# Holoparadigma: Desenvolvimento de Software Multiparadigma Distribuído

Jorge Luis Victória Barbosa[1]

Universidade Católica de Pelotas - UCPel
Escola de Informática
Caixa Postal 402 - CEP 96010-000
Pelotas, RS, Brasil
barbosa@atlas.ucpel.tche.br

Cláudio Fernando Resin Geyer[2]

Universidade Federal do Rio Grande do Sul - UFRGS
Instituto de Informática
Caixa Postal 15064 - CEP 91591-970
Porto Alegre, RS, Brasil
geyer@inf.ufrgs.br

## Resumo

Este artigo propõe um paradigma para o desenvolvimento de software distribuído, denominado Holoparadigma. O Holoparadigma possui uma semântica simples e distribuída. Sendo assim, estimula a modelagem subliminar da distribuição e sua exploração automática (distribuição implícita). A proposta é baseada em estudos relacionados com multiparadigma e arquitetura de software. Inicialmente, o texto apresenta tópicos relacionados com a gênese do paradigma. Logo após, discute-se sua semântica (Holosemântica), a distribuição e mobilidade no escopo do trabalho, o estilo arquitetônico proposto e os princípios de uma linguagem baseada no paradigma (Hololinguagem). Finalmente, aborda-se sua plataforma de desenvolvimento e execução.

**Palavras-chave:** Multiparadigma, Arquitetura de Software e Sistemas Distribuídos.

## Abstract

This paper proposes a development paradigm to distributed software, called Holoparadigm. Holoparadigm has a simple and distributed semantic. Therefore, it stimulates the subliminal modeling of the distribution and its automatic exploitation (implicit distribution). The proposal is based on multiparadigm and software architecture researches. First of all, topics related to genesis of paradigm are presented. After that, its semantic (Holosemantic), the distribution and mobility in the work's scope, the architecture style proposed and the principles of a language based on paradigm (Hololanguage) are discussed. Finally, a developement and execution plataform is described.

**Keywords:** Multiparadigm, Software Architecture and Distributed Systems.

---

[1]Professor na Universidade Católica de Pelotas (UCPel/RS), Tecnólogo em Processamento de Dados (UCPel/RS, 1989), Engenheiro Eletricista (UCPel/RS, 1990), Especialista em Engenharia de Software (UCPel/RS, 1992), Mestre em Ciência da Computação (UFRGS/RS, 1996), Doutorando em Ciência da Computação (UFRGS/RS, 2000).

[2]Professor na Universidade Federal do Rio Grande do Sul (UFRGS/RS), Engenheiro Mecânico (UFRGS/RS, 1978), Mestre em Ciência da Computação (UFRGS/RS, 1986), Doutor em Informática (Université Joseph Fourier, Grenoble, França, 1991), Pós-doutorado em Informática (Université Joseph Fourier, Grenoble, França, 1996).

# 1 Introdução

Nos últimos anos o tema multiparadigma vem sendo pesquisado continuamente [PLA91, MUL95, NGK95, AMA96, CIA96, LEE97, ROY97, HAR98, HAR99, HOL00]. 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 consideradas benéficas.

O paradigma imperativo dificulta a exploração automática do paralelismo. Este fato resulta do uso de comandos de controle e variáveis que suportam atribuição destrutiva. Um programa pode ter vários níveis de dependências de dados e de controle. Por outro lado, os paradigmas não convencionais possuem fontes implícitas e paralelismo, estimulando assim sua exploração automática. Por exemplo, o paradigma em lógica suporta a exploração do paralelismo OU e paralelismo E [DUT99]. 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) [CIA96].

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 no estudo da exploração automática de paralelismo no software multiparadigma [NGK95, CIA 96]. A ampliação dos estudos nessa área conduz a considerações sobre sistemas distribuídos onde devem ser consideradas a mobilidade [ROY97, IEE98, LAN98], o uso de redes como arquiteturas paralelas [JOU97, IEE98] e a heterogeneidade de hardware em redes [CAB97]. Atinge-se assim, o interesse na criação de software distribuído com o uso de propostas multiparadigma [CIA96, ROY97, HAR98, HAR99].

