Alessandro Fabricio Garcia

Objetos e Agentes: Uma Abordagem Orientada a Aspectos

TESE DE DOUTORADO

DEPARTAMENTO DE INFORMÁTICA

Programa de Pós-Graduação em Informática

Rio de Janeiro

Abril de 2004

Pontifícia Universidade Católica
do Rio de Janeiro

Alessandro Fabricio Garcia

Objetos e Agentes:
Uma Abordagem Orientada a Aspectos

Tese de Doutorado

Tese apresentada como requisito parcial para obtenção do título de Doutor pelo Programa de Pós-Graduação em Informática da PUC-Rio.

Orientador: Carlos José Pereira de Lucena

Departamento de Informática, PUC-Rio, Abril de 2004

Pontifícia Universidade Católica
do Rio de Janeiro

Alessandro Fabricio Garcia

Objetos e Agentes: Uma Abordagem Orientada a Aspectos

Thesis presented to the Graduate Program in Computer Science of the Pontifical Catholic University of Rio de Janeiro in partial fulfillment of the requirements for the degree of Doctor of Science.

Prof. Carlos José Pereira de Lucena
Advisor
Departamento de Informática – PUC-Rio

Prof. Brian Henderson-Sellers
Department of Software Engineering - University of Technology at Sydney

Profa. Claudia Maria Lima Werner
COPPE – UFRJ

Prof. Arndt von Staa
Departamento de Informática – PUC-Rio

Profa. Simone Barbosa
Departamento de Informática – PUC-Rio

Prof. José Eugenio Leal
Coordenador Setorial do Centro
Técnico Científico – PUC-Rio

Rio de Janeiro, April 2nd, 2004

Alessandro Fabricio Garcia

Graduou-se em Ciência da Computação na Universidade Estadual de Maringá (UEM) em 1998. Obteve o título de Mestre em Ciência da Computação na Universidade Estadual de Campinas (Unicamp) em 2000. É pesquisador associado ao Laboratório de Engenharia de Software (LES) da PUC-Rio.

Ficha Catalográfica

Garcia, Alessandro Fabricio

Objetos e agentes : uma abordagem orientada a aspectos / Alessandro Fabricio Garcia ; orientador: Carlos José Pereira de Lucena. – Rio de Janeiro : PUC, Departamento de Informática, 2004.

298 f. : il. ; 30 cm

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

Inclui referências bibliográficas.

1. Informática – Teses. 2. Sistemas multi-agentes. 3. Agentes de software. 4. Aspectos. 5. Padrões de projeto. 6. Arquitetura de software. 7. Métricas. 8. Engenharia de software experimental. I. Lucena, Carlos José Pereira de. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

A meus pais Antonio e Cida,
e minha irmã Andresa

Agradecimentos

Em primeiro lugar, gostaria de agradecer a meu orientador Carlos Lucena pois, sem o seu apoio e amizade, este trabalho não teria sido possível. Ele me permitiu ter liberdade e proporcionou meios para amadurecer minhas idéias, indicando-me sempre o direcionamento correto. Seu profundo conhecimento de ciência, engenharia e da natureza humana o tornam um cientista único que, sem dúvidas, será um padrão que sempre procurarei seguir. Sua busca incessante por similaridades mostrou-me que uma série de boas idéias surge simplesmente da conexão de peças de trabalho não conectáveis à primeira vista. Muitas foram suas contribuições em minha pesquisa. Em particular, arranjou todos os detalhes de minha colaboração com o grupo CSG da Universidade de Waterloo e com o Centro de Pesquisa da IBM em Almaden/EUA. Ao Professor Lucena, minha profunda gratidão.

