição de uma configuração se inicia com a seguinte declaração:

<identificador do
tipo da configuração> : CONFIGURATION

por exemplo:

PatientConf : CONFIGURATION

define um tipo de configuração com nome PatientConf

Cada configuração è composta por mòdulos ou sub-configurações que pertencem a determinadas estações lògicas. Essas estações devem ser declaradas da seguinte forma:

STATIONS < lista de estações lògicas>

por exemplo:

STATIONS Bed , Nurse

A lista de estações pode conter um ou mais elementos.

Em Modula-2 os mòdulos interagem através de interfaces bem definidas, geralmente compostas de listas de procedimentos exportados. Estes serão chamados por outros mòdulos que precisam importar explicitamente as interfaces que os contêm.

No nível de configuração de um sistema composto de mòdulos escritos em Modula-2, è então necessàrio associar esses procedimentos às instàncias adequadas dos mòdulos criados. As interligações dentro das configurações e entre diferentes configurações são feitas através de importações e exportações de mòdulos que são declarados explicitamente depois da declaração STATIONS. Estas declarações são opcionais, isto è, nem todas as configurações importam ou exportam mòdulos; por exemplo, a configuração de mais alto nível, numa estrutura hieràrquica, não importa nem exporta mòdulos.

A sintaxe destas declarações è a seguinte:

IMPORT <lista de mòdulos>;

EXPORT <lista de mòdulos>;

Na mesma configuração esta ordem das declarações deve ser respeitada.

Por exemplo em configurações diferentes poderiam estar as duas declarações seguintes:

Na configuração NurseConf:

```
EXPORT    Selector : psel;
          ConsoleNurse : zterm
```

e na configuração PatientConf:

```
IMPORT    Select : psel;
          ConsoleN : zterm
```

Numa fase posterior, podemos então ligar uma ou mais instâncias de PatientConf a uma instância de NurseConf, casando assim as listas de exportação e importação.

A declaração indica o nome do módulo e seu tipo. Na exportação o nome do módulo corresponde a um módulo cuja instância é criada na configuração à qual pertence a declaração de exportação. Na importação o tipo do módulo corresponde a um módulo pertencente a outra configuração, e o nome usado tem as características de um parâmetro formal do módulo que declara a importação.

A próxima declaração define as instâncias de módulos ou de sub-configurações que compõem a configuração, explicitando também a alocação lógica.

A sintaxe da declaração CREATE é a seguinte:

CREATE <lista de criação de instâncias de módulos>
ou
CREATE <lista de criação de instâncias de sub-configurações>

A lista de criação de instâncias de módulos é composta por declarações com a seguinte sintaxe:

```
<um ou mais nomes       tipo do
 de módulos>       :   Módulo     ON   nome de estação lógica
```

por exemplo:

```
CREATE
    ErrorWindow, Window : kwind ON Bed
    Scanner : psim ON Bed
    AlarmN : palarm ON Nurse
```

A declaração CREATE cria várias instâncias de módulos de tipos definidos, (por exemplo, a primeira declaração cria duas instâncias do mesmo tipo), indicando em todos os casos a sua alocação lógica.

A lista de criação de instâncias de sub-configurações é composta por declarações com a seguinte sintaxe:

```
<um ou mais nomes        tipo  da            <lista   de
configurações>      : configuração  ON   estações lógicas>
```

A lista de estações lógicas deve conter as estações na ordem e número correspondentes às estações declaradas anteriormente na declaração STATIONS na declaração do tipo da configuração.

por exemplo:

```
CREATE
    Maria : NurseConf ON Nurse
    Pat1  : PatientConf ON Bed1, Nurse
```

A configuração do tipo PatientConf foi declarada com duas estações lògicas; por isto, na hora de criar instâncias de configurações desse tipo, precisa-se associar a cada instância duas estações lògicas.

Depois de ter definido as instâncias de màdulos ou sub-configurações que compàem a configuração è necessàrio estabelecer as ligações entre importadores e exportadores. Isto è realizado atravès da declaração LINKS que tem o seguinte formato:

LINKS < lista de ligações de instâncias de màdulos>
ou
LINKS <lista de ligações de instâncias de sub-configurações>

A lista de ligações de instâncias de màdulos està composta por declarações com a seguinte sintaxe:

<um ou mais nomes de màdulos> WITH <um ou mais nomes de màdulos>

Esta declaração liga uma instância de cada um dos màdulos importadores à esquerda do WITH com as instâncias dos màdulos exportadores indicados à direita do WITH. Estes últimos são os que exportam as interfaces que são importadas pelos màdulos especificados à esquerda do WITH.

