Um Modelo Multiparadigma para Desenvolvimento de Software Paralelo e Distribuído

Jorge Luis Victória Barbosa[1], Cláudio Fernando Resin Geyer[2]

[1] Escola de Informática, Universidade Católica de Pelotas (UCPel)
Caixa Postal 402, Pelotas, RS, Brasil
{barbosa@atlas.ucpel.tche.br}
[2] Instituto de Informática, Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15064, Porto Alegre, RS, Brasil
{geyer@inf.ufrgs.br}

Abstract — **This paper presents the Holoparadigm, a new paradigm to parallel and distributed software development. Holoparadigm stimulates the subliminal modeling of the parallelism and its automatic exploitation (implicit parallelism). The proposal is based on multiparadigm and software architecture researches. First of all, the text describes the paradigm's genesis and its semantic (Holosemantic). After that, the distribution and mobility in the work's scope, the architecture style proposed and the principles of a language (Hololanguage) based on the paradigm are discussed. Finally, a plataform of development and execution are described.**

Keywords — **Multiparadigm, Software Architecture, Distributed and Parallel Processing, Holoparadigm.**

I. INTRODUÇÃO

Nos últimos anos o tema multiparadigma vem sendo pesquisado continuamente [PLA 91, MUL 95, NGK 95, AMA 96, CIA 96, LEE 97, ROY 97, HAR 98, HAR 99, HOL 00]. Os pesquisadores deste tema propõem a criação de modelos de desenvolvimento de software através da integração de paradigmas básicos (principalmente, os paradigmas imperativo, em lógica, funcional e orientado a objetos). Através desta proposta buscam dois objetivos, ou seja, a superação das limitações de cada paradigma e a exploração conjunta das suas características benéficas.

Cada paradigma possui fontes implícitas de paralelismo. Por exemplo, o paradigma em lógica suporta a exploração do paralelismo OU e paralelismo E [DUT 99]. Por sua vez, o paradigma orientado a objetos permite a exploração do paralelismo inter-objetos (entre objetos) e intra-objetos (entre métodos de um objeto) [CIA 96].

A integração de paradigmas envolve a integração de suas características. Em complemento, as características de cada paradigma estão relacionados com suas fontes de paralelismo implícito. Portanto, a integração de paradigmas ocasiona a integração de fontes de paralelismo. Surge assim, o interesse na exploração automática de paralelismo no software multiparadigma [NGK 95, CIA 96]. A ampliação dos estudos nessa área conduz a sistemas distribuídos onde podem ser consideradas a mobilidade [ROY 97, IEE 98, LAN 98], o uso de redes como arquiteturas paralelas [JOU 97, IEE 98] e a heterogeneidade de hardware em redes [CAB 97]. Atinge-se assim, o interesse na criação de software paralelo e distribuído com o uso de propostas multiparadigma.

Neste escopo, surge o **Holoparadigma** (de forma resumida, Holo [HOL 00]). Holo é um paradigma de software que possui uma semântica simples e distribuída. Este paradigma integra conceitos de paradigmas básicos e estimula a exploração automática do paralelismo.

Holo está sendo desenvolvido no âmbito do projeto Opera [OPE 00]. Este projeto iniciou suas atividades na Universidade *Joseph Fourier* (Grenoble/França). Atualmente, encontra-se em desenvolvimento na Universidade Federal do Rio Grande do Sul (UFRGS) e na Universidade Católica de Pelotas (UCPel) uma ramificação do Opera.

No âmbito do Opera/Brasil foram desenvolvidas diversas atividades visando a exploração do paralelismo implícito existente no paradigma em lógica [AZE 99, DUT 99, FER 99]. Entre os anos de 1997 e 1999, as atividades do Opera foram englobadas por um projeto multi-institucional denominado Appelo [GEY 99, OPE 00]. As atividades do Opera/Appelo servem de base e estímulo para a criação do Holoparadigma. O paradigma em lógica é um dos paradigmas integrados na proposta.

