Boletim Técnico da Escola Politécnica da USP

Deptº de Engª de Computação e Sistemas Digitais

BT/PCS/9306

Uma Ferramenta de Monitoração para um Núcleo de Resolução Distribuída de Problemas Orientado a Objetos

Jaime Simão Sichman
Eleri Cardoso

São Paulo - 1993

O presente trabalho é baseado na dissertação de mestrado de mesmo título,apresentada em 1991 pelo Engº Jaime Salomão Sichman, sob orientação do Prof. Dr. Eleri Cardoso.

A íntegra da dissertação encontra-se à disposição com o autor e na biblioteca de Engenharia de Eletricidade da Escola Politécnica da USP.

Sichman, Jaime Simão
Uma ferramenta de monitoração para um núcleo de resolução distribuída de problemas orientado a objetos / J.S. Sichman, E. Cardoso. -- São Paulo : EPUSP, 1993.
22p. -- (Boletim Técnico da Escola Politécnica da USP, Departamento de Engenharia de Computação e Sistemas Digitais, BT/PCS/9306)

1. Programação orientada a objetos (Computação) 2. Processamento distribuído (Computação) 3. Ferramentas de software I. Cardoso, Eleri II. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais III. Título IV. Série

CDU 005.1
004.36
005.43

UMA FERRAMENTA DE MONITORAÇÃO PARA UM NÚCLEO DE RESOLUÇÃO DISTRIBUÍDA DE PROBLEMAS ORIENTADO A OBJETOS

Jaime Simão Sichman
EPUSP - PCS
Av. Prof. Luciano Gualberto, 158 trav. 3
05508 - São Paulo - SP

Eleri Cardozo
CTA - ITA - IEE
12225 - São José dos Campos - São Paulo

RESUMO

Este artigo apresenta uma ferramenta de monitoração para um núcleo de resolução distribuída de problemas denominado DPSK+P [CARD 91]. Tal núcleo permite que diversos agentes, desenvolvidos em diferentes linguagens que suportam o paradigma de objetos, compartilhem dados e métodos entre si, de modo a cooperarem no processo de resolução de determinado problema. A ferramenta de monitoração é baseada em eventos de comunicação e controle entre agentes, utilizando a técnica de animação de processos para efeito de exibição dos resultados. Além disso, propõe um método eficiente de exibição do fluxo de comunicação em sistemas baseados em memória compartilhada, aproveitando a compacidade presente nas hierarquias de classes que compõem um sistema orientado a objetos.

ABSTRACT

This work consists of the design and implementation of a monitoring facility for an object-oriented distributed problem solving kernel named DPSK+P. This kernel allows both data and method sharing between agents, that may have been developed in different object-oriented programming languages, so that they can cooperate in a problem solving activity. The monitoring facility is based on control and communication events, and its exhibition of results uses process animation techniques. It also presents an efficient framework for the exhibition of the communication flow in shared memory based systems, which profites the existing compactness in object-oriented class hierarchies.

PALAVRAS-CHAVE: Resolução Distribuída de Problemas, Orientação a Objetos, Depuração e Monitoração de Sistemas Distribuídos.

1 MOTIVAÇÃO

As futuras aplicações computacionais, a serem desenvolvidas na década de 90, serão cada vez maiores e complexas. Tais aplicações deverão ter como requisitos fundamentais de projeto o suporte à modificação, extensão e reconfiguração de sistemas. O modelo de execução será paralelizado, quer no que diz respeito ao ambiente computacional utilizado (multiprocessado ou distribuído) quer no que se refere à natureza da própria aplicação (multiprogramação).

Já do ponto de vista da Engenharia, o projeto de sistemas heterogêneos é hoje cada vez mais comum. Tal característica se refere tanto à plataforma utilizada (diferentes máquinas conectadas em rede) como aos diversos módulos que compõem o software do sistema, os quais podem ter sido desenvolvidos em diferentes linguagens de programação. Atualmente, a integração de módulos desenvolvidos em diferentes linguagens de programação é bastante complicada e ineficiente. Além disso, em tais casos torna-se imprescindível a existência de um ambiente adequado para o desenvolvimento dessas aplicações, especialmente ferramentas de depuração e monitoração.

Para se tratar convenientemente tais situações, podem ser utilizadas técnicas de resolução distribuída de problemas e o paradigma de objetos.

2 RESOLUÇÃO DISTRIBUÍDA DE PROBLEMAS

Resolução distribuída de problemas é uma metodologia proveniente da inteligência artificial utilizada para o desenvolvimento de grandes organizações de software. Um dado problema original, bastante complexo, é sucessivamente dividido em subproblemas, menores e mais simples, de modo a facilitar o desenvolvimento de módulos computacionais que os solucionem. A solução do problema

original é então obtida a partir da integração dos resultados desses subproblemas.

Os módulos computacionais são denominados agentes e têm as seguintes características [CARD 87]:

- alta complexidade, significando que processam tarefas bastante complexas;
- alta granularidade, significando que o tempo empregado em processamento local é muito superior àquele empregado na interação com outros agentes;
- alta heterogeneidade, significando que as tarefas processadas pelos diferentes módulos são bastante diversas;
- alta generalidade quanto ao hardware, significando que as máquinas onde são processados os módulos também podem ser bastante distintas.

Tal metodologia também é conhecida por processamento cooperativo, uma vez que os agentes cooperam no processo de resolução do problema originalmente proposto. Tal cooperação se dá através de ações de comunicação e controle. Ações de comunicação se referem à troca de informações sobre o domínio de solução do problema, enquanto que ações de controle se referem a atividades de coordenação na execução dos diversos agentes, envolvendo mecanismos de ativação, finalização, sincronização e interrupção de agentes. Uma vez utilizadas de modo adequado, tais ações de controle garantem que um agente será executado no tempo correto e de posse dos recursos necessários ao processamento de sua tarefa.

