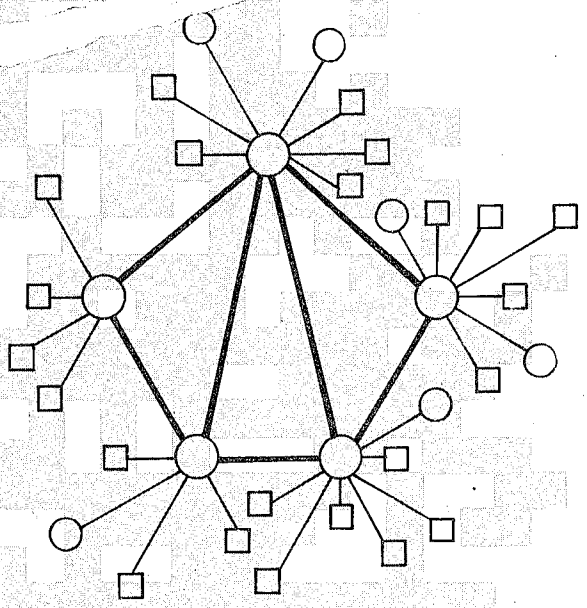
ANAIS

2° SIMPÓSIO BRASILEIRO SOBRE REDES DE COMPUTADORES

(2° SBRC)



Campina Grande
16 a 18 Abril 1984

ANAIS

2º SIMPÓSIO BRASILEIRO SOBRE REDES DE COMPUTADORES (2º SBRC)

16 a 18 DE ABRIL, 1984

CAMPINA GRANDE, PB

Promoção

Sociedade Brasileira de Computação (SBC)

Universidade Federal da Paraíba (UFPB)

Organização

Grupo de Redes de Computadores da UFPB (GRC/UFPB)

Patrocínio

Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq)

2º SIMPÓSIO BRASILEIRO SOBRE REDES DE COMPUTADORES (2º SBRC)

PROGRAMAÇÃO PARALELA: UMA EXPERIÊNCIA NA DEPURAÇÃO DE PROGRAMAS NA MAD

SELMA S. S. MELNIKOFF
Universidade de São Paulo
Escola Politécnica/Depto. de Engenharia de Eletricidade

CARLOS JOSÉ LUCENA
Pontifícia Universidade Católica do Rio de Janeiro
Depto. de Informática

RESUMO

Este trabalho tem por objetivo relatar uma experiência real na depuração de um programa paralelo projetado e implementado em um ambiente de processamento não convencional — a Máquina de Arquitetura Distribuída. Este sistema é constituído por microprocessadores interligados entre si através de um meio físico de comunicação e totalmente confinado em armários ("racks"). O projeto e a elaboração do seu software demostraram a necessidade de se estabelecer metodologias e ferramentas adequadas para o seu desenvolvimento.

1. INTRODUÇÃO

A MAD — Máquina de Arquitetura Distribuída — é um sistema que apresenta uma infraestrutura de multiprogramação e multiprocessamento, com características apropriadas para apoiar

aplicações de tempo real. Seu projeto e a implementação da primeira aplicação se concretizaram no Laboratório de Sistemas Digitais, da Escola Politécnica da Universidade de São Paulo.

A concepção da MAD foi motivada pela necessidade de um sistema de computação que deveria apresentar características de disponibilidade tal que a sua manutenção fosse feita sem a desativação das funções normais nela executadas O resultado foi um sistema altamente modular, com módulos de reserva cujas funções podem ser configuradas pelos próprios programas, quando as falhas são detetadas no funcionamento dos módulos ativos. Sua primeira aplicação foi no sistema de supervisão e controle do subúrbio da FEPASA - Ferrovia Paulista S.A.| [1] |

O hardware da MAD se baseia em módulos que possuem capacidade própria de processamento e de armazenamento, donde a denominação de sistema distribuído, segundo o enfoque dado por Le Lann | [2] | e Ruggiero | [3] |. Os módulos são implementados utilizando-se microprocessadores, INTEL 8085 e são interligados por meio de dois anéis, através dos quais fluem as informações trocadas pelos programas alocados nos processadores | [4] |.

O software da MAD recebeu uma influência direta da sua arquitetura. As funções do programa são alocadas nos módulos constituintes da MAD e a comunicação e a sincronização entre elas são realizadas através da troca de mensagens. A não existência de uma memória comum dos processadores conduziu ao método de programação que foi denominado de programação com processos e mensagens | [5] |.

