INSTITUTO MILITAR DE ENGENHARIA

1° Ten QEM JANILMA AFFIFE RAFFID DE VILLARA PERES

INTEGRAÇÃO E COMPOSIÇÃO DE CARACTERÍSTICAS TRANSVERSAIS EM SISTEMAS MULTIAGENTES

Dissertação de Mestrado apresentada ao Curso de Mestrado em Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para obtenção do título de Mestre em Ciências em Sistemas e Computação.

Orientador: Prof. Ricardo Choren Noya - D. Sc.

Rio de Janeiro
2006

INSTITUTO MILITAR DE ENGENHARIA
Praça General Tibúrcio, 80-Praia Vermelha
Rio de Janeiro-RJ CEP 22290-270

P434i Peres, Janilma Affife Raffid de Villara
Integração e composição de características transversais em sistemas multiagentes / Janilma Affife Raffid de Villara Peres. - Rio de Janeiro: Instituto Militar de Engenharia, 2006.
117 p.: il.

Dissertação de mestrado do curso de Sistemas e Computação - Instituto Militar de Engenharia- Rio de Janeiro, 2006

1. Engenharia de Software 2. Sistemas Multiagentes. 3. Orientação a Aspectos. I. Peres, Janilma Affife Raffid de Villara II. Instituto Militar de Engenharia. III. Título.

CDD 005.1

INSTITUTO MILITAR DE ENGENHARIA

1° TEN QEM JANILMA AFFIFE RAFFID DE VILLARA PERES

INTEGRAÇÃO E COMPOSIÇÃO DE CARACTERÍSTICAS TRANSVERSAIS EM SISTEMAS MULTIAGENTES

Dissertação de Mestrado apresentada ao Curso de Mestrado em Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para obtenção do título de Mestre em Ciências em Sistemas e Computação.

Orientador: Prof. Ricardo Choren Noya - D. Sc..

Aprovada em 15 de dezembro de 2006 pela seguinte Banca Examinadora:

Prof. Ricardo Choren Noya - D. Sc. do IME - Presidente

Prof. Paulo Fernando Ferreira Rosa - Ph.D do IME

Prof. Carlos José Pereira de Lucena - Ph.D da PUC-Rio

Rio de Janeiro
2006

Ao Vitor.

AGRADECIMENTOS

Aos meus pais, pois sem eles eu nada seria. Meus pais nunca mediram esforços para proporcionar-me a melhor educação e sempre enfatizaram que o conhecimento é, de todos os bens, o mais precioso.

Ao meu marido Vitor, também conhecido como Ten Draeger, que está ao meu lado desde os saudosos tempos de Colégio Militar do Rio de Janeiro. Meu colega de turma no IME, cursou também Engenharia de Computação e seguiu junto comigo para o Curso de Mestrado. Além de seu amor e de sua dedicação como marido, sempre encontrei no Vitor um engenheiro brilhante para trocar idéias e, neste mestrado, não foi diferente.

Ao meu orientador Prof. Ricardo Choren Noya, que acreditou em meu potencial desde o início, incentivou-me bastante, orientou-me de forma muito objetiva e demonstrou grande comprometimento com o trabalho realizado. Trabalhamos este tempo todo em total sintonia, o que foi decisivo para o sucesso alcançado. Além de um profissional de grande capacidade técnica, o Prof. Choren é uma ótima pessoa, de modo que, termino este mestrado certa de ter feito um novo amigo.

Aos professores Alessandro Fabrício Garcia, da Lancaster University, e Christina von Flach Garcia Chavez, da Universidade Federal da Bahia, que me foram apresentados pelo Prof. Choren e tiveram papel importante na dissertação, contribuindo com idéias, questionamentos e escrevendo artigos junto comigo e o Prof. Choren. Essa troca não poderia ter sido mais produtiva, afinal são profissionais extremamente capazes e conceituados.

Ao TC Ulf Bergmann, pelos ensinamentos importantes ministrados no primeiro ano do mestrado e, sobretudo, pelo aconselhamento e pela confiança no meu trabalho. Por questões da carreira, em 2006, o TC Bergmann teve que deixar o IME para fazer o Curso de Chefia e Direção para Engenheiros Militares na ECEME e não pôde acompanhar de perto o meu segundo ano de mestrado. Levarei seu exemplo e seus ensinamentos profissionais para a vida toda.

Ao Prof. Carlos José Pereira de Lucena, da Pontifícia Universidade Católica do Rio de Janeiro, também grande incentivador do meu trabalho, figura de referência nacional e internacional no meio acadêmico e um dos grandes nomes da Computação brasileira. Travar contato com um profissional de tamanho gabarito é sempre uma experiência ímpar.

Aos professores e funcionários da Seção de Engenharia de Computação (SE/8), por sempre estarem dispostos a ajudar. Agradeço especialmente aos professores que me deram aulas na graduação e pós-graduação, afinal devo bastante a eles por terem me proporcio-

nado uma formação sólida.

Aos meus amigos e a todos que contribuíram direta ou indiretamente para dar-me condições de alcançar mais este sucesso, pois um grande feito está sempre amparado nos esforços de muitos.

"It is not the task of the University to offer what society asks for, but to give what society needs. The things society asks for are generally understood, and you don't need a University for that; the University has to offer what no one else can provide."

E. DIJKSTRA

SUMÁRIO

LISTA DE ILUSTRA ÕES

LISTA DE TABELAS

LISTA DE SÍMBOLOS E ABREVIATURAS

ABREVIATURAS

ACROSS	-	*Agent Crosscutting Concerns Modeling Framework* (Framework de Modelagem de Características Transversais)
AMC	-	Ao Meu Comando
AORE	-	*Aspect-Oriented Reuirements Engineering* (Engenharia de Requisitos Orientada a Aspectos)
BDI	-	*Belief-Desire-Intention* (Crença-Desejo-Intenção)
C2	-	Comando e Controle
C 6-40	-	Manual de Campanha de Técnica de Tiro de Artilharia (do Exército Brasileiro)
CP	-	Chefe de Peça
CLF	-	Comandante da Linha de Fogo
DF	-	*Directory Facilitator*
GAC	-	Grupo de Artilharia de Campanha
JADE	-	*Java Agent Development*
IC	-	Interface de Crosscutting
LMROA	-	Linguagem de Modelagem de Requisitos Orientada a Aspectos
MRO	-	Mensagem Resposta (ao Observador)
MT	-	Missão de Tiro
OA	-	Observador Avançado
O Lig	-	Oficial de Ligação
OO	-	Orientação a Objetos
POA	-	Programação Orientada a Aspectos
QP	-	Quando Pronto
S/3	-	Oficial de Operações (Escalão Unidade)

SMA	-	Sistema Multiagentes
SMAC	-	Sistema Multiagentes para Artilharia de Campanha
UML	-	*Unified Modeling Language*
XML	-	*eXtensible Markup Language*

RESUMO

O princípio da separação de características é um conceito bem estabelecido na Engenharia de Software, significando que cada característica do sistema deve ser tratada como uma unidade conceitual isolada. A Orientação a Aspectos surgiu como um paradigma para promover a modularização efetiva das características que estão espalhadas e entrelaçadas em um sistema, i.e., as chamadas caracterísiticas transversais.

Embora na sua origem seja um paradigma complementar à Orientação a Objetos, a Orientação a Aspectos foi estendida e passou a cobrir outras abordagens, como os Sistemas Multiagentes. Os Sistemas Multiagentes são compostos por entidades autônomas, os agentes de software, que executam ações para atingir seus objetivos. Os agentes possuem algumas propriedades internas, como a autonomia, interatividade, pró-atividade, adaptação, aprendizado, colaboração, mobilidade e ser situado.

Muitas características transversais podem estar presentes na modelagem dos Sistemas Multiagentes, sendo conseq ências de seus requisitos funcionais, não-funcionais e das propriedades internas dos agentes. No entanto, as linguagens e metodologias de modelagem para Sistemas Multiagentes existentes não oferecem suporte à modularização das características transversais.

Este trabalho objetiva propor um framework de modelagem que modularize características transversais de amplo escopo em Sistemas Multiagentes pela aplicação de conceitos modelagem orientada a metas. Este framework, chamado ACROSS, utiliza as abstrações convencionais do paradigma dos agentes de software, como agentes, objetivos de agentes, planos e ações e traz novas abstrações como aspectos e objetivos de aspecto para modularizar elementos que possuem impacto transversal na modelagem dos agentes. O framework também apresenta relacionamentos que definem, respectivamente, a hierarquia entre essas abstrações e como a transversalidade ocorre em cada nível. Uma notação baseada em XML para descrever os modelos gerados foi proposta.

Além disso, desenvolveu-se um estudo de caso de um sistema de Comando e Controle para a Artilharia de Campanha e sugeriu-se formas de mapear os modelos em código, considerando as tecnologias de implementação JADE, para Sistemas Multiagentes e AspectJ, para aspectos. Por fim, conclui-se sobre a aplicabilidade e limitações do ACROSS e propõe-se algumas orientações para pesquisa e desenvolvimentos futuros.

ABSTRACT

The concern separation principle is a well-established concept in Software Engineering, meaning that each system concern should be treated as a single conceptual unit. Aspect Orientation appeared as a software paradigm to promote the effective modularization of the concerns that are scattered and tangled across a system, i.e., the so-called crosscutting concerns.

Although it was born as a complementary paradigm to Object Orientation, Aspect Orientation was extended and started to cover other approaches, such as Multi-Agent Systems. Multi-Agent Systems are composed of autonomous entities, the software agents, that perform actions to accomplish their goals. Agents have some internal properties like autonomy, interactivity, proactivity, adaptation, learning, mobility and situatedness.

Many crosscutting concerns may be present in Multi-Agent Systems modeling as consequence of their functional and non-functional requirements and agents' internal properties. However, existing modeling languages and methodologies for Multi-Agent Systems do not support crosscutting concerns modularization.

This work aims to propose a modeling framework to modularize broadly scoped crosscutting concerns in a Multi-Agent System by applying goal-oriented modeling concepts. This framework, called ACROSS, uses conventional abstractions of the software agent paradigm, such as agents, agent goals, plans and action and provides new abstractions such as aspects and aspect goals to modularize elements that have a crosscutting impact over agent models. The framework also presents vertical and horizontal relationships to respectively define the hierarchy between these abstractions and how crosscutting occurs in each level. A XML-based notation to describe the generated models is proposed.

In addition, a case study of a Field Artillery Command and Control System is developed and ways to map the proposed models into code are suggested, considering implementation technologies such as JADE, for Multi-Agent Systems and AspectJ, for aspects. At last, some conclusions of the aplicability and limitatios of ACROSS are reached and some research orientations and future developments are proposed.

1 INTRODU ÃO

No cenário atual das pesquisas em Engenharia de Software, duas áreas vem recebendo uma grande atenção: Sistemas Multiagentes (SMAs) e Orientação a Aspectos. Ambas objetivam propor soluções mais vantajosas para o desenvolvimento de sistemas cada vez mais complexos e distribuídos, bem como favorecer a prática do reuso e promover manutenção e evolução menos custosas dos sistemas de software.

A Engenharia de Software de Sistemas Multiagentes vem emergindo como um paradigma promissor e mais adaptado à realidade dos sistemas atuais, que ficaram muito mais complexos e distribuídos com a onipresença da Internet e demais recursos de comunicações. Conceito herdado da Inteligência Artificial, os agentes de software diferem dos objetos porque, dentre outras razões, podem possuir grande grau de autonomia, refletindo a necessidade de sistemas cada vez mais descentralizados e complexos, ou seja, compostos de diversos subsistemas que comunicam-se entre si (JENNINGS, 2001).

Os SMAs são compostos por entidades autônomas denominadas agentes de software (JENNINGS, 2001) (WOOLDRIDGE, 1997). Estes agentes executam planos visando atingir objetivos individuais ou do sistema. Os objetivos são estados finais que devem ser alcançados pelos agentes. Os planos são conjuntos de ações que, por sua vez, são as computações que promovem a mudança de estados nos agentes. Desta maneira, *agentes*, *objetivos*, *planos* e *ações* são consideradas abstrações essenciais do paradigma multiagentes.

Agentes de software distinguem-se por apresentar algumas propriedades peculiares (WOOLDRIDGE, 1997). Dentre essas propriedades, destaca-se a capacidade de tomar decisões sem interferência externa (*autonomia*), a capacidade de comunicar-se com outros agentes para trocar informações ou compartilhar recursos (*interatividade*), a capacidade de agir por iniciativa própria (*pró-atividade*) e não apenas reagir ao ambiente e a estímulos externos, como faz o objeto, que simplesmente responde às chamadas externas aos seus métodos públicos. Soma-se a isto o próprio fato de que o agente deve estar situado em um ambiente, podendo perceber as alterações sofridas por este e, oportunamente, reagindo a elas.

Na Engenharia de Software, o princípio de separação de características está relacionado à decomposição e modularização (PARNAS, 1972). Sistemas de software complexos devem ser decompostos em unidades modulares menores, cada uma tratando de uma única

característica. Um módulo é uma unidade compilável e integrável para formar o programa, possuindo as seguintes propriedades (VONSTAA, 2000): encapsulamento, acoplamento e coesão. O modúlo deve estar encapsulado, o que significa tornar sua implementação interna invisível aos módulos clientes, disponibilizando serviços apenas por sua interface. Além disso, o módulo deve ter o menor acoplamento possível com os demais, ou seja, precisa minimizar sua dependência a eles. Por fim, os elementos que constituem um módulo precisam estar coesos, isto é, dotados de um forte interrelacionamento, devendo relacionar-se a um único conceito ou característica.

Estas propriedades são essenciais para gerenciar a complexidade de partes do sistema e dele como um todo. Entretanto, cada método de modelagem e programação utilizado no desenvolvimento impõe uma maneira dominante de decomposição e modularização, como por exemplo, objetos, explicitando algumas características e ocultando outras, igualmente importantes (TARR, 1999).

No entanto, além de decompor as características de um sistema, é preciso compô-las. Muitas vezes estas características são fortemente relacionadas, entrelaçadas e/ou sobrepostas, influenciando ou restringindo umas às outras, sendo chamadas de características transversais. Isto torna difícil a separação e análise das partes do sistema bem como a análise do impacto que umas exercem sobre as outras.

A Orientação a Aspectos tem suas origens na Programação Orientada a Aspectos (POA) proposta por KICZALES (1997) e surgiu como um paradigma complementar à Orientação a Objetos, buscando melhorar a separação e composição de características transversais, isto é, espalhadas e entrelaçadas no código das classes, através de uma nova entidade: o aspecto. Linguagens de programação orientadas a aspectos dão suporte à composição de características separadas e encapsuladas em aspectos, tornando a tarefa de composição transparente para o programador. A partir da POA, foi nascendo uma nova disciplina de Engenharia de Software que trata dos aspectos em todas as fases do ciclo de desenvolvimento de software, seja no nível de requisitos, arquitetural ou mesmo de testes. Com a Orientação a Aspectos deseja-se obter um acoplamento mais fraco, aumentar as possibilidades de reuso, bem como favorecer a evolução e manutenção do software.

Muitos horizontes de pesquisa estão sendo abertos em função destes dois paradigmas. Pesquisadores do mundo inteiro, sejam acadêmicos ou representantes da indústria, estão trabalhando as potencialidades da Orientação a Aspectos e dos Sistemas Multiagentes em cada uma das fases do ciclo de desenvolvimento de software, propondo modelos de requisitos, arquiteturais, linguagens e frameworks de programação e procedimentos de teste. Alguns trabalhos mais recentes estão começando a explorar aspectos em SMAs e a

presente dissertação vem somar esforços neste sentido, no que se refere à modelagem de características transversais em SMAs.

Assim, esta dissertação tem como objetivo prover recursos para viabilizar a modelagem de características transversais em SMAs através da proposta de um framework de modelagem que enderece tal questão. Na Seção 1.1 é explicado o problema a ser tratado nesta dissertação. Na Seção 1.2 são apresentados os objetivos deste trabalho e o resumo da solução proposta. Já a Seção 1.3 cita as contribuições oferecidas pela dissertação e, finalmente, a Seção 1.4 descreve a organização do restante do trabalho.

1.1 DESCRIÇÃO DO PROBLEMA

Tratar características transversais somente no nível de implementação não é suficiente porque muitas decisões de planejamento e de desenho já foram tomadas sem levar em consideração esta natureza transversal (NUSEIBEH, 2004). Existem atualmente muitas propostas de linguagens e metodologias de modelagem de SMAs tais como AUML (ODELL, 2000), ANote (CHOREN, 2005), MAS-ML (DASILVA, 2004b), Tropos (BRINKKEMPER, 2000), Gaia (JENNINGS, 2000), MaSE (DELOACH, 2000) e Message (CAIRE, 2001). Embora estas abordagens possuam numerosas divergências sobre as abstrações e diagramas a considerar ou se os agentes devem ou não ser vistos como extensões de objetos, uma questão todas elas têm em comum: não são oferecidos recursos capazes de modularizar as características transversais na modelagem de SMAs.

Além de caracterísiticas transversais tradicionais, como é o caso do tratamento de erros, trabalhos como, por exemplo, GARCIA (2005) mostram que as propriedades de agência, tais como autonomia, aprendizado, interatividade, entre outras, podem ser encaradas como características tranversais, uma vez que podem estar espalhadas e entremeadas às funcionalidades centrais dos agentes do sistema. Mecanismos que concretizam tais propriedades como algoritmos e heurísticas de tomada de decisão ou de aprendizado e protocolos de comunicações, se convenientemente modularizados, podem ser reutilizados em SMAs para atender os mais diversos propósitos.

Além disso, um conceito muito importante na Engenharia de Software é o de rastreabilidade. Rastreabilidade é a habilidade de descrever e acompanhar as informações sobre o ciclo de vida de um requisito tanto para frente (evolução) quanto para trás (passado) (GOTEL, 1994). Assim, não basta apenas implementar as caracteríticas transversais usando os recursos da POA, é também preciso manter modelos que reflitam essa modularização sob forma de aspecto.

A introdução de aspectos na modelagem de SMAs, no entanto, não é um processo imediato, uma vez que as abordagens orientadas a aspectos vem sendo exploradas principalmente no contexto de sistemas orientados a objetos ou baseados em componentes. A ausência do suporte explicíto à separação de características tranversais no desenvolvimento de SMAs leva a alguns problemas (GARCIA, 2006b) como (i) a limitação do raciocínio modular e composicional, o que leva a um projeto *ad hoc* de artefatos de modelagem, projeto e implementação; (ii) a replicação de elementos orientados a agentes como, por exemplo, o espalhamento de uma mesma ação por diversos agentes; (iii) a perda de informações essenciais, pois a dispersão e entrelaçamento das características tornam difícil sua recuperação e avaliação de seu impacto estrutural e comportamental no sistema e (iv) a redução de oportunidades de reuso e evolução, uma vez que os problemas do espalhamento e entrelaçamento dificultam esta prática.

modularização de características transversais na modelagem de SMAs, foram estudados conceitos de agentes, aspectos e orientação a metas e buscou-se chegar às abstrações e relacionamentos que atendessem a tal demanda. A partir do modelo concebido foi proposta uma descrição baseada em XML para representação dos modelos. Foi desenvolvido um estudo de caso para avaliação do framework.

1.3 LISTA DE CONTRIBUIÇÕES

Ao atingir os objetivos enunciados na seção anterior, espera-se que haja relevantes contribuições para a área de modelagem de características transversais em SMAs, bem como áreas correlatas. É possível levantar preliminarmente algumas destas prováveis contribuições e, ao fim do trabalho, atestar se elas foram, de fato, concretizadas. Assim, segue uma lista das contribuições esperadas.

a) Prover separação de características transversais na modelagem de SMAs: propor abstrações de modelagem que encapsulem tais características, permitindo seu isolamento das demais características do sistema;

b) Prover mecanismos de composição das características tranversais modeladas em separado com o resto da modelagem do sistema: propor relacionamentos que permitam caracterizar como ambos se juntam;

c) Propor um framework de modelagem independente de linguagem ou metodologia de modelagem de SMA em uso que congregue as abstrações e relacionamentos propostos: criar uma notação para representar as abstrações e relacionamentos, permitindo a construção de diagramas que ofereçam a visão integrada descrita no item anterior;

d) Estudar a aplicação de conceitos de Modelos de Metas na modelagem de características transversais em SMAs: investigar como os conceitos destes modelos podem ser úteis na proposição das abstrações e relacionamentos para modelar características transversais em SMAs;

e) Sugerir diretrizes para mapeamento da forma de modelagem proposta em código: baseando-se em tecnologias existentes para a implementação de SMAs e de aspectos, discorrer sobre maneiras de refletir no código os modelos concebidos.

Embora algumas abordagens existentes enderecem algumas destas contribuições, cabe-se ressaltar que o presente trabalho traz uma forma distinta e peculiar de fazê-lo. Comparações com abordagens existentes serão oportunamente realizadas. Algum esforço neste sentido já foi realizado em PERES (2006).

1.4 ORGANIZAÇÃO

A presente dissertação está organizada de forma que o **Capítulo 2 - Fundamentação Teórica** apresenta e explica os conceitos fundamentais que vão permear a compreensão do problema e o desenvolvimento de soluções. Além de apresentar os principais fundamentos de Sistemas Multiagentes e de Desenvolvimento Orientado a Aspectos, este capítulo traz conceitos de Modelos de Metas, mostrando como estes foram explorados no trabalho realizado na dissertação.

O **Capítulo 3 - Trabalhos Relacionados** expõe alguns trabalhos dentro da mesma proposta da presente dissertação, ou seja, modelar características transversais em SMAs. Além de discorrer um pouco sobre tais trabalhos, é feita uma comparação destes com a abordagem desenvolvida no decorrer desta dissertação, apontando diferenças, vantagens e desvantagens.

O **Capítulo 4 - O Framework ACROSS para a Modelagem Orientada a Aspectos no Desenvolvimento de Sistemas Multiagentes** é o núcleo do trabalho, uma vez que apresenta a solução proposta. ACROSS (*Agent CROSScutting Features Modeling Framework*) é o nome que foi dado ao framework de modelagem proposto para separar, compor e integrar características transversais na modelagem de SMAs, partindo das abstrações fundamentais destes últimos. Neste capítulo, são apresentadas as abstrações e relacionamentos que compõem o framework, bem como sua descrição baseada em XML e a avaliação dos resultados obtidos.

O **Capítulo 5 - Estudo de Caso** traz um estudo de caso de modelagem de um SMA de Comando e Controle voltado para atuação da Artilharia de Campanha, nível Brigada, em exercícios e operações de combate. Além disso, buscou-se, a partir deste exemplo, conjecturar como seria o mapeamento da modelagem realizada , usando como referência a plataforma JADE (BELLIFEMINE, 2005) para SMAs e a linguagem de aspectos AspectJ (KICZALES, 2001).

Por fim, o **Capítulo 6 - Conclusões** examina os resultados obtidos e aponta como foram concretizadas as contribuições enunciadas no Capítulo 1. Também são feitos comentários sobre trabalhos futuros e horizontes abertos a partir do trabalho realizado.

2 FUNDAMENTA ÃO TEÓRICA

Este capítulo trata dos alicerces conceituais sobre os quais o trabalho realizado na presente dissertação foi construído. Desta forma, serão apresentados os fundamentos teóricos de Sistemas Multiagentes, Desenvolvimento Orientado a Aspectos e Modelos de Metas. Com o estudo de Sistemas Multiagentes, foi possível familiarizar-se com seus conceitos fundamentais e peculiaridades, bem como situar-se a respeito do estado-da-arte das pesquisas e dos novos desafios trazidos por este paradigma. A pesquisa sobre Aspectos buscou compreender em que consiste esta proposta, além de investigar as motivações que impulsionaram sua criação, a evolução das pesquisas e os benefícios advindos de seu emprego. O estudo de fundamentos dos Modelos de Metas, por sua vez, teve importante papel no desenvolvimento do presente trabalho, pois foi através da perspectiva de objetivos que foi vislumbrada a forma apresentada para integrar e compor características transversais na modelagem de Sistemas Multiagentes.

2.1 SISTEMAS MULTIAGENTES

A Engenharia de Software, como qualquer outro conhecimento científico, está sujeita à evolução e ao surgimento de novos paradigmas. Tais paradigmas podem propor novas soluções frente a desafios decorrentes do aumento de complexidade dos sistemas computacionais ou, simplesmente, tornar o processo de desenvolvimento de software mais fácil (JENNINGS, 2001).

A *Orientação a Agentes* é um paradigma emergente de Engenharia de Software com raízes nos conceitos de agentes inteligentes oriundos da Inteligência Artificial (WOOLDRIDGE, 1997) e nasceu como resposta ao fato de que, com o avanço da Internet e com as oportunidades por ela criadas, sistemas computacionais fechados e *standalone* estão perdendo terreno para sistemas abertos, distribuídos, descentralizados e dinâmicos (SYCARA, 1998) (LUCK, 2004). Uma entidade autônoma é aquela que encapsula um determinado estado (não acessível às demais entidades) e é capaz de tomar decisões baseadas neste estado sem intervenção direta de outras entidades de qualquer natureza (humana ou computacional) (WOOLDRIDGE, 1997). *Sistemas Multiagentes* (SMAs) são compostos de entidades autônomas chamadas *agentes de software*.

Enquanto na Inteligência Artificial diz-se que um agente inteligente é uma entidade

capaz de perceber o ambiente através de sensores e de interagir com este ambiente através de atuadores (RUSSELL, 2003), na Engenharia de Software prefere-se dizer que um agente (de software) é uma entidade computacional encapsulada, situada em um ambiente, capaz de executar *ações* de maneira autônoma e flexível, de modo a buscar atingir os *objetivos* para os quais foi concebido (JENNINGS, 2001) (WOOLDRIDGE, 1997). Desta forma, agentes, objetivos e ações estão entre as principais abstrações do paradigma multiagentes.

Os objetivos são estados que devem ser alcançados por um ou mais agentes em um SMA e as ações são computações atômicas que promovem as mudanças de estados nos agentes. As ações executadas pelos agentes podem ser agrupadas em *planos*, que definem tanto como o agente se relaciona com o ambiente ao seu redor como a forma como ele interage com outros agentes. Assim, para cada objetivo a ser perseguido haverá um ou mais planos que possam ser executados pelo agente, cabendo a este a decisão de escolher o plano que julgue mais adequado. As ações que compõem um plano podem ser arbitradas estática ou dinamicamente, de modo que um agente possa abandonar ou alterar o plano original conforme sua percepção do ambiente e dos outros agentes for mudando.

Agentes podem ter objetivos individuais ou organizacionais. As organizações agrupam os agentes em SMA, definindo um conjunto de axiomas (regras, leis ou princípios) que devem ser obedecidos pelos agentes do sistema, estabelecendo papéis e comportamentos para os agentes (DASILVA, 2004b). É possível que agentes em um SMA possuam objetivos conflitantes ou concorrentes e não existem garantias que todos os objetivos de um agente devam ser atingidos, pois isso irá depender da eficiência e eficácia na escolha e execução de seus planos de ações frente ao desempenho de outros agentes e às diversas configurações assumidas pelo ambiente.

Um dos mais consagrados modelos de agente é o BDI (*Belief-Desire-Intention* , em português Crença-Desejo-Intenção) (RAO, 1991). Neste modelo, como o próprio nome sugere, o estado interno do agente é definido por estruturas de dados correspondentes aos três elementos-base, ou seja, crenças, desejos e intenções (RAO, 1991). As crenças (*beliefs*) representam a informação que o agente tem sobre o mundo (ambientes em que pode transitar). Já os desejos (*desires*) são os diversos objetivos dos agentes, que não necessariamente serão buscados ou atendidos, podendo inclusive existir inconsistência entre eles. As intenções (*intentions*) podem ser encaradas como os desejos selecionados pelos agentes, havendo compromisso por parte dos agentes de tentar concretizá-los por meio de execução de tarefas (ações). O modelo BDI formaliza a expressão das crenças, desejos e intenções do agente utilizando lógica proposicional e definindo operadores adequados

Um dos grandes questionamentos trazidos pelo paradigma multiagentes diz respeito a quando ele deve ser empregado, isto é, que tipo de sistema deve ser desenvolvido com base neste paradigma. Atualmente, existe um consenso que a orientação a agentes pode trazer bastante benefícios se aplicada ao desenvolvimento de sistemas complexos, ou seja, sistemas distribuídos e compostos de vários subsistemas que interagem entre si (JENNINGS, 2001). Um exemplo clássico de sistema desta natureza que desperta bastante interesse nos meios militares são os sistemas de Comando e Controle, que permitem o exercício de chefia e direção da autoridade militar e o acompanhamento das missões a serem cumpridas (LUCK, 2004) (IOERGER, 2003).

Em suma, a Orientação a Agentes constitui um novo e poderoso paradigma de desenvolvimento de software, oferecendo abstrações de mais alto nível que o paradigma da Orientação a Objetos (YU, 2001), aproximando-se mais ainda das características do mundo real. As subseções seguintes irão discutir brevemente as propriedades que distinguem os SMAs, bem como apresentar algumas tecnologias atualmente disponíveis para modelagem e implementação destes sistemas.