Este artigo propõe o **Holoparadigma** (de forma resumida, **Holo** [HOL00]). 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 da distribuição. Holo está sendo desenvolvido no âmbito do projeto OPERA [OPE00]. 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 paralelimo implícito existente no paradigma em lógica. 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 nove seções. A segunda seção apresenta a gênese do paradigma, ou seja, as principais considerações que motivaram sua criação. A seção três descreve a semântica do Holo (Holosemântica). A quarta seção é dedicada à distribuição e mobilidade no paradigma. A seção cinco apresenta o estilo arquitetônico proposto. A sexta seção descreve uma linguagem que implementa os conceitos do Holo (Hololinguagem). A seção sete apresenta a plataforma de desenvolvimento e execução. A oitava seção discute trabalhos relacionados. Por sua vez, a seção nove contém conclusões e trabalhos futuros.

## 2 Gênese do Holoparadigma

A criação do Holoparadigma pode ser percebida em três níveis de temas (figura 2.1), 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.

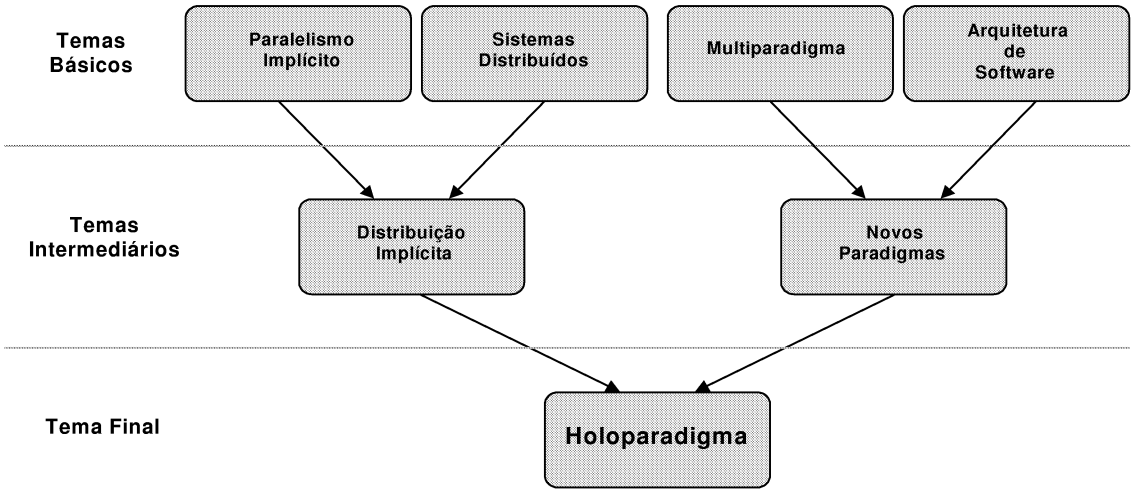


Figura 2.1 - Níveis da gênese do Holoparadigma

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 [AZE99, DUT99, FER99]. 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 [JOU97, IEE98], mobilidade de código e dados através das redes [ROY97, IEE98, LAN98] e tratamento da heterogeneidade de hardware nas redes de computadores [CAB97].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 [SHA96]. 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 [IEE95, GAR95, SHA96]. 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 [NGK95, CIA96, LEE97]. Por outro lado, as propostas de arquitetura de software surgem na comunidade que pesquisa engenharia de software [SHA95, VRA95, ZAV96]. 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.

# 3 Holosemântica

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

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

A modelagem em Holo utiliza somente essas unidades. Sendo assim, torna-se simples a criação e compreensão dos modelos computacionais. O principal objetivo da Holosemântica é o estímulo a exploração automática da distribuição (distribuição implícita). A figura 3.1 mostra uma representação da semântica do Holo.