O suporte para este tipo de programa é dado na MAD pelo Sistema de Comunicação (SC) | [3] | que é um núcleo de sistema operacional e fornece as funções básicas necessária para a existência dos processos e para a comunicação entre eles.

Uma vez estabelecidas as características do programa a ser executada na MAD, surgiu a necessidade de planejar as atividades para seu teste e sua depuração. O paralelismo da execução das tarefas em vários processadores traz vantagens quanto à eficiência do programa, mas introduz, por outro lado, uma série de mecanismos adicionais que podem causar um mal funcionamento do sistema, se não forem utilizados corretamente. Portanto, metodologias e ferramentas se fizeram necessárias, para validar os programas projetados para MAD.

2. MAD E PROGRAMAÇÃO PARALELA

A MAD é uma máquina do tipo sistema distribuído, composta por um conjunto de módulos processadores — os nós — implementados através de microprocessadores INTEL 8085. Estes módulos processadores foram classificados em duas categorias:

- processadores de uso geral (PUG), com recursos de processamento e armazenamento;

- processadores dedicados (PD), com pelo menos um recurso adicional, além de processamento e armazenamento, o qual caracteriza a sua função específica.

As funções dos PDs foram duplicadas na MAD para que a ocorrência de falha em um deles não comprometa a disponibilidade dos recursos adicionais. Os PUGs, por outro lado, foram configurados em número maior que a quantidade necessária para comportar os módulos do programa. Desta forma, quando é detetada uma falha em um PUG, é disparado o mecanismo de reconfiguração que aloca a função do módulo em falha no módulo livre, para que a continuidade do funcionamento do sistema não seja interrompida. Uma descrição detalhada do me

canismo de reconfiguração pode ser encontrada no trabalho de Schaffa [6].

O sistema físico de interligação da MAD é constituído por um sistema de anel duplo, onde as informações são transmitidas em sentidos opostos em cada anel. Os módulos processadores são ligados aos anéis através de processadores de comunicação (PC). Os PCs foram implementados através de um processador microprogramado, com memória de microprogramação cuja capacidade é de 64 palavras de 24 bits. A velocidade máxima permitida pelo sistema físico de interligação da MAD é de 100 Kbytes/seg.

A configuração esquemática da MAD é fornecida através da Figura 1.

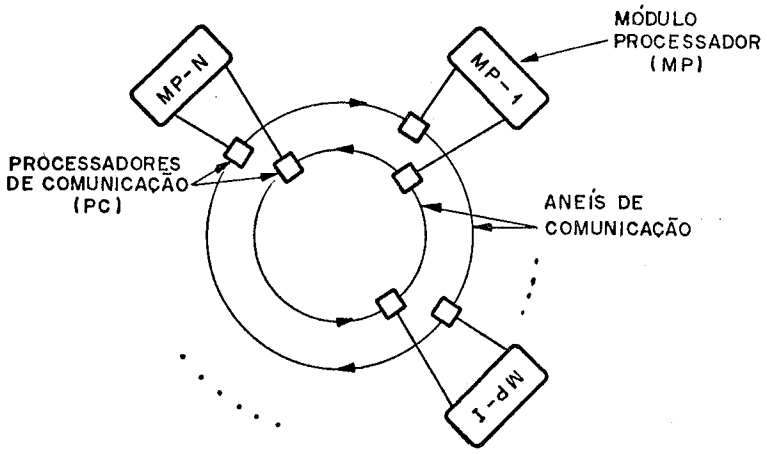


Figura 1

Estrutura da MAD/I

O SC da MAD é um núcleo de sistema operacional e fornece infraestrutura para multiprogramação em cada nó e para a comunicação entre os processos. Este programa foi escrito em linguagem "assembly" (ocupando cerca de 6 Kbytes de memória) e está residente em todos os nós.

A multiprogramação é realizada na MAD através de um elemento do SC denominado de escalador de processos. A sincronização entre os componentes do SC foi implementada através dos semáforos de Dijkstra [7]. A manipulação da fila de recursos obedece a prioridade dos processos; no caso de processos de mesma prioridade, prevalece a ordem de chegada do processo na fila.

A troca de mensagens através de processos é realizada através da área de mensagens do SC, por meio das primitivas de manipulação de mensagens. Estas primitivas permitem o envio e a recepção (seletiva ou não) das mensagens e a pesquisa da fila de mensagens pelos processos. A interface de comunicação entre os processos é uniforme, independentemente se os processos pertencem ou não ao mesmo nó.

