



**Gustavo Robichez de Carvalho**

**Uma Arquitetura para a Coordenação e  
a Composição de Artefatos de Software**

**Dissertação de Mestrado**

Dissertação apresentada ao Programa de Pós-Graduação em Informática do Departamento de Informática da PUC-Rio como parte dos requisitos parciais para obtenção do título de Mestre em Informática.

Orientadores: Carlos José Pereira de Lucena  
Arndt von Staa

Rio de Janeiro  
Agosto de 2003

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

### **Gustavo Robichez de Carvalho**

Graduou-se em Engenharia de Computação na PUC-Rio em 2000. É pesquisador associado ao Laboratório de Engenharia de Software (LES) da PUC-Rio, atuando na área de Engenharia de Software Baseada em Componentes.

Carvalho, Gustavo Robichez de

Uma arquitetura para a coordenação e a composição de artefatos de software / Gustavo Robichez de Carvalho; orientadores: Carlos José Pereira de Lucena, Arndt von Staa. – Rio de Janeiro: PUC-Rio, Departamento de Informática, 2003.

102 f. ; il. ; 30 cm

Dissertação (mestrado) – Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática.

Inclui referências bibliográficas.

1. Informática – Teses. 2. Engenharia de software. 3. Software – Componentes. 4. Software – Desenvolvimento. 5. Software – Reutilização. I. Lucena, Carlos José Pereira de. II. Staa, Arndt von. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. IV. Título



**Gustavo Robichez de Carvalho**

**Uma Arquitetura para a Coordenação e a  
Composição de Artefatos de Software**

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre pelo Programa de Pós-graduação em Informática do Departamento de Informática do Centro Técnico e Científico da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

**Prof. Carlos José Pereira de Lucena**

Orientador

Departamento de Informática – PUC-Rio

**Prof. Arndt von Staa**

Co-orientador

Departamento de Informática – PUC-Rio

**Prof. Renato Fontoura de Gusmão Cerqueira**

Departamento de Informática – PUC-Rio

**Profª Noemi de La Rocque Rodriguez**

Departamento de Informática – PUC-Rio

**Prof. Hugo Fuks**

Departamento de Informática – PUC-Rio

**Prof. Ney Dumont**

Coordenador Setorial do Centro

Técnico Científico – PUC-Rio

Rio de janeiro, 22 de agosto de 2003

A todos aqueles que, de uma forma ou de outra,  
ajudaram a fazer este trabalho.

## Agradecimentos

A Deus, por tudo o que já aconteceu e por aquilo que está por vir.

Aos meus pais, meus avôs, e minhas irmãs pelo apoio, carinho, suporte e encorajamento.

Ao meu orientador, Professor Carlos José Pereira de Lucena, pela oportunidade oferecida para desenvolver esta pesquisa, por incentivar idéias e apresentar momentos e oportunidades desafiadores. Sua orientação e apoio foram fundamentais durante esta jornada.

Ao meu co-orientador, Professor Arndt von Staa, pelo apoio e tempo despendido ao longo de meu trabalho. Sua orientação, atenção, comentários construtivos, revisões sucessivas e apoio foram fundamentais para o desenvolvimento deste trabalho. Suas observações sempre contribuíram para um entendimento melhor do problema e a identificação de soluções alternativas.

Aos meus colegas do LES pelas valiosas contribuições.

Em especial, não podia deixar de mencionar Ricardo Choren. Sua amizade, experiência, constantes revisões e atuação presente foram fundamentais para a elaboração deste trabalho.

Ao meu amigo, Leandro Daflon, grande companheiro em jornadas de trabalho e na caminhada feita ao longo do mestrado.

Aos Professores Renato Cerqueira, Noemi Rodriguez e Hugo Fuks por participarem da Comissão Examinadora. A todos os professores e funcionários do Departamento pelos ensinamentos e pela ajuda.

À CAPES, ao CNPq e à PUC-Rio, pelos auxílios concedidos, sem os quais este trabalho não poderia ter sido realizado.

## Resumo

Carvalho, Gustavo Robichez de; Lucena, Carlos José Pereira de; Staa, Arndt von. **Uma Arquitetura para a Coordenação e a Composição de Artefatos de Software**. Rio de Janeiro, 2003. 101p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

A engenharia de software baseada em componentes é uma abordagem que prevê a reutilização de artefatos de software na geração de um conjunto de aplicações. Ao desenvolver aplicações com esta abordagem, é preciso reunir ou compor componentes de software já existentes. Após compor estas unidades, é necessário coordenar as interdependências estabelecidas entre elas para adequar a aplicação em desenvolvimento à resolução do problema.

Esta dissertação propõe uma arquitetura de software que separa e estrutura os conceitos de coordenação, composição e componentes de software em camadas arquiteturais. A partir desta estrutura, espera-se que modificações específicas em construções de uma camada tenham o mínimo de influência sobre as demais. ACCA (Arquitetura para a Coordenação e a Composição de Artefatos de Software) deve ser entendida como uma estrutura conceitual utilizada para organizar o desenvolvimento de software baseado em componentes. Além disto, são apresentados um *framework* para ilustrar a realização da camada de composição de ACCA, o processo de reificação de ACCA e um processo de desenvolvimento de software utilizando a abordagem proposta.