Um modelo muito utilizado em resolução distribuída de problemas é o modelo "Blackboard" [ENGE 88]. Trata-se de uma coleção de agentes, denominados fontes de conhecimento, agrupados em torno de uma memória

compartilhada hierárquica. Os agentes trocam informações entre si lendo e escrevendo nesta memória, sendo que cada agente tem acesso apenas a alguns níveis hierárquicos dessa memória. Existe também um agente especial, denominado monitor, que analisa o conteúdo da memória e elege a próxima fonte de conhecimento a ser ativada. O sistema HEARSAY-II [ERMA 80], destinado a compreender a fala humana em um certo domínio específico, foi o primeiro sistema construído segundo este modelo.

Tentativas também vêm sendo realizadas no sentido de aproveitar resultados provenientes da teoria das organizações no desenvolvimento de grandes organizações de software. Tais tentativas contemplam, inclusive, a construção de modelos computacionais, baseados em "Blackboards", que simulem organizações tais como hierarquias simples, múltiplas e mercados [FOX 81].

3 PARADIGMA DE OBJETOS

O chamado paradigma de objetos surgiu como uma tentativa de solucionar certos problemas detectados pela Engenharia de Software, tais como a dificuldade em estender e evoluir sistemas. Foi influenciado, basicamente, pelas áreas de linguagens de programação, bases de dados e inteligência artificial [TAKA 90]. Da área de linguagens, importou-se o conceito de tipo abstrato de dados, ou seja, um forte encapsulamento de dados e procedimentos que os acessam em módulos específicos. Da área de bases de dados, importou-se o conceito de modelagem conceitual, onde as entidades do mundo real que devem ser representadas no sistema a ser construído são classificadas em hierarquias do tipo classificação/instanciação, generalização/ especialização e agregação/decomposição. Finalmente, da área de inteligência artificial importou-se o conceito de herança, utilizado em um método de representação de conhecimento denominado "frame".

Os principais conceitos envolvidos no paradigma de objetos são objetos, classes, mensagens e métodos [STRO 88] [STEF 86].

Objetos são unidades conceituais, resultantes de um processo de abstração do mundo real, compostos por um estado interno, onde valores podem ser armazenados e modificados ao longo da vida do objeto, e por um comportamento, onde são definidas ações (denominadas métodos) através das quais o objeto responderá à demanda de processamento por parte de outros objetos.

Objetos se comunicam entre si através do envio e recebimento de mensagens. Mensagens são compostas por um seletor e por parâmetros. Um objeto, ao receber uma mensagem, realiza um acoplamento entre o seletor da mensagem e algum de seus métodos. Uma vez selecionado, o método é então executado e o resultado da computação é enviado ao objeto que gerou a mensagem.

Utilizando um mecanismo de abstração do tipo classificação, objetos cuja estrutura e comportamento sejam idênticos são agrupados em classes. Tal mecanismo permite que a descrição das propriedades e do comportamento de uma classe de objetos seja realizada apenas uma única vez, independentemente do número de instâncias dessa classe que possam existir na aplicação.

Classes são inseridas em uma hierarquia de especialização, sendo que uma classe mais específica herda todas as propriedades da classe localizada imediatamente acima na hierarquia, denominada sua superclasse direta. Desta forma, caso um objeto receba uma mensagem cujo seletor não corresponda a nenhum de seus métodos, um método definido nas classes superiores pode ser ativado.

Os métodos definidos em uma dada classe podem ter ainda diferentes níveis de visibilidade. Certos métodos, denominados privados, que acessam atributos internos ao

objeto somente podem ser ativados por instâncias da própria classe. Outros, denominados protegidos, estendem tal direito às instâncias de suas subclasses. Os métodos públicos podem ser ativados por quaisquer instâncias.

O grande atrativo do paradigma de objetos resulta na sua estensibilidade. Uma hierarquia de classes já desenvolvida para uma determinada aplicação pode ser reutilizada e especializada (criação de novas classes e/ou modificação de métodos em classes já existentes), de modo a aumentar a eficiência do processo de produção de software.

4 O SISTEMA DPSK+P

O paradigma de objetos apresenta naturalmente um enfoque distribuído. A independência e forte encapsulamento de dados e métodos em um dado objeto somente vem reforçar tal enfoque. Deste modo, tornam-se desejáveis, e até mesmo naturais, esforços no sentido de estender o alcance de tal paradigma a ambientes distribuídos. Deste modo, diversos agentes desenvolvidos segundo o paradigma poderiam compartilhar objetos, mesmo que estes tivessem sido definidos em outros agentes, sendo processados na mesma máquina ou em máquinas distintas.

O objetivo básico do sistema DPSK+P [CARD 91] é o de incorporar algumas práticas provenientes do paradigma de objetos num núcleo para resolução distribuída de problemas. Tais práticas se referem à possibilidade de definir hierarquias de classes, métodos e mecanismos de instanciação para objetos compartilhados.