O artigo está organizado em oito seções. A segunda seção discute a gênese do paradigma. A terceira apresenta a semântica do Holo (Holosemântica). A quarta é dedicada à distribuição e mobilidade no paradigma. A seção cinco apresenta o estilo arquitetônico proposto. Por sua vez, a seção seis descreve uma linguagem que implementa o paradigma (Hololinguagem). A sétima descreve a plataforma de desenvolvimento e execução. Finalmente, a seção oito contém as considerações finais.

II. GÊNESE DO HOLOPARADIGMA

A criação do Holoparadigma pode ser percebida em três níveis de temas (figura 1), ou seja: temas básicos, temas intermediários e tema final. Cada nível abrange os temas de pesquisa abordados durante a gênese. O primeiro nível contém os temas básicos. Neste nível, cada par de temas origina um tema intermediário no próximo nível. Por sua vez, os temas intermediários servem de base para o surgimento do Holoparadigma. Essa seção discute os dois primeiros níveis.

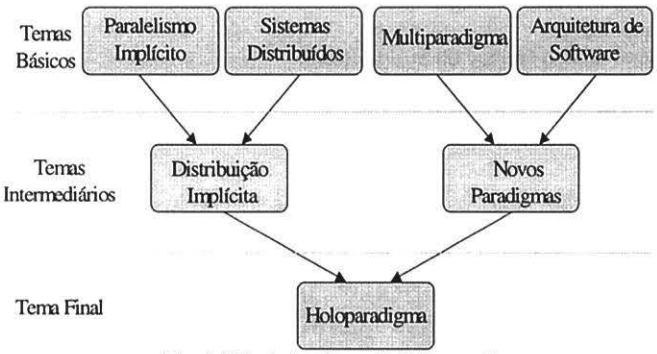

Fig. 1 Níveis da gênese do Holoparadigma

O paralelismo implícito propõe a detecção automática do paralelismo e sua exploração através de mecanismos inseridos no software básico dos sistemas computacionais. Diversos estudos mostram que a exploração do paralelismo implícito em paradigmas não convencionais é mais simples do que sua exploração no paradigma imperativo [AZE 99, DUT 99, FER 99]. Por sua vez, nos últimos anos os sistemas distribuídos têm recebido cada vez mais dedicação tanto dos centros de pesquisa quanto das empresas. Entre os novos tópicos de pesquisa surgidos merecem destaque: uso de redes de computadores como arquiteturas paralelas [JOU 97, IEE 98], mobilidade de código e dados através das redes [ROY 97, IEE 98, LAN 98] e tratamento da heterogeneidade de hardware nas redes de computadores [CAB 97].Uma análise da situação atual e das perspectivas futuras permite a previsão de que em breve os sistemas computacionais serão compostos por uma grande variedade de estações de alto desempenho conectadas por redes de alta velocidade organizadas em topologias diversas. Neste contexto, torna-se interessante a unificação dos temas paralelismo implícito e sistemas distribuídos em um tópico de pesquisa denominado distribuição implícita (tema intermediário). Este tópico busca a exploração automática da distribuição através de mecanismos inseridos no software básico. No âmbito da distribuição implícita devem ser considerados temas que não fazem parte dos estudos no paralelismo implícito, tais como, tratamento automático da mobilidade e heterogeneidade de hardware.

O tema multiparadigma propõe a criação de modelos de desenvolvimento de software através da integração de paradigmas considerados básicos. Através dessa integração busca-se a superação das limitações de cada paradigma e a exploração das características consideradas benéficas. Além disso, na medida em que aumentam o tamanho e a complexidade do software, o projeto e a especificação da estrutura dos sistemas tornam-se mais importantes do que a escolha de algoritmos e estruturas de dados [SHA 96]. Entre os tópicos relacionados com o projeto estrutural de sistemas destacam-se: organização do sistema em componentes; protocolos de comunicação, sincronização e acesso de dados; distribuição física dos componentes; desempenho e evolução dos sistemas. Neste contexto, surge a arquitetura de software [GAR 95, IEE 95, SHA 96]. Este tema de pesquisa dedica-se a descrição de componentes, as interações entre eles e os padrões que guiam sua composição.