2.1.1 PROPRIE moiv-OSestes

mento de seus objetivos e não apenas reagir ao ambiente (WOOLDRIDGE, 1997), sendo assim uma conseq ência da autonomia. O nível de pró-atividade de um agente pode variar de acordo com sua especificação, não excluindo a existência de agentes puramente reativos.

Um agente sempre está situado em um ambiente, que pode ser um conjunto de sistemas operando em rede ou mesmo a própria Internet (WOOLDRIDGE, 1997). Assim sendo, o agente deve ser capaz de perceber este ambiente, extraindo informações dele e notando mudanças para poder responder prontamente ao cenário corrente com a tomada de decisões apropriada.

Neste trabalho, considera-se que as quatro propriedades acima definidas caracterizam, de fato, a agência. As demais propriedades podem também ser encontradas nos agentes, porém definem peculiaridades destes, ou seja, nem todo agente é capaz de adaptar-se, aprender, colaborar ou mover-se.

A adaptabilidade confere ao agente a capacidade de modificar em algum grau seus objetivos em função de mudanças no ambiente ou interesses de outros agentes (GARCIA, 2002). Autores como GARCIA (2002) consideram a adaptabilidade como propriedade básica, mas neste trabalho, julga-se que o agente pode ter consciência de sua posição em ambiente e tomar decisões autônomas sem que, necessariamente, estas reflitam uma resposta direta às mudanças no ambiente, não havendo, por exemplo, possibilidade de assumir compromisso com novos objetivos. É bom frisar que as propriedades dos agentes não são ortogonais, pois geralmente elas interagem como é o caso da adaptabilidade e autonomia, ou seja, sem a habilidade de tomar decisões por si próprio não há como o agente mudar de comportamento, ou melhor, adaptar-se em função do ambiente (GARCIA, 2004b).

O aprendizado é claramente uma propriedade adicional, pois nem todo agente é capaz de aprender com experiências passadas (GARCIA, 2002). Para tal, o agente precisa dispor de mecanismos específicos de aprendizado, que armazenem experiências prévias, ou seja, configuração do ambiente, decisões tomadas pelos agentes e resultados decorrentes destas decisões.

A colaboração é a propriedade dos agentes de cooperar uns com os outros para atingir objetivos individuais e do sistema (GARCIA, 2004b). Obviamente, para haver colaboração, pressupõe-se que os agentes são capazes de interagir, de modo que coordenem e sincronizem seus esforços.

A mobilidade, por sua vez, é a capacidade do agente de transportar-se de um ambiente de uma rede para outro para obter informações e recursos fora de seu *locus* original

(GARCIA, 2004b). Muitos pesquisadores e desenvolvedores de SMAs vêem os agentes como programas trafegando em uma rede coletando informações de negócio em aplicações de *e-business*, por exemplo (LUCK, 2004). Isto justifica os investimentos que vêm sendo feitos na tecnologia de agentes móveis, incluindo a pesquisa na área de segurança da informação.

2.1.2 MODELAGEM DE SMAS

Existem atualmente diversas propostas de linguagens e metodologias de modelagem de SMAs. Enquanto as linguagens de modelagem propõem notação padronizada com semântica própria para representar o domínio do problema, as metodologias consistem em conjuntos de procedimentos a serem seguidos para o desenvolvimento de SMAs, podendo empregar linguagens de modelagem para representação do sistema e seus componentes. É interessante ressaltar que nenhuma dessas linguagens incorpora o conceito de separação de características tranversais.

Dentre as linguagens de modelagem para SMAs estão a Agent UML (AUML) (ODELL, 2000), a MAS-ML (DASILVA, 2004a) e o ANote (CHOREN, 2005), sendo que as duas primeiras propõem-se a ser extensões de *Unified Modeling Language* (UML) (OMG, 2006b), enquanto a última quebra este paradigma tratando o agente como o conceito central. A MAS-ML, no entanto, apóia-se sobre uma camada de meta-modelo que vem a ser o framework conceitual *Taming Agents and Objects* (TAO) (DASILVA, 2004b), enquanto a AUML é construída sobre o meta-modelo de UML, o *Meta-Object Facility* (MOF) (OMG, 2006a). As linguagens que estendem UML incorporam a visão que o agente é uma extensão do objeto, dotada, no entanto, da capacidade de tomar decisões autônomas.

Diversas metodologias foram propostas para orientar o desenvolvimento de SMAs, como por exemplo, Gaia (JENNINGS, 2000) , Tropos (BRINKKEMPER, 2000) , Message (CAIRE, 2001) e MaSE (DELOACH, 2000). A Gaia baseia-se no conceito do SMA como uma organização computacional constituída de vários papéis que interagem uns com os outros, tratando da fase de análise e de projeto de SMA. O Tropos é uma metodologia dirigida por requisitos, mas que cobre tanto a fase de análise quanto projeto arquitetural, projeto detalhado e implementação. O Message, por sua vez, estende os diagramas de classe e de atividades de UML agregando conceitos relacionados a agentes como organizações, papéis, objetivos e ações. A MaSE enxerga o agente como uma abstração de mais alto nível que o objeto e propõe-se a ser uma metodologia essecialmente gráfica que cobre desde a captura de objetivos do sistema até a fase de projeto.

2.1.3 IMPLEMENTAÇÃO DE SMAS

Recentemente, os rumos das pesquisas na área de SMAs vêm apontando para a necessidade de desenvolvimento de linguagens de programação e plataformas apropriadas para a implementação destes sistemas, buscando firmar o paradigma através da garantia de oferecer um ciclo completo de desenvolvimento de software. Este é um dos desafios cruciais para a orientação a agentes, pois permitirá que o paradigma possa vir a ser, de fato, um padrão de indústria.

Dentre os paradigmas de linguagens de programação orientada a agentes que vem sendo estudados, destaca-se o desenvolvimento de linguagens declarativas baseadas em formalismo lógico, linguagens imperativas como o *JACK Agent Language* (JAL) (AOS, 2005) e plataformas de agentes como o *Java Agent Development* (JADE) (BELLIFEMINE, 2005). BORDINI (2006) traz um atualizado resumo sobre o estado-da-arte das tecnologias para implementação de SMAs.

2.2 DESENVOLVIMENTO ORIENTADO A ASPECTOS

Na Engenharia de Software, *concern* é o termo consagrado em inglês para definir uma característica, preocupação ou interesse de um sistema, ou seja, um elemento de sua especificação. Traduzir tal termo para o português conservando toda a sua expressividade não é tarefa das mais fáceis e os trabalhos nacionais na área de Desenvolvimento Orientado a Aspectos pesquisados dividem-se entre os termos "interesse" (PIVETA, 2004) e "característica" (DASILVA, 2006). A opção adotada pelo presente trabalho é a última, acolhendo a justificativa de DASILVA (2006) que alega que o termo interesse já tem, na Engenharia de Requisitos, uma outra conatação ligada à noção de *stakeholder*, ou seja, os interesados no sistema (usuários, clientes, etc).

A princípio separação de características é conceito bem estabelecido de Engenharia de Software, indicando que cada característica do sistema deve ser tratada como uma unidade conceitual isolada (DIJKSTRA, 1976). Este princípio deve ser levado em consideração durante todo o desenvolvimento de software, de forma tornar mais fácil encarar a complexidade através do emprego de abstrações e mecanismos para aumentar a decomposição e modularização do sistema (PARNAS, 1972). Decompor um sistema significa dividi-lo em partes menores, constituindo módulos que, por sua vez, devem abrigar apenas uma característica. Para atingir tais objetivos na modelagem de um sistema, é necessário que a linguagem de modelagem em uso e seu meta-modelo disponham de abstrações e mecanismos de composição adequados (GARCIA, 2006b).

A Orientação a Aspectos é uma disciplina de Engenharia de Software que surgiu motivada pela necessidade de obter, de fato, a separação de características. Este paradigma surgiu com a Programação Orientada a Aspectos (POA), proposta por KICZALES (1997), mas foi estendido para todas as fases do ciclo de desenvolvimento de software. A proposta da POA era complementar a programação OO, pois esta era muito adequada para representar o comportamento (funcionalidades) dos objetos, porém deficiente para tratar aspectos estruturais ou comportamentais de características do sistema que não são encapsuláveis em construções modulares como classes, operações, etc. Tais características são conhecidas como características transversais (KICZALES, 1997) (TARR, 1999).

As características transversais estão relacionadas à ocorrência de dois problemas na modularização: o espalhamento (*scattering problem*) e o entrelaçamento (*tangling problem*) (KICZALES, 1997). O problema do espalhamento ocorre quando uma dada característica encontra-se espalhada por diversos módulos do sistema, já o problema do entrelaçamento é fruto da co-existência em um mesmo módulo de mais de uma característica.

Para lidar com características transversais uma nova abstração foi criada: o *aspecto*. Aspectos são entidades de primeira-classe que provém representação modular às características tranversais (TARR, 1999). O propósito da orientação a aspectos é permitir a implementação modular de comportamento que não se encaixa em um modelo particular em uso (BANIASSAD, 2004). Além disso, deve oferecer novos mecanismos de composição para proporcionar raciocínio modular e composicional na presença de características transversais. As linguagens de modelagem e metodologias orientadas a agentes, como as citadas em 2.1.2, atualmente não oferecem tais mecanismos capazes de refletir a natureza transversal de muitas características dos SMAs.

2.2.1 PROGRAMAÇÃO ORIENTADA A ASPECTOS

Conforme foi dito anteriormente, a Orientação a Aspectos começou como paradigma de programação. No esteio da teoria apresentada surgiram linguagens de programação como o AspectJ (KICZALES, 2001) que lidam com programas-base orientados a objetos. Essas linguagens introduziram conceitos bastante interessantes para caracterizar o *crosscutting*, tais como *join points*, *pointcuts*, *advice*. *Join point*, ou pontos de junção, é um ponto bem definido na execução do programa em que incidem um ou mais aspectos, *pointcut* é um predicado que especifica coleções de *join points* onde o comportamento do aspecto deve incidir, *advice* é uma construção assemelhada aos métodos na OO que pode ser ligada aos

pointcuts. Um *advice* pode ser executado antes, durante ou depois de um *pointcut* ser alcançado.

Duas propriedades são necessárias para POA: a *transparência* e a *quantificação* (FILMAN, 2005). Na POA, a transparência significa que, olhando apenas o código do programa-base, não é possível determinar se e quando o código dos aspectos será executado. Isso significa que o programa-base, codificado convencionalmente, isto é, não orientado a aspectos, fica "limpo", ou seja, não existem chamadas ou qualquer outro traço que indiquem que antes, durante ou depois da execução de um deteminado trecho de código deverá ser executado um outro trecho encapsulado em um aspecto. Desta forma, é na programação do aspecto que se define, através dos *join points*, *pointcuts* e *advices*, o momento em que determinado código do aspecto precisa ser executado.

Segundo FILMAN (2005), a POA reflete o desejo de construir programas que obedeçam a seguinte estrutura: "No programa P, sempre que a condição C for alcançada, execute a ação A." Neste caso, P é um programa codificado convencionalmente. A POA fez com que três questionamentos importantes surgissem: (i) quais os tipos de condições C podem ser especificadas (quantificação); (ii) como uma ação A deve interagir como o programa P e com as demais ações (interface) e (iii) como o sistema deve compor e integrar a execução do programa P com as ações apropriadas (*weaving*).

Assim, a quantificação incorpora as noções de definição de *join points*, especificando as condições C, e de mecanismos para associá-los a determinados aspectos, fazendo esta ponte entre o programa-base e a codificação dos aspectos. Mecanismos de quantificação podem ser estáticos ou dinâmicos. A quantificação estática ocorre de forma determinística, ou seja, sempre que um dado ponto é atingido, o comportamento do aspecto deve ser considerado. A quantificação dinâmica, por sua vez, baseia-se em eventos assíncronos como, por exemplo, a ocorrência e tratamento de exceções.

A definição de interface trata da estrutura do código dos aspectos, definindo interação de um aspecto e o código-base e entre os demais aspectos. Inclui também questões de parametrização, definição de contextos do programa-base acessíveis a um aspecto e ordenação de aspectos em um mesmo *locus* (FILMAN, 2005).

Por fim, *weaving* é o mecanismo que define como o sistema irá juntar o código-base com os aspectos. O mecanismo de *weaving* da POA pode, por si só, juntar as assertivas quantificadas (estáticas e dinâmicas) do código dos aspectos com o programa-base e produzir os direcionamentos primitivos a serem executados.

2.2.2 *EARL ASPECTS*

Os conceitos apresentados na proposta da POA deram origem ao que ficou conhecido como *Early Aspects* que é a parte do estudo da Orientação a Aspectos que cuida das características transversais nas fases iniciais do desenvolvimento de software, ou seja, a Engenharia de Requisitos e Projeto Arquitetural.

A Engenharia de Requisitos Orientada a Aspectos (AORE, em inglês) é um tópico diretamente relacionado à presente dissertação, uma vez que engloba a modelagem de sistemas. A AORE objetiva tratar de forma sistemática a identificação, separação e visualização e modelagem dos interesses transversais presentes em requisitos funcionais e não-funcionais de um sistema, bem como permitir o rastreamento destes interesses durante todo o ciclo de desenvolvimento do sistema.

ARAÚJO (2005) define o aspecto no nível de requisitos como uma propriedade de escopo geral representada por um requisito ou por um conjunto coerente de requisitos, como, por exemplo, os requisitos de segurança, que afetam muitos outros requisitos do sistema de forma a poder restringir e/ou alterar o comportamento específico dos requisitos afetados. ROSENHAINER (2004) expressa o conceito de requisitos tranversais de uma forma mais explícita, afirmando que um requisito A atravessa um requisito B se a decomposição do software é feita de tal maneira que B não pode ser satisfeito sem levar A em conta.

Dentre as abordagens para identificação de características transversais em estágios precoces do desenvolvimento de sistemas destacam-se o emprego de técnicas de recuperação de informação e a inspeção de requisitos, que pode ser apoiada pela consulta a catálogos que tragam os exemplos mais recorrentes da ocorrência de tais características (ROSENHAINER, 2004). Pode-se pensar também em abordagens híbridas que mesclem ambas as técnicas. A forma como estão estruturados os requisitos do sistema é algo que precisa ser levado em conta, uma vez que existem abordagens que tratam apenas da mineração de aspectos em requisitos estrurados segundo algum padrão léxico como o *Theme/Doc* (BANIASSAD, 2004) e outras que cobrem requisitos informalmente documentados e heterogêneos (entrevistas, linguagem natural não estruturada, etc) como na proposta de SAMPAIO (2005).

Segundo CHITCHYAN (2005), as principais abordagens de modelagem de requisitos orientadas a aspectos baseiam-se em paradigmas conhecidos tais como *viewpoints*, orientação a metas, cenários, *use cases*, abordagens baseadas em componentes, bem como outros direcionamentos, como é o caso do *Theme*.

Enquanto as abordagens baseadas em *viewpoints* (pontos-de-vista) procuram focar os requisitos aspectuais e não-aspectuais do sistema sob a perspectiva de seus atores e demais entidades que agem sobre o sistema, as abordagens orientadas a metas tratam tais requisitos sob a perspectiva dos objetivos do sistema, definindo como os objetivos podem afetar uns aos outros. As abordagems baseadas em cenários e *use cases*, por sua vez, tratam das funcionalidades do sistema, de modo que são buscadas formas de estender os artefatos de modelagem existentes para suportar a modelagem de características não-funcionais do sistema, como fez ARAÚJO (2002) ao criar extensões de UML. O Theme, como o próprio nome sugere, é baseado na noção de tema, que representa uma característica do sistema que pode ser básica ou tranversal (aspectual).

2.3 MODELOS DE METAS

Objetivos são uma das abstrações essenciais do paradigma multiagentes, o que leva alguns autores a afirmar categoricamente que agentes são orientados a objetivos (SHEN, 2005), (PARK, 2000). Abordagens orientadas a objetivos podem ser muito úteis na fase de modelagem de requisitos por serem capazes de refletir as propriedades características dos SMAs (PARK, 2000) através de uma reprodução mais fiel do modelo mental do agente, o que inclui a explicitação de seus objetivos. Com base nestas informações e no fato de que a literatura consagra que muitas propriedades dos sistemas multiagentes podem ser tratadas como aspectos (GARCIA, 2005) (GARCIA, 2006c) (LOUGHRAN, 2006), conclui-se que valia a pena explorar as potencialidades dos chamados *modelos de metas*.

Os *goals* (objetivos, metas) são componentes essenciais no processo de Engenharia de Requisitos, uma vez que explicam o porquê da existência do sistema e do comportamento de suas entidades. A vantagem de explicitar os objetivos em atividades como a elicitação e modelagem de requisitos é discutida em muitos trabalhos, como, por exemplo, VAN-LANSWEERDE (2001). Na Análise Orientada a Objetos, o primeiro passo é especificar os casos de uso do sistema, mostrando como os atores deverão interagir com este. Esta etapa, na verdade, encobre, de certa forma, o importante papel dos objetivos, afinal sem conhecê-los, não é possível conjecturar como deve ser a interação dos atores com o sistema. Os objetivos, no contexto da compreensão e especificação dos requisitos do sistema, representam os fins que o sistema e seu ambiente se propõem a atingir, bem como as intenções de cada um dos *stakeholders* (usuários, clientes, interessados no sistema em geral) e racionalizam as dependências estratégicas entre eles (YU, 1994) (BRESCIANI, 2004).

Objetivos podem estar associados tanto às características funcionais, ou seja, serviços

oferecidos pelo sistema, quanto às características não funcionais, como a qualidade destes serviços (VANLANSWEERDE, 2001). Além disso, eles podem ser formulados em diferentes níveis de abstração, deste o alto nível (características estratégicas) até o baixo nível (características técnicas), que é bem mais detalhado que o primeiro e dependente de soluções de desenvolvimento particulares. Objetivos de alto nível, no entanto, são bem mais estáveis, uma vez que refletem os fins para os quais o sistema foi concebido, algo que dificilmente sofre expressivas mudanças.

Abordagens orientadas a metas descrevem sistemas e suas entidades a partir de seus objetivos. A análise de objetivos é baseada em métodos relacionados aos problemas de redução de Inteligência Artificial, como as decomposições em árvores AND/OR e contribuições qualitativas positivas ou negativas, permitindo também a modelagem de espaços de soluções alternativas (NILSSON, 1971). A análise de objetivos pode ser encarada com um guia sistemático para identificação de objetivos e a modelagem de objetivos oferece uma forma natural de estruturar documentação de requisitos complexos, tornado-os mais legíveis. A modelagem de objetivos caracteriza-se usualmente pela determinação de seu tipo, seguindo alguma taxonomia pré-definida, de seus atributos e seus relacionamentos (VANLANSWEERDE, 2001).

A perspectiva da Orientação a Metas pode ser bem útil para apoiar o raciocínio de separação, composição e integração de características transversais em Sistemas Multiagentes, uma vez que objetivo é uma das abstrações centrais do paradigma. O presente trabalho buscou explorar idéias e conceitos de modelos de metas na construção do framework de modelagem de características tranversais em SMAs. Três modelos de metas foram particularmente estudados neste sentido: o i* (YU, 1997), o KAOS (DARDENNE, 1993) e o V-Graph (YU, 2004).

O framework i* foi desenvolvido para modelagem e raciocínio acerca de ambientes organizacionais e seus sistemas de informação. O i* apresenta as noções de objetivos, atores e dependências entre os atores para realizar os objetivos. O conceito central do i* é o chamado ator intencional que possui propriedades tais como objetivos, crenças, habilidades e compromissos. Atores dependem uns dos outros para atingir seus objetivos, seja compatilhando recursos ou realizando ações que os impactem mutuamente, tal como acontece com os agentes, que interagem com propósito semelhante. Embora não tenha influenciado diretamente na construção da proposta desta dissertação, o i* permite a compreensão da essência dos modelos de metas, que é similar a dos agentes, ou seja, trabalhar com objetivos bem definidos e executar ações para alcançá-los.

O KAOS é uma abordagem de três componentes: um modelo conceitual para adquirir

e estruturar modelos de requisitos com linguagem própria associada, um conjunto de estratégias para elaborar modelos de requisitos e um assistente automático para guiar a aquisição de requisitos segundo o conjunto de estratégias proposto. O KAOS também trabalha o conceito de agente como um objeto que processa algumas ações, controlando, assim, sua transição de estados e tendo poder de decisão sobre seu comportamento (autonomia). O termo objeto, para o KAOS, representa um ponto de interesse que possa ser referenciado pelos requisitos. O KAOS, além dos agentes e ações, também traz objetivo como uma abstração essencial. Os objetivos, isto é, estados a serem atingidos pelo sistema, estão relacionados aos agentes, de forma que cada agente tenha um conjunto de objetivos que almeja alcançar. O KAOS relaciona objetivos com agentes alternativos através de relacionamentos de responsabilidade *responsibility links* do tipo *OR* (VANLANSWEERDE, 2001) e trata conflitos de objetivos pela gerência de relacionamentos de desejo *wish links*. Além disso, o KAOS introduz relacionamentos de operacionalização *operationalization links* do tipo *AND/OR* para associar objetivos a ações, assegurando o cumprimento de pré-requisitos e respeitando as restrições do sistema. Embora seja um framework conceitual que trata de agentes, o KAOS não trabalha o mérito das características transversais, mas contribui para o presente trabalho, uma vez que reforça o entendimento da hierarquização entre principais abstrações do paradigma multiagentes e, sobretudo, trabalha a associação de agentes a objetivos e objetivos a ações, servindo de inspiração para esta dissertação.

De todos os modelos de metas estudados, o que teve aplicação mais direta na dissertação foi o V-Graph. O V-Graph representa requisitos funcionais e não-funcionais em termos de modelos de metas. Requisitos não-funcionais, como segurança, performance, etc encontram-se, em geral, espalhados e entrelaçados nos requisitos funcionais, de modo que os primeiros, muitas vezes, são bons candidatos a aspectos. O V-Graph é um grafo com o formato da letra "V" em que os dois vértices superiores representam os requisitos funcionais e não-funcionais, chamados respectivamente de metas e softmetas [1] (DASILVA, 2006). Metas e softmetas distinguem-se pelos conceitos de *satisfy* e *satisfice* (MYLOPOULOS, 1992). O primeiro indica que uma meta pode ser completamente satisfeita se o conjunto de metas em que se decompõe é satisfeito, já o último diz que uma meta pode ser suficientemente satisfeita pelas metas, softmetas e ações que contribuem ou estão correlacionadas positivamente a ela.

[1] DASILVA (2006) utilizou a palavra *meta* como tradução para o termo *goal* e, na falta de um correspondente mais adequado, mas buscando manter coerência, adotou *softmeta* como correspondente de *softgoal*.

Os relacionamentos de correlação entre metas e softmetas caracterizam como metas podem impactar em softmetas. Para tal estabeleceu-se os relacionamentos *make*, *help*, *hurt* e *break* (YU, 2004). *Make* e *help* representam impacto positivo, sendo o primeiro mais forte que o segundo, enquanto *hurt* e *break* trazem a idéia de impacto negativo, sendo o último mais forte. Além disso, ainda pode-se definir o relacionamento *unknowm* para denotar que não há impacto conhecido (DASILVA, 2006). A iniciativa de quantificar e qualificar a influência de uma entidade sobre a outra e a escala utilizada é uma concepção bastante interessante no esforço de obter um entendimento global do sistema em questão. Assim sendo, os relacionamentos de correlação do V-Graph se redefinidos e aplicados a outros contextos podem ser bastante úteis no sentido de obter o entendimento global do sistema.

2.4 CONSIDERAÇÕES FINAIS

Com o estudo das peculiaridades dos SMAs e da aplicabilidade da Orientação a Aspectos vislumbrou-se as possibilidades de estender os conceitos de aspectos para que também atuem de forma complementar ao paradigma multiagentes na modelagem de sistemas de tal natureza. Da mesma forma como na OO, os SMAs podem apresentar características transversais e os benefícios da Orientação a Aspectos, ou seja, favorecer reuso, manutenção e evolução são igualmente desejáveis ao tratar dos SMAs. Assim, o desafio é justamente conceber uma forma de viabilizar esta extensão na modelagem de SMAs.

Conclui-se também que um dos conceitos-chave para trabalhar o *rationale* dos requisitos de SMAs, ou seja, suas intenções e motivações, é o de objetivo. Sabendo que os Modelos de Metas são focado em objetivos, julgou-se que poderiam oferecer sensíveis contribuições na construção da solução proposta, permitindo hierarquizar e definir relacionamentos entre abstrações associadas a agentes e a seus aspectos. De fato, conceitos e idéias oriundos destes modelos influenciaram o presente trabalho.

3 TRABALHOS RELACIONADOS

Neste capítulo serão apresentados alguns trabalhos relacionados ao desenvolvido na presente dissertação, de modo tanto a situar o trabalho realizado no universo das pesquisas que vêm sendo promovidas em áreas afins, como também confrontá-lo com outras propostas que tratem de modularização de características transversais na modelagem de SMAs.

Desta forma, o capítulo está organizado de maneira que a Seção 3.1 trata de expor alguns trabalhos sobre *Early Aspects*, para dar uma medida de como estes influenciaram a abordagem proposta. A Seção 3.2, por sua vez, trata de trabalhos que transitam dentro da mesma proposta desta dissertação, ou seja, separação, composição e integração de aspectos em SMAs. Por fim, na Seção 3.3 é feita uma discussão versando sobre as limitações dos trabalhos relacionados.

3.1 TRABALHOS SOBRE *EARL ASPECTS*

Early Aspects é a parte da disciplina de Orientação a Aspectos que cuida das características transversais nas fases iniciais do desenvolvimento de software, isto é, a Engenharia de Requisitos e Projeto Arquitetural. A modelagem de requisitos, foco desta dissertação, é uma das atividades relacionadas à Engenharia de Requisitos, que segundo a classificação sugerida por DOPRADO LEITE (2001) é composta das tarefas de elicitação, modelagem e análise de requisitos. Na modelagem, os requisitos elicitados, ou seja, levantados junto ao cliente, são representados de acordo com uma técnica, notação ou linguagem específica. A modelagem de requisitos orientada a aspectos oferece suporte a separação, composição e integração de características transversais.

O ano de 2002 é considerado um marco para as pesquisas de Engenharia de Requisitos Orientada a Aspectos (AORE), já que foi quando começaram a surgir vários trabalhos sobre esta área, tais como RASHID (2002), MOREIRA (2002) e BRITO (2002) em conferências internacionais. A motivação para estes trabalhos decorre do fato de que abordagens como *use cases* e *viewpoints* conseguiam modularizar características ligadas aos *stakeholders*, porém não eram capazes de assegurar sua consistência com os requisitos e restrições globais (RASHID, 2002). Uma abordagem orientada a aspectos, no entanto, seria capaz de prover separação das propriedades transversais funcionais e não-funcionais

no nível de requisitos.

RASHID (2002) expôs um modelo para a AORE que consiste em descobrir e especificar características e, a partir delas, identificar quais delas possuem comportamento transversal, classificando-as de candidatos a aspectos. Além disso, é preciso especificar os candidatos a aspectos, detalhando sua associação a requisitos elicitados, de modo que se compreenda melhor sua atuação e a extensão de sua influência para, então, priorizá-los. Por fim, devem ser especificadas suas dimensões de produzir um mapeamento para definir quais serão, de fato, os aspectos e quais serão simplesmente funções ou decisões arquiteturais, por exemplo. Os aspectos irão influenciar todo o ciclo de desenvolvimento do sistema pautando-se em alguns requisitos do sistema. Este trabalho, embora traga um entendimento fundamental sobre as atividades iniciais associadas a modelagem de aspectos, não trata a representação e especificação interna do aspecto, apenas o separa dos demais candidatos a aspecto.

O trabalho de DASILVA (2006), por sua vez, tratou especificamente de separação, integração e composição de características transversais na modelagem de requisitos. (DASILVA, 2006) propõe um metamodelo para integração de características transversais, registrando, controlando e analisando a interação entre requisitos. Neste metamodelo, separa-se as caracterísiticas e relacionamentos tranversais presentes nos requisitos, empregando-se algumas diretrizes de separação e representado-os conforme uma linguagem especificada, produzindo, assim, um modelo sobre o qual serão aplicadas regras de composição, formando um modelo integrado. Do modelo integrado podem ser extraídas diversas visões que permitam exposição de modelos parciais. Apoiando-se no processo de identificação de aspectos em modelos de metas definido por YU (2004), DASILVA (2006) aplica sua estratégia orientada a aspectos para modelagem de requisitos em modelos orientados a metas, mais especificamente o V-Graph, produzindo assim um V-Graph orientado a aspectos, o AOV-Graph.

Os modelos de requisitos orientados a aspectos de DASILVA (2006) são expressos por uma linguagem de modelagem denominada Linguagem de Modelagem de Requisitos Orientada a Aspectos (LMROA), que representa os modelos, seus componentes e relacionamentos entre eles, incluindo os relacionamentos transversais, bem como um modelo de *join points*. Trata-se, no entanto, de uma abordagem complexa (*heavyweight*) para especificar os relacionamentos transversais (modelo-núcleo), apoiando-se em mecanismos específicos de implementação do AspectJ (PERES, 2006).