Além das primitivas de manipulação de mensagens, o SC fornece, aos processos, algumas funções especiais tais como, temporização para programação de eventos, relógio de tempo real, iniciação e reconfiguração do sistema, etc.

A programação da MAD foi realizada, utilizando-se a PLM que é linguagem de alto nível existente no sistema de desenvolvimento da INTEL. Note-se que esta linguagem é seqüencial, não possuindo nenhuma característica para programação concorrente ou paralela. Estes recursos foram introduzidas na programação da MAD através das chamadas do SC e de uma certa disciplina de programação, para compensar a não existência de ferramentas adequadas.

Desta forma, foram introduzidos os seguintes mecanismos:

- mecanismo de modularidade, para permitir a declaração e ativação dos processos;

- mecanismo de comunicação, para introduzir as manipulações das mensagens;

- mecanismos diversos, para permitir o acesso a outros recursos do SC.

Os recursos de programação paralela, implementados desta maneira, não fornecem nenhum teste de consistência em relação aos mecanismos introduzidos. Fica, portanto, ao cargo dos programadores verificar a existência dos processos destinos das mensagens enviadas, a existência dos processos origens das mensagens esperadas, a coerência entre as mensagens enviadas e esperadas, etc.

As atividades de programação na MAD podem ser melhor amparadas com um sistema do tipo AP^2, onde é possível descrever a estrutura do programa paralelo, dos seus processos e das mensagens, para então analisar a consistência entre os seus elementos [8].

3. METODOLOGIA DE DEPURAÇÃO DE PROGRAMAS NA MAD

O desenvolvimento de programas na MAD alertou a necessidade de elaborar uma sistemática de testes, uma vez que a estrutura do programa e da arquitetura da máquina apresentam características não convencionais.

A filosofia básica destes testes, no entanto, não difere fundamentalmente da metodologia adotada para sistemas de pro

gramação de grande porte. Pode-se considerar, bem a grosso modo, que o envio de uma mensagem e a espera da resposta correspondente seja equivalente à chamada de uma subrotina. As informações enviadas em uma mensagem seriam correspondentes aos parâmetros de entrada; as informações recebidas em uma mensagem seriam correspondentes aos parâmetros de saída. Considerando um processo como referência, as ações dos demais processos poderiam ser considerados como ações das subrotinas chamadas pelo processo em questão. É necessário, no entanto, não desprezar a concorrência das ações (observada, por exemplo, na intervenção de vários processos para a execução de uma tarefa), a seqüencialização e o sincronismo entre as ações e ainda a existência do fator tempo na seqüência dos eventos.

Em linhas gerais, a metodologia de depuração na MAD apresenta as seguintes atividades:

a) análise das funções do programa, em relação à especificação, identificando as transações que realizam tais funções;

b) acompanhamento manual das transações, verificando o fluxo das mensagens, os tipos das mensagens e as suas estruturas internas;

c) teste do programa propriamente dito, através da execução parcial ou total do programa paralelo.

As atividades do item (a) permitem fazer uma verificação da especificação do programa em relação as transações que o constituem. Nesta fase é possível estabelecer um roteiro para os testes, determinando uma prioridade para os testes das transações.

As atividades do item (b) permitem conferir as transações em relação aos processos envolvidos e às mensagens trocadas, evitando-se desta forma que na fase seguinte existam erros de endereçamento na manipulação de mensagens e na interpretação do seu conteúdo.

As atividades do item (c) devem ser desenvolvidas em várias fases, de tal forma que a integração das funções do sistema seja feita através de incorporação sistemática dos processos. De uma forma geral, estas fases são as seguintes:

fase 1: teste individual de processos, onde o objetivo principal é testar as funções internas de cada processo, fornecendo as informações das mensagens esperadas e analisando o conteúdo das mensagens enviadas;

fase 2: teste parcial, quando conjuntos de processos são analisados. Os processos podem ser agrupados por transação ou por pertencerem ao mesmo nó. A escolha deverá ser feita conforme a conveniência do programa a ser testado;

fase 3: teste total, quando todos os processos que constituem um programa paralelo devem estar ativos.

Observe-se que as fases anteriormente descritas devem ser seqüenciais no tempo, para que a integração de novos processos ao programa em teste seja gradativa, até que se atinja a configuração total.

