



Renato Figueiró Maia

**Uma Avaliação do Uso de Linguagens
Dinâmicas no Desenvolvimento de Middleware:
Um Estudo de Caso da Implementação de um
ORB na Linguagem Lua**

Tese de Doutorado

Tese apresentada ao Programa de Pós-graduação em Informática
do Departamento de Informática da PUC-Rio como requisito
parcial para obtenção do título de Doutor em Informática

Orientador: Prof. Renato Fontoura de Gusmão Cerqueira

Rio de Janeiro
Abril de 2009



Renato Figueiró Maia

**Uma Avaliação do Uso de Linguagens
Dinâmicas no Desenvolvimento de Middleware:
Um Estudo de Caso da Implementação de um
ORB na Linguagem Lua**

Tese apresentada ao Programa de Pós-graduação em Informática do Departamento de Informática do Centro Técnico Científico da PUC-Rio como requisito parcial para obtenção do título de Doutor em Informática. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Renato Fontoura de Gusmão Cerqueira

Orientador

Departamento de Informática — PUC-Rio

Profa. Noemi de la Rocque Rodriguez

Departamento de Informática — PUC-Rio

Prof. Roberto Ierusalimschy

Departamento de Informática — PUC-Rio

Prof. Fabio Kon

Departamento de Ciência da Computação — IME-USP

Profa. Thaís Vasconcelos Batista

Departamento de Informática e Matemática Aplicada — UFRN

Prof. Antônio Tadeu Azevedo Gomes

Coordenação de Sistemas e Redes — LNCC

Prof. José Eugenio Leal

Coordenador Setorial do Centro Técnico Científico — PUC-Rio

Rio de Janeiro, 06 de Abril de 2009

Todos os direitos reservados. É proibida a reprodução total ou parcial do trabalho sem autorização da universidade, do autor e do orientador.

Renato Figueiró Maia

Graduou-se em Bacharelado em Ciência da Computação pela UFPA. Obteve o título de Mestre em Ciências pela PUC-Rio. É pesquisador do laboratório Tecnologia em Computação Gráfica (Tecgraf) da PUC-Rio desde 2003, onde trabalhou no desenvolvimento de sistemas de controle de processos industriais em parceria com a Petrobras S/A, assim como em ferramentas de programação para Lua disponíveis como software livre na Internet.

Ficha Catalográfica

Maia, Renato

Uma Avaliação do Uso de Linguagens Dinâmicas no Desenvolvimento de Middleware: Um Estudo de Caso da Implementação de um ORB na Linguagem Lua / Renato Figueiró Maia; orientador: Renato Fontoura de Gusmão Cerqueira. — Rio de Janeiro : PUC–Rio, Departamento de Informática, 2009.

v., 165 f: il. ; 29,7 cm

1. Tese (doutorado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática.

Inclui referências bibliográficas.

1. Informática – Tese. 2. middleware. 3. linguagens dinâmicas. 4. flexibilidade de software. 5. dimensões cognitivas. 6. Lua. 7. OIL. 8. CORBA. I. Cerqueira, Renato. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Agradecimentos

Ao meu orientador, Renato Cerqueira, por ter me dado apoio, incentivo e liberdade para conduzir este trabalho.

Ao CNPq e à PUC-Rio, em particular ao Tecgraf, pelos auxílios concedidos, sem os quais este trabalho não poderia ter sido realizado.

À minha mãe, meus irmãos e minha família, em especial à minha mulher, Kylme, pela compreensão durante minhas ausências em virtude da execução deste trabalho.

Aos meus colegas da PUC-Rio, pela amizade e por tornarem o trabalho diário mais divertido e estimulante, em particular, ao Tomás Guisasola, pelas inúmeras discussões e críticas construtivas sobre este trabalho.

Ao pessoal do Departamento de Informática para a ajuda de todos os dias.

Resumo

Maia, Renato; Cerqueira, Renato. **Uma Avaliação do Uso de Linguagens Dinâmicas no Desenvolvimento de Middleware: Um Estudo de Caso da Implementação de um ORB na Linguagem Lua.** Rio de Janeiro, 2009. 165p. Tese de Doutorado — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

A pesquisa e desenvolvimento de middleware mudou seu foco em alto desempenho para maior flexibilidade. Algo similar aconteceu no desenvolvimento de aplicações em geral com a recente popularização do uso de linguagens dinâmicas, consideradas mais versáteis. Neste trabalho, é feita uma análise da adequação do uso das linguagens dinâmicas na implementação de middleware através de um estudo de caso. O caso em questão é o desenvolvimento do OiL, um middleware na linguagem Lua baseado no modelo de objetos distribuídos, que segue uma implementação modular baseada em componentes reconfiguráveis. O OiL foi desenvolvido durante este trabalho com o objetivo de tirar proveito das facilidades da linguagem Lua para uma implementação simples, flexível e eficiente. É apresentada uma análise da implementação do OiL que busca identificar as características de Lua que permitem implementações de middleware mais flexíveis, ou seja, mais fáceis de modificar. Essa análise baseia-se no arcabouço de Dimensões Cognitivas de Notações proposto para avaliação de notações e linguagens, que é utilizada neste trabalho como critério de avaliação da facilidade de adaptação da implementação do OiL para diferentes usos. São apresentadas também avaliações de desempenho do OiL em relação a outras implementações similares em linguagens menos dinâmicas, como C++ e Java.