3.2 TRABALHOS SOBRE MODULARIZAÇÃO DE CARACTERÍSTICAS TRANSVERSAIS EM SMAS

Os trabalhos citados até aqui não contemplam especificamente as peculiaridades do paradigma de agentes de software, não tratando, assim, da modularização de características transversais em SMAs. Recentemente, no entanto, foram surgindo trabalhos que exploram este ramo específico. Duas abordagens destacam-se no cenário atual que são os trabalhos de GARCIA (2006a) e GARCIA (2006b), bem como o de SILVA (2006a).

GARCIA (2006a) e GARCIA (2006b) apresentam um framework de modelagem para enriquecer modelos orientados a agentes com aspectos, permitindo a modularização de características transversais em amplo escopo, sejam elas conseq encias de requisitos funcionais, não-funcionais ou das propriedades peculiares dos agentes. Este framework baseia-se em um modelo de agentes que leva em conta objetivos e ações como abstrações fundamentais dos agentes de software. Além disso, reutiliza o modelo de aspectos proposto por CHAVEZ (2004) em que os elementos são organizados em três modelos conceituais inter-relacionados: (i) o modelo de componentes, que é o modelo da entidade-base, ou seja, o modelo de agentes; (ii) o modelo de *join point*, que trata das especificações e restrições dos tipos de *join points* considerados e, por fim, (iii) o modelo-núcleo, que descreve os aspectos e o *crosscutting*.

Quatro categorias de composição são estabelecidas e associadas a proposição de operadores bem definidos. São elas: (i) aspecto-agente, (ii) objetivo-objetivo, (iii) objetivo-ação, (iv) ação-ação. Na primeira categoria, define-se o operador INTRODUCE, que indica que um aspecto introduz um conjunto de objetivos (com as respectivas ações associadas) em um agente. Na segunda, que exprime a composição de objetivos entre agentes e aspectos, são empregados operadores que caracterizam formas de substituição e decomposição de objetivos como o MERGE, REPLACE, AND, OR, XOR, BEFORE, AFTER (estes dois últimos equivalem a um AND com ordem de atendimento pré-estabelecida e só constam de GARCIA (2006b)). A composição objetivo-ação, por sua vez, sinaliza que um aspecto introduz um ação em um objetivo do agente (operador INTRODUCE). Por fim, a composição ação-ação estabelece que uma ação de um aspecto relaciona-se com um ação de um agente seja por fundir-se a ela (operadores MERGE-BEFORE e MERGE-AFTER) ou por substitui-la (REPLACE). Além disso, são sugeridas extensões na notação do ANote (CHOREN, 2005) para representar as abstrações e os relacionamentos propostos, embora o framework proposto não seja obrigatoriamente atrelado a esta linguagem de modelagem.

SILVA (2006a) propôs uma abordagem sistemática para especificar aspectos e aplicá-la

no projeto arquitetural de SMAs. Este trabalho buscou empregar o conceito de papéis de modelagem (*model roles*) para especializar o metamodelo de agência descrito por SILVA (2006b). Este modelo de agência estende o metamodelo de UML 2.0, que oferece suporte à descrição arquitetural e reflete um padrão cliente-servidor em que os agentes requisitam e oferecem serviços uns aos outros (SILVA, 2006a).

Esta abordagem trabalha com elementos arquiteturais que tratam do agente e de recursos organizacionais. Considera-se os objetivos, planos e ações como abstrações dos agentes. Por ser centrada em um padrão cliente-servidor, estabelece que a colaboração entre agentes é marcada pela presença de um agente dependente de um serviço (*Depender*) e de outro provedor do serviço requisitado (*Dependee*) que acordam sobre a prestação deste serviço (*Dependum*). Ocorre, no entanto, que tal abordagem fundamenta-se no conceito de que papéis de agentes podem definir objetivos e ações compartilhadas por outros papéis, ou seja, que alguns papéis são aspectuais. Este é o caso, por exemplo, do agente DF oriundo do FIPA-OS (FIPA-OS, 2001) e também implementado pelo JADE. Como este agente registra e permite a consulta aos serviços oferecidos por cada agente do SMA, ele tem que colaborar com os demais agentes sempre que eles objetivarem registrar ou buscar um serviço, atuando, assim, de forma transversal. Esta é uma visão bastante diferente tanto da abordagem de GARCIA (2006a) e GARCIA (2006b), como também da presente dissertação.

Embora SILVA (2006a) use largamente estereótipos, como preconiza o *profile* de UML 2.0 que trata de descrição arquitetural, não há forma explicíta de definir um aspecto (algo como «Aspect», por exemplo). Além disso, considera que a representação das propriedades dos agentes é inerente à plataforma de implementação, não preocupando-se em modularizá-las na modelagem arquitetural.

3.3 DISCUSSÃO

DASILVA (2006) traz um foco de modelagem de requisitos bastante genérico, de forma a não definir as abstrações para a representação de agentes e aspectos em SMAs nem, sobretudo, especificar quais seriam os tipos de relacionamentos transversais presentes. É interessante ressaltar que relacionamentos transversais entre objetivos e entre planos ou ações possuem natureza bastante distinta, afinal o objetivo pode ser considerado um conceito subjetivo, uma vez que é um estado a ser atingido pelo sistema. Já as ações são computações concretas, ou seja, compõem planos que especificam o passo a passo a ser executado por elementos do sistema. Embora DASILVA (2006) não discuta questões

como as conexões conceituais que podem existir entre relacionamentos de correlação e de *crosscutting*, acredita-se que a exploração deste tópico possa render bons frutos quando se trata especificamente de modelagem de SMAs.

Apesar disto, este trabalho inspirou bastante a presente dissertação. Possibilidades foram visualizadas ao analisar o AOV-Graph, embora a transversalidade neste modelo seja expressa simplesmente em termos de um relacionamento denominado *cross* e especificada de acordo com os mecanismos do AspectJ. Enxergou-se a possibilidade de redefinir os relacionamentos de correlação para que eles expressassem a influência dos aspectos no modelo dos agentes, incrementando o raciocínio global sobre os objetivos do sistema,

de linguagens orientadas a objeto, a concretização do conceito de objetivo torna-se algo não trivial. No entanto, a composição de objetivos pode trazer uma melhor compreensão global do sistema se empregados relacionamentos expressivos para caracterizá-las. Por fim, acreditou-se que a definição de relacionamentos transversais entre agentes e aspectos devem ocorrer sempre no mesmo nível, não admitindo composições do tipo objetivo-ação. Isso simplesmente significaria que há uma relação entre objetivos de agentes e aspectos que implica na existência de um relacionamento entre seus planos e entre suas ações, tornando o raciocínio de modelagem mais metódico.

SILVA (2006a), além de tratar de projeto arquitetural em vez de modelagem de requisitos, afirma que não trata de caracterísiticas tranversais relacionadas a propriedades internas do agentes (autonomia, interatividade, adaptação, etc), pois considera que estas dizem respeito à plataforma de implementação, que irá definir como estas propriedades se manifestam. Esta dissertação, no entanto, não compartilha desta postura, pois reconhece que mecanismos reutilizáveis que promovam as propriedades internas dos agentes podem (e devem) ser modularizados.

Outro ponto é que SILVA (2006a) trata alguns papéis de agentes como aspectos, embora não defina um estereótipo específico para sinalizar isto. Tal ponto-de-vista é também divergente com a presente dissertação que considera que, sem exercer papel, o agente, enquanto entidade de software, não tem funcionalidades e, logo, não vai ter objetivos, planos e ações, pois estes são definidos em função do papel exercido. O exemplo motivacional de SILVA (2006a) trata do agente DF (*Directory Facilitator*) do FIPA-OS (FIPA-OS, 2001) que registra os serviços oferecidos pelos agentes do SMA e permite a busca por agentes que prestam determinado serviço (páginas amarelas). Pela abordagem do trabalho, como diversos agentes oferecem e buscam serviços, os objetivos e ações associados ao papel de oferecimento e busca de serviços podem ser tratados como aspectos. Já esta dissertação enxergaria um aspecto que manipularia as páginas amarelas (*ellowPagesHandle*, por exemplo) que atravessaria a modelagem dos agentes que precisassem acessar este serviço. Considera-se que esta última abordagem é mais intuitiva na modelagem de requisitos, pois bastaria estabelecer que os objetivos deste aspecto seriam o registro e busca de serviços e definir os planos e ações que devem ser executados pelos agentes que desejarem utilizar as páginas amarelas.

Desta forma, em resumo, a abordagem da presente dissertação pauta-se por considerar o aspecto uma entidade de primeira-classe que, no paradigma multiagentes, deve compartilhar de suas abstrações, de forma semelhante ao proposto por GARCIA (2006b) e GARCIA (2006a). Além disso, considera-se que as propriedades internas do agente manifestam-

se através da presença de mecanismos que as concretizem. Sobretudo, enxerga-se que os aspectos podem constituir módulos reutilizáveis e busca-se promover a rastreabilidade dos modelos durante o ciclo de desenvolvimento de software, bem como tornar o processo de modelagem de requisitos uma tarefa que permita ao usuário uma melhoria no entendimento global do sistema.

4 O FRAMEWORK ACROSS PARA A MODELAGEM ORIENTADA A ASPECTOS NO DESENVOLVIMENTO DE SISTEMAS MULTIAGENTES

Neste capítulo será apresentado o framework de modelagem encarregado de modularizar as características transversais em SMAs. Este framework foi batizado com o nome de ACROSS (Agent CROSScutting Concerns Modeling Framework). A palavra *across*, em inglês significa "através de", que evoca também ao sentido de transversalidade, remontando às características transversais presentes na modelagem dos SMAs. O ACROSS oferece uma gama de abstrações e relacionamentos para separar, compôr e integrar as características transversais em modelos de agentes.

Para apresentar e ilustrar as definições das abstrações e relacionamentos propostos, será feito uso do estudo de caso utilizado por GARCIA (2004a), um sistema de gerenciamento de conferências chamado Expert Committee (EC). Desta forma, a Seção 4.1 descreve sucintamente o EC. A Seção 4.2 apresenta o framework proposto para separar, compor e integrar as características transversais na modelagem de SMAs. Já a Seção 4.3 propõe-se a evidenciar o diferencial trazido pela modularização das características transversais, discutindo o que é agregado à modelagem de SMAs sem aspectos. A Seção 4.4, por sua vez, discute como o ACROSS atende às duas propriedades fundamentais da orientação a aspectos, que são a transparência e a quantificação. Considerações sobre as potencialidades do ACROSS, vislumbrando quais seriam os próximos passos e como eles contribuiriam para integrar o ACROSS a todo o ciclo de desenvolvimento de software aparecem na Seção 4.5, enquanto a Seção 4.6 avalia o ACROSS segundo os critérios sugeridos por JUNIOR (2005) para qualificar um framework para modelagem de características. Por fim, a Seção 4.7 traz os comentários finais a respeito do trabalho desenvolvido.

4.1 O EXPERT COMMITTEE

O Expert Committee é um SMA para Internet que visa gerenciar o envio de trabalhos e o processo de seleção para uma conferência, sendo derivado de um estudo de caso homônimo realizado pelo Laboratório de Engenharia de Software da Pontifícia Universidade Católica do Rio de Janeiro (LES/ PUC-Rio) (GARCIA, 2004a). Além de GARCIA (2004a), PERES (2005) e OLIVEIRA (2006), entre outros, também utilizam este exemplo e o descrevem. Este sistema é caracterizado pela presença de várias características de

agência, como interatividade, autonomia, adaptação, aprendizado e mobilidade.

Os agentes do EC são assistentes de software que representam as pessoas que atuam no ambiente de submissão e revisão de trabalhos para uma conferência desempenhando os diversos papéis envolvidos, a saber: (i) autores de artigos, (ii) revisores, (iii) membros do Comitê de Programa (CP), (iv) *chair* do evento e (v) coordenador geral. Para simplificar o texto, em vez de agente autor, por exemplo, o agente será referenciado como apenas autor. Agora, se fôr necessário referenciar a pessoa do autor, por exemplo, será empregada a terminologia usuário autor.

Os autores, basicamente, ficam encarregados de submeter os artigos escritos pelo usuário autor para a conferência. O coordenador geral é responsável pelo recrutamento de revisores. A missão do chair é a mais complexa de todas, pois ele, além de receber os artigos submetidos, deve elaborar as propostas de revisão, ou seja, convidar revisores para avaliar os artigos. Para isso, é necessário que sejam respeitadas algumas restrições. Um artigo tem que ser examinado por pelo menos três revisores cuja áreas de interesses enquadrem-se no tema do artigo. Por questões de imparcialidade e transparência, nenhum dos revisores pode ser afiliado à mesma instituição de qualquer um dos autores do artigo. Para não sobrecarregar os revisores, estes não podem avaliar mais do que três artigos. Além disso, o chair precisa encaminhar os resultados da revisão para os membros do Comitê de Programa para que seja definido o programa final da conferência.

Ao receber convite para rever algum artigo, o revisor deve julgar se é conveniente para o usuário revisor aceitar tal tarefa, baseando-se em sua agenda e áreas de interesse. Os membros do CP atuarão apenas no recebimento dos resultados das revisões para apreciação de seus respectivos usuários e envio do parecer final ao chair.

A Figura 4.1 sintetiza a atuação de agentes e usuários humanos no Expert Committee e foi retirada dos slides da apresentação do artigo PERES (2005) no *I Workshop on Software Engineering for Agent-oriented Systems* (SEAS 2005), realizada pela autora desta dissertação na cidade de Uberlândia, Brasil, em outubro de 2005.

4.2 O FRAMEWORK ACROSS

Antes de propor qualquer solução de modelagem para modularizar as características transversais em SMAs, deve-se investigar como um aspecto seria estruturado no paradigma de agentes de software. Na Seção 2.2, ao discutir o princípio da separação de características, mostrou-se que uma característica de um sistema convenientemente modularizada ainda assim fará parte deste sistema e, por conta disso, respeita seu paradigma

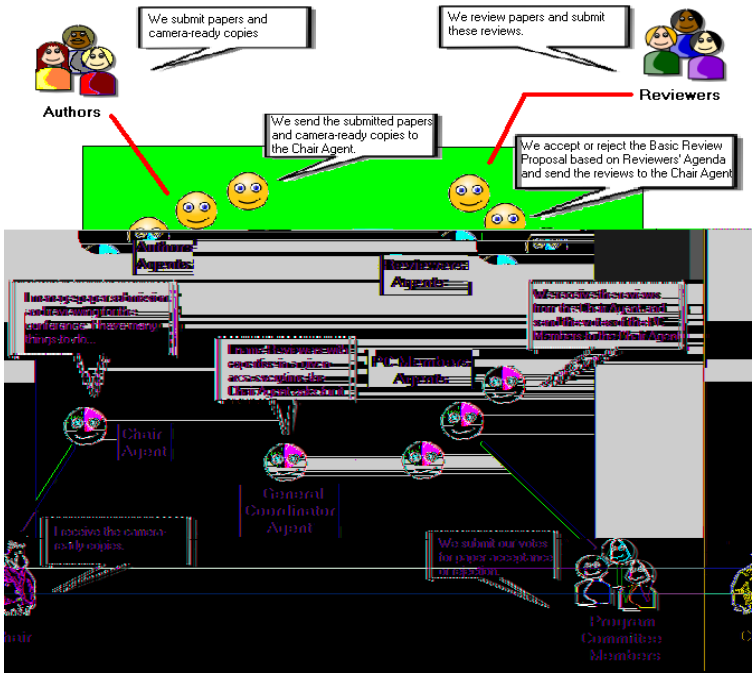

FIG. 4.1: Agentes e usuários humanos no Expert Committee (em inglês).

de desenvolvimento, o que indica que para descrever tal módulo é preciso empregar as mesmas abstrações que caracterizam o resto do sistema (GARCIA, 2006b) (GARCIA, 2006a).

Assim, aspectos em SMAs devem contar com as mesmas abstrações convencionais de um agente, ou seja, terão objetivos, planos e ações associados a eles, além de, obviamente, necessitarem da caracterização de seus pontos de incidência pelo estabelecimento de *join points* e *pointcuts*, bem como referenciar o momento correto disto ocorrer com a definição do *advice*. Enquanto no paradigma OO encarava-se o *advice* como uma construção semelhante ao método (KULESZA, 2005), em SMAs pode-se imaginar que sua construção trará os planos e as ações associados ao aspecto em questão.

Na Engenharia de Software, um framework é uma estrutura de suporte passível de ser estendida e convenientemente instanciada de modo a permitir o desenvolvimento de aplicações específicas (THAYER, 2003). Assim, um framework de modelagem deve oferecer as abstrações e relacionamentos necessários para representar os requisitos elicitados do sistema. Desta forma, esta seção apresentará as abstrações (Seção 4.2.1) e relacionamentos (Seção 4.2.2) que compõem o framework ACROSS para separação, composição e integração de características transversais na modelagem de SMAs. Além disso, a Seção 4.2.3 traz a descrição baseada em XML das abstrações e relacionamentos propostos.

4.2.1 ABSTRAÇÕES DO ACROSS

Conforme foi discutido anteriormente, para modularizar as características transversais é necessário empregar as mesmas abstrações do paradigma em questão. Como as abstrações principais dos SMAs são agentes, objetivos, planos e ações deve-se buscar abstrações análogas para representar as características transversais neste paradigma.

Agentes e aspectos são entidades de primeira-classe, porém, enquanto agentes modularizam características funcionais dos SMAs, os aspectos estão atrelados às características transversais, que podem, por exemplo, estar ligadas às propriedades dos agentes ou representar alguma qualidade desejável do sistema. GARCIA (2004a), GARCIA (2005), GARCIA (2006c) e LOUGHRAN (2006) trazem um catálogo de características transversais típicas dos SMAs, que incluem propriedades tais como autonomia, interatividade, adaptabilidade, aprendizado, mobilidade, sensibilidade ao contexto, entre outras [2]. No exemplo do EC conforme estabelecido por GARCIA (2004a), tem-se como agentes o chair, os autores, os membros do Comitê de Programa, os revisores e o coordenador geral e como aspectos a interatividade, a adaptabilidade, a autonomia, o aprendizado e a mobilidade.

Agentes visam atingir objetivos, assim, dentro do raciocínio que vem sendo empregado, é razoável afirmar que aspectos também intencionam atingir seus objetivos. Desta forma, o ACROSS propõe uma taxonomia que divide os objetivos em duas categorias: objetivos de agente e objetivos de aspecto. Enquanto os primeiros estão relacionados a características funcionais, ou seja, serviços oferecidos pelos agentes, os últimos dizem respeito tanto à qualidades dos sistemas (características não-funcionais) quanto a mecanismos reutilizáveis associados às propriedades de agência e outras características dos SMAs.

Assim, no exemplo do EC, pode-se estabelecer como objetivos do chair o armazenamento de artigos submetidos, a elaboração da proposta de revisão e o levantamento do conflitos nas revisões para encaminhamento aos membros do comitê de programa. Já o aspecto interatividade, cujo objetivo central é promover a comunicação entre agentes, deve perseguir metas como enviar e receber mensagens.

Os planos de ação, ou simplesmente, planos, podem ser considerados um *composite* (GAMMA, 1995) de outros planos menores e de ações atômicas. Cada objetivo a ser cumprido pelo agente ou aspecto deve ter associado a si ao menos um plano encarregado de concretizá-lo. Os planos definem um conjunto de ações que podem ser arbitradas

[2] Na presente dissertação, a identificação de aspectos na modelagem de SMAs baseou-se nos exemplos catalogados consagrados pela literatura, como as características transversais típicas dos SMAs e também de exemplos atrelados a requisitos não-funcionais como os que podem ser vistos em CHUNG (1999).

estática ou dinamicamente com o intuito de promover as mudanças de estado necessárias para a satisfação do objetivo em foco. Ações (tarefas) são computações que promovem as mudanças de estados dos agente, representando, assim, as abstrações de mais baixo nível do paradigma multiagentes.

Retornando ao EC, associado ao objetivo do agente chair que visa a elaboração da proposta de revisão, pode-se associar vários planos como, por exemplo, um plano para selecionar os revisores aptos a revisar um determinado artigo, um plano para pedir novos revisores ao coordenador geral e, ainda, um plano para montar o documento de proposta de revisão. No plano para selecionar os revisores, pode-se pensar em incluir ações como o descarte de revisores que não compartilhem a mesma área de interesse do artigo, o descarte de revisores da mesma afiliação que os autores, o descarte dos revisores que já revisam três artigos e a montagem de uma lista com os revisores restantes. Para atingir o objetivo de enviar mensagens, deve-se executar um plano que promova a identificação dos destinatários, a elaboração do conteúdo da mensagem e a atualização da caixa de saída.

Desta forma, o ACROSS trabalha com as seguintes abstrações: *agente* (*agent*), *aspecto* (*aspect*), *objetivo do agente* (*agent goal*), *objetivo do aspecto* (*aspect goal*), *plano* (*plan*) e *ação* (*action*). Forma-se, então uma hierarquia de abstrações para modularizar as características dos agentes e outra para as características transversais, conforme pode ser percebido na Figura 4.2. Com estas abstrações é possível realizar a separação das as características transversais em modelos de agentes.

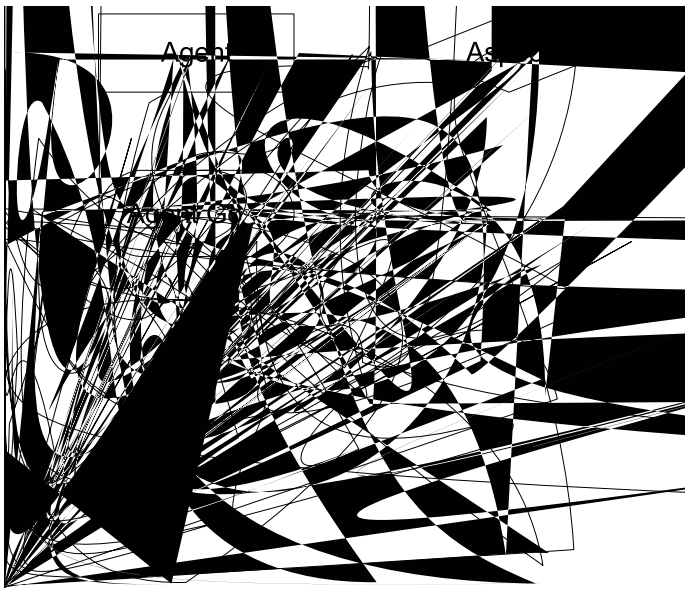

FIG. 4.2: Hierarquia de abstrações do ACROSS para agentes e aspectos.

Pela Figura 4.2, também nota-se que foi arbitrada uma notação específica para re-

presentar agentes, aspectos, objetivos de agente e objetivos de aspecto, planos e ações. O uso do losango para os aspectos foi extraído trabalhos como CHAVEZ (2004), já o octógono utilizado para os objetivos de agente e o hexágono para representar as ações foram inspirados na notação do V-Graph (YU, 2004).

4.2.2 RELACIONAMENTOS

As abstrações propostas na seção anterior promovem a separação das características transversais em objetivos, planos e ações, mas não são capazes de mostrar como tais características devem ser compostas com as funcionalidades dos agentes e como resulta o modelo integrado. É preciso, pois, definir relacionamentos que caracterizem a composição entre agentes e aspectos, objetivos de agente e objetivos de aspecto, planos e ações de agente e planos e ações de aspectos. Além disso, é vital que sejam estabelecidos relacionamentos que associem agentes e aspectos aos seus objetivos e estes aos respectivos planos, bem como explicitar as ações que compõem cada plano.

Assim, os relacionamentos entre as abstrações propostas podem ser *verticais* ou *horizontais*. Os relacionamentos verticais ocorrem entre abstrações de níveis diferentes e são tipicamente decomposições que podem ser expressas em termos de árvores AND/OR/XOR. Já os relacionamentos horizontais (ou laterais) envolvem abstrações de mesmo nível hierárquico, uma ligada ao agente e outra associada ao aspecto, caracterizando a composição entre eles.

4.2.2.1 RELACIONAMENTOS VERTICAIS

Abordagens dirigidas por metas colocam que os objetivos devem ser decompostos e refinados em termos de árvores AND/OR (MYLOPOULOS, 1992). De forma análoga, é necessário associar um agente ou aspecto aos seus objetivos, decompor estes objetivos em subobjetivos, atribuir planos a serem executados visando o cumprimento de cada objetivo-folha e decompor os planos em subplanos e ações. Árvores AND/OR são uma boa forma de realizar estas tarefas, sobretudo se acrescidas do operador XOR (OU exclusivo). Com isso, por exemplo, pode-se saber não só que determinados planos estão associados a um objetivo, como também saber se, para o objetivo poder ser atingido, deve-se executar somente um dos planos, pelo menos um deles ou simplesmente todos. O atendimento dos objetivos segue a lógica da sua árvore de decomposição. Enquanto os objetivos-folha dependem do sucesso na execução dos planos para ser atendidos, os objetivos raízes têm o seu cumprimento atrelado ao atendimento de seus subobjetivos, descendo até os

objetivos-folha.

No framework ACROSS, os relacionamentos verticais caracterizam a decomposição de uma abstração em outras de nível hierárquico inferior ou de mesmo nível sob a forma de uma estrutura de árvore AND/OR/XOR. Enquanto o operador AND carrega o sentido de "todos", o operador OR significa "pelos menos um" e o XOR exprime "somente um". A Tabela 4.1 traz, para cada relacionamento vertical possível, o significado do operador, definindo a forma de leitura destes relacionamentos no ACROSS. É importante ressaltar que o padrão *composite* (GAMMA, 1995), que pode ser aplicado a objetivos e planos, parece quebrar a verticalidade destes relacionamentos. No entanto, os objetivos e planos relacionados estão em uma hierarquia de decomposição, o que faz com que não estejam no mesmo nível de abstração.

TAB. 4.1: Definição dos relacionamentos verticais do ACROSS.

A Figura 4.3, por sua vez, exibe a notação gráfica destes relacionamentos no ACROSS, que será utilizada na construção dos modelos.

Para ilustrar a construção destas árvores, a Figura 4.4 mostra a decomposição de

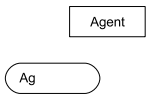

FIG. 4.3: Representação gráfica dos possíveis relacionamentos verticais entre as abstrações no ACROSS.

objetivos do chair no exemplo do EC. A Figura 4.5 (mais adiante) traz a decomposição dos planos associados ao objetivo "elaborar proposta de revisão". Da mesma forma, é feita a decomposição dos objetivos do aspecto interatividade, bem como dos planos e ações associados aos objetivos "montar mensagem" e "gerenciar caixa de saída" no contexto do envio de mensagens, conforme mostram as Figuras 4.6 e 4.7.

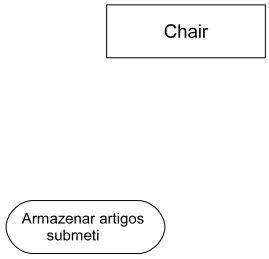

FIG. 4.4: Representação da árvore de objetivos do agente chair e do aspecto interatividade.

4.2.2.2 RELACIONAMENTOS HORIZONTAIS: COMPOSIÇÃO DE OBJETIVOS, DE PLANOS E DE AÇÕES

Enquanto os relacionamentos verticais se ocupam de modelar isoladamente os agentes e aspectos, os relacionamentos horizontais denotam como os aspectos interagem com os agentes, ou seja, quais são os pontos de combinação existentes. Nesta proposta, os agentes e aspectos estão sendo modelados em dois níveis principais: nível de objetivos e nível de planos e ações. Assim, cabe aos relacionamentos horizontais definir mecanismos de composição entre objetivos, entre planos e entre ações. Em suma, tais relacionamentos visam determinar (i) quais os agentes que são atravessados por cada aspecto identificado, (ii) como um objetivo de aspecto afeta um objetivo de agente e (iii) como caracterizar o cruzamentos entre os planos e ações do agente e os planos e ações do aspecto.

Para modularizar características transversais é preciso ter um conhecimento global do sistema (KICZALES, 2005). A presença de características transversais no nível de objetivos provê o conhecimento global necessário para tornar uma funcionalidade/requisito mais explícito, favorecendo o raciocínio modular sobre a transversalidade no nível das ações. Desta forma, esta seção traz a proposta do ACROSS para compor objetivos de agentes e aspectos, trazendo à tona a forma como os objetivos dos aspectos vão afetar os objetivos do sistema.

FIG. 4.5: Representação dos planos associados ao objetivo "elaborar proposta de revisão" a ser executado pelo chair.

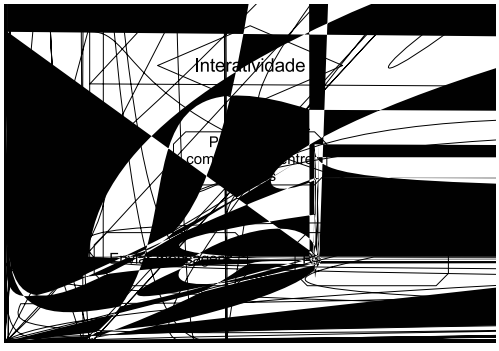

FIG. 4.6: Representação da árvore de objetivos do aspecto interatividade.