Existe uma interessante distinção entre os temas de pesquisa multiparadigma e arquitetura de software. A maioria das propostas relacionadas com o tema multiparadigma surgem na comunidade que pesquisa linguagens de programação [NGK 95, CIA 96, LEE 97]. Por outro lado, as propostas de arquitetura de software surgem na comunidade que pesquisa engenharia de software [SHA 95, VRA 95, ZAV 96]. Ambos os temas de pesquisa dedicam-se ao desenvolvimento de software. No entanto, tratam de níveis diferentes de abstração. A arquitetura enfoca a modelagem, posicionando-se assim em um alto nível de abstração. Por sua vez, as linguagens encontram-se em um baixo nível, pois enfocam a implementação. Ambos podem ser unidos em uma única abordagem de pesquisa dedicada à criação de novos paradigmas de software (tema intermediário). Um paradigma orienta todo o desenvolvimento de software, desde a modelagem até a implementação. A semântica do paradigma deve permear todos os instrumentos a serem utilizados na criação de sistemas computacionais. A unificação dos temas de pesquisa no nível intermediário conduz à criação do Holoparadigma. As próximas seções são dedicadas à sua descrição.

III. HOLOSEMÂNTICA

A semântica do Holo é denominada Holosemântica. A Holosemântica estabelece a utilização de duas unidades de modelagem, ou seja:

- **Unidade de Existência - Ente:** a existência é modelada através de um ente;
- **Unidade de Informação - Símbolo:** a informação é modelada através de símbolos.

A modelagem em Holo utiliza somente essas unidades. O principal objetivo da Holosemântica é o estímulo a exploração automática da distribuição (distribuição implícita).

O ente é a principal abstração do Holoparadigma. Existem dois tipos de entes:

- **Ente Elementar:** ente atômico que não possui níveis de composição;
- **Ente Composto:** ente formado pela composição de outros entes. Não existe limite para níveis de composição.

Um ente elementar é organizado em três partes: **Interface**, **Comportamento** e **História**. A interface descreve suas possíveis relações com os demais entes. O comportamento contém ações que implementam sua funcionalidade. Por sua vez, a história é um espaço de armazenamento compartilhado no interior de um ente.

Um ente composto (figura 2a) possui a mesma organização do elementar, no entanto, suporta a existência de outros entes na sua composição (**entes componentes**). Cada ente possui uma história. A história fica encapsulada no ente e, no caso dos entes compostos, é compartilhada pelos entes componentes. Os entes componentes participam do desenvolvimento da história compartilhada e sofrem os reflexos das mudanças históricas. Sendo assim, podem existir vários níveis de encapsulamento da história. Os entes acessam somente a história no seu nível. A figura 2b mostra um ente composto de três níveis e exemplifica os níveis de encapsulamento da história.

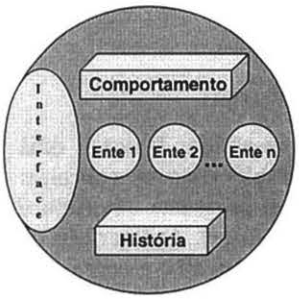

Fig. 2a Ente Composto

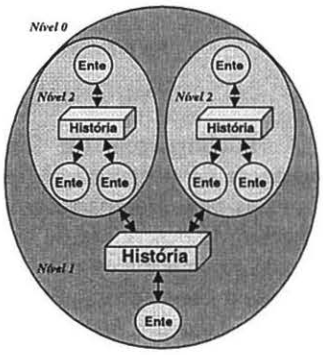

Fig. 2b Exemplo de composição (3 níveis)

Um ente assemelha-se a um **objeto** do paradigma orientado a objetos. Do ponto de vista estrutural, a principal diferença consiste na história, a qual atua como uma forma alternativa de comunicação e sincronização. Além disso, existe maior enfoque na composição e suporte implícito da mobilidade. Um ente composto assemelha-se a um **grupo** [BIR 93]. Neste caso, a história atua como um espaço compartilhado vinculado ao grupo (ente composto).