A idéia básica é a de permitir que um dado agente possa tornar certos dados e métodos compartilháveis quando desejar que sejam acessíveis por outros agentes. Para atingir tal objetivo, os dados contidos nos diferentes objetos são divididos em três níveis: um nível privado, cujo acesso é permitido apenas ao próprio

objeto, um nível público, cujo acesso é permitido aos outros objetos definidos no mesmo agente, e um nível compartilhado, cujo acesso é permitido a quaisquer outros agentes, não necessariamente localizados na mesma máquina. Os dois primeiros níveis são usuais em linguagens que suportam o paradigma. A contribuição do sistema DPSK+P reside no terceiro nível, que contempla justamente o aspecto distribuído. Neste último nível ainda, torna-se também necessário prover um mecanismo pelo qual um agente possa ativar métodos definidos em objetos mantidos por outros agentes, para manipular os dados compartilhados.

Para tornar mais claros os objetivos do sistema DPSK+P, seja tomada como exemplo a seguinte hierarquia de classes, mostrada na figura 1.

Os métodos e atributos do nível privado da classe Aluno só podem ser manipulados por métodos definidos na própria classe. O nível protegido estende este direito às suas subclasses, no caso Aluno de Graduação e Aluno de Pós-Graduação. Finalmente, o nível público define o protocolo de manipulação das instâncias da classe Aluno para todos os outros objetos definidos no programa. Tais direitos, entretanto, se restringem a um único agente ou programa. O que o sistema DPSK+P se propõe é estender este direito a outros agentes, ou seja, possibilitar, por exemplo, que atributos da classe Aluno definida no agente 1 possam ser acessados por objetos pertencentes à classe Aluno definida no agente 2 e vice-versa. Tais dados compartilhados é que consistirão o chamado nível compartilhado. De um certo modo, trata-se de estender o nível público para outros agentes, localizados ou não na mesma máquina e codificados ou não na mesma linguagem de programação.

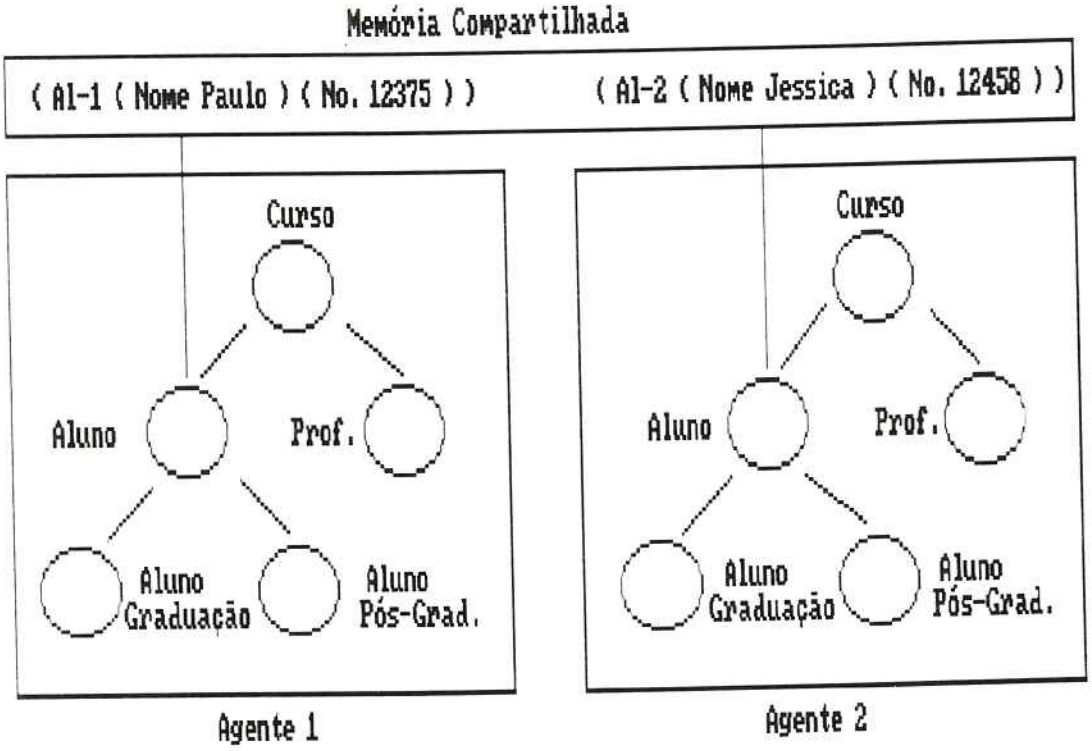

Figura 1 - Classes Compartilhadas no Sistema DPSK+P

Para efeito de controle, são previamente definidas certas classes responsáveis pela criação, interrupção, bloqueio e sincronização entre processos, bem como pela manutenção de consistência dos dados compartilhados.

5 DEPURAÇÃO DE SISTEMAS PARALELOS

No caso de sistemas paralelos, os processos clássicos de monitoração e depuração encontram dificuldades adicionais que não aparecem em sistemas seqüenciais. As principais dificuldades são a existência de vários focos de controle, a grande quantidade de dados a serem monitorados e analisados, a inexistência de um estado global, a presença de características não determinísticas no processamento e o caráter intrusivo da ferramenta de monitoração ou depuração [DOWE 89] [JOYC 87].

O fato de existirem vários focos de controle impede que técnicas clássicas de depuração, como "traces" e

"breakpoints", sejam utilizadas de forma eficiente em sistemas paralelos. No caso de sistemas distribuídos, é desejável ainda que a ferramenta de monitoração esteja localizada em apenas uma máquina, de modo a facilitar o processo de análise pelo usuário, devendo então a ferramenta concentrar informações de controle de máquinas distintas.