4. SDPP COMO FERRAMENTA DE APOIO À DEPURAÇÃO

As atividades de testes do programa da MAD tiveram o Sistema de Depuração de Programas Paralelos (SDPP) como ferramenta básica de apoio. Este sistema é residente em cada nó da MAD e pode ser configurado de tal forma que permite realizar as três fases de teste descritas no item anterior. Além disso, o SPDD fornece recursos para monitorar a execução do programa paralelo no nível das rotinas de interface entre os módulos de programa e o SC.

O princípio desta ferramenta consiste em isolar um processo ou um conjunto de processos que se deseja testar e ligar as conexões deste módulo, com o restante do programa, ao SDPP que funciona como um meio de atuação e observação para o programador. Desta forma, as mensagens enviadas aos processos não configurados no módulo em teste são apresentadas ao programador. De forma semelhante, quando um processo em teste espera uma mensagem de processos não configurados, esta informação é fornecida ao programador que deve inserir as informações relativa à mensagem esperada.

As Figuras 2 a 5 ilustram alguns exemplos de configuração dos módulos para teste, juntamente com o SDPP.

A ativação do SDPP para o teste de processos é feito em duas fases:

a) fase de iniciação, na qual o SDPP é configurado através das informações fornecidas pelo programador, que são do tipo: dimensão da área de troca de mensagens, identificação dos processos ativos alocados no próprio nó e nos outros nós, primitivas a serem monitoradas, processos a serem simulados pelo programador, etc.