Palavras-chave

middleware. linguagens dinâmicas. flexibilidade de software. dimensões cognitivas. Lua. OiL. CORBA.

Abstract

Maia, Renato; Cerqueira, Renato. **An Evaluation of the Use of Dynamic Languages in Development of Middleware**. Rio de Janeiro, 2009. 165p. DSc Thesis — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

The focus of middleware research and development has changed from high performance to more flexibility. A similar change has happened in the application development scenario with the recent popularization of dynamic languages. In this work, we present an analysis of the use of dynamic languages in the implementation of middleware through a use case. Such case is the design and implementation of OiL, a middleware in the Lua language based on the model of distributed objects, which follows a modular implementation based on reconfigurable components. OiL was developed during this work with the purpose of taking advantage of specific Lua features that enable a simple, flexible, and efficient implementation. We present an analysis of OiL's implementation that identifies Lua's characteristics that contributed for a more flexible implementation, that is, an implementation easier to change. This analysis is based on the Cognitive Dimensions of Notations, a framework for evaluation of notations and languages, which is used in this work as the evaluation criteria to measure how easy it is to adapt OiL's implementation in different scenarios. We also compare the performance of OiL with other similar implementations in languages that are less dynamic, such as C++ and Java.

Keywords

middleware. dynamic languages. software flexibility. cognitive dimensions. Lua. OiL. CORBA.

Sumário

1	Introdução	12
1.1	Estrutura do Texto	16
1.2	Convenções Tipográficas	16
2	Conceitos Básicos	17
2.1	Middleware	17
2.2	Linguagens Dinâmicas	18
2.2.1	A Linguagem Lua	20
2.3	Flexibilidade de Software	26
2.3.1	Avaliação de Flexibilidade	27
2.4	Dimensões Cognitivas de Notações	28
2.4.1	Viscosidade	30
2.4.2	Compromissos Prematuros	31
2.4.3	Dependências Ocultas	32
2.4.4	Provisão	33
2.4.5	Avaliação Progressiva	34
2.4.6	Propensão a Erros	34
2.4.7	Mecanismos de Abstração	36
2.4.8	Operações Complicadas	37
2.4.9	Visibilidade	38
2.4.10	Prolixidade	39
2.4.11	Expressividade de Papéis	40
2.4.12	Proximidade de Mapeamento	41
2.4.13	Consistência	42
2.4.14	Notação Secundária	42
3	OiL - <i>ORB in Lua</i>	45
3.1	Visão Geral	46
3.2	Montagens do OiL	48
3.3	Camada de Apresentação	52
3.3.1	<i>Proxies</i> Protegidos	53
3.3.2	<i>Proxies</i> Assíncronos	54
3.3.3	Extensão para Suporte a Interfaces	57
3.3.4	Extensão para Concorrência	58
3.3.5	Interação com a Camada do Protocolo	60
3.3.6	Interceptação de Chamadas	64
3.4	Camada do Protocolo LuDO	65
3.4.1	Extensão para Passagem por Referência	67
3.5	Camada do Protocolo CORBA	68
3.5.1	Extensão para Interceptação	71
3.5.2	Extensão para <i>Marshaling</i> Otimizado	72
3.6	Problemas Encontrados	75
4	Avaliação de Flexibilidade	77
4.1	Referência de Comparação: Mico	78

4.2	Mudança do Protocolo de RMI	80
4.2.1	GIOP no OiL	81
4.2.2	GIOP no Mico	85
4.2.3	LuDO no OiL	92
4.2.4	LuDO no Mico	93
4.3	Mudança de Modelo de Programação	95
4.3.1	Chamadas Síncronas e Assíncronas no OiL	97
4.3.2	Chamadas Síncronas e Assíncronas no Mico	98
4.4	Mudança de Recursos Oferecidos	101
4.4.1	Funcionalidades Modulares no OiL	101
4.4.2	Funcionalidades Modulares no Mico	103
4.4.3	Funcionalidades Transversais no OiL	104
4.4.4	Funcionalidades Transversais no Mico	110
4.5	Avaliação Geral	112
5	Avaliação de Desempenho	121
5.1	Desempenho das Chamadas	122
5.2	Utilização de Memória	125
6	Trabalhos Relacionados	130
6.1	Middleware Flexíveis	130
6.1.1	Mico	130
6.1.2	TAO	131
6.1.3	OpenORB	133
6.1.4	LegORB e UIC	136
6.2	Middleware para Linguagens Dinâmicas	137
6.2.1	CORBA em Linguagens Dinâmicas	138
6.2.2	Pyro	139
7	Conclusão	142
7.1	Utilização do OiL	143
7.2	Avaliação do OiL	145
7.3	Trabalhos Futuros	148
8	Referências Bibliográficas	151
A	Implementação dos Testes de Desempenho	162
A.1	Servidor Utilizado nos Testes	162
A.2	Aplicação Cliente que Realiza os Testes	162
A.2.1	Infra-estrutura para coleta e gravação dos dados de resultado	162
A.2.2	Execução dos testes de invocação simples	163
A.2.3	Execução dos testes de invocação com passagem de seqüências	164