O processo de análise poderá também se tornar muito complicado, caso não se consiga exibir de modo claro as ligações entre as informações exibidas e os processos a partir dos quais tais informações foram colhidas. Dada a natureza extremamente complexa das interações que ocorrem em um sistema paralelo, a quantidade de dados a serem exibidos é muito grande e heterogênea (especialmente em resolução distribuída de problemas, onde os agentes podem ter sido desenvolvidos em várias linguagens e tratando níveis conceituais diferentes do problema), devendo ser providos mecanismos para filtrar tal quantidade de dados, de modo a apresentar ao usuário somente aqueles que efetivamente o interessam no momento.

Sistemas multiprocessados têm geralmente associado um relógio a cada uma das CPUs que os compõem. Em sistemas distribuídos, também não existe um relógio global que sincronize o processamento nas diversas máquinas. Nestes casos, o conceito de estado global de processamento inexiste, uma vez que não se pode determinar com razoável precisão a ordem dos processos que estão sendo executados concorrentemente, em processadores distintos. Mais ainda, supondo um sistema distribuído onde a ferramenta de monitoração se localize em apenas uma máquina, o processo de coleta de dados apresenta atrasos inerentes à utilização de uma rede, fazendo com que dados coletados em diferentes máquinas possam refletir o estado de cada uma delas em instantes diferentes, e não no mesmo instante como se desejava a princípio.

Em sistemas paralelos, nem sempre um mesmo conjunto de entradas produz um mesmo conjunto de saídas, tornando o seu comportamento não determinístico. Tal comportamento se deve ao aparecimento de corridas entre processos, sempre que duas atividades possam ser processadas em paralelo, introduzindo uma característica de não repetibilidade de processamento. Devido a isso, o processo de depuração pode se tornar muito complicado, pois nem sempre é possível repetir a ocorrência de uma falha, se a causa desta se deve a condições críticas de carga das máquinas ou processadores envolvidos.

O caráter não determinístico descrito acima é ressaltado pelo chamado "efeito de ponta de prova". Quaisquer tentativas no sentido de colher mais informações para análise podem aumentar a dificuldade em repetir um comportamento errôneo que tenha sido observado. Isto se deve ao caráter intrusivo das ferramentas de depuração ou monitoração, no sentido de que novos processos serão executados no sistema, alterando a carga dos processadores ou máquinas e introduzindo, eventualmente, novas corridas. Estas últimas podem alterar o comportamento do sistema em relação à situação original, na qual as ferramentas de depuração ou monitoração não se encontravam ativas.

Uma das técnicas que vêm sendo utilizadas para superar em parte tais dificuldades é a criação de eventos. Eventos podem ser definidos como ações atômicas visíveis acima do escopo de um determinado processo, que representam mecanismos de controle ou comunicação entre processos, e que são dependentes do sistema computacional utilizado. Em sistemas baseados em troca de mensagens, um envio ou recebimento de mensagens constitui um evento, enquanto que um acesso a uma posição de memória compartilhada constitui um evento em sistemas baseados neste modelo de comunicação entre processos. Sistemas mais flexíveis permitem que o próprio usuário limite a quantidade de eventos a ser examinada, ignorando aqueles

que não o interessam no momento e crie hierarquias de eventos, através da combinação de eventos primitivos [BATE 89].

Do ponto de vista de apresentação dos dados monitorados, basicamente quatro diferentes técnicas são utilizadas: apresentação textual, diagramas temporais, animação de processos e utilização de múltiplas janelas [DOWE 89].

Apresentação textual é a técnica mais simples: trata-se de exibir mensagens que correspondam ao conteúdo do evento sendo analisado. Em alguns casos, quando não existe um monitor com capacidades gráficas, trata-se da única técnica possível de ser utilizada.

Diagramas temporais são uma representação bidimensional do estado de um sistema paralelo no tempo. Em um eixo é representado explicitamente a passagem do tempo e no outro a ocorrência dos eventos. Tais diagramas são extremamente úteis quando se deseja priorizar o aspecto evolutivo dos eventos no tempo, porém fornecem um retrato muito pobre do estado global dos processos sendo monitorados.

Em animação de processos, cada processo tem associado a si uma porção da tela, sendo representado através de figuras geométricas ou ícones. A ligação entre os processos pode ser feita por meio de segmentos de reta, que se utilizam de mudança de cores ou de largura para indicar a ocorrência de algum evento referente ao nível interprocessos. Assim, em cada instante, tem-se um retrato apurado do estado global do sistema, sem haver, no entanto, uma idéia de evolução dos eventos no tempo.

Sistemas de múltiplas janelas são usuais nos dias de hoje, especialmente em estações de trabalho. Um sistema que certamente seria muito eficiente poderia constar de duas janelas, cada uma exibindo o fluxo de controle segundo uma das duas últimas técnicas descritas acima, de

modo a fornecer ao usuário informações precisas sobre o estado global instantâneo e sobre a evolução de comportamento do processos.

6 UMA FERRAMENTA DE MONITORAÇÃO PARA O SISTEMA DPSK+P

Para efeito de monitoração do sistema DPSK+P, foram definidos vários eventos elementares correspondentes a ações de comunicação e controle entre agentes [SICH 91]. Além destes eventos, informações sobre classes e instâncias contidas na memória compartilhada encontram-se também disponíveis para fins de consulta pelo usuário.

Os eventos de controle definidos e tratados pela ferramenta de monitoração referem-se à ativação, suspensão e retomada de processamento, ativação de uma exceção e término normal ou forçado de processamento de um agente. Além disso, também existem eventos que sinalizam a ativação, processamento e término de uma chamada remota de procedimentos por um agente.