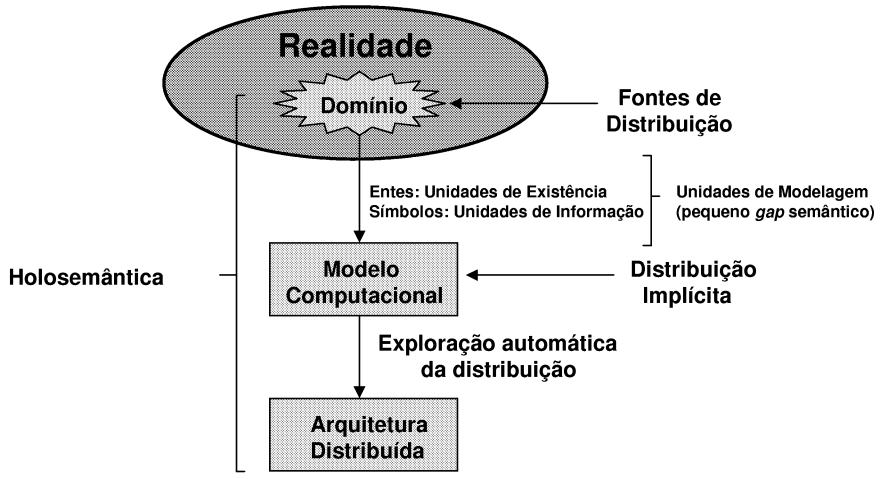


Figura 3.1 -Holosemântica aplicada a modelagem

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 (figura 3.2a) 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 3.2b) possui a mesma organização, 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 3.2c mostra um ente composto de três níveis e exemplifica os níveis de encapsulamento da história.

Um ente elementar 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** [BIR93]. Neste caso, a história atua como um espaço compartilhado vinculado ao grupo.

O símbolo é o átomo de informação no Holoparadigma. Os símbolos são utilizados para descrição dos entes (**símbolo existencial**) e de suas relações (**símbolo relacional**). 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.

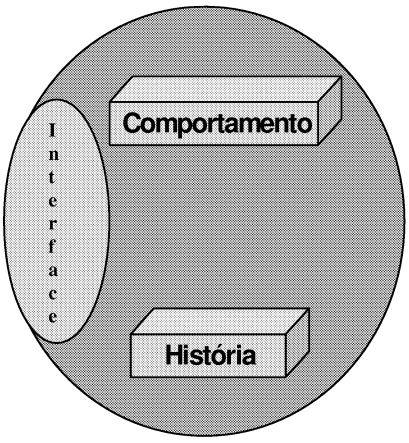


Figura 3.2a – Ente Elementar

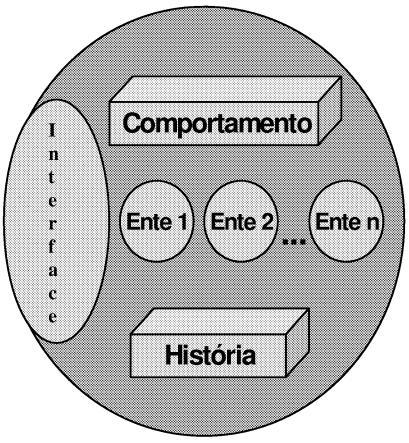


Figura 3.2b – Ente Composto

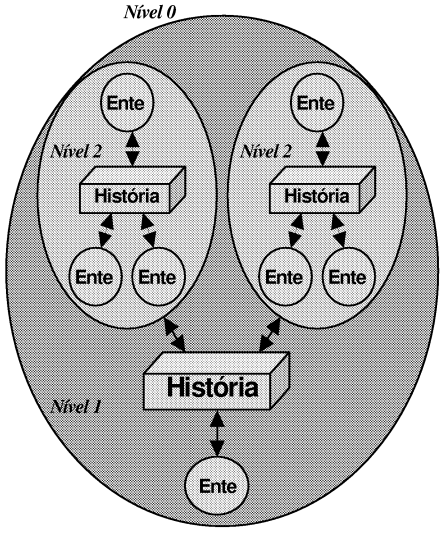


Figura 3.2c – Exemplo de composição (3 níveis)

# 4 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.

A figura 4.1 exemplifica uma possível distribuição para o ente apresentado na figura 3.2c. 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** [PRO99].