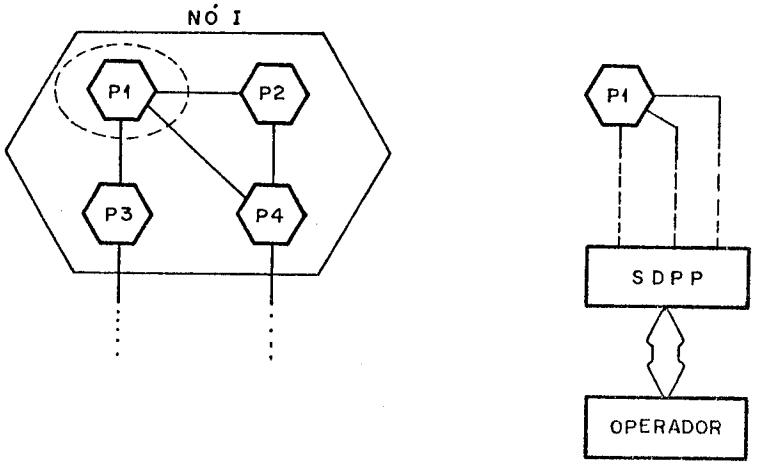


Figura 2 - Teste de um processo

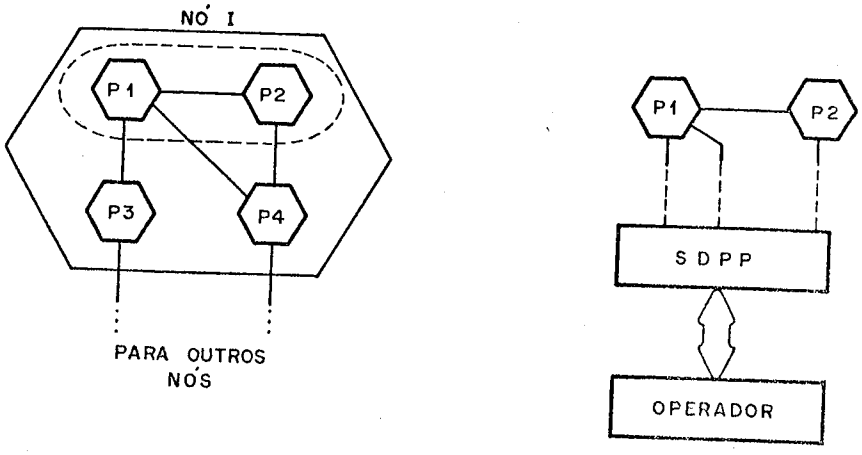


Figura 3 - Exemplo de teste de uma transação contida em um nõ

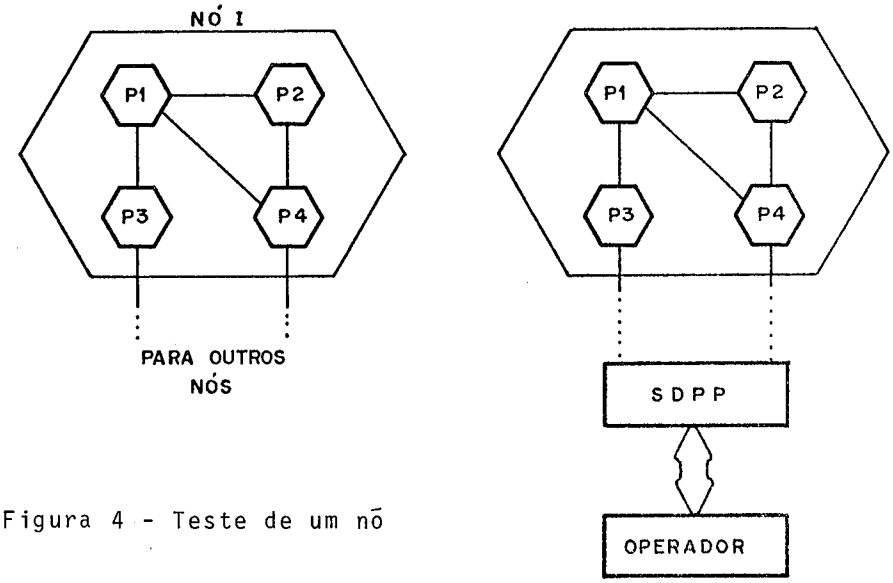


Figura 4 - Teste de um nõ

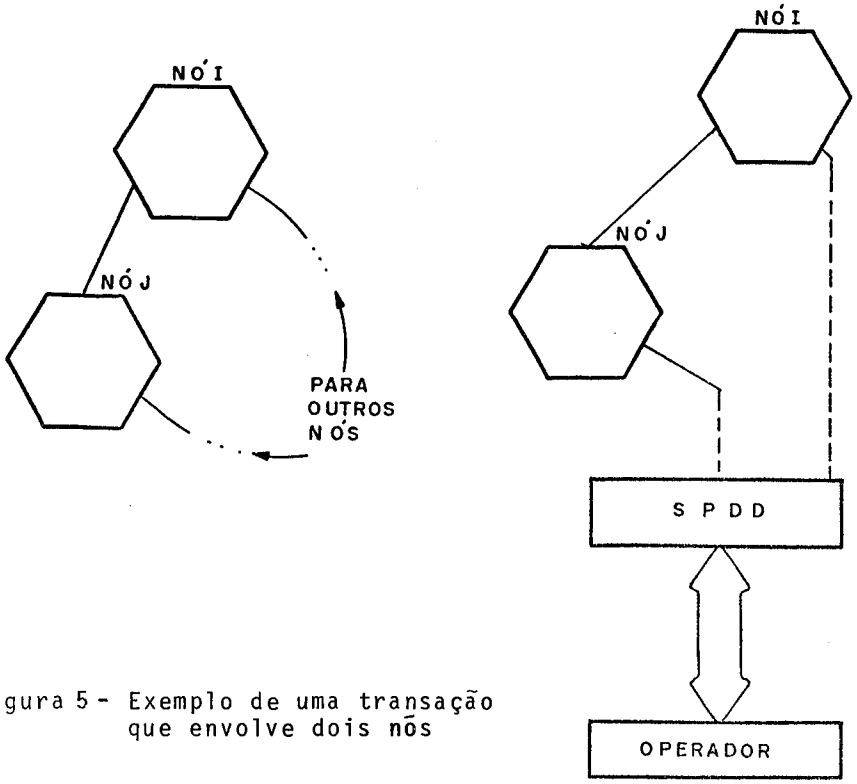


Figura 5 - Exemplo de uma transação que envolve dois nõs

b) fase de execução, na qual o SDPP apresenta a monitoração dos processos e das primitivas por ele requisitados, permitindo o acompanhamento do módulo do programa a ser testado. Durante esta fase, o programador pode interagir com o SDPP através de comandos que permitem a alteração da monitoração definida anteriormente, a análise das variáveis do programa, o preenchimento de uma área da memória com dados, o envio de uma mensagem pelo programador como processo, externo etc.

O SDPP é um instrumento bastante poderoso para a depuração dos programas na MAD, pois permite introduzir os pontos de observação em locais estratégicos. Levando-se em consideração a característica concorrente destes programas, com a execução em mais de um processador, os seus testes teriam sido praticamente impossíveis sem o auxílio do SDPP.

5. CARACTERIZAÇÃO DO SC PARA A DEPURAÇÃO

Os processos e as transações testadas com o auxílio do SDPP estão livres de mensagens com erro no conteúdo, processos origens e destinos indefinidos e má formação na seqüencialização de troca de mensagens. Isto, no entanto, ainda não quer dizer que os testes a eles relativos estejam concluídos. É necessário ainda analisar os seus comportamentos sem a influência do SDPP, nas condições reais de funcionamento.