O Holoparadigma propõe a utilização do processamento simbólico como a base para o tratamento de informações. Essa característica é herdada do paradigma em lógica. Neste sentido, a variável lógica e o casamento de padrões através da unificação são consideradas a base do tratamento de símbolos. O símbolo é o átomo de informação no Holoparadigma. Os símbolos são utilizados para descrição dos entes e de suas relações.

IV. Distribuição e Mobilidade

O Holoparadigma busca a distribuição implícita através da Holosemântica. Neste escopo, um ente pode assumir dois **estados de distribuição**:

- **Centralizado:** um ente está centralizado quando localiza-se em apenas um nodo de um sistema distribuído. Entes elementares estão sempre centralizados. Um ente composto está centralizado se todos os seus entes componentes estão centralizados;
- **Distribuído:** um ente está distribuído quando localiza-se em mais de um nodo de um sistema distribuído. Entes elementares não podem estar distribuídos. Um ente composto está distribuído se os entes componentes estão distribuídos.

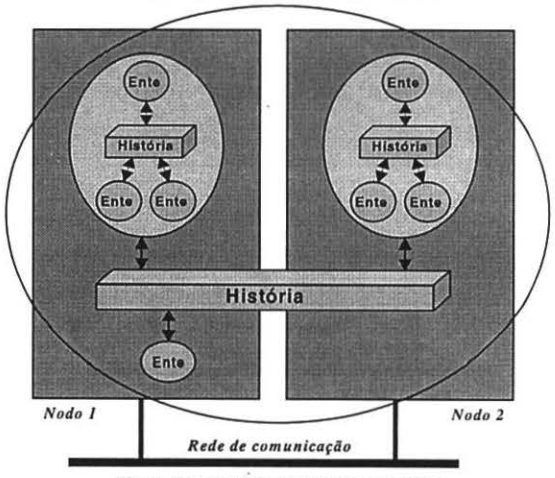

Fig. 3 Distribuição no Holoparadigma

A figura 3 exemplifica uma possível distribuição para o ente mostrado na figura 2b. O ente encontra-se distribuído em dois nodos da arquitetura distribuída. A história de um ente distribuído é denominada **história distribuída**. A distribuição da história é baseada em técnicas de **memória compartilhada distribuída** (DSM) [PRO 99].

A mobilidade [ROY 97, IEE 98, LAN 98] é a capacidade que permite o deslocamento de um ente. No âmbito do Holoparadigma existem dois tipos de mobilidade:

- **Mobilidade Lógica:** a mobilidade lógica relaciona-se com o deslocamento a nível de modelagem, ou seja, sem considerações sobre a plataforma de execução. Neste contexto, um ente se move quando cruza uma ou mais fronteiras de entes;
- **Mobilidade Física:** a mobilidade física relaciona-se com o deslocamento entre nodos de uma arquitetura distribuída. Neste contexto, um ente se move quando desloca-se de um nodo para outro.

A figura 4 exemplifica duas possíveis mobilidades no ente mostrado na figura 3. Após um deslocamento, o **ente móvel** não possui mais acesso a história do **ente origem**

(figura 4, mobilidade A). No entanto, passa a ter acesso a história do **ente destino**. Neste caso, somente ocorrerá mobilidade física se os entes origem e destino estiverem alocados em diferentes nodos de uma arquitetura distribuída. As mobilidades lógica e física são independentes. A ocorrência de um tipo de mobilidade não implica na ocorrência da outra. Merece atenção o caso da mobilidade física não implicar obrigatoriamente na mobilidade lógica (figura 4, mobilidade B). Neste caso, o ente movido continua com a mesma visão da história (suportada pela memória compartilhada distribuída).