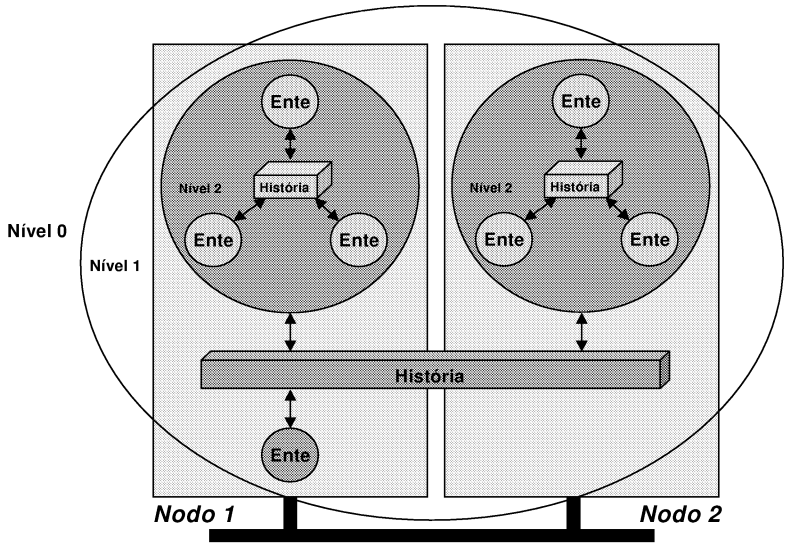


Figura 4.1 - Ente distribuído

A mobilidade [ROY97, IEE98, LAN98] 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 4.2a exemplifica uma possível mobilidade lógica no interior do ente apresentado na figura 3.2c. Conforme exemplificado, após o deslocamento, o **ente móvel** não possui mais acesso a história do **ente origem**. 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 deslocamento não implica na ocorrência do outro. Merece atenção o caso da mobilidade física não implicar obrigatoriamente na mobilidade lógica. Considere-se o ente distribuído mostrado na figura 4.1. Se o ente elementar localizado no nível um, realizar um deslocamento físico conforme mostrado na figura 4.2b, não haverá mobilidade lógica apesar de ter ocorrido mobilidade física. Neste caso, o ente movido continua com a mesma visão da história (suportada pela memória compartilhada distribuída).

# 5 Estilo Arquitetônico

Um estilo arquitetônico define um padrão organizacional para o software [IEE95, GAR95, SHA96]. Shaw e Garlan [SHA96] discutem os estilos mais comumente utilizados. Neste contexto destaca-se o estilo denominado **repositório**. Neste estilo existem dois tipos de componentes, ou seja: uma estrutura de dados central que representa o estado corrente e uma coleção de componentes independentes que operam no armazenamento central.