Nesta fase, a deteção e a solução dos problemas são mais difíceis, uma vez que o meio de observação da execução do programa é totalmente retirado, para não interferir nas funções propriamente ditas. Esta medida se faz necessária para fazer acerto dos parâmetros de tempo, principalmente em aplicações de tempo real.

As falhas no funcionamento do programa se manifestam, geralmente, através da falta de resposta a uma mensagem enviada. O diagnóstico é feito, neste caso, por meio do estado dos nós, fornecido pelas variáveis do SC. As experiências demonstraram que as variáveis mais consultadas são aquelas relativas a:

. estado dos processos, que fornece em que ponto se encontra cada um dos processos;

. última mensagem recebida ou enviada pelos processos;

. conteúdo das filas de mensagens de cada processo;

. número de blocos livres na área de troca de mensagens;

. variáveis relativas à transmissão das mensagens pelos anéis;

. filas associadas aos semáforos utilizados pelo SC, em casos mais raros.

A análise destas variáveis normalmente levam à causa dos problemas detetados, permitindo novos ajustes e correções aos programas em teste.

Como sugestão para uma nova versão do SC vale observar que o acesso a certas variáveis é feito de um modo indireto, o que dificulta a sua consulta. A reestruturação da estrutura de dados visando as necessidades da depuração, poderia simplificar esta tarefa. Uma outra proposta interessante é a de incluir no SC, algumas primitivas de acesso a esta estrutura de dados, que, embora não sejam utilizados na execução efetiva das funções do programa, seriam de grande valia na fase de depuração.

6. CONSIDERAÇÕES FINAIS

O projeto e a implementação do programa de aplicação para a MAD mostrou a necessidade de uma metodologia de depuração, devido à complexidade e ao volume dos programas e também devido à dificuldade intrínseca apresentada por suas atividades paralelas.

Este programa realiza a supervisão e controle de ferrovia e possui na sua totalidade cerca de 600 Kbytes de programa, distribuídos em cerca de 20 módulos processadores. Atualmente o sistema se encontra em fase final de integração das funções da supervisão, com os testes sendo executados sem o auxílio do SDPP. A sua depuração passou pelas atividades descritas neste trabalho, nem sempre de forma ordenada, devido à inexperiência em relação às características de sistemas distribuídos e programas paralelos. Apesar disto pode-se dizer que os resultados que estão sendo obtidos são de grande valia na área de programação concorrente e paralela, podendo servir de base, inclusive para sistemas que envolvam distâncias maiores entre os processadores.

7. REFERÊNCIAS BIBLIOGRÁFICAS

| 1 | Martucci, M. Jr.; Moscato, L.A.
"Sistema Centralizado de Supervisão e Controle"
Anais: 1º Simpósio em Controle de Processos por Computador, 1981.

| 2 | Le Lann, G.
"Distributed Systems — Towards a Formal Approach"
IFIP, North Holland Publishing Company, 1977. p. 155-160.

[3] Ruggiero, W.V.; Melnikoff, S.S.S.; Lucena, C.J.
"Implementação de um Sistema de Comunicação entre os Processos através de uma Máquina de Arquitetura Distribuída".
Anais do 7º Seminário Integrado de Software e Hardware, 1980. p. 1-9.

[4] Shimizu, E. Y.; Ruggiero, W.V., Moscato. L.A.
"Componentes Básicos de uma Máquina de Arquitetura Distribuída: Identificação e Implementação".
Anais do 7º Seminário Integrado de Software e Hardware, 1980. p. 77-86.

[5] Cunha, P.; Lucena, C.J.; Maibaum, T.
"On the Design and Specification of Mesage Oriented Programs".
International Journal of Computer and Information Sciences, Vol. 9, 1980.

[6] Schaffa, F. A.; Moscato, L. A.
"The Distributed Architeture Machine (DAM): Recovery through Reconfiguration".
15th International Symposium on Mini and Micro Computers, 1981. p. 259-263.

[7] Dijkstra, E. W.
"Cooperating Sequential Processes", em "Programming Languages".
F. Genuys (ed.), Academic Press, 1968.

[8] Melnikoff, S.S.S.
"Sistema AP^2· um Ambiente para a Programação Paralela".
Tese de Doutoramento, Escola Politécnica da Universidade de São Paulo, 1982.