A maior parte da minha pesquisa foi desenvolvida no contexto do Laboratório de Engenharia de Software (LES) da PUC-Rio. Gostaria de agradecer a todos meus colegas e professores pelo ambiente de trabalho estimulante oferecido. Em particular, gostaria de agradecer aos professores Arndt von Staa e Julio Leite os ensinamentos e discussões que influenciaram enormemente meu trabalho. Arndt, obrigado pelas discussões e incentivo durante minhas andanças pelos estudos empíricos e métricas de software. Arndt sempre ofereceu sugestões úteis e críticas importantes para o progresso da minha pesquisa, contribuindo de forma decisiva para a qualidade deste trabalho. Com o Julio, participei de duas das disciplinas mais interessantes em Engenharia de Software, uma sobre evolução de software e outra sobre engenharia de software experimental. Ambas impactaram de maneira significativa minha visão em engenharia de software e influenciaram vastamente o conteúdo desta tese.

Ilimitados agradecimentos vão para todos meus companheiros de pesquisa no LES. Eles forneceram feedback contínuo durante o desenvolvimento e redação desta tese. Agradeço especialmente a meus amigos Christina Chavez, Cláudio Sant'Anna, Otávio Rezende e Uirá Kulesza, com os quais desenvolvi pesquisas e escrevi vários artigos em

conjunto. Vocês contribuíram decisivamente para o meu trabalho, sempre fornecendo comentários e sugestões fundamentais para a melhoria da qualidade da minha tese. Nossas discussões sempre foram úteis e inspiradoras. Também não poderia deixar de mencionar e agradecer Viviane Silva e Anarosa Brandão. Juntos trabalhamos mais de um ano na primeira versão do TAO, o framework conceitual posteriormente estendido e apresentado nesta tese. Obrigado Viviane e Anarosa por todos os momentos que passamos juntos e pelas experiências compartilhadas.

Este trabalho de pesquisa também foi conduzido em conjunto com o grupo CSG de Waterloo. A definição dos principais problemas tratados neste trabalho foi inicialmente elaborada no contexto do CSG. As diversas discussões com os colegas do grupo foram muito importantes para o refinamento do trabalho. Sou grato especialmente aos professores Donald Cowan e Paulo Alencar pelo suporte oferecido em relação a todas as questões da minha estadia em Waterloo e cooperação com o CSG. Além de Waterloo, também passei três meses trabalhando no projeto TSpaces no Centro de Pesquisa da IBM em Almaden. Gostaria de agradecer ao amigo Marcus Fontoura e aos meus gerentes Thomas Truong e Toby Lehman a oportunidade e todo o suporte prestado.

Agradeço também a toda comunidade envolvida no projeto SELMAS e aos pesquisadores e colegas que forneceram importante feedback para o amadurecimento deste trabalho, em especial Adenilso Simão, Alexander Romanovsky, Analia Amandi, Anand Tripathi, Andrea Omicini, Ana Perini, Awais Rashid, Barbara Kitchenham, Brian Henderson-Sellers, Dan Berry, Don Cowan, Eric Ernst, Franco Zambonelli, Gail Murphy, Gary Leavens, Jeff Gray, José Sardinha, Liz Kendall, Jan Hanneman, Lodewijk Bergmans, Marco Mammei, Paulo Alencar, Ruy Milidiú, Toacy Oliveira, Tom Holvoet, Tom Maibaum e Torsten Nelson.

A burocracia pode tornar a vida dos estudantes de doutorado muito difícil devido aos inúmeros contratempos e questões administrativas a serem resolvidos. Entretanto, não tive maiores problemas com essas questões graças ao esforço e a dedicação incansável de Vera Menezes. Sua postura amigável e seu senso de humor facilitaram enormemente

a resolução desses problemas. Ela sempre respondia prontamente a todas as minhas freqüentes requisições para resolver questões de reuniões, viagens e muitas outras. Agradeço a ela todo seu esforço e, mais importante ainda, sua amizade especial.

Também agradeço as agências de fomento pelo apoio financeiro. Este trabalho foi financiado parcialmente pelo CNPq por meio do processo No. 141457/2000-7 e pela FAPERJ pelo processo No. E-26/150.699/2002. Minha pesquisa também recebeu suporte do projeto PRONEX, processo 7697102900, e do projeto ESSMA, processo 552068/2002-0.