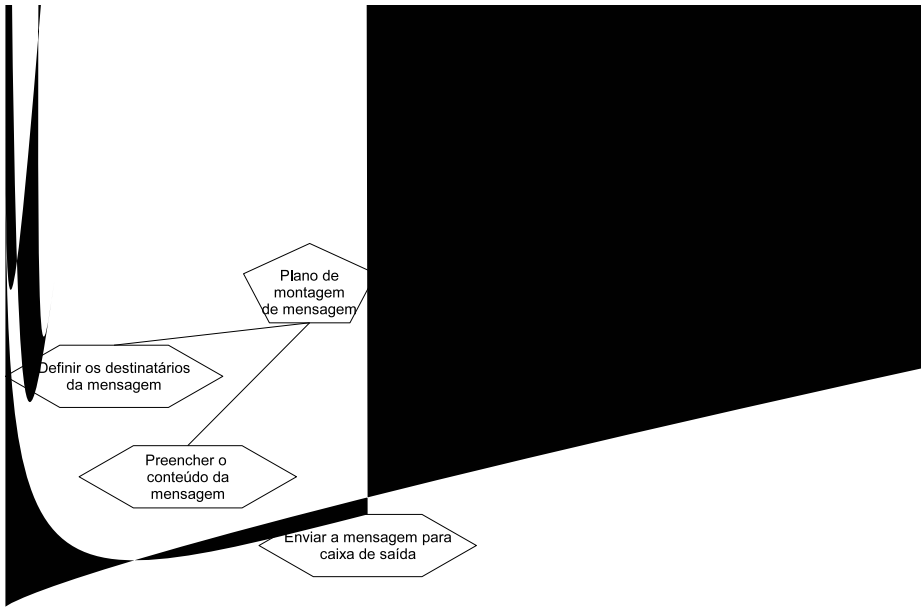

FIG. 4.7: Representação dos planos associados ao objetivo "enviar mensagem" do aspecto interatividade

Um agente é atravessado por um aspecto se pelo menos um de seus objetivos é afetado por objetivos de aspecto. CHAVEZ (2004) introduz um conceito muito útil para caracterização da composição dos aspectos com as entidades-base: *interface de crosscutting* (ou interface transversal). Uma interface de crosscutting (IC) é um conjunto características transversais com um nome associado que descreve o comportamento transversal de um aspecto. No ACROSS, os objetivos de aspecto podem ser considerados os elementos de crosscutting que constituem as ICs, já que elas definem como um agente é atravessado por um aspecto, ou seja, como é o comportamento transversal do aspecto.

Os objetivos de aspecto podem influenciar positiva ou negativamente os objetivos de agente. A análise de correlação ajuda no descobrimento de relacionamentos laterais positivos e negativos entre objetivos, permitindo melhor compreensão do *rationale* dos requisitos do sistema. Aproveitando-se da nomenclatura dos relacionamentos de correlação entre metas e softmetas do V-Graph, o ACROSS redefiniu seu significado e modo de aplicação, de maneira a formar um conjunto completo de graus de influência do cumprimento de um objetivo de aspecto no cumprimento de um objetivo de agente. Desta forma, estabelece-se os seguintes tipos de relacionamentos horizontais entre um objetivo de aspecto e um objetivo de agente:

- *Make*: O objetivo de agente pode ser atingido se o objetivo de aspecto é atingido;
- *Help*: O objetivo de agente pode ser melhor atingido (melhor performance,

confiabilidade, precisão, etc) se o objetivo de aspecto é atingido;

- *Hurt*: O cumprimento do objetivo de agente pode ser comprometido se o objetivo de aspecto for cumprido;
- *Break*: O objetivo de agente não pode ser atingido se o objetivo de aspecto for cumprido;
- *Unknown*: Não há relação entre o cumprimento do objetivo de aspecto e do objetivo de agente: não favorece, nem prejudica.

A Figura 4.8 apresenta a representação gráfica de tais relacionamentos, que constitui de uma seta do objetivo de aspecto para o objetivo de agente com a inscrição do relacionamento correspondente acima da seta, indicando como o objetivo do aspecto interage com o objetivo do agente.

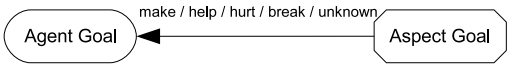

FIG. 4.8: Representação gráfica dos relacionamentos horizontais para composição de objetivos entre agentes e aspectos.

Para ilustrar o uso destes relacionamentos, a Figura 4.9 traz um exemplo do EC, mostrando como os objetivos dos aspectos interatividade e aprendizado podem influenciar os objetivos do agente chair. Sem o recebimento de mensagens, o chair é incapaz de armazenar os artigos submetidos, pois, simplesmente, eles não chegariam até ele. Da mesma forma, sem conseguir enviar o documento de proposta de revisão aos revisores escalados e receber a resposta dos revisores, o referido objetivo não está, de fato, concluído. Para levantar as revisões conflituosas, é imperioso que o chair seja capaz de receber as revisões de cada artigo e de enviar o resultado da análise final das revisões aos membros do Comitê de Programa para que eles possam estabelecer a agenda final da conferência. Por outro lado, conhecer as novas áreas de interesse que os revisores estão trabalhando e ter uma noção daqueles que estão habitualmente declinando do convite por falta de tempo na agenda ajuda a otimizar o processo de fechamento da alocação de revisores, uma vez que permite uma seleção mais eficiente dos revisores, evitando um grande número de recusas e a conseq ente necessidade de reestruturar a proposta.

Enquanto a composição de objetivos vai contribuir para ampliar o entendimento global

FIG. 4.9: Representação gráfica dos relacionamentos horizontais para composição de objetivos entre agentes e aspectos.

definir quais são os pontos de combinação (ou *join points*) e como os aspectos irão incidir sobre eles.

O AspectJ caracteriza a transversalidade através da definição de *join point*, *pointcut* e *advice*. O *advice* indica se um dado comportamento do aspecto deve ser executado antes, durante ou depois (*before*, *around*, *after*, respectivamente) de um *join point* na operação-base, parte integrando de *pointcut* associado ao *advice*. A abordagem proposta nesta dissertação também incorpora estes conceitos, pois estabelece quais planos ou ações dos aspectos vão interagir com os planos ou ações dos agentes e define com se dará esta interação.

CHAVEZ (2004), ao definir a estrutura das ICs, coloca que elas podem oferecer suporte para especificar o *crosscutting* estático e dinâmico. *Additions* (adições) definem uma nova característica comportamental estática introduzida por um aspecto em uma entidade-base. *Refinements* (refinamentos) e *redefinitions* (redefinições) lidam com o *crosscutting* dinâmico. Enquanto o primeiro indica que o comportamento do aspecto deve ser combinado com o comportamento da entidade-base, o outro indica que o comportamento do aspecto irá substitui-lo.

Estabelecer que um objetivo de aspecto afeta um objetivo de agente significa dizer que a execução de algum plano associado ao aspecto irá atravessar a execução de um plano associado ao agente. Isto faz com que duas situações sejam possíveis: (i) o plano associado ao aspecto deve ser inteiramente executado antes ou depois do plano associado ao agente ou (ii) o plano associado ao aspecto deve ser fundido ao plano associado ao agente.

Procurando estender os conceitos de IC, o ACROSS define dois tipos de relacionamentos entre planos: o *add* e o *merge*. O relacionamento *add* é subdividido em *add before* e *add after* para denotar que o plano do aspecto deve ser inteiramente executado antes ou depois, respectivamente, do plano do agente, sendo, portanto, adicionado a este. Cabe ressaltar que, uma vez que os agentes podem executar várias *threads*, nada assegura que um plano de um objetivo de um aspecto será executado imediatamente antes ou depois de um plano ou objetivo de um agente. Já o relacionamento *merge* indica que deve ocorrer a fusão entre os dois planos, sem porém especificar como ela deve acontecer. A Figura 4.10 traz a representação destes relacionamentos.

Para esclarecer a questão deixada em aberto pelo relacionamento *merge* é preciso descer o nível de abstração para ações para descrever como realmente ocorre o *crosscutting*. Neste caso, duas situações básicas podem ocorrer. A primeira é que ações associadas ao aspectos substituam ações associadas ao agente, o que caracteriza o relacionamento *redefine* entre ações. A outra situação é o caso em que ações associadas ao aspecto devem

combinar-se a ações associadas ao agente, refinado-as. Essas combinações podem ocorrer antes, durante ou depois da execução da ação do agente, o que resulta nos relacionamentos *refine before*, *refine around* e *refine after*, respectivamente, conforme também mostra a Figura 4.10.

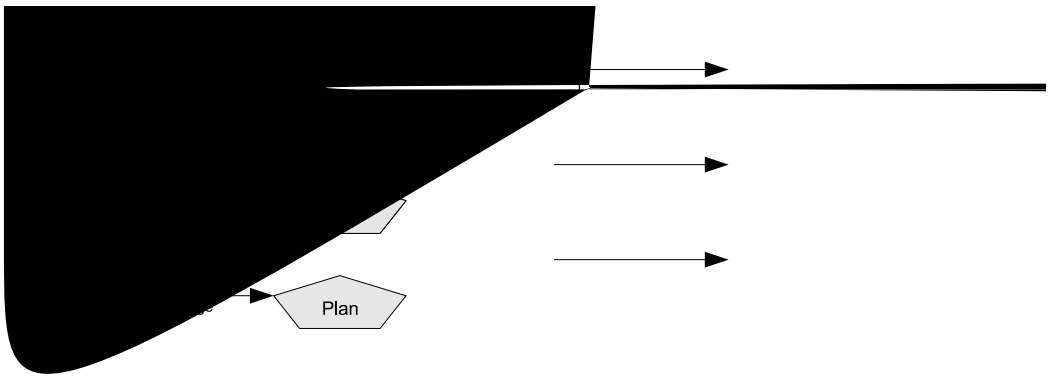

FIG. 4.10: Representação gráfica dos relacionamentos horizontais para composição de planos e de ações entre agentes e aspectos.

Para ilustrar a aplicação destes relacionamentos, a Figura 4.11 apresenta a composição de planos e ações entre os aspectos interatividade e apredizado e o agente chair. Por ela, percebe-se que o plano associado ao recebimento de mensagens precisa ser inteiramente executado antes que o agente manipule a informação recebida, o que denota o relacionamento *add before*. De forma análoga, depois que foi construída toda informação que precisa chegar ao conhecimento de outros agentes, é preciso executar todo o plano de envio de mensagens, configurando o relacionamento *add after*. O aprendizado sobre as novas áreas de interesse e disponibilidade de tempo dos revisores ocorre durante o plano de reestruturação da proposta de revisão, afinal decorre da análise das respostas dos revisores aos convites, uma vez que podem alegar tanto que estão com a agenda cheia por um determinado tempo quanto que não estão interessados na área relacionada ao artigo, informando, assim, quais são seus atuais interesses de pesquisa e atuação. Esta argumentação, portanto, explica o emprego do relacionamento *refine after*. Mais exemplos poderão ser vistos no Capítulo 5, já que este trará um estudo de caso mais completo.

4.2.3 DESCRIÇÃO DO MODELO BASEADA EM XML

O XML (*eXtensible Markup Language*)(BRAY, 2004) é uma linguagem textual de descrição de dados em que *tags* precisam ser definidas pelo usuário, sendo muito usada para troca de dados por ser independente de plataforma. A exploração deste padrão para

FIG. 4.11: Composição de planos e ações entre os aspectos interatividade e apredizado e o agente chair.

descrição e troca de dados, sobretudo em função da Internet, vem sendo bem vasta.

Construir uma representação das abstrações e relacionamentos do framework ACROSS baseada em XML pode ser visto como um passo importante no sentido de desenvolver uma ferramenta gráfica que dê suporte ao framework proposto. Descrever os elementos que compõem o ACROSS em termos de uma estrutura em árvore composta por *tags* fixas e customizáveis e poder arquivar tais descrições em um formato conhecido traz diversos benefícios, favorecendo o intercâmbio e o reuso de modelos. A ferramenta gráfica, além de exportar os modelos para formatos gráficos tradicionais (.bmp, .jpg, etc), os armazenariam como arquivos XML, facilitando sua manipulação. Assim, seria possível "ver" os modelos de diversas formas: ver o todo ou visualizar somente a composição de objetivos ou apenas a modelagem em todos os níveis de um dado aspecto e um dado agente e assim por diante.

A representação baseada em XML do ACROSS deve contar com duas grandes seções: cabeçalho (*head*) e modelo (*model*). A primeira apresenta informações gerais sobre o projeto como nome, sigla, prazos, gerente ou quaisquer outras que puderem ser de interesse do usuário. A proposta é justamente que esta parte seja composta de *tags* customizáveis, dando ao usuário total liberdade de registrar as informações gerenciais convenientes do projeto. A segunda parte do documento, no entanto, possui uma estrutura de *tags* bem rígida, uma vez que é encarregada de descrever os elementos do ACROSS. A Figura 4.12 mostra a estrutura geral do documento.

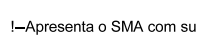

FIG. 4.12: Estrutura geral do documento de descrição do modelo baseado em XML.

A seção de modelo traz a modelagem de agentes e aspectos em termos de seus objetivos,

planos e ações, bem como a modelagem do crosscutting, ou seja, definição dos mecanismos de composição de objetivos e de planos e ações. Desta forma, a seção está dividida em modelagem do agente (<*agent-modeling*>), modelagem do aspecto (<*aspect-modeling*>) e modelagem do crosscutting (<*crosscutting-modeling*>).

Para modelar os agentes e aspectos, basta preencher corretamente os campos definidos pelas *tags* que caracterizam a entidade (<*agent*> ou <*aspect*>), seus objetivos (<*agent-goal*> ou <*aspect-goal*>), os planos (<*action-plan*>) atrelados a cada objetivo e as ações (<*atomic-action*>) atreladas a cada plano. Além disso, deve-se definir a árvore de objetivos do agente ou aspecto, bem como a árvore de planos atrelados aos objetivos e a árvore de subplanos e ações que compõem um plano. Para tal, deve-se usar as *tags* <*AND*>, <*OR*> ou <*XOR*>. Em resumo, deve-se instanciar as abstrações do ACROSS e descrever os relacionamentos verticais entre elas. A Figura 4.13 apresenta a estrutura da seção de modelagem do agente, enquanto a Figura 4.14 mostra a totalmente análoga estrutura da seção de modelagem do aspecto.

A modelagem de crosscutting trata da descrição dos relacionamentos horizontais entre os agentes e aspectos (<*crosscutting*>), objetivos de agente e objetivos de aspecto (<*goal-correlation*>), planos e ações de agentes e planos e ações de aspectos (<*crosscutting-point*>). Assim, será possível constatar (i) quais aspectos atravessam quais agentes; (ii) quais os objetivos de agentes são afetados pelos objetivos de aspecto e qual relacionamento de correlação (*make*, *help*, *hurt*, *break* ou *unknown*) caracteriza tal influência e (iii) como planos e ações do aspecto atravessam os planos e ações dos aspectos através da atribuição do operador conveniente (*add before*, *add after*, *refine before*, *refine around*, *refine after* ou *redefine*). A Figura 4.15 apresenta a estrutura desta seção do documento.

Assim, é possível perceber que a representação XML dos elementos do ACROSS é capaz de mapear todas as informações contidas na notação gráfica, uma vez que especifica as abstrações e relacionamentos propostos. Como o XML é uma linguagem de marcação usada para representação e descrição de informações (BRAY, 2004), a estrutura proposta favorece a busca e recuperação da informação pelo emprego de *tags* expressivas que façam as marcações (referenciações) convenientes.

4.3 COMPARAÇÃO COM A MODELAGEM SEM ASPECTOS

Para melhor entender os benefícios da modularização de características transversais na modelagem de SMAs, basta fazer algumas reflexões a respeito do exemplo do EC. Tome-se, por exemplo, o aspecto interatividade atravessando o agente chair. Este as-

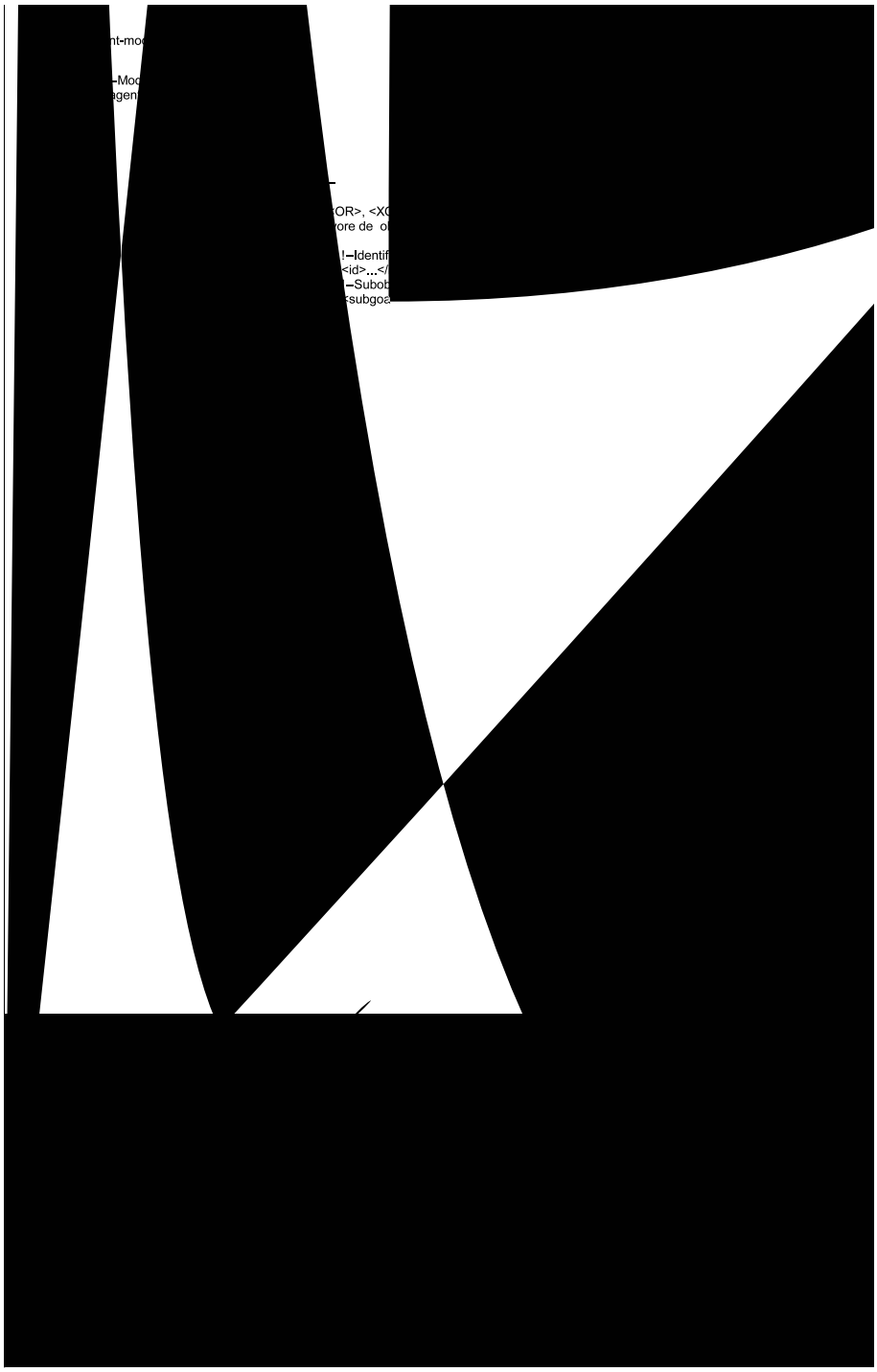

FIG. 4.13: Estrutura da seção de modelagem dos agentes, com seus objetivos, planos e ações.

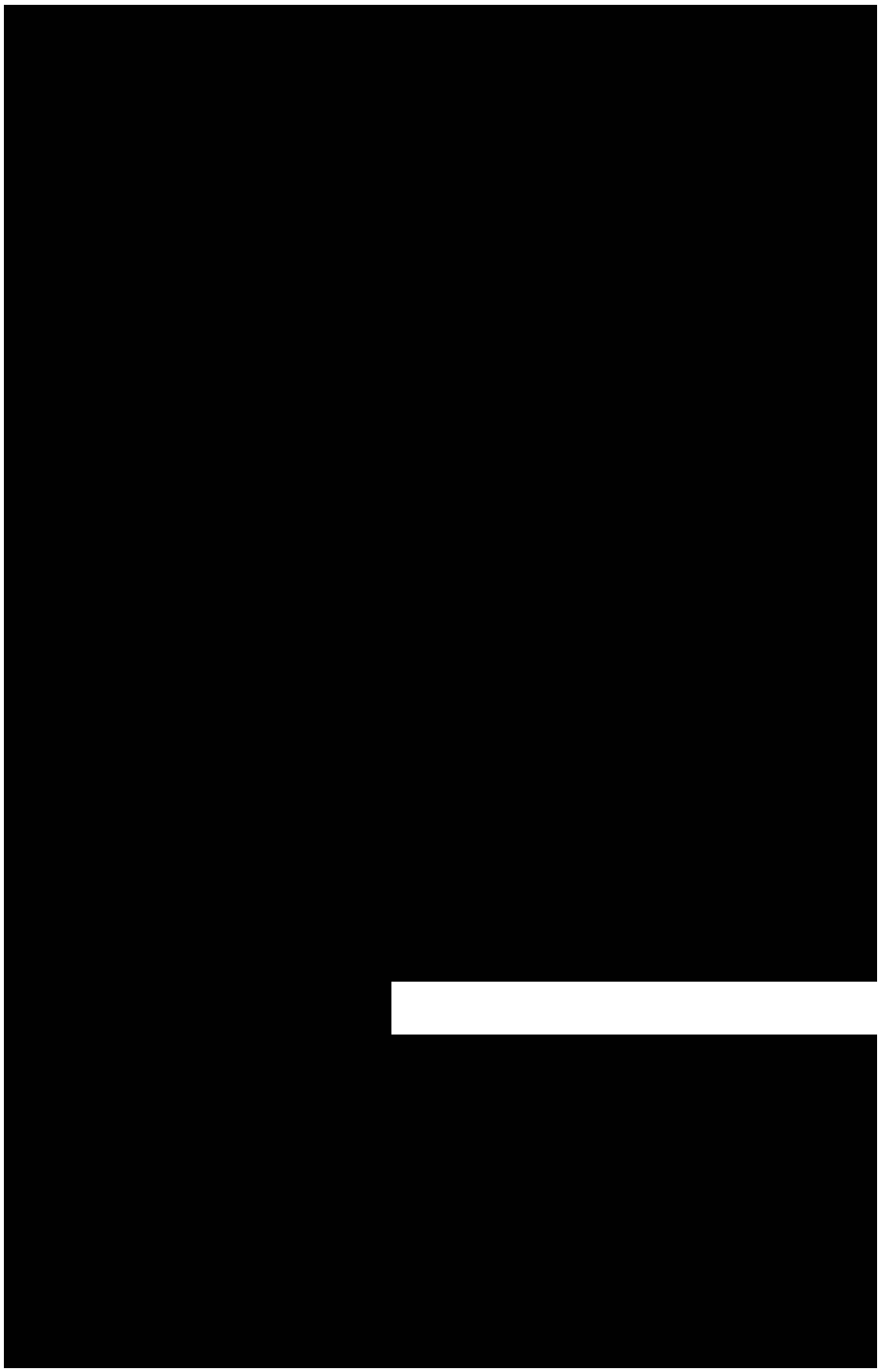

FIG. 4.14: Estrutura da seção de modelagem dos aspectos, com seus objetivos, planos e ações.

FIG. 4.15: Estrutura da seção de modelagem do crosscutting, mostrando a composição de objetivos e o crosscutting no nível de planos e ações.

pecto, basicamente, define mecanismos para tratar o envio e recebimento de mensagens. Assim, sem haver a modularização dos aspectos, estes mecanismos deveriam aparecer na modelagem do agente chair sempre que fosse preciso promover a comunicação com outros agentes. Assim, as ações associadas ao recebimento de mensagens apareceriam replicadas quando o chair precisasse receber os artigos submetidos para posterior armazenamento, quando o chair precisasse receber novos revisores contratados, quando o chair precisasse receber as revisões em si e assim por diante. Isto, sem mencionar os demais agentes, uma vez que todos precisam de mecanismos de interação. Uma das formas para modelar este caso seria a incorporação de novos objetivos ligados à interatividade na modelagem do agente chair, como sugere a Figura 4.16. Visivelmente, a solução é menos legível e reutilizável que a proposta nesta dissertação.

FIG. 4.16: Objetivos do agente chair sem modularização do aspecto interatividade.

Se forem examinados os outros exemplos, ligados aos demais aspectos presentes na modelagem do EC, serão tiradas as mesmas conclusões, pois os planos e ações ou mesmo os objetivos associados aos aspectos terão que aparecer em algum momento atrelados à modelagem dos objetivos, planos e ações dos agentes. Separar, compor e integrar estes aspectos à modelagem do SMA torna os modelos mais legíveis e, sobretudo, conduz a uma implementação que considere a modularização das características transversais.

4.4 TRANSPARÊNCIA E QUANTIFICAÇÃO

Por tratar-se de um framework orientado a aspectos, é importante avaliar como o ACROSS se posiciona diante das propriedades básicas deste paradigma. Transparên-

cia e quantificação (ver Seção 2.2.1) são duas propriedades fundamentais dos aspectos (FILMAN, 2005) (CHAVEZ, 2004). O ACROSS também adota estas propriedades como essenciais.

Isolados, os modelos de agente, com seus objetivos, planos e ações não sinalizam quais aspectos irão atravessá-los e como isso ocorrerá, afinal isso fica a cargo das setas que representam os relacionamentos horizontais. Garante-se, portanto, que a propriedade da transparância é observada.

Mecanismos de quantificação definem as condições que devem ser alcançadas para que um aspecto incida sobre a entidade-base (FILMAN, 2005). Ao explicitar sobre quais objetivos, planos e ações do agente incidem determinados objetivos, planos e ações do aspecto e caracterizar a natureza do relacionamento presente, o ACROSS assegura a observância desta propriedade. Desta forma, a leitura dos modelos do ACROSS conduz a sentenças do tipo *"no agente A, sempre que o objetivo B fôr perseguido, ele será influenciado pelo objetivo C do aspecto D segundo o relacionamento de correlação make (help/hurt/break/unknown)"* ou *"no agente A, sempre que o plano B fôr executado, o plano C do aspecto D será executado conforme o relacionamento add before (add after/merge)"* ou ainda *"no agente A, sempre que a ação B fôr executada, a ação C do aspecto D será executada conforme o relacionamento refine before (refine around/refine after/redefine)"*.

4.5 POTENCIALIDADES DO ACROSS

Não basta avaliar somente o estado atual do ACROSS, é preciso pensar também nas portas por ele abertas. O desenvolvimento de ferramenta gráfica e o estabelecimento de diretrizes para mapeamento dos modelos em código podem fazer com que o ACROSS se torne uma ferramenta interessante para garantir a rastreabilidade dos requisitos. Poderia-se imaginar, por exemplo, o uso integrado da descrição XML dos modelos gerados pelo ACROSS e de ferramentas de documentação de código baseadas em hipertexto, como o Javadoc (SDN, 2006) para atrelar as implementações de agentes, aspectos, objetivos, planos e ações aos respectivos modelos, bem como a geração automática de código a partir do modelo.

Para gerar código a partir dos modelos seria necessário definir as plataformas de implementação a serem utilizadas. Atualmente, poderia-se raciocinar com o framework JADE para SMAs e o AspectJ, embora as limitações de ambos para refletir as abstrações do paradigma multiagentes precisassem ser contornadas, pois são fundamentados no paradigma

orientado a objetos. De qualquer forma, os agentes modelados no ACROSS poderiam ser implementados como classes que estenderiam jade.core.Agent (BELLIFEMINE, 2005), de modo que os planos e ações seriam convenientemente implementados a partir de *behaviours* do JADE (extensões de jade.core.behaviour.*). Já os aspectos da modelagem seriam *aspects* do AspectJ e os planos e ações dos aspectos seriam convenientemente implementados a partir de *advices*. Os objetivos, tanto de agentes quanto de aspectos, a priori, seriam de entendimento subjetivo, de modo que o seu cumprimento fosse atestado pelo fluxo de execução dos planos e ações implementadas ou até mesmo sinalizado por *flags*. O ideal, no entanto, seria conceber uma plataforma que integrasse aspectos e agentes, usando as abstrações propostas pelo ACROSS, pois permitiria um mapeamento mais direto.

As Tabelas 4.2, 4.3, 4.4 e 4.5 apresentam, respectivamente, uma sugestão de mapeamento das abstrações de agentes, aspectos, bem como relacionamentos verticais e horizontais do ACROSS utilizando o JADE conjugado com o AspectJ. O uso de variáveis de controle (ou *flags*) serve tanto para sinalizar que um objetivo foi cumprido quanto para indicar o próximo objetivo a ser perseguido, plano ou ação a ser executado. Conjugado a tais variáveis emprega-se *statements* condicionais (*if..then*,*switch..case*, etc).