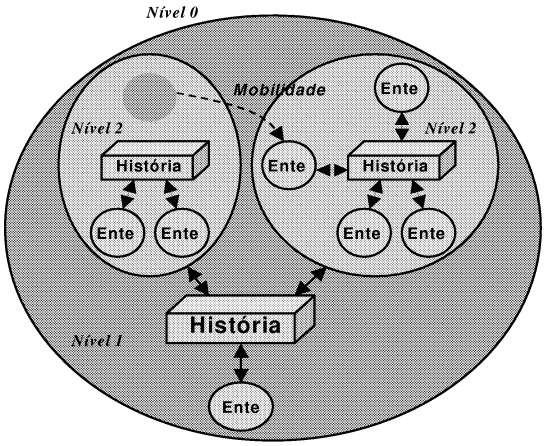


Figura 4.2a – Mobilidade lógica

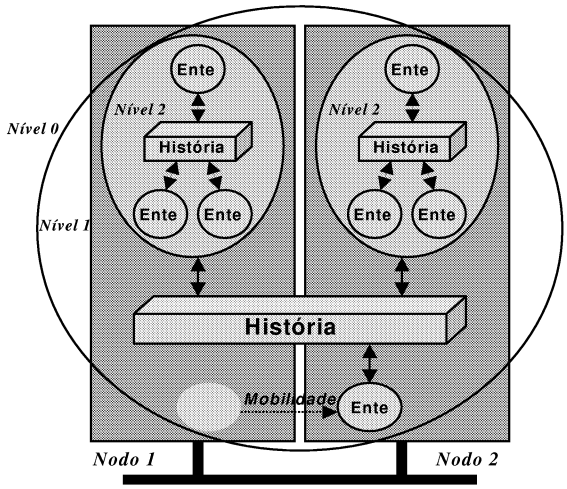


Figura 4.2b – Mobilidade física sem mobilidade lógica

A interação entre o repositório e os componentes pode variar de forma significativa. Se o estado corrente do repositório é a principal fonte de controle dos componentes, o repositório pode ser um ***blackboard*** [VRA95]. Este estilo é composto de três partes (figura 5.1a):

- **Fontes de conhecimento (*Knowledge Sources* - KS):** componentes da aplicação que interagem (comunicação e sincronismo) somente através do armazenamento central;
- **Armazenamento Central (*Blackboard*):** armazena o estado da resolução do problema;
- **Controle:** Controla o *blackboard* fazendo com que os KSs respondam a suas alterações.

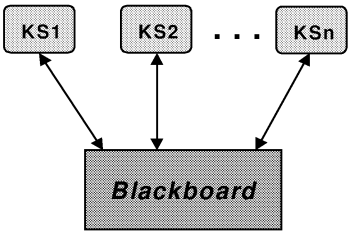


Figura 5.1a – Modelo *Blackboard*

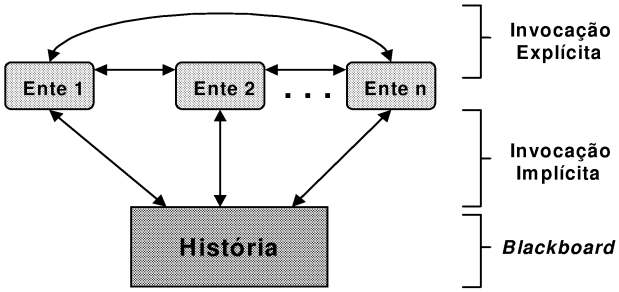


Figura 5.1b – Estilo Arquitetônico do Holo

Este estilo é baseado na **invocação implícita**, ou seja, os componentes anunciam eventos através do *blackboard*. Neste caso, os componentes podem registrar seu interesse em um determinado evento. Sendo assim, quando um evento é anunciado, o controle invoca os KSs que aguardavam o anúncio. O anúncio pode invocar de forma implícita vários KSs.

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 os respositórios são organizados, 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). Existem várias limitações estabelecidas pelo uso da invocação implícita [SHA96]. 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 utililizados. Os entes influenciam outros entes através da história (invocação implícita), mas também podem trocar informações diretamente (invocação explícita). O estilo do Holoparadigma é mostrado na figura 5.1b.

# 6 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 6.1a mostra a organização de um *being*.

```
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;
 }

}
```

Figura 6.1a – Estrutura de um ente na linguagem Holo

```
programa → ente | ente programa

ente → cabeçalho corpo

cabeçalho → being <nome> | being <nome> inherit ( herança )

herança → <nome> ( seleção ) | <nome> ( seleção ) , herança

seleção → opção | opção , seleção

opção → creation | interface | behavior | history | extinction

corpo → criação interface comportamento história extinção

criação → ε | creation { ações }

interface → interface { "Cabeçalho de uma ação" }

comportamento → behavior { ações }

história → history { "Programa em Lógica" }

extinção → ε | extinction { ações }

ações → ação | ação ações

ação → "Ação lógica"
      | "Ação Imperativa"
      | "Ação Modular Lógica"
      | "Ação Modular Imperativa"
      | "Ação Modular Multiparadigma"
```

Figura 6.1b – Gramática simplificada da linguagem Holo

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. A figura 6.1.b mostra uma versão simplificada da gramática da Hololinguagem. A figura abstrai a descrição das ações e o corpo da *interface* e *history* (colocadas entre aspas).