Finalmente, ilimitados agradecimentos vão para as três pessoas que julgo mais responsáveis por tudo que alcancei até aqui: meus preciosos pais Antonio e Cida e minha irmã Andresa. O apoio constante da minha família, mesmo a muitos quilômetros de distância, foi essencial para encontrar forças e continuar a lutar pelos meus objetivos. Palavras não podem expressar a imensidão da gratidão que tenho por eles.

Resumo

Garcia, Alessandro. **Objetos e Agentes: Uma Abordagem Orientada a Aspectos**. Departamento de Informática, 2004. 298p. Tese de Doutorado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Agentes de software incorporam várias propriedades específicas, como autonomia, adaptação, interação, aprendizagem e mobilidade. A inclusão dessas propriedades de agência é uma das maiores fontes de complexidade na construção de sistemas multiagentes. Dificilmente elas são modularizadas com abstrações e mecanismos da engenharia de software orientada a objetos. À medida que a complexidade da arquitetura interna dos agentes aumenta, essas propriedades tendem a se espalhar através dos vários módulos ou objetos do sistema. O espalhamento é observado desde fases preliminares de desenvolvimento, como a fase de definição arquitetural. O uso de abstrações e mecanismos existentes conduz ao projeto e à implementação de sistemas multiagentes que são difíceis de manter e reutilizar.

Este trabalho apresenta uma abordagem orientada a aspectos para o desenvolvimento de sistemas baseados em agentes. A abordagem provê suporte para modularização e composição das propriedades de agência por meio de abstrações e mecanismos do paradigma orientado a aspectos. Além disso, tais propriedades são incorporadas de forma transparente à funcionalidade básica do sistema de software, desde a fase de definição arquitetural. A abordagem compreende três componentes: (i) um método arquitetural, (ii) uma linguagem de padrões e (iii) um framework para avaliação quantitativa. O método e a linguagem apresentam um conjunto de soluções orientadas a aspectos para a definição arquitetural, projeto e implementação de agentes de software. O framework define um conjunto de métricas e um modelo de qualidade que permite a avaliação empírica da nossa abordagem em termos de reusabilidade e manutenibilidade. Estudos experimentais qualitativos e quantitativos foram realizados para avaliar nossa proposta em diferentes domínios de aplicação. Os resultados empíricos concluíram que nossa abordagem permite a construção de sistemas baseados em agentes com modularização superior, menor acomplamento, menos linhas de código e menor complexidade interna dos componentes.

Palavras-chave

Sistemas multiagentes, agentes de software, aspectos, padrões de projeto, arquitetura de software, engenharia de software experimental, métricas.

Abstract

Garcia, Alessandro. **From Objects to Agents: An Aspect-Oriented Approach.** Computer Science Department, 2004. 298p. Doctoral Thesis - Computer Science Department, Pontifical Catholic University of Rio de Janeiro.

Software engineers of Multi-Agent Systems (MASs) are faced with different concerns (properties), such as autonomy, adaptation, interaction, collaboration, learning, and mobility. Many of these agent concerns cannot be modularized based only on object-oriented abstractions. MAS developers however have relied mostly on object-oriented design techniques and on object-oriented programming languages, such as Java. As the agent complexity increases, the agent concerns tend to spread across several system components at the architectural, design and implementation levels. It often leads to a poor separation of agent concerns in the software system, and in turn to the production of MASs that are difficult to maintain and reuse.