## Palavras-chave

Arquitetura de Software, Componentes de Software, Coordenação de Componentes, Composição de Componentes, Evolução de Software, Reutilização de Software.

## Abstract

Carvalho, Gustavo Robichez de; Lucena, Carlos José Pereira de; Staa, Arndt von. **Architecture for Coordination and Composition of Software Artifacts**. Rio de Janeiro, 2003. 101p. Master Dissertation – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Component Based Software Engineering is an approach for reusing software artifacts when developing applications. In order to develop solutions using this approach, it is necessary to compose software using components that have already been developed. After putting those pieces together, we need to coordinate the interdependencies established among those compositions to fulfill the requirements, needed to resolve a problem.

This dissertation proposes a software architecture that separates and structures the concepts of coordination, composition and software components in different architectural layers. Using this approach, we expect that specific modifications in layer constructions have the minimum impact on the others layers. ACCA (Architecture for Coordination and Composition of software Artifacts) must be understood as a conceptual structure that is used to organize component based software development. It also presents a composition framework, the reification process for ACCA and a software development process organized using this approach.

## Keywords

Software Architecture, Software Components, Component Coordination, Component Composition, Software Evolution, Software Reuse.

# Sumário

1	Introdução	1
1.1.	Abordagem	1
1.2.	Motivação	2
1.3.	Trabalhos Relacionados	4
1.4.	Guia do Leitor	7
2	Conceitos Básicos	9
2.1.	Componentes de Software	9
2.2.	Composição e Coordenação de Componentes de Software	10
2.3.	Arquitetura de Software	21
3	Arquitetura para a Coordenação e a Composição de Artefatos de Software	23
3.1.	A Abordagem Proposta	23
3.2.	Abstração para a Camada de Artefatos de Software de ACCA	25
3.3.	Abstração para a Camada de Composição de ACCA	26
3.4.	Abstração para a Camada de Coordenação de ACCA	28
3.5.	Avaliação Conceitual de ACCA	29
4	Realização de ACCA	31
4.1.	O Problema Exemplo	31
4.2.	Contratos de Coordenação	33
4.3.	Um <i>Framework</i> para a Composição de Artefatos de Software	39
4.4.	Apoio ao Armazenamento e Catalogação de Artefatos de Software	50
5	Processo de Reificação e de Desenvolvimento com ACCA	53
5.1.	Processos de Desenvolvimento de Software	53
5.2.	Processo de Reificação de ACCA	54
5.3.	Processo de Desenvolvimento a partir de Componentes de Software	57

6 Estudos - Provas de Conceito	61
6.1. Sistema de Busca de Publicações Acadêmicas	61
6.2. Implementação de Padrões de Concorrência	75
7 Conclusões e Trabalhos Futuros	81
7.1. Lista de Contribuições	82
7.2. Trabalhos Futuros	82
8 Referências	84

## Lista de Figuras

Figura 1 – Níveis de abstração: solução x componente	3
Figura 2 – Ilustração de composição funcional	13
Figura 3 – Ilustração de composição orientada a dados	14
Figura 4 – Ilustração de composição por meio de extensões	14
Figura 5 – Ilustração de composição orientada a serviço ou orientada a conexão	15
Figura 6 – Ilustração de composição orientada a contexto	16
Figura 7 – Ilustração dos padrões de composição apresentados	16
Figura 8 – Padrão de coordenação proposto por Andrade et al. (2000)	19
Figura 9 – Padrão de coordenação proposto por Cruz e Tichelaar (1998)	19
Figura 10 – Exemplo de padrão de coordenação	20
Figura 11 – Padrão arquitetural <i>Layer</i> (Buschman, 1996)	22
Figura 12 – Arquitetura / Coordenação / Composição / Artefato de Software (ACCA)	23
Figura 13 – Camada de artefatos de software de ACCA: uma abstração	25
Figura 14 – Camada de composição de ACCA : uma abstração	26
Figura 15 – Estímulo a uma composição : fluxo de mensagens	27
Figura 16 – Invocação de um serviço: fluxo de mensagens	27
Figura 17 – Padrão de coordenação: interdependências entre composições	28
Figura 18 – Camada de coordenação de ACCA: uma abstração	28
Figura 19 – Ilustração de cenário de serviços de marcação de compromissos	32
Figura 20 – Regras para o cenário de uso de marcação de compromissos	33
Figura 21 – Modelo de coordenação utilizado (Lano et al., 2002)	34
Figura 22 – Contrato de coordenação: sintaxe	34
Figura 23 – Realização da camada de coordenação	35
Figura 24 – Regras impostas para compromissos entre João, Maria, José e Marcus	36