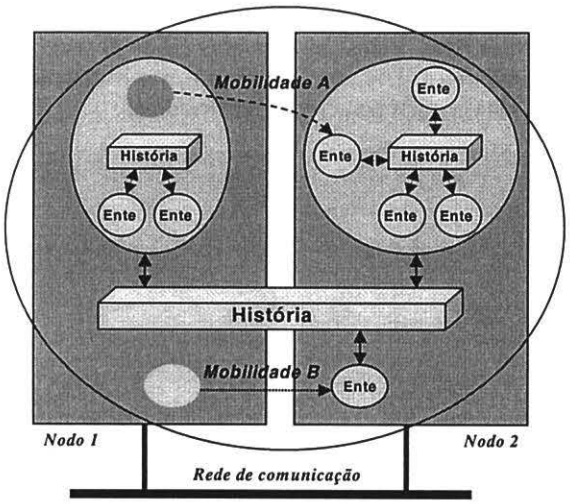

Fig. 4 Mobilidade no Holoparadigma

V. ESTILO ARQUITETÔNICO

Um paradigma cria abstrações que servem como instrumentos de modelagem de software. No âmbito da arquitetura de software [IEE 95, SHA 96] foram definidos vários estilos arquitetônicos para modelagem de sistemas. Neste contexto, destaca-se o estilo ***blackboard*** [VRA 95]. Este estilo é composto de três partes, ou seja: componentes de software (*knowledge sources*), armazenamento central (*blackboard*) e controle. Este estilo é baseado na **invocação implícita**, ou seja, os componentes anunciam eventos através do *blackboard*. Além disso, componentes podem registrar seu interesse em um determinado evento. Sendo assim, quando um evento é anunciado, o controle invoca os componentes que aguardavam o anúncio. O anúncio pode invocar de forma implícita vários componentes.

A implementação do Holoparadigma depende da escolha de um estilo arquitetônico que suporte de forma adequada suas principais abstrações. A organização de um ente composto assemelha-se a forma como um *blackboard* é organizado, ou seja, vários componentes que compartilham um armazenamento. Neste caso, os componentes são entes e o repositório é a história. Em Holo, a história não serve apenas para armazenamento de informações compartilhadas. Parte do controle dos entes é responsabilidade da história. Essa característica é obtida com a utilização do estilo *blackboard* (invocação implícita). A história é um *blackboard* lógico (figura 5) onde os entes podem realizar consultas, inserções (*assert*) e extrações (*retract*).

Existem várias limitações estabelecidas pelo uso da invocação implícita [SHA 96]. O uso da **invocação explícita** conjuntamente com a invocação implícita é salientando como uma solução para essas limitações. Em Holo ambos os estilos de invocação são utilizados (figura 5). Os entes influenciam outros entes através da história (invocação implícita), mas também podem trocar informações diretamente (invocação explícita).

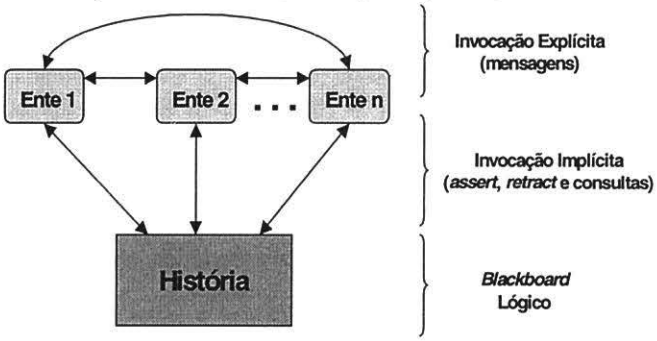

Fig. 5 Estilo Arquitetônico do Holo

VI. HOLOLINGUAGEM

A Hololinguagem (de forma resumida, Holo) é uma linguagem de programação que implementa os conceitos propostos pelo Holoparadigma. Um programa em Holo é composto de descrições de entes (*beings*). A descrição de um ente é composta de duas partes: cabeçalho e corpo. Por sua vez, o corpo é organizado em cinco partes: *creation*, *interface*, *behavior*, *history* e *extinction*. A figura 6 mostra a organização de um *being*.