# 7 Plataformas 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 7.1 apresenta a organização da plataforma do Holoparadigma (Holoplataforma). A Holoplataforma é composta de duas partes, ou seja:

- **Plataforma de desenvolvimento:** conjunto de ferramentas de software utilizadas na construção dos sistemas computacionais (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 as arquiteturas distribuídas. Quando o hardware for distribuído, 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).

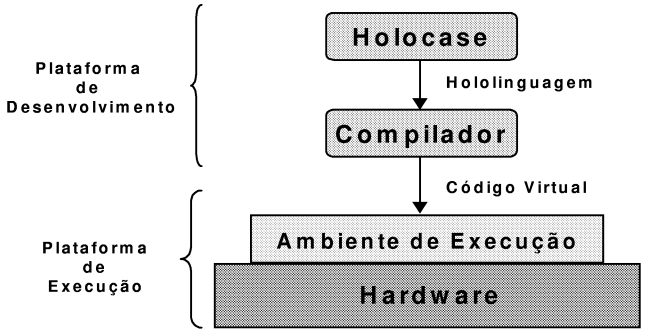


Figura 7.1 - Holoplataforma

# 8 Trabalhos relacionados

Placer [PLA91] e Muller et al [MUL95] apresentam resumos da evolução dos trabalhos de pesquisa multiparadigma. Em ambos os textos destaca-se o interesse da comunidade científica na criação de um modelo multiparadigma ideal que unifique os avanços introduzidos pelos paradigmas básicos. Por sua vez, Ng e Luk [NGK95] apresentam uma interessante tabela que compara vinte e três propostas multiparadigma. Conforme salientado por Placer, grande parte do esforço dedicado à pesquisa multiparadigma enfoca a integração entre dois paradigmas básicos. Entre os primeiros trabalhos que seguiram essa tendência destacam-se as propostas de integração dos paradigmas em lógica e funcional. Hanus [HAN94] apresenta os conceitos básicos envolvidos na integração desses paradigmas. Além disso, o texto descreve várias propostas para uso conjunto da lógica e funções. Neste escopo, merecem destaque ainda os trabalhos de Chakravarty e Lock [CHA97] e Hanus [HAN97].

Bugliesi et al [BUG94] apresentam os esforços da comunidade científica em aplicar no paradigma em lógica os conceitos de modularidade. A modularização soluciona uma das principais limitações do paradigma em lógica, ou seja, a falta de estrutura para desenvolvimento organizado de

grandes sistemas. O surgimento do paradigma orientado a objetos fez com que a comunidade científica percebe-se a possibilidade de sua integração com o paradigma em lógica. Davison [DAV93] apresenta um estudo sobre a integração desses paradigmas destacando as motivações, técnicas e modelos propostos. Wegner [WEG93] apresenta uma análise dos problemas existente na integração objetos/lógica. Amandi e Price [AMA 96] propõem a aplicação dessa integração na criação de agentes. Por sua vez, Lee e Pun [LEE97] criam uma metodologia (OLI) que suporta a modelagem de sistemas com a aplicação conjunta de lógica e objetos.

Merecem destaque ainda, os esforços para o desenvolvimento de sistemas distribuídos baseados em modelos multiparadigma. Ciampolini et at [CIA96] descrevem uma proposta para criação de objetos lógicos distribuídos (DLO). Por sua vez, o modelo $I^+$ [NGK95] suporta a distribuição de objetos que possuem métodos implementados com funções ou predicados lógicos. Além disso, o modelo multiparadigma Oz [SMO95] serve de base para criação de uma plataforma distribuída denominada Mozart [ROY97, HAR98, HAR99].