A associação de exportadores com as interfaces importadas utiliza o modelo posicional, isto è, a lista de nomes de màdulos à direita do WITH deve corresponder em número, ordem e tipo à lista de importações de màdulos declarada dentro do tipo de màdulo sendo ligado. Quando duas declarações WITH tiverem listas de màdulos à direita que são idênticas, estas poderão ser combinadas, colocando uma lista dos nomes das instâncias importadoras à esquerda de WITH.

Os màdulos nomeados depois da palavra WITH podem ou não estar na mesma estação lògica ou configuração que os màdulos nomeados à esquerda da palavra WITH. Dependendo da alocação fisica dos màdulos estas ligações podem ser locais ou remotas. Isto serà definido no final da configuração quando for especificado o mapeamento entre a configuração lògica e a configuração fisica.

Exemplo:

```
LINKS
    Command  WITH  Monitor, Console
    Window   WITH  Console
    Monitor  WITH  Select , Alarm
```

A lista de ligações de instâncias de sub-configurações é composta por declarações com a seguinte sintaxe:

```
<um ou mais nomes                          <um ou mais nomes de
 de configurações>        WITH             módulos qualificados>

nome de                          nome de          {.nome de          nome
módulo          ::=          configuração     configuração}.          de
qualificado                                                          módulo
```

Esta declaração liga uma instância de cada uma das configurações que aparecem à esquerda do WITH a um ou mais módulos que serão determinados através da identificação da hierarquia de configurações à qual pertencem. O uso de listas à esquerda ou à direita de WITH tem o mesmo significado que no caso de ligação de instâncias de módulos, com a evidente ressalva que a lista de módulos à direita corresponde posicionalmente à lista de importações da instância de configuração sendo ligada.

Exemplo:

```
LINKS
     Pat1, Pat1 WITH Maria.Selector, Maria.ConsoleNurse
```

Cabe ao Construtor verificar que os módulos citados como exportadores nas declarações LINKS realmente sejam compatíveis com as interfaces importadas pelos módulos e configurações importadores.

Para ter uma idéia global da definição de configuração podemos mostrar o seguinte esquema informal. Uma declaração de configuração será definida por:

```
<nome do tipo de configuração> : CONFIGURATION
   STATIONS <lista de estações>
   [IMPORT <lista de módulos de importação>]
   [EXPORT <lista de módulos de exportação>]
   CREATE  <lista de criação de instâncias de módulos ou de
            instâncias de sub-configurações>
   LINKS  <lista de ligações de instâncias de módulos ou de
           instâncias de sub-configurações>
END <nome do tipo de configuração>
```

O último passo para concluir a configuração é especificar o mapeamento entre a configuração lógica e a configuração física, ou seja, definir a alocação física dos módulos que compõem a configuração. Isto pode ser feito usando a seguinte declaração:

LOAD nome da configuração : nome do tipo de configuração
ON <lista de estações físicas>

A lista de estações físicas pode conter nomes simbólicos ou endereços físicos, mas ela deve corresponder em número e ordem à lista de estações declaradas na definição da configuração.

Exemplo:

```
LOAD  Ward : WardConf ON  Station1, Station2, Station3, Station4,
                          Station5, Station6
```

III - Um exemplo completo

Vamos agora apresentar o exemplo escolhido para ser especificado na nossa proposta de linguagem. O exemplo introduzido por Stevens, Myers e Constantine [Stevens 1974] é o seguinte:

"Um hospital precisa de um sistema de controle de pacientes. Cada paciente é controlado por um dispositivo análogo que mede fatores tais como pulso, temperatura, pressão do sangue e resistência da pele. O sistema lê esses fatores periodicamente. Para cada paciente são especificados limites de variação de segurança para cada fator. Se um fator tomar um valor fora do limite de variação definido ou se um instrumento falhar, a estação da enfermeira é notificada".

Este exemplo já foi desenvolvido na linguagem CONIC [Magee 1984] e nós o reestruturamos para poder ser escrito na linguagem Modula-2, usando a linguagem de configuração proposta.

Este projeto decompõe a função do sistema em sub-funções que podem ser associadas a módulos exportando e importando interfaces específicas.