O cabeçalho contém o nome do *being* e uma descrição das heranças. Holo utiliza herança múltipla seletiva. Um ente pode herdar de outros entes quaisquer uma de suas partes (por exemplo, *interface*, *behavior* ou *history* de entes diferentes). No entanto, uma parte pode ser herdada de apenas um ente (por exemplo, a *behavior* não pode ser herdada de dois entes). Na *creation* são colocadas as ações que serão automaticamente executadas no momento de criação de um ente. Na *interface* são inseridos os cabeçalhos das ações que podem ser acessadas por outros entes. Na *behavior* são descritas as ações que suportam a funcionalidade de um ente. Uma ação somente pode constar na *interface* se existe na *behavior*. A *history* é uma área de memória compartilhada pelas ações descritas na *behavior* e pelos entes componentes. Nessa área são armazenadas informações no formato de lógica, ou seja, predicados lógicos no formato Prolog. Por sua vez, a *extinction* contém ações que serão executadas no momento da extinção de um

ente. A *creation* e a *extinction* são opcionais. A nível organizacional um *being* não explicita seu tipo, ou seja, elementar ou composto. Essa é uma característica que varia durante a execução e sofre influência da mobilidade.

Em Holo existem cinco tipos de ações, ou seja: ações lógicas (*logic actions* - LA), ações imperativas (*imperative actions* - IA), ações lógicas modulares (*modular logic actions* - MLA), ações imperativas modulares (*modular imperative actions* - MIA) e ações multiparadigma modulares (*modular multiparadigm actions* - MMA). As LAs são implementadas com a utilização de predicados lógicos. As IAs utilizam comandos imperativos adaptados para o Holo (mobilidade, acesso à história, envio e recebimento de mensagens). As MLAs e MIAs implementam módulos contendo ações lógicas e imperativas. Finalmente, as MMAs suportam módulos contendo ações lógicas e imperativas integradas.

```
being <nome> inherit ( <nome>( <seleção> ),..,<nome>( <seleção>))
{
 creation
 {
  Ações automaticamente executadas na criação de um ente;
 }
 interface
 {
  Cabeçalho das ações que podem ser acessadas por outros entes;
 }
 behavior
 {
  Ações que descrevem a funcionalidade de um ente;
 }
 history
 {
  Armazenamento compartilhado pelas ações e entes componentes;
 }
 extinction
 {
  Ações automaticamente executadas na extinção de um ente;
 }
}
```

Fig. 6 Estrutura de um ente na linguagem Holo

VII. PLATAFORMA DE DESENVOLVIMENTO E EXECUÇÃO

Uma plataforma é um conjunto de software e hardware que suporta o desenvolvimento e a execução de sistemas computacionais. A figura 7 apresenta a organização da plataforma do Holoparadigma (Holoplataforma).

A Holoplataforma é composta de duas partes, ou seja:

- **Plataforma de desenvolvimento:** conjunto de ferramentas utilizadas na construção de softwares (ferramenta CASE e compilador);
- **Plataforma de execução:** conjunto de hardware e software utilizado como suporte à execução de programas. A camada de software que interage com o hardware é denominada **ambiente de execução**.