This thesis presents an innovative aspect-oriented approach for the seamless integration of agents into object-oriented software engineering from the architectural stage to the implementation stage. Aspect is the abstraction used to modularize agent concerns that crosscut several system components. The proposed approach encourages the separate handling of agent properties, and provides a disciplined scheme for their composition. The approach is composed of an architectural method, a pattern language, and an assessment framework. The architectural method and the pattern language provide aspect-oriented solutions for modularizing the agent concerns at different stages of design and implementation. The purpose of the assessment framework is to support the evaluation of the reusability and maintainability of aspect-oriented solutions based on a metrics suite and a quality model. Experimental studies in different application domains have been conducted to assess the proposed approach based on qualitative and quantitative criteria. The use of the aspect-oriented solutions resulted in fewer lines of code, fewer design and implementation components, lower internal complexity of system components, and lower coupling.

Keywords

Multi-agent systems, software agents, aspect-oriented software development, design patterns, software architecture, empirical software engineering, software metrics.

Índice

1 Introdução 1

1.1 O problema 2

1.2 Limitações de trabalhos relacionados 4

1.3 A solução proposta 5

1.4 Avaliação empírica 9

1.5 Descrição da tese 10

2 Separação de concerns 11

2.1 Padrões 13

2.1.1 Padrões arquiteturais e arquiteturas de software 13

2.1.2 Padrões de projeto 14

2.1.3 Linguagens de padrões 16

2.2 Separação avançada de concerns 17

2.2.1 Crosscutting Concerns 17

2.2.2 Aspectos e técnicas de programação 19

Aspectos 20

Join Points e Pointcuts 21

Advices e Inter-Type Declarations 21

Um exemplo de aspecto 22

Processo de combinação e AspectJ 25

2.2.3 Desafios no desenvolvimento orientado a aspectos 25

2.3 Resumo 29

3 Agentes e objetos 31

3.1 TAO: o framework conceitual 33

3.1.1 Estrutura 34

3.1.2 Categorias de concerns 35

3.2 Concerns fundamentais 35

3.2.1 Agentes, objetos e ambientes 35
3.2.2 Eventos 37
3.2.3 Tipos de agente 38
3.3 Concerns de agência 40
3.3.1 Conhecimento 41
3.3.2 Interação 43
3.3.3 Adaptação 44
3.3.4 Autonomia 45
3.4 Concerns adicionais 47
3.4.1 Colaboração 47
3.4.2 Papéis 48
3.4.3 Aprendizagem 50
3.4.4 Mobilidade 51
3.5 Relacionamentos entre concerns do agente 52
3.6 Soluções orientadas a objetos: pontos fortes e pontos fracos 53
3.6.1 Agentes vistos como objetos 54
3.6.2 Explosão de classes e replicação de código 55
3.6.3 Esquizofrenia de agente 56
3.6.4 Crosscutting concerns do agente 57
3.7 Arquiteturas de agente e suas limitações 58
3.7.1 Arquiteturas em camadas 59
3.7.2 Arquiteturas reflexivas 62
3.7.3 Arquiteturas baseadas em mediadores 65
3.8 Resumo 65

4 Arquiteturas de agente: um método orientado a aspectos 67
4.1 Agentes no Portalware: um estudo de caso 68
4.2 Uso de uma arquitetura de agente baseada em mediadores 71
4.3 O método arquitetural orientado a aspectos 75
4.3.1 O propósito 75
4.3.2 Arquiteturas de agente orientadas a aspectos 75

Interfaces dos componentes 77
Propriedades arquiteturais 78
4.4 As diretrizes 79
4.4.1 Etapa 1: definição do kernel do agente 81
4.4.2 Etapa 2: definição das propriedades de agência 82
4.4.3 Etapa 3: definição de tipos de agente 84
4.4.4 Etapa 4: definição das propriedades adicionais 84
4.4.5 Etapa 5: definição dos papéis do agente 86
4.4.6 Etapa 6: composição do aspecto 87
4.4.7 Etapa 7: evolução de agente 90
4.5 Questões de implementação 91
4.6 Discussão e trabalhos relacionados 93
4.7 Resumo 95