TAB. 4.2: Mapeamento das abstrações dos agentes no JADE

ABSTRAÇÃO DO AGENTE	IMPLEMENTAÇÃO NO JADE
Agent	classe estendendo jade.core.Agent
Agent Goal	variável de controle
Plan	método (chama os métodos que implementam as ações)
Action	método

TAB. 4.3: Mapeamento das abstrações dos aspectos no AspectJ

ABSTRAÇÃO DO ASPECTO	IMPLEMENTAÇÃO NO ASPECTJ
Aspect	aspect
Aspect Goal	variável de controle
Plan	método (chama os métodos que implementam as ações)
Action	método

TAB. 4.4: Mapeamento dos relacionamentos verticais, tanto para o JADE quanto para o AspectJ

RELACIONAMENTO VERTICAL	IMPLEMENTAÇÃO
Árvore de Subobjetivos	variáveis de controle, *statements* condicionais
Árvore de Planos e Ações	variáveis de controle, *statements* condicionais, chamadas de métodos

TAB. 4.5: Mapeamento dos relacionamentos horizontais no AspectJ

RELACIONAMENTO HORIZONTAL	IMPLEMENTAÇÃO NO ASPECTJ
Aspect Goal - Agent Goal	não mapeável
Plan (Aspect) - Plan (Agent)	definição de pointcut e advice
Action (Aspect) - Action (Agent)	definição de pointcut e advice, *statements* condicionais, variável de controle

Inserindo hipertexto aos comentários do código de forma a referenciar a representação baseada XML dos modelos, poderia-se percorrer o caminho inverso, rastreando que parte do modelo está sendo endereçada pelo código. Assim, poderia-se relacionar a modelagem de agentes, aspectos, objetivos, planos, ações, relacionamentos verticais e horizontais aos pontos em que são implementados.

4.6 O ACROSS E AS PROPRIEDADES DESEJÁVEIS NA MODELAGEM DE CARACTERÍSTICAS

Para atingir seus fins, o ACROSS promove a modelagem de características tanto transversais quanto das entidades-base de modo a mostrar como as primeiras são separadas, compostas e integradas com as últimas. Características são interesses (ou assuntos) que devem ser levados em consideração no desenvolvimento e operação de um sistema (JUNIOR, 2005). JUNIOR (2005) discute a modelagem de características no desenvolvimento orientado a aspectos e apresenta alguns critérios desejáveis para artefatos que se proponham a tal fim. Com base nestes critérios, é possível avaliar o desempenho do ACROSS e tirar algumas conclusões sobre suas capacidades, limitações, aplicabilidade e, ainda, enxergar desafios a serem superados no futuro. Os critérios considerados foram os seguintes:

- *Generalidade*: Permitir a captura dos mais diversos tipos de características, servindo, assim, a muitos propósitos;
- *Independência*: Não estar atrelada a métodos de desenvolvimento ou abordagens específicas;
- *Propriedade*: Permitir a representação dos diversos tipos de características com todas as informações atreladas a elas desejada, dando liberdade ao usuário para fazê-la;
- *Complitude*: Exprimir tudo o que fôr importante na modelagem de características, como o relacionamento entre elas e condições de consistência;

- *Facilidade de Uso*: Não requerer muito conhecimento prévio dos usuários, o que favorece a adoção da técnica;
- *Ferramental*: Ter ferramenta computacional de apoio ao uso;
- *Metodologia*: Possuir metodologia de modelagem e instanciação associada.

Sobre o critério de generalidade, tem-se que o ACROSS captura características ligadas às funcionalidades específicas dos agentes e as características transversais, trabalhando com as abstrações convencionais dos agentes (objetivos, planos e ações). No entanto, o ACROSS, por ter sido desenvolvido para ser utilizado de forma complementar às linguagens e metodologias de modelagem de agentes conhecidas, não trata de características que também podem ser de interesse para a modelagem de SMAs, como, por exemplo, o conceito de *organização* (DASILVA, 2004a) (CAIRE, 2001). Essa limitação pesa na avaliação do critério de propriedade, embora o ACROSS, por empregar linguagem natural para expressar objetivos e ações, confira liberdade para que o usuário exprima as informações associadas a estas abstrações, não o prendendo a formalismos lógicos, como o uso de predicados. Além disso, a montagem das árvores de objetivos, planos e ações podem conter o nível de detalhamento compatível com o estágio corrente na modelagem de requisitos, variando desde um conteúdo de mais alto nível na fase inicial até a inserção de informações apoiadas nos recursos a serem usados para a implementação da solução, como linguagem, bibliotecas e protocolos específicos.

Sobre a complitude, ao expor como o *crosscutting* ocorre nos níveis de objetivos e ações, o ACROSS define o relacionamento existente entre características funcionais específicas dos agentes e características funcionais. Embora a construção das árvores de objetivos com o emprego dos operadores AND, OR e XOR e a atribuição dos relacionamentos de correlação entre objetivos de agentes e de aspectos envolvam questões de condições de consistência, como o conflito de objetivos, o ACROSS não se aprofunda nisto. Pode-se, portanto, pensar em refinamentos futuros para o ACROSS que contemplem explicitamente tais questionamentos.

Também pelo fato de estar centrado nas abstrações básicas dos SMAs, o ACROSS é totalmente independente de linguagem ou metodologia de modelagem. Além disso, o ACROSS é um framework simples e enxuto (*lightweight*), não empregando formalismos matemáticos e construção de diversos diagramas, o que torna o ACROSS fácil de usar. Os conceitos de objetivos, planos e ações, fundamentais ao ACROSS, são muito familiares à vida cotidiana, bastando que o usuário traga estes conceitos para a modelagem do sistema.

Embora ainda não tenha sido desenvolvida ferramenta gráfica computacional que dê suporte ao uso do ACROSS, já foram dados os primeiros passos neste sentido à medida em que foi proposta a descrição baseada em XML dos elementos de modelagem do framework. O desenvolvimento da ferramenta levará à tona as potencialidades do ACROSS, permitindo a promoção de práticas como o reuso de modelos, afinal um mesmo objetivo pode ser compartilhado por diversos agentes, um determinado aspecto pode ser reutilizado em diferentes sistemas e assim por diante.

O ACROSS, até o presente momento, não conta com uma metodologia específica formalizada. No entanto, para construir as árvores de objetivos dos agentes e aspectos, pode-se usar os passos tradicionais da análise dirigida por objetivos, que são a modelagem dos objetivos mais genéricos, seguido por sua decomposição em subobjetivos e montagem da estrutura de planos para atendimento dos objetivos-folha (SHEN, 2005). Feito isso, o usuário deve identificar os agentes que são atravessados por cada aspecto e descobrir quais os objetivos dos agentes são afetados por quais objetivos dos aspectos e, por fim, determinar como os planos e ações dos aspectos cruzarão os planos e ações dos agentes.

Desta forma, conclui-se que, além de independente de linguagens e metodologias de desenvolvimento específicas e de fácil entendimento, o ACROSS é bastante flexível e customizável, sendo capaz de representar adequadamente separação, composição e integração das características funcionais específicas dos agentes e as características transversais em termos de seus objetivos, planos e ações. Cabe ressaltar que o ACROSS foi desenvolvido para dar suporte explícito à modularização de características transversais sem prescindir do emprego de linguagens e metodologias de modelagem de SMAs para modelar outros pontos importantes, como organizações e interação dinâmica de agentes (CHOREN, 2005) (ODELL, 2000). Além disso, não consegue representar objetos (DASILVA, 2004a).

4.7 CONSIDERAÇÕES FINAIS

Empregando as abstrações convencionais dos SMAs para a modelagem de agentes e aspectos e definindo mecanismos de composição de objetivos, planos e ações entre estas duas entidades, o ACROSS procurou modularizar efetivamente as características transversais em SMAs. Enquanto os relacionamentos transversais entre objetivos contribuem para ampliar o entendimento global do sistema, os relacionamentos de *crosscutting* entre planos e entre ações fornecem informações sobre a maneira como o aspecto atravessa o agente e esta definição será refletida na implementação, já que apontam como serão estruturados os *pointcuts* e caracterizam cada *advice*, estabelecendo se ele será executado

antes, durante ou depois dos *pointcuts* associados. Os planos e ações do aspecto serão de alguma forma implementados e acessados a partir do *advice* cabível.

É importante destacar que o ACROSS abre interessantes horizontes de pesquisa com viés prático, buscando a promoção da rastreabilidade de suas abstrações e relacionamentos durante todo o ciclo de desenvolvimento de software. As idéias do ACROSS motivam o desenvolvimento de uma plataforma de implementação que integre agentes e aspectos, os definindo e relacionando em termos de seus objetivos, planos e ações.

5 ESTUDO DE CASO

Neste capítulo será apresentado um estudo de caso para demonstrar a aplicação do framework ACROSS e reforçar as conclusões expostas no capítulo anterior. Sistemas de Comando e Controle (C2) são bons candidatos a serem concebidos como SMA, além de serem de interesse militar. Desta forma, o estudo de caso a ser desenvolvido neste capítulo terá como foco a modelagem de um SMA de Comando e Controle voltado para atuação da Artilharia de Campanha, nível Brigada. A elaboração deste estudo de caso levou em consideração o padrão doutrinário do Exército Brasileiro, além de ser fortemente inspirado nas especificações do Sistema Gênesis, que vem sendo desenvolvido pela Indústria de Material Bélico do Brasil - Fábrica de Material de Comunicações e Eletrônica (IMBEL/FMCE) (IMBEL, 2006). Cabe ressaltar, no entanto, que o Sistema Gênesis não é concebido como um SMA, mas sim como um sistema orientado a objetos.

Além de ser um exercício de estruturação de um sistema de C2 segundo o paradigma multiagentes, a escolha deste estudo de caso em particular deve-se ao fato da autora ter trabalhado ativamente por cerca de três anos (dezembro de 2001 a janeiro de 2005) no desenvolvimento do Sistema Gênesis, tendo acumulado conhecimentos sobre sistemas para planejamento e direção de tiro de Artilharia. Somando-se a isto, a elaboração da especificação a ser exposta na Seção 5.1 foi também baseada no Manual de Campanha de Técnica de Tiro de Artilharia do Exército Brasileiro, conhecido como C 6-40 (EME, 2001).

O sistema modelado é, na verdade, um sistema fictício, batizado como SMAC (Sistema Multiagentes para a Artilharia de Campanha). Dada a complexidade do funcionamento da Artilharia, algumas simplificações e reduções de escopo precisaram ser adotadas, delineando, assim, uma primeira versão do sistema, o SMAC 1.0. Desta forma, a Seção 5.2 traz a modelagem de agentes e aspectos do SMAC e a Seção 5.3 apresenta alguns comentários sobre este estudo de caso.

5.1 DESCRIÇÃO DO SISTEMA

A Artilharia é a arma responsável pela realização do tiro indireto em combate. Isto é feito sob a forma de execução de Missões de Tiro (MTs). Uma MT consiste em um conjunto de tiros (ou fogos) que são desferidos sobre um alvo inimigo. Uma MT possui

uma série de parâmetros técnicos e táticos em função do efeito que se deseja produzir. Alguns dos possíveis efeitos de uma MT são a neutralização, a destruição, a interdição, a iquietação e a regulação, bem como tiros com outras finalidades como é o caso do fumígeno, iluminativo, etc (EME, 2001).

Cada efeito está associado a um *modus operandi* específico, podendo haver diferenças sensíveis entre eles. Assim, o SMAC 1.0 irá cobrir apenas as MTs de neutralização. De acordo com EME (2001), a neutralização é um tiro que é executado com a finalidade de causar baixas, interromper movimentos e ações, anular a eficiência combativa do inimigo e forçá-lo a abrigar-se. No SMAC 1.0 as MTs de neutralização ficarão restritas às inopinadas, ou seja, desencadeadas com dados colhidos após analisado o alvo, não havendo a possibilidade de realização de tiros previstos, que são antecipadamente planejados, havendo alocação prévia de recursos. Além disso, trabalhar-se-á apenas com MTs observadas, conduzidas por observadores terrestres e que cessarão apenas quando o observador sinalizar que a finalidade foi atingida (missão cumprida). Além disto, estas MTs observadas podem seguir duas dinâmicas diferentes: ou elas são do tipo "Quando Pronto" (QP) ou do "Ao Meu Comando" (AMC). A primeira indica que a linha de fogo pode atirar assim que receber a MT, enquanto a última significa que a linha de fogo deve aguardar o comando para, enfim, atirar sobre o alvo.

5.1.1 OS AGENTES DO SMAC

A grosso modo, a Artilharia no combate nível Brigada atua de forma que hajam (i) Observadores Avançados (OA) para identificar alvos, pedir MTs e acompanhá-las até que sejam consideradas cumpridas e, conseq entemente, encerradas; (ii) Oficiais de Ligação (O Ligs) junto à arma-base (Infantaria ou Cavalaria) para receber os pedidos de MTs, apreciá-los e decidir se é o caso de repassar tais alvos para serem batidos pelo Grupo de Artilharia de Campanha (GAC); (iii) o Oficial de Operações do GAC (S/3 GAC), responsável por gerenciar as baterias que compõem o GAC que, ao receber a MT repassada pelo O Lig, decide por dar ou não prosseguimento a ela, aloca as baterias necessárias para realizá-la e revê os parâmetros do tiro, procedendo, assim, à análise de alvos; (iv) os Comandantes de Linha de Fogo (CLFs) que recebem as MTs a serem cumpridas e comandam o tiro das peças, manted(p)-26(e15(man)GA)26(C)-36(einformad15(man)da)-325(ev)27(olução)-3an por fim, (v) (CPs) que são os praças junto às peças que dirigem os trabalhos neste local, de modo que cada obuseiro seja ajustado para os valores calculados de deriva e elevação, receba a muniçãocom a granada, espoleta e carga correta e execute o tiropropriamentedito.

Desta forma, o SMAC 1.0 conta com cinco tipos de agente que representam os usuários humanos que desempenham os papéis fundamentais para a condução das MTs. Esses agentes são modelados em conformidades com as funções destes usuários no campo de batalha. Desta maneira, tem-se os seguintes tipos de agentes:

- *Observador Avançado (OA)*: Responsáveis pela transmissão do pedido da MT e por seu acompanhamento, uma vez que os usuários correspondentes são os únicos elementos que têm vistas sobre o alvo. Assim, além de levar adiante que a missão foi cumprida, também repassam dados para correção do tiro, sinalizam o momento apropriado para efetuar o tiro e alertam para a necessidade de cessar fogo.

- *Oficial de Ligação (O Lig)*: Representando os elementos de Artilharia ligados à arma-base, estes agentes recebem o pedido de MT expedidos pelos OAs e auxiliam na decisão de continuar ou não com a MT, uma vez que conhece o posicionamento das tropas amigas e a zona de atuação de cada tropa. Servem também de elo de comunicação entre os OAs e a linha de fogo, repassando mensagens em ambos os sentidos.

- *S3 do Grupo de Artilharia de Campanha (S/3 GAC)*: Trata-se de um agente que controla os recursos de Artilharia do grupo, recebendo os pedidos de MT repassados pelo O Lig e apoiando a decisão de dar ou não prosseguimento à MT, definir quais os parâmetros adequados do tiro e as baterias que devem ser empregadas, conforme seu alcance e disponibilidade. Além disso, também atuam como elos de comunicação, assim como os O Ligs.

- *Comandante da Linha de Fogo (CLF)*: São agentes que recebem a ordem de executar a MT e tratam de comandar a execução do tiro, repassando informações para as peças, bem como são responsáveis por receber informações das peças a respeito do tiro (se foi ou não executado, consumo de munição, etc) e repassá-las ao S3.

- *Chefe de Peça (CP)*: Leva ao conhecimento do respectivo agente CLF as informações sobre a execução do tiro reportadas pelo usuário CP, bem como recebe do CLF o comando de tiro.

Cada agente acima descrito tem uma interface com o usuário humano que executa a função correspondente no teatro de operações. Os OAs podem ser alimentados de

informações vindas de usuários humanos ou de sensores automatizados, como telêmetro laser, teodolito, etc. Os O Lig e o S3/GAC têm alguma autonomia e atuam no suporte à decisão do usuário humano, fazendo escolhas baseadas em suas crenças e no contexto. Já os CLFs e CPs são agentes basicamente reativos que recebem ordens, realizam algum tipo de processamento interno (realização de cálculos balísticos, por exemplo), informando aos usuários humanos como proceder, bem como recebendo destes informações sobre a execução do tiro para serem repassadas ao S3/GAC. A Figura 5.1 exibe os agentes que compõem o sistema e os usuários humanos que eles representam.

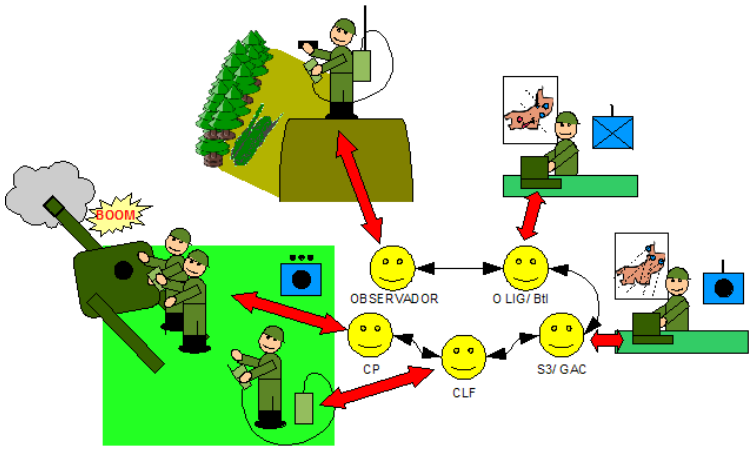

FIG. 5.1: O SMA de C2 para a Artilharia de Campanha, batizado como SMAC, com seus agentes e usuários humanos.

5.2 MODELAGEM DO SMAC 1.0

A presente seção encarrega-se de apresentar a modelagem do SMAC utilizando o framework ACROSS. Assim, a Seção 5.2.1 exibe a modelagem de agentes do SMAC, explicitando seus objetivos, planos e ações, bem como os relacionamentos verticais entre eles, enquanto a Seção 5.2.2 faz o mesmo, considerando certos aspectos do sistema. Finalmente, a Seção 5.2.3 modela os relacionamentos horizontais existentes entre agentes e aspectos em todos os níveis.

5.2.1 MODELAGEM DOS AGENTES

Como a modelagem dos cinco tipos de agente que compõem o SMAC seria muito extensa, optou-se apresentar apenas a modelagem de dois tipos de agentes: os OAs e os

O Ligs. Esta escolha é justificada pelo fato que são os dois primeiros agentes a atuar no processo de realização de MT, uma vez que o OA é responsável por pedir a MT e o O Lig é o primeiro a receber e apreciar este pedido. Desta forma, pretende-se demonstrar a aplicação do ACROSS na representação de suas árvores de objetivos, planos e ações. A Seção 5.2.1.1 trata da modelagem do agente OA, enquanto a Seção 5.2.1.2 expõe a modelagem do agente O Lig.

5.2.1.1 MODELAGEM DO OA

Sendo representante do único elemento que tem vistas sobre o alvo, o agente OA tem como missão primordial auxiliar na condução da MT. Assim, o OA deve permitir a identificação dos alvos em potencial, deve conseguir que seja pedida a MT para neutralizar tais alvos, deve receber a Mensagem Resposta (MRO), onde é definido se a MT será ou não levada adiante, fornecendo os parâmetros do tiro a ser observado. Em caso da MT ser executada, o OA deve oferecer condições para o acompanhamento do tiro. Por acompanhamento do tiro, entende-se prover a correção do tiro, de modo a aproximá-lo cada vez mais do alvo; suspender fogo se necessário fôr (presença de tropas amigas, por exemplo); comandar o fogo, seja para cessar a suspensão ou para avisar a respeito do momento oportuno para efetuação do tiro (MT AMC); e, por fim, pedir o encerramento da MT, a partir do momento em que o alvo já puder ser considerado neutralizado. Assim a Figura 5.2 expõe a modelagem da árvore de objetivos do agente OA, segundo o ACROSS.

Para cumprir cada um dos objetivos enunciados, é necessário conceber planos. Para atender ao objetivo "Buscar alvo" pode-se estabelecer um plano de busca de alvos que consiste basicamente em um subplano para a identificação de alvos, a ação de cruzar os dados do alvo identificado com as informações de inteligência disponíveis no intuito de certificar-se se o pedido de MT é apropriado e, finalmente, a ação de sugirir a conduta diante do alvo, ou seja, explicitar se deve ou não ser pedida uma MT para batê-lo. Para identificar um alvo, é preciso que o agente receba dados como as coordenadas geográficas do alvo, a natureza deste (por exemplo, Pelotão de Carros de Combate) e de suas dimensões (largura e profundidade no terreno). A Figura 5.3 traz a modelagem dos planos e ações associados a este objetivo.

No caso do objetivo "Pedir MT", o SMAC 1.0 só contempla o plano para o pedido de MT de neutralização. Numa evolução, deveria ser incluído um XOR entre os planos apropriados para os diversos efeitos (barragem, fumígeno, iluminativo, etc). O plano para pedido de MT de neutralização (5.4) inclui as ações de cruzar a natureza e dimensões do

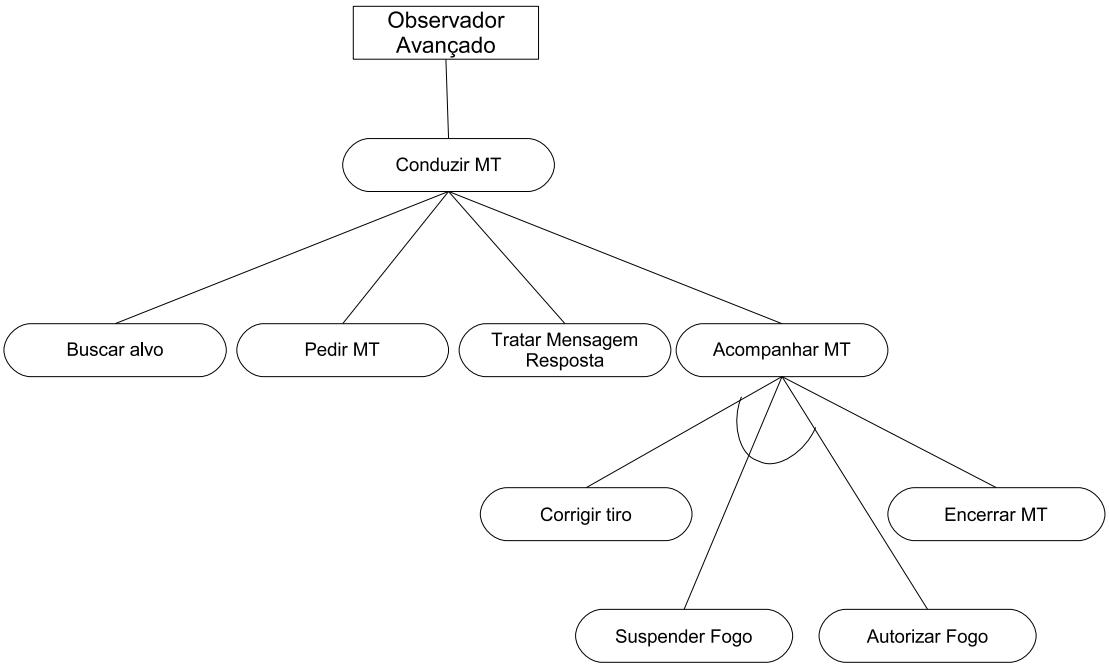

FIG. 5.2: Representação gráfica da árvore de objetivos do agente OA.

FIG. 5.3: Representação gráfica dos planos e ações associados ao objetivo "Buscar alvo".

alvo com informações de nomogramas (padrões usuais de número de rajadas, munição, etc para bater um determinado tipo de alvo, constante de manuais como o C 6-40), sugerir os parâmetros para a execução da MT e receber os parâmetros finais do pedido uma vez que o usuário humano, embora conte com as habilidades do agente como ferramenta de suporte à decisão, tem sempre a palavra final, uma vez que é um combatente treinado e experimentado para executar este trabalho.

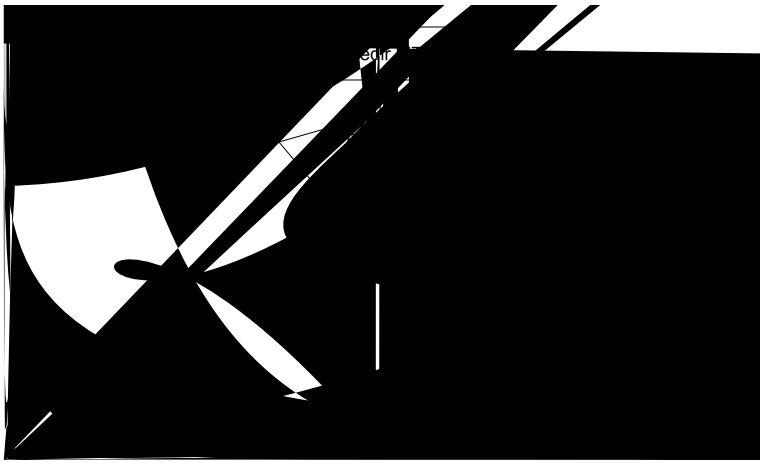

FIG. 5.4: Representação gráfica dos planos e ações associados ao objetivo "Pedir MT".

"Tratar Mensagem Resposta" (Figura 5.5) está associado a um plano de uma única ação, uma vez que simplesmente repassa ao usuário OA as informações contidas na MRO para que ele possa ter ciência se a MT ocorrerá e, em caso positivo, que tipo de tiro esperar (número de rajadas, munição, etc). Quem, de fato, define os parâmetros de execução da MT é o S/3 GAC, mas tanto ele quanto o O Lig podem optar pelo chamado "Não atirarei", ou seja, declinar da realização da MT, seja por conveniências táticas, informações de inteligência privilegiadas ou limitação de suprimentos de munição, por exemplo.

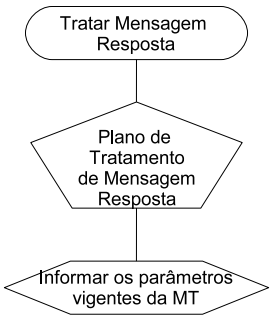

FIG. 5.5: Representação gráfica dos planos e ações associados ao objetivo "Tratar Mensagem Resposta".

Acompanhar a MT é parte essencial do trabalho do OA. Durante o acompanhamento da MT, o agente OA pode atuar na correção do tiro (Figura 5.6), na suspensão do fogo (Figura 5.7), na autorização do fogo (Figura 5.8) e, no momento oportuno, no encerramento da MT (Figura 5.9). Corrigir o tiro implica em indicar o deslocamento que o tiro precisa sofrer nos dois eixos para melhor atingir o alvo ou, ainda, ordenar a mudança do controle do tiro de ajustagem para eficácia, quando todas peças da bateria atirarão no alvo. Os planos associados aos objetivos "Suspender Fogo" e "Autorizar fogo" são compostos apenas da ação de receber os respectivos comandos do usuário humano. Já o plano associado ao objetivo "Encerrar MT", além do recebimento do comando de missão cumprida, o agente necessita que o usuário informe sua avaliação a respeito do percentual de baixas provocadas pela MT.

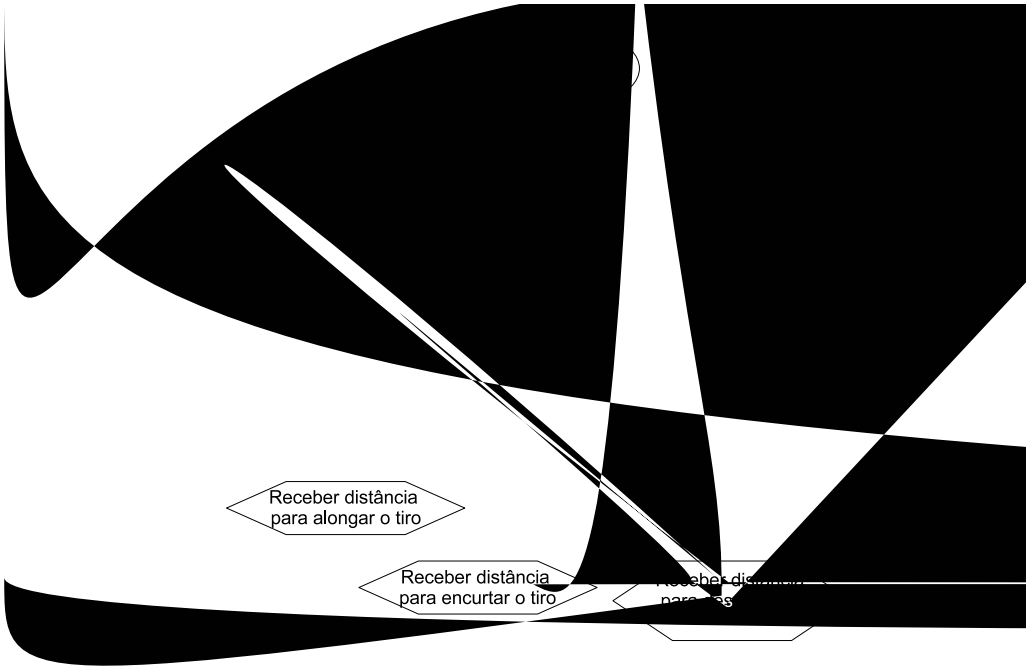

FIG. 5.6: Representação gráfica dos planos e ações associados ao objetivo "Corrigir tiro".