Toda a comunicação entre agentes no sistema DPSK+P se dá através de operações de leitura e escrita em objetos compartilhados. A manutenção da consistência desses objetos pode ser realizada pelo núcleo, através de transações [CARD 91], se o usuário assim o desejar.

Os eventos de comunicação definidos e tratados pela ferramenta de monitoração refem-se ao início, interrupção ou fim de uma transação e a operações de leitura e escrita na memória compartilhada.

O conteúdo da memória compartilhada, no que se refere às classes e instâncias criadas pelos diversos agentes, também é convenientemente tratado pela ferramenta de monitoração. Foi criado um navegador, que permite ao usuário visualizar a hierarquia de classes, seu conteúdo (definição dos "slots", agente criador) e suas instâncias (máquina onde se encontram, identificador).

Elegeu-se a técnica de animação de processos para a exibição dos eventos elementares de comunicação e controle mostrados acima. Tal escolha deveu-se à priorização de informações referentes ao estado global de processamento, em detrimento de uma evolução temporal mais apurada.

A fim de minimizar os problemas decorrentes desta ausência de informações temporais mais precisas, o método de animação de processos adotado foi ligeiramente modificado em relação àquele originalmente descrito. Optou-se, no projeto da ferramenta, pela manutenção permanente da última ação de controle ou comunicação realizada.

6.1 Exibição dos Eventos de Controle

Para efeito de monitoração das ações de controle, cada agente encontra-se representado na ferramenta de monitoração por um círculo, localizado em uma determinada posição na tela. Os diferentes círculos serão preenchidos com cores diferentes, de modo a representar a ocorrência dos diferentes eventos de controle. Além disso, setas ligarão os círculos entre si, com o objetivo de representar logicamente a ligação entre o agente ativo (aquele que ativou a ação de controle) e o agente passivo (aquele ao qual a ação de controle foi endereçada). No caso de chamada remota de procedimentos ou da ativação de exceções, o nome do método sendo ativado ou o grupo e sinal referentes à exceção também serão respectivamente exibidos. Um exemplo de exibição destes eventos de controle encontra-se representado na figura 2.

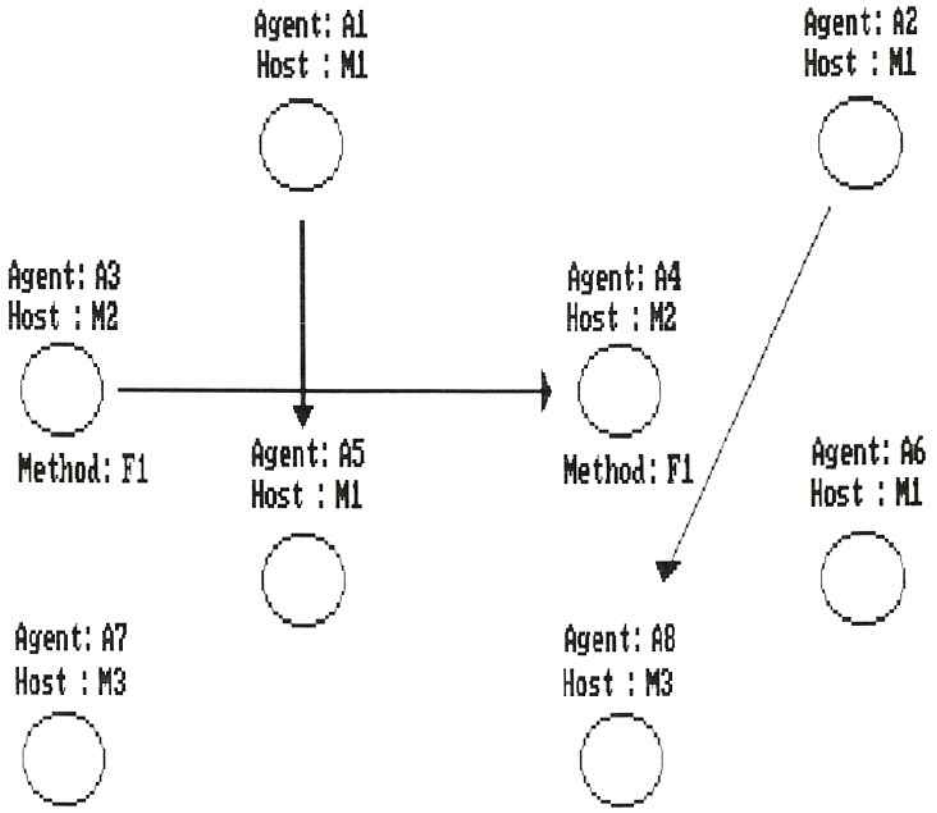

Figura 2 - Exibição dos Eventos de Controle entre Agentes

Esta figura representa um conjunto de oito agentes, fisicamente localizados em três máquinas distintas. Os agentes A1, A2, A5 e A6 localizam-se na máquina M1, os agentes A3 e A4 na máquina M2 e os agentes A7 e A8 na máquina M3. Os agentes A1 e A2 ativaram ações de controle nos agentes A5 e A8, respectivamente, sendo que a especificação destas ações se dá em função da cor que tomam os agentes que as sofreram. Já os agentes A3 e A4 estão envolvidos num mecanismo de chamada remota de procedimentos, através do método F1.

As setas ligando os agentes somente serão apagadas quando o agente que sofreu a ação de controle sofrer uma outra ação de controle posterior. Tal persistência permite que, a cada instante, estejam representados na tela os últimos eventos nos quais os agentes estiveram envolvidos, minimizando em parte a inexistência de um diagrama temporal de eventos, que contivesse informações

mais precisas sobre a evolução de comportamento dos mesmos.