Figura 25 – Restrições 1.1 e 1.2 de Maria	37
Figura 26 – Contrato com as regras de Maria para marcar compromissos com João	37
Figura 27 – Restrições 1.3.1 e 1.3.3 de Maria e José	37
Figura 28 – Contrato com as regras de José para marcar compromissos com Maria	38
Figura 29 – Restrições 1.3.2.1 e 1.3.2.2 de Marcus e José	38
Figura 30 – Dependências para compromissos entre Vera e João	38
Figura 31 – Contrato com as regras de Vera para marcar compromissos com João	39
Figura 32 – Dependências para a marcação de compromisso entre Deborah e João	39
Figura 33 – Contratos com as regras de Deborah para marcar compromissos com João	39
Figura 34 – Pontos de flexibilização do <i>framework</i> de composição	40
Figura 35 – Visão geral do <i>framework</i>	41
Figura 36 – <i>Framework</i> foco na classe <i>Context</i>	42
Figura 37 – A seqüência para tratamento de um estímulo a um contexto	43
Figura 38 – <i>Framework</i> foco na classe <i>Handler</i>	43
Figura 39 – <i>Framework</i> foco nas classes <i>Client</i> e <i>ClientPolicy</i>	44
Figura 40 – <i>Framework</i> foco na classe <i>Compositor</i>	45
Figura 41 – <i>Framework</i> foco na classe <i>CommunicationBus</i>	46
Figura 42 – <i>Framework</i> foco na classe <i>CompositionFactory</i>	46
Figura 43 – Ilustração de relacionamento entre as composições e os serviços	48
Figura 44 – Instância do <i>framework</i> : comunicação, transporte e <i>Context-Handler</i>	49
Figura 45 – Componentes de software disponíveis para a solução do problema	51
Figura 46 – Ilustração da realização conceitual da camada de artefatos de software	52
Figura 47 – Papéis no processo de desenvolvimento associado a ACCA	53
Figura 48 – Ciclo de vida de ACCA	54

Figura 49 – Visão geral do processo de desenvolvimento de software	57
Figura 50 – Seqüência principal de atividades do processo de desenvolvimento	58
Figura 51 – Fase de reestruturação evolutiva da solução	59
Figura 52 – Fase de adaptação da solução	59
Figura 53 – Métodos presentes em uma composição	62
Figura 54 – Seqüência de utilização de instância de <i>Compositor</i>	62
Figura 55 – Serviços de busca de publicações acadêmicas	64
Figura 56 – Ilustração conceitual da realização das camadas de ACCA	65
Figura 57 – Modelo de coordenação para o redirecionar chamadas (I/II)	67
Figura 58 – Modelo de coordenação para o redirecionar chamadas (II/II)	67
Figura 59 – Exemplo de interceptação e redirecionamento de chamada a serviços	67
Figura 60 – Modelo de coordenação de encadeamento de serviços	68
Figura 61 – Encadeamento de serviços de busca	68
Figura 62 – Modelo de coordenação de atualização de <i>cache</i> de informações	68
Figura 63 – Atualização de <i>cache</i> de serviço de busca	69
Figura 64 – Modelo de coordenação de acesso ao <i>cache</i>	69
Figura 65 – Acesso ao <i>cache</i> : obtenção de informações	70
Figura 66 – Modelo de coordenação de seqüência de serviços	71
Figura 67 – Estabelecimento de prioridade para a obtenção das informações	71
Figura 68 – Ilustração de relacionamento entre as composições e os serviços	71
Figura 69 – Instância do <i>framework</i> : transporte, comunicação e <i>Context-Handler</i>	72
Figura 70 – Cenário de composição de tratadores para a utilização de serviço de <i>cache</i>	72
Figura 71 – Exemplo de diagrama de seqüência de tratador para serviço de <i>cache</i>	73
Figura 72 – Ilustração da resolução de padrões de concorrência com ACCA	76

Figura 73 – Interações do padrão <i>Scheduler</i> e modelo de coordenação correspondente	76
Figura 74 – Componente Processador implementado de forma <i>ad hoc</i>	77
Figura 75 – Conceitos de coordenação do padrão <i>Scheduler</i>	77
Figura 76 – Artefato de software correspondente ao componente Processador	77
Figura 77 – Interações do padrão <i>Read/Write Lock</i>	78
Figura 78 – Modelo de coordenação do padrão <i>Read/Write Lock</i>	78
Figura 79 – Componente Lance implementado de forma <i>ad hoc</i>	79
Figura 80 – Conceitos de coordenação do padrão <i>Read/Write Lock</i>	79
Figura 81 – Artefato de software corresponde ao componente Lance	79

## Lista de Abreviaturas

ACCA – Arquitetura para a Coordenação e a Composição de Artefatos de Software

CORBA – Common Object Request Broker Architecture

CVS – *Concurrent Versions System*

EJB – *Enterprise JavaBeans*

HTTP – *Hypertext Transfer Protocol*

IBM – *International Business Machine*

LES – Laboratório de Engenharia de Software / PUC-Rio

MDA – *Model Driven Architecture*

PC – *Personal Computer*

PDA – *Personal Digital Assistant*

POA – *Programação Orientada a Aspectos*

SMTP – *Simple Mail Transfer Protocol*

SOAP – *Simple Object Access Protocol*

UDDI – *Universal Description, Discovery and Integration*

WAS – *Websphere Application Server*

XML – *Extensible Markup Language*