Lista de figuras

2.1	Estruturas de dados em Lua usando diferentes formas sintáticas.	21
2.2	Fecho de função em Lua usando diferentes formas sintáticas.	21
2.3	Objeto em Lua usando diferentes formas sintáticas.	22
2.4	Interpretação de código-fonte em um programa Lua.	22
2.5	Uso de meta-tabelas para implementação de <i>proxies</i> de valores.	24
2.6	Uso de meta-tabelas para implementação de classes de objeto.	25
2.7	Definição de tipo como uma classe Java.	35
2.8	Comparação de proximidade na declaração de classes em C++ e Java.	39
2.9	Relevância das dimensões cognitivas em cada tipo de modificação.	43
2.10	Aspectos positivos e negativos de ling. dinâmicas na flexibilidade.	44
3.1	Criação de <i>servant</i> e acesso com <i>proxy</i> .	47
3.2	Criação de <i>servant</i> CORBA e acesso com <i>proxy</i> tipado.	48
3.3	Especificação da arquitetura básica de ORBs OiL.	49
3.4	Criação de componentes usando o suporte oferecido pelo LOOP.	51
3.5	Arquitetura da camada de apresentação de ORBs OiL	52
3.6	Notificação de eventos com chamadas protegidas e concorrência.	55
3.7	Notificação de eventos com chamadas assíncronas.	56
3.8	Arquitetura da camada de apresentação com extensões.	57
3.9	Interfaces de integração com a camada do protocolo.	62
3.10	Arquitetura da camada básica com interceptação de chamadas.	64
3.11	Arquitetura da camada do protocolo LuDO.	65
3.12	Arquitetura estendida da camada do protocolo LuDO.	67
3.13	Arquitetura da camada do protocolo CORBA.	69
3.14	Arquitetura da chamada CORBA com interceptação de chamadas.	72
3.15	Decodificação iterativa de uma <i>struct</i> de CORBA.	73
3.16	Geração de decodificador de uma <i>struct</i> de CORBA.	74
3.17	Decodificador especializado de uma <i>struct</i> de CORBA.	74
3.18	IDL utilizada para ilustrar a geração de código de <i>marshaling</i> .	75
4.1	Arquitetura do <i>micro-kernel</i> do Mico.	79
4.2	Criação de <i>servant</i> associando informação da sua interface.	83
4.3	Interface da classe <i>IORProfile</i> do Mico.	88
4.4	Trecho que mostra uso de tipagem dinâmica no Mico.	89
4.5	Definição explícita de dependências entre classes no Mico.	91
4.6	Uso de IOR de CORBA contendo referências LuDO no Mico	94
4.7	Implementação de operações síncronas no OiL.	97
4.8	Implementação de operações síncronas no OiL.	98
4.9	Montagens do OiL com suporte para cliente ou servidor.	102
4.10	Sincronização de <i>threads</i> em componentes da camada de RMI.	106
4.11	Resumo das avaliação de flexibilidade do OiL com CDN.	120
5.1	Implementações avaliadas nos experimentos de desempenho.	121
5.2	IDL da interface utilizada nos experimentos de desempenho.	123

5.3	Duração de amostras de 100 chamadas durante a estabilização do ORB.	123
5.4	Duração de 100 chamadas sem parâmetro.	124
5.5	Duração de 100 chamadas cujo parâmetro é uma seqüência de 10 estruturas com valores simples.	126
5.6	Duração de 100 chamadas cujo parâmetro é uma seqüência de 10 referências de objeto.	126
5.7	Utilização de memória dos diferentes servidores durante os experimentos.	127
6.1	Arquitetura do núcleo do TAO.	132
6.2	Arquitetura em camadas do OpenORB v2.	135
6.3	Criação de <i>servant</i> e acesso com <i>proxy</i> usando ORBit2 em Python.	139
6.4	Criação de <i>servant</i> e acesso com <i>proxy</i> usando Pyro.	140

*Coragem! Mais vale errar se arrebatando, do
que preparar-se para nada.*

Darcy Ribeiro (1922–1997)