A princípio, sempre que um novo agente for ativado, ele será exibido pela ferramenta, desde que o limite de agentes simultaneamente visíveis não seja excedido. O usuário pode, entretanto, controlar a visibilidade destes agentes, especificando aqueles que deseja observar num determinado momento. Este efeito, que corresponde a uma filtragem de eventos de controle, é extremamente útil em resolução distribuída de problemas, quando se deseja depurar o processo de resolução de um determinado subproblema, no qual estão envolvidos apenas um determinado número de agentes.

6.2 Exibição dos Eventos de Comunicação

Para efeito de monitoração do mecanismo de comunicação entre agentes, estes serão representados por linhas horizontais, que acessarão classes compartilhadas, representadas por retângulos. As operações de escrita e leitura de instâncias dessas classes serão representadas por setas ligando agentes a classes, respectivamente saindo e chegando nos agentes, como mostra a figura 3. Os valores associados às setas referem-se a identificadores das instâncias acessadas.

Segundo esta figura, o agente A1 realizou uma operação de escrita na instância I1 da classe C1. Já os agentes A3 e A1 estabeleceram uma comunicação efetiva com os agentes A2 e A4, respectivamente, através das instâncias I2 e I3 das classes C2 e C4.

A ocorrência de transações será visualizada através da atribuição de cores aos agentes e classes envolvidos. Assim, referindo-se ainda à figura 3, caso o acesso do agente A1 à classe C1 tenha sido feito através de uma transação, a linha que o representa bem como o conteúdo do retângulo que representa C1 tomaram a mesma cor quando

a transação se iniciou. Ao final da mesma, ambos voltaram a exibir as cores padrões.

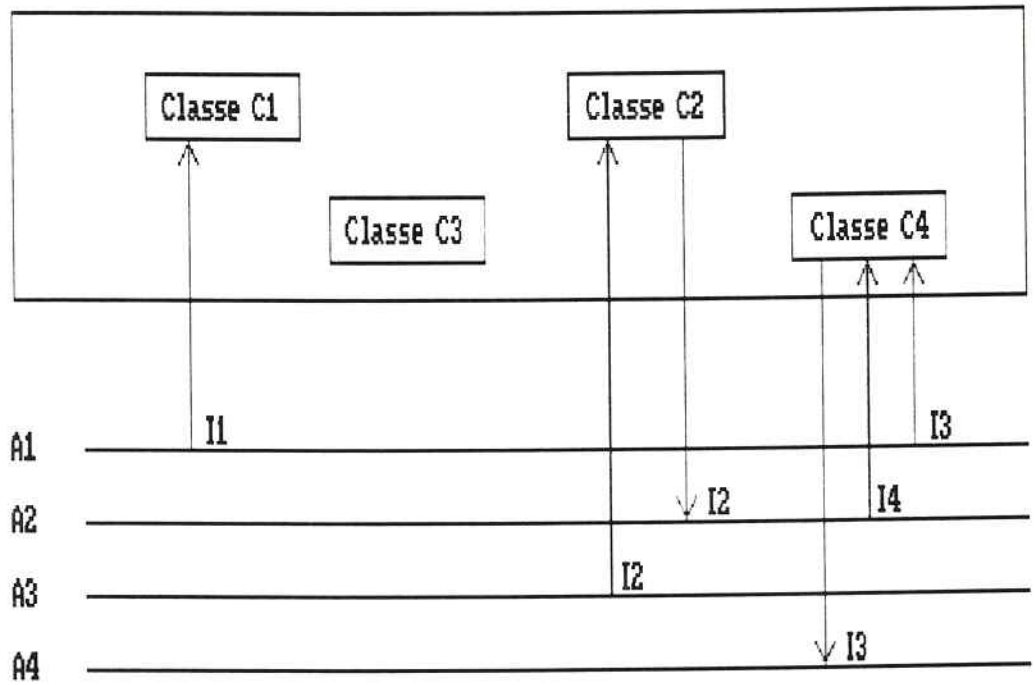

Figura 3 - Exibição dos Eventos de Comunicação entre Agentes

A última operação realizada na memória compartilhada será sinalizada através de uma cor diferente nas setas ligando agentes a classes. Novamente, os últimos acessos realizados em cada classe terão persistência até o acesso seguinte, minimizando em parte a ausência de um diagrama temporal de eventos, como no caso da exibição dos eventos de controle.

Tal procedimento, entretanto, apresenta um outro efeito, muito mais importante, e no qual consiste a contribuição básica deste trabalho. Trata-se de um mecanismo poderoso que permite representar ações de comunicação efetiva entre agentes, num esquema de memória compartilhada. Tentativas de representar tais ações em esquemas de troca de mensagens são usuais na literatura,

o que não ocorre no caso de sistemas baseados em memória compartilhada. Tal fato decorre de que cada elemento em tal memória é, potencialmente, um mecanismo de comunicação, inviabilizando uma representação concisa e eficiente, notadamente para sistemas com aplicação prática. No caso do sistema DPSK+P, isso somente é possível pela utilização do paradigma de objetos na memória compartilhada. Cabe notar que a especificação das instâncias acessadas pelos diversos agentes é extremamente compacta, sendo representadas somente aquelas diretamente envolvidas com a ocorrência dos últimos eventos de comunicação para cada classe. Embora tal esquema impeça a inspeção direta de valores contidos nestas instâncias, ele deve ser considerado como um mecanismo de abstração, ou seja, suficiente para efeitos de monitoração de alto nível, quando eventos de alto nível são criados a partir de eventos primitivos. Somente através deste mecanismo de abstração torna-se viável exibir de modo compacto as ações de comunicação, através da limitação da quantidade de "canais de comunicação" lógicos, representando o conteúdo da memória compartilhada. Além disso, os meios de exibição da estrutura de classes compartilhadas descritos a seguir permitem que o usuário, ao detectar uma falha de processamento, inspecione estas instâncias com um maior grau de profundidade.