5 A linguagem de padrões 97
5.1 Expert Committee: um estudo de caso 100
5.2 A linguagem de padrões: uma visão geral 102
5.2.1 O propósito 103
5.2.2 Por que são padrões de projeto? 103
5.2.3 A estrutura da linguagem de padrões 104
5.3 O padrão Kernel do Agente 106
5.4 O padrão Interação 111
5.5 O padrão Autonomia 125
5.6 O padrão Adaptação 135
5.7 O padrão Papel 145
5.8 O padrão Mobilidade 158
5.9 O padrão Aprendizagem 170
5.10 Questões de implementação e implantação 181
5.11 Discussão e trabalhos relacionados 182
5.11.1 Vantagens e desvantagens 183
5.11.2 Lições aprendidas 184

Exposição de Join Points 184
Gerenciamento da complexidade do aspecto 185
Aspectos como “Conectores” entre hierarquias de classes 185
Geração de código 186
5.12 Resumo 186

6 O framework de avaliação quantitativa 187
6.1 Aspectos: requisitos de medida 189
6.2 A estrutura do framework 191
6.3 O modelo de qualidade 191
6.3.1 Qualidades e fatores 193
6.3.2 Atributos Internos 195
6.4 O conjunto de métricas 196
6.4.1 Métricas de separação de concerns 197
Difusão de Concerns em Componentes (CDC) 198
Difusão de Concerns em Operações (CDO) 198
Difusão de Concerns em LOC (CDLOC) 198
6.4.2 Métricas de acoplamento 201
Acoplamento entre Componentes (CBC) 201
Profundidade de Árvore de Herança (DIT) 202
6.4.3 Métrica de coesão 202
Falta de Coesão nas Operações (LCOO) 202
6.4.4 Métricas de tamanho 203
Tamanho do vocabulário (VS) 203
Linhas de Código (LOC) 203
Número de Atributos (NOA) 204
Operações Consideradas por Componente (WOC) 204
6.5 Avaliação empírica 205
6.6 Discussão e trabalhos relacionados 206
6.7 Resumo 208

7 Estudos de Caso **209**
7.1 Arquitetura do simulador de tráfego 212
7.2 O estudo comparativo 215
7.2.1 A abordagem orientada a padrões 216
Concerns de agência 217
Concerns adicionais 217
Papéis 218
Composição dos concerns do agente 220
Evolução de agente 220
7.2.2 A comparação qualitativa 221
7.3 Mapeamento de uma implementação de AspectJ I para uma implementação de Hyper/J 224
7.3.1 De aspectos a hyperslices 225
7.3.2 O processo de transformação 226
7.4 Expert Committee 227
7.5 Trabalhos relacionados 228
7.6 Resumo 229

8 O estudo quantitativo **231**
8.1 As abordagens investigadas 232
8.2 A metodologia 233
8.3 Objetivos e questões 233
8.4 Hipótese 235
8.5 O projeto do SMA 235
8.6 Os sujeitos e as fases do estudo 237
8.7 Avaliação das abordagens 239
8.7.1 A fase de construção do SMA 239
Resultados da métrica de tamanho 240
Resultados da métrica de coesão e acoplamento 242
Resultados da métrica de separação de concerns 244

8.7.2 A fase de evolução e reutilização do SMA 245
8.8 Discussão 247
8.8.1 Comparação das abordagens 248
8.8.2 Agentes *versus* objetos 249
8.8.3 Avaliação do framework de avaliação 250
8.8.4 Ameaças à validade 251
8.9 Comparação com trabalhos relacionados 252
8.10 Resumo 253

9 Conclusão e trabalhos futuros 255
9.1 Contribuições 256
9.2 Trabalhos em andamento e trabalhos futuros 258

Apêndice I Implementação da linguagem de padrões: Problemas e Código de amostra em AspectJ 283
O padrão Kernel do Agente 283
O padrão Interação 287
O padrão Adaptação 294
O padrão Autonomia 299
O padrão Papel 304
O padrão Aprendizagem 306
O padrão Mobilidade 311