O esquema físico do sistema de controle pode ser representado pela figura 2:

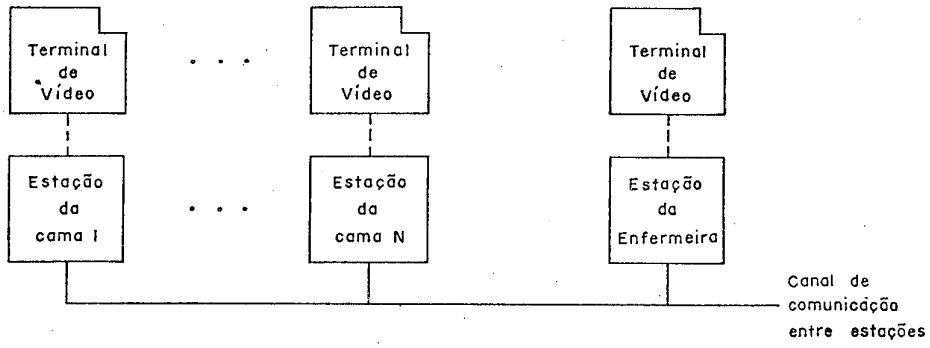



Figura 2: Sistema de controle de pacientes

As estações das camas são todas iguais contendo um terminal de video onde são mostrados os valores dos fatores que estão sendo controlados e através do qual são introduzidos os limites de variação de segurança. A estação da enfermeira emite alarmes, e pode querer repetir no seu terminal de video os valores dos fatores que estão sendo controlados em cada paciente.

Cada estação lògica, que pode ou não coincidir com uma estação física, è constituida por um conjunto de mòdulos interligados que executam as funções da estação.

Para especificar este sistema definimos três tipos de configurações, uma correspondente à Configuração de Paciente, da qual serão criadas tantas instâncias quantas camas existem, uma Configuração de Enfermeira e uma Configuração de Enfermaria, composta pelas instâncias criadas a partir das duas configurações já mencionadas.

A Configuração de Enfermeira utiliza unicamente a estação lògica da Enfermeira. A Configuração de Paciente precisa de duas estações lògicas, a dele e a da enfermeira, já que algumas das funções são executadas perto da cama e outras são executadas na estação da enfermeira, tais como acionar o alarme e mostrar os dados do paciente no video da enfermeira quando estes forem requisitados. Estas funções precisam de mòdulos especificos para cada cama.

Vamos então mostrar a especificação da configuração do sistema usando nossa linguagem de configuração sem nos preocupar

em explicar os detalhes das funções de cada módulo que estão programados em Modula-2. A descrição completa do sistema seria muito grande e para efeitos deste trabalho achamos suficiente a abordagem escolhida.

Descrevemos a seguir, sucintamente, as funções dos tipos de módulos utilizados para formar a configuração do exemplo apresentado:

psim: este módulo simula as entradas dos fatores do paciente a serem controlados e importa a interface de pmonit.

pmonit: este módulo compara as leituras recebidas com os limites de variação e provoca um alarme quando o valor estiver fora do limite. Armazena todas as informações do paciente para poder fornecê-las quando forem solicitadas. Ele exporta uma interface para psim e importa as interfaces de palarm, de psel e de pdisp.

pdisp: este módulo recebe informações de um cliente para formatá-las e encaminhá-las para serem exibidas. Ele importa a interface de kwind e exporta uma interface para o cliente.

pcom: este módulo permite que sejam introduzidas novas informações do paciente através do console da cama do paciente. Ele importa a interface de pmonit e de zterm.

palarm: este módulo aceita as mensagens de alarme enviadas a partir das camas e as formata para que sejam exibidas. Ele exporta uma interface para pmonit e importa a interface de kwind.

kwind: este módulo formata a tela do console. Ele exporta uma interface para pdisp e zlogw e importa a interface de zterm.

zlogw: este módulo formata os erros para serem exibidos na tela. Ele importa a interface de kwind.

psel: este módulo requisita informações das camas e as armazena para tê-las disponíveis para pedidos da enfermeira. Ele exporta uma interface para nursecom e pmonit e importa a interface de pdisp.

nursecom: este módulo executa as funções de interpretação dos comandos que permitem à enfermeira escolher uma cama para monitorar. Ele importa as interfaces de zterm e de psel.