6.3 Exibição das Classes Compartilhadas

A ferramenta de monitoração permite ao usuário visualizar as classes e instâncias contidas na memória compartilhada em três níveis de abstração distintos.

Um primeiro nível apresenta a hierarquia de classes. Caso o conteúdo de uma determinada classe necessite de um maior nível detalhe, este é fornecido por um segundo nível, conforme mostra a figura 4.

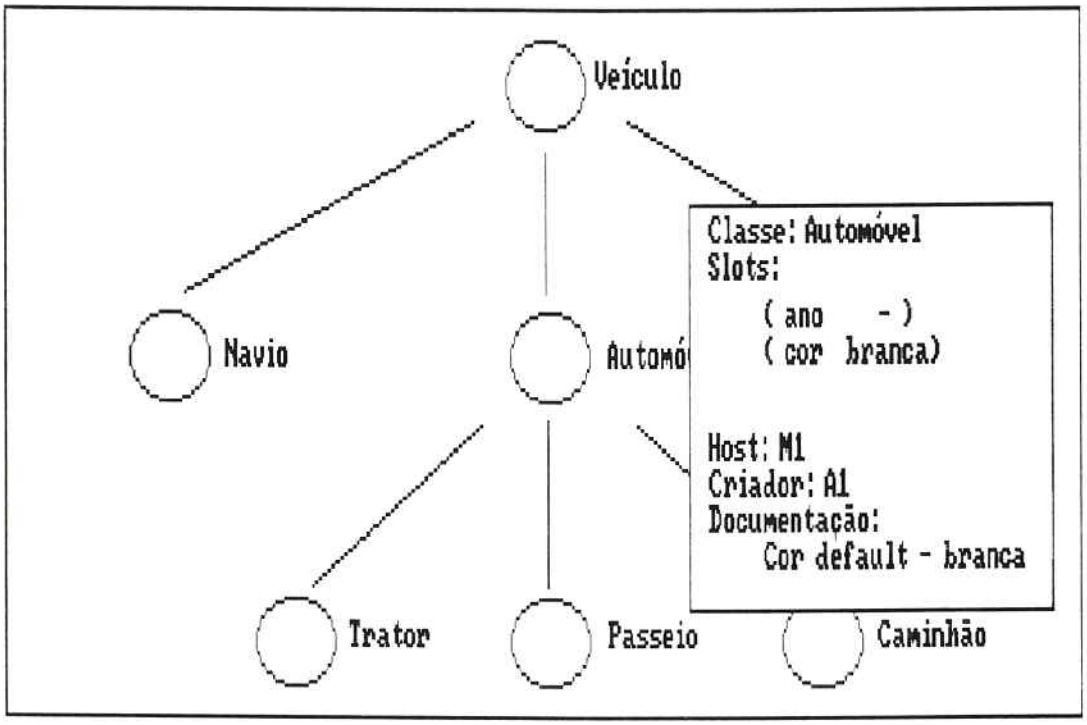

Figura 4 - Exibição do Conteúdo de uma Classe Compartilhada

Finalmente, um terceiro nível provê ao usuário informações sobre as instâncias criadas a partir das classes.

Através do manuseio de um "mouse", a raiz da estrutura sendo exibida pode ser deslocada, permitindo a navegação na hierarquia, tanto vertical como horizontalmente. Além disso, a utilização do "mouse" permite também a mudança entre os diferentes níveis de abstração descritos acima.

6.4 Estrutura da Ferramenta

A ferramenta de monitoração se compõe de três módulos distintos: um navegador de eventos de controle, um navegador de eventos de comunicação e um navegador da hierarquia de classes compartilhadas. Seu desenvolvimento foi baseado na utilização de um sistema com múltiplas janelas, de modo a associar a cada navegador uma janela

distinta. Com isso, torna-se possível a exibição dos dados referentes aos diferentes módulos ao mesmo tempo, caso assim deseje o usuário. Além disso, um desenvolvimento modular permite a incorporação de facilidades futuras, como por exemplo um módulo responsável pela criação e exibição de um diagrama temporal de eventos.

Do ponto de vista de comportamento dinâmico, a ferramenta de monitoração compreende três processos distintos, executados simultaneamente, que correspondem a cada um dos navegadores citados. Tal enfoque permite a atualização das três janelas ao mesmo tempo, mesmo que alguma delas se encontre total ou parcialmente obscurecida por outra.

A ferramenta de monitoração se comporta como um grupo de agentes particular frente ao núcleo, que ativa algumas primitivas especialmente criadas para efeito de monitoração, cujo único objetivo é o de fornecer informações à ferramenta de monitoração. Não existem, portanto, acessos adicionais à memória compartilhada, uma vez que o núcleo tem disponíveis para si todas as informações referentes a ações de comunicação e controle entre agentes. Do ponto de vista da ferramenta de monitoração, basta ativar estas primitivas para obter uma cópia das informações desejadas, sem o ônus de um acesso à memória compartilhada.

6.5 Implementação

A ferramenta de monitoração está sendo desenvolvida numa estação de trabalho do tipo SUN SPARC-II, com monitor colorido, rodando em ambiente UNIX [CHRI 85]. A linguagem C [KERN 78] está sendo utilizada para seu desenvolvimento, para não penalizar sua portabilidade. Além disso, o sistema X WINDOWS v. 11 [JONE 89] está sendo utilizado tanto como gerenciador de janelas quanto para gerar figuras gráficas.