Apêndice II Regras de transformação: De AspectJ a Hyper/J 315
Hyper/J 315
Regras de transformação 318
Descobertas 319

Lista de Figuras

Figura 1. Entrelaçamento de códigos em aplicações orientadas a agentes.....3
Figura 2. A abordagem orientada a aspectos: uma visão geral. 6
Figura 3. Tratamento de erros: um exemplo de Crosscutting Concern. 19
Figura 4. O projeto do aspecto FaultHandler. 23
Figura 5. A dinâmica da classe Server e o aspecto FaultHandler. 23
Figura 6. Exemplo de AspectJ. 24
Figura 7. Ambiente, objetos e agentes.....37
Figura 8. Eventos. 38
Figura 9. Tipos de agente: classificação baseada em serviços de agente. 39
Figura 10. Classificação baseada em capacidades cognitivas. 40
Figura 11. Concerns de agência.....41
Figura 12. Concern de conhecimento. 42
Figura 13. Concern de interação.....44
Figura 14. Concern de adaptação. 45
Figura 15. Concern de autonomia. 46
Figura 16. Concerns de papel e colaboração. 48
Figura 17. Concern de aprendizagem. 51
Figura 18. Concern de mobilidade. 52
Figura 19. Relacionamentos entre concerns do agente. 53
Figura 20. Uso da delegação para a composição de concerns do agente. 56
Figura 21. O padrão arquitetural de agentes em camadas [137]. 60
Figura 22. Adaptação na abordagem de agentes em camadas: comportamento crosscutting. 61
Figura 23. Estruturação baseada em padrões das camadas de adaptação e conhecimento. 62
Figura 24. Agentes de brainstorm [8]. 63
Figura 25. Agentes no Portalware. 70

Figura 26. Um subconjunto do projeto orientado a objetos do SMA do Portalware. 72
Figura 27. Código Java do Portalware. 73
Figura 28. A arquitetura orientada a aspectos dos agentes de informação.....77
Figura 29. O método arquitetural orientado a aspectos para o desenvolvimento de SMAs. 80
Figura 30. Kernel do agente. 81
Figura 31. Os aspectos de agência da arquitetura de agente. 83
Figura 32. Tipos de agente. 85
Figura 33. Outros aspectos do agente.....86
Figura 34. Os aspectos de papéis de agentes de informação.....87
Figura 35. Um diagrama de interação para os agentes de informação do Portalware. 89
Figura 36. Introdução do aspecto Mobility a agentes de informação.....91
Figura 37. A arquitetura orientada a aspectos dos agentes do EC.....102
Figura 38. A estrutura da linguagem de padrões. 104
Figura 39. Vários padrões crosscutting. 106
Figura 40. A visão estática do padrão Kernel do Agente. 108
Figura 41. Definição dos tipos de agente. 109
Figura 42. O projeto orientado a objetos do concern de interação. 112
Figura 43. Comportamento detecção nos agentes do EC. 113
Figura 44. A visão dinâmica do padrão Adaptador. 114
Figura 45. A visão estática do padrão Interação.....116
Figura 46. O padrão de interação para o agente de usuário do EC. 119
Figura 47a. Padrão Interação: recebendo uma mensagem. 121
Figura 47b. Padrão Interação: enviando uma mensagem.....121
Figura 48. Padrão interação: detectando um evento externo.....122
Figura 49. A visão estática do padrão Autonomia. 127
Figura 50. O padrão Autonomia do papel de Revisor. 130
Figura 51. Padrão Autonomia: tornando um agente autônomo.....131
Figura 52. Padrão Autonomia: Criando um objetivo proativo. 132