A tabela 8.1 compara seis propostas multiparadigma ($I^+$,OWB, DLO, OLI, Mozart e Holo). A tabela está organizada em quatro colunas: nome da proposta, paradigmas integrados, sistema de distribuição e estilo utilizado na integração.

Tabela 8.1 - Comparação entre modelos multiparadigma

| Modelo | Paradigmas integrados | Distribuição | Estilo |
|---|---|---|---|
| $I^+$ *[NGK95]* | Objetos, Lógica e Funcional | Objetos distribuídos | Programação declarativa orientada a objetos, ou seja, especificação de objetos através da lógica ou funções |
| OWB *[AMA96]* | Objetos e Lógica | Não enfoca | Suporte à criação de agentes através da inserção de lógica em objetos (nova classe *LogicKnowledge*) |
| DLO *[CIA96]* | Objetos e Lógica | Objetos distribuídos | Processos organizados em Objetos Lógicos implementados através de Cláusulas de Múltiplas Cabeças |
| OLI *[LEE97]* | Objetos e Lógica | Não enfoca | Mapeamento de Objeto ➔ Lógica (classe *Pterm*) e Lógica ➔ Objeto (*Enriched Herbrand Universe*) |
| Mozart *[HAR98]* | Objetos, Lógica e Funcional | Objetos distribuídos | Tarefas conectadas através de um armazenamento compartilhado (*constraint store*) |
| Holo *[HOL00]* | Objetos, Lógica e Imperativo | Entes distribuídos | Entes organizados em níveis hierárquicos e comunicação/sincronização implícita (*blackboard* lógico) e explícita (mensagens) |

# 9 Conclusão

Este artigo propõe um paradigma para o desenvolvimento de software distribuído, denominado Holoparadigma. O paradigma possui uma semântica simples e distribuída. Sendo assim, estimula-se a modelagem subliminar da distribuição e sua exploração automática (distribuição implícita). Destacam-se como principais conclusões surgidas durante o desenvolvimento do Holoparadigma:

- a exploração automática da distribuição é estimulada com a utilização de um paradigma de desenvolvimento que possua uma semântica distribuída;
- os temas multiparadigma e arquitetura de software podem ser integrados para a criação de novos paradigmas de desenvolvimento de software;
- 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.

Futuros trabalhos poderão aperfeiçoar a proposta. Durante o desenvolvimento do paradigma foi detectada uma analogia da organização em níveis dos entes com a organização hierárquica de arquiteturas distribuídas. Essa analogia deve ser pesquisada e formalizada com o intuito de aperfeiçoamento do mapeamento dos entes em nodos de sistemas distribuídos. Em complemento, deverá ser criado um ambiente de execução que suporte os conceitos introduzidos pelo Holo, principalmente, mobilidades lógica e física independentes e memória distribuída compartilhada dinâmica e com níveis hierárquicos. Por sua vez, a Hololinguagem deve ser aperfeiçoada através da inserção do paradigma funcional e melhorias no compilador.

## Referências Bibliográficas


[AMA96] AMANDI, Analía; PRICE, Ana. *A Linguagem OWB: Combinando Objetos e Lógica*. I Simpósio Brasileiro de Linguagens de Programação, SBC, p.305-318, 1996.

[AZE99] 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.

[BIR93] BIRMAN, Kenneth P. *The Process Group Approach to Reliable Distributed Computing*. Communications of the ACM, v.36, n.12, p.37-53, december 1993.

[BUG94] BUGLIESI, Michele; LAMMA Evelina; MELLO, Paola. *Modularity in Logic Programming*. Journal of Logic Programming, New York, v.19/20, p.443-502, May/July 1994.

[CAB97] CABILLIC, Gilbert; PUAUT, Isabelle. *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.