A Configuração do Paciente, considerando os módulos principais que a constituem, é a seguinte:

```
PatientConf : CONFIGURATION
  STATIONS  Bed,Nurse
  IMPORT  Select : psel,ConsoleN:zterm
  CREATE
      Console:zterm ON Bed
      Monitor:pmonit ON Bed
      Command:pcom ON Bed
      ErrorWindow, Window:kwind ON Bed
      Display:pdisp ON Bed
      Scanner:psim ON Bed
      ErrorFormat:zlogw ON Bed
      AlarmN:palarm ON Nurse
      WindowN:kwind ON Nurse

   LINKS
      WindowN WITH ConsoleN
      AlarmN WITH WindowN
      Monitor WITH Select,AlarmN,Display
      Command WITH Monitor,Console
      ErrorWindow,Window WITH Console
      Display WITH Window
      Scanner WITH Monitor
      ErrorFormat WITH ErrorWindow
END PatientConf
```

As ligações dentro desta configuração ligam módulos locais entre eles e módulos desta configuração com módulos de outras configurações explicitados na lista de importações. Os nomes dos módulos importados são locais a este tipo de configuração e serão substituidos por módulos reais na hora de definir as ligações das configurações, como será visto mais na frente.

Convêm destacar que, já que as estações lógicas Bed e Nurse serão mapeadas em estações físicas diferentes, a terceira declaração de LINKS estabelece uma ligação remota, já que o módulo AlarmN pertence à estação Nurse, enquanto Monitor pertence à estação Bed. Além disto a ligação de Monitor com Select è potencialmente remota, mas só saberemos isto quando for ligada esta (sub-)configuração com outra no próximo nível da hierarquia. A primeira declaração è local porque os dois módulos pertencem à mesma estação NURSE.

A Configuração de Enfermeira, considerando os módulos principais que a constituem, è a seguinte:

```
NurseConf:CONFIGURATION
STATIONS Nurse
EXPORT Selector:psel,ConsoleNurse:zterm
CREATE
    Command:nursecom ON Nurse
    Display:pdisp ON Nurse
    Selector:psel ON Nurse
    ConsoleNurse:zterm ON Nurse
```

```
    ErrorFormat:zlogw ON Nurse
    ErrorWindow,Window:kwind ON Nurse

LINKS
    Window,ErrorWindow WITH ConsoleNurse
    ErrorFormat WITH ErrorWindow
    Display WITH Window
    Command WITH Selector,ConsoleNurse
    Selector WITH Display
END NurseConf
```

Passaremos agora a definir a Configuração de Enfermaria que è composta por instàncias das Configurações jà declaradas.

```
WardConf:CONFIGURATION
STATIONS Bed1,Bed2,Bed3,Bed4,Bed5,Nurse
CREATE
   Maria:NurseConf ON Nurse
   Pat1:PatientConf ON Bed1,Nurse
   Pat2:PatientConf ON Bed2,Nurse
   Pat3:PatientConf ON Bed3,Nurse
   Pat4:PatientConf ON Bed4,Nurse
   Pat5:PatientConf ON Bed5,Nurse

LINKS
   Pat1,Pat2,Pat3,Pat4,Pat5
   WITH Maria.Selector,Maria.ConsoleNurse
END WardConf
```

O sistema aqui descrito è composto por uma enfermeira controlando cinco camas de pacientes. Podemos ressaltar que a configuração de mais alto nìvel nàо exporta nem importa nada. É nesta configuração que são feitas as ligações entre as Configurações de Pacientes e a Configuração de Enfermeira atravès da associação entre os mòdulos importados pelos pacientes e exportados pela enfermeira.

## IV - Aspectos da implementação

### a) Gerenciamento de programas

O gerenciamento de programas, que trata do carregamento e ligação de mòdulos escritos em Modula-2 e armazenados numa biblioteca na forma compilada, è dividido numa parte local, replicada em cada estação, e numa parte nàо local. A parte nàо local, que pode ser implementada de forma centralizada, cuida de aspectos globais do gerenciamento de configuração, tais como a interpretação das pròprias configurações, e o armazenamento em memòria secundària destas e dos mòdulos utilizados para compô-las. A partir de uma especificação de configuração, o ligador

gera uma sèrie de operações de carga de mòdulos nas estações especificadas, junto com a informação sobre as ligações que devem existir entre estes, em forma de tabelas a serem executados pelo carregador global.

Ao executar um comando LOAD o carregador global interpreta as operações geradas pelo ligador resultando no carregamento de instâncias de mòdulos em estações remotas ("downline loading") e no fornecimento a estas estações de informação sobre as ligações locais e não locais de cada mòdulo carregado. No caso das ligações não serem locais, a comunicação entre os mòdulos utilizarà a chamada remota de procedimentos (RPC) atravès do uso de "stubs", como descrito em [Segre 1985]. Neste caso, serão carregados os "stubs" apropriados, e ao "RPC runtime" de um servidor serão fornecidas as informações sobre a disponibilidade do servidor para receber chamadas remotas.