Figura 53. Comportamento adaptação nos agentes do EC. O padrão Observador [137]. 136
Figura 54. A visão estática do padrão Adaptação. 138
Figura 55. O padrão Adaptação do papel de revisor. 141
Figura 56. Padrão Adaptação: adaptando Agent no recebimento de uma mensagem. 142
Figura 57. Padrão Adaptação: adaptando planos ao definir um novo objetivo.....143
Figura 58. Padrão Adaptação: adaptando planos quando surge uma exceção. 143
Figura 59. Papéis: o padrão Objeto do Papel com o padrão Decorador [146]. 147
Figura 60. A visão estática do padrão Papel.....151
Figura 61. O padrão Papel para o agente do usuário do EC.....152
Figura 62. Padrão Papel: ligando uma instância do papel a uma instância do agente. 153
Figura 63. Padrão Papel: ativando um papel quando a informação não é encontrada. 154
Figura 64. Concern de mobilidade: afetando papéis, planos e tipos de agente. 160
Figura 65. A visão estática do padrão Mobilidade. 163
Figura 66. O padrão Mobilidade do papel de Chair. 164
Figura 67. Padrão Mobilidade: movendo quando um plano não consegue encontrar informações. 166
Figura 68. Padrão Mobilidade: movendo quando uma ação do papel não consegue encontrar informações. 167
Figura 70. Aprendizagem: o padrão Observador com o padrão Estratégia.....171
Figura 71. A visão estática do padrão Aprendizagem. 174
Figura 72. O padrão Aprendizagem para o agente de usuário do EC. 176
Figura 73. A visão dinâmica do aspecto ReviewerLearning. 177
Figura 74. A visão dinâmica do aspecto ChairLearning. 178
Figura 76. Dimensões de acoplamento em um DSOA. 190
Figura 77. O framework de avaliação. 192
Figura 78. O modelo de qualidade. 193
Figura 79. Um exemplo de sombreamento de código. 199

Figura 80. A evolução dos estudos de caso.....210
Figura 81. Usando o padrão Mediador para desenvolver concerns de agência.....218
Figura 82. Usando o padrão Objeto do Papel para desenvolver papéis do agente...219
Figura 83. O modelo de projeto para aspectos de agência de agentes do Portalware. 227
Figura 85. Um subconjunto do projeto orientado a padrões do SMA.....236
Figura 86. Um subconjunto do projeto orientado a aspectos do SMA.....238
Figura 87. Uso do Together [25] para aplicar as métricas definidas.....240
Figura 88. Comparação dos resultados dos dois projetos.....241

Lista de Tabelas

Tabela 1. O modelo de uma descrição de padrão 16
Tabela 2. Elementos do Diagrama de Características.....35
Tabela 3. Uma visão geral de concerns do agente.....36
Tabela 4. Características dos sistemas multiagentes 209
Tabela 5. Comparação de AspectJ e Hyper/J 225
Tabela 6. Métricas, questões GQM e atributos internos.....235
Tabela 7. Os resultados dos cenários de manutenção e reutilização 245

Lista de Acrônimos e Abreviações

ADG – Advice Dependence Graph

ADS – Ambiente de Desenvolvimento de Software

CBC – Coupling between Components

CDC – Concern Diffusion over Components

CDLOC – Concerns Diffusion over LOC

CDO – Concern Diffusion over Operations

CK – Chidamber & Kemerer

CS – Content Supplier

DIT – Depth of Inheritance Tree

DSOA – Desenvolvimento de Software Orientado a Aspectos

EC – Expert Committee

FIPA – Foundation for Intelligent Physical Agents

GoF – Gang of Four

GQM – Goal-Question-Metric

IDG – Introduction Dependence Graph

LCOM – Lack of Cohesion in Methods

LCOO – Lack of Cohesion in Operations

LMS – Least Means Square

LOC – Lines of Code

MDG – Method Dependence Graph

NOA – Number of Attributes

OA – Orientado a Aspecto

OMG – Object Management Group

OO – Orientado a Objetos

POA – Programação Orientada a Aspectos

SAM – Sistema Multiagentes

SoC – Separação de Concerns

TAO – Taming Agents and Objects

TD-Learning – Temporal Distance Learning

VS – Vocabulary Size

WOC – Weighted Operations per Component