Até o presente momento, os navegadores de eventos de controle e comunicação já foram desenvolvidos e encontram-se integrados ao núcleo. Tais módulos contém aproximadamente 2500 linhas de código.

7 CONCLUSÕES

Técnicas de resolução distribuída de problemas e o paradigma de objetos serão utilizados nas próximas décadas durante o desenvolvimento de grandes organizações de software, que contemplarão suporte à evolução e reutilização de sistemas. Durante o processo de projeto de tais sistemas, ferramentas de auxílio poderosas, como depuradores e monitores, deverão ser intensamente utilizadas.

A ferramenta de monitoração construída para o sistema DPSK+P [SICH 91] consiste num conjunto de três navegadores, que exibem eventos de comunicação e controle entre agentes, bem como o conteúdo da hierarquia de classes compartilhadas. Além disso, a técnica tradicional de animação de processos foi ligeiramente modificada de modo a minimizar em parte a ausência de informações mais precisas sobre a evolução temporal dos eventos sendo analisados.

Do ponto de vista de implementação, a ferramenta foi desenvolvida de maneira modular, possibilitando a incorporação futura de novos módulos.

Finalmente, introduziu-se ainda um mecanismo poderoso para a representação de eventos de comunicação em esquemas baseados em memória compartilhada, através da utilização da abstração fornecida pelas classes no paradigma de objetos. Tal mecanismo permite que o tratamento de eventos deste tipo seja realizado de modo extremamente conciso e eficiente.

REFERÊNCIAS BIBLIOGRÁFICAS

[CARD 91] CARDOZO, E. et al. **DPSK+P:** a distributed problem solving kernel for object oriented applications. /A ser publicado/

[CARD 87] CARDOZO, E.; TALUKDAR, S. N. A kernel for distributed problem solving. In: SIMPÓSIO BRASILEIRO EM INTELIGÊNCIA ARTIFICIAL, 4., Uberlândia, 1987. **Anais**. Uberlândia, UFU, 1987. p.267-77.

[ENGE 88] ENGELMORE, R.; MORGAN, T., ed. **Blackboard systems**. Reading, Addison-Wesley, 1988.

[ERMA 80] ERMAN, L. D. et al. The Hearsay-II speech-understanding system: integrating knowledge to resolve uncertainty. **Computing Surveys**, v.12, n.2, p.213-53, June 1980.

[FOX 81] FOX, M. S. An organizational view of distributed systems. **IEEE Transactions on Systems, Man, and Cybernetics**, v.11, n.1, p.70-80, Jan. 1981.

[TAKA 90] TAKAHASHI, T.; LIESENBERG, H. K. E. **Programação orientada a objetos**. São Paulo, IME-USP, 1990. /Apresentado à 7. Escola de Computação, São Paulo, 1990/

[STRO 88] STROUSTRUP, B. What is object-oriented programming. **IEEE Software**, v.5, n.3, p.10-20, May 1988.

[STEF 86] STEFIK, M.; BOBROW, D. G. Object-oriented programming: themes and variations. **The AI Magazine**, v.6, n.4, p.40-62, Jan. 1986.

[DOWE 89] MCDOWELL, C. E.; HELMBOLD, D. P. Debugging concurrent programs. **Computing Surveys**, v.21, n.4, p.593-622, Dec. 1989.

[JOYC 87] JOYCE, J. et al. Monitoring distributed systems. **ACM Transactions on Computer Systems**, v.5, n.2, p.121-50, May 1987.

[BATE 90] BATES, P. Debugging heterogeneous distributed systems using event-based models of behavior. **SIGPLAN Notices**, v.24, n.1, p.11-22, Jan. 1989. /Apresentado ao ACM SIGPLAN and SIGOPS Workshop on Parallel and Distributed Debugging, Madison, 1988/

[SICH 91] SICHMAN, J. S. **Uma ferramenta de monitoração para um núcleo de resolução distribuída de problemas orientado a objetos.** São Paulo, 1991. 166 p. Dissertação (Mestrado) - Escola Politécnica, Universidade de São Paulo.

[CHRI 85] CHRISTIAN, K. **Sistema operacional UNIX.** Rio de Janeiro, Campus, 1985.

[KERN 78] KERNIGHAN, B. W.; RITCHIE, D. M. **The C programming language.** Englewood Cliffs, Prentice Hall, 1978.

[JONE 89] JONES, O. **Introduction to the X Window system.** 4.ed. Englewood Cliffs, Prentice Hall, 1989.

BOLETINS TÉCNICOS - TEXTOS PUBLICADOS

BT/PCS/9301 - Interligação de Processadores através de Chaves Ômicron - GERALDO LINO DE CAMPOS, DEMI GETSCHKO

BT/PCS/9302 - Implementação de Transparência em Sistema Distribuído - LUÍSA YUMIKO AKAO, JOÃO JOSE NETO

BT/PCS/9303 - Desenvolvimento de Sistemas Especificados em SDL - SIDNEI H. TANO, SELMA S. S. MELNIKOFF

BT/PCS/9304 - Um Modelo Formal para Sistemas Digitais à Nível de Transferência de Registradores - JOSE EDUARDO MOREIRA, WILSON VICENTE RUGGIERO

BT/PCS/9305 - Uma Ferramenta para o Desenvolvimento de Protótipos de Programas Concorrentes - JORGE KINOSHITA, JOÃO JOSÉ NETO