A descrição baseada em XML da modelagem do agente OA pode ser encontrada no Apêndice 8.1.

5.2.1.2 MODELAGEM DO O LIG

O O Lig tem como missão principal tratar a MT sob a perspectiva da manobra da arma-base, estando integrado a um Batalhão ou Regimento. Assim sendo, ele tem acesso à informações como, por exemplo, o posicionamento destas tropas, seus planos de combate e outras informações de inteligência. Tais informações (crenças) irão influir em seus

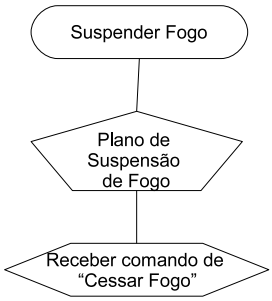

FIG. 5.7: Representação gráfica dos planos e ações associados ao objetivo "Suspender Fogo".

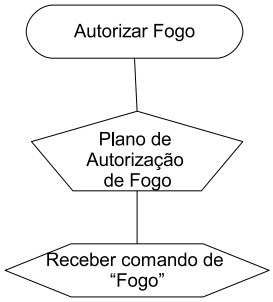

FIG. 5.8: Representação gráfica dos planos e ações associados ao objetivo "Autorizar Fogo".

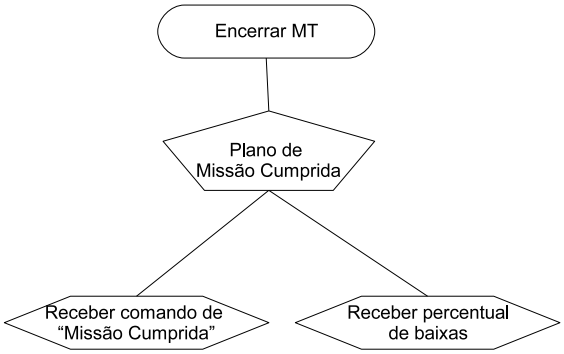

FIG. 5.9: Representação gráfica dos planos e ações associados ao objetivo "Encerrar MT".

subobjetivos, pois ao receber um pedido de MT, ele deve cruzar os dados do pedido com a situação da manobra, de modo a decidir se a melhor conduta é não atirar no alvo ou repassá-lo para o S/3 GAC para que a MT possa vir a ser concretizada. Durante o acompanhamento da MT, o O Lig precisará fazer uma análise da situação da manobra, confrontando-a com os acontecimentos presentes da MT, uma vez que informações resultantes podem ser determinantes para que o O Lig ordene a suspensão e retomada do fogo. Fora isso, também no acompanhamento da MT, o agente O Lig deve dar conhecimento ao respectivo usuário sobre as mensagens trocadas entre observadores e linha de fogo (correções, cessar fogo, comandar fogo, missão cumprida), para que tenha ciência do que está acontencendo, de modo que possa intervir sempre que os interesses da manobra estejam sendo afetados. A Figura 5.10 retrata a árvore de objetivos do O Lig.

FIG. 5.10: Representação gráfica da árvore de objetivos do agente O Lig.

Para cumprir o objetivo "Analisar o pedido de MT" (Figura 5.11) deve-se cruzar os dados do pedido com a situação dos elementos manobra e, a partir daí iniciar um plano para definir a reação do O Lig. Neste plano, uma única conduta deve ser adotada: ou o O Lig sugere pela não realização da MT através do comando "Não atirarei" ou sugere que o alvo seja repassado para o S/3 para que este decida se empregará seus recursos de Artilharia na MT e como o fará. De qualquer maneira, a sugestão do agente precisa ser ratificada (ou retificada) pelo usuário O Lig. De forma bastante semelhante, durante o acompanhamento das MTs em atividade, o agente O Lig precisa avaliar se alguma delas está comprometendo a manobra da arma-base e alertar o usuário sobre necessidade de suspender ou retomar fogo, como pode ser conferido na Figura 5.12. o objetivo "Tratar mensagens vindas de outros agentes" (Figura 5.13) está associado à ação de informar ao

usuário sobre as mensagens circulantes, para mantê-lo ciente do desdobramento das MTs. Por fim, reutiliza-se a modelagem dos objetivos "Suspender Fogo" e "Autorizar Fogo" associados ao OA (Figuras 5.7 e 5.8, respectivamente).

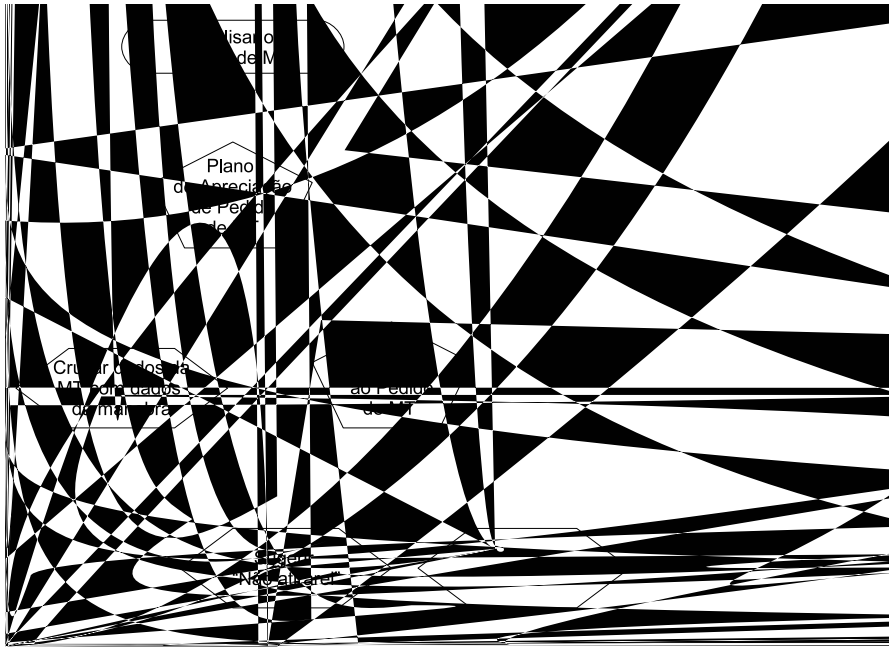

FIG. 5.11: Representação gráfica dos planos e ações associados ao objetivo "Analisar o pedido de MT".

5.2.2 MODELAGEM DOS ASPECTOS

Dois aspectos próprios dos SMAs são ortogonais à modelagem dos agentes OA e O Lig: a interatividade e a autonomia. O aspecto interatividade (Seção 5.2.2.1) surge à medida que ambos precisam comunicar-se um com o outro (o O Lig também interage com o S/3 GAC). Já a autonomia (Seção 5.2.2.2) entremeia a modelagem destes agentes, pois algumas vezes é necessário que eles tomem decisões baseadas nas suas crenças e no contexto. Isso ocorre, por exemplo, quando o OA, baseado no tipo de alvo, sugere parâmetros da MT baseados em informações dos nomogramas ou quando o O Lig precisa consultar a situação da manobra e confrontá-la com parâmetros como a dispersão do tiro e seu raio letal para concluir sobre a segurança das tropas amigas no terreno.

Cabe ressaltar que muitas outras características transversais podem ser modularizadas na modelagem do SMAC. Por ser um sistema de C2 militar, existem requisitos de segurança que atravessam as funcionalidades do sistema, como a autenticação de usuário e a criptografia de dados. A manipulação do banco de dados e a atualização na carta do posicionamento das tropas e de linhas como as referentes a Medidas de Coordenação e

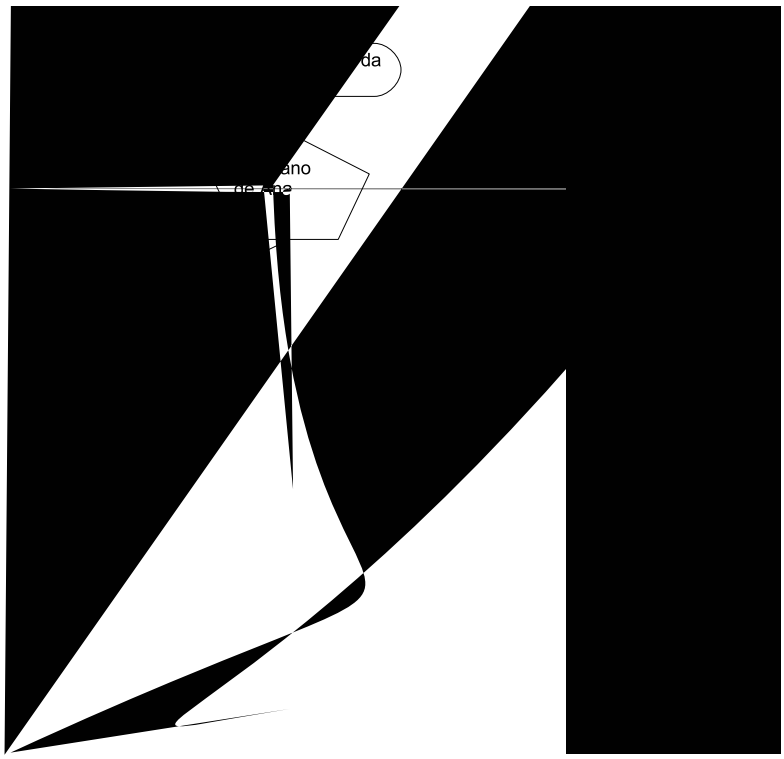

FIG. 5.12: Representação gráfica dos planos e ações associados ao objetivo "Analisar a situação da manobra".

FIG. 5.13: Representação gráfica dos planos e ações associados ao objetivo "Tratar mensagens vindas de outros agentes".

Controle e a limites de atuação das unidades também podem ser modularizadas. Outras propriedades dos agentes também estão presentes e podem ser modularizadas. Uma delas, por exemplo, é o aprendizado, pois, como o S/3 GAC pode modificar os parâmetros da MT originalmente pedida pelo OA, este último poderia identificar padrões nestas alterações e cuidar para que a sugestão de parâmetros no pedido da MT considere o perfil de raciocínio do S/3 GAC ao proceder a análise de alvos. Da mesma forma, o agente S/3 GAC pode aprender com a forma de analisar um alvo do usuário humano e incorporar esta percepção na sua sugestão de parâmetros da MT.

Assim, a escolha dos aspectos interatividade e autonomia deveu-se ao fato de serem bastante presentes, não só atravessando a modelagem do OA e do O Lig, bem como a de outros agentes, uma vez que todos eles trocam mensagens entre si e o S/3 GAC também tem que tomar decisões autônomas.

5.2.2.1 MODELAGEM DO ASPECTO INTERATIVIDADE

O objetivo central do aspecto interatividade é promover a comunicação entre agentes, tendo como subojetivos promover o envio o recebimento de mensagens. Assim, pode-se reutilizar a modelagem de objetivos exposta na Figura 4.6, do Capítulo 4. Da mesma forma, os planos associados ao objetivos "Enviar mensagem" e "Receber mensagem" também são os mesmos.

As Figuras 5.14, 5.15 e 5.16 reproduzem a modelagem deste aspecto.

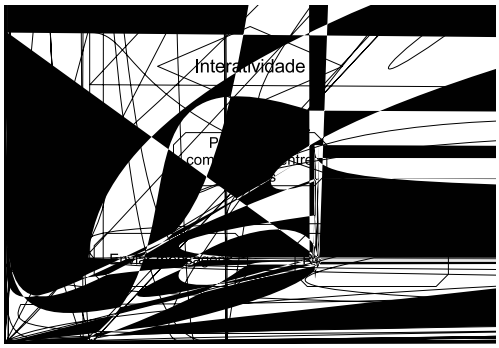

FIG. 5.14: Representação da árvore de objetivos do aspecto interatividade.

5.2.2.2 MODELAGEM DO ASPECTO AUTONOMIA

De acordo com LOUGHRAN (2006), a propriedade de autonomia significa que o agente tem controle sobre suas ações e pode agir independentemente de outros para atingir seus objetivos internos, ou seja, a característica autonomia realiza a gerência dos objetivos do

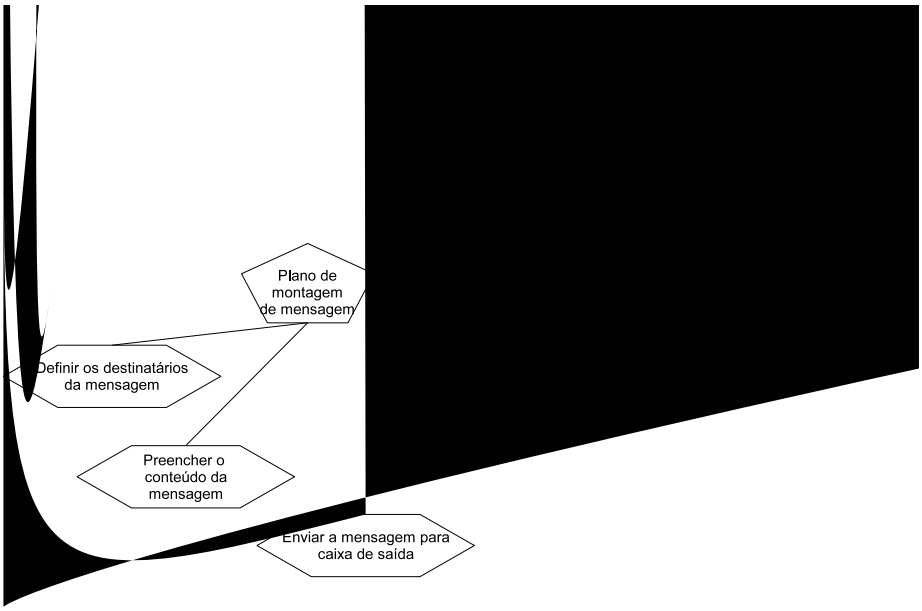

FIG. 5.15: Representação dos planos associados ao objetivo "Enviar mensagem" do aspecto interatividade.

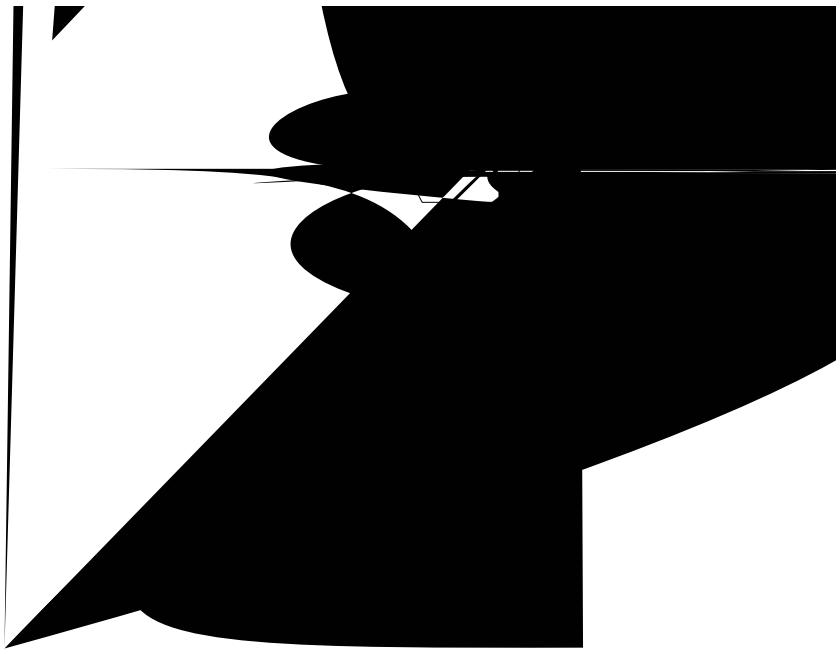

FIG. 5.16: Representação dos planos associados ao objetivo "Receber mensagem" do aspecto interatividade.

agente. Assim, a autonomia pode ser classificada como autonomia de execução, autonomia de decisão e autonomia pró-ativa. A primeira dimensão mostra que, para atingir seus objetivos, os agentes devem ter suas próprias *threads* de controle. Já a segunda, indica que os agentes devem poder tomar decisões na instanciação de objetivos e a última aponta que os agentes podem executar ações sem intervenção externa direta.

Em alguns momentos, tanto o OA quanto o O Lig podem estar envolvidos em questões de autonomia decisória, seja para sugerir se deve ser feito o pedido da MT ou se este pedido deve ser encaminhado ao GAC, por exemplo, instanciando assim seus respectivos objetivos. Desta forma, os mecanismos de autonomia decisória podem ser isolados e modelados como um aspecto. Mecanismos como estes podem ser compartilhados por outros agentes. Para citar um caso, o S/3 GAC também precisará definir os parâmetros da MT durante a análise de alvos.

Assim, no SMAC, o objetivo central do aspecto autonomia é tomar decisões (Figura 5.17). Outros objetivos associados às demais dimensões de autonomia poderiam ser futuramente acrescidos a esta modelagem, se necessário. Poderia-se pensar em inúmeras estratégias para promover a tomada de decisões, mas na modelagem do SMAC, até então, só foi requerido um plano que baseasse a decisão nas crenças e contexto que nada mais são do que informações disponíveis aos agentes como informes de inteligência, dados de manuais de campanha, informações geográficas, informações provenientes de sensores (como GPS, por exemplo) e assim por diante. Tais dados fariam parte de uma base de conhecimento acessível aos agentes interessados. Com isso, basta o agente acessar esta base e, de posse dos parâmetros de busca adequados, procurar por informações e tomar decisões em função do retorno, o que pode ser verificado no modelo exposto pela Figura 5.18.

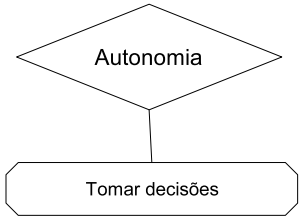

FIG. 5.17: Representação do objetivo do aspecto autonomia.

A descrição baseada em XML dos aspectos interatividade e autonomia podem ser encontradas no Apêndice 8.1.

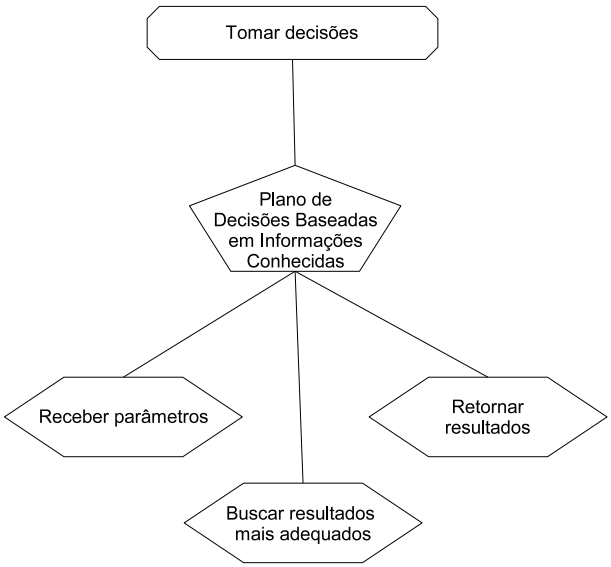

FIG. 5.18: Representação dos planos associados ao objetivo "Tomar decisões" do aspecto autonomia.

5.2.3 MODELAGEM DO *CROSSCUTTING*

A modelagem do *crosscutting* define os relacionamentos horizontais existentes entre agentes e aspectos em todos os níveis. Assim, o aspecto interatividade atravessa os agentes OA e O Lig sempre que estes precisarem enviar ou receber mensagens de outros agentes. Já o aspecto autonomia atravessará os agentes sempre que eles precisarem recorrer a informações disponíveis para a tomada de decisão.

Assim sendo, o OA, antes de tratar a MRO, precisa recebê-la. Como sem recebê-la, não há como tratá-la, o relacionamento de correlação entre os objetivos é o *make*. Como o plano de recepção tem que ser inteiramente executado antes de tratar a mensagem, o relacionamento entre planos deve ser o *add before*. De forma análoga, para concretizar o pedido de MT, bem como dar conhecimento das correções, do cessar fogo, do fogo e da missão cumprida é imprescindível que ocorra o envio destas mensagens, o que caracteriza o relacionamento *make* entre objetivos e o relacionamento *add after* no nível de planos, afinal o envio de mensagens deve ser a última coisa a ser feita. O mesmo vale para o agente O Lig que precisa receber a MT para analisá-la, bem como precisa receber mensagens de outros agentes para repassá-las (o O Lig serve como elo de ligação entre OAs e linha de fogo). Além disso, após analisar a MT, deve enviar o "Não atirarei" para o OA ou repassar o pedido para o S/3 GAC. A suspensão e autorização de fogo precisa ser avisada tanto para o OA quanto para o S/3 GAC. Desta forma, o *crosscutting* entre o aspecto interatividade e o agente OA está representado na Figura 5.19, enquanto a Figura 5.20

traz o agente O Lig sendo atravessado pelo respectivo aspecto.

FIG. 5.19: *Crosscutting* entre o agente OA e o aspecto interatividade.

O aspecto autonomia atravessa o OA quando este busca atingir os objetivos "Buscar alvo" e "Pedir MT", de acordo com o exposto na Figura 5.21. O mesmo ocorre com o agente O Lig quando este persegue os objetivos "Analisar o pedido de MT" e "Analisar a situação da manobra", conforme mostra a Figura 5.22. Em ambos os casos, a composição de objetivos é caracterizada pelo relacionamento *help*, uma vez que configuram sugestões para auxílio na tomada de decisões que poderiam ser tomadas exclusivamente pelo usuário. Desta forma, tanto o usuário OA quanto o usuário O Lig poderiam cumprir os objetivos valendo-se apenas de seu raciocínio, treinamento e experiência, mas, sem dúvida, podem otimizar este cumprimento valendo-se deste tipo de apoio a decisão capaz de chamar a atenção para fatos porventura não percebidos pelo usuário humano, como, por exemplo, ressaltar que uma determinada tropa amiga encontra-se em área que pode sofrer impacto dos fogos de Artilharia ou lembrar que o alvo inimigo sobre o qual se pede uma MT já está destinado a sofrer ataque aéreo e assim por diante. O plano de tomada de decisões funde-se com os planos para atingir os objetivos citados, pois podem surgir como partes integrantes destes, não como algo que deve ser realizado por completo antes ou depois destes. As ações de "Receber parâmetros" e "Buscar resultados mais adequados" nada mais são do que a redefinição das ações que realizam cruzamentos de dados (relacionamento *redefine*), pois irão, de fato, fazê-las acontecer pela conveniente manipulação da base de dados. Já

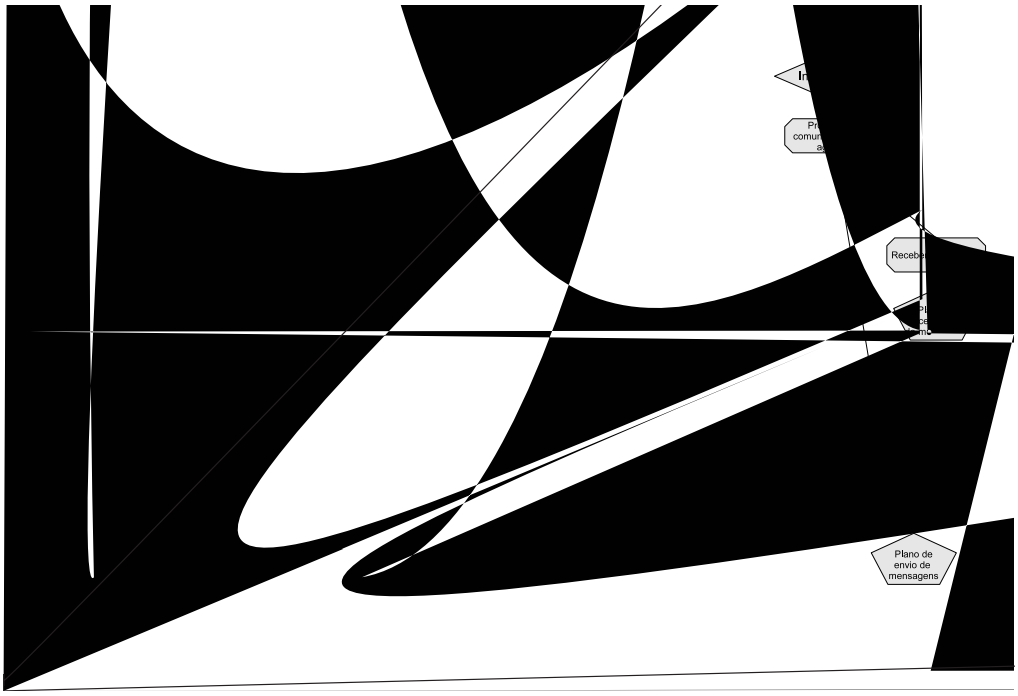

FIG. 5.20: *Crosscutting* entre o agente O Lig e o aspecto interatividade.

a ação "Retornar resultados" oferece subsídios para a sugestão de reação, refinando esta ação, sendo o primeiro passo a ser executado e, por isso, configurando o *refine before*.

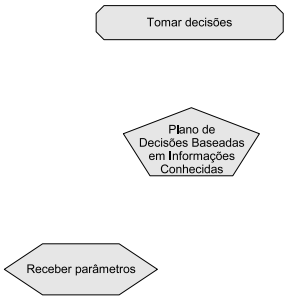

FIG. 5.21: *Crosscutting* entre o agente OA e o aspecto autonomia.

O *crosscutting* entre o agente OA e os aspectos autonomia encontra-se descrito segundo a notação baseada em XML arbitrada no Apêndice 8.1.

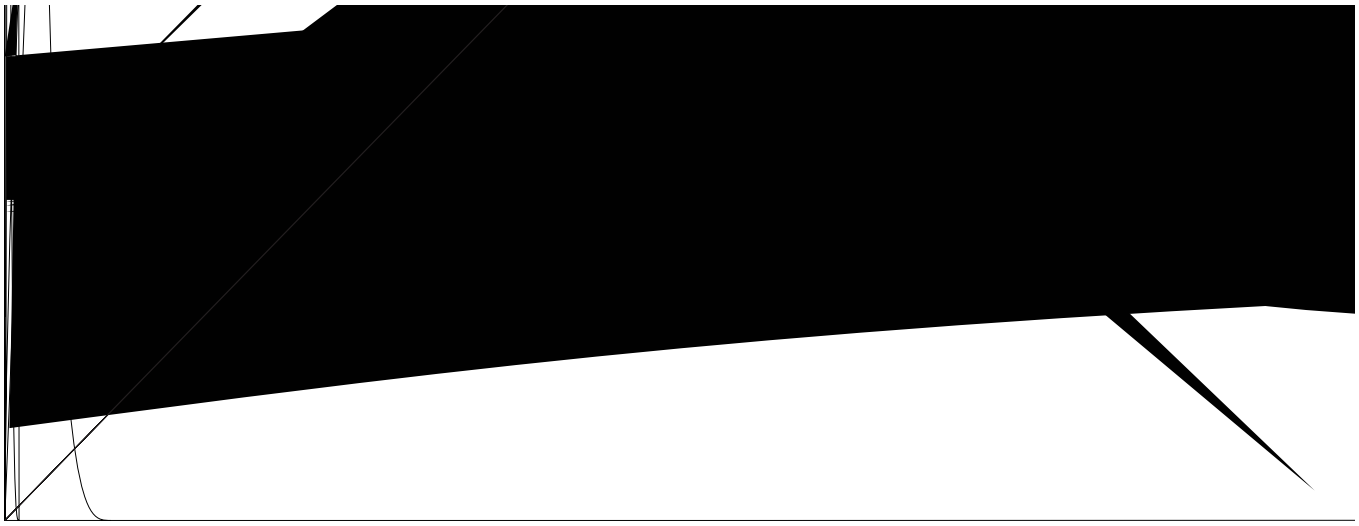

FIG. 5.22: *Crosscutting* entre o agente O Lig e o aspecto autonomia.

5.3 CONSIDERAÇÕES FINAIS

Na modelagem do estudo de caso realizada no presente capítulo ficaram evidenciadas as principais qualidades do ACROSS. Destaca-se, sobretudo, a criação e promoção de oportunidade de reuso de modelos, fruto da modularização de características promovida pelo framework. Conseguiu-se, desta forma, modelar objetivos, planos e ações dos agentes do sistema, bem como dos aspectos identificados. Além disso, foi possível definir os relacionamentos horizontais entre agentes em aspectos no nível de objetivos, incrementando assim o raciocínio modular e composicional das características transversais na modelagem de SMAs, bem como trabalhou-se os níveis de planos e de ações, sugerindo como serão definidos os *join points*, *pointcuts* e *advices* na implementação.