As abstrações propostas pelo Holoparadigma são independentes de hardware. No entanto, existe uma orientação do paradigma para arquiteturas paralelas e distribuídas. Quando o hardware for paralelo (por exemplo, *cluster* de estações), as duas principais características a serem exploradas são:

- **Suporte à mobilidade:** o código virtual cria uma camada de abstração que facilita o tratamento da mobilidade dos entes;
- **Suporte à DSM hierárquica e dinâmica:** a distribuição envolve o compartilhamento de armazenamento (história) entre entes localizados em nodos diferentes. Existem vários níveis de história (hierárquica) e adaptação à mobilidade durante a execução (dinâmica).

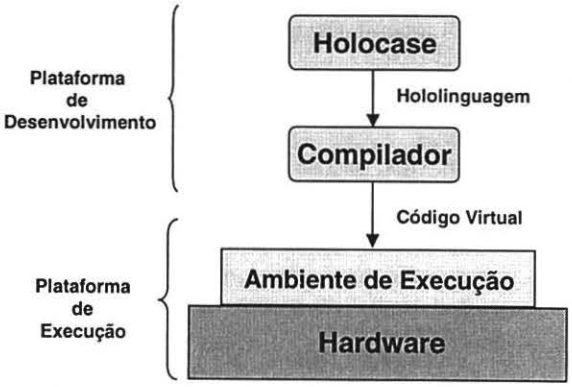

Fig. 7 Plataforma do Holoparadigma

VIII. CONCLUSÕES

Este artigo propõe o Holoparadigma. Destacam-se como principais conclusões do projeto:

- a exploração automática do paralelismo é estimulada com a utilização de uma semântica paralela e distribuída;
- a utilização de memória distribuída compartilhada permite a implementação da história distribuída. No entanto, os níveis de composição e a mobilidade necessitam de memória compartilhada hierárquica e dinâmica;
- as mobilidades lógica e física são independentes e podem ser tratadas de forma distinta;
- a utilização conjunta das invocações implícita e explícita cria um estilo arquitetônico que permite a implementação das principais abstrações (ente e história) propostas pelo Holoparadigma.

Futuras atividades concretizarão a proposta. Será criado um ambiente de execução que suporte os conceitos do Holoparadigma, principalmente, DSM dinâmica e hierárquica e distinção entre mobilidades lógica e física. Além disso, estão sendo especificadas a Hololinguagem, o código virtual multiparadigma e implementado um compilador. A evolução do projeto pode ser acompanhada através de páginas WWW [HOL 00].

REFERÊNCIAS

[AMA 96] AMANDI, Analía; PRICE, Ana. A Linguagem OWB: Combinando Objetos e Lógica. *I Simpósio Brasileiro de Linguagens de Programação*, p.305-318, 1996.

[AZE 99] AZEVEDO, Silvana C., BARBOSA, Jorge L. V.; GEYER, Cláudio F. R. *Automatização da Análise Global no modelo Granlog*. XXV Congresso Latinoamericano de Informática, Asuncion, Paraguai,p.601-612, 1999.

[BIR 93] BIRMAN, Kenneth P. *The Process Group Approach to Reliable Distributed Computing*. Communications of the ACM, v.36, n.12, p.37-53, december 1993.

[CAB 97] CABILLIC, G.; PUAUT, I. *Stardust: An Environment for Parallel Programming on Networks of Heterogeneous Workstations*. Journal of Parallel and Distributed Computing, v.40, n.1, p.65-80, january 1997.

[CIA 96] CIAMPOLINI, A. et al. *Distributed Logic Objects*. Computer Languages, v.22, n.4, p.237-258, december 1996.

[DUT 99] DUTRA, Inês de C., COSTA, Vítor S., BARBOSA, Jorge L. V.; GEYER, Cláudio F. R. *Using Compile-Time Granularity Information to Support Dynamic Work Distribution in Parallel Logic Programming Systems*. XI Symposium on Computer Architecture and High Performance Computing, SBC,p.248-254, 1999.