Instâncias de mòdulos são obtidas a partir de tipos de mòdulos guardados num sistema de armazenamento em memòria secundària (sistema de arquivos). A linguagem Modula-2 não define tipos de mòdulos, mas podemos suprir esta falta facilmente da seguinte maneira, sem alterar a sintaxe da linguagem. Modula-2 requer que o nome de um mòdulo de implementação seja o mesmo que do mòdulo de definição correspondente. Convencionalmente, guardamos o mòdulo de implementação num arquivo com este mesmo nome. Para alcançar nossas metas, permitimos que os nomes dos arquivos contendo o còdigo objeto de um mòdulo de implementação sejam diferentes do nome usado na sua declaração em Modula-2. Logo a um mòdulo de implementação associamos dois nomes: um que corresponde ao mòdulo de definição (sua interface) e outro que corresponde ao arquivo que o contèm. Este segundo nome serà utilizado como o nome do tipo de mòdulo para criar vàrias instâncias dele.

Para permitir o uso de màltiplas instâncias de um tipo de mòdulo de implementação, criamos um segmento de dados pròprio para cada instância. Frisamos que pode ser usada a mesma còpia do segmento de còdigo por todas as instâncias de um tipo de mòdulo carregadas na mesma estação.

Os serviços oferecidos pelo ligador, carregador global, sistema de arquivos e gerador de "stubs" em princìpio podem estar disponìveis em uma ùnica estação da rede - uma espècie de estação de controle. Porèm, estes serviços poderão tambèm ser providos de forma distribuida, o que poderà oferecer maior robustez e eficiência no processamento. Para maior simplicidade, a primeira implementação desta proposta deverà centralizar o gerenciamento de programas.

b) Gerenciamento local

Este nìvel do software bàsico de cada estação trata de prover serviços para o gerenciamento local de programas, tais como a carga de instâncias de mòdulos iniciada externamente

("down-line loading") pelo carregador global, e a manutenção de informações sobre suas ligações. Estas informações são usadas para efetuar acessos a variáveis e procedimentos importados, inclusive aos procedimentos chamados remotamente.

## V - Conclusões

Este trabalho apresentou uma proposta de linguagem de configuração estática para desenvolvimento de software distribuido a ser programado usando as extensões de Modula-2 descritas em [Segre 1985]. Também foi esboçada uma implementação desta linguagem.

Adiamos a consideração de configuração dinâmica por acharmos que ela poderá ser implementada como uma extensão à configuração estática. Concentramos, portanto, nossos esforços na definição do esquema estático e gostaríamos de ganhar mais experiência com este antes de estendê-lo para o esquema dinâmico.

Parece-nos importante explorar a interação entre o tratamento de falhas como apresentado em [Segre 1985] e a configuração dinâmica de forma que seja possível programar respostas automáticas a ocorrência de falhas que incluam reconfiguração.

A pesquisa descrita aqui faz parte de um projeto maior sobre ambientes de desenvolvimento modular de software distribuido, e lhe será dado prosseguimento na direção apontada no último parágrafo.

**********

Os autores agradecem o apoio dado a esta pesquisa pela FINEP, EMBRATEL e CNPq.

## Bibliografia

[De Renner 1976] De Renner,F. e Kron,H.H., "Programming-in-the-large versus programming-in-the-small", IEEE Trans.Soft.Eng. SE-2,2 , p.80-86 (1976).

[Magee 1984] Magee,J.N., "Provision of Flexibility in Distributed Systems", Tese de Doutorado, Department of Computing, Imperial College, London(1984).

[Mitchell 1979] Mitchell J.G. et al., "Mesa language manual, version 5.0", Report CSL-79-3, Xerox PARC (1979).

[Segre 1985] Segre,L.M., Stanton,M.A., "Sobre o uso de Modula-2 para programação em ambientes distribuidos", submetido para apresentação no V Simpòsio sobre Desenvolvimento de Software Bàsico, a se realizar em Belo Horizonte, (Nov. 1985).

[Sloman 1984] Sloman,M. et al., "Building flexible distributed systems in Conic", Proc. SERC Distributed Computing 84 Conference, Bringhton (Set. 1984).

[Stevens 1974] Stevens,W.P., Myers,G.F. e Constantine,L.C., "Structured Design", IBM Systems Journal, Vol.13, No.2, p.115-139 (1974).