6 CONCLUSÕES E TRABALHOS FUTUROS

Embora os paradigmas dos Sistemas Multiagentes e da Orientação a Aspectos já venham sendo bastante explorados no universo da pesquisa em Engenharia de Software, muitos assuntos ainda estão longe de ser exauridos. Os sistemas demandados são cada dia mais descentralizados e complexos, o que favorece que os SMAs venham a assumir uma importância comparável a que ostenta hoje os Sistemas OO, podendo transformar-se em um padrão de indústria. É verdade que para tal é preciso vencer obstáculos ainda associados a esta tecnologia, como a padronização de conceitos e técnicas de modelagem e o desenvolvimento de linguagem de programação (realmente) orientada a agentes.

Outro pilar fundamental para esta dissertação é que a Orientação a Aspectos trouxe consigo grandes vantagens para o desenvolvimento de software ao modularizar as características transversais, reduzindo o acoplamento, aumentando as possibilidades de reuso e facilitando as atividades de evolução e manutenção dentro do ciclo de desenvolvimento de software.

Desta forma, a presente dissertação destina-se a atacar um ponto de grande importância que, no estado-da-arte das pesquisas, foi alvo de muito poucos esforços, carecendo de amadurecimento e de questionamentos mais aprofundados. Este ponto é justamente a modularização de características transversais em SMAs, considerando as peculiaridades destes sistemas e as conseq entes oportunidades de separação, composição e integração de suas características transversais. Compartilhando o mesmo raciocínio dos pesquisadores de *Early Aspects*, decidiu-se propor uma forma de modularização de características transversais em modelos de requisitos, pois quanto mais cedo identificados e modelados os candidatos a aspectos, menores serão os impactos nas etapas subseq entes do desenvolvimento. Cabe ressaltar também que as linguagens e metodologias de modelagem para SMAs existentes atualmente (ODELL, 2000) (CHOREN, 2005) (DASILVA, 2004b) (BRINKKEMPER, 2000) (JENNINGS, 2000) (DELOACH, 2000) (CAIRE, 2001) não oferecem suporte à modularização de características transversais.

Neste contexto, começaram a surgir trabalhos que objetivam tratar as características transversais em SMAs. SILVA (2006a) propôs uma abordagem sistemática para especificar aspectos, aplicando-a no projeto arquitetural de SMAs. Esta abordagem, no entanto, absteve-se de dar um tratamento especial às propriedades internas próprias dos agentes, embora estas possuam acentuado caráter transversal e possam, de fato, ser modularizadas

GARCIA (2005). GARCIA (2006b) e GARCIA (2006a), apesar defenderem a aspectualização destas propriedades e introduzirem um framework de modelagem apoiado nas abstrações convencionais dos agentes não se preocuparam em tratar os aspectos como unidades modulares reutilizáveis, mas como subobjetivos modularizados dos agentes. Com isso, algumas oportunidades de reuso de mecanismos que promovam as propriedades dos agentes não conseguem ser convenientemente registradas na modelagem. Além disso, em face das tecnologias hoje existentes para implementação de SMAs e de aspectos, raciocinar com o *crosscutting* no nível de objetivos torna o mapeamento de modelos em código mais complexo, pois, ao contrário de planos e ações, que se traduzem em métodos, o objetivo não possui um mapeamento concreto, sendo mais um elemento útil à compreensão global do sistema.

Assim, para resolver o problema identificado e agregar pontencialidades não observadas em abordagens afins, a presente dissertação concentrou-se no desenvolvimento de um framework de modelagem capaz de modularizar as caracterísiticas transversais identificáveis na modelagem de SMAs, sejam elas relacionadas às qualidades do sistema, a requisitos funcionais ou mesmo a mecanismos relacionados às propriedades características dos SMAs. Este framework promoveu a separação das caracterísiticas transversais, propondo abstrações capazes de fazê-lo; a composição destas caracterísiticas pela definição de relacionamentos convenientes e, por fim, a integração dos aspectos modelados com as entidades-base (agentes e suas árvores de abstrações) via aplicação dos mecanismos de composição propostos.

Trabalhou-se também para que o framework desenvolvido fosse independente de linguagem ou metodologia de modelagem específica, fundamentando-se nas abstrações convencionais dos agentes de software e atuando de forma complementar aos recursos de modelagem conhecidos. Somado-se a isto, adotou-se a perspectiva que os aspectos podem ser encarados como unidades modulares reutilizáveis e procurou-se refletir tal pensamento na concepção do framework. Alguns exemplos e um estudo de caso foram desenvolvidos com intuito tanto de apresentar a abordagem proposta, como também de validá-la.

6.1 LISTA DE CONTRIBUIÇÕES

As contribuições inicialmente listadas como metas para o presente trabalho foram concretizadas da seguinte forma:

a) **Prover separação de características transversais na modelagem de SMAs: propor abstrações de modelagem que encapsulem tais ca-**

racterísticas, permitindo seu isolamento das demais características do sistema.

Convencionando objetivos, planos e ações como as abstrações essenciais do paradigma de agentes de software e compreendendo que os aspectos em SMAs, para respeitar o paradigma de desenvolvimento, também compartilham as mesmas abstrações, definiu-se que as características transversais devem ser modularizadas como aspectos e representadas de modo a explicitar seus objetivos, mantendo a simetria no framework proposto.

b) **Prover mecanismos de composição das características tranversais modeladas em separado com o resto da modelagem do sistema: propor relacionamentos que permitam caracterizar como ambos se juntam.**

Além da definição das abstrações, estipulou-se dois tipos de relacionamentos. Os relacionamentos verticais expressam a decomposição das árvores de objetivos associados a cada agente ou aspecto, bem como a árvore de planos associada a cada objetivo e a árvore de ações que define cada plano. Isto é feito pelo emprego dos operadores AND, OR e XOR. Já os relacionamentos horizontais caracterizam o *crosscutting* existente entre aspectos e agentes, mostrando como o objetivo de um aspecto influencia o objetivo de um agente e em que ponto planos e ações do aspecto atravessarão planos e ações do agente. A composição de objetivos é denotada pelo relacionamentos de correlação MAKE, HELP, HURT, BREAK e UNKNOWN. A composição de planos é definida pelos relacionamentos ADD BEFORE, ADD AFTER e MERGE. Este último relacionamento é detalhado no nível das ações, através dos relacionamentos REFINE BEFORE, REFINE AROUND, REFINE AFTER e REDEFINE.

c) **Propor um framework de modelagem independente de linguagem ou metodologia de modelagem de SMA em uso que congregue as abstrações e relacionamentos propostos: criar uma notação para representar as abstrações e relacionamentos, permitindo a construção de diagramas que ofereçam a visão integrada descrita no item anterior.**

O ACROSS, conjungando as abstrações fundamentais do paradigma dos

agentes de software com os relacionamentos verticais e horizontais, mostrou-se ser um framework de modelagem independente de linguagem ou metodologia de modelagem de SMA em uso, podendo ser empregado de forma complementar a elas desde que possuam os conceitos de objetivo, plano e ação. Além disso, convencionou-se uma notação para representar cada abstração e relacionamento proposto e deu-se flexibilidade para que o usuário possa isolar do restante da modelagem apenas o que lhe interesse, seja o *crosscutting* no nível de objetivos ou árvore de planos associadas a um determinado objetivo e assim por diante.

d) **Estudar a aplicação de conceitos de Modelos de Metas na modelagem de características transversais em SMAs: investigar como os conceitos destes modelos podem ser úteis na proposição das abstrações e relacionamentos para modelar características transversais em SMAs.**

Como os agentes de software são orientados a objetivo, o estudo da Orientação a Metas pode ser muito valioso na compreensão do modelo mental dos agentes. Assim, foram estudadas algumas abordagem de modelagem de metas, como o i*, KAOS e V-Graph, no intuito de investigar como as idéias e técnicas de modelagem do sistema sob perspectiva de seus objetivos poderiam contribuir para caracterizar o *crosscutting*. A idéia dos relacionamentos de correlação do V-Graph, em particular, chamaram bastante atenção, pois foi vislumbrada uma maneira de redefinir o significado de cada tipo de relacionamento existente, de modo a formar um conjunto completo de possibilidades de um objetivo de aspecto influenciar um objetivo de um agente, contribuindo para o melhor entendimento global do sistema, localizando dependências, colaborações e conflitos de objetivos. O amadurecimento desta idéia deu origem aos relacionamentos horizontais do ACROSS no nível de objetivos. Além disso, estudou-se a questão da decomposição de objetivos e planos, formando os conceitos dos relacionamentos verticais.

e) **Sugerir diretrizes para mapeamento da forma de modelagem proposta em código: baseando-se em tecnologias existentes para a implementação de SMAs e de aspectos, discorrer sobre maneiras de refletir no código os modelos concebidos.**

Considerando o JADE e o AspectJ como duas plataformas populares para desenvolvimento de SMAs e de POA, respectivamente, discorreu-se sobre a implementação de agentes e aspectos, ressaltando que os objetivos modelados e seus relacionamentos transversais não seriam explicitamente mapeados. Uma das formas de mapeamento seria o emprego de variáveis e fluxos de controle. Já os planos e ações apareceriam representados de maneira mais marcante, através da definição de chamadas a métodos e métodos propriamente ditos. Ficou, porém, o desafio de produzir soluções que promovam um mapeamento mais direto.

6.2 TRABALHOS FUTUROS

Algumas questões referentes ao trabalho desenvolvido ainda ficaram em aberto. Assim, é possível propor algumas orientações para pesquisas e desenvolvimentos futuros, a saber:

a) **Desenvolvimento de ferramenta computacional que implemente as abstrações e relacionamentos do ACROSS.**

Tal ferramenta, além de permitir o desenho dos modelos, de acordo com as abstrações e relacionamentos definidos no ACROSS, deveria gerar a descrição dos modelos segundo a notação baseada em XML proposta. Também deveria ser capaz de ler um documento XML que descresse a modelagem de um sistema e gerar os desenhos correpondentes. Seria importante que a ferramenta desse ao usuário a opção de visualizar as partes da modelagem por ele desejadas, como determinados agentes ou aspectos, determinados níveis de abstração e de *crosscutting*, implementando para tal os devidos filtros e manipulando o documento XML do modelo. Por fim, seria interessante ligar esta ferramenta às plataformas de implementação a serem definidas, oferecendo opções de geração automática de código.

b) **Desenvolvimento de plataforma de implementação que conjugue agentes e aspectos.**

Apesar da possibilidade de buscar um mapeamento dos modelos gerados a partir das tecnologias existentes, como o JADE e AspectJ, percebeu-se que o ideal seria dispor de plataforma que viabilizasse um mapeamento direto das abstrações e relacionamentos do ACROSS. Desta forma, um trabalho sugerido seria buscar formas de integrar e estender o JADE e

o AspectJ, por exemplo, de modo que fosse possível instanciar agentes, aspectos, objetivos de agente, objetivos de aspecto, planos e ações, bem como explicitar a decomposição em árvore dos objetivos e planos e definir o *crosscutting* baseando-se nos relacionamentos horizontais propostos. A plataforma-resultante asseguraria a rastreabilidade no desenvolvimento, usando comentários em hipertexto que referenciassem partes do modelo. Além disso, deveria ser possível gerar código a partir dos modelos confeccionados.

c) **Experimentação do ACROSS em outros estudos de caso.**

Para verificar a aplicabilidade, identificar as limitações do ACROSS e propor refinamentos poderiam ser desenvolvidos mais estudos de caso, inclusive em conjunto com outras linguagens e metodologias de modelagem de SMAs.

6.3 PALAVRAS FINAIS

Esta dissertação propôs o framework ACROSS para modularização de características transversais na modelagem de Sistemas Multiagentes. Modularizar características transversais em Sistemas Multiagentes é importante, pois reduz o acoplamento, aumenta as possibilidades de reuso e facilita as atividades de evolução e manutenção do software. Além disso, foi capaz de trazer novos questionamentos, ampliando os horizontes de pesquisa em uma área ainda pouco explorada. As oportunidades de continuidade identificadas estimulam a produção de conhecimento científico e tecnológico que explore as pontencialidades dos paradigmas multiagentes e orientado a aspectos durante todo o ciclo de desenvolvimento de software.

7 REFERÊNCIAS BIBLIOGRÁFICAS

AOS. **JACK Intelligent Agents Agent Manual**. http://www.agent-software.com/shared /demos Ndocs/Agent_Manual.pdf, 2005. [capturado em 3 de outubro de 2006].

ARAÚJO, J., BANIASSAD, E., CLEMENTS, P., MOREIRA, A., RASHID, A. e TEKINERDOGAN, B. **Early Aspects: The Current Landscape**. Technical Report COMP-001-2005, Lancaster University, 2005.

ARAÚJO, J., MOREIRA, A., BRITO, I. e RASHID, A. Aspect-oriented requirements with uml. Em KANDÉ, M., ALDAWUD, O., BOOCH, G. E HARRISON, B., editores, **Workshop on Aspect-Oriented Modeling with UML**, 2002. Disponível em http://lglwww.epfl.ch/workshops/uml2002/papers.html [capturado em 08 de novembro de 2006].

BANIASSAD, E. e CLARKE, S. Finding aspects in requirements with theme/doc. Em **Workshop on Early Aspects 2004: Aspect-Oriented Requirements Engineering and Architecture Design**, 2004. Disponível em http://early-aspects.net/ [capturado em 02 de novembro de 2005].

BELLIFEMINE, F., CAIRE, G., TRUCCO, T. e RIMASSA, G. **JADE Programmer's Guide**. http://jade.tilab.com/doc/programmersguide.pdf", 2005. [capturado em 2 de setembro de 2006].

BORDINI, R., BRAUBACH, L., DASTANI, M., SEGHROUCHNI, A. E. F., GOMEZ-SANZ, J., LEITE, J., O'HARE, G., POKAHR, A. e RICCI, A. A survey of programming languages and platforms for multi-agent systems. Em **Informatica** (30), págs. 33–44. 2006. ISSN 0350-5596.

BRAY, T., PAOLI, J., MCQUEEN, S., MALER, E., YERGEAU, F. e COWAN, J. **Extensible Markup Language (XML) 1.1**. http://www.w3.org/TR/2004/REC-xml11-20040204/, 2004. [capturado em 08 de maio de 2006].

BRESCIANI, P., GIORGINI, P., GIUNCHIGLIA, F., MYLOPOULOS, J. e PERINI, A. Tropos: An agent-orientes software development methodology. **International Journal of Autonomous Agents and Multi Agent Systems**, 8(3):203–236, 2004. ISSN 1573-7454.

BRINKKEMPER, J., MYLOPOULOS, J., SOLVBERG, A. e YU, E. Tropos: A framework for requirements-driven software development. Em **Information Systems Engineering: State of the Art and Research Themes**. Springer-Verlag, 2000. ISBN 1-852-33317-0.

BRITO, I., MOREIRA, A. e ARAÚJO, J. A requirements model for quality attributes. Em **Workshop on Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design**, Holanda, 2002. Disponível em

http://trese.cs.utwente.nl/AOSD-EarlyAspectsWS/workshop_papers.htm [capturado em 08 de novembro de 2005].

CAIRE, G., COULIER, W., GARIJO, F. J., GOMEZ, J., PAVON, J., LEAL, F., CHAINHO, P., KEARNEY, P. E., STARK, J., EVANS, R. e MASSONET, P. Agent oriented analysis using message/uml. Em WOOLDRIDGE, M., WEISS, G. E CIANCARINI, P., editores, **Agent-Oriented Software Engineering**, volume 2222 of Lecture Notes in Computer Science, págs. 119–135, 2001. ISBN 3-540-43282-5.

CHAVEZ, C. **Um Enfoque Baseado em Modelos para o Design Orientado a Aspectos**. Tese de Doutorado, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brasil, Abril 2004.

CHITCHYAN, R., RASHID, A., SAWYER, P., GARCIA, A., ALARCON, M. P., BAKKER, J., TEKINERDOGAN, B. e SIOBHÁN CLARKE AND, A. J. **Survey of Aspect-Oriented Analysis and Design Approaches**. Technical Report AOSD-Europe-ULANC-9, AOSD-Europe, Maio 2005.

CHOREN, R. e DE LUCENA, C. J. P. Modeling multi-agent systems with anote. **Software System Modeling Journal**, 4(2):199–208, 2005. ISSN 1619-1374.

CHUNG, L., NIXON, B. A., YU, E. e MYLOPOULOS, J. **Non-Functional Requirements in Software Engineering**. Kluwer Academic Publishers, 1999. ISBN 0-792-38666-3.

DA SILVA, L. F. **Uma Estratégia Orientada a Aspectos para a Modelagem de Requisitos**. Tese de Doutorado, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brasil, Março 2006.

DA SILVA, V. T., CHOREN, R. e DE LUCENA, C. J. P. A uml-based approach for modeling and implementing multi-agent systems. Em **Autonomous Agents and Multi-Agent Systems Conference**, págs. 914–921, New York, EUA, Agosto 2004a. ISBN 1-58113-864-4.

DA SILVA, V. T. **Uma Linguagem de Modelagem para Sistemas Multi-Agentes Baseada em um Framework Conceitual para Agentes e Objetos**. Tese de Doutorado, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brasil, Março 2004b.

DARDENNE, A., VAN LAMSWEERDE, A. e FICKAS, S. Goal-directed requirements acquisition. **Science of Computer Programming**, 20:3–50, 1993. ISSN 0167-6423.

DELOACH, S. A. e WOOD, M. An overview of the multiagent systems engineering methodology. Em CIANCARINI, P. E WOOLDRIDGE, M., editores, **Agent-Oriented Software Engineering**, volume 1957 of Lecture Notes in Computer Science. Springer, 2000. ISBN 3-540-41594-7.

DIJKSTRA, E. W. **A Discipline of Programming**. Prentice-Hall Series in Automatic Computation, Englewood Cliffs:Prentice Hall, 1976. ISBN 0-132-15871-X.

DO PRADO LEITE, J. C. S. Gerenciando a qualidade de software com base em requisitos. Em ROCHA, A., MALDONADO, J. E WEBER, K., editores, **Qualidade de Software: Teoria e Prática**, págs. 238–246. Prentice-Hall, São Paulo, Brasil, 2001. ISBN 8-587-91854-0.

EME. **Manual de Campanha C 6-40 - Técnica de Tiro de Artilharia de Campanha**, volume 1,2. EGGCF, 2001.

FILMAN, R. E. e FRIEDMAN, D. P. Aspect-oriented programming is quantification and obliviousness. Em FILMAN, R., ELRAD, T., CLARKE, S. E AKSIT, M., editores, **Aspect-Oriented Software Development**, chapter 2. Addison-Wesley, 2005. ISBN 0-321-21976-7.

FIPA-OS. **FIPA-OS Developers Guide**. http://fipa-os.sourceforge.net/docs/ Developers_Guide.pdf, 2001. [capturado em 30 de novembro de 2006].

GAMMA, E., HELM, R., JOHNSON, R. e VLISSIDES, P. **Design Patterns: Elements of Reusable Object-Oriented Software**. Addison-Wesley, 1995. ISBN 0-201-63361-2.

GARCIA, A. **Objetos e Agentes: Uma Abordagem Orientada a Aspectos**. Tese de Doutorado, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brasil, Abril 2004a.

GARCIA, A., CHAVEZ, C. e CHOREN, R. An aspect-oriented modeling framework for mas design. Em **7th Workshop on Agent-Oriented Software Engineering**, Japão, 2006a. Disponível em http://www.comp.lancs.ac.uk/computing/aop/papers/garcia_chavez _choren.pdf [capturado em 10 de julho de 2006].

GARCIA, A., CHAVEZ, C. e CHOREN, R. Enhancing agent-oriented models with aspects. Em **5th International Joint Conference on Autonomous Agents and Multi-Agents Systems**, págs. 1332–1334, Japão, 2006b. ISBN 1-59593-303-4.

GARCIA, A. e DE LUCENA, C. J. P. Taming heterogeneous agent architectures with aspects. **Communications of the ACM**, Julho 2006c. [aceito para publicação].

GARCIA, A., KULESZA, U., SANT'ANNA, C., CHAVEZ, C. e DE LUCENA, C. J. P. Aspects in agent oriented software engineering: Lessons learned. Em **Agent Oriented Software Engineering VI**, Lecture Notes in Computer Science, págs. 231–247, Holanda, 2005. Springer Berlin/Heidelberg. ISSN 0302-9743.

GARCIA, A. F., DA SILVA, V. T., CHAVEZ, C. e DE LUCENA, C. J. P. Engineering multi-agent systems with aspects and patterns. **Jounal of Brazilian Computer Society**, 8(1):57–72, 2002. ISSN 0104-6500.

GARCIA, A. F., DE LUCENA, C. J. P. e COWAN, D. D. Agents in object-oriented software engineering. **Software Practice & Experience**, 34(5):489–521, 2004b. ISSN 0038-0644.

GOTEL, O. e FINKELSTEIN, A. An analysis of the requirements tracebility problem. Em **IEEE International Conference on Requirements Engineering**, págs. 94–101, 1994. ISBN 0-8186-5480-5.

IMBEL. **Homepage da IMBEL - Sistema Genesis**. http://www.imbel.gov.br/ index.php?centro=verproduto&id_produto=25&id_categoria=4&lang= pt, 2006. [capturado em 2 de setembro de 2006].

IOERGER, T. R. e HE, L. Modeling command and control in multi-agent systems. Em **8th International Command and Control Research and Technology Symposium**, Washington, DC, EUA, Junho 2003. Disponível em http://dodccrp.org/events/8th_ICCRTS/Tracks/track_4.htm [capturado em 19 de julho de 2005].

JENNINGS, N. R., KINNY, D. e WOOLDRIDGE, M. The gaia methodology for agent-oriented analysis and design. **Journal of Autonomous Agents and Multi-Agent Systems** 3 (3), págs. 285–312, 2000. ISSN 1573-7454.

JENNINGS, N. e WOOLDRIDGE, M. Agent-oriented software engineering. Em **Handbook of Agent Technology**. AAAI/MIT Press, 2001. Disponível em http://www.ecs.soton.ac.uk/ nrj/download-files/agt-handbook.pdf [capturado em 23 de abril de 2005].

JUNIOR, S. M. S. e ROUVELLOU, I. Concern modeling for aspect-oriented software development. Em ET AL, R. F., editor, **Aspect-Oriented Software Development**, chapter 21. Addison-Wesley, 2005. ISBN 0-321-21976-7.

KICZALES, G., HILSDALE, E., HUGUNIN, J., KERSTEN, M., PALM, J. e GRISWOLD, W. G. An overview of aspectj. Em **15th European Conference on Object-Oriented Programming**, págs. 327–353, Londres, Inglaterra, 2001. Springer-Verlag. ISBN 3-540-42206-4.

KICZALES, G., LAMPING, J., MENHDHEKAR, A., MAEDA, C., LOPES, C., LOINGTIER, J.-M. e IRWIN, J. Aspect-oriented programming. Em AKŞIT, P. E MATSUOKA, S., editores, **European Conference on Object-Oriented Programming**, volume 1241, págs. 220–242. Springer-Verlag, Berlim, Alemanha, 1997. ISBN 3-540-63089-9.

KICZALES, G. e MENZINI, M. Aspect-oriented programming and modular reasoning. Em **International Conference on Software Engineering**, págs. 49–58, Saint Louis, EUA, 2005. ISBN 1-59593-963-2.

KULESZA, U., SANT'ANNA, C. e DE LUCENA, C. J. P. Técnicas de projeto orientado a aspectos [cd-rom]. Em **XIX Simpósio Brasileiro de Engenharia de Software - Minicurso**, 2005. [capturado em 10 de outubro de 2005].

LOUGHRAN, N., RASHID, A., CHITCHYAN, R., LEIDENFROST, N., FABRY, J., CACHO, N., GARCIA, A., SANEN, F., TRUYEN, E., WIN, B. D., JOOSEN, W., BOUCKÉ, N., HOLVOET, T., JACKSON, A., NEDOS, A., HATTON, N., MUNNELLY, J., FRITSCH, S., SIOBHÁN CLARKE, M. A., FUENTES, L., PINTO, M. e CANAL, C. **A Domain Analysis of Key Concerns - Known and New**

Candidates. Technical Report AOSD-Europe Deliverable D43, AOSD-Europe-KUL-6, Katholieke Universiteit Leuven, Leuven, Bélgica, Fevereiro 2006.

LUCK, M., MCBURNEY, P. e PREIST, C. A manifesto for agent technology: Towards next generation computing. **Autonomous Agents and Multi-Agent Systems**, 9 (3):203–252, 2004. ISSN 1573-7454.

MOREIRA, A., ARAÚJO, J. e BRITO, I. Crosscutting quality attributes for requirements engineering. Em **14th International Conference on Software Engineering and Knowledge Engineering**, págs. 167–174, Ischia, Itália, 2002. ISBN 1-58113-556-4.

MYLOPOULOS, J., CHUNG, L. e NIXON, B. Representing non-functional requirements: A process-oriented approach. **IEEE Transactions on Software Engineering**, 18 (6):483–497, 1992. ISSN 0098-5589.

NILSSON, N. **Problem Solving Methods in Artificial Intelligence**. McGraw Hill, 1971. ISBN 0-070-46573-8.

NUSEIBEH, B. Crosscutting requirements. Em **3rd International Conference on Aspect-Oriented Software Development**, págs. 3–4, Lancaster, Inglaterra, 2004. ISBN 1-58113-842-3.

ODELL, J., PARUNAK, H. D. e BAUER, B. Extending uml for agents. Em **Agent-oriented Information Systems Workshop**, págs. 3–17, Austin, Texas, USA, 2000. Disponível em http://www.jamesodell.com/ExtendingUML.pdf [capturado em 30 de março de 2005].

OLIVEIRA, A. P. A., CYSNEIROS, L. M., DO PRADO LEITE, J. C. S., FIGUEIREDO, E. M. L. e DE LUCENA, C. J. P. Integrating scenarios, i* and aspectt in the context of multi-agent systems. Em **16th Annual International Conference on Computer Science and Software Engineering**, págs. 16–19, Toronto, Canadá, Outubro 2006. Disponível em http://www.math.yorku.ca/cysneiro/articles/COLOUR%20CameraReady CASCON%20RE-%2096.pdf [capturado em 12 de novembro de 2006].

OMG. **Meta-Object Facility Core Specification, version 2.0**. http://www.omg.org /technology/documents/formal/MOFCore.htm, 2006a. [capturado em 30 de outubro de 2006].

OMG. **Unified Modeling Language: Superstructure, version 2.0**. http://www.omg.org /technology/documents/formal/uml.htm, 2006b. [capturado em 30 de outubro de 2006].

PARK, K., KIM, J. e PARK, S. Goal based agent-oriented software modeling. Em **7th Asia-Pacific Software Engineering Conference**, págs. 320–324, Washington, DC, EUA, 2000. ISBN 0-769-50917-7.

PARNAS, D. L. On the criteria to be used in decomposing systems into modules. **Communications of the ACM**, 15(12):1053–1058, Dezembro 1972. ISSN 0001-0782.

PERES, J. e BERGMANN, U. Experiencing auml for mas modeling: A critical view. Em **I Workshop on Software Engineering for Agent-oriented Systems**, Uberlândia, Brasil, Outubro 2005. Disponível em http://www.les.inf.puc-rio.br/seas2005/file/jPeres.pdf [capturado em 29 de outubro de 2005].

PERES, J., CHOREN, R., CHAVEZ, C. e GARCIA, A. Aspect-oriented modeling in multi-agent systems development. Em ET AL, U. K., editor, **III Workshop Brasileiro de Desenvolvimento de Software Orientado a Aspectos**, págs. 80–89, Florianópolis, Brasil, Outubro 2006. ISBN 8-5766-908-7.

PIVETA, E. K., GARCIA, V. C., ZANCANELLA, L. C. e DO PRADO, A. F. Termos em português para desenvolvimento de software orientado a aspectos. Em **I Workshop Brasileiro de Desenvolvimento Orientado a Aspectos**, págs. 155–156, Brasília, Brasil, Outubro 2004. Disponível em http://twiki.im.ufba.br/pub/WAsp/WASPpublica/ProceedingsWASP2004 - WebVersion.pdf [capturado em 02 de novembro de 2006].

RAO, A. S. e GEORGEFF, M. P. Modeling rational agents within a bdi-architecture. Em ALLEN, J., FIKES, R. E SANDEWALL, E., editores, **2nd International Conference on Principles of Knowledge Representation and Reasoning**, págs. 473–484. Morgan Kaufmann Publishers Inc, San Mateo, CA, USA, 1991. ISBN 1-558-60165-1.

RASHID, A., SAWYER, P., MOREIRA, A. e ARAÚJO, J. Early aspects: a model for aspect-oriented requirements engineering. Em **IEEE Joint International Conference on Requirements Engineering**, Alemanha, 2002. ISBN 0-7695-1465-0.