[FER 99] FERRARI, Débora N.; VARGAS, Patrícia K.; GEYER, Cláudio F. R.; BARBOSA, Jorge L. V. *Modelo de Integração PloSys-GRANLOG: aplicação da análise de granulosidade na exploração do paralelismo OU*. XXV Congresso Latinoamericano de Informática,Asuncion, Paraguai, p.911-922, 1999.

[GAR 95] GARLAN, David et al. *Research Directions in Software Engineering*. ACM Computing Surveys, v.27, n.2, p.257-276, june 1995.

[GEY 99] GEYER, Cláudio F. R.; BARBOSA, Jorge L. V.; et al. *The APPELO Project - Parallel Environment for Logic Programming*. PROTEM-CC'99 Projects Evaluation Workshop Fase III, CNPQ, p.421-454, 1999.

[HAR 98] HARIDI, Seif et al. *Programming Languages for Distributed Applications*. New Generating Computing, v.16, n.3, p.223-261, 1998.

[HAR 99] HARIDI, Seif et al. *Efficient Logic Variables for Distributed Computing*. ACM Transactions on Programming Languages and Systems, v. 21, n.3, p.569-626, may 1999.

[HOL 00] *Holoparadigma*. Páginas WWW do projeto, http://www.inf.ufrgs.br/~holo, julho de 2000 .

[IEE 95] *IEEE Transactions on Software Engineering*, v.21, n.4, april 1995. (Special Issue on Software Architecture)

[IEE 98] *IEEE Transactions on Software Engineering*, v.24, n.5, may 1998. (Special Issue on Mobility)

[JOU 97] *Journal of Parallel and Distributed Computing*, v.40, n.1, january 1997. (Special Issue on Workstation Clusters and Network-Based Computing)

[LAN 98] LANGE, Danny B. *Mobile Objects and Mobile Agents: The Future of Distributed Computing?* ECOOP'98 Object-Oriented Programming, Springer-Verlang, p.1-12, 1998.

[LEE 97] LEE, J. H. M.; PUN, P. K. C. *Object Logic Integration: A Multiparadigm Design Methodology and a Programming Language*. Computer Languages, v.23, n.1, p.25-42, april 1997.

[MUL 95] MULLER, Martin et al. *Multiparadigm Programming in Oz*. Visions for the Future of Logic Programming: Laying the Foundations for a Modern Sucessor of Prolog. Oregon, 1995.

[NGK 95] NG, K. W.; Luk, C. K. *I+: A Multiparadigm Language for Object-Oriented Declarative Programming*. Computer Languages, v.21, n.2, p. 81-100, july 1995.

[OPE 00] *OPERA*. Páginas WWW do projeto. http://www.inf.ufrgs.br/procpar/opera, julho de 2000.

[PLA 91] PLACER, John. *The Multiparadigm Language G*. Computer Languages, v.16, n.3/4, p.235-258, 1991.

[PRO 99] *Proceedings of the IEEE*, v.87, n.3, march 1999. (Special Issue on Distributed Shared Memory Systems)

[ROY 97] ROY, Peter V. et al. *Mobile Objects in Distributed Oz*. ACM Transactions on Programming Languages and Systems, v.19, n.5, p.804-851, september 1997.

[SHA95] SHAW, M. et al. *Abstractions for Software Architecture and Tools to Support Them*. IEEE Transactions on Software Engineering, v.21, n.4, p.314-335, april 1995.

[SHA 96] SHAW, Mary; Garlan, David. *Software Architecture: Perspectives on an Emerging Discipline*. New Jersey: Prentice-Hall, 1996. 242p.

[VRA 95] VRANED, Sanja; Stanojevic, Mladen. *Integrating Multiple Paradigms within the Blackboard Framework*. IEEE Transactions on Software Engineering, v.21, n.3, p.244-262, march 1995.

[ZAV96] ZAVE, Pamela; JACKSON, Michael. *Where Do Operations Come From? A Multiparadigm Specification Technique*. IEEE Transactions on Software Engineering, v.22, n.7, p.508-528, july 1996.