[CHA97] CHAKRAVARTY, Manuel M. T.; LOCK, Hendrik C. R. *Towards the Uniform Implementation of Declarative Languages*. Computer Languages, Elmsford, v.23, n.2-4, p.121-160, July-December 1997.

[CIA96] CIAMPOLINI, A.; LAMMA, E.; STEFANELLI, C; MELLO, P. *Distributed Logic Objects*. Computer Languages, v.22, n.4, p.237-258, december 1996.

[DAV93] DAVISON, Andrew. *A Survey of Logic Programming-Based Object-Oriented*. In: AGHA, G.; WEGNER, P.; YONEZAWA, A. (eds.). Research Direction in Concurrent Object-Oriented Programming. Cambridge: Mit Press, 1993. p.42-106

[DUT99] 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.

[FER99] 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.

[GAR95] GARLAN, David et al. *Research Directions in Software Engineering*. ACM Computing Surveys, v.27, n.2, p.257-276, june 1995.

[GEY99] 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.

[HAN94] HANUS, Michael. *The Integration of Functions into Logic Programming from Theory to Practice*. Journal of Logic Programming, New York, v.19/20, p.583-628, May/July 1994.

[HAN97] HANUS, Michael. *Lazy Narrowing with Simplification*. Computer Languages, Elmsford, v.23, n.2-4, p.61-85, July-December 1997.

[HAR98] HARIDI, Seif et al. *Programming Languages for Distributed Applications*. New Generating Computing, v.16, n.3, p.223-261, 1998.

[HAR99] 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.

[HOL00] *Holoparadigma*. Páginas WWW do projeto, http://www.inf.ufrgs.br/~holo, julho de 2000 .

[IEE95] *IEEE Transactions on Software Engineering*, v.21, n.4, april 1995.(Special Issue on Software Architecture)

[IEE98] *IEEE Transactions on Software Engineering*, v.24, n.5, may 1998. (Special Issue on Mobility)

[JOU97] *Journal of Parallel and Distributed Computing*, v.40, n.1, january 1997. (Special Issue on Workstation Clusters and Network-Based Computing)

[LAN98] LANGE, Danny B. *Mobile Objects and Mobile Agents: The Future of Distributed Computing?* ECOOP'98 Object-Oriented Programming, Springer-Verlang, p.1-12, 1998.

[LEE97] 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.

[MUL95] MULLER, Martin; MULLER, Tobias; ROY, Peter V. *Multiparadigm Programming in Oz*. Visions for the Future of Logic Programming: Laying the Foundations for a Modern Sucessor of Prolog. Oregon, 1995.

[NGK95] 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.

[OPE00] *OPERA*. Páginas WWW do projeto. http://www.inf.ufrgs.br/procpar/opera, abril 2000.

[PLA91] PLACER, John. *The Multiparadigm Language G*. Computer Languages, v.16, n.3/4, p.235-258, 1991.

[PRO99] *Proceedings of the IEEE*, v.87, n.3, march 1999. (Special Issue on Distributed Shared Memory Systems)

[ROY97] 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.

[SHA96] SHAW, Mary; GARLAN, David. *Software Architecture: Perspectives on an Emerging Discipline*. New Jersey: Prentice-Hall, 1996. 242p.

[SMO95] SMOLKA, Gert. *The Oz Programming Model*. In: COMPUTER SCIENCE TODAY, 1995. Proceedings... Berlin: Springer-Verlag, 1995. p. 324-343. (Lecture Notes in Computer Science, v.1000).

[VRA95] VRANES, Sanja; STANOJEVIC, Mladen. *Integrating Multiple Paradigms within the Blackboard Framework*. IEEE Transactions on Software Engineering, v.21, n.3, p.244-262, march 1995.

[WEG93] WEGNER, Peter. *Tradeoffs between Reasoning and Mod*eling. In: AGHA, G.; WEGNER, P.; YONEZAWA, A. (eds.). Research Direction in Concurrent Object-Oriented Programming. Cambridge: Mit Press, 1993. p.22-41

[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.