ROSENHAINER, L. Identifying crosscutting concerns in requirements specification. Em **Workshop on Early Aspects: Aspect-oriented Requirements Engineering and Architecture Design**, 2004. Disponível em http://trese.cs.utwente.nl/workshops/oopsla-early-aspects-2004/Papers/Rosenhainer.pdf [capturado em 30 de outubro de 2005].

RUSSELL, S. e NORVIG, P. **Artificial Intelligence: A Modern Approach**. Prentice-Hall, Englewood Cliffs, NJ, 2003. ISBN 0-137-90395-2.

SAMPAIO, A., LOUGHRAN, N., RASHID, A. e RAYSON, P. Mining aspects in requirements. Em **Workshop on Early Aspects: Aspect-oriented Requirements Engineering and Architecture Design**, 2005. Disponível em http://www.comp.lancs.ac.uk/computing/aopapers/Mining_EA2005.pdf [capturado 17 de novembro de 2005].

SDN. **Javadoc Tool Homepage**. http://java.sun.com/j2se/javadoc/, 2006. [capturado em 2 de setembro de 2006].

SHEN, Z., LI, D., MIAO, C., GAY, R. e MIAO, Y. Goal-oriented methodology for agent system development. Em **IEEE/WIC/ACM International Conference on Intelligent Agent Technology**, págs. 95–101, Washington, DC, EUA, 2005. ISBN 0-7695-2416-8.

SILVA, C., ARAÚJO, J., MOREIRA, A., CASTRO, J., PENAFORTE, D. e CARVALHO, A. Towards an aspect oriented modeling in multi-agent systems. Em ET AL, U. K., editor, **III Workshop Brasileiro de Desenvolvimento de Software Orientado a Aspectos**, págs. 70–80, Outubro 2006a. ISBN 8-5766-908-7.

SILVA, C., ARAÚJO, J., MOREIRA, A., CASTRO, J., TREDESCO, P., ALENCAR, F. e RAMOS, R. Modeling multi-agent systems using uml. Em MASIERO, P. C., editor, **XX Simpósio Brasileiro de Engenharia de Software**, págs. 81–96, Outubro 2006b. ISBN 85-7669-079-9.

SYCARA, K. Multiagent systems. **AI Magazine**, 10(2):79–93, 1998. ISSN 0738-4602.

TARR, P. L., OSSHER, H., HARRISON, W. H. e JR., S. M. S. N degrees of separation: Multi-dimensional separation of concerns. Em **International Conference on Software Engineering**, págs. 107–119, 1999. ISBN 1-58113-074-0.

THAYER, R. H. Software engineering glossary. **IEEE Software**, 20(3), 2003. ISSN 0740-7459.

VAN LANSWEERDE, A. Goal-oriented requirements engineering: a guided tour. Em **International Joint Conference on Requirements Engineering**, págs. 249–263, 2001. ISBN 0-7695-1125-2.

VON STAA, A. **Programação Modular: Desenvolvendo Programas Complexos de Forma Organizada e Segura**. Editora Campus, 2000. ISBN 8-535-20608-6.

WOOLDRIDGE, M. Agent-based software engineering. **IEEE Proceedings on Software Engineering**, 144(1):26–37, 1997. ISSN 1364-5080.

YU, E. S. K. Towards modelling and reasoning support for early-phase requirements engineering. Em **3rd IEEE International Symposium on Requirements Engineering**, págs. 226–235, 1997. ISBN 0-8186-7740-6.

YU, E. S. K. Agent-oriented modelling: Software versus the world. Em WOOLDRIDGE, M., WEISS, G. E CIANCARINI, P., editores, **Agent-Oriented Software Engineering**, volume 2222 of Lecture Notes in Computer Science, págs. 206–225. Springer, 2001. ISBN 3-540-43282-5.

YU, E. S. K. e MYLOPOULOS, J. Understanding 'why' in software process modelling, analysis, and design. Em **16th International Conference on Software Engineering**, págs. 159–168, Los Alamitos, EUA, 1994. ISBN 0-8186-5855-X.

YU, Y., DO PRADO LEITE, J. C. S. e MYLOPOULOS, J. From goals to aspects: Discovering aspects from requirements goal models. Em **12th IEEE International Requirements Engineering Conference**, págs. 38–47, 2004. ISBN 0-7695-2174-6.

8 APÊNDICES

8.1 APÊNDICE 1: MODELAGEM DO AGENTE OBSERVADOR E SEUS ASPECTOS NO SMAC 1.0

```
<MAS>
!- Cabeçalho do projeto -
<head>
  <name>Sistema Multiagentes de Artilharia de Campanha</name>
  <acronym>SMAC</acronym>
  <version>1.0</version>
</head>
!-Fim do cabeçalho -
!- Modelagem do SMAC 1.0 -
<model>
!-Modelagem dos agentes -
   <agent-modeling>
!- Modelagem do agente OA -
     <agent>
       <id>A001</id>
       <name>OA</name>
       <goals>
         <goal>
           <id>AG001</id>
           <subgoals>
             <AND>
               <goal>
                 <id>AG002</id>
               </goal>
               <goal>
                 <id>AG003</id>
               </goal>
               <goal>
                 <id>AG004</id>
               </goal>
               <goal>
                 <id>AG005</id>
                 <subgoals>
                   <AND>
                     <OR>
                       <goal>
                         <id>AG006</id>
                       </goal>
                       <goal>
                         <id>AG007</id>
                       </goal>
                       <goal>
                         <id>AG008</id>
                       </goal>
                     </OR>
                     <goal>
                       <id>AG009</id>
                     </goal>
                   </AND>
                 </subgoals>
               </goal>
             </AND>
           </subgoals>
         </goal>
       </goals>
     </agent>
!-Fim da modelagem do agente Observador -
!-Modelagem dos objetivos dos agentes -
     <agent-goal>
       <id>AG001</id>
       <name>Conduzir MT</name>
     </agent-goal>
```

```
    <agent-goal>
      <id>AG002</id>
      <name>Buscar Alvo</name>
      <plans>
        <plan>
          <id>AP001</id>
        </plan>
      <plans>
    </agent-goal>
    <agent-goal>
      <id>AG003</id>
      <name>Pedir MT</name>
      <plans>
        <plan>
          <id>AP003</id>
        </plan>
      </plans>
    </agent-goal>
    <agent-goal>
      <id>AG004</id>
      <name>Tratar Mensagem Resposta</name>
      <plans>
        <plan>
          <id>AP004</id>
        </plan>
      </plans>
    </agent-goal>
    <agent-goal>
      <id>AG005</id>
      <name>Acompanhar MT</name>
    </agent-goal>
    <agent-goal>
      <id>AG006</id>
      <name>Corrigir tiro</name>
      <plans>
        <plan>
          <id>AP011</id>
        </plan>
      </plans>
    </agent-goal>
    <agent-goal>
      <id>AG007</id>
      <name>Suspender fogo</name>
      <plans>
        <plan>
          <id>AP008</id>
        </plan>
      </plans>
    </agent-goal>
    <agent-goal>
      <id>AG008</id>
      <name>Autorizar fogo</name>
      <plans>
        <plan>
          <id>AP009</id>
        </plan>
      </plans>
    </agent-goal>
    <agent-goal>
      <id>AG009</id>
      <name>Encerrar MT</name>
      <plans>
        <plan>
          <id>AP010</id>
        </plan>
      </plans>
    </agent-goal>
!-Fim da modelagem dos objetivos dos agentes -
!-Modelagem dos planos dos agentes -
    <action-plan>
      <id>AP001</id>
```

```
  <name>Plano de Busca de Alvos</name>
  <actions>
    <AND>
      <composite-plan>
        <id>AP002</id>
      </composite-plan>
      <action>
        <id>AA001</id>
      </action>
      <action>
        <id>AA002</id>
      </action>
    </AND>
  </actions>
</action-plan>
<action-plan>
  <id>AP002</id>
  <name>Plano de Identificação de Alvo</name>
  <actions>
    <AND>
      <action>
        <id>AA003</id>
      </action>
      <action>
        <id>AA004</id>
      </action>
      <action>
        <id>AA005</id>
      </action>
    </AND>
  </actions>
</action-plan>
<action-plan>
  <id>AP003</id>
  <name>Plano de Pedido de Neutralização</name>
  <actions>
    <AND>
      <action>
        <id>AA006</id>
      </action>
      <action>
        <id>AA007</id>
      </action>
      <action>
        <id>AA008</id>
      </action>
    </AND>
  </actions>
</action-plan>
<action-plan>
  <id>AP004</id>
  <name>Plano de Recebimento de Mensagem Resposta</name>
  <actions>
    <action>
      <id>AA009</id>
    </action>
  </actions>
</action-plan>
<action-plan>
  <id>AP005</id>
  <name>Plano de Mudança de Controle</name>
  <actions>
    <action>
      <id>AA010</id>
    </action>
  </actions>
</action-plan>
<action-plan>
  <id>AP006</id>
  <name>Plano de Correção Frontal</name>
  <actions>
```

```
    <XOR>
      <action>
        <id>AA011</id>
      </action>
      <action>
        <id>AA012</id>
      </action>
    </XOR>
      </actions>
</action-plan>
<action-plan>
  <id>AP007</id>
  <name>Plano de Correção Lateral</name>
  <actions>
    <XOR>
      <action>
        <id>AA013</id>
      </action>
      <action>
        <id>AA014</id>
      </action>
    </XOR>
  </actions>
</action-plan>
<action-plan>
  <id>AP008</id>
  <name>Plano de Suspensão de Fogo</name>
  <actions>
    <action>
      <id>AA015</id>
    </action>
  </actions>
</action-plan>
<action-plan>
  <id>AP009</id>
  <name>Plano de Autorização de Fogo</name>
  <actions>
    <action>
      <id>AA016</id>
    </action>
  </actions>
</action-plan>
<action-plan>
  <id>AP010</id>
  <name>Plano de Missão Cumprida</name>
  <actions>
    <AND>
      <action>
        <id>AA017</id>
      </action>
      <action>
        <id>AA018</id>
      </action>
    </AND>
  </actions>
</action-plan>
<action-plan>
  <id>AP011</id>
  <name>Plano de Correção de Tiro</name>
  <actions>
    <OR>
      <composite-plan>
        <id>AP005</id>
      </composite-plan>
      <composite-plan>
        <id>AP006</id>
      </composite-plan>
      <composite-plan>
        <id>AP007</id>
      </composite-plan>
    </OR>
```

```
    </actions>
  </action-plan>
!–Fim da modelagem dos planos dos agentes –
!–Modelagem das ações dos agentes –
  <atomic-action>
    <id>AA001</id>
    <name>Cruzar dados com informações de inteligência</name>
  </atomic-action>
  <atomic-action>
    <id>AA002</id>
    <name>Sugerir reação</name>
  </atomic-action>
  <atomic-action>
    <id>AA003</id>
    <name>Receber coordenadas do alvo</name>
  </atomic-action>
  <atomic-action>
    <id>AA004</id>
    <name>Receber natureza do alvo</name>
  </atomic-action>
  <atomic-action>
    <id>AA005</id>
    <name>Receber dimensões do alvo</name>
  </atomic-action>
  <atomic-action>
    <id>AA006</id>
    <name>Cruzar natureza e dimensões do alvo com nomogramas</name>
  </atomic-action>
  <atomic-action>
    <id>AA007</id>
    <name>Sugerir parâmetros para MT</name>
  </atomic-action>
  <atomic-action>
    <id>AA008</id>
    <name>Receber parâmetros finais do pedido de MT</name>
  </atomic-action>
  <atomic-action>
    <id>AA009</id>
    <name>Receber parâmetros vigentes da MT</name>
  </atomic-action>
  <atomic-action>
    <id>AA010</id>
    <name>Receber comando de 'Eficácia'</name>
  </atomic-action>
  <atomic-action>
    <id>AA011</id>
    <name>Receber distância para alongar o tiro</name>
  </atomic-action>
  <atomic-action>
    <id>AA012</id>
    <name>Receber distância para encurtar o tiro</name>
  </atomic-action>
  <atomic-action>
    <id>AA013</id>
    <name>Receber distância para deslocar o tiro à esquerda</name>
  </atomic-action>
  <atomic-action>
    <id>AA014</id>
    <name>Receber distância para deslocar o tiro à direita</name>
  </atomic-action>
  <atomic-action>
    <id>AA015</id>
    <name>Receber comando de 'Cessar Fogo'</name>
  </atomic-action>
  <atomic-action>
    <id>AA016</id>
    <name>Receber comando de 'Fogo'</name>
  </atomic-action>
  <atomic-action>
    <id>AA017</id>
    <name>Receber comando de 'Missão Cumprida'</name>
```

```
    </atomic-action>
    <atomic-action>
      <id>AA018</id>
      <name>Receber percentual de baixas</name>
    </atomic-action>
!–Fim da modelagem das ações dos agentes –
  </agent-modeling>
!–Fim da modelagem dos agentes –
!–Modelagem dos aspectos –
  <aspect-modeling>
!–Modelagem do aspecto Autonomia –
    <aspect>
      <id>S001</id>
      <name>Autonomia</name>
      <goals>
    <goal>
        <id>SG001</id>
      </goal>
      </goals>
    </aspect>
!–Fim da modelagem do aspecto Autonomia –
!–Modelagem do aspecto Interatividade –
    <aspect>
      <id>S002</id>
      <name>Interatividade</name>
      <goals>
        <OR>
          <goal>
            <id>SG002</id>
          </goal>
          <goal>
            <id>SG003</id>
          </goal>
        </OR>
      </goals>
    </aspect>
!–Fim da modelagem do aspecto Interatividade –
!–Modelagem dos objetivos dos aspectos –
    <aspect-goal>
      <id>SG001</id>
      <name>Tomar decisões</name>
      <plans>
        <plan>
          <id>SP001</id>
        </plan>
      </plans>
    </aspect-goal>
    <aspect-goal>
      <id>SG002</id>
        <name>Receber mensagens</name>
      <plans>
        <plan>
          <id>SP002</id>
        </plan>
      </plans>
    </aspect-goal>
    <aspect-goal>
      <id>SG003</id>
      <name>Enviar mensagens</name>
      <plans>
        <plan>
          <id>SP003</id>
        </plan>
      </plans>
    </aspect-goal>
!–Fim da modelagem dos objetivos dos aspectos –
!–Modelagem dos planos dos aspectos –
    <action-plan>
      <id>SP001</id>
      <name>Plano de Decisões Baseadas em Informações Conhecidas</name>
      <actions>
```

```
      <AND>
        <action>
          <id>SA001</id>
        </action>
        <action>
          <id>SA002</id>
        </action>
        <action>
          <id>SA003</id>
        </action>
      </AND>
    </actions>
  </action-plan>
  <action-plan>
    <id>SP002</id>
    <name>Plano de Recebimento de Mensagens</name>
    <actions>
      <AND>
        <action>
          <id>SA004</id>
        </action>
        <action>
          <id>SA005</id>
        </action>
        <action>
          <id>SA006</id>
        </action>
      </AND>
    </actions>
  </action-plan>
  <action-plan>
    <id>SP003</id>
    <name>Plano de Envio de Mensagens</name>
    <actions>
      <AND>
        <action>
          <id>SA007</id>
        </action>
        <action>
          <id>SA008</id>
        </action>
        <action>
          <id>SA009</id>
        </action>
      </AND>
    </actions>
  </action-plan>
!-Fim da modelagem dos planos dos aspectos -
!-Modelagem das ações dos aspectos -
  <atomic-action>
    <id>SA001</id>
    <name>Receber parâmetros</name>
  </atomic-action>
  <atomic-action>
    <id>SA002</id>
    <name>Buscar resultados mais adequados</name>
  </atomic-action>
  <atomic-action>
    <id>SA003</id>
    <name>Retornar resultados</name>
  </atomic-action>
  <atomic-action>
    <id>SA004</id>
    <name>Esperar por mensagem</name>
  </atomic-action>
  <atomic-action>
    <id>SA005</id>
    <name>Identificar remetente</name>
  </atomic-action>
  <atomic-action>
    <id>SA006</id>
```

```
      <name>Tratar mensagem recebida</name>
    </atomic-action>
    <atomic-action>
      <id>SA007</id>
      <name>Montar mensagem</name>
    </atomic-action>
    <atomic-action>
      <id>SA008</id>
      <name>Identificar destinatário(s)</name>     </atomic-action>
    <atomic-action>
      <id>SA009</id>
      <name>Transmitir mensagem</name>
    </atomic-action>
!- Fim da modelagem das ações dos aspectos -
  </aspect-modeling>
!-Fim da modelagem dos aspectos -
!-Modelagem do crosscutting entre agentes e aspectos -
  <crosscutting-modeling>
!-Crosscutting entre o agente Observador e o aspecto Autonomia -
    <crosscutting>
      <id>CC001</id>
      <agent>
        <id>A001</id>
      </agent>
      <aspect>
        <id>S001</id>
      </aspect>
      <correlations>
        <correlation>
          <id>GC001</id>
        </correlation>
        <correlation>
          <id>GC002</id>
        </correlation>
      </correlations>
    </crosscutting>
!-Fim do Crosscutting entre o agente Observador e o aspecto Autonomia -
!-Crosscutting entre o agente Observador e o aspecto Interatividade -
    <crosscutting>
      <id>CC002</id>
      <agent>
        <id>A001</id>
      </agent>
      <aspect>
        <id>S002</id>
      </aspect>
      <correlations>
        <correlation>
          <id>GC003</id>
        </correlation>
        <correlation>
          <id>GC004</id>
        </correlation>
        <correlation>
          <id>GC005</id>
        </correlation>
        <correlation>
          <id>GC006</id>
        </correlation>
        <correlation>
          <id>GC007</id>
        </correlation>
        <correlation>
          <id>GC008</id>
        </correlation>
      </correlations>
    </crosscutting>
!-Fim do crosscutting entre o agente Observador e o aspecto Interatividade -
!-Correlação entre Buscar Alvo (Observador) e Tomar decisões (Autonomia) -
    <goal-correlation>
      <id>GC001</id>
```

```
      <agent-goal>
        <id>AG002</id>
      </agent-goal>
      <aspect-goal>
        <id>SG001</id>
      </aspect-goal>
      <correlation-link>help</correlation-link>
      <crosscutting-points>
        <point>
          <id>CO001</id>
        </point>
        <point>
          <id>CO002</id>
        </point>
        <point>
          <id>CO003</id>
        </point>
      </crosscutting-points>
    </goal-correlation>
!-Fim da correlação entre Buscar Alvo (Observador) e Tomar decisões (Autonomia) -
!-Correlação entre Pedir MT (Observador) e Tomar decisões (Autonomia) -
    <goal-correlation>
      <id>GC002</id>
      <agent-goal>
        <id>AG003</id>
      </agent-goal>
      <aspect-goal>
        <id>SG001</id>
      </aspect-goal>
      <correlation-link>help</correlation-link>
      <crosscutting-points>
        <point>
          <id>CO004</id>
        </point>
        <point>
          <id>CO005</id>
        </point>
        <point>
          <id>CO006</id>
        </point>
      </crosscutting-points>
    </goal-correlation>
!-Fim da correlação entre Pedir MT (Observador) e Tomar decisões (Autonomia) -
!-Correlação entre Pedir MT (Observador) e Enviar mensagem (Interatividade) -
    <goal-correlation>
      <id>GC003</id>
      <agent-goal>
        <id>AG003</id>
      </agent-goal>
      <aspect-goal>
        <id>SG003</id>
      </aspect-goal>
      <correlation-link>make</correlation-link>
      <crosscutting-points>
        <point>
          <id>CO007</id>
        </point>
      </crosscutting-points>
    </goal-correlation>
!-Fim da correlação entre Pedir MT (Observador) e Enviar mensagem (Interatividade) -
!-Correlação entre Corrigir tiro (Observador) e Enviar mensagem (Interatividade)-
    <goal-correlation>
      <id>GC004</id>
      <agent-goal>
        <id>AG006</id>
      </agent-goal>
      <aspect-goal>
        <id>SG003</id>
      </aspect-goal>
      <correlation-link>make</correlation-link>
      <crosscutting-points>
```

```
      <point>
        <id>CO008</id>
      </point>
    </crosscutting-points>
  </goal-correlation>
!- Fim da correlação entre Corrigir tiro (Observador) e Enviar mensagem (Interatividade) -
!-Correlação entre Encerrar MT (Observador) e Enviar mensagem (Interatividade) -
  <goal-correlation>
    <id>GC005</id>
    <agent-goal>
      <id>AG009</id>
    </agent-goal>
    <aspect-goal>
      <id>SG003</id>
    </aspect-goal>
    <correlation-link>make</correlation-link>
    <crosscutting-points>
      <point>
        <id>CO009</id>
      </point>
    </crosscutting-points>
  </goal-correlation>
!-Fim da correlação entre Encerrar MT (Observador) e Enviar mensagem (Interatividade) -
!-Correlação entre Receber Mensagem Resposta(Observador) e Receber mensagem (Interatividade)-
  <goal-correlation>
    <id>GC006</id>
    <agent-goal>
      <id>AG004</id>
    </agent-goal>
    <aspect-goal>
      <id>SG002</id>
    </aspect-goal>
    <correlation-link>make</correlation-link>
    <crosscutting-points>
      <point>
        <id>CO010</id>
      </point>
    </crosscutting-points>
  </goal-correlation>
!-Fim da correlação entre Receber Mensagem Resposta (Observador) e Receber mensagem (Interatividade) -
!-Correlação entre Autorizar fogo (Observador) e Enviar mensagem (Interatividade) -
  <goal-correlation>
    <id>GC005</id>
    <agent-goal>
      <id>AG008</id>
    </agent-goal>
    <aspect-goal>
      <id>SG003</id>
    </aspect-goal>
    <correlation-link>make</correlation-link>
    <crosscutting-points>
      <point>
        <id>CO011</id>
      </point>
    </crosscutting-points>
  </goal-correlation>
!-Fim da correlação entre Autorizar Fogo (Observador) e Enviar mensagem (Interatividade) -
!-Correlação entre Suspender fogo (Observador) e Enviar mensagem (Interatividade) -
  <goal-correlation>
    <id>GC005</id>
    <agent-goal>
      <id>AG007</id>
    </agent-goal>
    <aspect-goal>
      <id>SG003</id>
    </aspect-goal>
    <correlation-link>make</correlation-link>
    <crosscutting-points>
      <point>
        <id>CO012</id>
      </point>
```

```
    </crosscutting-points>
  </goal-correlation>
!–Fim da correlação entre Suspender Fogo (Observador) e Enviar mensagem (Interatividade) –
!– Operação de crosscutting entre Cruzar dados de inteligência (Observador) e Receber parâmetros (Autonomia) –
  <crosscutting-point>
    <id>CO001</id>
    <agent-element>
      <id>AA001</id>
    </agent-element>
    <aspect-element>
      <id>SA001</id>
    </aspect-element>
    <crosscutting-relationship>redefine</crosscutting-relationship>
  </crosscutting-point>
!– Fim da operação de crosscutting entre Cruzar dados de inteligência (Observador) e Receber parâmetros (Autonomia)–
!– Operação de crosscutting entre Cruzar dados de inteligência (Observador) e Buscar resultados mais adequados (Autonomia)–
  <crosscutting-point>
    <id>CO002</id>
    <agent-element>
      <id>AA001</id>
    </agent-element>
    <aspect-element>
      <id>SA002</id>
    </aspect-element>
    <crosscutting-relationship>redefine</crosscutting-relationship>
  </crosscutting-point>
!– Fim da operação de crosscutting entre Cruzar dados de inteligência (Observador) e Buscar resultados mais adequados
(Autonom.)–
!– Operação de crosscutting entre Sugerir reação (Observador) e Retornar resultados (Autonomia)–
  <crosscutting-point>
    <id>CO003</id>
    <agent-element>
      <id>AA002</id>
    </agent-element>
    <aspect-element>
      <id>SA003</id>
    </aspect-element>
    <crosscutting-relationship>refine before</crosscutting-relationship>
  </crosscutting-point>
!– Fim da Operação de crosscutting entre Sugerir reação(Observador) e Retornar resultados(Autonomia)–
!– Operação de crosscutting entre Cruzar nat. e dim. do alvo com nomogramas(Observador) e Receber parâmetros (Autonomia)–
  <crosscutting-point>
    <id>CO004</id>
    <agent-element>
      <id>AA006</id>
    </agent-element>
    <aspect-element>
      <id>SA001</id>
    </aspect-element>
    <crosscutting-relationship>redefine</crosscutting-relationship>
  </crosscutting-point>
!– Fim da operação de crosscutting entre Cruzar nat. e dim. do alvo com nomogramas(Observ.) e Receber parâmetros (Autonom.)–
!– Operação de crosscutting entre Cruzar nat. e dim. do alvo com nomogramas(Observ) e Buscar resultados + adequados
(Autonom)–
  <crosscutting-point>
    <id>CO005</id>
    <agent-element>
      <id>AA006</id>
    </agent-element>
    <aspect-element>
      <id>SA002</id>
    </aspect-element>
    <crosscutting-relationship>redefine</crosscutting-relationship>
  </crosscutting-point>
!– Fim da op de crosscutting entre Cruzar nat. e dim. do alvo com nomogramas(Observ) e Buscar resultados + adequados
(Autonom)–
!– Operação de crosscutting entre Sugerir parâmetros da MT(Observador) e Retornar resultado (Autonomia)–
  <crosscutting-point>
    <id>CO006</id>
    <agent-element>
      <id>AA007</id>
```

```
      </agent-element>
      <aspect-element>
        <id>SA003</id>
      </aspect-element>
      <crosscutting-relationship>refine before</crosscutting-relationship>
    </crosscutting-point>
!- Fim da operação de crosscutting entre Sugerir parâmetros da MT(Observador) e Retornar resultado (Autonomia)-
!-Operação de crosscutting entre Plano de Pedido de Neutralização(Observador) e Plano de Envio de Mensagens(Interat.)-
    <crosscutting-point>
      <id>CO007</id>
      <agent-element>
        <id>AP003</id>
      </agent-element>
      <aspect-element>
        <id>SP003</id>
      </aspect-element>
      <crosscutting-relationship>add after</crosscutting-relationship>
    </crosscutting-point>
!-Fim da operação de crosscutting entre Plano de Pedido de Neutralização(Observador) e Plano de Envio de Mensagens(Interat)-
!-Operação de crosscutting entre Plano de Correção de Tiro(Observador) e Plano de Envio de Mensagens(Interatividade)-
    <crosscutting-point>
      <id>CO008</id>
      <agent-element>
        <id>AP011</id>
      </agent-element>
      <aspect-element>
        <id>SP003</id>
      </aspect-element>
      <crosscutting-relationship>add after</crosscutting-relationship>
    </crosscutting-point>
!-Fim da operação de crosscutting entre Plano de Correção de Tiro(Observador) e Plano de Envio de Mensagens(Interatividade)-
!-Operação de crosscutting entre Plano de Missão Cumprida(Observador) e Plano de Envio de Mensagens(Interatividade)-
    <crosscutting-point>
      <id>CO009</id>
      <agent-element>
        <id>AP010</id>
      </agent-element>
      <aspect-element>
        <id>SP003</id>
      </aspect-element>
      <crosscutting-relationship>add after</crosscutting-relationship>
    </crosscutting-point>
!-Fim da Operação de crosscutting entre Plano de Missão Cumprida(Observador) e Plano de Envio de Mensagens(Interatividade)-
!-Operação de crosscutting entre Plano de Recebimento de MRO(Observador) e Plano de Recebimento de Mensagens(Interat.)-
    <crosscutting-point>
      <id>CO010</id>
      <agent-element>
        <id>AP004</id>
      </agent-element>
      <aspect-element>
        <id>SP002</id>
      </aspect-element>
      <crosscutting-relationship>add before</crosscutting-relationship>
    </crosscutting-point>
!-Fim da operação de crosscutting entre Plano de Recebimento de MRO(Observ) e Plano de Recebimento de Mensagens(Interat)-
!-Operação de crosscutting entre Plano de Suspensão de Fogo(Observador) e Plano de Envio de Mensagens(Interat)-
    <crosscutting-point>
      <id>CO011</id>
      <agent-element>
        <id>AP008</id>
      </agent-element>
      <aspect-element>
        <id>SP002</id>
      </aspect-element>
      <crosscutting-relationship>add after</crosscutting-relationship>
    </crosscutting-point>
!-Fim da operação de crosscutting entre Plano de Suspensão de Fogo(Observador) e Plano de Envio de Mensagens(Interat)-
!-Operação de crosscutting entre Plano de Autorização de Fogo(Observador) e Plano de Envio de Mensagens(Interat)-
    <crosscutting-point>
      <id>CO012</id>
      <agent-element>
```

```
        <id>AP009</id>
      </agent-element>
      <aspect-element>
        <id>SP002</id>
      </aspect-element>
      <crosscutting-relationship>add after</crosscutting-relationship>
    </crosscutting-point>
!–Fim da operação de crosscutting entre Plano de Autorização de Fogo(Observador) e Plano de Envio de Mensagens(Interat)–
  </crosscutting-modeling>
!–Fim da modelagem de crosscutting –
</model>
!–Fim da modelagem do SMAC 1.0 –
</MAS>
```

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo