UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

PABLO SOUZA GRIGOLETTI

Uma Arquitetura Baseada em Sistemas Multiagentes para Simulações em Geoprocessamento

Dissertação apresentada como requisito parcial
para a obtenção do grau de
Mestre em Ciência da Computação

Prof. Dr. Antônio Carlos da Rocha Costa
Orientador

Porto Alegre, abril de 2007

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Grigoletti, Pablo Souza

Uma Arquitetura Baseada em Sistemas Multiagentes para Simulações em Geoprocessamento / Pablo Souza Grigoletti. – Porto Alegre: PPGC da UFRGS, 2007.

132 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2007. Orientador: Antônio Carlos da Rocha Costa.

1. Sistemas multiagentes. 2. Geoprocessamento. 3. Simulações. I. Costa, Antônio Carlos da Rocha. II. Título.

Dedico este trabalho à minha família,
Sylvio, Lúcia e Shana.

...
Quis a "mala suerte" ingrata
Que eu errasse aquele pealo
E que rodasse o cavalo
"Virge" quase que me mata.
Mas eu como sou vaqueano
Cruzei a perna ligeiro
Só escutei o "entrevero"
De pingo, terra e boléu...
Quando finco meu chapeú
Bem debochado na nuca
Nem diabo, nem arapuca
Me "cambeiam" lá pro céu.
...
Pressentindo a coisa feia
Virei o mango na mão
E aprumei o "pitangueira"
Bem no miolo do tirano.
Nisso já vinha meu zaino
Me procurando no espaço...
Coisas da lida de laço
Pra quem anda de a cavalo.
Se eu não derrubo de um pealo
Derrubo a cusco e mangaço.

— Anomar Danúbio Vieira
(trechos da música "A Cusco e Mangaço")

AGRADECIMENTOS

A *Deus* por conceder-me o dom da vida; por iluminá-la até este momento e permitir-me finalizar mais esta etapa; por estar junto de mim nos momentos mais difíceis.

Aos meus pais, *Sylvio* e *Lúcia*, pelo apoio em todos os passos desta caminhada; pela compreensão nos momentos em que me fiz ausente de suas vidas. À minha irmã *Shana* por me propiciar momentos familiares, mesmo estando eu tão longe de casa; pelas inúmeras vezes que me encomoda sem parar (*Ai jura, Disney Disney*).

Os meus sinceros e profundos agradecimentos ao professor *Rocha Costa*, por ter me dado a oportunidade de fazer o curso de mestrado; pela ajuda e dedicação, guiando minha caminhada; por ser um orientador exemplar.

À minha namorada, *Graciliana*, por sempre estar próxima, seja ajudando ou encomodando (*brincadeirinha*). Teu amor me dá força e certeza de que tudo é possível.

Aos meus companheiros de graduação, *Eduardo Bastos* (Buba), *Juliano Freitas* (Cazuzinha) e *Eduardo Menna* (Polenta), pela força e companheirismo em todos os momentos; por possibilitarem que compartilhássemos ânsias, dificuldades, alegrias e vitórias nesta etapa de vida (mestrado), a qual todos nós enfrentamos juntos; pela parceria de hoje e sempre. Ao *Eduardo Bastos*, pelas incontáveis vezes que me ajudou (trabalhos de SCA, ZMAS, etc); pela amizade de longa data (SJ, Yázigi, UCPel, UFRGS, SISNEMA); por sempre estar disposto a "*trocar uma idéia alí no bar*"; por ser meu "grupo de pesquisa" na UFRGS. Ao *Juliano* por todas as azarações juntos; por ser um verdadeiro irmão nestes dois anos de convivência (*apê* 338). Ao *Eduardo Menna*, por ser o grande amigo de sempre, mesmo morando tão longe. A amizade de vocês é muito importante para mim.

Aos amigos que levo sempre comigo dentro do peito, *Rodrigo Fagundes* (Negão), *Rafael Fagundes* (Gczão), *Rafael Mendes* (Presidente), *Diego Menna* (Badeco), *Cristóvão* (Basper), *Sabrina* (Lima) e *Henrique* (Piru), por entenderem minhas ausências e me darem, sempre que possível, a esperança de que "*existe vida após o mestrado*"; por serem sempre únicos e especiais. Ao *Rodrigo*, *Rafael*, *Sabrina* e *Piru*, por este longo tempo de amizade (dos tempos do Laranjal). Ao *Cristóvão*, pelas noites de viola, pagode e gelada. Ao *Diego* e ao *Rafael Mendes*, por, mesmo sendo os dois amigos mais encolhidos que tenho, serem grandes amigos.

Aos amigos de laboratório, *Márcia*, *Fábio* e *Juliana*, por me proporcionarem um ótimo ambiente de trabalho; por me ajudarem sempre que precisei.

Aos professores da UCPel, *Marilton*, *Rocha*, *Graça*, *Renata*, *Luciano*, *Daniel*, *Barbosa* e *Adenauer*, por serem sempre referências na minha vida profissional e pessoal; por, mesmo que a distância, estarem muito presentes em minha caminhada.

Aos professores e funcionários do PPGC, por toda a ajuda durante estes dois anos de mestrado. Ao CNPq, pela bolsa, sem a qual seria impossível realizar esta etapa. A todas as pessoas que participaram, direta ou indiretamente, desta fase.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

AC	Autômato Celular
ACL	*Agent Communication Language*
ACs	Autômatos Celulares
API	*Application Program Interface*
BDG	Banco de Dados Geográficos
BDGs	Bancos de Dados Geográficos
BDI	*Belief-Desire-Intention*
CAD	*Computer Aided Design*
ESRI	*Environmental Systems Research Institute, Inc.*
FIPA	*Foundation for Intelligent Physical Agents*
GAS	*Geographic Automata Systems*
GEOS	*Geometry Engine, Open Source*
GPS	Sistema de Posicionamento Global
JTS	*JTS Topology Suite*
IA	Inteligência Artificial
IAD	Inteligência Artificial Distribuída
MBA	Modelos Baseados em Agentes
MBE	Modelos Baseados em Equações
MER	Modelo Entidade-Relacionamento
OBEUS	*Object-Based Environment for Urban Systems*
OGC	*Open Geospatial Consortium*
POO	Programação Orientada a Objetos
RDP	Resolução Distribuída de Problemas
Repast	*Recursive Porous Agent Simulation Toolkit*
ROAD	*Repast Organization for Architecture and Design*
SGBD	Sistema de Gerência de Bancos de Dados

SGBDG	Sistema de Gerência de Bancos de Dados Geográficos
SIG	Sistema de Informação Geográfica
SIGs	Sistemas de Informação Geográfica
SFS	*Simple Features Specification*
SMA	Sistema Multiagente
SMAs	Sistemas Multiagentes
UML	*Unified Modeling Language*
VHLL	*Very High Level Language*
XML	*Extensible Markup Language*
ZMAS	*ZOPE MultiAgent Systems*
ZOPE	*Z Object Publishing Environment*

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Este trabalho situa-se nas áreas de Sistemas Multiagentes e Geoprocessamento. Em Sistemas Multiagentes, a especificação de um sistema pode ser realizada através da modelagem do ambiente, agentes, mecanismos de interação e organização dos agentes envolvidos. Um aspecto pouco explorado, porém extremamente importante, é a modelagem e representação do ambiente em que os agentes estão situados e através do qual irão interagir.

Por outro lado, o Geoprocessamento sempre enfatizou a representação de fenômenos espaciais de forma estática. No entanto, alguns fenômenos espaciais são inerentemente dinâmicos e as representações estáticas não os capturam de forma adequada.

Assim, o foco principal deste trabalho é fornecer uma arquitetura, baseada em Sistemas Multiagentes, para a criação e execução de simulações na área de Geoprocessamento. Uma característica importante é a utilização de dados vetoriais, provenientes de um Banco de Dados Geográficos, na geração da representação espacial do ambiente e das entidades existentes nas simulações. Com esta forma de representação contínua e precisa, é possível criar modelos mais próximos da realidade, representando adequadamente um maior número de características. Além disto, a arquitetura permite a representação de fenômenos espaço-temporais dinâmicos, uma necessidade da área de Geoprocessamento.

Focando no desenvolvimento das funcionalidades da arquitetura proposta, foi realizada uma análise das características positivas, negativas e das necessidades de algumas das principais plataformas para criação e execução de simulações baseadas em agentes. Considerando os resultados desta análise, foram criadas a arquitetura proposta e suas funcionalidades. Além disto, neste trabalho é apresentado o protótipo implementado, no qual foram realizados estudos de caso de diferentes cenários, objetivando avaliar e demonstrar o uso das funcionalidades desenvolvidas.

Palavras-chave: Sistemas multiagentes, geoprocessamento, simulações.

An Architecture Based on Multi-Agent Systems for Simulations in Geocomputing

ABSTRACT

This work is situated in the intersection of two areas: Multi-Agent Systems and Geocomputing. In Multi-Agent Systems, the specification of a system can be achieved by the modelling of environment, agents, mechanisms of interaction and organization of the agents. Although being a very important aspect, the modelling and representation of the environment has not been yet fully explored.

On the other hand, Geocomputing has always focused in the static representation of spatial phenomena. However, some spatial phenomena are dynamic on time and space and usual representation adopted in Geocomputing do not capture them well.

The aim of this work is to provide an architecture, based on Multi-Agent Systems, for the development and execution of Geocomputing simulations. A very important feature of the proposed architecture is the use of vectorial data, from a Geographic Database, for the representation of the environment and the spatial entities that exist in it. Using this continuous and accurate representation form, it is possible to develop more realistic models, representing appropriately geographic features. Moreover, this architecture allows the representation of dynamic phenomena in time and space, an old need of the Geocomputing area.

Focusing on the development of the features of this architecture, an analysis of some related works was realized. The development of this architecture and its features was done based on the results of this analysis. Moreover, a prototype was presented in this work, in which some case studies of different scenes were performed, aiming at the evaluation and demonstration of the developed model and its features.

Keywords: Multi-agent systems, geocomputing, simulations.

1 INTRODUÇÃO

Estudos sobre as relações entre o homem e o meio ambiente estão cada dia mais presentes na comunidade científica. Neste contexto, a Ciência da Computação tem contribuído intensamente, fornecendo importantes ferramentas para a representação, visualização e análise destas relações.

Neste sentido, diversos modelos computacionais têm sido desenvolvidos para estudar os complexos sistemas sociais, biológicos e físicos que compõem tais relações. A simulação computacional tem sido amplamente utilizada como ferramenta para a análise e o entendimento dos comportamentos emergentes das ações do homem sobre o meio ambiente. Estas simulações permitem, principalmente, a compreensão de sistemas complexos de comportamento caótico, nos quais métodos matemáticos puramente analíticos não tem sido suficientes. Sendo assim, freqüentemente têm sido utilizados modelos baseados em Sistemas Multiagentes (SMAs) para a criação destas simulações.

Por outro lado, o estudo de fenômenos envolvendo o conceito de espaço desencadeou, nas últimas décadas, o desenvolvimento de sistemas computacionais específicos para modelagem e análise do espaço geográfico: os Sistemas de Informação Geográfica (SIGs). Um SIG nada mais é do que um sistema de informação que torna possível a captura, modelagem, manipulação, recuperação, análise e apresentação de dados referenciados geograficamente[1] (WORBOYS, 1995). Em um SIG, o componente de armazenamento é comumente denominado Banco de Dados Geográficos (BDG).

1.1 Motivações

As principais motivações para o desenvolvimento deste trabalho são destacadas abaixo:

- conforme descrito em (OKUYAMA, 2003), a modelagem e a representação do ambiente em que os agentes estão situados e através do qual irão interagir são aspectos extremamente importantes, no entanto pouco explorados na área de SMAs. De forma mais geral, quando um SMA é um sistema computacional completo, a modelagem do ambiente compartilhado é parte essencial do seu projeto e desenvolvimento, pois permite a análise de mecanismos de interação de maneira detalhada, muitas vezes com mais detalhes que em aplicações situadas em um ambiente real (FERBER, 1999);

- historicamente, o Geoprocessamento enfatizou a representação de fenômenos espaciais de forma estática. Isto se deve ao fato de que a principal abstração utilizada em

[1] Dados georeferenciados, ou dados referenciados geograficamente, são dados que descrevem fenômenos geográficos cuja localização está associada a uma posição sobre/sob a superfície terrestre (LISBOA FILHO, 2000).

um SIG é o mapa. No entanto, diversos fenômenos espaciais são inerentemente dinâmicos e as representações estáticas, comumente utilizadas neste contexto, não os capturam de forma adequada. Sendo assim, o desenvolvimento de abstrações que sejam capazes de representar adequadamente fenômenos espaço-temporais dinâmicos ainda é um grande desafio na área de Geoprocessamento (PEDROSA; CÂMARA, 2004);

- atualmente, os modelos computacionais de simulação em Geoprocessamento têm sido amplamente utilizados no estudo de sistemas complexos caracterizados pelas relações entre o homem e o meio em que este vive. Estes modelos têm como principal objetivo estudar o comportamento dos fenômenos emergentes da tais relações. Exemplos de fenômenos resultantes da interação entre o homem e o meio ambiente são: crescimento/desmatamento florestal, processos hidrológicos, propagação de fogo em florestas, crescimento/desenvolvimento urbano, mudanças no uso do solo, erosão do solo, etc. Todos estes processos evoluem tanto no tempo quanto no espaço. A utilização de agentes no contexto de simulações em Geoprocessamento é recente. Há algum tempo, na maioria das vezes Autômatos Celulares (ACs) eram utilizados. No entanto, a fundamentação teórica dos ACs tradicionais não possibilitava que os autômatos das células se deslocassem pelo cenário que estava sendo modelado, só as informações trocadas entre eles é que se movimentavam. Já utilizando uma abordagem baseada em agentes, é possível que as informações e os próprios autômatos (neste caso, os agentes) se movimentem, proporcionando uma modelagem muito mais realística;

- reforçando a necessidade de ferramentas para modelar o espaço e os fenômenos dinâmicos atuantes sobre este, Santos (1996) descreve o espaço como indivisível dos seres humanos que o habitam e que o modificam a todo momento. Criando uma visão nova e mais geral sobre os conceitos de espaço, Santos (1996) afirma também que "*o espaço geográfico é um sistema de objetos e um sistema de ações*". Esta caracterização objetiva contrapor os elementos de composição do espaço (os objetos geográficos estáticos que representam o mundo real) aos condicionantes de modificação da estrutura deste espaço (as ações humanas e os processos físicos ao longo do tempo). Este conceito de espaço enfatiza a necessidade de unir a visão estática do espaço aos processos variantes no tempo, como parte essencial do espaço. Baseado neste conceito, é possível definir a diferença entre espaço e paisagem: a paisagem é o conjunto de formas que num dado momento exprime as heranças que representam as sucessivas relações entre o homem e o meio ambiente; o espaço é formado por estas formas e a vida que as anima.

1.2 Objetivos

Baseado nas motivações apresentadas acima, o objetivo principal deste trabalho é fornecer um modelo de arquitetura computacional baseada nos conceitos de SMAs, para a criação e realização de simulações na área de Geoprocessamento. Pretende-se analisar e explorar as vantagens de integrar as áreas de SMAs e Geoprocessamento no contexto da criação de modelos e da realização de simulações de sistemas complexos e dinâmicos, focando na representação espacial do ambiente e das entidades que modificam este ambiente.

A idéia é agregar funcionalidades à arquitetura proposta, de modo a permitir que os

usuários finais se preocupem somente com a lógica de sua aplicação, desenvolvendo os agentes para tal finalidade.

Neste contexto, um BDG é utilizado para fornecer os dados precisos sobre a forma e a localização das entidades existentes na simulação, principalmente. Desta forma, as entidades de uma simulação representam dinamicamente os dados estáticos do BDG.

1.3 Contribuições

De forma geral, as contribuições deste trabalho focam principalmente em duas áreas:

- na área de SMAs, fornecer uma arquitetura de simulação que possibilite a modelagem espacial do ambiente e das entidades de forma mais próxima da realidade, utilizando estruturas da área de Geoprocessamento, mais precisamente dados provenientes de um BDG;

- na área de Geoprocessamento, fornecer uma abstração para a representação de eventos e processos espaço-temporais dinâmicos, utilizando simulações baseadas em SMAs.

A contribuição final, pode ser descrita como sendo: fornecer a estrutura necessária para permitir que os SMAs sejam utilizados em simulações que façam uso de modelos geográficos gerados por BDGs, e que os SMAs também possam gerar dados espaciais para serem utilizados pelos BDGs. Desta forma, tanto os modelos geográficos quanto os modelos baseados em SMAs podem se beneficiar do acoplamento entre SMA-BDG, proposto neste trabalho.

1.4 Organização do Documento

Este documento está estruturado como apresentado a seguir. O Capítulo 2 apresenta a área de Geoprocessamento, suas ferramentas e os principais modelos e estruturas para a representação de dados espaciais do mundo real, nos sistemas computacionais. No Capítulo 3 é descrita a área de Inteligência Artificial Distribuída, bem como discutidos os conceitos de agentes e Sistemas Multiagentes, suas arquiteturas, modelos, ambientes e formas de comunicação. O Capítulo 4 aborda a área das simulações, revisando definições, principais conceitos, objetivos, etapas e áreas de aplicação. No Capítulo 5 são apresentadas as características de algumas das principais plataformas para criação e execução de simulações baseadas em agentes. Também é realizada uma comparação destas plataformas. O Capítulo 6 descreve as características do modelo proposto, uma arquitetura baseada em Sistemas Multiagentes para simulações na área de Geoprocessamento. O Capítulo 7 descreve as diversas características e funcionalidades do protótipo desenvolvido para avaliar o modelo de arquitetura proposto. Também são descritas as tecnologias utilizadas neste desenvolvimento. No Capítulo 8 são apresentados alguns estudos de caso realizados para demonstrar a usabilidade e funcionalidade do modelo proposto. No Capítulo 9 são descritas as considerações finais deste trabalho, bem como sugestões de trabalhos futuros. A organização deste documento é ilustrada na Figura 1.1:

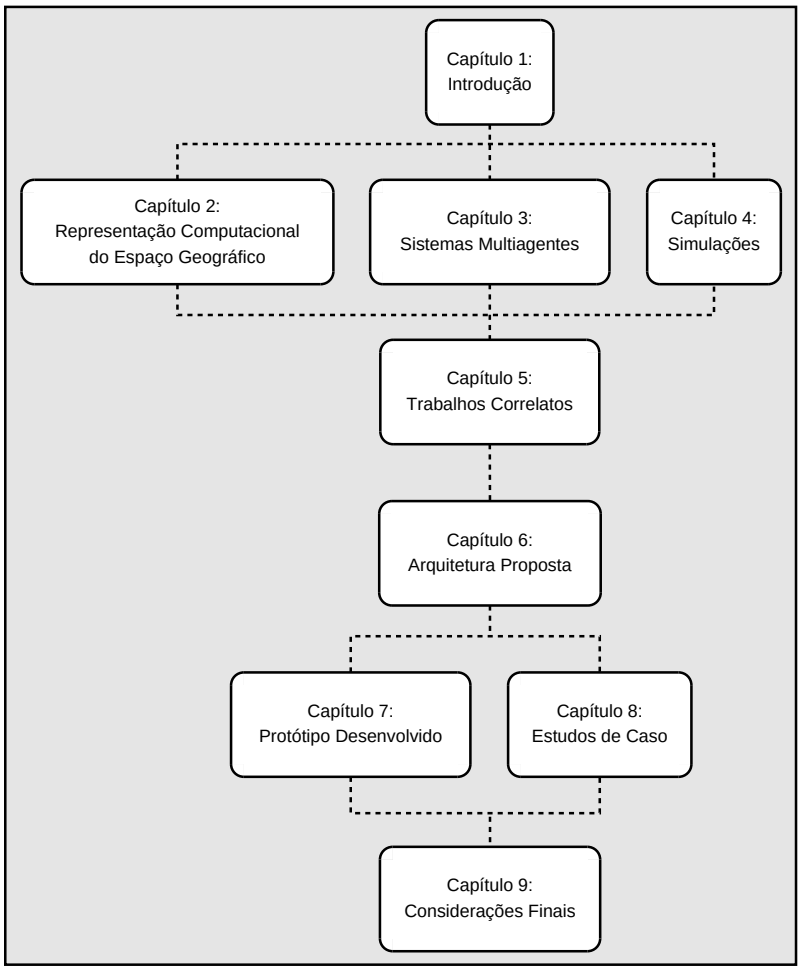

Figura 1.1: Organização deste documento.

2 REPRESENTAÇÃO COMPUTACIONAL DO ESPAÇO GEOGRÁFICO

Este capítulo apresenta sucintamente a área de Geoprocessamento, destacando as principais características de duas de suas principais ferramentas: o Sistema de Informação Geográfica (SIG) e o Banco de Dados Geográficos (BDG). Também é focada a apresentação de modelos, estruturas e técnicas para a representação de dados espaciais do mundo real nos sistemas computacionais.

2.1 Introdução

Antigamente, o armazenamento de informações geográficas era realizado utilizando apenas documentos e mapas em papel, dificultando assim a realização de análises complexas que combinassem uma grande quantidade de dados. A evolução na área da computação permitiu que o armazenamento e a representação de tais informações fossem realizados computacionalmente, abrindo espaço para o aparecimento do Geoprocessamento.

Neste contexto, a recente popularização das técnicas de Geoprocessamento tem gerado algumas confusões na atribuição dos termos Geoprocessamento e Sistema de Informação Geográfica, que vêm sendo utilizados como sinônimos quando, na verdade, dizem respeito a entidades distintas.

De acordo com Carvalho, Pina e Santos (2000), o Geoprocessamento é um termo amplo, que engloba diversas tecnologias de tratamento e manipulação de dados geográficos, através de ferramentas computacionais. Já Câmara e Davis (2001) descrevem o termo Geoprocessamento como a disciplina do conhecimento que utiliza técnicas matemáticas e computacionais para o tratamento de informações geográficas.

Sendo uma das principais ferramentas computacionais utilizadas na área de Geoprocessamento, o SIG permite a realização de análises complexas, ao integrar dados de diversas fontes, e a criação de bancos de dados georeferenciados.

No entanto, além dos SIGs, diversas outras tecnologias são utilizadas na área de Geoprocessamento, dentre as quais destacam-se: o sensoriamento remoto, a digitalização de dados, a automação de tarefas cartográficas, a utilização do GPS (Sistema de Posicionamento Global), etc.

2.2 Sistemas de Informação Geográfica

Nos diversos trabalhos que envolvem o conceito de SIG, um fato constante é a complexidade de sua própria definição. Tal fato pode ser observado na diversidade de definições existentes. Em (MAGUIRE, 1991) são expostos os motivos para tal complexidade

e diversidade, além de algumas definições para o termo SIG. Dentre as várias definições existentes, algumas são apresentadas abaixo:

- "*Um poderoso conjunto de ferramentas para coletar, armazenar, recuperar, transformar e apresentar dados espaciais do mundo real.*" (BURROUGH, 1986);
- "*Um sistema de suporte à decisão que integra dados referenciados espacialmente num ambiente de respostas a problemas.*" (COWEN, 1988);
- "*Um conjunto manual ou computacional de procedimentos utilizados para armazenar e manipular dados georreferenciados.*" (ARONOFF, 1989);
- "*Um sistema de informação que torna possível a captura, modelagem, manipulação, recuperação, análise e apresentação de dados referenciados geograficamente.*" (WORBOYS, 1995);
- "*Um sistema que realiza o tratamento computacional de dados geográficos, permitindo o gerenciamento destes dados, o processamento de imagens, a produção de gráficos, a modelagem espacial, a visualização dos mais variados tipos de dados e a automatização da produção de documentos cartográficos.*" (CÂMARA; MEDEIROS, 1998).

Segundo Câmara e Queiroz (2001), as diferentes definições de SIG refletem, cada uma à sua maneira, a multiplicidade de usos e visões possíveis desta tecnologia e apontam perspectivas interdisciplinares para sua utilização. A partir destes conceitos é possível definir, de forma geral, um SIG como sendo um sistema computacional, essencial para a área de Geoprocessamento, que tem como funções adquirir, armazenar, manipular, analisar e visualizar dados sob três aspectos:

- **dados geográficos:** aqueles definidos espacialmente e geralmente representados através de mapas. Descrevem a localização, as feições geográficas, ou seja, a descrição gráfica;
- **suas características, ou atributos descritivos:** normalmente compostos por valores alfanuméricos armazenados em forma de tabelas. Descrevem os fatos e fenômenos, sociais e naturais, representados nos mapas;
- **as relações espaciais entre os elementos:** podem ser relações topológicas, métricas ou de ordem.

2.2.1 Estrutura Geral

Em uma visão geral, Câmara e Medeiros (1998) definem que a estrutura de um SIG é dividida em:

- interface com o usuário;
- entrada e integração de dados;
- funções de processamento gráfico e de imagens;
- visualização e plotagem;

- armazenamento e recuperação de dados (Banco de Dados Geográficos).

Estes componentes se relacionam de forma hierárquica, como apresentado na Figura 2.1. No nível mais externo, a interface com o usuário descreve como o sistema é operado e controlado. No nível intermediário, devem existir mecanismos de processamento de dados espaciais. A entrada de dados inclui os mecanismos de conversão de dados. Os algoritmos de consulta e análise espacial incluem as operações topológicas, álgebra de mapas, estatística espacial, modelagem numérica de terreno e processamento de imagens. Os mecanismos de visualização e plotagem devem oferecer suporte adequado para a apreensão dos aspectos relevantes dos dados pesquisados. Devido à necessidade de gerenciamento de grandes volumes de dados a maioria dos SIGs trabalha de forma integrada com um Sistema de Gerência de Bancos de Dados Geográficos (SGBDG) (LISBOA FILHO, 2000). Essa integração ocorre no nível mais interno do SIG e fornece funcionalidades de armazenamento e recuperação dos dados espaciais e seus atributos.

De acordo com os objetivos e necessidades de cada sistema, os componentes apresentados são implementados de forma distinta, mas todos os subsistemas citados devem estar presentes em um SIG (CÂMARA, 2005).

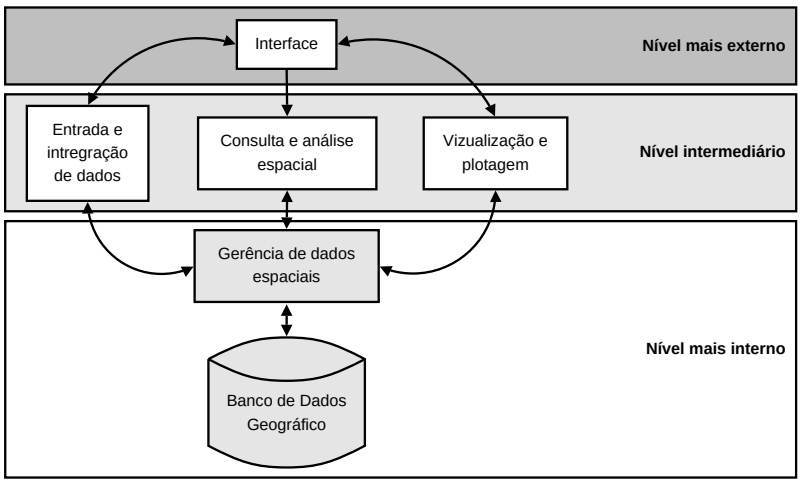

Figura 2.1: Estrutura geral dos SIGs.

2.2.2 Funções Principais

A estrutura dos SIGs possibilita a realização de três funções principais:

- o armazenamento, manejo e integração de grandes quantidades de dados referenciados espacialmente. Um dado espacialmente referenciado pode ser concebido como contendo dois tipos de informações: dados de atributos e dados de localização. Dados cartográficos ou de localização são coordenadas de pontos (nós) bi ou tridimensionais, linhas (segmentos) e áreas (polígonos). Dados descritivos ou não-localizados são as características ou os atributos destes pontos, linhas e áreas;

- prover meios para realizar análises relacionadas especificamente às componentes geográficas dos dados. As operações mais comuns são a pesquisa de dados e a busca de informações de acordo com algum critério de seleção (por exemplo, pela localização, proximidade, tamanho, valor, etc), e a análise espacial que envolve modelagem e análise de padrões espaciais e de relacionamento de dados;

- a organização e o manejo de grandes quantidades de dados e a forma como estas informações podem ser facilmente acessadas por todos os usuários. Um SIG precisa ser ágil para exibir dados em mapas de boa qualidade. Entretanto, para diferentes propósitos, outras formas de apresentação dos dados (gráficos e tabelas) algumas vezes são necessárias.

Neste contexto, a principal diferença do SIG para um Sistema de Informação convencional é a sua capacidade de armazenar tanto os atributos descritivos como as geometrias dos diferentes tipos de dados geográficos, fato este que o torna essencial para uma melhor análise e entendimento de fatos e fenômenos que ocorrem no espaço geográfico.

2.3 Bancos de Dados Geográficos

De acordo com Ginzburg (2001), desde a sua origem a palavra "representação" esteve associada a uma forma abstrata de descrição do mundo. No entanto, os objetos e fenômenos reais são complexos demais para permitir uma representação completa, considerando os recursos à disposição dos sistemas de banco de dados atuais. Desta forma, se faz necessário a utilização de abstrações dos objetos e fenômenos do mundo real, de modo a obter uma forma de representação conveniente, embora simplificada, que seja adequada às finalidades das aplicações de banco de dados.

Em um SIG, o componente de armazenamento, denominado Banco de Dados Geográficos (BDG), estrutura e armazena os dados que representam os fenômenos acerca do mundo real em uma dada época (correspondendo a uma abstração ou representação da realidade), de forma a possibilitar a realização das operações de análise envolvendo dados espaciais. Cabe salientar que os SIGs fazem uso de BDG, no entanto este último pode ser utilizado de forma isolada.

Na maioria dos casos, os dados geográficos são organizados em forma de planos de informação (*layers*), ou seja, como uma série de camadas, cada uma contendo feições geográficas espacialmente relacionadas. Cada camada, que representa um tema ou uma classe de informação, é um conjunto de feições que estão posicionalmente relacionadas às outras camadas através de um sistema de coordenadas comum.

A organização por planos de informação é definida segundo os temas de interesse no estudo, tais como: trechos de rua, setores censitários, eixos viários, curvas de nível, localização de serviços, etc. Esta organização caracteriza a estratificação das informações em níveis ou camadas distintas, permitindo flexibilidade e eficiência no acesso.

Estas camadas de informação podem estar estruturadas de duas formas principais: estrutura vetorial ou matricial, que condicionam diferentes maneiras de tratar os dados e diferentes possibilidades de manipulá-los para a realização das análises.

2.4 Representação Computacional do Espaço

Quando se busca a representação da realidade geográfica em um sistema computacional, capturando a semântica do domínio que está sendo analisado, é necessário que várias características sejam avaliadas, dentre as quais destacam-se:

- se o espaço será modelado de forma absoluta ou relativa:
  - caso sejam necessários dados no espaço absoluto, pode-se escolher entre os modelos de campos e de objetos;

  - caso o espaço relativo seja necessário, pode-se utilizar o modelo de redes;

- se os dados serão representados de forma vetorial ou matricial.

2.4.1 Espaço Absoluto e Espaço Relativo

De grande importância para a modelagem espacial, a distinção entre espaço absoluto e espaço relativo decorre da possibilidade de se representar a localização dos objetos no espaço ou apenas o posicionamento relativo entre eles, como ilustrado na Figura 2.2. Nesta figura, à esquerda são representados distritos de uma cidade fictícia, identificados por suas fronteiras, o que caracteriza o modelo de espaço absoluto. Do lado direito, mostra-se um grafo com as conexões dos distritos, que formam uma rede (repetiu-se a imagem dos distritos para uma melhor legibilidade da figura). No modelo de redes, a localização exata de cada distrito não é armazenada, pois a rede só captura as relações de adjacência, representando um modelo de espaço relativo.

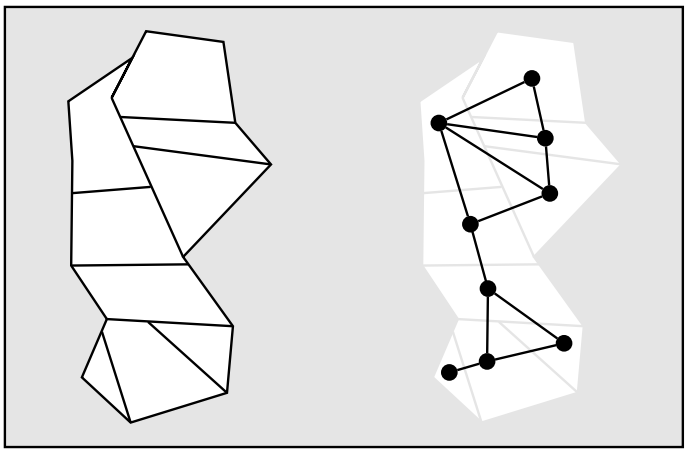

Figura 2.2: Comparação entre espaço absoluto e espaço relativo.

Segundo Couclelis (1997), o espaço absoluto, também chamado cartesiano, é um container de objetos e eventos, uma estrutura para localizar pontos, trajetórias e objetos. Espaço relativo, ou leibnitziano, é o espaço constituído pelas relações espaciais entre os objetos.

A escolha entre a forma absoluta ou relativa de representar o espaço depende principalmente dos tipos de análise desejados. Normalmente, consultas espaciais que envolvem dois tipos de entidades ("*quais estradas cruzam esta cidade?*") requerem a representação no espaço absoluto. O mesmo é válido para questões de álgebra de mapas ("*áreas inaptas para o plantio tem declividade maior que 10% ou solos arenosos.*"). Quando a análise envolve apenas as relações de conectividade ("*como chegar no banco, partindo do supermercado?*" ou "*qual é a média de idade dos meus vizinhos?*") pode-se utilizar representações no espaço relativo. Quando são modeladas entidades como estradas, linhas de transmissão, conexões de água e esgoto e linhas de comunicação, o espaço relativo é, na maioria das vezes, plenamente adequado.

Desta forma, é interessante ressaltar que se a localização exata é fundamental, ou se é necessário saber o valor do fenômeno em todos os pontos da região de estudo, então é necessário utilizar o modelo de espaço absoluto. Se o fluxo e as conexões são essenciais, então pode-se fazer uso do modelo de espaço relativo.

2.4.2 Modelo de Campo e Modelo de Objetos

Segundo Burrough e Frank (1995), os modelos de dados geográficos devem refletir a maneira como as pessoas enchergam o mundo. Um dos princípios filosóficos da percepção humana dos fenômenos geográficos é que a realidade é composta de entidades exatas e de superfícies contínuas.

Para Goodchild (1992), a realidade geográfica pode ser observada segundo duas visões: de campo e de objetos. Na visão de campo, a realidade é modelada por variáveis que possuem uma distribuição contínua no espaço e todas as posições no espaço geográfico podem ser caracterizadas através de um conjunto de atributos, como, por exemplo, temperatura, tipo de solo e relevo, medidos para um conjunto de coordenadas geográficas. Já na visão de objetos, a realidade consiste de entidades individuais, bem definidas e identificáveis, onde as fronteiras são partes essenciais destas entidades. Cada entidade tem suas propriedades e ocupa um determinado lugar no espaço. A realidade é modelada como um grande espaço onde entidades estão distribuídas sem que, necessariamente, todas as posições do espaço estejam ocupadas. Neste modelo, duas ou mais entidades podem estar situadas sobre uma mesma posição geográfica.

2.4.3 Modelo de Redes

Segundo Câmara (2005), o modelo de redes concebe o espaço geográfico como um conjunto de pontos no espaço (chamados de nós), conectados por linhas (chamadas de arcos). Os fenômenos modelados por redes incluem fluxo de pessoas ou materiais, conexões de influência, linhas de comunicação e acessibilidade, etc.

Como os nós de uma rede são abstrações de entidades existentes no espaço, eles podem estar associados a atributos descritivos. Por exemplo, na rede apresentada na Figura 2.2, do lado direito, cada nó está associado a um distrito de uma cidade fictícia e poderiam existir diferentes atributos para descrever as características deste distrito. Os arcos de uma rede também podem ter propriedades, como, por exemplo, o custo de percorrimento de um nó até outro.

2.4.4 Estruturas de Representação de Dados

As duas principais estruturas utilizadas para representar dados geográficos em um meio digital são: vetorial e matricial. Ambas surgiram como solução para a estruturação de dados geográficos, contudo, tanto uma quanto a outra possuem vantagens e desvantagens de acordo com a utilização a que se destinam. Desta forma, é importante ressaltar que nenhuma das estruturas é ideal em todas as ocasiões e que os critérios de escolha devem basear-se fundamentalmente nos objetivos de cada projeto.

2.4.4.1 Estrutura Vetorial

Na estrutura vetorial, a representação de um elemento ou objeto é uma tentativa de reproduzí-lo com a maior precisão possível. De forma geral, qualquer entidade ou elemento gráfico de um mapa é reduzido a três formas básicas: pontos, linhas, áreas (ou polígonos). As entidades geográficas são representadas pelas coordenadas de suas fronteiras.

Um ponto é um par ordenado (x, y) de coordenadas espaciais. Além das coordenadas, outros dados não-espaciais (atributos) podem ser arquivados para indicar de que tipo de ponto se está tratando. As linhas poligonais, arcos, ou elementos lineares são um conjunto de pontos conectados. Além das coordenadas dos pontos que compõem a linha, deve-se

armazenar informação que indique de que tipo de linha se está tratando, ou seja, a que atributo ela está associada. Um polígono é a região do plano limitada por uma ou mais linha poligonais conectadas de tal forma que o último ponto de uma linha seja idêntico ao primeiro da próxima. As formas básicas utilizadas são apresentadas na Figura 2.3.

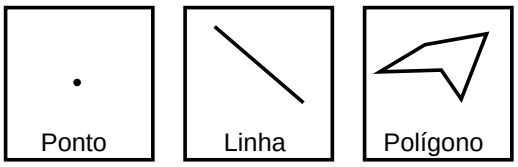

Figura 2.3: Formas básicas de representação utilizadas na estrutura vetorial.

Observa-se também que o polígono divide o plano em duas regiões: o interior, que convencionalmente inclui a fronteira, e o exterior. Ainda é possível utilizar combinações das formas básicas na criação da representação dos objetos. Por exemplo, pode-se ter objetos de área mais complexa, formados por um polígono básico e vários outros polígonos contidos no primeiro, delimitando buracos. Pode-se também ter objetos compostos por mais de um polígono, como seria necessário no caso da representação do estado de Santa Catarina, onde, além da parte continental, existe a ilha de Florianópolis e outras como parte de seu território.

Os polígonos podem ser utilizados de duas formas: como objetos isolados ou como objetos adjacentes. O caso de objetos isolados é simples de ser tratado, visto que não existem segmentos poligonais compartilhados entre os objetos. No entanto, existe o caso dos objetos adjacentes compartilhando suas fronteiras, exemplificados por todas as modalidades de divisão territorial: bairros, municípios, estados, etc. Neste caso, pode-se armazenar as coordenadas de cada objeto isoladamente, e assim duplicar as fronteiras em comum com outros objetos, ou armazenar cada fronteira comum uma única vez, indicando a que objetos estas fronteiras estão associadas. O primeiro caso é chamado de polígonos sem topologia e o segundo, de topologia arco-nó-polígono.

No modelo vetorial, a localização e a aparência gráfica de cada objeto são representadas por um ou mais pares de coordenadas. Este tipo de representação não é exclusivo dos SIGs: sistemas CAD (*Computer Aided Design*) e outros tipos de sistemas gráficos também utilizam a representação vetorial. Isto porque o modelo vetorial é bastante intuitivo para engenheiros e projetistas, embora estes nem sempre utilizem sistemas de coordenadas ajustados à superfície da Terra para realizar seus projetos, pois para estas aplicações um simples sistema de coordenadas cartesianas é suficiente. Mas o uso de vetores em SIGs é bem mais sofisticado do que em sistemas CAD, pois em geral os SIGs trabalham com volumes de dados maiores e contam com recursos para tratamento de topologia, associação de atributos alfanuméricos e indexação espacial.

Na Tabela 2.1 é apresentando um resumo das características da estrutura vetorial.

2.4.4.2 Dados 2,5D

Uma das possibilidades relacionadas aos dados vetoriais é a associação de valores que denotem a variação espacial de uma grandeza numérica. No caso mais simples, pode-se associar a cada localização no espaço um valor numérico de atributo. Neste caso, como os valores de localização estão no plano, o valor adicional descreve uma superfície sobre este

Tabela 2.1: Resumo das vantagens e desvantagens da estrutura vetorial. Adaptada de (CARVALHO; PINA; SANTOS, 2000; LISBOA FILHO; IOCHPE, 1996).

Vantagens	**Desvantagens**
Estrutura compacta.	Estrutura complexa.
Eficiência na análise de relacionamentos espaciais.	Operações de superposição de níveis de informação (*overlay*) mais complexas.
Feições são representadas precisamente, por pontos, linhas e polígonos.	Representação de áreas com alta variação espacial não muito eficiente.

plano. Os dados resultantes são chamados de dimensão "*dois e meio*", pois não se tratam estritamente de dados tridimensionais, já que o suporte espacial ainda é de localizações 2D.

A maneira mais comum de armazenar estes dados é através de estruturas matriciais, no entanto existem três alternativas que usam estruturas vetoriais:

- conjunto de amostras esparsas 2,5D, constituído de ternos ordenados (x, y, z), onde (x, y) é uma localização no plano e z um valor numérico de atributo;
- conjunto de isolinhas (curvas de nível), que são linhas às quais estão associados valores numéricos. As isolinhas não se cruzam, e são entendidas como estando "empilhadas" umas sobre as outras;
- malha triangular é uma estrutura com topologia do tipo nó-arco e representa uma superfície através de um conjunto de faces triangulares interligadas. A malha triangular é a estrutura vetorial mais utilizada para armazenar dados 2,5D. Cada um dos três vértices da face do triângulo armazenam as coordenadas de localização (x, y) e o atributo z, com o valor de elevação ou altitude. Quanto mais equiláteras forem as faces triangulares, maior a exatidão com que se descreve a superfície. O valor de elevação em qualquer ponto dentro da superfície pode ser estimado a partir das faces triangulares, utilizando-se interpoladores.

2.4.4.3 Estrutura Matricial

Na estrutura matricial (também conhecida como estrutura *raster*) o espaço é regularmente subdividido em células de uma matriz. Ou seja, esta estrutura utiliza uma grade regular sobre a qual se representa, célula a célula, os elementos. Para cada célula, atribui-se um código referente ao atributo analisado, de tal forma que o computador saiba a que elemento ou objeto pertence determinada célula.

Nesta estrutura, o espaço é representado como uma matriz $P(m, n)$ composta de m linhas e n colunas, onde cada célula é identificada por um número de linha, um número de coluna, um valor correspondente ao seu atributo e pode ser individualmente acessada por suas coordenadas. A representação matricial supõe que o espaço possa ser tratado como uma superfície plana, onde cada célula está associada a uma porção do terreno e o seu valor indica o tipo de objeto/entidade ou condição que é encontrada naquela localização.

A estrutura matricial pode ser utilizada para representar diferentes tipos de dados:

- **grade regular:** representação matricial de dimensão "*dois e meio*" na qual cada elemento da matriz está associado a um valor numérico, como apresentado na Figura 2.4 à esquerda;

- **matriz temática:** representação matricial 2D na qual cada valor da matriz é um código correspondente à uma classe do fenômeno estudado, como ilustrado na Figura 2.4 à direita.

15	31	41	35	12
38	43	28	17	21
48	21	19	33	51
25	06	33	59	20
09	29	40	21	58

01	02	03	02	01
02	03	02	01	02
03	02	01	02	03
02	01	02	03	02
01	02	03	02	03

Figura 2.4: À esquerda, grade regular com valores de altitude em metros. À direita, matriz temática com dados classificados (01 = “0-19 metros”, 02 = “20-39 metros”, 03 = “40-59 metros”).

Cabe salientar que na representação matricial, a área que cada célula representa define a resolução espacial, que é diretamente proporcional ao espaço de armazenamento e inversamente proporcional ao tamanho das células. A Figura 2.5 mostra uma mesma área utilizando representações matriciais de diferentes resoluções espaciais. Nesta figura, como o mapa do lado esquerdo possui uma resolução dez vezes maior que o do lado direito, as avaliações de áreas e distâncias serão bem mais exatas. Em contrapartida, o espaço de armazenamento necessário para o mapa da esquerda será dez vezes maior.

Figura 2.5: Representações matriciais de diferentes resoluções para uma mesma área geográfica.

Como os dados são codificados, célula a célula, atribuindo-se a cada uma o código correspondente à uma classe referente ao fenômeno estudado, é necessário estabelecer um critério a ser obedecido em toda a operação. Pode-se, por exemplo, atribuir a cada célula o código da classe sobre a qual estiver o centro da quadrícula. Outra possibilidade é adotar-se o critério da maior ocorrência. Neste caso, o código corresponde ao da classe que ocupar a maior parte da célula.

Uma das grandes vantagens da utilização da estrutura matricial é a facilidade de implementação das operações de superposição de níveis de informação. A superposição nada mais é do que uma operação matemática entre matrizes, combinando as células de mesma posição (mesma linha e coluna), nos diversos níveis de informação.

Na Tabela 2.2 é apresentando um resumo das características da estrutura matricial.

Tabela 2.2: Resumo das vantagens e desvantagens da estrutura matricial. Adaptada de (CARVALHO; PINA; SANTOS, 2000; LISBOA FILHO; IOCHPE, 1996).

Vantagens	Desvantagens
Simplicidade de implementação das operações de superposição.	Dificuldade de representação de relacionamentos topológicos.
Estrutura simples.	Dificuldade na associação de atributos a feições.
Representação eficiente de áreas com alta variação espacial.	Arquivos muito grandes.

2.4.4.4 Espaço Celular

Segundo Câmara (2005), um espaço celular é definido como uma estrutura matricial onde cada célula está associada a diversos atributos . O espaço celular tem várias vantagens sobre a estrutura matricial simples. Usando matrizes com um único atributo, um fenômeno espaço-temporal complexo precisa de várias matrizes separadas para ser representado. Num espaço celular, a mesma célula está associada a diferentes informações, com ganhos significativos no manuseio dos dados. Os espaços celulares ainda não são estruturas de dados comuns nos BDGs atuais.

2.4.4.5 Comparação entre as Estruturas Vetorial e Matricial

Comparando as duas estruturas de representação, é possível afirmar que a estrutura vetorial é a mais adequada quando existem operações onde é necessário um elevado grau de precisão ou o conhecimento acerca dos relacionamentos espaciais entre os objetos. Já as operações de álgebra de mapas são mais facilmente realizadas na estrutura matricial.

Geralmente, quando as duas estruturas apresentam um grau de precisão semelhante, o espaço de armazenamento requerido por uma representação matricial é substancialmente maior.

É importante salientar que, ao contrário da estrutura vetorial, as entidades representadas na estrutura matricial não correspondem às entidades espaciais do mundo real. Isto acontece pois as entidades espaciais na estrutura matricial, são as células individuais. Por exemplo, um rio nunca existe como uma entidade matricial distinta. Cada uma das células que representam o rio é que são as entidades. Um exemplo comparativo do uso das duas estruturas é apresentado na Figura 2.6. Na Tabela 2.3 são descritas, de forma resumida, as características de cada uma das duas entruturas.

2.5 Modelagem Dinâmica

Grande parte das aplicações em Geoprocessamento utiliza apenas representações estáticas dos fenômenos espaciais. Isto se deve ao fato de que a principal abstração utilizada nos SIGs é o mapa. Sendo assim, um dos principais desafios nesta área é transformar os SIGs, essencialmente estáticos, em ferramentas capazes de prover representações realísticas de fenômenos espaço-temporais dinâmicos, ou seja, que sejam capazes de representar adequadamente processos que variam tanto no espaço como no tempo (PEDROSA; CÂMARA, 2004).

Figura 2.6: Comparação entre as estruturas de representação de um espaço geográfico: à esquerda, a estrutura matricial de pior resolução; ao centro, a estrutura matricial de melhor resolução; à direita, mapa criado utilizando a estrutura vetorial.

Tabela 2.3: Resumo comparativo das características das estruturas vetorial e matricial.

Aspectos	Representação vetorial	Representação matricial
Relações espaciais entre objetos	Relacionamentos topológicos entre objetos disponível.	Relacionamentos espaciais devem ser inferidos.
Ligação com o BDG	Facilita a associação de atributos a elementos gráficos.	Associa atributos apenas a classes do mapa.
Escalas de trabalho	Adequado para qualquer escala.	Mais adequado para pequenas escalas.
Algoritmos	Problemas com erros geométricos.	Processamento mais rápido e eficiente.
Armazenamento	Por coordenada (mais eficiente).	Por matrizes.

A representação de processos inerentemente dinâmicos se faz extremamente necessária no estudo de diversas áreas, já que a forma estática, comumente utilizada, não os capturam adequadamente. Dentre as principais áreas de aplicação e utilização dos modelos espaço-temporais dinâmicos pode-se destacar: sistemas ecológicos (estudos climáticos, impacto ambiental, fluxo hidrológico e de poluição, etc) e sócio-econômicos (dinâmica populacional, cadastro urbano, uso e ocupação da terra, etc).

2.6 Considerações do Capítulo

A área de Geoprocessamento disponibiliza diversos modelos, estruturas, técnicas e ferramentas para a modelagem do ambiente geográfico em um computador. No contexto deste trabalho, estes diversos modelos foram analisados objetivando a seleção dos mais apropriados para serem utilizados no desenvolvimento do modelo proposto.

Como visto, atualmente ainda se faz necessário a criação de abstrações que permitam a representação dinâmica de fenômenos espaciais. No próximo capítulo, a área de Sistemas Multiagentes será apresentada como sendo uma alternativa para a representação de tais fenômenos.

3 SISTEMAS MULTIAGENTES

Inicialmente são apresentados alguns conceitos que irão ser utilizados ao decorrer deste capítulo. Após, é descrita a área de Inteligência Artificial Distribuída e as principais características de suas duas subáreas. Em seguida, é focada a apresentação dos conceitos de agentes e Sistemas Multiagentes, suas arquiteturas, modelos, ambientes e formas de comunicação.

3.1 Introdução

Em (DEMAZEAU, 1995) é proposta a decomposição de um sistema de acordo com uma metodologia de Inteligência Artificial Distribuída (IAD). Os conceitos fundamentais do sistema decomposto são relacionados abaixo (SICHMAN, 1995):

- **agente:** cada uma das entidades ativas do sistema;
- **sociedade:** um conjunto de agentes;
- **ambiente:** conjunto de entidades passivas do sistema;
- **interação:** trocas de informações que podem ocorrer entre os agentes;
- **organização:** restrições aplicadas aos agentes de uma sociedade.

Tais conceitos serão utilizados em todo este capítulo e poderão, se necessário, ser redefinidos de forma mais detalhada. Cabe ressaltar que nem sempre tais conceitos possuem significado compartilhado por todos os pesquisadores da área.

3.2 Inteligência Artificial Distribuída

Os sistemas computacionais baseados em Inteligência Artificial (IA) tornaram-se cada vez mais complexos e, com o passar do tempo, surgiu a necessidade de distribuir tais sistemas, objetivando aproveitar as facilidades oferecidas pelos sistemas distribuídos. Com a junção das técnicas de IA com as de sistemas distribuídos surgiu a IAD.

Além disto, ao contrário da IA clássica, onde a metáfora da inteligência é dada pelo comportamento humano individual (RUSSEL; NORVIG, 2003), dando ênfase aos aspectos de representação do conhecimento e métodos de inferência, a IAD é baseada no comportamento social, destacando as ações e iterações sociais.

Segundo Oliveira (1996), a IAD pode ser caracterizada como um paradigma no qual o comportamento inteligente é visto como o resultado das iterações de uma sociedade.

De modo geral, isto significa que se os comportamentos individuais forem organizados de forma correta, o conjunto deverá exibir uma inteligência maior que a soma das inteligências individuais.

A resolução de problemas pela IAD está ramificada em duas subáreas principais: Resolução Distribuída de Problemas (RDP) e Sistemas Multiagentes (SMAs), as quais são apresentadas a seguir.

3.2.1 Resolução Distribuída de Problemas

A RDP é motivada principalmente pela existência inicial de um problema bem determinado. Este problema é analisado e então são definidos os agentes necessários para resolvê-lo, bem como o modo como estes se organizam e iteragem. A abordagem utilizada na resolução deste problema é representada pela Figura 3.1 e baseia-se nas seguintes características (SICHMAN, 1995):

- o problema é resolvido por um conjunto de agentes criados especificamente para tal função, não sendo possível, *a priori*, sua reutilização em outros tipos de problema;
- a organização dos agentes é definida na fase de concepção do sistema, restringindo o comportamento de tais agentes de forma a focar na resolução do problema;
- a interação entre os agentes é realizada por troca de mensagens ou compartilhamento de dados. Os modelos de interação também são completamente definidos na fase de concepção do sistema, sendo completamente relacionados com o problema em questão;
- geralmente, a execução dos agentes ocorre de modo concorrente, buscando maior desempenho na resolução do problema;
- os agentes dividem entre si as diversas partes do problema original (tarefas, subproblemas) e as realizam de forma cooperativa;
- no sistema, o comportamento global coerente é garantido por um controle global, que pode ser centralizado ou distribuído.

3.2.2 Sistemas Multiagentes

Os SMAs possuem um enfoque diferente da RDP, priorizando o estudo de agentes autônomos em um ambiente multiagente. Nesta abordagem não existe um problema definido *a priori*. Deste modo, o objetivo da área é estudar modelos genéricos de agentes, organizações e iterações que possam ser aplicados a qualquer problema particular. Esta abordagem é representada pela Figura 3.2 e caracterizada por Sichman (1995):

- os agentes devem ser desenvolvidos independentemente de um problema em particular a ser solucionado. Desta forma é possível a reutilização dos mesmos em uma vasta gama de problemas;
- a concepção das organizações e iterações também devem ser realizada de forma genérica, independente de um problema particular;
- no sistema, não existe um controle global, já que este é implementado de forma descentralizada, nos agentes.

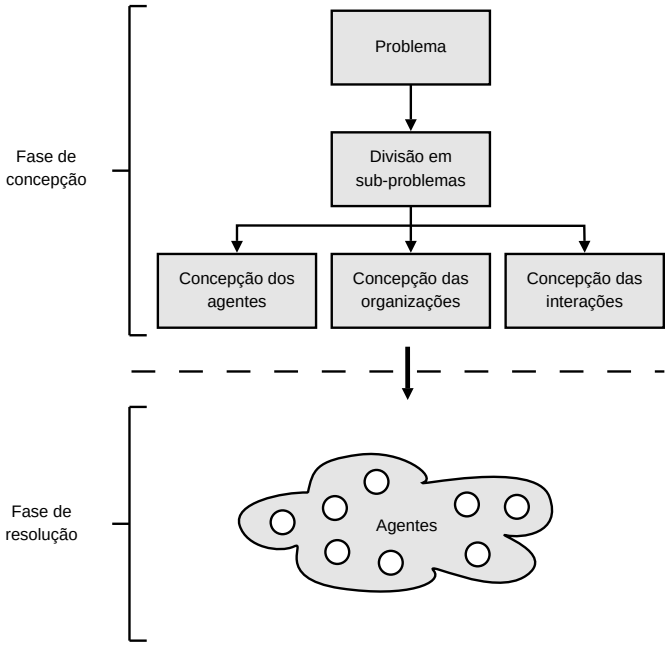

Figura 3.1: A abordagem baseada na RDP. Adaptada de (SICHMAN, 1995).

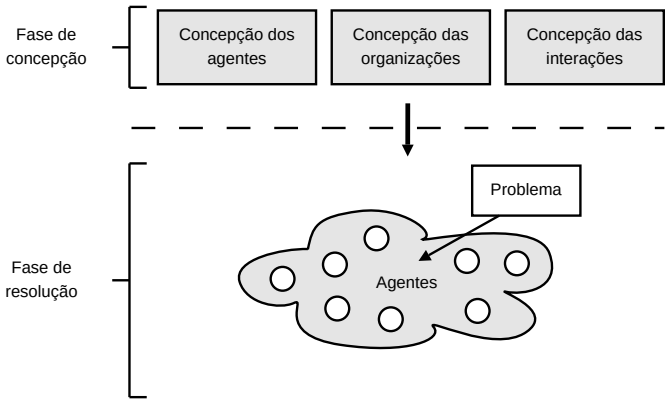

Figura 3.2: A abordagem baseada em SMAs. Adaptada de (SICHMAN, 1995).

3.3 Agentes

Não existe uma definição universalmente aceita para o termo agente. Este fato provém do seu uso nas mais diversas áreas, sendo que em cada uma delas podem existir diferentes aspectos relevantes. Este fato pode ser facilmente observado nas definições apresentadas abaixo. Russel e Norvig apresentam uma definição genérica de agente (ilustrada na Figura 3.3):

> *"Um agente é tudo o que pode ser considerado capaz de perceber seu ambiente por meio de sensores e de agir sobre esse ambiente por intermédio de atuadores. Por exemplo, nos agentes humanos, os olhos e ouvidos são sensores; as mãos e a boca são executores."* (RUSSEL; NORVIG, 2003)

Já Wooldridge e Jennings (1995) apresentam duas noções de agente: a fraca, que geralmente não gera controvérsias entre os pesquisadores, e a forte, a qual é potencialmente mais contestável. A noção fraca define um agente como sendo um sistema de hardware

ou software com as seguintes características:

- **autonomia:** os agentes operam sem qualquer intervenção direta e possuem alguma forma de controle sobre suas ações e estados internos;
- **capacidade social:** os agentes interagem com outros agentes por meio de algum tipo de linguagem de comunicação;
- **reatividade:** os agentes percebem o ambiente e reagem a estas percepções;
- **pró-atividade:** os agentes não simplesmente reagem ao ambiente, mas são capazes de tomar a iniciativa para realizar seus objetivos.

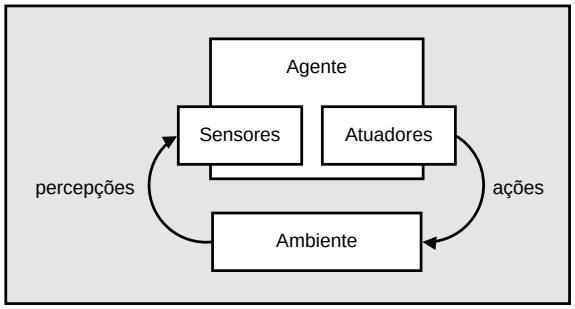

Figura 3.3: Relação entre agente e ambiente: o agente recebe informações do ambiente (percepções) através de seus sensores e produz ações que afetam o ambiente através de seus atuadores. Adaptada de (WOOLDRIDGE, 1999).

Além das características apresentadas acima, na noção forte os agentes são concebidos utilizando conceitos normalmente aplicados a seres humanos, podendo possuir noções mentais, como conhecimentos, crenças, desejos, intenções, obrigações e emoções. No entanto, em (WOOLDRIDGE; JENNINGS, 1995) diversas outras características são discutidas, tais como: mobilidade, veracidade, benevolência e racionalidade.

Uma outra definição interessante para agentes autônomos é apresentada por Pattie Maes:

> *"Agentes autônomos são sistemas computacionais que habitam um ambiente complexo e dinâmico, percebem e atuam autonomamente neste ambiente, e, fazendo isto, realizam um conjunto de objetivos ou tarefas para o qual foram projetados."* (MAES, 1995)

3.3.1 Arquiteturas dos Agentes

Tendo como base os trabalhos de Wooldridge (1999) e Russel e Norvig (2003), são apresentadas de, forma resumida, as principais arquiteturas de agentes:

- **arquiteturas baseadas em lógica:** nestas arquiteturas as informações são armazenadas sob forma de representação simbólica e as decisões são tomadas baseadas em deduções lógicas. Os mecanismos de inferência de decisão são robustos, resultando em agentes de grande complexidade;

- **arquiteturas reativas:** as decisões baseiam-se no mapeamento direto entre situação e ação. Tais arquiteturas foram concebidas como alternativa para o uso da IA simbólica, visto a intratabilidade de alguns problemas através desta última abordagem. Nestas arquiteturas não existe uma representação simbólica centralizada do modelo de mundo;

- **arquiteturas em camadas:** as tomadas de decisão são desenvolvidas em diversos níveis, onde cada camada analisa o ambiente em diferentes níveis de abstração. A organização em camadas pode ser horizontal ou vertical. Na organização horizontal, todas as camadas tem acesso a entrada sensorial e a saída de ações. Na organização vertical, apenas uma camada acessa a entrada sensorial, que é passada para as camadas superiores, até chegar na camada que controla a saída de ações;

- **arquitetura BDI (*Belief-Desire-Intention*):** tem suas origens no estudo de atitudes mentais. Nesta arquitetura as decisões tomadas pelos agentes são baseadas nos conceitos de crenças, desejos e intenções. As crenças seriam as informações que o agente tem sobre o ambiente. Os desejos seriam os mundos possíveis com os quais o agente pode escolher se comprometer a tornar realidade. As intenções representam estados que o agente pretende alcançar e que tem recursos comprometidos para este fim (JENNINGS; SYCARA; WOOLDRIDGE, 1998).

3.4 Sistemas Multiagentes

A área de SMAs é relativamente recente na Ciência da Computação. Embora suas pesquisas tenham iniciado em meados dos anos 80, só no início da década de 90 esta área ganhou destaque, sendo atualmente responsável por grande parte das investigações científicas realizadas no âmbito da IAD.

Nos SMAs dois ou mais agentes interagem ou trabalham em conjunto objetivando realizarem determinados objetivos. Cada agente é basicamente um elemento capaz de resolução autônoma de problemas, operando assincronamente com respeito aos outros agentes e são classificados de acordo com suas propriedades e arquitetura, podendo ser reativos, cognitivos ou híbridos. Os SMAs reativos normalmente são formados por inúmeros agentes, pois a inteligência surge das interações entre os agentes e o ambiente. Já os SMAs cognitivos são normalmente formados por uma quantidade menor de agentes, e a inteligência emerge do conhecimento, competência, intenções e crenças de cada agente, o que permite coordenar e negociar suas ações visando à resolução de um problema. Os sistemas híbridos misturam as características interessantes dos modelos reativos e cognitivos.

Segundo Demazeau (1995), os componentes básicos de um SMA são os agentes, as interações, o ambiente e as organizações. Os agentes poderiam ser desde simples autômatos até sistemas complexos baseados em conhecimento. O ambiente é extremamente dependente do domínio da aplicação, no entanto em alguns casos é um ambiente espacial. As interações podem ser físicas ou baseadas em atos de fala. As organizações podem ser baseadas em estudos biológicos ou sociais.

3.4.1 Sistemas Multiagentes Reativos

Os SMAs reativos baseiam-se na ideia de que não são necessários comportamentos complexos para a realização de tarefas complexas e que o comportamento inteligente

pode ser gerado sem representação explícita ou raciocínio abstrato, mas como uma propriedade emergente da interação de diversos agentes simples. Alvares e Sichman (1997) apresentam as principais características dos SMAs reativos:

- **não há representação explícita de conhecimento:** o conhecimento é implícito e se manifesta através do comportamento dos agentes;
- **não há representação do ambiente:** o comportamento dos agentes é baseado na percepção do ambiente, mas sem uma representação explícita deste. Desta forma, todas as informações relativas ao seus comportamentos estão no ambiente e suas reações dependem unicamente das percepções deste ambiente;
- **não há memória das ações:** os agentes reativos não mantém um histórico de suas ações, de forma que o resultado de uma ação passada não exerce nenhuma influência direta sobre suas ações futuras;
- **organização etológica:** similar a de sociedades de insetos, em oposição à organização social;
- **comunicação baseada em estruturas compartilhadas:** usualmente é utilizada a comunicação indireta, utilizando uma estrutura compartilhada para comunicação dos agentes;
- **grande número de agentes:** na ordem de dezenas, centenas ou mesmo milhares;
- **comportamento simples:** tipo estímulo-resposta, através de uma tabela com regras de condição-ação.

3.4.2 Sistemas Multiagentes Cognitivos

Segundo Alvares e Sichman (1997), os SMAs cognitivos são baseados em modelos organizacionais humanos e suas principais características são:

- **há representação explícita do ambiente:** os agentes cognitivos podem armazenar uma representação explícita de seu ambiente e dos outros agentes da sociedade;
- **há memória das ações:** os agentes cognitivos podem manter um histórico das interações e ações passadas, isto é, têm memória do passado;
- **comunicação direta:** a comunicação entre os agentes geralmente é feita de modo direto, através da troca de mensagens;
- **comportamento mais complexo:** os mecanismos de controle dos agentes são deliberativos, ou seja, tais agentes raciocinam e decidem sobre quais objetivos devem alcançar, que planos devem seguir e quais ações devem ser executadas num determinado momento;
- **organização sociológica:** seu modelo de organização é baseado em modelos sociológicos, como as organizações sociais;
- **pequeno número de agentes:** uma sociedade contém tipicamente poucos agentes, na ordem de dezenas.

3.4.3 Comunicação em Sistemas Multiagentes

Entre os aspectos mais relevantes em um SMA estão as iterações baseadas na comunicação. Neste escopo, torna-se necessário definir as formas de comunicação entre estes agentes. Diversos tópicos fazem parte deste contexto, dentre os quais pode-se destacar: modelos de comunicação, teoria dos Atos de Fala, ontologias, linguagens e protocolos de comunicação. Estes tópicos são apresentados extensivamente em (BORDINI; VIEIRA; MOREIRA, 2001; OKUYAMA, 2003; BASTOS, 2004). Desta forma, abaixo serão detalhados apenas os modelos de comunicação, que podem ser classificados como diretos ou indiretos.

3.4.3.1 Troca de Mensagens

De acordo com (HUHNS; STEPHENS, 1999), um subsistema de comunicação entre os agentes pode assumir uma de duas arquiteturas: direta ou assistida.

Na comunicação direta (apresentada na Figura 3.4) cada agente é responsável por sua comunicação, sem a intervenção de qualquer outro agente. Nesta abordagem existem dois principais problemas:

- **complexidade de implementação:** cada agente necessita armazenar informações sobre todos os outros agentes para poder realizar a comunicação;
- **custo e volume de comunicação:** todos agentes podem decidir se comunicar ao mesmo tempo, podendo causar o bloqueio total do sistema.

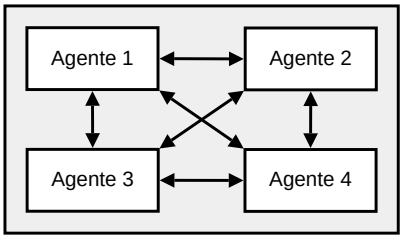

Figura 3.4: Modelo direto de comunicação de agentes. Adaptada de (BASTOS, 2004).

Na comunicação assistida (apresentada na Figura 3.5) os agentes apóiam-se em entidades especiais chamadas de "*agentes facilitadores*", os quais existem para auxiliar no processo de comunicação entre os outros agentes. Neste modelo o problema da comunicação é resolvido parcialmente, reduzindo de forma considerável a complexidade necessária aos agentes individuais para a realização da comunicação. Todavia, a existência dos "*agentes facilitadores*" pode resultar em uma certa centralização e gargalo (*bottleneck*) no sistema de comunicação. Se estes agentes pararem de funcionar, a comunicação fica comprometida.

3.4.3.2 Quadro Negro

O modelo de quadro negro (ENGELMORE; MORGAN, 1988) fornece uma estrutura de dados central, única e compartilhada, entre os vários agentes, onde as informações podem ser lidas e escritas durante o desenvolvimento das tarefas. Como não há comunicação propriamente dita entre os agentes, todas as interações ocorrem através do quadro negro e, por esta razão, os agentes devem consultar de tempos em tempos a estrutura para verificar

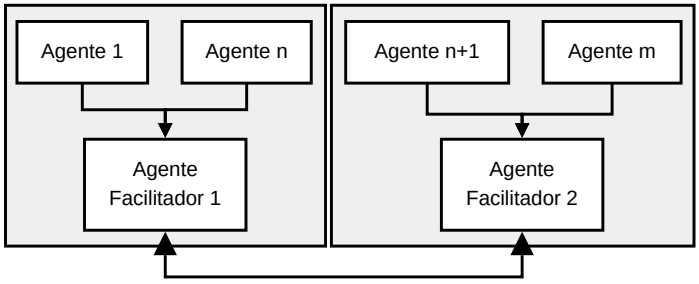

Figura 3.5: Modelo assistido de comunicação de agentes. Adaptada de (BASTOS, 2004).

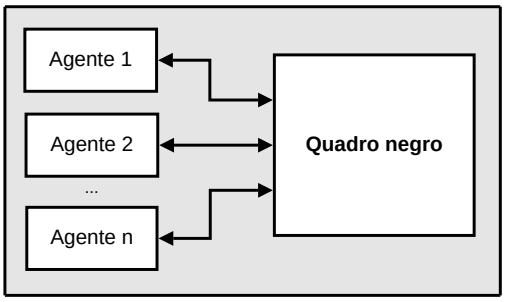

Figura 3.6: Modelo indireto de comunicação de agentes, baseado na estrutura de quadro negro. Adaptada de (BASTOS, 2004).

se existe alguma informação nova endereçada para eles. O esquema desta estrutura pode ser visualizado na Figura 3.6.

O quadro negro geralmente não impõe uma estrutura particular para as suas notificações. No entanto, é necessário um formato de mensagem bem compreendido para que os agentes possam interagir de forma correta. Alguns modelos de quadro negro fornecem serviços de filtragem para que os agentes não vejam todas as notificações, apenas aquelas de um tipo particular. Outros modelos enviam notificações automaticamente para os agentes que tenham registrado um interesse particular.

Apesar desta abordagem ser simples e eficiente, uma limitação visível é a falta de paralelismo no controle de escrita na estrutura de dados, pois apenas um único agente a cada momento pode utilizar o quadro negro. Outros problemas são a forma de sincronização e resolução de conflitos entre os agentes, devido à centralização da informação e do controle de acesso.

3.4.3.3 Comparação entre as Propostas

A troca de mensagens entre os agentes é geralmente o modelo mais utilizado, por garantir a privacidade das informações e a rapidez na transmissão das mensagens. No entanto, considerando a comunicação direta, é necessário garantir que os agentes recebam suas mensagens e prever a possibilidade de uma mensagem não chegar ao seu agente destinatário. Este problema pode ser solucionado utilizando-se um "*agente facilitador*", fazendo com que este tipo de agente armazene as mensagens até que consiga enviar com sucesso para o agente destinatário. Desta forma, o modelo de comunicação assistido resulta em uma menor eficiência na velocidade de transmissão das mensagens.

Já no modelo de quadro negro, é necessária a consideração de fatores como: distribuição de dados, sincronização dos agentes, privacidade das informações e velocidade de acesso a memória.

Os modelos apresentados se complementam e, em alguns casos, podem ser utilizados de forma integrada para o desenvolvimento de sistemas de comunicação mais complexos.

3.4.3.4 FIPA

A FIPA (*Foundation for Intelligent Physical Agents*) (FIPA, 2007) surgiu como um esforço na tentativa de padronizar as tecnologias de comunicação entre agentes, que até o seu surgimento eram completamente heterogêneas. Para tal, foi desenvolvido um conjunto de documentos, entre os quais está a descrição de uma linguagem de comunicação totalmente voltada para a comunicação entre agentes: a FIPA-ACL (SILVEIRA; GOMES; VICARI, 2005).

3.4.4 Ambientes Multiagentes

Conforme descrito em (OKUYAMA, 2003), na especificação de um SMA, a modelagem e representação do ambiente em que os agentes estão situados possui uma enorme importância, principalmente para simulações sociais onde é necessário um ambiente virtual no qual os agentes irão interagir. No entanto, neste contexto, a modelagem do ambiente ainda é um aspecto pouco explorado. Desta forma, quando o SMA é um sistema computacional completo (não situado em um ambiente real), a modelagem do ambiente compartilhado entre os agentes é fundamental, pois permite a análise de certos mecanismos de interação de maneira detalhada, muitas vezes com mais detalhes que em aplicações situadas em um ambiente real (FERBER, 1999). Considerando um agente situado em um ambiente, quaisquer outros agentes serão, do ponto de vista deste agente, parte importante do ambiente (OKUYAMA, 2003).

Nos sistemas reativos os agentes não possuem memória e, neste caso, o ambiente tem um papel fundamental, pois é apenas a percepção deste que permite aos agentes tomarem suas decisões. Já em sistemas cognitivos, os agentes possuem uma representação interna do ambiente no qual estão situados. Neste caso, estes agentes tomam suas decisões baseados em mudanças que a percepção do ambiente causa em sua representação interna. Desta forma, segundo (OKUYAMA, 2003), a modelagem do ambiente é importante tanto para os agentes reativos quanto para os cognitivos.

Mesmo com a vasta variedade de ambientes que podem existir no contexto dos SMAs, estes podem ser classificados de acordo com um pequeno número de dimensões, conforme apresentado a seguir (RUSSEL; NORVIG, 2003):

- **completamente observável ou parcialmente observável:** se os sensores dos agentes fornecem o estado completo do ambiente em cada instante, então o ambiente é considerado completamente observável. Um ambiente é considerado de fato completamente observável se os sensores detectam todos os aspectos relevantes para a escolha das ações dos agentes. Ambientes completamente observáveis são convenientes pois os agentes não precisam manter qualquer estado interno para controlar o mundo. Um ambiente pode ser parcialmente observável devido ao ruído e a sensores imprecisos ou porque partes do estado estão ausentes nos dados do sensor;

- **determinístico ou não-determinístico (estocástico):** se o próximo estado do ambiente é totalmente determinado pelo estado atual e pelas atuações dos agentes, então o ambiente é determinístico. A princípio, os agentes não precisam se preocupar com a incerteza em um ambiente completamente observável e determinístico. Porém se o ambiente é parcialmente observável, então este pode parecer não-determinístico, do ponto de vista dos agentes;

- **episódico ou não-episódico (seqüencial):** em um ambiente episódico, as experiências dos agentes são divididas em episódios atômicos. Em cada episódio, o comportamento dos agentes consiste apenas em perceber e agir, pois estas ações não irão interferir nos próximos episódios. Já em um ambiente seqüencial, a ação atual pode afetar as ações futuras;

- **estático ou dinâmico:** caso possam ocorrer mudanças no ambiente enquanto um agente está deliberando, o ambiente é dito dinâmico para este agente; caso contrário, ele é estático;

- **discreto ou contínuo:** a diferença entre discreto e contínuo pode ser aplicada ao estado do ambiente, ao modo como o tempo é tratado, e ainda às percepções e ações dos agentes.

3.5 Considerações do Capítulo

Neste capítulo foram apresentadas as características dos SMAs que serão tratadas no modelo proposto. Também foi ressaltada a necessidade de uma melhor modelagem do ambiente dos agentes, fato este que sugere a necessidade da utilização de técnicas, modelos, estruturas e ferramentas mais adequadas para tal objetivo.

Cabe salientar ainda que a simulação de situações reais é bastante compatível com a modelagem baseada em agentes. Sendo assim, o próximo capítulo apresenta, de forma mais detalhada, a área de simulações, focando na apresentação das simulações baseadas em agentes.

4 SIMULAÇÕES

Este capítulo aborda a área das simulações, revisando definições, principais conceitos, objetivos, etapas e áreas de aplicação. Também são detalhados os modelos de simulação baseados em agentes, as simulações sociais e as simulações no contexto de Geoprocessamento.

4.1 Introdução

De modo geral, uma simulação pode ser analisada como um tipo particular de modelagem. Desenvolver um modelo é uma conhecida forma de entender aspectos do mundo real. Um modelo é uma simplificação – menor, menos detalhado, menos complexo – de uma estrutura ou sistema (GILBERT; TROITZSCH, 1999).

As diversas definições de simulação são muito semelhantes e tendem a apresentar sempre um mesmo objetivo: realizar uma representação (criar um modelo) do mundo real, de forma artificial. Abaixo, são apresentadas algumas definições de simulação, objetivando reforçar a afirmação acima:

- "*O propósito dos modelos de simulação é permitir a realização de estudos sobre os sistemas reais, analisando suas reações ante a influências internas e externas, ou sua abrangência no meio ambiente.*" (STRACK, 1984);
- "*Simulação é a tentativa científica que consiste em realizar uma reprodução artificial, chamada de modelo, de um fenômeno real que se deseja estudar e observar o comportamento.*" (DROGOUL, 1993);
- "*Simular significa reproduzir o funcionamento de um sistema, com o auxílio de um modelo, o que permite testar hipóteses sobre o valor de alguns componentes controlados.*" (SILVA et al., 1998).

Atualmente, métodos de simulação têm sido utilizados com bastante sucesso como ferramentas auxiliares no processo de tomada de decisão, principalmente em planejamentos a médio e longo prazo e em situações que envolvem custo e risco elevados. A utilização das simulações permite a análise de detalhes específicos com grande precisão. Desde modo, segundo Rebonatto (1999), utilizar modelos computacionais de simulação é uma forma poderosa de projetar, planejar, controlar e avaliar novas alternativas ou mudanças de estratégias em sistemas no mundo real. Sintetizando as definições apresentadas acima, pode-se afirmar que a utilização de modelos de simulação visa (ADAMATTI, 2003):

- descrever o comportamento de um sistema real;

- elaborar teorias e hipóteses que sigam as regras deste comportamento;
- utilizar o modelo desenvolvido para testar as hipóteses criadas a fim de prever comportamentos futuros.

4.2 Etapas do Processo de Simulação

Segundo Strack (1984), o processo de simulação pode ser dividido em três etapas principais:

- **etapa de modelagem:** criar um modelo teórico para representar os fenômenos do sistema real que está sendo analisado;
- **etapa de experimento:** simular o modelo teórico desenvolvido e aplicar variações sobre o mesmo, alterando parâmetros que possam influenciar no processo de resolução;
- **etapa de validação:** comparar os dados experimentais obtidos utilizando o modelo teórico (depois da realização dos ajustes necessários) com os dados do sistema real em questão (análise dos resultados).

A Figura 4.1 representa as relações entre o sistema real, o modelo teórico e o simulador. Também podem ser visualizadas as três etapas do processo de simulação, apresentadas acima. Neste processo, a partir do sistema real é realizada a modelagem do problema desejado e criado um modelo teórico. Fazendo uso do modelo criado e de um simulador são realizadas as simulações. Se necessário, de forma intercalada com as simulações, são realizados ajustes no modelo, até este ser considerado satisfatório. Na avaliação final, é realizada a comparação entre os resultados do modelo teórico e as observações do sistema real.

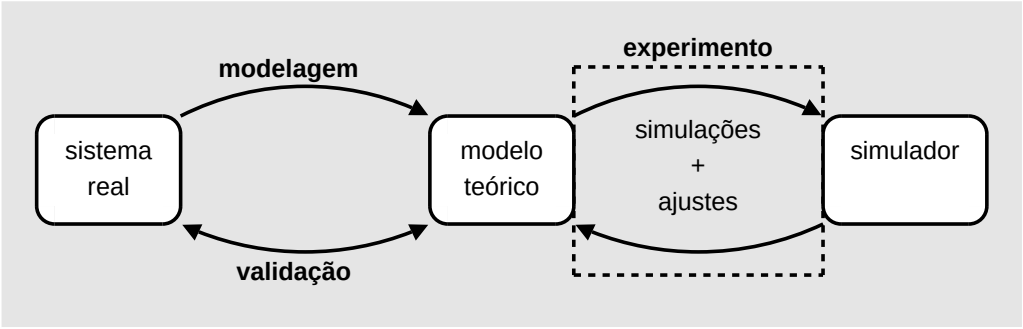

Figura 4.1: Etapas do processo de simulação. Adaptada de (GONÇALVES, 2003; FROZZA, 1997).

4.3 Modelos de Simulação

Segundo Parunak, Savit e Riolo (1998), os modelos de simulação podem ser classificados em:

- **Modelos Baseados em Agentes (MBA):** são formados por um conjunto de agentes que encapsulam os comportamentos dos diversos indivíduos que compõem o

sistema real. A execução destes modelos consiste na emulação destes comportamentos;

- **Modelos Baseados em Equações (MBE):** são formados por um conjunto de equações que descrevem o sistema real. A resolução destes modelos consiste na avaliação dos resultados destas equações.

Geralmente, o desenvolvimento dos MBA é mais simples do que o dos modelos matemáticos puramente analíticos (MBE), pois nem sempre é trivial representar analiticamente o comportamento de indivíduos reais. Nos MBA cria-se o comportamento dos agentes e então os resultados das simulações são gerados pela interação entre estes agentes.

Segundo Oliveira (2003), em inúmeros contextos os MBA geram resultados mais próximos da realidade do que os MBE, já que estes últimos utilizam médias de variáveis críticas para a análise dos resultados e assumem um alto nível de homogeneidade no comportamento dos indivíduos, o que geralmente não ocorre em situações reais.

Nas simulações de sistemas com dinâmica não-linear, as variações locais das médias podem afetar de forma significativa o comportamento geral do sistema. Sendo assim, em domínios mais complexos os MBA geralmente vêm sendo utilizados com resultados mais expressivos que os MBE.

4.4 Modelos Baseados em Agentes

Segundo Frozza (1999), o processo de modelagem de um sistema sob a perspectiva de SMAs pode ser organizado nas seguintes etapas:

1. decompor o fenômeno real em um conjunto de elementos autônomos;
2. modelar cada um dos elementos como um agente, definindo seus conhecimentos, funções, comportamentos e modos de interação;
3. definir o ambiente destes agentes;
4. definir quais agentes possuem as capacidades de ação e comunicação.

Como apresentado anteriormente, as simulações de situações reais são bastante compatíveis com os MBA, pois nas simulações baseadas em agentes o fenômeno real é decomposto em um conjunto de elementos e suas interações. Estes modelos estudam as conseqüências globais das interações locais dos membros da população.

4.4.1 Simulações Sociais

O estudo das sociedades naturais é o objetivo original das simulações sociais. Na prática, o foco principal é o estudo experimental e a modelagem de conjuntos de efeitos imprevisíveis relacionados com uma população de agentes atuando em um ambiente compartilhado, o que representa o estudo da complexidade, emergência, auto-organização e dinâmica originadas pelo comportamento de tais agentes (CASTELFRANCHI, 1998).

Fazendo uso das simulações é possível analisar fenômenos sociais que não são diretamente acessíveis, ou por não existirem mais ou por possuírem uma complexidade muito grande a ponto de não poderem ser facilmente previstos (GILBERT; CONTE, 1995). Em algumas ocasiões, as pesquisas científicas focam fenômenos em tais condições. A observação do comportamento da sociedade humana na pré-história ou a evolução demográfica

de uma cidade são exemplos que podem exigir um diferente método de análise, que vai além de métodos puramente matemáticos e de análises teóricas. Este tipo de análise se adapta perfeitamente ao objetivo principal das simulações sociais (DREHMER, 2003).

Desta forma, a principal razão para o crescente interesse dos cientistas sociais nas simulações computacionais é o seu grande potencial para descobertas e formalizações (criações de teorias), pois as simulações permitem que os cientistas criem modelos simples com foco em algum aspecto específico do mundo social e visualizem as conseqüências de suas teorias nas "sociedades artificiais" desenvolvidas (GILBERT; TROITZSCH, 1999).

Sendo assim, o uso de SMAs como ferramenta para as criação de simulações sociais potencializa os estudos científicos desta área, já que as metáfora de agentes e de sociedades de agentes podem ser utilizadas diretamente na modelagem de sociedades naturais. Com isto, o estudo e a modelagem das sociedades naturais se afasta de questões como: "*o que aconteceu?*" e se volta para questões como: "*quais condições são necessárias para que determinado fenômeno possa ocorrer?*". Geralmente, as respostas para esta última pergunta podem refletir diretamente na modelagem dos agentes e do sistema em questão.

4.5 Simulações na Área de Geoprocessamento

Os modelos de simulação desenvolvidos entre o final dos anos 50 e a metade dos anos 80, de uma maneira geral, não trabalhavam focando a representação da dimensão espacial. De fato, os avanços na representação espacial ocorreram apenas no final dos anos 80, quando os Autômatos Celulares (ACs)[2] começaram a ser utilizados em larga escala. A utilização de modelos espaciais dinâmicos no processo de simulação deu origem ao que hoje é conhecido como *Geosimulation*.

4.5.1 *Geosimulation*

Fundamentalmente, a *Geosimulation* pode se diferenciar de outros métodos de simulação por possuir uma atenção explícita ao espaço e a geografia, na simulação de fenômenos dinâmicos. Este fato pode ser observado na descrição dos objetos, na especificação dos comportamentos e da dinâmica da simulação. A *Geosimulation* está centrada em metodologias baseadas em autômatos para simular sistemas espaciais discretos e dinâmicos, utilizando ACs e SMAs em um contexto espacial (BENENSON; TORRENS, 2004,).

Como campo de estudo, a *Geosimulation* é influenciada por diversas áreas, tais como: teoria dos Sistemas Complexos, Geoprocessamento e Programação Orientada a Objetos (POO), principalmente.

4.6 Considerações do Capítulo

No contexto deste trabalho, objetiva-se fazer uso de modelos de simulação baseados em agentes em um ambiente onde as características geográficas recebam especial atenção. Em outras palavras, objetiva-se fazer uso das opções existente para a criação de ambiente espaciais com um alto nível de detalhamento e inserir agentes modificadores neste contexto, criando assim simulações mais próximas da realidade.

No entanto, em diversas situações se faz necessário um sistema computacional que facilite a criação, execução e visualização de tais simulações. Algumas das principais

[2]Detalhes sobre ACs e suas aplicações podem ser encontrados em (WOLFRAM, 1983; BARONE, 2003; AGUIAR, 2004; GRIGOLETTI, 2005).

ferramentas existentes atualmente para estes propósitos são apresentadas no próximo capítulo, visando conhecer suas principais características.

5 TRABALHOS CORRELATOS

Neste capítulo são apresentadas as principais características de algumas plataformas para a criação e execução de simulações baseadas em agentes. Também são abordados alguns modelos específicos de simulação que integram os conceitos de SMAs e SIG. Por fim, é realizada uma comparação, apresentando as vantagens e desvantagens das plataformas e dos modelos apresentados.

5.1 Introdução

No contexto das simulações baseadas em agentes, atualmente existe uma infinidade de plataformas. De forma semelhante aos objetivos do trabalho de Castle e Crooks (2006), esta seção apresenta uma análise das características de 5 das principais plataformas existentes: Swarm, Repast, SeSAm, NetLogo e OBEUS. São focadas algumas características como: arquitetura geral da plataforma, entidades existentes, formas de modelar o ambiente e integração com os SIGs.

5.2 Swarm

A plataforma Swarm (SWARM DEVELOPMENT GROUP, 2006; MINAR et al., 1996), desenvolvida na Universidade de Santa Fé, Estados Unidos, é uma ferramenta computacional para a realização de simulações de sistemas complexos adaptativos, baseada nos conceitos de SMAs. O Swarm engloba um conjunto de bibliotecas orientadas a objetos que permite a criação de simulações nas linguagens Java e Objective-C.

5.2.1 Entidades

Nos sistemas desenvolvidos em Swarm, a unidade básica de uma simulação é o agente. Um agente é qualquer entidade ativa capacitada a produzir eventos que alterem outros agentes do sistema. Os agentes possuem estados internos, percepções (meio de modificar os seus estados internos) e comportamento (meio de modificar o ambiente). O principal componente das simulações é chamado de *swarm* e consiste em uma coleção de agentes com um escalonador de eventos aplicado sobre estes agentes.

Além de ser uma coleção de agentes, um *swarm* pode representar um único agente. Neste caso, o comportamento deste agente é definido como a resultante dos fenômenos emergentes do comportamento dos agentes que fazem parte do *swarm*. Desta forma é possível representar modelos de forma hierárquica. Nestes modelos, não existe uma entidade central, e sim coleções de indivíduos. A idéia é que cada agente possa buscar, dentro de seus conhecimentos, ou por meio da comunicação com outros agentes, a resposta aos

seus problemas.

Nos modelos desenvolvidos em Swarm, normalmente existem dois tipos de agentes, os primários e os auxiliares. Os agentes primários são aqueles definidos pelos pesquisadores dentro de seus modelos. Os agentes auxiliares têm como objetivo facilitar o desenvolvimento do modelo e também "observar" as simulações, a fim de coletar dados para a criação de estatísticas.

5.2.2 Ambiente

Na plataforma Swarm, não existe uma forma especial para definição do ambiente. Mesmo que, dentro de uma simulação, o ambiente tenha um papel diferente dos agentes normais, é tratado como um simples agente. No entanto, o Swarm disponibiliza algumas bibliotecas e extensões para a criação de ambientes discretos em duas e três dimensões.

5.2.3 Utilização de Dados Matriciais

O Kenge (BOX, 1999), uma extensão para a plataforma Swarm, possibilita a criação de superfícies bidimensionais discretas com funcionalidades semelhantes as dos ACs. Estas superfícies são criadas a partir de dados matriciais e utilizadas para criar o ambiente para os agentes.

5.3 Repast

O Repast (*Recursive Porous Agent Simulation Toolkit*) (NORTH; COLLIER; VOS, 2006; ROAD, 2006) é uma plataforma livre para a criação de simulações baseadas em agentes. Concebido inicialmente como uma biblioteca de classes Java utilizada para simplificar o uso da plataforma Swarm[3], atualmente é uma plataforma independente, possuindo um conjunto abstrato de características e três diferentes implementações: Repast Java, Repast .NET e Repast Python.

5.3.1 Arquitetura

A arquitetura da plataforma Repast é dividida em módulos, os quais podem ser classificados em fixos ou flexíveis. Os módulos fixos devem necessariamente existir em qualquer implementação da plataforma Repast. Já os módulos flexíveis são opcionais.

Esta arquitetura é organizada em seis módulos, como apresentado na Figura 5.1: núcleo do Repast, módulo de log, módulo de execução interativa, módulo de execução em lote, módulo de comportamento adaptativo e módulos específicos.

O *núcleo do Repast* (módulo fixo) é responsável pelo controle das atividades na simulação. Este módulo contém quatro componentes: controlador, escalonador, ações e agentes. O controlador trata da execução das simulações (iniciar, parar, reiniciar, pausar). O escalonador, normalmente implementado como um gerenciador de eventos discretos no tempo (LAW; KELTON, 2000), faz o controle do tempo. O componente ações gerencia os eventos que ocorrem durante as simulações. O componente agentes implementa os agentes das simulações.

O *módulo de log* (módulo fixo) é responsável por armazenar os resultados das simulações. Este módulo é formado por 2 componentes: um para armazenar valores primitivos (log de dados) e outro para armazenar o estado dos objetos (agentes).

[3]Esta concepção inicial foi abandonada e tornou-se redundante com o lançamento do *Java Swarm*, uma camada Java que é executada em conjunto com o simulador Swarm.

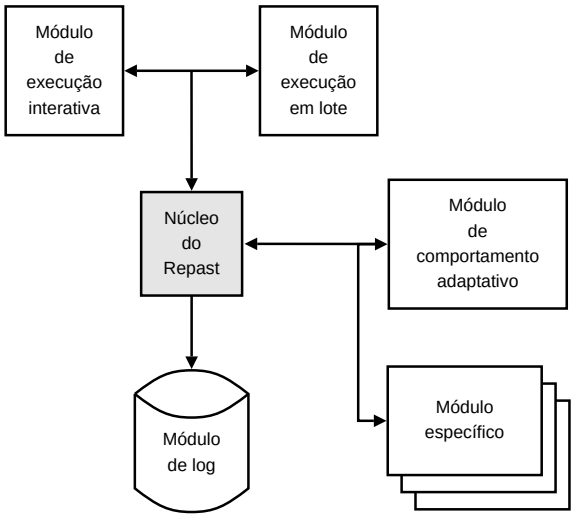

Figura 5.1: Arquitetura abstrata da plataforma Repast. Adaptada de (NORTH; COLLIER; VOS, 2006).

O *módulo de execução interativa* (módulo fixo) é responsável por gerenciar as simulações que possuem intervenções por parte do usuário (controladas pelo usuário). Este módulo geralmente atua como intermediário entre o núcleo do Repast e o usuário. Já o *módulo de execução em lote* (módulo fixo) é responsável por gerenciar simulações realizadas em lote, ou seja, sem a intervenção do usuário.

O *módulo de comportamento adaptativo* (módulo flexível) agrupa diversos componentes que fornecem comportamento aos agentes: redes neurais, algoritmos genéticos, etc. Os *módulos específicos* (módulos flexíveis) agrupam funcionalidades de diversas áreas. Em alguns casos, as funcionalidades são tão complexas que se tornam módulos propriamente ditos. Exemplos destes componentes incluem funcionalidades para as áreas de sistemas sociais, sistemas dinâmicos, teoria dos jogos e SIGs.

5.3.2 Modelos de Representação do Ambiente

O Repast fornece diversos modelos para a representação do ambiente dos agentes. Estes modelos podem ser classificados em dois tipos principais:

- **modelo de rede (*network model*):** é formado por duas estruturas básicas, nodos e arestas;
- **modelo de grade (*grid model*):** é uma grade bidimensional discreta particionada em células.

O modelo de grade pode variar de acordo com algumas características (ROAD, 2006):

- as células podem conter um ou mais objetos (agentes);
- o formato das células pode ser quadrangular ou hexagonal;
- os objetos (agentes) em uma célula podem estar ordenados (em forma de fila) ou não;
- o ambiente pode ser toroidal ou não;

- em células quadradas, pode-se definir a vizinhança de Von Neumann ou de Moore;
- em células hexagonais, a vizinhança é em forma de anel.

5.3.3 Utilização de Dados Vetoriais

Segundo Najlis e North (2005), atualmente a plataforma Repast suporta operações de leitura, escrita e apresentação de dados no formato Shapefile (ESRI, 1998). Na integração entre o Repast e o SIG, as tarefas são divididas em duas diferentes classes: uma de dados e outra de apresentação. A classe de dados permite que o Repast realize operações de leitura e escrita de dados no SIG. A classe de apresentação trabalha em conjunto com o SIG para coordenar a apresentação dos dados da simulação. Apesar da apresentação não ser realmente realizada pela classe de apresentação, esta classe realiza algumas operações para que tal processo possa ocorrer.

Para que os agentes possam ser apresentados em um mapa é necessário realizar a criação dos mesmos, utilizando os dados no formato Shapefile, como apresentado em (NAJLIS; NORTH, 2005). No entanto, para que esta integração seja realmente útil, se faz necessário a implementação de métodos nos agentes que correspondam ao campo dos dados Shapefile. Por exemplo, se nos dados existir um campo denominado `uso_de_solo`, o agente necessitará de dois métodos: um para ler o campo `uso_de_solo` do arquivo de dados e definir tal informação como atributo do agente e outro para atualizar tal informação no arquivo de dados, a partir do atributo do agente.

Após a definição da classe do agente, é necessário criar (instanciar) os agentes. Então, para cada registro do arquivo de dados será criado um agente. A atual implementação da plataforma foca na utilização de dois sistemas para a apresentação dos dados, antes, durante e após as simulações: *ESRI ArcMap* (ESRI, 2006) e *OpenMap* (BBNT, 2006).

5.4 SeSAm

O ambiente SeSAm (*Shell for Simulated Agent Systems*) (KLÜGL; HERRLER; FEHLER, 2006; KLÜGL et al., 2006), desenvolvido no Departamento de Inteligência Artificial e Ciência da Computação Aplicada da Universidade de Würzburg, Alemanha, fornece uma plataforma genérica para o desenvolvimento e simulação de modelos baseados em agentes. O foco principal deste ambiente é propiciar uma linguagem visual para a construção de modelos complexos, independentes de domínio.

Com uma interface gráfica para a implementação dos modelos, funcionalidades para a criação de animações, geração e análise estatística dos dados das simulações, o SeSAm possibilita que cientistas sem nenhum conhecimento de programação possam desenvolver seus modelos.

Em sua versão inicial, o SeSAm foi desenvolvido na linguagem Lisp, podendo ser executado apenas em sistemas Macintosh. Este fato tornou-o pouco portável. Desta forma, o sistema passou a ser implementado na linguagem Java, garantindo maior portabilidade.

As principais entidades em um modelo SeSAm são os agentes, os recursos e o mundo.

5.4.1 Agentes e Recursos

Os agentes representam as entidades ativas do modelo e são constituídos de um corpo, um conjunto de variáveis de estado e um comportamento, especificado de forma semelhante a um diagrama de atividades UML (*Unified Modeling Language*).

Os recursos representam as entidades passivas do modelo. De forma similar aos agentes, possuem corpo e um conjunto de variáveis de estado, no entanto não possuem comportamento.

5.4.2 Mundo

No SeSAm, o ambiente dos agentes é chamado de "mundo". Este mundo também possui variáveis e comportamento. Sendo assim, do ponto de vista de implementação, o mundo nada mais é do que um tipo especial de agente. O mundo se diferencia dos agentes por possuir uma posição fixa na seqüência da atualização (sempre ou antes ou após todos os outros agentes), enquanto que os agentes normais são atualizados sempre aleatoriamente. Sendo assim, se necessário, o mundo pode servir como forma de coordenação entre os agentes. Durante as simulações, o mundo também pode monitorar o comportamento dos agentes, criar e remover agentes e recursos.

A representação do mundo em forma de grade é apenas uma abstração utilizada para fins de visualização, já que tal representação não possui significado real na realização das simulações. Os valores de linhas e colunas são definidos em função das células, no entanto, cada célula é particionada em 100×100 pontos, nos quais os agentes podem ser posicionados. Sendo assim, o mundo realmente é discreto, porém, de forma muito mais detalhada do que aparenta ser. Um mundo de 20×20 células contém, na verdade, 2000×2000 possíveis posições.

O mundo pode ser definido como toroidal ou não, o que altera automaticamente as formas de percepção e movimentação dos agentes.

5.4.3 Utilização de Dados Vetoriais

Recentemente foi desenvolvido um *plugin* objetivando possibilitar ao SeSAm acessar dados geográficos, mais especificamente dados no formato Shapefile. Basicamente, este *plugin* importa e converte dados geográficos para variáveis dos agentes, possibilitando aos agentes possuírem atributos geográficos. Também são disponibilizadas algumas operações quem levam em consideração a feição geométrica dos agentes.

5.4.4 Arquitetura

A arquitetura simplificada da plataforma SeSAm é apresentada na Figura 5.2. Nesta arquitetura, a modelagem visual é o principal componente. Com este recurso, é possível representar as entidades (agentes, recursos e o mundo) e seus comportamentos, sem a necessidade de conhecimento em uma linguagem de programação. As simulações podem resultar em animações e dados estatísticos. As animações apresentam o posicionamento dos agentes no ambiente. As rotinas de análise estatística possibilitam gerar dados estatísticos das simulações.

5.5 NetLogo

O NetLogo (WILENSKY, 1999) é uma plataforma para simulação de fenômenos naturais e sociais, desenvolvido na linguagem Java. Foi criado por Uri Wilensky em 1999 e está em contínuo desenvolvimento no "*Center for Connected Learning and Computer-Based Modeling*".

Basicamente, o NetLogo fornece uma ferramenta multiplataforma para a criação de simulações baseadas em agentes, uma linguagem de programação própria e uma vasta bi-

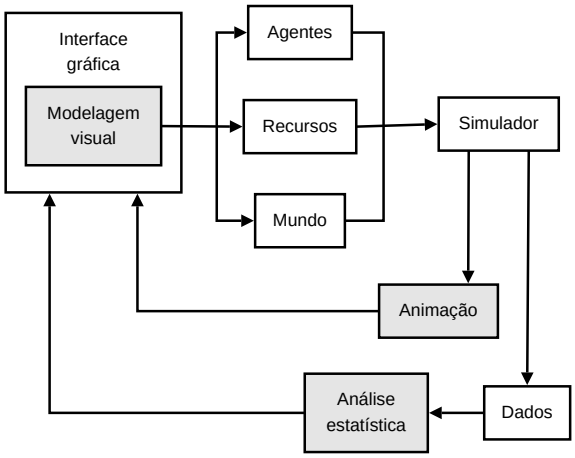

Figura 5.2: Arquitetura simplificada da plataforma SeSAm. Adaptada de (KLÜGL; HERRLER; OECHSLEIN, 2003; KLÜGL; PUPPE, 1998; ADAMATTI, 2001; OLIVEIRA, 2003).

blioteca de classes. Nesta ferramenta, os resultados das simulações podem ser observados fazendo uso da representação espacial ou do sistema de plotagem existente.

5.5.1 Entidades

As simulações realizadas na plataforma Netlogo são compostas por diversos agentes, os quais são definidos como entidades que podem seguir instruções. Existem 3 tipos de agentes:

- **tartarugas:** as tartarugas são agentes que podem se movimentar pelo ambiente. Sendo assim, possuem coordenadas, definidas como valores decimais, para identificar o local onde estão posicionadas. Em uma simulação, diferentes tipos de tartarugas podem ser criados e cada tipo pode possuir um comportamento diferenciado;

- **lotes:** o ambiente é bidimensional e formado por uma grade, onde cada célula é um lote ("*patch*"), ou seja, um quadrado que representa um pedaço do terreno por onde as tartarugas podem se movimentar. Os lotes também possuem coordenadas, sempre definidas como valores inteiros;

- **observador:** o observador não possui uma posição definida e é utilizado para vigiar o comportamento das tartarugas e dos lotes.

A diferença na representação das coordenadas das tartarugas e dos lotes determina que uma tartaruga pode estar posicionada em qualquer parte de um lote, não necessariamente no seu centro.

5.5.2 Ambiente

O NetLogo permite que o ambiente seja definido de acordo com uma das topologias existentes:

- **toroidal:** é a topologia padrão. Significa que o ambiente não é cercado, ou seja, quando um agente ultrapassa uma das margens, reaparece na margem oposta. Nesta topologia todos os lotes possuem o mesmo número de vizinhos;

- ***box*:** ou formato não toroidal. Todas as margens do ambiente são cercadas, ou seja, os agentes não podem ultrapassar estas margens. Os lotes localizados nas margens possuem sempre menos de 8 vizinhos: os lotes dos cantos possuem 3 vizinhos e o restante 5 vizinhos;

- **cilindro horizontal e vertical:** são ambientes cercados apenas em um sentido (horizontal ou vertical). No cilindro horizontal a parte de cima do ambiente está conectada com a parte de baixo e os lados direito e esquerdo estão cercados. No cilindro vertical, o lado direito está conectado com o lado esquerdo e a parte de cima e de baixo do ambiente estão cercadas.

São disponibilizadas também algumas formas de vizinhança: ***(i)*** vizinhança de Moore, ***(ii)*** vizinhança de Von Neumann e ***(iii)*** vizinhança por raio.

5.5.3 Formas de Modelagem

Na modelagem baseada em agentes, normalmente utilizada no ambiente NetLogo, cada agente possui um comportamento e o resultado de uma simulação é o comportamento emergente da interação entre estes agentes.

No entanto, também é disponibilizada uma estrutura para modelagem do tipo "*top-down*". Assim, não são definidos comportamentos individuais para os agentes, apenas o comportamento dos conjuntos de agentes.

5.5.4 Utilização de Dados Matriciais

O ambiente NetLogo não fornece real suporte a leitura de dados provenientes de um SIG. Alguns modelos têm sido desenvolvidos fazendo uso de dados matriciais, no entanto é possível importar apenas mapas no formato de imagem. Não são conhecidos modelos que façam uso de dados vetoriais.

5.5.5 Funcionalidade "*Tie*"

A funcionalidade "*Tie*", ainda em caráter experimental, permite conectar duas tartarugas de forma que a movimentação de uma (definida como "tartaruga mestre") interfira diretamente na localização e direção da outra (definida como "tartaruga escrava"). Quando uma "tartaruga mestre" se move, a "tartaruga escrava" também se move, na mesma distância e direção, no entanto a sua direção não se modifica. Quando uma "tartaruga mestre" gira para a direita ou esquerda, mudando de direção, a "tartaruga escrava" também gira, a mesma quantidade no mesmo sentido. A movimentação da "tartaruga mestre" afeta a "tartaruga escrava", no entanto o contrário não acontece.

5.6 OBEUS

A plataforma OBEUS (*Object-Based Environment for Urban Systems*) (BENENSON; ARONOVICH; NOAM, 2005; BENENSON; TORRENS, 2005), desenvolvida no Laboratório de Simulações Ambientais da Universidade de Tel Aviv, Israel, é a implementação das características do modelo GAS (*Geographic Automata Systems*) (BENENSON; TORRENS, 2004; TORRENS; BENENSON, 2005; BENENSON; TORRENS, 2003).

Segundo Benenson e Torrens (2005), existem três características que precisam ser consideradas no desenvolvimento de simulações urbanas:

- uma topologia de entidades que considere o uso do ambiente;

- as relações espaciais entre as entidades;
- os processos que governam a mudança de posição das entidades no ambiente.

De forma isolada, nem SMAs nem ACs conseguiram suprir estas necessidades. Os ACs não fornecem estruturas que possibilitem o deslocamento de entidades. Comumente, os SMAs subestimam a importância do ambiente e dos comportamentos de movimentação das entidades. Desta forma, o modelo GAS objetiva integrar conceitos de SMAs, ACs e SIGs na realização de simulações de sistemas urbanos (BENENSON; TORRENS, 2005).

5.6.1 Modelo GAS

Basicamente, o modelo GAS consiste na interação de diversos tipos de autômatos geográficos. Geralmente, um autômato é caracterizado por estados, informações de entrada e regras de transição. Um autômato geográfico estende estas idéias, possibilitando a consideração explícita do espaço e de comportamentos espaciais.

Neste modelo, a especificação do conjunto de estados considera a localização dos autômatos geográficos no espaço. As informações de entrada podem variar no tempo e espaço, diferentemente do que ocorre nos ACs, na medida em que é possível que as relações de vizinhança também se modifiquem. As regras de transição de estados determinam a mudança de estados não espaciais, enquanto que as regras de movimentação permitem o deslocamento do autômato geográfico no ambiente.

5.6.2 Acoplamento entre o Modelo GAS e os SIGs

A natureza de um SIG, como um tipo especial de SGBD, fornece muitas funcionalidades úteis para o modelo GAS, dentre as quais podem-se destacar as capacidades de armazenar e recuperar informações sobre objetos espaciais.

O modelo GAS e os SIGs são baseados em objetos, os quais representam entidades do mundo real em uma escala microscópica. O modelo GAS considera uma coleção de objetos dinâmicos. Usualmente, os SIGs representam objetos, de forma estática, utilizando o Modelo Entidade-Relacionamento (MER).

Sendo assim, para realizar um acoplamento entre o modelo GAS e as funcionalidades de um SIG é necessário implementar o GAS como um modelo de base de dados orientada a objetos, contendo autômatos geográficos.

5.6.3 Entidades

Os autômatos geográficos podem ser de dois tipos: fixos ou móveis. Os fixos representam objetos que não podem mudar sua localização no ambiente ao longo do tempo. Desta forma, pode-se realizar uma analogia com as células dos ACs. Tais objetos podem possuir regras de transição de estados e de vizinhança, mas não regras de movimentação. Autômatos geográficos móveis (agentes) simbolizam entidades que podem movimentar-se, podendo possuir regras de movimentação e de transição de estados.

5.6.4 Estados e Regras de Transição

As regras de transição de estados podem fazer uso dos dois tipos de autômato existentes. Neste modelo, as regras de transição dos autômatos fixos podem depender, também, do estado dos autômatos móveis. Desta forma, os "agentes" podem ser responsáveis pela determinação do estado dos autômatos fixos. Como apresentado, as transições de estado

podem ser governadas por outros autômatos, sendo este fato importante para representar as decisões/atuações dos agentes sobre o ambiente e o efeito do mesmo sobre os agentes.

5.6.5 Posicionamento das Entidades

São apresentadas duas formas de referência geográfica: direta e indireta. A forma direta é feita por uma representação vetorial do objeto, usando uma lista de coordenadas. Normalmente, autômatos fixos são referenciados desta forma. Atualizar as coordenadas de um autômato móvel seria uma tarefa custosa e complexa, portanto estes autômatos são referenciados de forma indireta, apontando para os autômatos fixos, como apresentado na Figura 5.3.

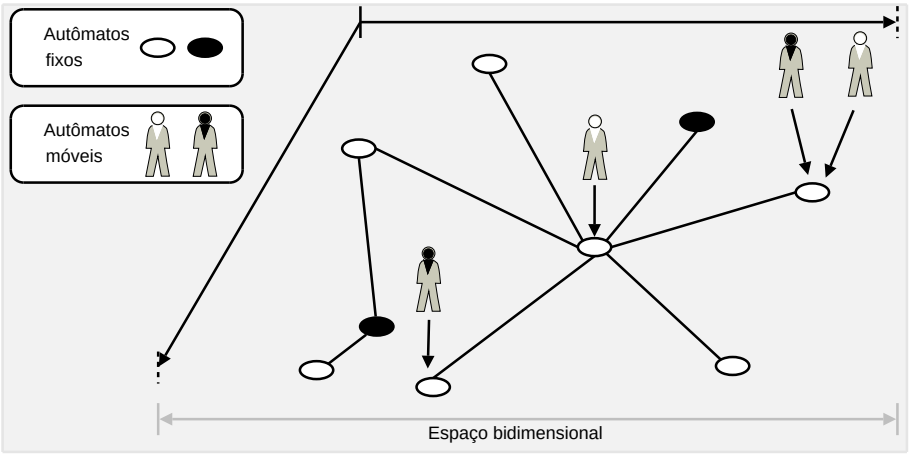

Figura 5.3: Autômatos geográficos fixos e móveis referenciados geograficamente de forma direta e indireta. Adaptada de (BENENSON; TORRENS, 2005).

Não é possível que um autômato móvel referencie um outro autômato móvel, como ilustrado na Figura 5.4. As razões para esta impossibilidade são:

- primeiramente, as relações espaciais entre autômatos fixos também se tornam fixas após o estabelecimento dos autômatos;
- na realidade, este tipo de relação só pode ser estimada e atualizada quando um novo autômato fixo é criado;
- as relações entre autômatos móveis pode ser obtida pela utilização das relações entre *autômatos móveis e fixos* e *autômatos fixos e fixos*;
- este tipo de relação pode se tornar intratável se muitos autômatos móveis se deslocarem simultaneamente.

O desenvolvimento de regras de movimentação mais realísticas se faz necessário para que as simulações apresentem melhores resultados (BENENSON; TORRENS, 2005).

5.6.6 Regras de Vizinhança

Nos autômatos geográficos, um conjunto de vizinhos (a vizinhança) é necessário, pois é este que determina as informações utilizadas pelas regras de vizinhança. Em contraste com a vizinhança simétrica e estática dos ACs, as relações espaciais nos autômatos geográficos podem variar no tempo e no espaço.

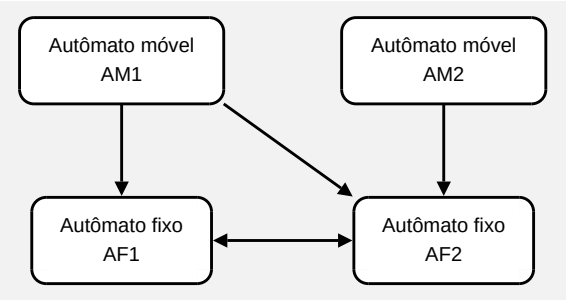

Figura 5.4: Possíveis formas de geo-referenciamento entre os autômatos geográficos. Adaptada de (BENENSON; ARONOVICH; NOAM, 2005).

Regras de vizinhança para autômatos fixos são, relativamente, de fácil determinação. No entanto, existe uma variedade de maneiras pelas quais as regras de vizinhança podem ser expressas: pelos autômatos adjacentes, pela conectividade dos nodos da rede, pela proximidade, etc. Noções espaciais relacionadas com a incorporação de comportamento humano aos autômatos, tais como acessibilidade e visibilidade, podem ser formalmente codificadas nas regras de vizinhança.

A definição das regras de vizinhança nos autômatos geográficos móveis é uma tarefa mais complexa, pois estes autômatos e, conseqüentemente, suas relações de vizinhança são dinâmicas no tempo e espaço. Estas relações podem ser realizadas utilizando os conceitos de vizinhos mais próximos ou a própria distância. Da mesma forma que as regras de movimentação, as regras de vizinhança se tornam mais complexas quando envolvem conceitos de visibilidade e acessibilidade. Nestes casos é mais apropriada a utilização da forma indireta de referenciamento geográfico.

Por exemplo, dois agentes proprietários de imóveis podem ser considerados vizinhos pela relação entre as casas em que moram. Mesmo quando estes agentes estiverem fisicamente separados (por exemplo, quando estes vão trabalhar) eles continuam sendo vizinhos, em virtude da relação entre suas casas.

5.6.7 Dimensão Temporal

Existem várias maneiras do tempo ser implementado em um sistema dinâmico. O tempo pode ser governado por um relógio externo, o qual determina a aplicação simultânea das regras para os autômato a cada passo de tempo. Outra forma de controlar o tempo é utilizando um relógio interno. Sendo assim, o tempo tem um significado diferente para cada autômato. Formalmente, estas abordagens são denominadas de modo síncrono e assíncrono de atualização dos estados dos autômatos, respectivamente.

5.6.8 Estrutura de Classes

O modelo GAS é implementado na plataforma OBEUS por meio de uma estrutura de classes. A Figura 5.5 apresenta, de forma simplifica, o diagrama de classes desta estrutura.

As classes *Estate* e *Agent* representam os autômatos fixos e móveis, respectivamente. Já as classes *EstateEstateRelation* e AgentEstateRelation representam as relações entre *autômatos fixos e fixos* e *autômatos móveis e fixos*, respectivamente. O relacionamento direto entre autômatos móveis não é permitido.

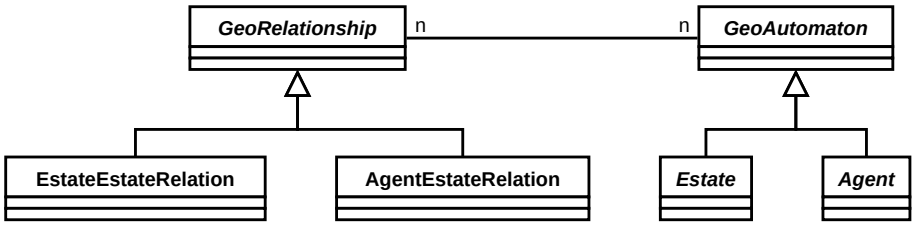

Figura 5.5: Diagrama de classes simplificado da plataforma OBEUS. Adaptado de (BENENSON; TORRENS, 2005).

5.7 Modelos que Integram SMAs e SIG

Da mesma forma que existem plataformas gerais para a criação de simulações baseadas em agentes, existem também modelos para simulação de cenários específicos. Geralmente, tais modelos apresentam características interessantes, que podem ser transferidas para plataformas gerais. São descritos abaixo 3 modelos de simulação, baseados na integração entre SMA e SIG.

O modelo SMA-SIG, proposto em (GONÇALVES, 2003), tem como objetivo apresentar um modelo para simulação do depósito de resíduos em aterros, por indústrias de transformação de pedra, integrando as camadas geográficas dos SIGs com os agentes dos SMAs. Sua principal contribuição é que os dados do SIG podem ser modificados dinamicamente, durante a simulação.

O modelo apresentado em (SILVA, 2003) tem como objetivo a modelagem comportamental de agentes autônomos em ambientes reais. Dois cenários são focados: a movimentação de soldados em um terreno, baseada em algumas regras simplificadas de combate, e a movimentação de pedestres em uma região urbana, baseada em alguns resultados empíricos da literatura. Sendo assim, os comportamentos de movimentação utilizados nestas simulações podem ser utilizados em arquiteturas gerais de simulação, objetivando permitir a criação de representações mais próximas da realidade.

O sistema ExpertCop (VASCONCELOS; FURTADO, 2005) é um simulador da criminalidade em regiões urbanas que também utiliza a abordagem SMA-SIG.

5.8 Considerações do Capítulo

Com a avaliação realizada, foi possível detectar algumas das principais características positivas e negativas das plataformas analisadas, as quais são apresentadas abaixo:

- **plataforma Swarm:**
  - são necessários profundos conhecimentos de programação para o desenvolvimento de modelos;
  - não fornece estruturas para a modelagem de ambientes contínuos;
  - utilizando o módulo Kenge, possibilita a utilização de dados geográficos matriciais;
  - utiliza escalonador de eventos discretos no tempo;
  - agentes auxiliares são utilizados para coletar dados das simulações;
- **plataforma Repast:**

  - possui 3 diferentes implementações, sendo que o Repast Python é considerado o modelo que mais facilita o desenvolvimento de modelos;
  - fornece estruturas para a modelagem de ambientes em forma de grade bidimensional discreta e rede;
  - fornece acesso a dados vetoriais no formato Shapefile;
  - necessita de programas externos para a apresentação das simulações que utilizam dados vetoriais;
  - não permite a movimentação de agentes representados geograficamente por polígonos;
  - utiliza escalonador de eventos discretos no tempo;

- **plataforma SeSAm:**
  - fornece uma interface gráfica que permite o desenvolvimento de modelos por usuários sem conhecimentos de programação;
  - fornece estruturas para a modelagem de ambientes em forma de grade bidimensional discreta;
  - o *plugin* para integração com os SIGs faz uso de diversos outros softwares, dificultando muito a sua instalação;
  - fornece acesso a dados vetoriais no formato Shapefile;
  - utiliza escalonador de eventos discretos no tempo;

- **plataforma NetLogo:**
  - é constituído de um ambiente de simulação, uma linguagem de programação própria e uma biblioteca de classes;
  - permite modelagem *bottom-up* e *top-down*;
  - fornece estruturas para a modelagem de ambientes em forma de grade bidimensional discreta;
  - não possui real integração com SIG, fornece apenas acesso a dados matriciais no formato de imagem;
  - um agente observador é utilizado para coletar informações das simulações;
  - permite que agentes sejam definidos de forma a se movimentarem em conjunto;

- **plataforma OBEUS:**
  - foi desenvolvida focando no desenvolvimento de simulações urbanas;
  - não permite o referenciamento geográfico de forma direta entre autômatos móveis, necessitando assim que todos os autômatos móveis sempre apontem para autômatos fixos.

Neste contexto, alguns pontos foram identificados:

- dificilmente as plataformas fornecem funcionalidades mais abstratas para a movimentação das entidades, sendo que tais funcionalidades são necessárias para a obtenção de simulações mais próximas da realidade;

- quando é possível utilizar dados vetorias, normalmente são utilizados dados no formato Shapefile. No entanto, a conexão dinâmica com um SGBDG permite o melhor aproveitamento dos dados geográficos, facilitando a realização de consultas espaciais complexas;
- funcionalidades que facilitem o desenvolvimento dos modelos são necessárias para que os usuários não precisem de conhecimentos profundos em programação;
- não basta modelar o ambiente de forma contínua e fornecer às entidades um atributo geográfico. É necessário desenvolver percepções e comportamentos, para estas entidades, que levem em consideração tais características;
- é necessário que funcionalidades para a comunicação entre os agentes sejam disponibilizadas nas plataformas de simulação;
- a utilização de um escalonador de eventos discretos no tempo facilita a análise das simulações, permitindo a verificação de informações a cada passo de tempo.

Sendo assim, se faz necessário o desenvolvimento de plataformas que reúnam as características positivas das plataformas avaliadas e que consigam suprir algumas das necessidades encontradas.

6 ARQUITETURA PROPOSTA

Este capítulo aborda a arquitetura de simulação proposta. Inicialmente, são descritas as vantagens da união das áreas de SMAs e Geoprocessamento, bem como reforçadas as motivações deste trabalho. Então, são apresentadas uma visão geral da arquitetura proposta e posteriormente, de forma detalhada, suas diversas características e funcionalidades.

6.1 Introdução

Baseado nas motivações apresentadas no Capítulo 1 e objetivando alcançar uma maior qualidade na realização de simulações, esta arquitetura fará uso de ferramentas de duas áreas distintas: SMAs e Geoprocessamento. Sendo assim, a arquitetura proposta visa fornecer funcionalidades de modo a facilitar a criação e execução de simulações baseadas em agentes na área de Geoprocessamento.

Neste contexto, as ferramentas, técnicas e estruturas existentes na área de Geoprocessamento são utilizadas nas simulações para fornecerem informações geográficas precisas, possibilitando assim a criação de um modelo espacial mais robusto. A abordagem de SMAs é utilizada nas simulações para possibilitar a representação de indivíduos autônomos em um determinado ambiente de simulação. A combinação destas duas abordagens, também chamada de *Geosimulation* (BENENSON; TORRENS, , 2004), vem sendo cada vez mais explorada, possibilitando a realização de simulações mais próximas da realidade.

Entidades reais se movimentam no espaço e é através desta movimentação que conseguem explorar determinados locais do ambiente e ter acesso a características inerentes a determinadas regiões. Como conseqüência desta habilidade, uma entidade pode realizar ações diferentes em locais diferentes, além de possibilitar o aparecimento de outras oportunidades antes de alcançar o seu destino. Desta forma, pela necessidade de se obter uma maior conformidade com a realidade, é necessário que, nestas simulações, os agentes se movimentem em um ambiente geográfico.

Além das motivações e características apresentadas acima, a arquitetura proposta visa preencher algumas lacunas existentes em outras arquiteturas de simulação baseadas em agentes (vide Capítulo 5). Desta forma, esta arquitetura visa reunir as características interessantes dos diversos modelos de simulação analisados, bem como suprir algumas de suas necessidades.

6.2 Visão Geral da Arquitetura

A arquitetura proposta é constituída basicamente por entidades organizadas em camadas. Em uma visão abstrata desta arquitetura, pode-se identificar três camadas principais:

- **camada social:** contém as entidades da sociedade que se deseja representar;
- **camada espacial:** contém as entidades do ambiente que se deseja representar;
- **camada auxiliar:** é uma camada opcional. Contém entidades auxiliares que têm como objetivo facilitar o desenvolvimento do modelo e também "observar" as simulações a fim de coletar dados para a criação de estatísticas. Geralmente, estas entidades não existem no modelo teórico desenvolvido pelos especialistas interessados na simulação.

Em algumas ocasiões as camadas social e espacial podem ser unificadas e representadas com uma única camada. Esta representação única reforça a idéia de Santos (1996), na qual o espaço geográfico é descrito como indivisível dos seres humanos que o habitam e que o modificam a todo momento, como sendo um sistema de objetos e um sistema de ações. Esta caracterização objetiva contrapor os elementos de composição do espaço (os objetos geográficos estáticos que representam o mundo real) aos condicionantes de modificação da estrutura deste espaço (as ações humanas e os processos físicos ao longo do tempo). A Figura 6.1 ilustra a visão geral desta arquitetura.

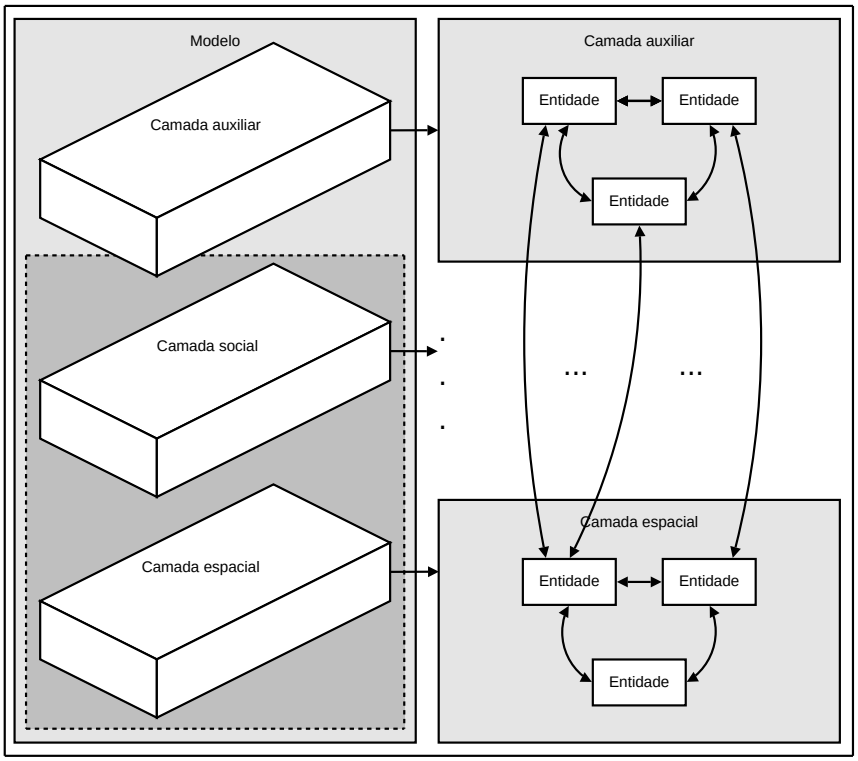

Figura 6.1: Visão geral da arquitetura.

Cabe salientar que cada uma das três camadas pode ser composta por diversas camadas de dados. Por exemplo, a camada social pode ser composta por uma camada de entidades do sexo masculino e outra camada de entidades do sexo feminino. Esta organização em camadas visa simplificar a integração com o Banco de Dados Geográficos (BDG), facilitando e otimizando a realização de consultas e operações espaciais.

6.3 Entidades

Nesta arquitetura, existem dois tipos básicos de entidade:

- **Agentes móveis:** entidades que possuem a capacidade de deslocamento;
- **Agentes fixos:** entidades que não possuem a capacidade de deslocamento.

Tais entidades são especializadas de acordo com sua forma geométrica, como descrito a seguir e apresentado na Figura 6.2:

- **Agente Móvel Ponto:** agente móvel cuja forma é definida por um ponto;
- **Agente Móvel Linha:** agente móvel cuja forma é definida por uma linha;
- **Agente Móvel Polígono:** agente móvel cuja forma é definida por um polígono;
- **Agente Fixo Ponto:** agente fixo cuja forma é definida por um ponto;
- **Agente Fixo Linha:** agente fixo cuja forma é definida por uma linha;
- **Agente Fixo Polígono:** agente fixo cuja forma é definida por um polígono;
- **Entidade Sem Forma:** entidade que não possui forma.

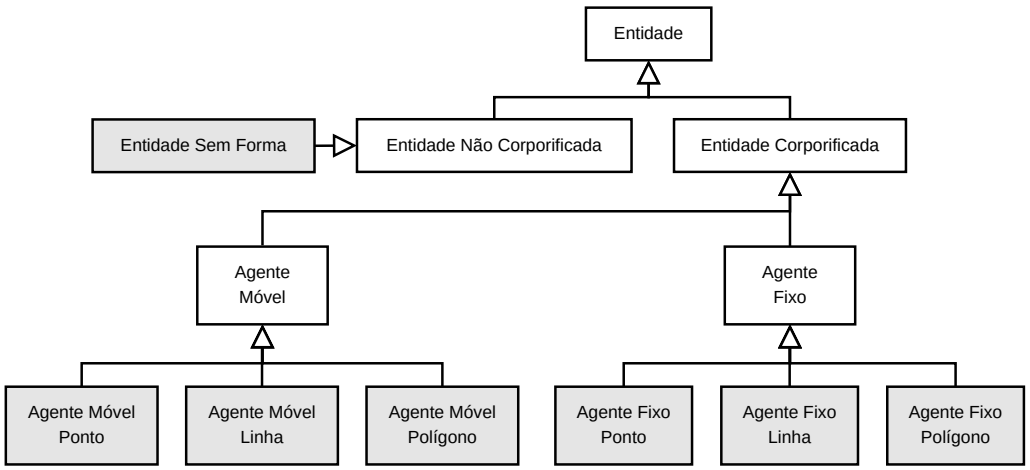

Figura 6.2: Tipos de entidade.

As formas de representação (ponto, linha e polígono) das entidades neste modelo estão diretamente relacionada com as formas utilizadas nos BDGs para representar geometricamente os objetos geográficos do mundo real. Deste modo, a utilização destas formas de representação se justifica por dois motivos principais: por permitir a representação de praticamente todas as entidades do mundo real e ainda por possibilitar o mapeamento direto entre as entidades da simulação e os dados dos BDGs, com relação as suas formas.

Cabe salientar que as *entidades sem forma* não possuem localização, nem podem se locomover no ambiente. Tais entidades tem como objetivo facilitar o desenvolvimento dos modelos de simulação, de modo semelhante aos agentes auxiliares existentes na plataforma Swarm, e geralmente situam-se na camada auxiliar. A criação deste tipo de entidade se justifica na medida em que não é necessário criar/modificar a percepção das

outras entidades a fim de não perceberem as entidades da camada auxiliar. Sendo assim, as *entidades sem forma* podem fazer parte da simulação, coletando dados e informações, e não interferindo na mesma.

Os agentes que não possuem nenhum comportamento são classificados como *recursos* (classificação baseada no modelo SeSAm).

6.4 Organização das Entidades

Foi definida uma hierarquia para organizar as entidades pertencentes a esta arquitetura. Nesta organização:

- cada entidade pertence a um único tipo;
- cada camada do ambiente pode conter entidades de diversos tipos, no entanto um tipo não pode estar contido em mais de uma camada (a organizacão em camadas caracteriza a estratificação das informações em níveis distintos, permitindo flexibilidade e eficiência no acesso aos dados);
- o ambiente é composto por uma ou mais camadas.

Esta organização define a seguinte ordem de pertinência:

$$Entidade \in Tipo \in Camada \in Ambiente$$

como apresentado na Figura 6.3.

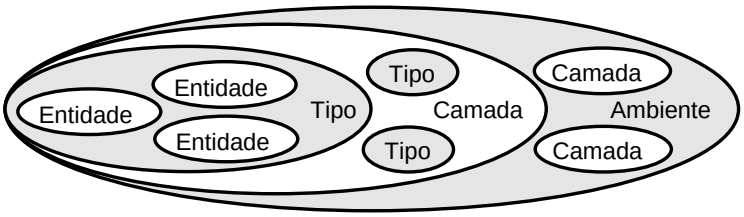

Figura 6.3: Organização das entidades da arquitetura.

Esta forma de organização se justifica pois, além de permitir o fácil acesso aos dados das simulações, ainda facilita a definição das percepções das entidades de forma refinada, como é apresentado na Seção 6.7.

6.5 Ambiente

Como é possível observar na seção anterior, não existe uma real distinção entre as entidades e o ambiente. O ambiente é formado por um conjunto de entidades de diversos tipos e camadas. Esta abordagem retrata a teoria de Santos (1996), apresentada anteriormente.

O ambiente desta arquitetura foi baseado nos seguintes modelos para representação computacional do espaço geográfico:

- **modelo de espaço absoluto:** é considerado um container de objetos, uma estrutura para localizar pontos, trajetórias e objetos;

- **modelo de objetos:** neste modelo, a realidade consiste de entidades individuais, bem definidas e identificáveis, onde as fronteiras são partes essenciais destas entidades. Cada entidade tem suas propriedades e ocupa um determinado lugar no espaço;
- **representação vetorial:** sendo uma das formas mais utilizadas de representação na área de Geoprocessamento, o modelo vetorial é utilizado na tentativa de reproduzir a forma e o posicionamento das entidades com a maior precisão possível.

Estes modelos foram utilizados pois a localização precisa e a modelagem individual das entidades são características fundamentais desta arquitetura.

De acordo com as classificações de ambiente existentes na área de SMAs, o ambiente desta proposta pode ser classificado em:

- **não-determinístico:** o ambiente é parcialmente observável e podem existir várias entidades atuando em paralelo, do ponto de vista de uma entidade;
- **contínuo:** o ambiente é representado de forma contínua, baseado no representação vetorial das entidades (diferentemente do modelo de grade bi-dimensional discreta);
- **dinâmico:** o ambiente pode se modificar enquanto as entidades estão deliberando, devido as ações de outras entidades;
- **parcialmente observável:** os sensores das entidades não necessariamente fornecem o estado completo do ambiente em cada instante. Esta classificação está diretamente relacionada com o tipo de percepção definido para as entidades da simulação;
- **seqüencial:** a ação de uma entidade em um determinado momento pode afetar as ações futuras, principalmente com relação ao posicionamento das entidades no ambiente;
- **não toroidal:** foi utilizado o modelo não toroidal já que os mapas, principal forma de representação na área de Geoprocessamento, geralmente representam ambientes não toroidais.

6.6 Criação/Inserção/Remoção das Entidades

No momento da criação das entidades, são definidos a forma, os atributos e o comportamento de cada uma. Ao serem inseridas na arquitetura de simulação as entidades recebem uma localização e mais alguns atributos opcionais. A percepção é definida quando as entidades já estão inseridas na simulação. Neste contexto, a localização e alguns atributos são dados provenientes do BDG. A forma, o comportamento, a percepção e outros atributos são definidos pelo usuário, como ilustrado na Figura 6.4. Sendo assim, *as entidades de uma simulação representam dinamicamente os dados estáticos do BDG.*

As entidades podem ser criadas e inseridas na simulação de duas formas distintas. Na primeira forma, antes da simulação ser iniciada, deve ser realizada a criação e a inserção das entidades na simulação. Deste modo, as entidades existirão desde o início da simulação. A segunda forma consiste em criar e inserir as entidades dinamicamente, durante a simulação. Para que este processo ocorra é necessário que alguma outra entidade, já existente na simulação, realize as etapas de criação e inserção, dentro de seu comportamento.

A criação/inserção das entidades pode ser dividida em cinco etapas:

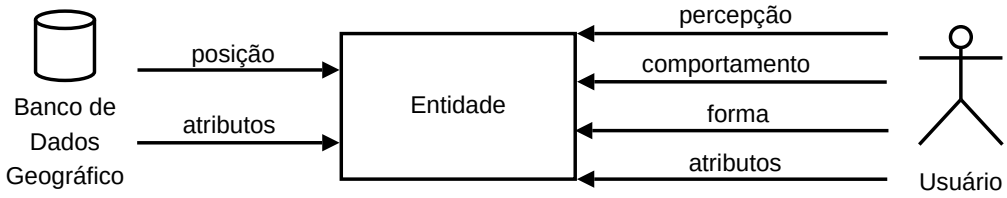

Figura 6.4: Composição da entidade.

- definição das características de um modelo particular de entidade (*definição da classe que representará esta entidade. Esta classe pode herdar de qualquer uma das classes que representam as entidades básicas desta arquitetura.*);
- inserção deste modelo de entidade na arquitetura (*necessário para a criação dinâmica*);
- criação das entidades (*instanciação dos objetos da classe definida no primeiro item*);
- inserção das entidades criadas na arquitetura (*definição de localização e outras características provenientes do BDG*);
- definição das percepções das entidades.

A inserção de um modelo de entidade na arquitetura é realizada objetivando facilitar o processo de criação/inserção dinâmica das entidades. Por exemplo, as entidade criadas dinamicamente deverão fazer uso dos modelos já existentes na arquitetura ao invés de definir um modelo próprio. A criação utilizando modelos é baseada no *modelo de reprodução por cópia*, apresentado por Santos (2003). Este autor definiu que as entidades podem ser criadas por usuários, através da produção, ou por outras entidades, através da reprodução por cópia, particionamento e junção:

- ***produção:*** é a forma mais simples de criar uma entidade. Nesta forma, as entidades básicas são criadas pelos usuários a partir da arquitetura de criação. Caso seja necessário uma entidade específica, deve-se desenvolver tal tipo específico de entidade, baseado nos tipos básicos fornecidos pela arquitetura;
- ***reprodução por cópia:*** a partir de entidades já existentes (modelos existentes, que podem ser copiados) criam-se as novas entidades, que herdam as características do modelo utilizado como base;
- ***reprodução por particionamento:*** a entidade que irá ser particionada inicia em um estado fundamental. Então esta entidade se duplica e é dividida, dando origem a duas entidades no estado fundamental (a própria entidade e uma nova);
- ***reprodução por junção:*** neste tipo de reprodução, duas entidades se juntam para criar uma terceira, que herda seletivamente características de ambas. As duas entidades iniciais continuam existindo após este processo.

A remoção das entidades também pode ocorrer de duas formas distintas: ao final da execução das atividades que compõe o comportamento de uma entidade ou dinamicamente, durante a execução da simulação (mesmo que a entidade não tenha finalizado suas

atividades). Neste último caso, alguma entidade existente na simulação deve realizar o processo de remoção (pode ser a própria entidade que será removida).

Sendo possível a criação e remoção dinâmica das entidades, esta arquitetura também permite a criação, remoção e atribuição dinâmica das percepções.

As funcionalidade apresentadas acima são extremamente interessantes em simulações de crescimento urbano, nas quais, durante as simulações, o número de entidades existentes pode variar.

6.7 Percepções das Entidades

Para que as entidades possam agir de forma autônoma no ambiente é necessário que sejam capazes de percebê-lo. As condições de percepção definidas nesta arquitetura se relacionam diretamente com as entidades e não com as características destas. Deste modo, só é possível filtrar quais entidades, mas não quais características das entidades, podem ser percebidas.

A definição e atribuição da percepção das entidades ocorre após as mesmas serem adicionadas na simulação. Para a determinação de uma percepção deve-se definir ao menos um *tipo de percepção* e uma *expressão geral*. Existem quatro possíveis tipos de percepção:

- ***percepção por distância:*** só é possível perceber as entidades que estejam a uma determinada distância;
- ***percepção por tipo:*** só é possível perceber as entidades que pertencem aos tipos especificados;
- ***percepção por camada:*** só é possível perceber as entidade que pertencem às camadas especificadas;
- ***percepção por entidade:*** só é possível perceber as entidades especificadas.

A expressão geral define qual será a percepção final, possibilitando integrar os quatro tipos existentes. Para tal, é definida uma expressão que conta com a utilização de 8 palavras-chave:

- ***PBD*:** conjunto com todas as entidades retornadas pela *percepção por distância*;
- ***PBK*:** conjunto com todas as entidades retornadas pela *percepção por tipo*;
- ***PBL*:** conjunto com todas as entidades retornadas pela *percepção por camada*;
- ***PBENT*:** conjunto com todas as entidades retornadas pela *percepção por entidade*;
- ***PBENV*:** conjunto com todas as entidades da simulação (menos entidades sem forma);
- ***AND*:** retorna o resultado da interseção ($\cap$) de dois conjuntos;
- ***OR*:** retorna o resultado da união ($\cup$) de dois conjuntos;
- ***NOT*:** retorna o complemento de um conjunto, relativamente ao **`PBENV`**.

Na expressão geral podem ser utilizados parênteses para determinar a ordem de execução das operações. Por exemplo, a expressão

```
(PBD AND (PBK AND (NOT(PBENT))))
```

representa a percepção de todas as entidades dos tipos definidos em **PBK** que estiverem na distância definida em **PBD**, menos as entidades definidas em **PBENT**.

A percepção por distância pode ser baseada em um raio de distância ou no modelo cardioidal, o qual sugere que a visão humana tem um alcance máximo frontal, que se reduz lateralmente até um ponto cego atrás da cabeça. A Figura 6.5 apresenta o modelo cardioidal e a simplificação proposta por Silva (2003), adotada nesta arquitetura. A simplificação consiste em utilizar um círculo para representar o campo perceptivo e posicionar a entidade não coincidente com o centro do círculo e foi adotada por motivos de desempenho computacional, já que os cálculos do modelo cardioidal são complexos e custosos.

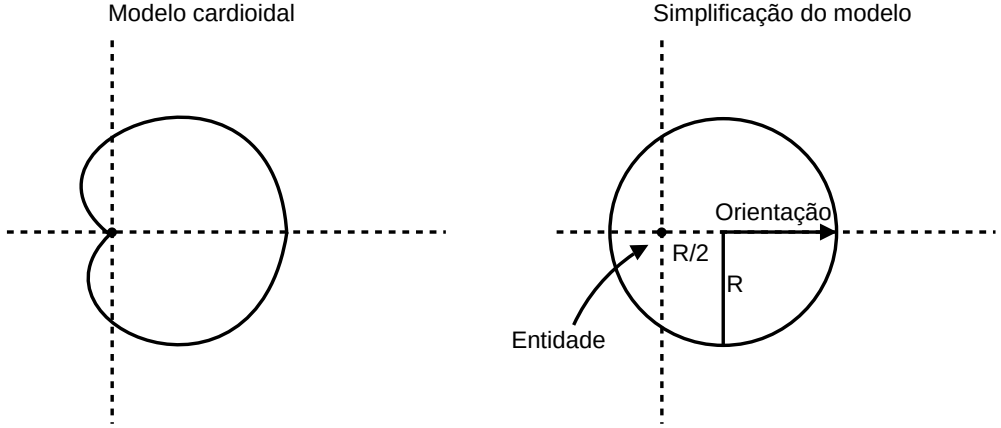

Figura 6.5: Modelo cardioidal e simplificação adotada.

As percepções podem ser definidas em 4 níveis distintos de prioridade: ***(i)*** para uma entidade, ***(ii)*** para todas as entidades de um tipo, ***(iii)*** para todas as entidades de uma camada ou ***(iv)*** para todas as entidades da simulação.

Tendo por base a organização hierárquica desta arquitetura, apresentada na Figura 6.3, é possível definir qual percepção será aplicada a uma determinada entidade, caso esta entidade seja referenciada em mais de uma percepção. Isto ocorre pois para cada entidade é possível aplicar uma única percepção, a de maior prioridade. A Figura 6.6 representa a ordem de prioridade da aplicação das percepções, a qual significa que:

- uma percepção definida para uma *entidade* sobrepõe qualquer outra percepção que faça referência a tal entidade;
- uma percepção definida para um *tipo* só não sobrepõe as percepções definidas para as entidades;
- uma percepção definida para uma *camada* sobrepõe apenas a percepção definida para todo o ambiente;
- uma percepção definida para todo o *ambiente* é válida apenas para as entidades às quais nenhuma outra percepção foi definida.

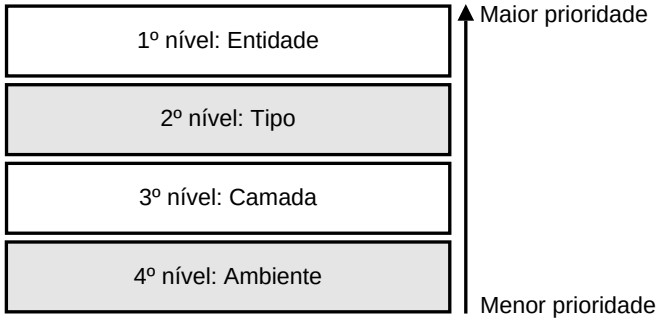

Figura 6.6: Ordem das prioridades.

De forma geral, a criação deste novo modelo para lidar com as percepções se justifica pois permite a criação de percepções de forma altamente refinada, com a utilização da expressão geral, bem como a atribuição destas percepções em diferentes níveis de granularidade (por entidade, tipo, camada, ambiente), baseado na ordem de prioridades.

6.8 Comunicação entre as Entidades

Esta arquitetura fornece um serviço de comunicação baseado em duas interfaces distintas para a troca de mensagens entre as entidades existentes: via passagem direta de mensagens e por meio do acesso a estrutura de quadro negro (*blackboard*) (vide Capítulo 3).

De modo geral, as entidades utilizam o *blackboard* para se comunicarem, porém quando as mensagens são emergenciais e/ou privativas pode ser utilizada a troca de mensagens, por permitir a rápida transmissão e a privacidade de informações.

O *blackboard* é utilizado no processo de comunicação indireto entre as entidades, cujas interações ocorrem através de uma estrutura de dados central, única e compartilhada. Neste contexto, o *blackboard* implementa alguns serviços de filtragem para que as entidades busquem determinadas mensagens.

As entidades possuem caixas de mensagens para armazenar as mensagens enviadas e recebidas. Além disto, os agentes possuem funcionalidades básicas, a fim de permitir o envio e a recepção das mensagens.

A comunicação é parte essencial dos SMAs, permitindo a troca de informações e a coordenação de atividades entre as entidades. No entanto, diversos modelos não consideram sua importância no desenvolvimento das simulações, principalmente modelos de simulação que focam especialmente a modelagem do ambiente (SILVA, 2003; GONÇALVES, 2003; TORRENS; BENENSON, 2005).

6.9 Movimentação dos Agentes Móveis

A modelagem de comportamentos relacionados com a movimentação das entidades no ambiente é fortemente dependente do fenômeno que se objetiva simular. Entretanto, é possível definir um conjunto genérico de comportamentos que pode ser utilizado independentemente dos fenômenos simulados. Estes comportamentos definem ações básicas e podem ser associados de forma a definirem novos comportamentos, mais complexos e específicos para determinados tipos de simulação (SILVA, 2003).

Em (REYNOLDS, 1999) são apresentados diversos comportamentos genéricos relacio-

nados com movimentação. Alguns destes comportamentos foram adaptados e definidos para os agentes móveis desta arquitetura.

Inicialmente, é preciso apresentar alguns atributos (estados internos) relacionados ao posicionamento das entidades no ambiente:

- **posição:** armazena a coordenada atual da entidade no ambiente (definido para todas as entidades) Esta coordenada é um par de valores para os eixos X e Y;
- **ângulo:** armazena o ângulo (em graus) entre o eixo Y e a direção da entidade, no sentido horário (definido somente para os agentes móveis);
- **ângulo máximo de rotação:** armazena o desvio angular máximo que uma entidade pode efetuar (definido somente para os agentes móveis).

Estes atributos, apresentados na Figura 6.7, são definidos quando as entidades são inseridas na simulação. A partir destes atributos podem ser criados alguns comportamentos básicos para os agentes móveis:

- **definir ângulo:** recebe o valor de um ângulo. Este comportamento define o atributo ângulo;
- **movimentar:** recebe o valor de uma distância. Este comportamento movimenta a entidade tal distância na direção definida pelo seu atributo ângulo;
- **rotacionar:** recebe o valor de um ângulo. Este comportamento rotaciona a entidade de acordo com este ângulo, no sentido horário.

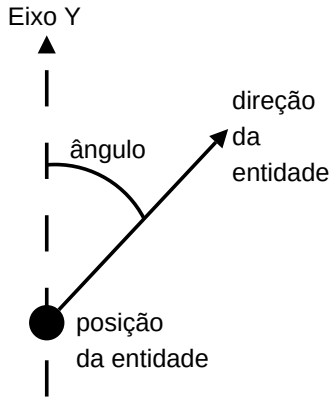

Figura 6.7: Atributos relacionados com o posicionamento das entidades.

Baseados nestes comportamentos básicos, foram definidos alguns movimentos mais complexos, descritos de forma abstrata a seguir:

- **procurar (*seek*):** recebe a posição de uma entidade. Este comportamento rotaciona a entidade até esta direcionar-se para a posição recebida. Este comportamento não movimenta realmente a entidade, apenas rotaciona a mesma;

- **evitar obstáculos (*obstacle avoidance*):** recebe uma distância, um ângulo e uma lista de tipos de obstáculos. Este comportamento verifica a possibilidade de mover a entidade tal distância na direção definida. Desta forma, é verificado se não ocorrerão colisões com os obstáculos definidos e percebidos pela entidade que se movimentará. Caso nenhuma colisão seja detectada, a entidade recebe um sinal de que pode se movimentar. Caso seja detectado que tal colisão ocorrerá, é definido um sentido aleatório (horário ou anti-horário) e rotaciona-se a entidade neste sentido (o ângulo recebido como parâmetro) até que não ocorram mais colisões. Então a entidade recebe um sinal de que pode se movimentar. Existe a possibilidade de a entidade ser rotacionada 360^o e mesmo assim continuarem ocorrendo colisões. Neste caso, a entidade recebe um sinal de que não pode ser movimentar. Cabe ressaltar ainda este comportamento examina apenas os obstáculos percebidos pela entidade que irá se movimentar. Sendo assim, a percepção é essencial para este comportamento;

- **movimentação aleatória (*random*):** rotaciona a entidade de forma totalmente aleatória (respeitando o ângulo máximo de rotação). Esta movimentação não gera resultados muito interessantes. Este comportamento não movimenta realmente a entidade, apenas rotaciona a mesma;

- **perambular (*wander*):** rotaciona a entidade de forma quase aleatória, definindo algumas restrições para que a entidade tenha uma movimentação mais linear. Neste tipo de comportamento a entidade é movimentada quase sempre na mesma direção, podendo ocorrer apenas pequenas variações aleatórias na direção a cada movimento. Este comportamento não movimenta realmente a entidade, apenas rotaciona a mesma.

Todos estes comportamentos devem ser utilizados em conjunto com o comportamento ***movimentar***. A partir dos comportamentos definidos acima, diversos outros podem ser definidos facilmente, de acordo com as necessidades das simulações.

O desenvolvimento destes comportamentos de movimentação se justificam pelas necessidades apresentadas por Benenson e Torrens (2005): "*o desenvolvimento de regras de movimentação mais realísticas se faz necessário para que as simulações apresentem melhores resultados*".

6.10 Escalonamento das Entidades

Grande parte das ferramentas de simulação realiza o escalonamento das entidades de forma discreta no tempo (LAW; KELTON, 2000). Este também é modelo de escalonamento utilizado nesta arquitetura, pois proporciona uma maior facilidade na verificação do estado das entidades, visto que as simulações podem ser executadas e os dados observados a cada época. Desta forma, a cada época, cada entidade realiza sua "ação atômica" e notifica o escalonador para que a próxima entidade possa ser escalonada. O escalonamento nesta arquitetura pode ocorrer de dois modos:

- **ordenado:** a cada época, as entidades executam suas ações na ordem em que foram inseridas na simulação;

- **aleatório:** a cada época, as entidades executam suas ações em uma ordem diferente, sorteada aleatoriamente.

Baseado nos problemas existentes na área de escalonamento apresentados em (MICHEL; FERBER; GUTKNECHT, 2001), a criação destes dois modos de escalonar as entidades é necessária para garantir que a ordem de execução não influenciará nos resultados das simulações.

6.11 Comportamento das Entidades

As plataformas apresentadas no Capítulo 5 não especificam o tipo de comportamento das entidades. Nelas podem ser desenvolvidos tanto entidades reativas como cognitivas, dependendo do objetivo de cada simulação. No modelo proposto o comportamento das entidades também não é especificado *a priori*, objetivando proporcionar uma maior liberdade aos usuários.

O comportamento é definido utilizando-se a sintaxe da linguagem Python, tão simples que é quase considerada um pseudo-código. Mesmo não sendo um mecanismo tão abstrato como o oferecido pela plataforma SeSAm, no qual nem é preciso ter conhecimento de uma linguagem de programação para a criação do comportamento das entidades, a simplicidade da linguagem Python facilita muito na criação (programação) do comportamento das entidades, já que é considerada uma linguagem de altíssimo nível (*VHLL - Very High Level Language*) (PSF, 2006). Sendo assim, desenvolver o comportamento das entidades na linguagem Python geralmente é bem mais simples que em outras linguagens de programação. Um exemplo de código é apresentado na Figura 6.8.

```
01.    class Agente(PointMobileAgent)
02.        def run(self):
03.            self.setRandomDirection()
04.
05.            while (self.getTime() <= 100):
06.                self.wander()
07.                if (self.avoidObstacles(45)):
08.                    self.move(2)
09.                yield True
10.
11.            while(1):
12.                self.directTo((400, 400))
13.                if (self.avoidObstacles(45)):
14.                    self.move(2)
15.                yield True
```

Figura 6.8: Exemplo de código Python que implementa o comportamento de um agente.

A linha `01` apresenta a definição da classe `Agente`, herdando da classe `PointMobileAgent`, que representa a entidade básica agente móvel ponto. A linha `02` apresenta a definição do início do método `run` dentro da classe `Agente`. A linha `03` define uma direção aleatória para o agente. O trecho de código entre as linhas `05` e `09` faz com que o agente se movimente aleatoriamente pelo ambiente sem bater nos obstáculos percebidos (até a época `100`). O trecho de código entre as linhas `11` e `15` faz com que o agente se dirija para o ponto `(400, 400)` do ambiente (até acabar a simulação).

É importante observar que a instrução `yield True` faz com que o agente notifique o escalonador que já executou suas ações para aquela época e que o escalonador pode dar a vez para a próxima entidade.

Desta forma, a utilização da linguagem Python no desenvolvimento do comportamento das entidades se justifica por facilitar tal tarefa, visto que esta linguagem possui uma sintaxe simples, permitindo ao usuário focar no problema a ser resolvido e não em aspectos da linguagem.

6.12 Agentes Móveis Adjacentes

Existe a possibilidade de dois ou mais **agentes móveis** serem definidos como adjacentes. Ocorrendo isto, a movimentação e a mudança de forma de um destes agentes resulta na modificação da forma dos adjacentes. As Figuras 6.9 e 6.10 apresentam as diferenças na movimentação e na mudança de forma entre agentes não adjacentes e agentes adjacentes.

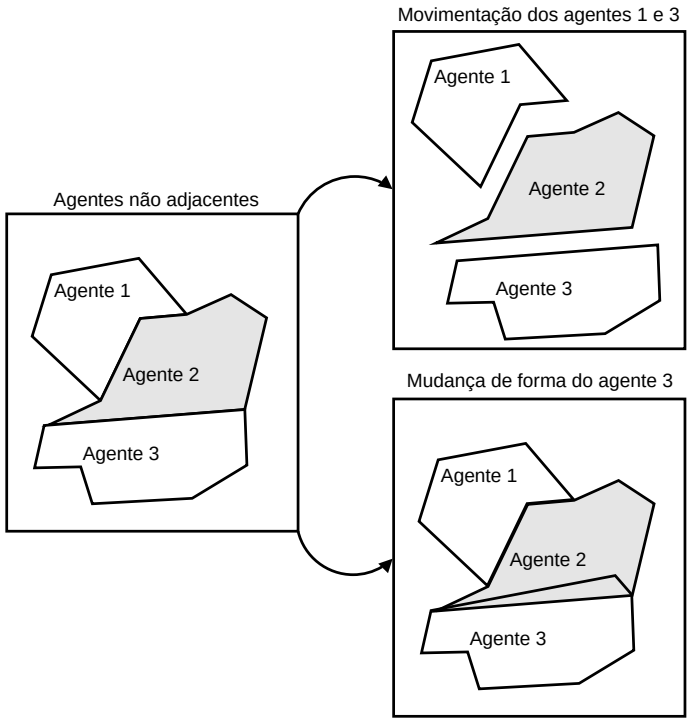

Figura 6.9: Movimentação e mudança de forma de agentes móveis não adjacentes.

A motivação principal para o desenvolvimento desta funcionalidade é facilitar a criação de agentes que possuam limites adjacentes, tais como divisões territoriais de bairros, cidades, a modelagem de margens de lagos, a divisão de áreas de campo, etc. Nestas entidades a modificação de um determinado limite em um agente produz automaticamente a modificação dos limites de seus agentes adjacentes (por exemplo, na compra de um lote de campo, a fazenda que comprar terá sua área aumentada e a fazenda que vender terá sua área diminuída).

A definição dos agentes adjacentes é armazenada em uma estrutura lógica de localização, na forma de grafo, objetivando facilitar a busca pelos agentes móveis adjacentes de um determinado agente móvel. Tal estrutura é apresentada na Figura 6.11.

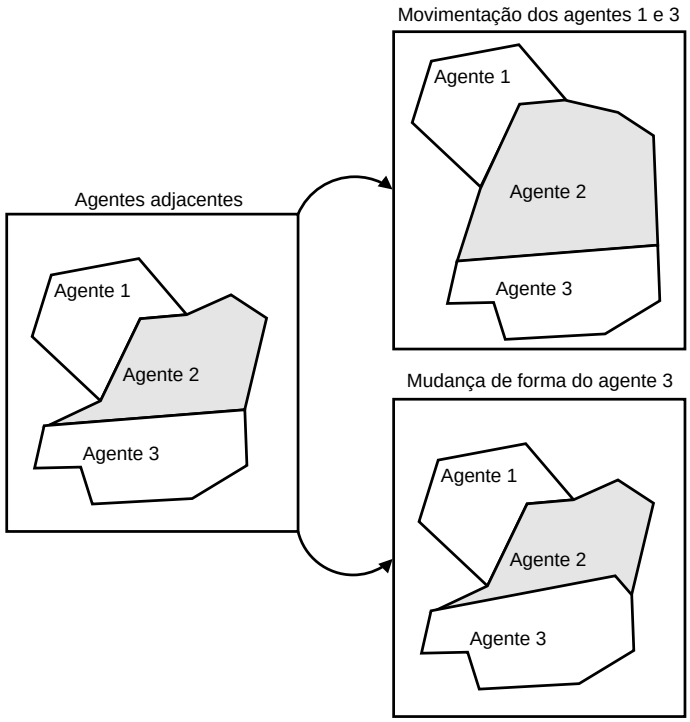

Figura 6.10: Movimentação e mudança de forma de agentes móveis adjacentes.

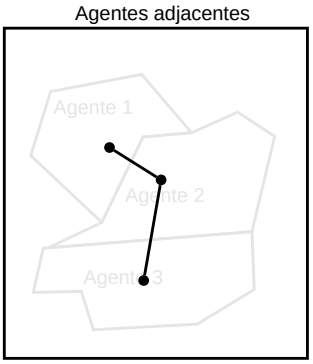

Figura 6.11: Estrutura lógica de localização resultante da definição dos agentes adjacentes.

6.13 Integração com o Banco de Dados Geográficos

A integração entre simulações baseadas em agentes e BDGs não é um processo simples. Isto ocorre devido à falta de mecanismos nos BDGs convencionais para a representação de fenômenos dinâmicos. Embora Koch (apud MANDL, 1996) apresente diferentes classificações, tal integração pode ser classificada basicamente em estática e dinâmica:

- **na integração estática**, os dados geográficos são importados pelo simulador no início e possívelmente atualizados no final da simulação (durante a simulação não é possível acessar o BDG). Normalmente, esta solução é constituída por dois componentes independentes: (***i***) o simulador que implementa funcionalidades para importar/exportar dados através do sistema de arquivos; (***ii***) o BDG usado para gerar o modelo geográfico;

- **na integração dinâmica**, o acesso aos dados geográficos é feito em tempo de exe-

cução (durante a simulação). Neste caso, a vasta gama de operadores e funcionalidades do BDG pode ser utilizada pelas entidades existentes nas simulações.

Deste modo, o desenvolvimento de uma arquitetura de simulação que integre SMAs e BDG de forma dinâmica é muito vantajoso pois potencializa ajustes mais rápidos do modelo à realidade do estudo de caso e simplifica o processo de simulação de vários cenários (RODRIGUES, 1999). Numa arquitetura de simulação com integração dinâmica, as entidades podem fazer uso de percepções espacial executando operadores espaciais diretamente no BDG. Caso contrário, estes operadores (ou pelo menos um subconjunto destes) precisariam ser desenvolvidos dentro da arquitetura de simulação.

A arquitetura proposta foi desenvolvida baseada no modelo de integração dinâmica, no entanto este modelo acarreta uma grande perda de desempenho devido a necessidade de atualizar todos os dados no BDG, cada vez que estes dados forem modificados durante as simulações.

Tendo em vista o problema de desempenho destacado anteriormente, a arquitetura proposta fornece às entidades da simulação a possibilidade de atualizar ou não cada um de seus atributos provenientes do BDG. Neste contexto, os atributos podem ser atualizados dentro do simulador ou então no BDG. Apenas os dados sobre o posicionamento das entidades sempre são atualizados no BDG, visto que é este que executa os operadores espaciais.

Cabe salientar que, existindo estas duas possibilidades de atualização de dados, é necessário que os usuários tenham o controle de onde os atributos serão atualizados e consultados. Se um dado for atualizado apenas no simulador, é necessário que este seja recuperado do simulador e não do BDG, pelo menos até ser atualizado no BDG também.

Para cada atributo proveniente do BDG, as entidades da arquitetura podem:

- definir o valor do atributo no simulador;
- recuperar o atributo a partir de seu valor no simulador;
- definir o valor do atributo no BDG;
- recuperar o atributo a partir de seu valor no BDG.

É importante ressaltar também que apenas os dados provenientes do BDG podem ser atualizados das duas formas apresentadas acima. Os dados criados apenas dentro da simulação só podem ser atualizados no simulador, visto que eles nem existem dentro do BDG.

6.14 Considerações do Capítulo

Neste capítulo foram apresentadas as características do modelo proposto, bem como a justificativa para cada uma delas, podendo-se destacar algumas.

A arquitetura proposta visa fornecer funcionalidades de modo a facilitar a criação de simulações baseadas em agentes para a área de Geoprocessamento. Em especial, Rodrigues (1999) conclui que os agentes podem ser utilizados em simulações que usam modelos geográficos gerados pelos BDGs, e que os agentes podem gerar dados espaciais para serem utilizados pelos BDG. Desta forma, tanto os modelos geográficos como os modelos baseados em agentes podem se beneficiar do acoplamento entre SMA e BDG.

A integração dinâmica entre SMA e BDG proporciona diversas vantagens, como apresentado anteriormente. Além destas vantagens, esta arquitetura permite a melhora no desempenho das simulações através das diferentes formas de atualização dos dados.

A comunicação entre as entidades, fator este que nem sempre é levado em consideração no desenvolvimento das arquiteturas de simulação, foi desenvolvido baseado no padrão FIPA, permitindo assim a troca de informações durante as simulações.

A facilidade sintática da linguagem Python permite a criação de modelos de simulação sem a necessidade de conhecimentos profundos de programação.

A movimentação das entidades é necessária para permitir que as entidades explorem todo o ambiente, simulando assim situações mais próximas da realidade.

A organização das entidades e as percepções são baseadas em camadas, sendo então totalmente compatíveis com a organização dos dados em um BDG.

Todas estas características visam fornecer uma arquitetura mais robusta para a criação de simulações, objetivando preencher algumas lacunas deixadas por outras arquiteturas de simulação. Para avaliar esta arquitetura, um protótipo foi desenvolvido e alguns estudos de caso realizados.

7 PROTÓTIPO DESENVOLVIDO

Este capítulo descreve o protótipo da arquitetura multiagentes para a criação de simulações na área de Geoprocessamento, desenvolvido com o objetivo de ilustrar e possibilitar a avaliação das funcionalidades desta arquitetura. Além dos detalhes desta implementação, também é apresentado o aparato tecnológico utilizado.

7.1 Introdução

O desenvolvimento de um protótipo é especialmente importante pois permite avaliar as características da arquitetura que está sendo proposta, visto que muitas vezes é difícil realizar tal avaliação sem a execução de testes acerca de seu comportamento em situações reais. Desta forma, com o intuito de obter com maior clareza os pontos positivos e negativos desta arquitetura de simulação, um protótipo foi desenvolvido. Este protótipo é composto de uma interface de programação (do inglês, "***A**pplication **P**rogram **I**nterface*") e de um ambiente de execução para as simulações.

7.2 Visão Geral do Protótipo

Em um alto nível de abstração, pode-se afirmar que o protótipo desenvolvido é composto por 3 módulos[4] principais: ***(i)*** "Interface Gráfica", ***(ii)*** "Entidades" e ***(iii)*** "Plataforma". Os 3 módulos e as relações entre eles podem ser observados na Figura 7.1. O módulo "Interface Gráfica" pode acessar dados tanto do módulo "Entidades" quanto do "Plataforma". Estes dois últimos módulos podem trocar informações entre si. O módulo "Plataforma" provê acesso aos dados dos BDGs interno e externo. O diagrama de classes do protótipo desenvolvido é apresentado na Figura A.1.

7.2.1 Módulo Entidades

O módulo "Entidades" tem como principal finalidade fornecer as classes básicas necessárias para a criação das entidades que serão simuladas. As classes que implementam as entidades básicas estão organizadas em 3 submódulos, como apresentado na Figura 7.2: ***(i)*** "Entidades Sem Forma", ***(ii)*** "Agentes Fixos" e ***(iii)*** "Agentes Móveis".

O submódulo "Entidades Sem Forma" contém apenas a classe que implementa a entidade sem forma. O submódulo "Agentes Fixos" contém as classes que implementam o agente fixo ponto, o agente fixo linha e o agente fixo polígono. O submódulo "Agentes Móveis" contém as classes que implementam o agente móvel ponto, o agente móvel linha

[4]Neste contexto, um módulo nada mais é do que um conjunto de classes. Além disto, um módulo pode ser composto por outros submódulos.

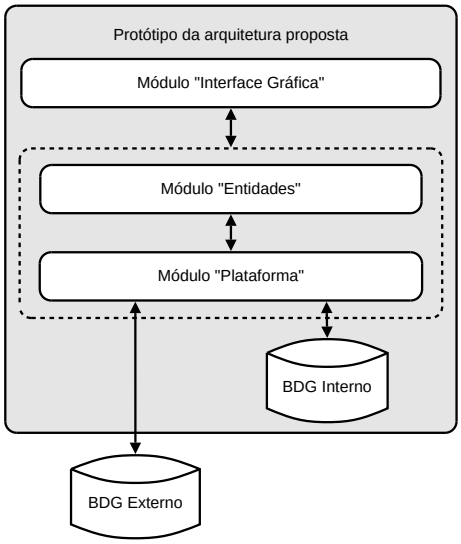

Figura 7.1: Diagrama informal abstrato do protótipo desenvolvido.

e o agente móvel polígono.

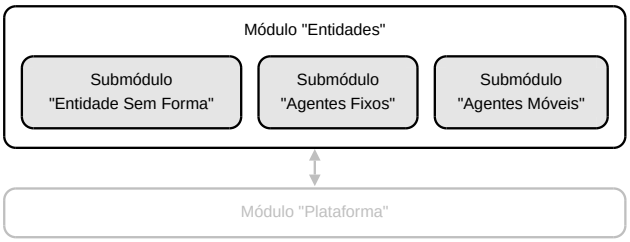

Figura 7.2: Diagrama informal abstrato do módulo “Entidades”.

Deste modo, o usuário pode criar classes que herdem das classes básicas disponibilizadas, dependendo apenas do formato da entidade e da possibilidade ou não de locomoção. Ou seja, o usuário cria suas classes, cria novos atributos e métodos, desenvolve o comportamento da entidade, instancia e adiciona o(s) objeto(s) na plataforma. Neste contexto, os “Agentes Fixos” e os “Agentes Móveis” têm os dados de suas posições provenientes de um BDG (neste caso, o BDG externo, informado pelo usuário).

Todas as entidades existentes são inseridas e gerenciadas pela plataforma, ou seja, pelo módulo “Plataforma”. Outra característica comum é que, a princípio, todas as entidades possuem comportamento, percepções e podem se comunicar.

7.2.2 Módulo Plataforma

O módulo “Plataforma” reune diversas classes que implementam os serviços necessários para a criação e execução das simulações. Estas classes estão organizadas em submódulos, como é apresentado na Figura 7.3.

7.2.2.1 Submódulo Comunicação

O submódulo “Comunicação” é uma versão modificada da plataforma ZMAS (*ZOPE MultiAgent Systems*) (BASTOS, 2004). O ZMAS fornece uma plataforma de comunicação para SMAs, fundamentada em especificações de padrões da FIPA, implementada para o *framework* ZOPE (*Z Object Publishing Environment*) (ZOPE CORPORATION, 2007).

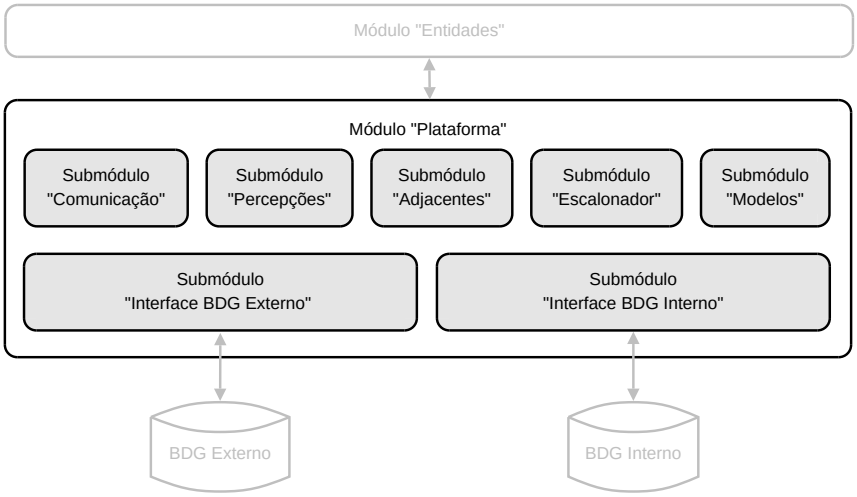

Figura 7.3: Diagrama informal abstrato do módulo “Plataforma”.

A modificação realizada consistiu em portar a versão em ZOPE para Python, filtrando apenas as funcionalidades para comunicação que são utilizadas neste protótipo, tais como troca direta de mensagens e a estrutura de *blackboard*.

Neste contexto, uma mensagem é composta por duas partes: um envelope e uma carga útil (*payload*). A carga útil contém a mensagem FIPA-ACL (FIPA, 2007) propriamente dita, codificada em uma determinada representação (String ou XML). O envelope carrega as informações necessárias para transportar a mensagem para o seu destino.

7.2.2.2 Submódulo Percepções

O submódulo “Percepções” implementa as características relacionadas com o gerenciamento das percepções das entidades, descritas no capítulo anterior.

7.2.2.3 Submódulo Adjacentes

O submódulo “Adjacentes” implementa funcionalidades para manipulação das entidades definidas como adjacentes. As entidades adjacentes possuem comportamentos de movimentação e mudança de forma diferenciados, necessitando de algumas funcionalidades específicas para o seu gerenciamento.

7.2.2.4 Submódulo Escalonador

O submódulo “Escalonador” implementa um escalonador de eventos discretos no tempo, que gerencia a execução das entidades existentes na simulação. O escalonamento pode ocorrer de dois modos, conforme especificado no modelo proposto: ***(i)*** ordenado e ***(ii)*** aleatório. Com a utilização deste escalonador é possível que a simulação seja realizada passo a passo, permitindo assim um melhor acompanhamento e visualização dos atributos das entidades.

7.2.2.5 Submódulo Modelos

O submódulo “Modelos” implementa funcionalidades objetivando facilitar a criação de novas entidades dinâmicamente (durante as simulações). É este submódulo que armazena cópias dos modelos de entidade (classes), desenvolvidos pelo usuário, para que estes possam ser recuperados no meio de uma simulação e utilizados na criação de novas

entidades.

7.2.2.6 Submódulos Interface BDG Externo e Interno

Os submódulos "Interface BDG Externo" e "Interface BDG Interno" implementam camadas abstratas com o objetivo de facilitar o acesso aos BDGs. Estes módulos automatizam diversas tarefas relacionadas com os BDGs, comumente necessárias na criação e execução das simulações. No entanto, também permitem a realização de consultas utilizando a própria linguagem SQL.

7.2.3 Módulo Interface Gráfica

O módulo "Interface Gráfica" fornece classes para a criação de uma interface gráfica que facilite a criação e execução de simulações, bem como a visualização de seus dados e resultados. Esta interface gráfica é dividida em duas partes principais: ***(i)*** uma barra de ferramentas e ***(ii)*** diversas abas com informações sobre as simulações.

A barra de ferramentas possui botões principalmente para: ***(i)*** criar uma nova simulação, ***(ii)*** configurar alguns parâmetros da simulação, ***(iii)*** abrir o arquivo Python, ***(iv)*** iniciar a simulação, ***(v)*** parar a simulação e ***(vi)*** salvar arquivo de log.

As abas apresentam principalmente: ***(i)*** informações sobre a plataforma (vide Figura B.1), ***(ii)*** imagem do posicionamento das entidades (vide Figura B.2), ***(iii)*** informações sobre o estado dos agentes móveis (vide Figura B.3), agentes fixos e entidades sem forma, ***(iv)*** informações sobre a comunicação entre as entidades (vide Figura B.4), ***(v)*** visualização e edição do arquivo Python (vide Figura B.5) e ***(iv)*** visualização do arquivo de log (vide Figura B.6).

7.2.4 Entrada e Saída de Dados

A entrada de dados para a criação das simulações é composta por duas partes:

- **base de dados geográficos:** no contexto deste protótipo, chamada de BDG externo. Este BDG armazena o posicionamento das entidades e possivelmente mais alguns atributos que serão atribuídos às entidades das simulações;
- **arquivo Python:** neste arquivo é realizada a criação das simulações, das entidades, definição de comportamentos, das percepções, etc. A estrutura deste arquivo pode ser definida basicamente como:
  - criação dos modelos (classes) que representarão as entidades. Para cada modelo:
    - definição de qual classe base a classe irá derivar;
    - definição dos métodos necessários, além dos já fornecidos pela classe base;
    - definição do método `run`, representando o comportamento das entidades do modelo;
  - armazenamento dos modelos criados;
  - criação das entidades (instanciação dos objetos);
  - definição das percepções.

Um exemplo de arquivo Python pode ser visualizado na Figura C.1.

A saída de dados pode ser de dois tipos:

- **posicionamento das entidades:** a imagem com o posicionamento das entidades da simulação em um determinado instante de tempo ou uma seqüência de imagens;
- **valor de atributos:** o valor de um ou mais atributos, em um arquivo texto. Este arquivo texto pode ser analisado para a criação de diversas estatísticas sobre a simulação em questão.

7.3 Aparato Tecnológico Utilizado

Para o desenvolvimento deste protótipo foram utilizadas algumas tecnologias, dentre as quais destacam-se:

- linguagem de programação Python (PSF, 2006);
- sistema gerenciador de banco de dados PostgreSQL (POSTGRESQL GLOBAL DEVELOPMENT GROUP, 2007);
- extensão PostGIS (REFRACTIONS RESEARCH, 2007);
- biblioteca GEOS (REFRACTIONS RESEARCH; VIVID SOLUTIONS; UNIVERSITY OF VICTORIA, 2007);
- módulo pyPgSQL (ALLIE; HÄRING, 2007).

Uma breve descrição de cada tecnologia é apresentada abaixo e suas relações são ilustradas na Figura 7.4.

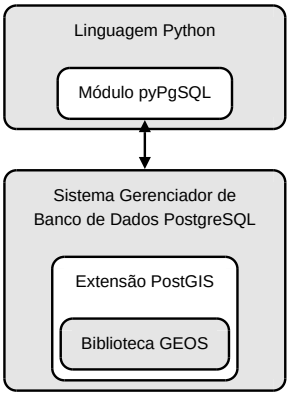

Figura 7.4: Relações entre as tecnologias utilizadas. A linguagem Python, fazendo uso das funcionalidades do módulo pyPgSQL, acessa o SGBD PostgreSQL com a extensão PostGIS e a biblioteca GEOS.

7.3.1 Linguagem Python

A linguagem de programação Python foi desenvolvida pelo holandês Guido Van Rossun, no início da década de 90, para o ensino de programação. Considerada uma linguagem de altíssimo nível (*VHLL*), Python é interpretada, orientada a objetos, possui tipagem dinâmica, é byte-compilada, multiplataforma, portável e possui uma sintaxe simples, intuitiva e muito próxima de um pseudo-código.

O interpretador da linguagem Python é facilmente extensível, incorporando novas funções e tipos de dados implementados em C ou C++ (ou qualquer outra linguagem acessível a partir de C).

7.3.2 SGBD PostgreSQL

O PostgreSQL é um SGBD (Sistema Gerenciador de Banco de Dados) objeto-relacional de código aberto, robusto, confiável, além de ser flexível e rico em recursos. É considerado objeto-relacional por implementar, além das características de um SGBD relacional, algumas características de orientação a objetos, como herança e tipos personalizados.

7.3.3 OGC e SFS

Criado em 1994, o OGC (*Open Geospatial Consortium*) (OGC, 2007a) era denominado, inicialmente, de Open GIS. O OGC possui a missão de desenvolver especificações para interfaces espaciais disponibilizadas livremente para uso geral.

A SFS (*Simple Features Specification*) (OGC, 2007b) define um formato, de acordo com o SQL padrão, para o armazenamento, leitura, análise (topológicas, espaciais, etc) e atualização de "feições simples" (dados geográficos).

7.3.4 Extensão PostGIS

O PostgreSQL foi o primeiro SGBD de código aberto a trabalhar com uma extensão específica para o tratamento dos dados geográficos vetoriais. Esta extensão, denominada PostGIS, segue a especificação SFS do OGC.

Esta extensão tem como objetivo principal permitir o armazenamento e tratamento de dados geográficos no PostgreSQL. Para que o PostGIS contemple toda a SFS, é necessário que ele seja compilado com a biblioteca GEOS. Com isso, o PostGIS passa a possuir mais de 130 funções e operadores para o tratamento de dados geográficos vetoriais.

7.3.5 Biblioteca GEOS

A biblioteca GEOS (*Geometry Engine, Open Source*) é uma versão C++ da biblioteca JTS (*JTS Topology Suite*) (VIVID SOLUTIONS, 2007), um poderoso conjunto de funcionalidades, desenvolvido em Java, para análises espaciais sobre geometrias 2D, que contempla inúmeros operadores topológicos e segue a especificação SFS do OGC. A biblioteca GEOS surgiu para atender uma demanda existente no código do PostGIS, pois este não contempla a especificação SFS em 100%. O desenvolvimento da GEOS viabilizou a total compatibilidade do PostGIS com a SFS, pois agora é possível compilar o PostGIS incluindo o código da GEOS.

7.3.6 Módulo pyPgSQL

Em Python, coleções de código-fonte que fornecem extensões para a linguagem são chamados de módulos (pode-se fazer uma associação com as bibliotecas usadas na linguagem C). Dentre os diversos módulos Python utilizados no desenvolvimento deste protótipo, pode-se destacar o pyPgSQL, um módulo que permite a fácil comunicação entre a linguagem Python e o SGBD PostgreSQL.

7.4 Considerações do Capítulo

Sistemas baseado em agentes possuem como principal forma de abstração o agente, no entanto não necessitam ser implementados utilizando estruturas de software que correspondam a agentes. Vários sistemas orientados a agentes são implementados utilizando os conceitos de orientação a objetos. Neste contexto, a linguagem Python foi escolhida por permitir o desenvolvimento baseado em objetos e por sua sintaxe ser simples e elegante, facilitando muito a criação das simulações, já que o arquivo de descrição das simulações, fornecido pelo usuário, é escrito fazendo uso da sintaxe da linguagem Python.

A extensão PostGIS (e conseqüentemente o SGBD PostgreSQL) foi utilizada neste projeto principalmente por seguir a especificação SFS do OGC. Outro motivo é a facilidade de comunicação entre a linguagem Python e o PostgreSQL.

O ZMAS foi utilizado como base para o submódulo "Comunicação" por seguir um padrão atualmente muito aceito na área de SMAs, o padrão FIPA.

O protótipo apresentado possibilitará a melhor avaliação das características e funcionalidades do modelo proposto. Tal avaliação será melhor realizada com a criação e execução de alguns estudos de caso.

8 ESTUDOS DE CASO

Neste capítulo são apresentados 4 estudos de caso criados e executados no protótipo desenvolvido. Inicialmente, são discutidas as dificuldades existentes na avaliação da arquitetura proposta.

8.1 Introdução

Sendo a proposta deste trabalho o desenvolvimento de uma arquitetura multiagentes para criação de simulações na área de Geoprocessamento, existe certa dificuldade na avaliação dos resultados da mesma. Neste contexto, é possível observar que em diversas áreas da computação existem problemas que são considerados *benchmarks* [5] e utilizados como métricas para avaliar e mensurar a qualidade das soluções propostas. Na revisão bibliográfica realizada, não foram encontrados trabalhos que fizessem uso de *benchmarks* para validar, ou mesmo avaliar, suas propostas.

Geralmente, o método de avaliação mais utilizado é a criação de diferentes simulações na plataforma que está sendo proposta, visando detectar eventuais falhas e necessidades. Neste trabalho, optou-se também por este método de avaliação. Desta forma, foram realizadas simulações de diferentes estudos de caso, objetivando principalmente apresentar e justificar as funcionalidades desenvolvidas.

Alguns estudos de caso foram desenvolvidos tendo como base simulações existentes na literatura científica. Outros, foram criados sem esta base, visto a dificuldade em encontrar detalhes técnicos sobre os modelos de simulação criados. Sendo assim, nestas simulações não se busca a realização de comparações de resultados, apenas a apresentação de como os modelos foram criados utilizando as funcionalidades da arquitetura proposta.

8.2 Estudo de Caso 1

O modelo "*Peripherisation*" (BARROS; SOBREIRA, 2002; BARROS, 2003) simula a forma específica de crescimento das cidades do terceiro mundo, mais especificamente da América Latina. Visa representar o processo de crescimento e sucessão do cenário habitacional de diferentes classes econômicas, além da formação de periferias, na tentativa de reproduzir os padrões das cidades da América Latina. São representadas três diferentes classes econômicas, divididas de acordo com a pirâmide de distribuição de renda dos países latino-americanos:

- a classe alta é a minoria (está no topo da pirâmide);

[5] O processo de comparação do desempenho entre dois ou mais sistemas é chamado de *benchmarking*, e as cargas (os dados) utilizadas são chamadas de *benchmarks*.

- a classe média (que está na parte central da pirâmide);
- a classe baixa é a maioria (está na base da pirâmide).

Todas as pessoas, independente de classe econômica, possuem a mesma preferência habitacional: locais com infra-estrutura, perto de áreas comerciais, com oportunidades de emprego, etc. Normalmente, no terceiro mundo, estes atributos são encontrados próximos das áreas onde habitam as pessoas da classe alta.

As diferenças existentes entre as classes econômicas são impostas justamente pelo poder econômico. Desta forma, a classe alta pode habitar em qualquer lugar que deseje. A classe média pode habitar em qualquer lugar, menos em locais já habitados pela classe alta. A classe baixa só pode habitar em locais onde ninguém esteja situado (locais livres).

O modelo "*City of Slums*" (BARROS; SOBREIRA, 2002) foi desenvolvido baseado na lógica do modelo "*Peripherisation*" combinada com algumas regras de consolidação. Estas regras referem-se ao processo no qual os assentamentos de baixa renda são gradualmente atualizados, até que, com o passar do tempo, se consolidam e transformam-se em favelas. Após a consolidação, estas áreas não se modificam mais.

É importante notar que os padrões espaciais de crescimento urbano possuem uma grande relação com características hidrológicas e topográficas das regiões analisadas. A existência de rios, lagos, montanhas, etc, também é um fator que influencia no resultado real. No entanto, como os dois modelos apresentados ainda estavam em um estágio inicial, estes aspectos não foram considerados. Também é afirmado por Barros e Sobreira (2002) que, fazendo uso de ferramentas de simulação, integradas com os SIGs, e dados espaciais de alta qualidade é possível alcançar uma base apropriada para a realização de pesquisas sobre os processos urbanos na América Latina.

Este primeiro estudo de caso visa reproduzir, utilizando as informações disponíveis, as simulações realizadas utilizando os modelos "*Peripherisation*" e "*City of Slums*".

8.2.1 Resultados Originais

Alguns dos resultados do modelo "*Peripherisation*", apresentados em (BARROS, 2003), estão nas Figuras 8.1 e 8.2. A área ocupada pela classe alta está representada pela cor cinza forte, a área ocupada pela classe média pela cor cinza fraco e a área ocupada pela classe baixa pela cor preta. Na Figura 8.1 são apresentados resultados para diferentes configurações da pirâmide de distribuição de renda dos países latino-americanos.

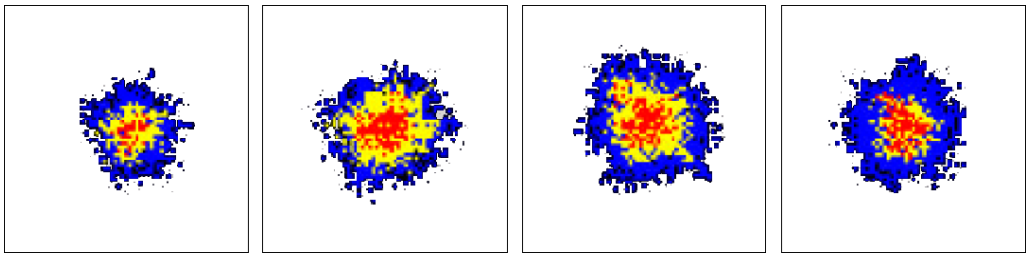

Figura 8.1: Resultados das simulações do modelo "*Peripherisation*" para diferentes valores de porcentagem relativos a quantidade de pessoas de cada classe econômica (alta, média, baixa): (5%, 30%, 65%), (10%, 40%, 50%), (10%, 30%, 60%), (10%, 20%, 70%). Adaptada de (BARROS, 2003).

Na Figura 8.2 são ilustradas as evoluções de duas simulações com diferentes configurações espaciais iniciais.

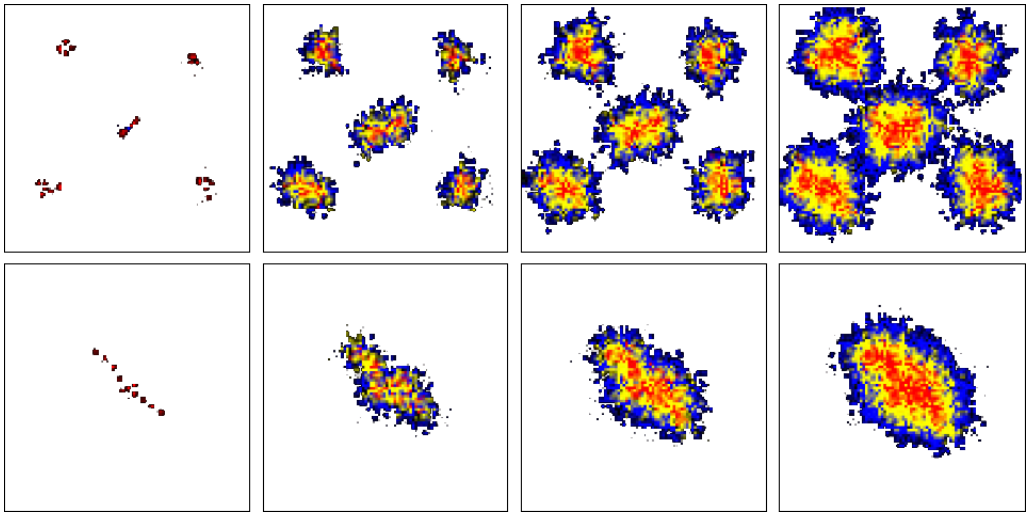

Figura 8.2: Resultados das simulações do modelo "*Peripherisation*" para diferentes configurações espaciais iniciais. Cada linha representa a evolução de uma simulação. Adaptada de (BARROS, 2003).

É possível observar que, apesar das variações nas porcentagens de distribuição de renda e das diferentes configurações espaciais iniciais, a estrutura gerada é sempre semelhante, no sentido de manter a relação entre a ocupação da área central e da área periférica.

Os resultados do modelo "*City of Slums*" são apresentados nas Figuras 8.3 e 8.4. Na Figura 8.3 são destacados os resultados para diferentes valores de tempo de consolidação das favelas. Na Figura 8.4 são ilustrados os estados finais de duas simulações (colunas 1 e 3) e os respectivos detalhes das periferias resultantes (colunas 2 e 4).

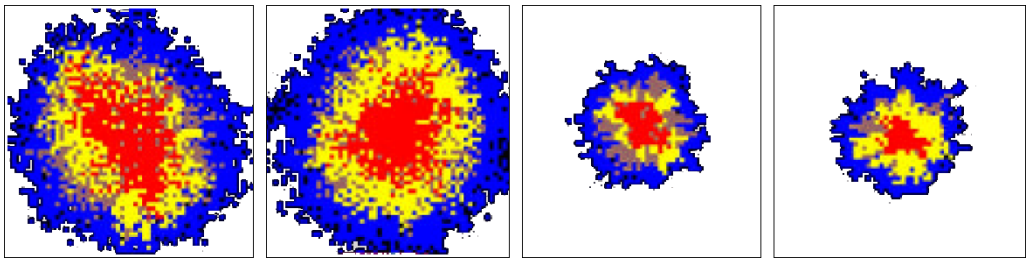

Figura 8.3: Resultados finais de simulações do modelo "*City of Slums*" para diferentes valores de tempo de consolidação. Adaptada de (BARROS, 2003).

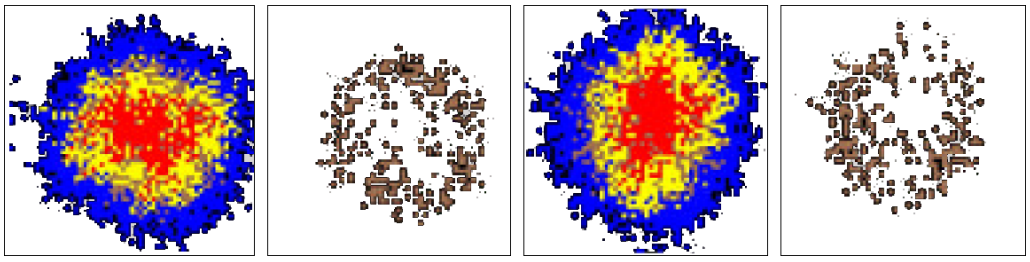

Figura 8.4: Resultados finais de simulações do modelo "*City of Slums*" (colunas 1 e 3), destacando as favelas resultantes (colunas 2 e 4). Adaptada de (BARROS, 2003).

8.2.2 Modelagem Realizada

Para apresentar a modelagem deste estudo de caso, é necessário descrever as entidades modeladas: a forma, o comportamento e como é determinado o posicionamento inicial de cada uma. Foram modeladas 6 entidades, as quais são descritas abaixo.

Os **terrenos** representam os locais onde os imóveis podem ser contruídos. São modelados como agentes fixos, possuem a forma de polígono e a posição inicial de cada terreno é definida pelos dados do BDG externo. Estes agentes fixos não possuem comportamento.

Os **lagos** e **rios** são modelados como agentes fixos, possuem a forma de polígono e a posição inicial de cada um é determinada pelos dados do BDG externo. Estes agentes fixos não possuem comportamento.

A **entrada** é modelada como um agente fixo, possui a forma de um ponto e representa a posição inicial dos agentes (que representam as pessoas de classe alta, média e baixa), quando criados dinamicamente nas simulações. Este agente fixo cria aleatoriamente, a cada passo de simulação, agentes móveis (que representam pessoas das classes alta, média e baixa), seguindo probabilidades aceitáveis, de acordo com a pirâmide de distribuição de renda dos países latino-americanos.

As **pessoas da classe alta** (maior poder aquisitivo), as **pessoas da classe média** (poder aquisitivo médio) e as **pessoas da classe baixa** (baixo poder aquisitivo) são modeladas como agentes móveis e possuem a forma de um ponto. Estes agentes móveis são criados dinamicamente, iniciam em uma posição definida pelo "agente fixo entrada" e se locomovem pelo ambiente em busca de seus objetivos, os quais foram definidos na Seção 8.2.

Também foi utilizada uma entidade sem forma para coletar dados sobre as simulações realizadas. Esta entidade tem como objetivo coletar, a cada passo de simulação, a quantidade de agentes existentes, bem como alguns de seus estados.

8.2.3 Resultados Obtidos 1

Com o objetivo de facilitar o entendimento dos resultados das simulações deste estudo de caso, na Figura 8.5 são apresentados os mapas relativos aos dados utilizados, provenientes do BDG externo.

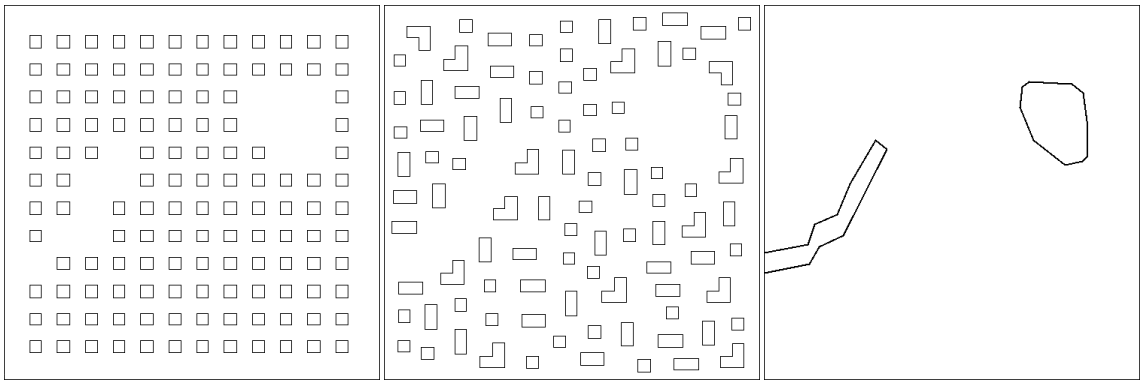

Figura 8.5: Do lado esquerdo, o mapa com a localização dos terrenos utilizados na primeira série de simulações. Ao centro, outro mapa com a localização dos terrenos utilizados na segunda série de simulações (apresentada na próxima seção). Do lado direito, o mapa das características hidrológicas da área (com o posicionamento de lagos e rios), utilizados nas duas séries de simulações.

Cabe salientar que, nas duas séries de simulações deste estudo de caso, as pessoas

de classe alta são representadas por pontos escuros (•), as pessoas de classe média são representadas por pontos claros (∘) e as pessoas de classe baixa são representadas por cruzes (+). Quando os terenos estiverem ocupados terão suas cores modificadas de acordo com a classe da pessoa que o estiver ocupando: a cor branca representa pessoas de classe alta, a cor cinza representa pessoas de classe média e a cor preta representa pessoas de classe baixa. Quando um terreno for considerado área de favela, terá a cor branca com uma borda dupla (‖).

Diversas simulações foram realizadas nesta arquitetura fazendo uso dos modelos apresentados anteriormente ("*Peripherisation*" e "*City of Slums*"). No entanto, os resultados apresentados a seguir são relativos ao segundo modelo, pois, mesmo que os dois sejam muito semelhantes, o segundo é mais complexo. Sendo assim, a evolução de uma destas simulações é apresentada nas Figuras 8.6, 8.7, 8.8, 8.9 e 8.10

8.2.4 Resultados Obtidos 2

Uma segunda série de simulações foi realizada para este mesmo estudo de caso. Nesta série foram utilizados diferentes dados para representar os terrenos onde os imóveis podem ser construídos. Nestes novos dados, é observado que os terrenos possuem diferentes formatos e que o espaçamento entre os mesmos é maior, possibilitando assim uma melhor movimentação por parte dos agentes móveis. A evolução de uma destas simulações é apresentada nas Figuras 8.11, 8.12, 8.13, 8.14 e 8.15.

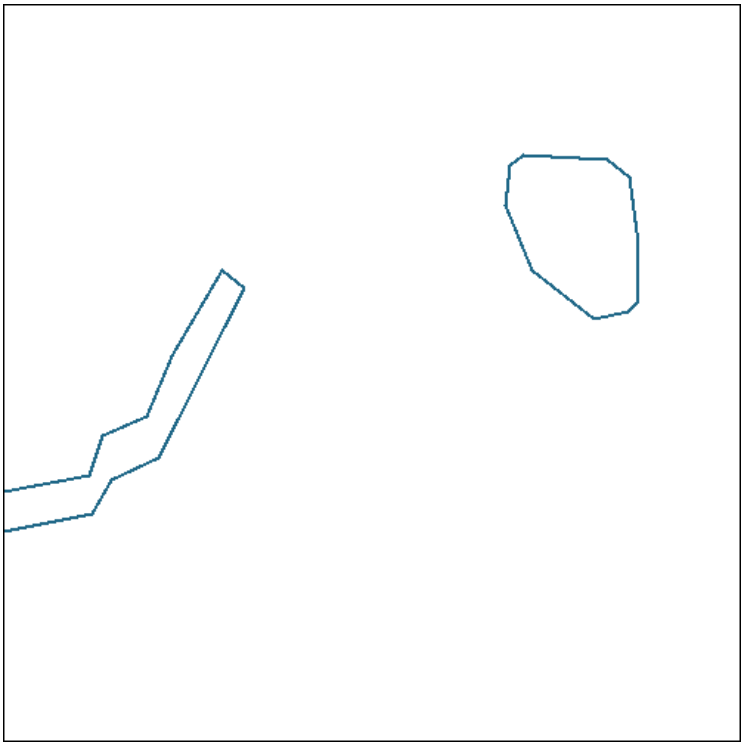

Figura 8.6: Representação espacial da simulação em seu estado inicial ($tempo = 0$). Os agentes móveis (que representam as pessoas de classe alta, média e baixa) ainda não foram criados.

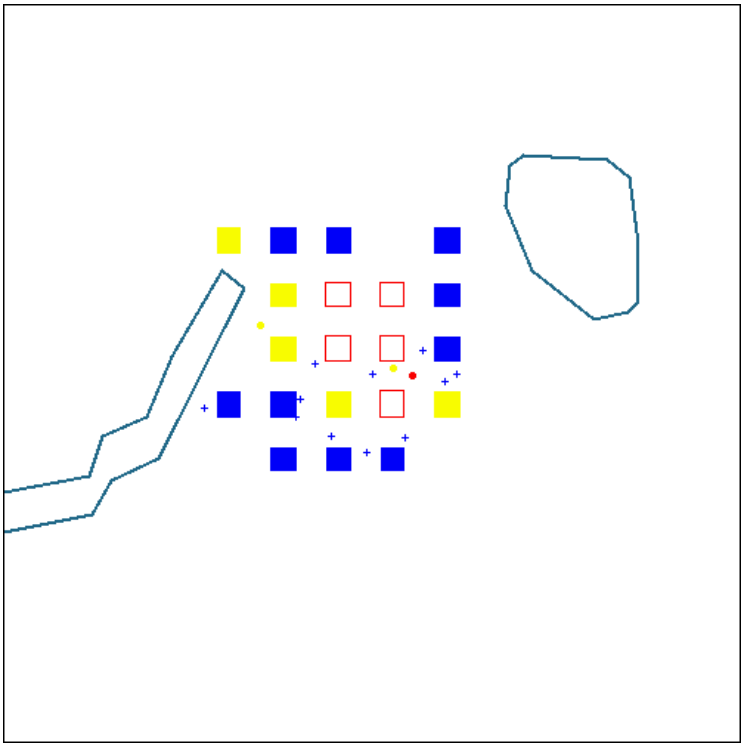

Figura 8.7: Representação espacial após alguns passos de simulação ($tempo = 35$). Vários agentes móveis (que representam pessoas) já foram criados e se alocaram nos terrenos de acordo com seus objetivos e restrições.

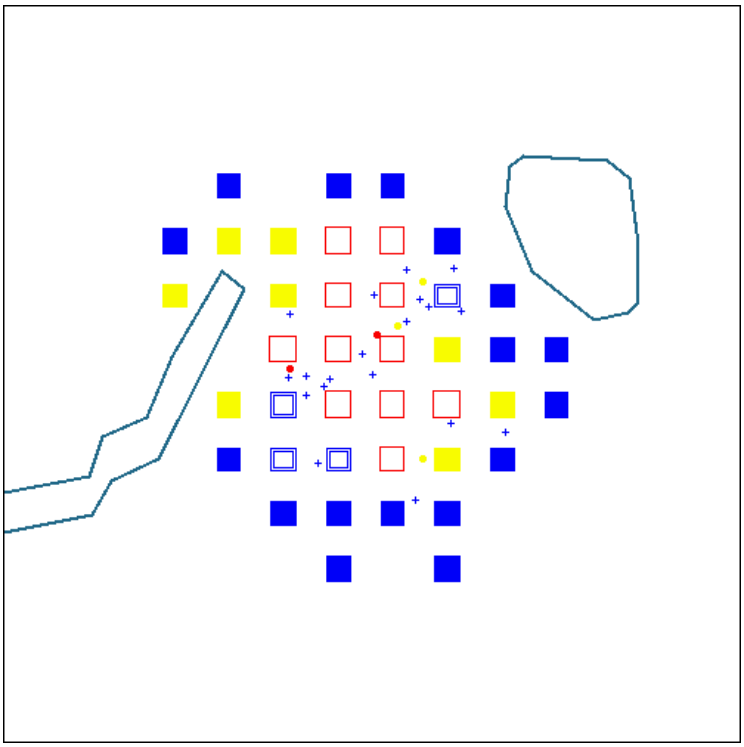

Figura 8.8: Representação espacial após mais alguns passos de simulação ($tempo = 65$). Iniciou-se o processo de estabilização das áreas de baixa renda (criação de favelas). É possível observar favelas próximas as áreas ocupadas pela classe alta.

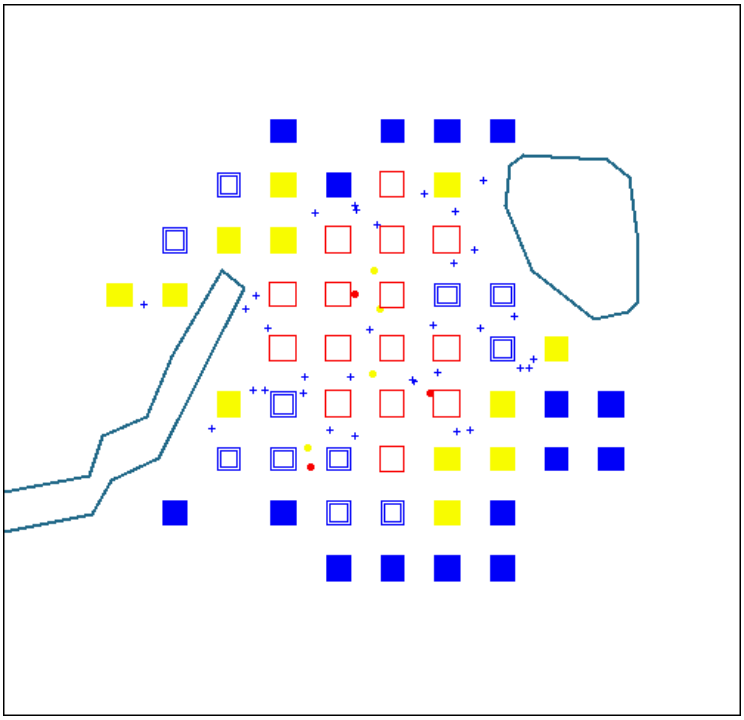

Figura 8.9: Representação espacial após a simulação evoluir um pouco mais ($tempo = 95$). O processo de alocação e estabilização continua ocorrendo. Uma área maior já está ocupada.

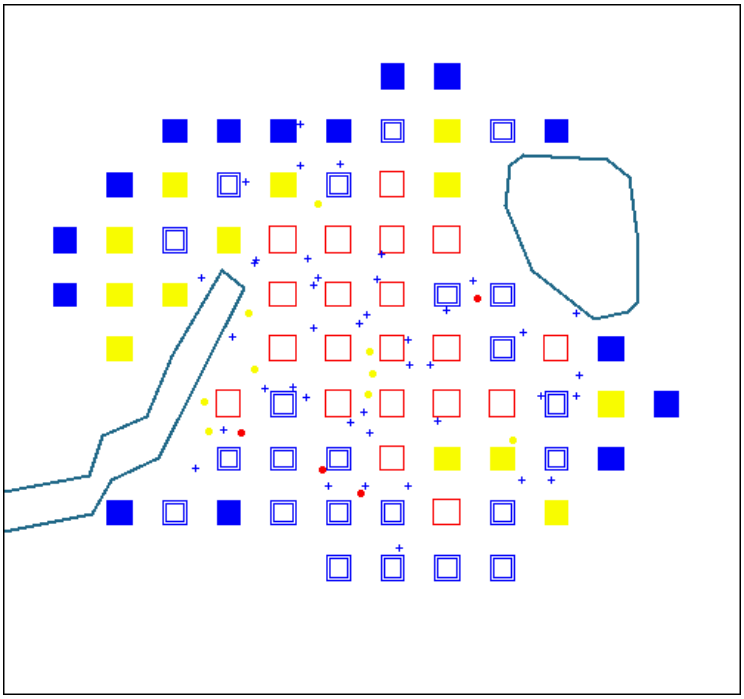

Figura 8.10: Por fim, representação espacial depois de algum tempo de simulação ($tempo = 126$). É possível observar que, conforme os resultados originais, uma área central é ocupada pela classe alta, a área ocupada pela classe baixa se localiza na periferia da área ocupada pela classe alta. A área ocupada pela classe média fica entre as duas primeiras áreas.

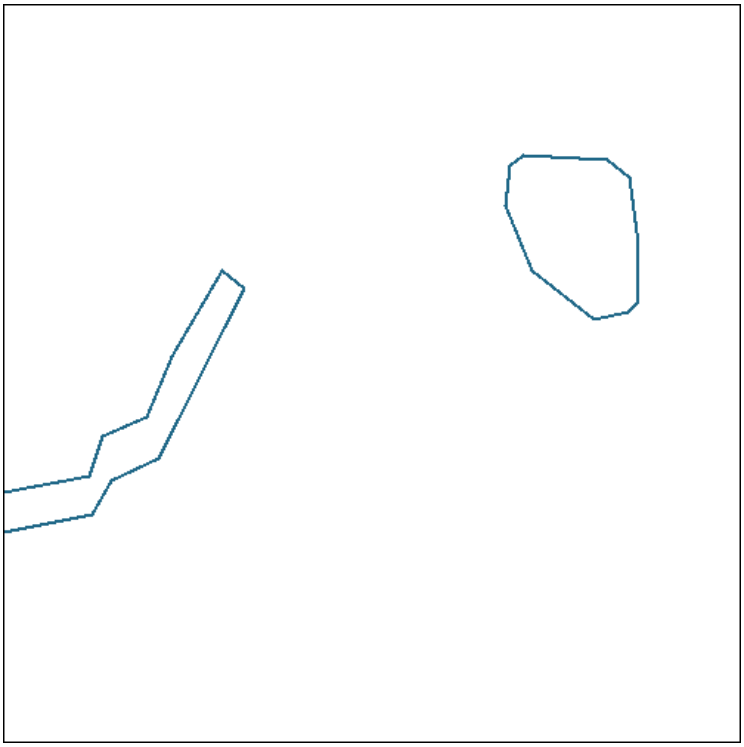

Figura 8.11: Representação espacial da simulação em seu estado inicial ($tempo = 0$). Neste momento, os agentes móveis (que representam as pessoas de classe alta, média e baixa) ainda não foram criados.

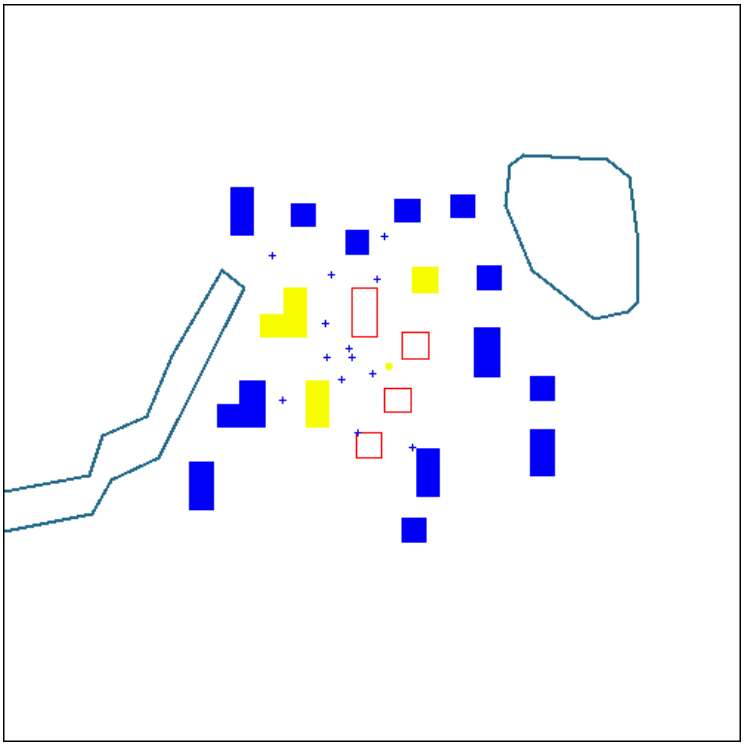

Figura 8.12: Representação espacial após alguns passos de simulação ($tempo = 35$). É possível observar que os agentes móveis (que representam as pessoas) conseguiram ocupar uma área maior, comparado-se com a simulação apresentada anteriormente.

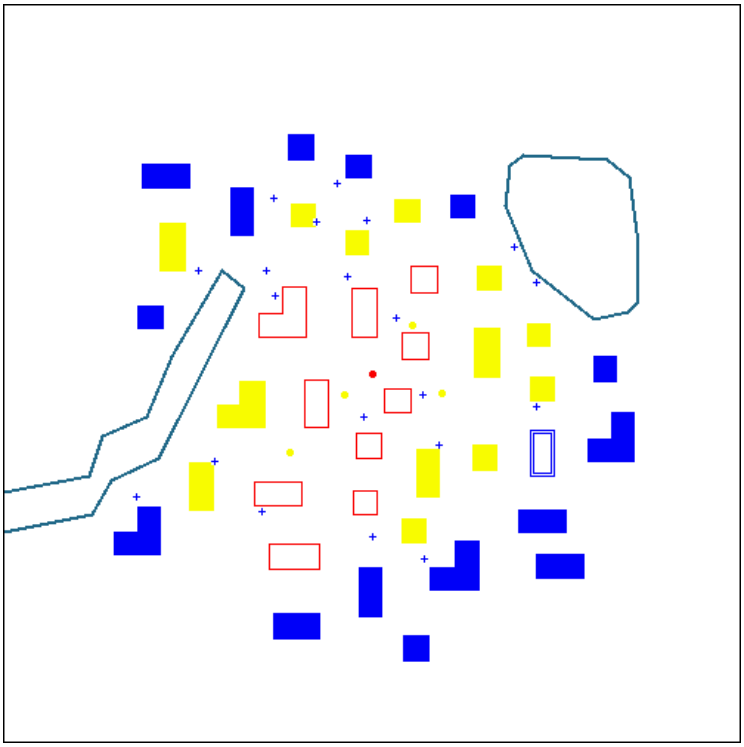

Figura 8.13: Representação espacial após mais alguns passos de simulação ($tempo = 65$). Neste momento, iniciou-se o processo de estabilização das áreas de baixa renda (criação de favelas).

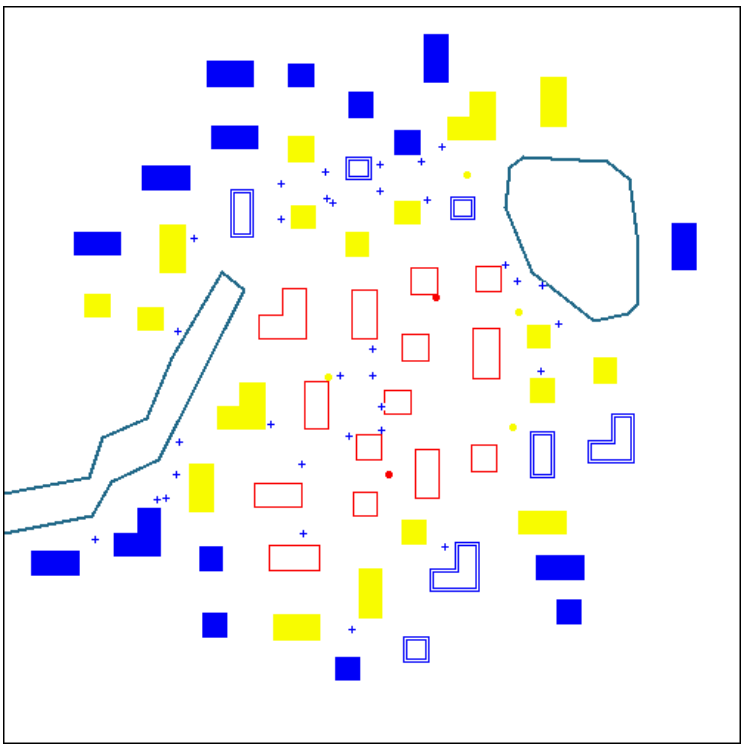

Figura 8.14: Representação espacial após a simulação evoluir um pouco mais ($tempo = 95$). O processo de alocação e estabilização continua ocorrendo. Uma maior área já está sendo ocupada.

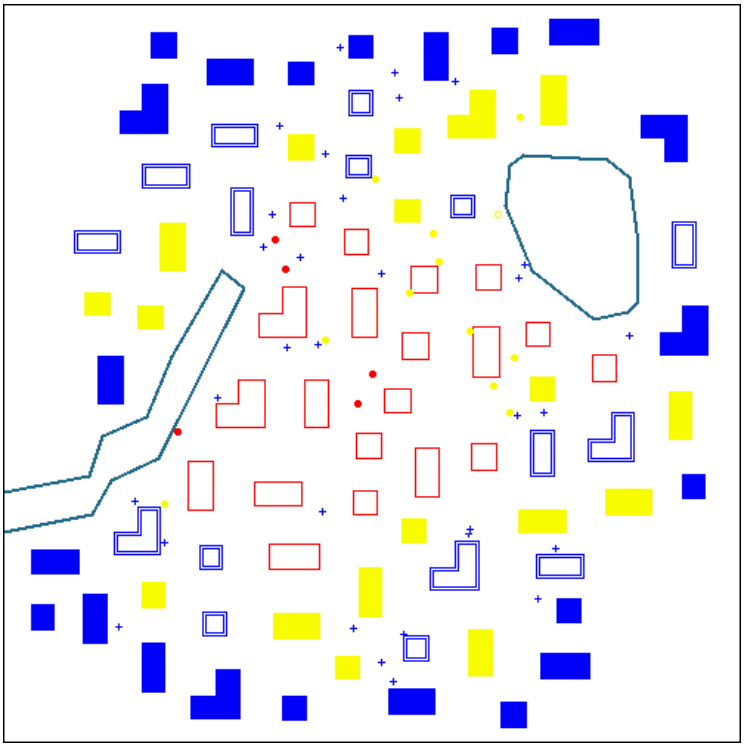

Figura 8.15: Por fim, depois de algum tempo de simulação ($tempo = 126$) é possível observar que, conforme os resultados anteriores, a área dos terrenos ocupados pela classe baixa encontra-se na periferia da área dos imóveis ocupados pela classe alta. Os imóveis ocupados pela classe média ficam entre as duas áreas anteriores. No entanto, devido ao maior espaçamento entre os terrenos é possível observar a ocupação de uma área maior.

Fazendo uso das informações coletadas pela entidade sem forma durante esta série de simulações, foi possível gerar alguns gráficos, como os apresentados na Figuras 8.16, 8.17 e 8.18.

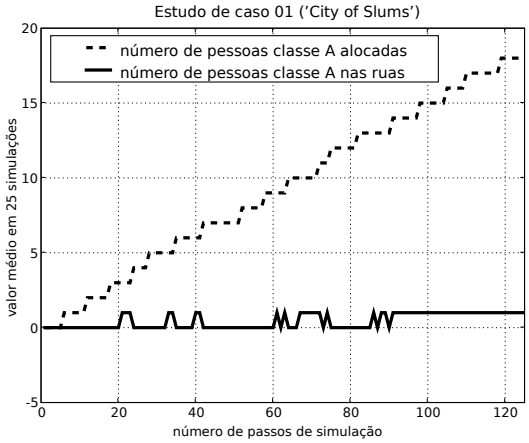

Figura 8.16: Comparação entre o número de pessoas da classe A alocadas e o número de pessoas da classe A que estão nas ruas.

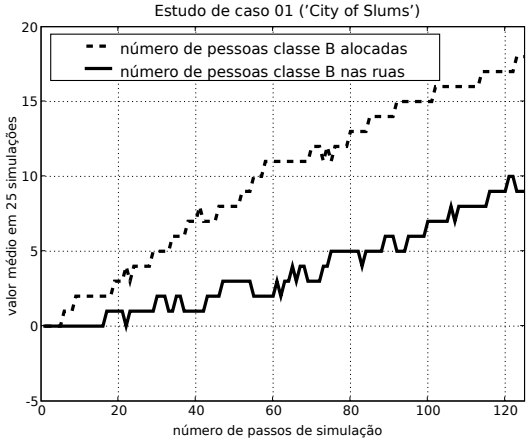

Figura 8.17: Comparação entre o número de pessoas da classe B alocadas e o número de pessoas da classe B que estão nas ruas.

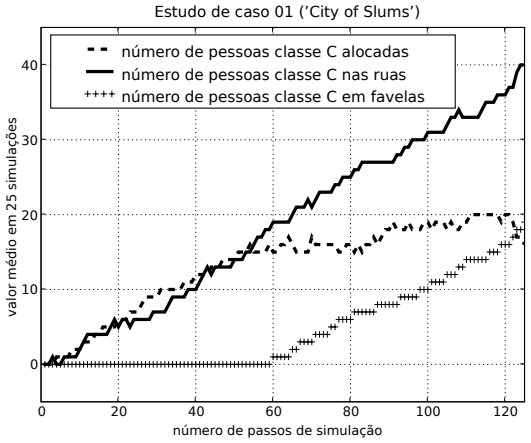

Figura 8.18: Comparação entre o número de pessoas da classe C alocadas, o número de pessoas da classe C que estão nas ruas e o número de pessoas da classe C alocadas em áreas de favela.

8.2.5 Funcionalidades Utilizadas

No desenvolvimento deste estudo de caso pode-se destacar o uso de algumas estruturas e funcionalidades da arquitetura proposta neste trabalho:

- **tipos de entidades:** foram utilizados agentes móveis do tipo ponto, agentes fixos do tipo polígono e uma entidade sem forma;
- **comportamentos de movimentação:** fez-se uso dos comportamentos de movimentação simples, de desvio de obstáculos (para os agentes móveis se movimentarem apenas entre os terrenos) e de perambular;
- **percepções:** foram definidas percepções por distância e por tipo;
- **criação de entidades:** algumas entidades foram criadas no início da simulação, outras dinamicamente, durante a simulação, utilizando modelos;

- **informações de localização:** foram utilizados dados do BDG externo e algumas entidades tiveram sua posição inicial definida por outras entidades da simulação;

- **coleta de informações:** realizada pela entidade sem forma.

8.3 Estudo de Caso 2

Este segundo estudo de caso foi criado com base em algumas idéias do modelo Expert-Cop (VASCONCELOS; FURTADO, 2005), no qual são criadas simulações sobre a alocação de policiais em regiões urbana, objetivando verificar o comportamento da criminalidade naquelas regiões. Desta forma, neste estudo de caso, são representadas as ruas e quadras de uma determinada região de uma cidade qualquer. Nesta região, existem algumas delegacias e também um presídio central. Nas ruas circulam policiais e criminosos e o objetivo dos policiais é capturar os criminosos.

8.3.1 Modelagem Realizada

Para apresentar a modelagem deste estudo de caso, é necessário descrever as entidades modeladas: a forma, o comportamento e como é determinado o posicionamento inicial de cada uma. Foram modeladas 5 entidades, as quais são descritas abaixo.

As **quadras** de uma determinada área da cidade são modeladas como agentes fixos, possuem a forma de polígono. A posição de cada agente fixo é definida pelos dados do BDG externo. Estes agentes fixos não possuem comportamento. Com o posicionamento das quadras é possível determinar a localização das ruas, locais por onde policiais e criminosos podem se locomover.

As **delegacias** são modeladas como agentes fixos e possuem a forma de polígono. A posição de cada uma é definida pelos dados do BDG externo. O comportamento das delegacias é criar, de acordo com uma certa probabilidade, os agentes móveis policiais. Quando alguma delegacia recebe uma mensagem de um agente móvel policial pedindo reforço, esta probabilidade de criação de um novo agente móvel policial é incrementada momentaneamente.

O **presídio central**, local onde os criminosos ficam presos, é modelado como um agente fixo e possui a forma de polígono. A sua posição é definida pelos dados do BDG externo. Este agente fixo não possui comportamento.

Os **criminosos** são modelados como agentes móveis e possuem o formato de ponto. Estes agentes móveis iniciam em posições aleatórias das ruas periféricas da área que está sendo representada. Estes agentes móveis se movimentam, vagando pelas ruas, até que são presos pelos policiais. Se um criminoso for capturado e enviado para o presídio central, ficará lá até o final da simulação.

Os **policiais** são modelados como agentes móveis e possuem o formato de ponto. Iniciam sempre dentro de uma das delegacias. Estes agentes móveis se movimentam pelas ruas capturando os criminosos, objetivando manter a segurança de uma certa região da cidade. Caso um agente policial perceba a existência de um agente criminoso por perto, faz a captura e o encaminha para o presídio central. Caso o agente policial perceba a existência de mais de um agente criminoso por perto, envia uma mensagem para a sua delegacia pedindo que reforços sejam enviados para as ruas.

Também foi utilizada uma entidade sem forma para coletar dados sobre as simulações realizadas. Esta entidade tem como objetivo coletar, a cada passo de simulação, a quantidade de agentes existentes.

8.3.2 Resultados Obtidos

Com o objetivo de facilitar o entendimento dos resultados, na Figura 8.19 são apresentados os mapas relativos aos dados do BDG externo, utilizados neste estudo de caso. Cabe salientar que, nas simulações, os agentes policiais são representados por pontos escuros (•) e os agentes criminosos são representados por pontos claros (○). No passo de tempo após ocorrer um pedido de reforço, os agentes policiais criados são representados por cruzes (+).

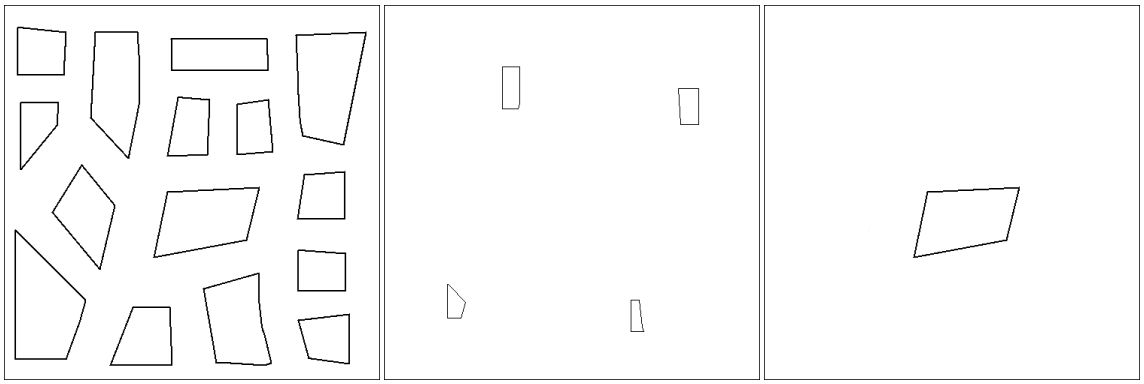

Figura 8.19: Do lado esquerdo, o mapa com a localização das ruas e quadras da área que está sendo modelada. Ao centro, o mapa com o posicionamento das delegacias. Do lado direito, o mapa com a posição do presídio central.

Várias simulações foram realizadas utilizando o modelo apresentando neste estudo de caso. A evolução de uma destas simulações é ilustrada nas Figuras 8.20, 8.21, 8.22 e 8.23.

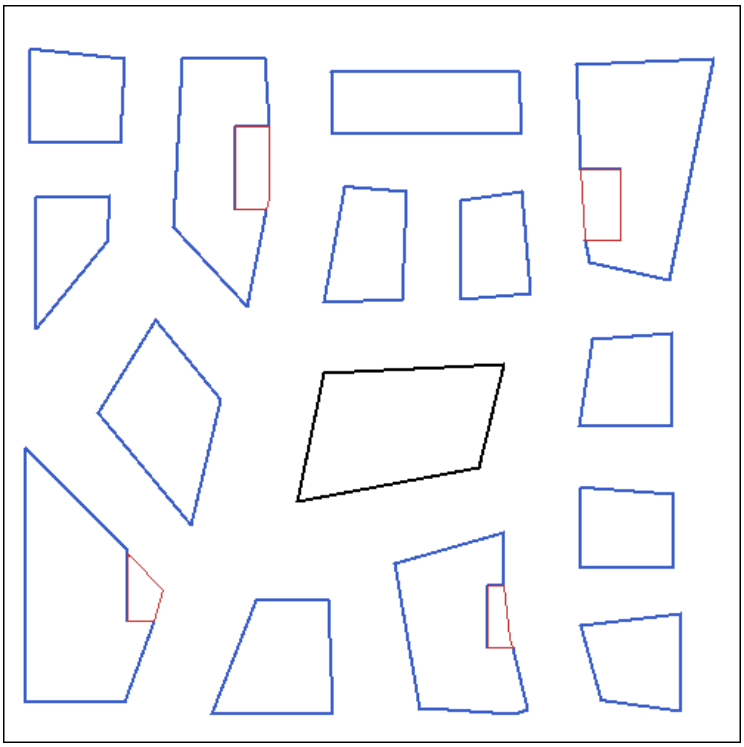

Figura 8.20: Representação espacial da simulação em seu estado inicial ($tempo = 0$). Nenhum agente policial ou criminoso foi criado.

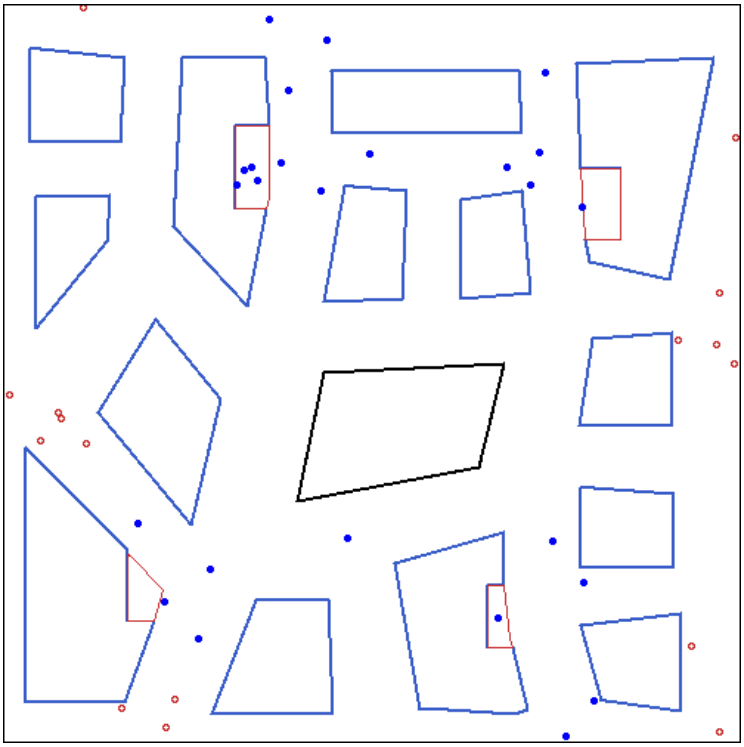

Figura 8.21: Representação espacial após alguns passos de simulação ($tempo = 18$), agentes policiais e criminosos foram criados e se movimentam pelas ruas. Os policiais estão principalmente na área central, perto das delegacias, e os criminosos na periferia.

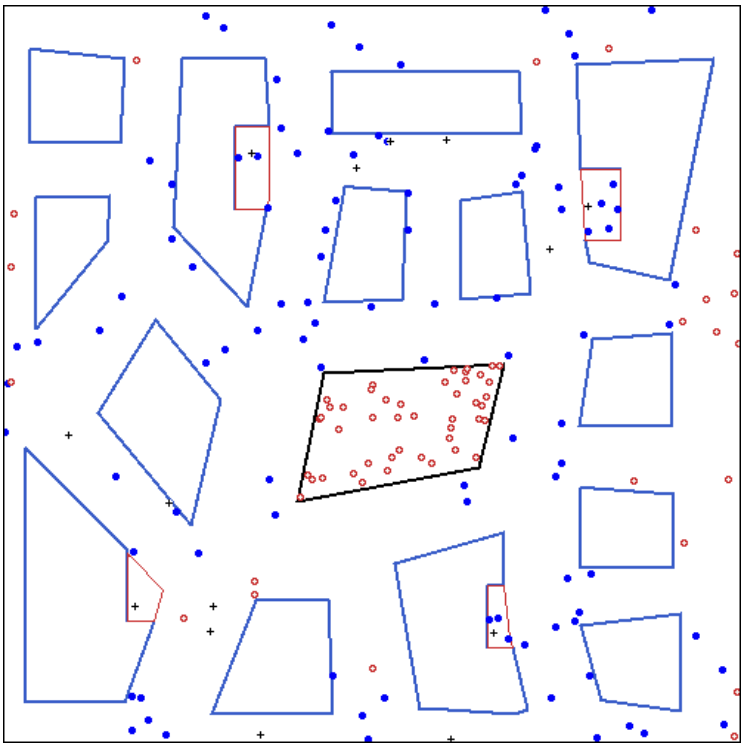

Figura 8.22: Representação espacial após mais alguns passos de simulação ($tempo = 68$). Mais agentes criminosos estão presos. Os agentes policiais estão espalhados pela área periférica e já pediram reforços, pois existem agentes representados por cruzes.

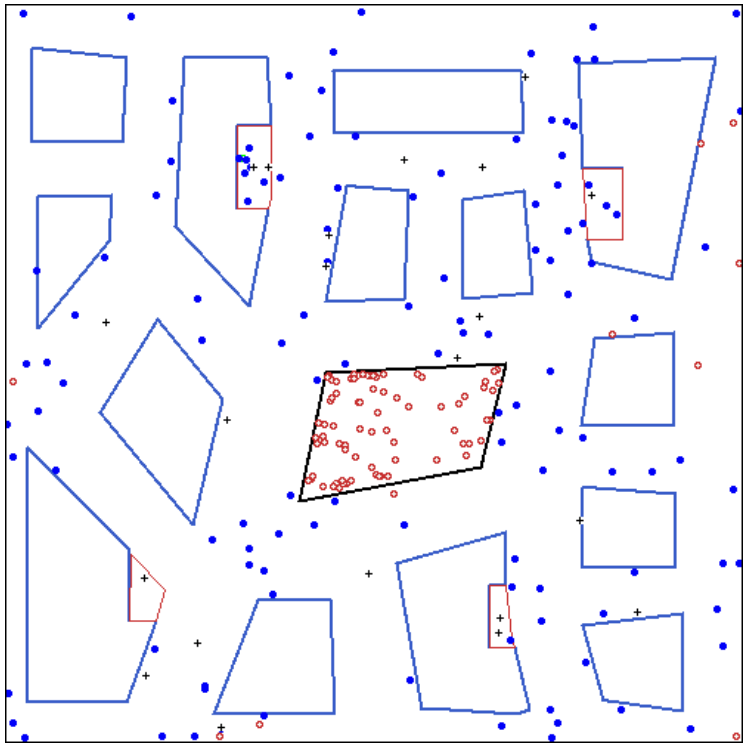

Figura 8.23: Por fim, depois de algum tempo de simulação ($tempo = 85$), é possível observar que os agentes policiais conseguiram capturar quase que totalmente os agentes criminosos.

Fazendo uso dos dados coletados pela entidade sem forma durante esta série de simulações, foi possível gerar um gráfico contendo a quantidade de cada tipo de agente, como apresentado na Figura 8.24.

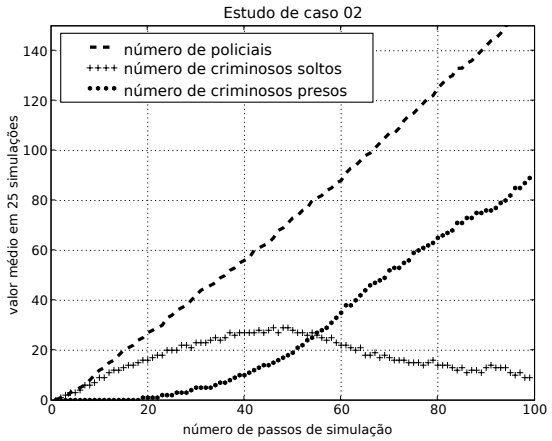

Figura 8.24: Comparação entre o número de agentes policiais e o número de agentes criminosos, soltos e presos. Este gráfico confirma que os agentes policiais capturaram quase todos os agentes criminosos.

8.3.3 Funcionalidades Utilizadas

No desenvolvimento deste estudo de caso pode-se destacar o uso de algumas estruturas e funcionalidades da arquitetura proposta:

- **tipos de entidades:** foram utilizados agentes móveis do tipo ponto, agentes fixos do tipo polígono e uma entidades sem forma;
- **comportamentos de movimentação:** fez-se uso dos comportamentos de movimentação simples, de desvio de obstáculos (para os agentes móveis se movimentarem apenas pelas ruas e não dentro das quadras) e de perambular;
- **percepções:** foram definidas percepções por distância e por tipo;
- **criação de entidades:** algumas entidades foram criadas no início da simulação, outras dinamicamente, durante a simulação, utilizando modelos;
- **informações de localização:** foram utilizados dados do BDG externo e algumas entidades tiveram sua posição inicial definida por outras entidades da simulação;
- **comunicação:** a comunicação entre os agentes policiais e as delegacias foi realizado de forma direta, por troca de mensagens;
- **coleta de informações:** realizada pela entidade sem forma.

8.4 Estudo de Caso 3

Este terceiro estudo de caso é baseado em algumas idéias do trabalho de Itami e Gimblett (2001), o qual apresenta simulações de um parque de diversões, objetivando auxiliar na otimização de seus processos de planejamento e manutenção. Neste modelo procura-se observar o comportamento dos visitantes dentro do parque.

8.4.1 Modelagem Realizada

Para apresentar a modelagem deste estudo de caso, é necessário descrever as entidades modeladas: a forma, o comportamento e como é determinado o posicionamento inicial de cada uma. Foram modeladas 8 entidades, as quais são descritas abaixo.

As **montanhas** e os **lagos** são modelados como agentes fixos e possuem a forma de polígono. A posição de cada agente fixo é definida pelos dados do BDG externo. Estes agentes fixos não possuem comportamento.

Os **limites** são modelados como agentes fixos e possuem a forma de linha. Estes agentes fixos representam algum modo de delimitar a área do parque (cercas, muros, etc). A posição de cada agente fixo é definida pelos dados do BDG externo. Estes agentes fixos não possuem comportamento.

As **entradas** representam os locais por onde as entidades entram no parque (na simulação), são modeladas como agentes fixos e possuem a forma de polígono. A posição de cada agente fixo é definida pelos dados do BDG externo. O comportamento destes agentes fixos é criar, a cada passo da simulação, de acordo com uma certa probabilidade, agentes alpinistas, mergulhadores, caminhantes ou guias.

As **saídas** representam os locais por onde as entidades saem do parque (da simulação), são modeladas como agentes fixos e possuem a forma de polígono. A posição de cada

agente fixo é definida pelos dados do BDG externo. O comportamento deste tipo de agente fixo é excluir da simulação as entidades que entram em sua área.

Os **alpinistas**, **mergulhadores**, **caminhantes** e **guias** são modelados como agentes móveis e possuem a forma de ponto. Iniciam em uma posição determinada por algum agente entrada. Os agentes alpinistas se movimentam pelo parque, desviando dos lagos e buscando as montanhas. Os agentes mergulhadores se deslocam pelo parque, desviando das montanhas e buscando os lagos. Os agentes guias se deslocam pelo parque, desviando das montanhas e dos lagos. Os agentes caminhantes se movimentam pelo parque, desviando das montanhas e dos lagos e quando percebem que existe algum agente guia por perto, seguem a trajetória deste agente.

8.4.2 Resultados Obtidos

Com o objetivo de facilitar o entendimento dos resultados, na Figura 8.25 são apresentados os mapas relativos aos dados utilizados neste estudo de caso. Cabe salientar que, nas simulações, os agentes mergulhadores são representados por pontos escuros ($\bullet$), os agentes alpinistas por pontos claros ($\circ$), os agentes guias por cruzes ($+$) e os agentes caminhantes por xizes ($\times$).

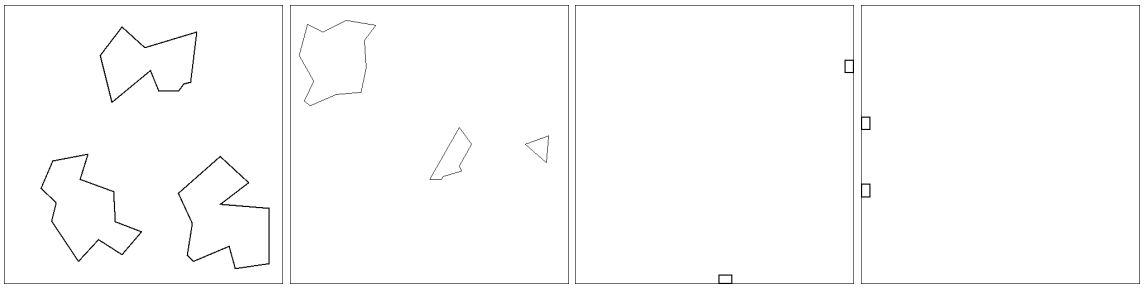

Figura 8.25: Da esquerda para a direita: a primeira imagem é o mapa com a localização das montanhas, a segunda é um mapa com a localização dos lagos, a terceira é um mapa com a localização das entradas do parque e a quarta é um mapa com a localização das saídas do parque.

Diversas simulações foram realizadas fazendo uso do modelo apresentado. A evolução de uma destas simulações é ilustrada nas Figuras 8.26, 8.27 e 8.28.

8.4.3 Funcionalidades Utilizadas

No desenvolvimento deste estudo de caso pode-se destacar o uso de algumas estruturas e funcionalidades da arquitetura proposta:

- **tipos de entidades:** foram utilizados agentes móveis do tipo ponto, agentes fixos do tipo linha e polígono;
- **comportamentos de movimentação:** fez-se uso dos comportamentos de movimentação simples, de desvio de obstáculos (os obstáculos variavam de acordo com o tipo de agente), de perambular e de perseguir;
- **percepções:** foram definidas percepções por distância e por tipo;
- **criação de entidades:** algumas entidades foram criadas no início da simulação, outras dinamicamente, durante a simulação, utilizando modelos;

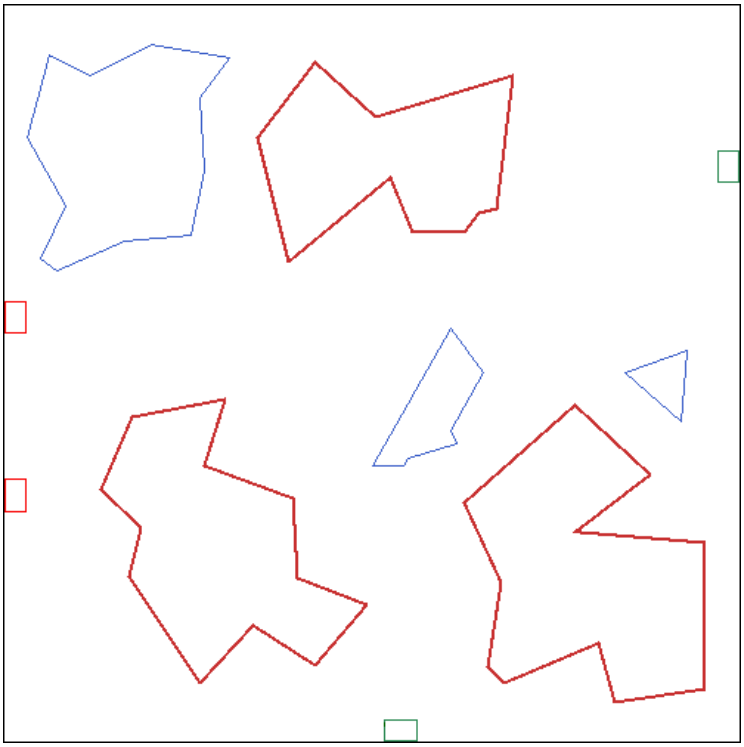

Figura 8.26: Representação espacial da simulação em seu estado inicial ($tempo = 0$). Neste momento, ainda não existem agentes móveis dentro da área do parque.

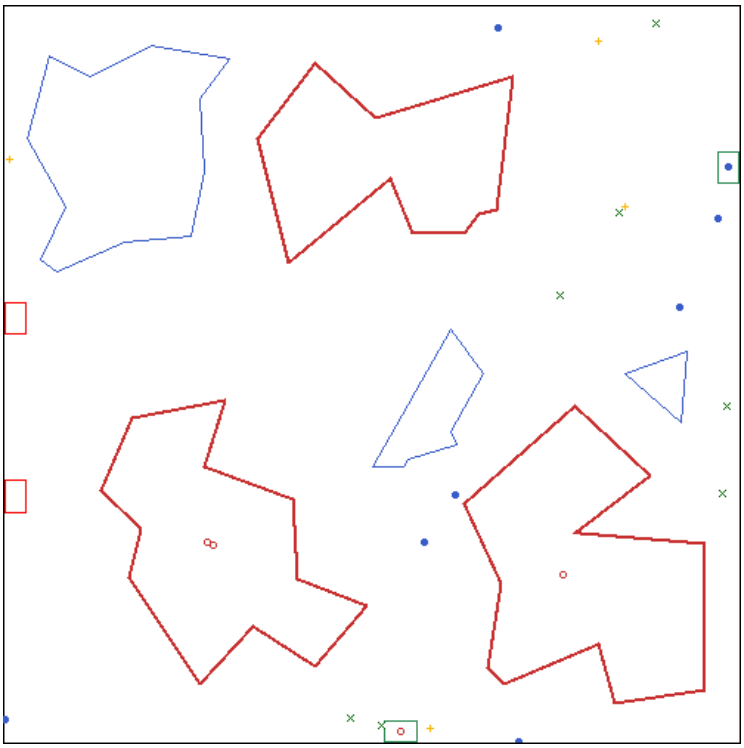

Figura 8.27: Representação espacial após alguns passos de simulação ($tempo = 44$). Agentes de todos os tipos já foram criados e se movimentam de acordo com seus objetivos e restrições.

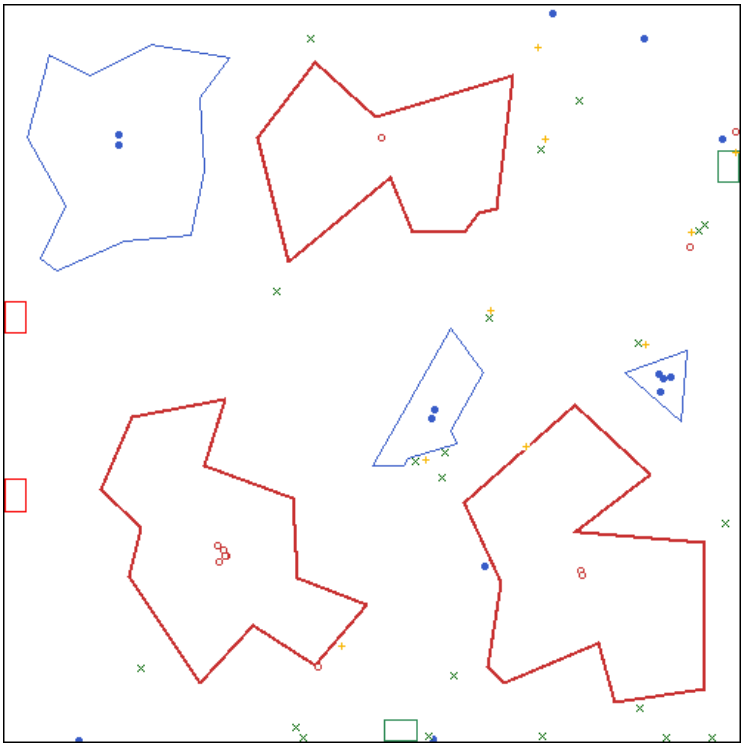

Figura 8.28: Por fim, depois de algum tempo de simulação ($tempo = 115$), é possível observar que grande parte dos agentes mergulhadores estão dentro dos lagos, vários agentes alpinistas estão nas montanhas, os agentes caminhantes estão se movimentando pelo parque, alguns acompanhando os guias. Cabe salientar ainda que, diversos agentes móveis já saíram da simulação.

- **remoção de entidades:** algumas entidades foram dinamicamente removidas da simulação pelo agente Saída;
- **informações de localização:** foram utilizados dados do BDG externo e algumas entidades tiveram sua posição inicial definida por outras entidades da simulação.

8.5 Estudo de Caso 4

A simulação do processo chuva-vazão tem sido muito utilizada por hidrólogos e engenheiros de recursos hídricos objetivando, principalmente, quantificar o volume escoado (vazão) como conseqüência de uma determinada precipitação.

Segundo Barbalho (2001), o processo chuva-vazão é, contudo, um processo hidrológico altamente não linear, variante no tempo e distribuído espacialmente. As tentativas de representá-lo através de modelos matemáticos têm produzido diversos modelos. Este processo é definido como:

> *"o volume de água precipitado dentro dos limites de uma bacia hidrográfica, antes de atingir a superfície do solo, sofre perdas por evaporação e retenção na vegetação natural (posteriormente transpirado). O volume que atinge a superfície do solo infiltra, sempre que as condições deste o permitam, caso contrário, escoa superficialmente. O volume de água que infiltra é, primeiramente, retido nas camadas superiores do solo, sofrendo aí, também, perdas por evaporação. O volume não*

> *retido escoa lateralmente, formando o escoamento subsuperficial, ou percola, por força da gravidade, até as camadas mais profundas, indo constituir o escoamento subterrâneo. As parcelas correspondentes aos escoamentos superficial, subsuperficial e subterrâneo vão compor o escoamento sobre os cursos d'água da bacia".* (BARBALHO, 2001)

Através de modelos de simulação é possível simular condições que não poderiam ser reproduzidas na natureza. Entre os problemas práticos que fazem uso deste tipo de simulação, destacam-se: ***(i)*** avaliação da vazão de cheia máxima para cálculo de sistemas de drenagem, ***(ii)*** avaliação de ondas de cheia para subsidiar sistemas de controle e prevenção de cheias, ***(iii)*** previsão de vazões para subsidiar programas de operação de reservatórios, ***(iv)*** avaliação de vazão mínima de estiagem para subsidiar sistemas de gerenciamento de recursos hídricos e ***(v)*** reconstrução de séries históricas de vazões.

Este quarto estudo de caso visa simular, de forma muito simplificada, a relação entre a ocorrência ou não de chuvas em uma determinada região e o aparecimento de fenômenos naturais como secas e inundações, ou seja, a relação chuva-vazão.

8.5.1 Modelagem Realizada

Para apresentar a modelagem deste estudo de caso, é necessário descrever as entidades modeladas: a forma, o comportamento e como é determinado o posicionamento inicial de cada uma. Foram modeladas 3 entidades, as quais são descritas abaixo.

O **terreno** é modelado como um agente móvel e possui a forma de polígono. Este agente tem sua posição inicial definida pelos dados do BDG externo. O polígono que representa o terreno possui alguns buracos (vide definição na Seção 2.4.4.1). Este agente não possui comportamento.

Os **lagos** são modelados como agentes móveis, possuem a forma de polígono e têm suas posições definidas por dados do BDG externo. O nível de água dos lagos pode variar de acordo com a quantidade de chuva na região. Esta variação causa uma mudança na área do lagos. Se as nuvens de chuva ficarem por 10 períodos seguidos na região de um lago, o nível de água aumentará. Caso contrário, se por 10 períodos seguidos não chover na região, o nível de água diminuirá.

As **nuvens de chuva** são modeladas como agentes móveis e possuem forma de polígono. A posição inicial de cada agente móvel é definida pelos dados do BDG externo. Estes agentes se movimentam (fazendo uso do comportamento "perambular") pela área representada na simulação. A posição das nuvens de chuva representa os locais onde está chovendo.

É importante salientar que o **terreno** e os **lagos** são definidos como agentes móveis adjacentes. Isto significa que a modificação na forma/posição dos lagos causa a modificação automática na forma/posição do terreno. Sendo assim, quando o nível de água de um lago aumenta (ou diminui), sua forma tende a se modificar, ou seja, a sua área fica maior (ou menor). Esta funcionalidade permite que os limites dos agentes adjacentes sejam atualizados igualmente.

8.5.2 Resultados Obtidos

Para facilitar o entendimento dos resultados, é necessário observar que o terreno é representado pela cor branca, as nuvens pela cor cinza e os lagos pela cor preta. Nestas simulações, os dois principais resultados observados são os fenômenos de seca e inundação das áreas dos lagos, decorrente da ocorrência ou não de chuvas. Diversas simulações

foram realizadas fazendo uso do modelo apresentado. A evolução de uma destas simulações é ilustrada nas Figuras 8.29, 8.30 e 8.31.

Sem a utilização da funcionalidade que trata dos agentes móveis adjacentes as simulações apresentariam resultados diferentes, visto que não haveria uma sincronização dos limites do terreno e dos lagos. Uma segunda série de simulações foi realizada, na qual apenas o lago maior foi definido como adjacente ao terreno. A evolução de uma destas simulações é ilustrada nas Figuras 8.32 e 8.33.

8.5.3 Funcionalidades Utilizadas

No desenvolvimento deste estudo de caso pode-se destacar o uso de algumas estruturas e funcionalidades da arquitetura proposta:

- **tipos de entidades:** foram utilizados agentes móveis e fixos do tipo polígono;
- **comportamentos de movimentação:** fez-se uso dos comportamentos de movimentação simples e de perambular;
- **percepções:** foram definidas percepções por distância e por tipo;
- **criação de entidades:** todas as entidades foram criadas no início da simulação;
- **informações de localização:** foram utilizados os dados do BDG externo;
- **entidades adjacentes:** foi utilizada esta funcionalidade para sincronizar os limites do terreno e dos lagos (a área dos "buracos" existentes no polígono que representa o terreno e a área dos polígonos que representam os lagos).

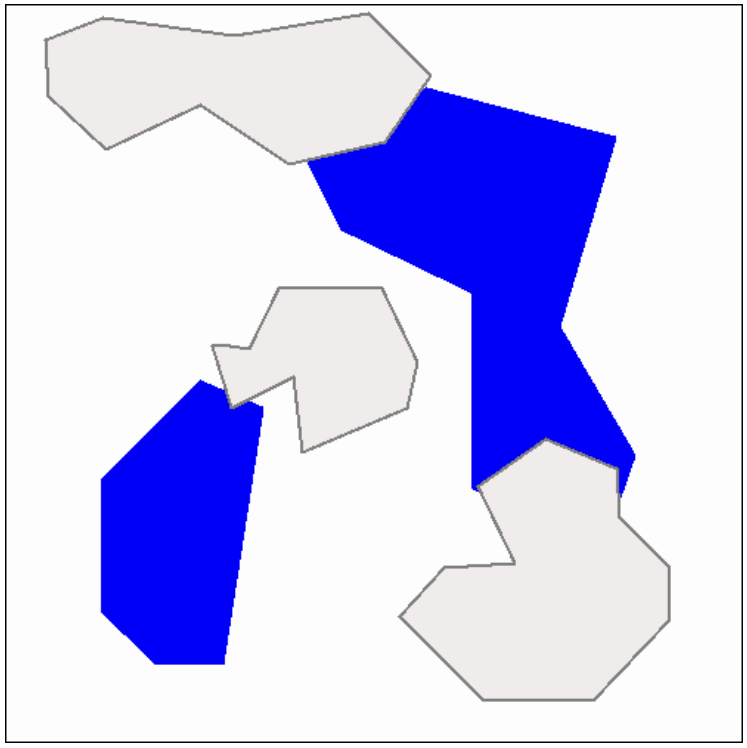

Figura 8.29: Representação espacial da simulação em seu estado inicial ($tempo = 0$). As nuvens estão em suas posições iniciais e os lagos no nível considerado normal.

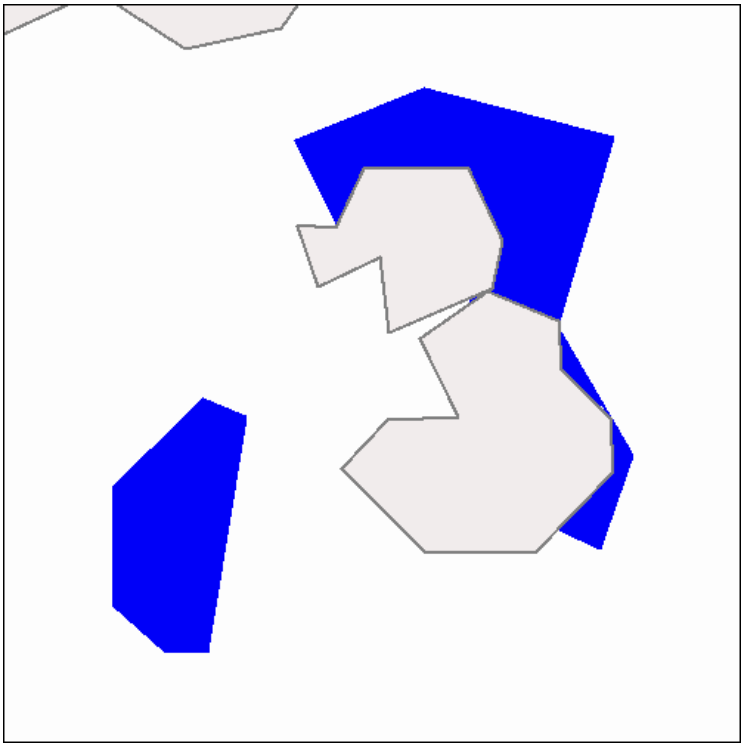

Figura 8.30: Com o passar do tempo ($tempo = 12$), as nuvens já se movimentaram, o lago maior continua no mesmo nível (mesma área) e o lago menor diminuiu (a área diminuiu) por falta de chuvas na sua região.

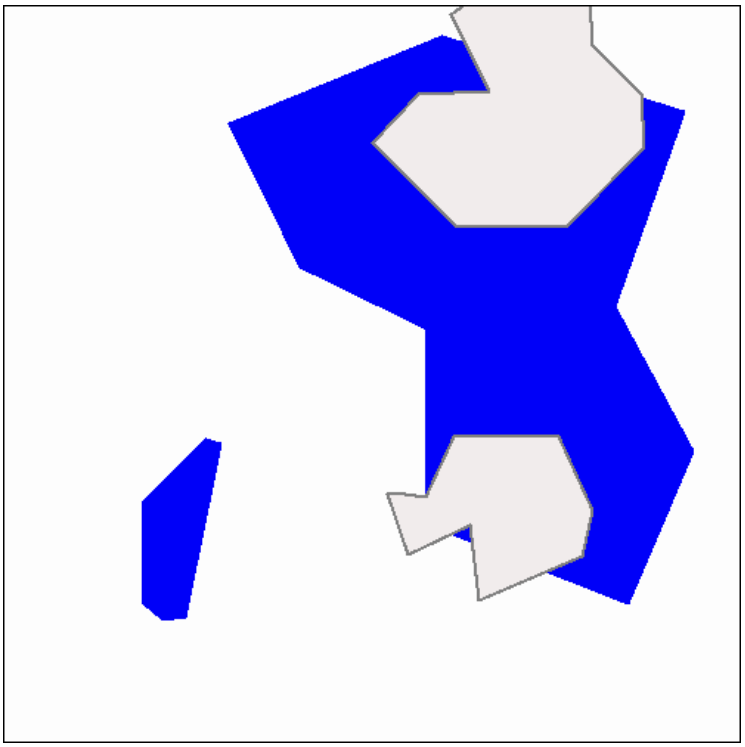

Figura 8.31: Representação espacial após mais alguns passos ($tempo = 39$), a área do lago maior aumentou devido ao grande número de ocorrências de chuva na região. A área do lago menor continua diminuindo devido a falta de chuvas.

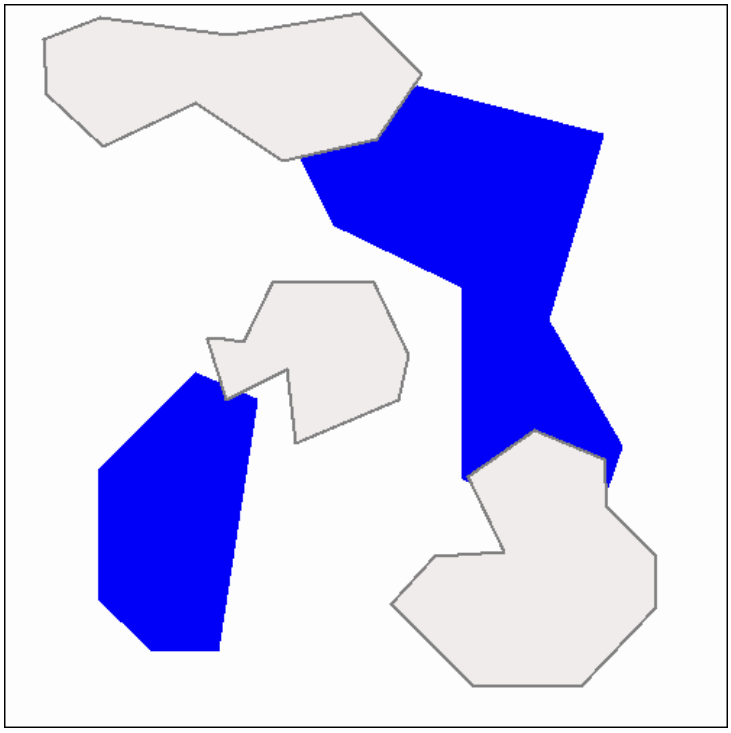

Figura 8.32: Representação espacial da simulação em seu estado inicial ($tempo = 0$). As nuvens estão em suas posições iniciais e os lagos no nível considerado normal.

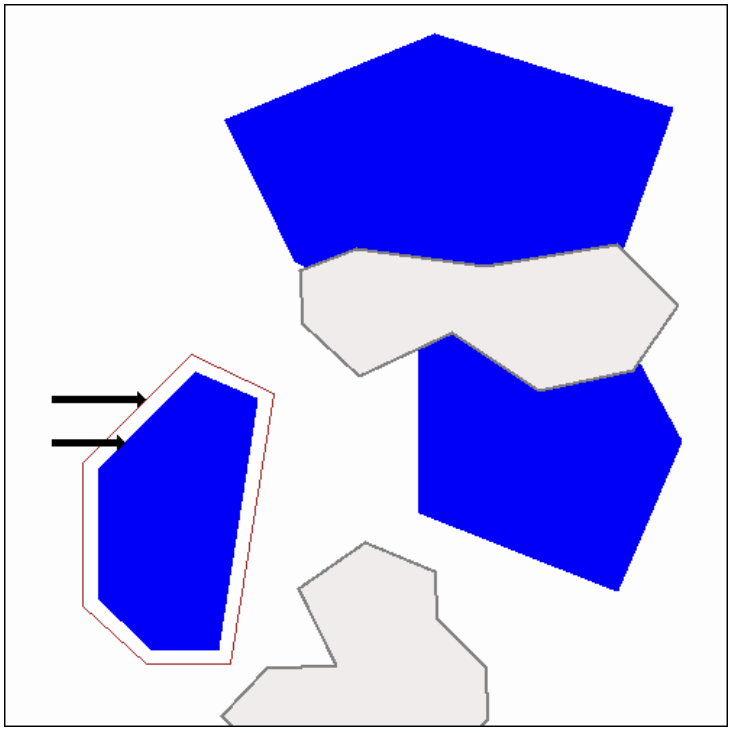

Figura 8.33: Representação espacial após algum tempo de simulação ($tempo = 39$), a área do lago maior aumentou (os limites do lago e do terreno são atualizados igualmente pois estes agentes móveis foram definidos como adjacentes) e a área do lago menor diminuiu (no entanto, os limites do terreno não foram modificados, acarretando a exibição de dois limites diferentes, um do lago e o outro do terreno, como apontado pelas setas).

8.6 Considerações do Capítulo

Este capítulo apresentou 4 estudos de caso. Para cada um destes, foram detalhadas a motivação, a forma de modelagem na arquitetura proposta, os resultados obtidos e as funcionalidades utilizadas. Cabe ressaltar que estes estudos de caso eram simples e que o objetivo principal das simulações era demonstrar que as funcionalidades desenvolvidas são úteis na modelagem de diferentes cenários.

9 CONSIDERAÇÕES FINAIS

Neste trabalho foi apresentado um modelo de arquitetura computacional, baseado em SMAs, para a criação e execução de simulações na área de Geoprocessamento. Nesta arquitetura, a forma das entidades existentes nas simulações é baseada em dados vetoriais de um BDG.

Sendo assim, a integração do modelo de agentes com dados oriundos de um BDG se faz extremamente interessante, principalmente porque os BDGs possuem ferramentas específicas para armazenar e gerenciar representações espaciais precisas. Além disto, a conexão dinâmica com um BDG permite o melhor aproveitamento dos dados geográficos, facilitando a realização de consultas espaciais complexas durante as simulações.

Permitindo que tanto os modelos geográficos quanto os modelos baseados em SMAs se beneficiem do acoplamento entre SMA-BDG, as contribuições resultantes do desenvolvimento desta arquitetura focam principalmente em 2 pontos:

- nas áreas de simulações e SMAs, mais especificamente no contexto da modelagem e desenvolvimento de SMAs, possibilitando a modelagem espacial contínua e precisa do ambiente e das entidades das simulações, de forma simples e realística, utilizando estruturas da área de Geoprocessamento, mais precisamente dados vetoriais de um BDG. A alta precisão na modelagem espacial está direntamente relacionada com a forma vetorial dos dados geográficos utilizados;

- na área de Geoprocessamento, fornecendo abstrações para a representação de eventos espaço-temporais dinâmicos, utilizando simulações baseadas em SMAs;

É importante salientar que a qualidade da representação espacial nas simulações está diretamente relacionada com a qualidade da base de dados utilizada no fornecimento dos dados geográficos.

Com o objetivo principal de verificar a viabilidade e usabilidade das funcionalidades da arquitetura proposta, foi desenvolvido um protótipo. Neste, foram criados e executados estudos de caso de diferentes cenários, possibilitando a verificação de algumas deficiências existentes na arquitetura. Algumas destas deficiências foram corrigidas durante o trabalho, outras deixadas para trabalhos futuros. Neste escopo, a plataforma desenvolvida também é uma contribuição significativa deste trabalho.

As simulações apresentadas no Capítulo 8 foram criadas com grande facilidade e executadas com sucesso no protótipo. No entanto, a dificuldade na avaliação da arquitetura decorre da falta de métricas e *benchmarks*, existente neste contexto.

No Capítulo 5, são apresentadas as características positivas e negativas de cada uma das plataformas de simulação analisadas. Com esta análise é possível observar que a arquitetura proposta agrupou diversas características positivas e conseguiu suprir algumas

falhas encontradas. Um resumo comparativo entre as funcionalidades das plataformas analisadas as do modelo proposto são apresentados na Tabela 9.1.

Tabela 9.1: Tabela comparativa entre funcionalidades das plataformas analisadas e da arquitetura proposta. **Legenda**: $\surd$: *funcionalidade implementada*, $\star$: *funcionalidade parcialmente implementada*, $\times$: *funcionalidade não implementada ou não encontrada*.

Funcionalidade	**Arquitetura proposta**	**Swarm**	**Repast**	**SeSAm**	**NetLogo**	**OBEUS**
Movimentação de alto nível	$\surd$	$\times$	$\times$	$\times$	$\times$	$\times$
Utilização de dados matriciais	$\times$	$\star$	$\surd$	$\times$	$\surd$	$\times$
Conexão dinâmica com o BDG	$\surd$	$\times$	$\times$	$\times$	$\times$	$\times$
Modelagem contínua	$\surd$	$\times$	$\times$	$\times$	$\times$	$\surd$
Facilidade de desenvolvimento	$\star$	$\times$	$\star$	$\surd$	$\star$	$\times$
Comunicação	$\surd$	$\times$	$\times$	$\surd$	$\times$	$\times$
Escalonador de tempo discreto	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
Entidades auxiliares	$\surd$	$\surd$	$\times$	$\times$	$\times$	$\times$
Entidades adjacentes	$\surd$	$\times$	$\times$	$\times$	$\surd$	$\times$

9.1 Trabalhos Futuros

Diversas linhas podem ser seguidas focando na continuidade deste trabalho:

- desenvolver um modelo de escalonamento mais complexo, permitindo que sejam definidas ordens de escalonamento (por exemplo, que todos as entidades do tipo x fossem escalonadas antes das entidades do tipo y). Além disto, é importante possibilitar que ações sejam escalonadas antes da simulação iniciar, após o fim da simulação, em momentos específicos e durante intervalos de tempo;

- integrar ao modelo proposto funções estatísticas, facilitando a análise dos resultados das simulações;

- na implementação do protótipo não houve grandes preocupações com questões de desempenho. Sendo assim, é necessário a realização de otimizações específicas na plataforma e também relacionadas com o BDG (utilização de índices, otimização de consultas), principalmente para a realização de simulações com uma grande quantidade de entidades;

- para o aperfeiçoamento do protótipo implementado, também é importante a criação de novas simulações, possibilitando assim a detecção das necessidades dos usuários, bem como de eventuais falhas e problemas existentes.

REFERÊNCIAS

ADAMATTI, D. F. **Estudo Comparativo de Ambientes de Simulação Baseados em Agentes**. 2001. Trabalho Individual (Mestrado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre, RS.

ADAMATTI, D. F. **AFRODITE - Ambiente de Simulação Baseado em Agentes com Emoções**. 2003. Dissertação (Mestrado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre, RS.

AGUIAR, M. S. d. **Um Modelo Categorizador Intervalar n-Dimensional com l-Camadas Baseados em Tesselações**. 2004. Tese (Doutorado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre, RS.

ALLIE, B. G.; HÄRING, G. **pyPgSQL Website**. 2007. Disponível em: <http://pypgsql.sourceforge.net/>. Acesso em: 14 jan. 2007.

ALVARES, L. O. C.; SICHMAN, J. S. Introdução aos Sistemas Multiagentes. In: JORNADA DE ATUALIZAÇÃO EM INFORMÁTICA, 16., 1997, Brasília, DF. **Anais...** Brasília, DF: UnB, 1997. p. 1–37.

ARONOFF, S. **Geographic Information Systems**: A Management Perspective. Ottawa, Canada: WDL Publications, 1989.

BARBALHO, V. M. d. S. **Sistemas Baseados em Conhecimento e Lógica Difusa para Simulação do Processo Chuva-Vazão**. 2001. Tese (Doutorado em Engenharia Civil) — Univesidade Federal do Rio de Janeiro, Rio de Janeiro, RJ.

BARONE, D. A. C. (Ed.). **Sociedades Artificiais**: A Nova Fronteira da Inteligência nas Máquinas. Porto Alegre, RS: Bookman, 2003.

BARROS, J. Simulating urban growth dynamics in Latin American cities. In: INTERNATIONAL CONFERENCE ON GEOCOMPUTATION, 7., 2003, Southampton, UK. **Proceedings...** [S.l.: s.n.], 2003.

BARROS, J.; SOBREIRA, F. City of Slums: self-organisation across scales. **CASA Working Paper Series**, London, UK, n. 55, 2002. Disponível em: <http://www.casa.ucl.ac.uk/working_papers/paper55.pdf>. Acesso em: 02 nov. 2006.

BASTOS, E. N. F. **ZMAS - Uma Plataforma de Sistemas Multiagentes para o Ambiente ZOPE**. 2004. Trabalho de Conclusão (Bacharelado em Ciência da Computação) — Escola de Informática, Universidade Católica de Pelotas, Pelotas, RS.

BBN TECHNOLOGIES. **OpenMap Website**. 2006. Disponível em: <http://openmap.bbn.com/>. Acesso em: 02 nov. 2006.

BENENSON, I.; ARONOVICH, S.; NOAM, S. Let's Talk Objects: Generic Methodology for Urban High-Resolution Simulation. **Computers, Environment and Urban Systems**, New York, v. 29, p. 425–453, 2005.

BENENSON, I.; TORRENS, P. M. Geosimulation: object-based modeling of urban phenomena. **Computers, Environment and Urban Systems**, New York, v. 28, n. 1, p. 1–8.

BENENSON, I.; TORRENS, P. M. Geographic Automata Systems: A New Paradigm for Integrating GIS and Geographic Simulation. In: INTERNATIONAL CONFERENCE ON GEOCOMPUTATION, 7., 2003, Southampton, UK. **Proceedings...** [S.l.: s.n.], 2003.

BENENSON, I.; TORRENS, P. M. **Geosimulation**: Automata-based Modeling of Urban Phenomena. London, UK: John Wiley & Sons, 2004. ISBN 0-470-84349-7.

BENENSON, I.; TORRENS, P. M. A Minimal Prototype for Integrating GIS and Geographic Simulation through Geographic Automata Systems. In: ATKINSON, P. M. et al. (Ed.). **Geodynamics**. Boca Raton, FL, USA: CRC Press, 2005. Cap. 23, p. 347-367.

BORDINI, R. H.; VIEIRA, R.; MOREIRA, A. F. Fundamentos de Sistemas Multiagentes. In: JORNADA DE ATUALIZAÇÃO EM INFORMÁTICA, 20., 2001, Fortaleza. **Anais...** [S.l.]: SBC, 2001. v. 2, p. 3–41.

BOX, P. **Kenge Website**. 1999. Disponível em: <http://www.gis.usu.edu/swarm/>. Acesso em: 02 nov. 2006.

BURROUGH, P. A. **Principles of Geographic Information Systems for Land Resource Assessment**. New York, USA: Oxford University Press, 1986.

BURROUGH, P. A.; FRANK, A. U. Concepts and paradigms in spatial information: are current geographical information systems truly generic? **International Journal of Geographical Information Systems**, London, England, v. 9, n. 2, p. 101–116, 1995.

CÂMARA, G. Representação computacional de dados geográficos. In: CASANOVA, M. A. et al. (Ed.). **Bancos de Dados Geográficos**. Curitiba, PR: MundoGEO, 2005. p. 504.

CÂMARA, G.; DAVIS, C. Apresentação. In: CÂMARA, G.; DAVIS, C.; MONTEIRO, A. M. V. (Ed.). **Introdução à Ciência da Geoinformação**. São José dos Campos, SP: INPE, 2001.

CÂMARA, G.; DAVIS, C.; MONTEIRO, A. M. V. (Ed.). **Introdução à Ciência da Geoinformação**. São José dos Campos, SP: INPE, 2001.

CÂMARA, G.; MEDEIROS, J. S. d. **Geoprocessamento para Projetos Ambientais**. 1998. Disponível em: <http://www.dpi.inpe.br/gilberto/tutoriais/gis_ambiente/>. Acesso em: 27 ago. 2006.

CÂMARA, G.; QUEIROZ, G. R. d. Arquitetura de Sistemas de Informação Geográfica. In: CÂMARA, G.; DAVIS, C.; MONTEIRO, A. M. V. (Ed.). **Introdução à Ciência da Geoinformação**. São José dos Campos, SP: INPE, 2001.

CARVALHO, M. S.; PINA, M. F.; SANTOS, S. M. (Ed.). **Conceitos Básicos de Sistemas de Informação Geográfica e Cartografia Aplicados à Saúde**. Brasília, DF: OPAS/Ministério da Saúde, 2000.

CASTELFRANCHI, C. Simulating with Cognitive Agents: The Importance of Cognitive Emergence. In: INTERNATIONAL WORKSHOP ON MULTI-AGENT SYSTEMS AND AGENT-BASED SIMULATION, 1., 1998, Paris. **Multi-Agent Systems and Agent-Based Simulation**: proceedings. Berlin: Springer-Verlag, 1998. p. 26–44. (Lecture Notes in Computer Science, v. 1534).

CASTLE, C. J. E.; CROOKS, A. T. Principles and Concepts of Agent-Based Modelling for Developing Geospatial Simulations. **CASA Working Paper Series**, London, UK, n. 110, 2006. Disponível em: <http://www.casa.ucl.ac.uk/working_papers/paper110.pdf>. Acesso em: 02 nov. 2006.

COUCLELIS, H. From cellular automata to urban models: new principles for model development and implementation. **Environment and Planning B: Planning and Design**, Pion, London, England, v. 24, p. 165–174, 1997.

COWEN, D. J. GIS versus CAD versus DBMS: What are the differences? **Photogrammetric Engineering & Remote Sensing**, [S.l.], v. 54, n. 11, p. 1551–1554, 1988.

DEMAZEAU, Y. From cognitive interactions to collective behaviour in agent-based systems. In: EUROPEAN CONFERENCE ON COGNITIVE SCIENCE, 1995, Saint-Malo, France. **Proceedings...** [S.l.: s.n.], 1995.

DREHMER, G. **Protótipo de um Gerente para a Plataforma MAS-SOC**. 2003. Trabalho de Conclusão (Bacharelado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre, RS.

DROGOUL, A. **De la simulation multi-agents à la résolution collective de problèmes**. 1993. Tese (Doutorado) — Université de Paris VI, Paris, France.

ENGELMORE, R.; MORGAN, T. **Blackboard Systems**. [S.l.]: Addison-Wesley, 1988.

ENVIRONMENTAL SYSTEMS RESEARCH INSTITUTE, INC. **ESRI Shapefile Technical Description**. 1998. Disponível em: <http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf>. Acesso em: 02 nov. 2006.

ENVIRONMENTAL SYSTEMS RESEARCH INSTITUTE, INC. **ESRI Website**. 2006. Disponível em: <http://www.esri.com/>. Acesso em: 02 nov. 2006.

FERBER, J. **Multi-Agent Systems**: An Introduction to Distributed Artificial Intelligence. Boston, USA: Addison-Wesley Longman, 1999.

FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS. **FIPA Website**. 2007. Disponível em: <http://www.fipa.org/>. Acesso em: 14 jan. 2007.

FROZZA, R. **SIMULA: Ambiente para Desenvolvimento de Sistemas Multiagentes Reativos**. 1997. Dissertação (Mestrado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre, RS.

FROZZA, R. **Estudo sobre Coordenação de Agentes em Ambientes Multiagentes**. 1999. Trabalho Individual (Mestrado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre, RS.

GILBERT, N.; CONTE, R. (Ed.). **Artificial Societies**: The Computer Simulation of Social Life. London, England: UCL Press, 1995.

GILBERT, N.; TROITZSCH, K. G. **Simulation for the Social Scientist**. Bristol, PA, USA: Taylor & Francis, 1999. 273 p.

GINZBURG, C. **Olhos de Madeira**: Nove Reflexões sobre a Distância. São Paulo, SP: Companhia das Letras, 2001.

GONÇALVES, A. **Multi-Agentes para Simulação em Sistemas de Informação Geográfica**. 2003. Dissertação (Mestrado) — Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, Lisboa, Portugal.

GOODCHILD, M. F. Geographical data modeling. **Computers & Geosciences**, London, England, v. 18, n. 4, p. 401–408, 1992.

GRIGOLETTI, P. S. **Modelos de Simulação de Fenômenos Espaço-Temporais Dinâmicos em Geoprocessamento**. 2005. Trabalho Individual (Mestrado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre, RS.

HUHNS, M.; STEPHENS, L. M. Multiagent Systems and Societies of Agents. In: WEISS, G. (Ed.). **Multiagent Systems**: A Modern Approach to Distributed Artificial Intelligence. [S.l.]: The MIT Press, 1999. p. 121–164.

ITAMI, R. M.; GIMBLETT, H. R. Intelligent Recreation Agents in a Virtual GIS World. **Complexity International Journal**, [S.l.], v. 08, 2001.

JENNINGS, N. R.; SYCARA, K. P.; WOOLDRIDGE, M. J. A Roadmap of Agent Research and Development. **Journal of Autonomous Agents and Multi-Agent Systems**, [S.l.], v. 1, n. 1, p. 7–38, 1998. Disponível em: <http://citeseer.ist.psu.edu/jennings98roadmap.html>. Acesso em: 02 nov. 2006.

KLÜGL, F.; HERRLER, R.; FEHLER, M. SeSAm: implementation of agent-based simulation using visual programming. In: INTERNATIONAL JOINT CONFERENCE ON AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS, AAMAS, 5., 2006, Hakodate, Japan. **Proceedings...** New York: ACM Press, 2006. p. 1439–1440.

KLÜGL, F. et al. **SeSAm Website**. 2006. Disponível em: <http://www.simsesam.de/>. Acesso em: 02 nov. 2006.

KLÜGL, F.; HERRLER, R.; OECHSLEIN, C. From Simulated to Real Environments: How to Use SeSAm for Software Development. In: GERMAN CONFERENCE IN MULTIAGENT SYSTEM TECHNOLOGIES, MATES, 1., 2003, Erfurt, Germany. **Proceedings...** Berlin/Heidelberg: Springer, 2003. p. 13–24. (Lecture Notes in Computer Science, v. 2831).

KLÜGL, F.; PUPPE, F. The Multi-Agent Simulation Environment SeSAm. In: WORKSHOP SIMULATION AND KNOWLEDGE-BASED SYSTEMS, 1998. **Proceedings...** [S.l.: s.n.], 1998.

KOCH, A. **Linking Multi Agent Systems and GIS — Modeling and Simulating Spatial Inter Actions**. 2001. Disponível em: <http://www.rwth-aachen.de/geo/Ww/deutsch/MultiAgentsKoch.PDF>. Acesso em: 23 ago. 2006.

LAW, A. M.; KELTON, W. D. **Simulation Modeling and Analysis**. 3rd ed. New York: McGraw-Hill, 2000.

LISBOA FILHO, J. **Projeto Conceitual de Banco de Dados Geográficos através da Reutilização de Esquemas, utilizando Padrões de Análise e um Framework Conceitual**. 2000. Tese (Doutorado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre, RS.

LISBOA FILHO, J.; IOCHPE, C. **Introdução a Sistemas de Informações Geográficas com Ênfase em Banco de Dados**. 1996. Disponível em: <http://www.dpi.ufv.br/~jugurta/papers/sig-bd-jai.pdf>. Acesso em: 13 set. 2006.

MAES, P. Artificial life meets entertainment: lifelike autonomous agents. **Communications of the ACM**, New York, v. 38, n. 11, p. 108–114, 1995.

MAGUIRE, D. J. An Overview and Definition of GIS. In: MAGUIRE, D. J.; GOODCHILD, M. F.; RHIND, D. W. (Ed.). **Geographical Information Systems**: Principles and Applications. London, England: Longman Scientific and Technical, 1991.

MANDL, P. **Fuzzy-System-Umgebungen als regelgesteuerte Simulationsmaschinen für Geographische Informationssysteme**. 1996. Disponível em: <http://www.uni-klu.ac.at/geo/gismosim/paper/mandl/mandl.htm>. Acesso em: 23 ago. 2006.

MICHEL, F.; FERBER, J.; GUTKNECHT, O. Generic Simulation Tools Based on MAS Organization. In: EUROPEAN WORKSHOP ON MODELLING AUTONOMOUS AGENTS IN A MULTI AGENT WORLD, MAMAAW, 10., 2001, Annecy, France. **Proceedings...** [S.l.: s.n.], 2001. Disponível em: <http://citeseer.ist.psu.edu/michel01generic.html>. Acesso em: 10 jan. 2007.

MINAR, N. et al. The swarm simulation system: a toolkit for building multi-agent simulations. **Working Papers of the Santa Fe Institute**, Santa Fe, New Mexico, USA, n. 96-06-042, 1996. Disponível em: <http://www.swarm.org/images/b/bb/MinarEtAl96.pdf>. Acesso em: 02 nov. 2006.

NAJLIS, R.; NORTH, M. J. Repast Vector GIS Integration. In: ANNUAL CONFERENCE OF THE NORTH AMERICAN ASSOCIATION FOR COMPUTATIONAL SOCIAL AND ORGANIZATIONAL SCIENCE, NAACSOS, 2005, Notre Dame, Indiana, USA. **Proceedings...** [S.l.: s.n.], 2005.

NORTH, M. J.; COLLIER, N. T.; VOS, J. R. Experiences creating three implementations of the repast agent modeling toolkit. **ACM Transactions on Modeling and Computer Simulation**, New York, v. 16, n. 1, p. 1–25, 2006.

OKUYAMA, F. Y. **Descrição e Geração de Ambientes para Simulações com Sistemas Multiagentes**. 2003. Dissertação (Mestrado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre, RS.

OLIVEIRA, D. d. **Estudo Comparativo e Projeto de um Ambiente de Simulação Baseado em Agentes**. 2003. Trabalho Individual (Mestrado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre, RS.

OLIVEIRA, F. M. d. Inteligência Artificial Distribuída. In: ESCOLA REGIONAL DE INFORMÁTICA, 4., 1996, Canoas, RS. **Anais...** [S.l.: s.n.], 1996. p. 54–73.

OPEN GEOSPATIAL CONSORTIUM. **OGC Website**. 2007. Disponível em: <http://www.opengeospatial.org/>. Acesso em: 14 jan. 2007.

OPEN GEOSPATIAL CONSORTIUM. **OpenGIS Simple Features Specification for SQL Revision 1.1**. 2007. Disponível em: <www.opengeospatial.org/docs/99-049.pdf>. Acesso em: 14 jan. 2007.

PARUNAK, H. V. D.; SAVIT, R.; RIOLO, R. L. Agent-Based Modeling vs. Equation-Based Modeling: a case study and user's guide. In: INTERNATIONAL WORKSHOP ON MULTI-AGENT SYSTEMS AND AGENT-BASED SIMULATION, 1., 1998, Paris. **Multi-Agent Systems and Agent-Based Simulation**: proceedings. Berlin: Springer-Verlag, 1998. p. 10–25. (Lecture Notes in Computer Science, v. 1534).

PEDROSA, B.; CÂMARA, G. Modelagem Dinâmica e Geoprocessamento. In: DRUCK, S. et al. (Ed.). **Análise Espacial de Dados Geográficos**. Brasília, DF: EMBRAPA, 2004.

POSTGRESQL GLOBAL DEVELOPMENT GROUP. **PostgreSQL Website**. 2007. Disponível em: <http://www.postgresql.org/>. Acesso em: 14 jan. 2007.

PYTHON SOFTWARE FOUNDATION. **Python Programming Language Website**. 2006. Disponível em: <http://www.python.org/>. Acesso em: 02 nov. 2006.

REBONATTO, M. T. **Um Estudo Sobre Simulação Paralela**. 1999. Trabalho Individual (Mestrado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre, RS.

REFRACTIONS RESEARCH. **PostGIS Website**. 2007. Disponível em: <http://postgis.refractions.net/>. Acesso em: 14 jan. 2007.

REFRACTIONS RESEARCH; VIVID SOLUTIONS; UNIVERSITY OF VICTORIA. **GEOS Website**. 2007. Disponível em: <http://geos.refractions.net/>. Acesso em: 14 jan. 2007.

REPAST ORGANIZATION FOR ARCHITECTURE AND DESIGN. **Repast Agent Simulation Toolkit Website**. 2006. Disponível em: <http://repast.sourceforge.net/>. Acesso em: 02 nov. 2006.

REYNOLDS, C. W. Steering Behaviors For Autonomous Characters. In: GAME DEVELOPERS CONFERENCE, 1999, San Jose, CA, USA. **Proceedings...** [S.l.: s.n.], 1999. p. 763–782.

RODRIGUES, A. **The Development of Spatial Intelligent Agents with GIS**. 1999. Tese (Doutorado) — City University, London, UK.

RUSSEL, S.; NORVIG, P. **Inteligência Artificial**. 2. ed. [S.l.]: Campus, 2003.

SANTOS, B. A. **Aspectos conceituais e arquiteturais para a criação de linhagens de agentes de software cognitivos e situados**. 2003. Dissertação (Mestrado) — Centro Federal de Educação Tecnológica de Minas Gerais, Belo Horizonte.

SANTOS, M. **A Natureza do espaço**: técnica e tempo, razão e emoção. São Paulo: Hucitec, 1996.

SICHMAN, J. S. **Du Raisonnement Social Chez les Agents**: Une Approche Fondée sur la Théorie de la Dépendance. 1995. Tese (Doutorado) — Institut National Polytechnique de Grenoble, Grenoble, France.

SILVA, C. A. d. **Modelagem Comportamental para Agentes Autônomos em Ambientes Reais**. 2003. Dissertação (Mestrado) — Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, Rio de Janeiro.

SILVA, E. M. d. et al. Simulação. In: **Pesquisa Operacional**. 3. ed. São Paulo, SP: Atlas, 1998.

SILVEIRA, R. A.; GOMES, E. R.; VICARI, R. M. Modelagem de ambientes de aprendizagem baseado na utilização de agentes FIPA. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 14., 2003, Rio de Janeiro, RJ. **Anais...** Rio de Janeiro, RJ: UFRJ/Nce, 2005. p. 892–901.

STRACK, J. **GPSS: Modelagem e Simulação de Sistemas**. Rio de Janeiro, RJ: LTC, 1984. 174 p.

SWARM DEVELOPMENT GROUP. **Swarm Website**. 2006. Disponível em: <http://www.swarm.org/>. Acesso em: 02 nov. 2006.

TORRENS, P. M.; BENENSON, I. Geographic Automata Systems. **International Journal of Geographical Information Science**, [S.l.], v. 19, n. 4, p. 385–412, 2005.

VASCONCELOS, E.; FURTADO, V. Um Simulador Tutorial Multi-Agente para Treinamento da Alocação de Equipes Policiais. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 25., 2005, São Leopoldo, RS. **A Univesidade da Computação**: um agente de inovação e conhecimento: anais. Porto Alegre, RS: SBC, 2005. p. 892–901.

VIVID SOLUTIONS. **JTS Website**. 2007. Disponível em: <http://www.vividsolutions.com/jts/>. Acesso em: 14 jan. 2007.

WILENSKY, U. **NetLogo**. 1999. Disponível em: <http://ccl.northwestern.edu/netlogo/>. Acesso em: 14 jan. 2007.

WOLFRAM, S. Statistical Mechanics of Cellular Automata. **Review of Modern Physics**, [S.l.], v. 55, p. 601–644, 1983.

WOOLDRIDGE, M. Intelligent agents. In: WEISS, G. (Ed.). **Multiagent Systems**: A Modern Approach to Distributed Artificial Intelligence. Cambridge, USA: The MIT Press, 1999.

WOOLDRIDGE, M.; JENNINGS, N. R. Intelligent Agents: Theory and Practice. **Knowledge Engineering Review**, [S.l.], v. 10, n. 2, p. 115–152, 1995. Disponível em: <http://citeseer.ist.psu.edu/97055.html>. Acesso em: 18 set. 2006.

WORBOYS, M. F. **GIS: A Computing Perspective**. London, England: Taylor & Francis, 1995.

ZOPE CORPORATION. **ZOPE Website**. 2007. Disponível em: <http://www.zope.org/>. Acesso em: 14 jan. 2007.

APÊNDICE A DIAGRAMA DE CLASSES DO PROTÓTIPO

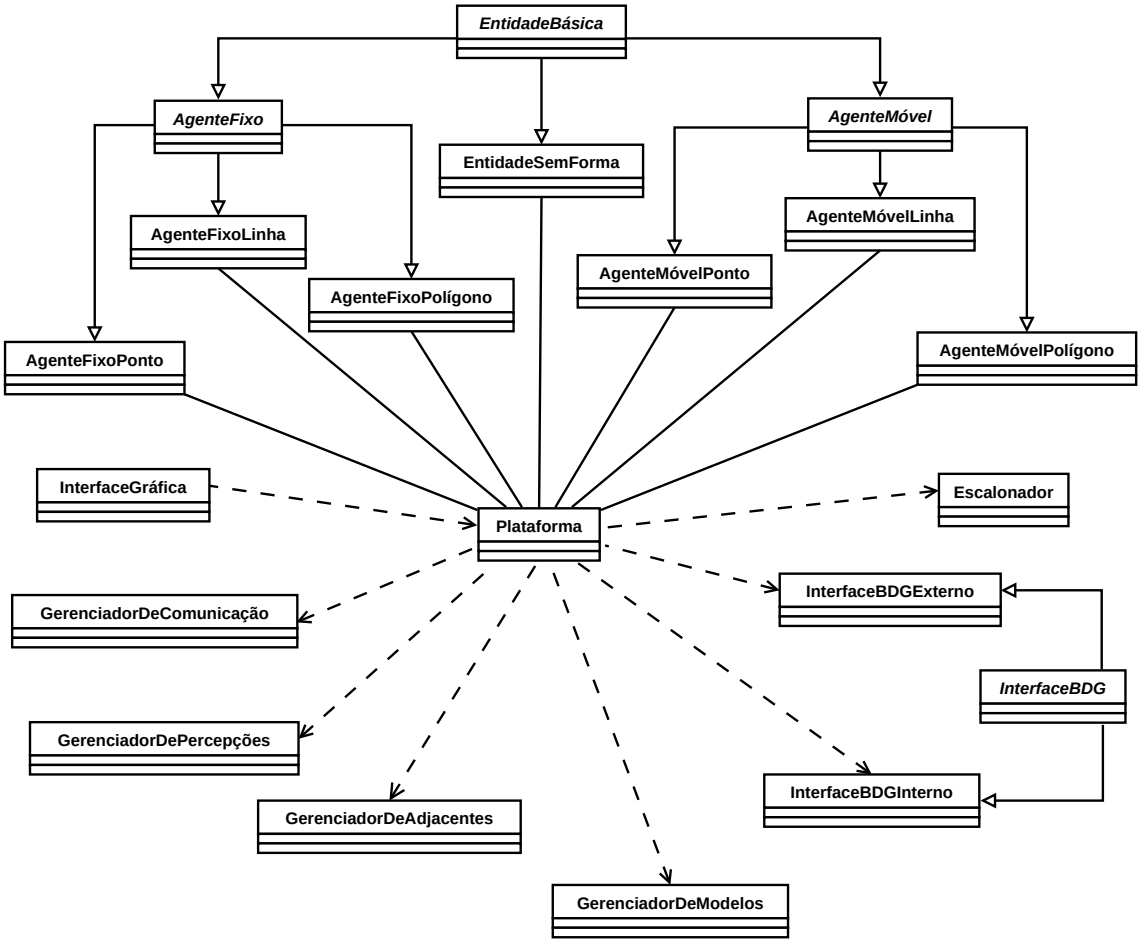

Figura A.1: Diagrama de classes do protótipo desenvolvido.

APÊNDICE B INTERFACE GRÁFICA DO PROTÓTIPO

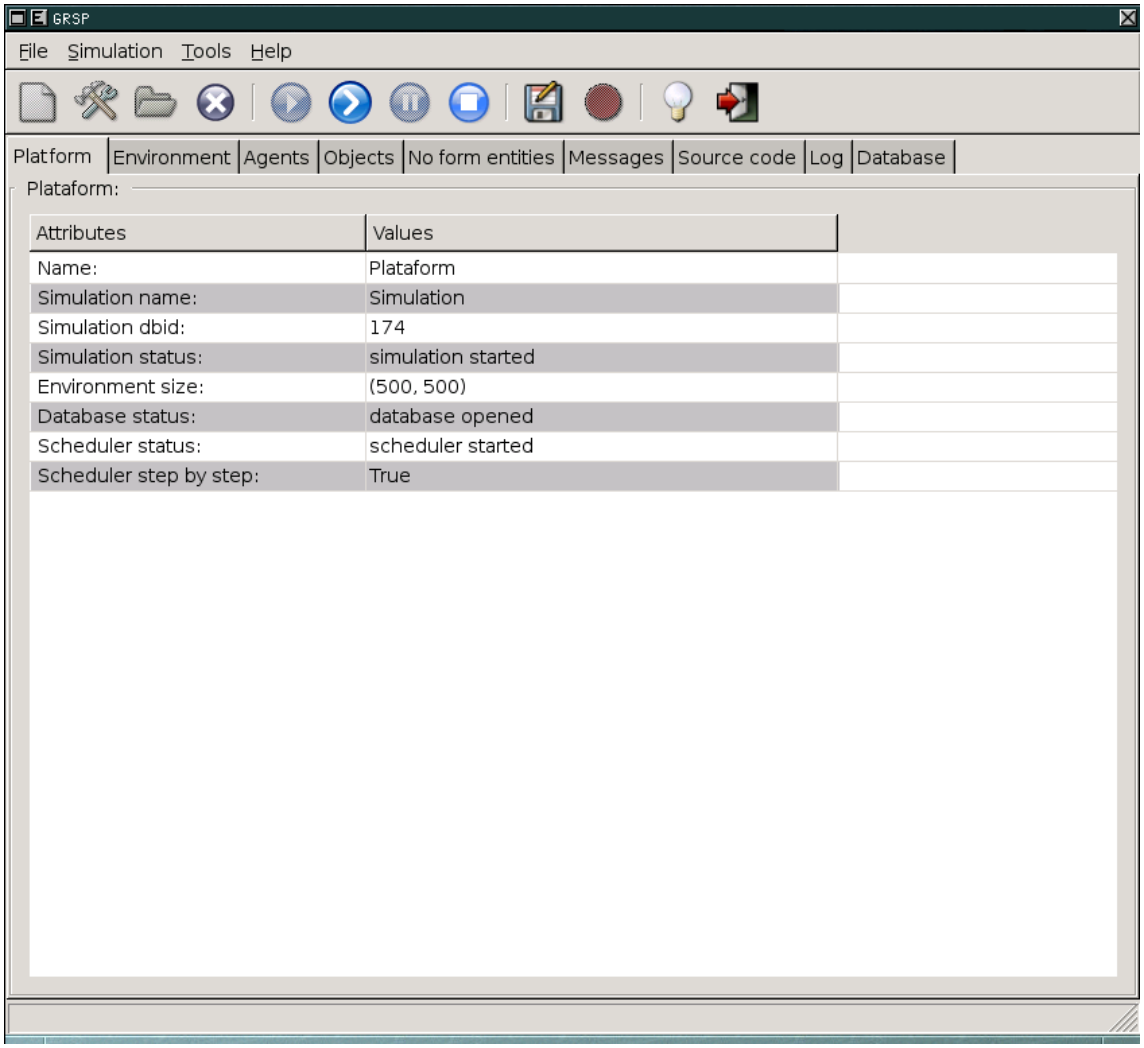

Figura B.1: Interface para visualização de diversas informações sobre a plataforma.

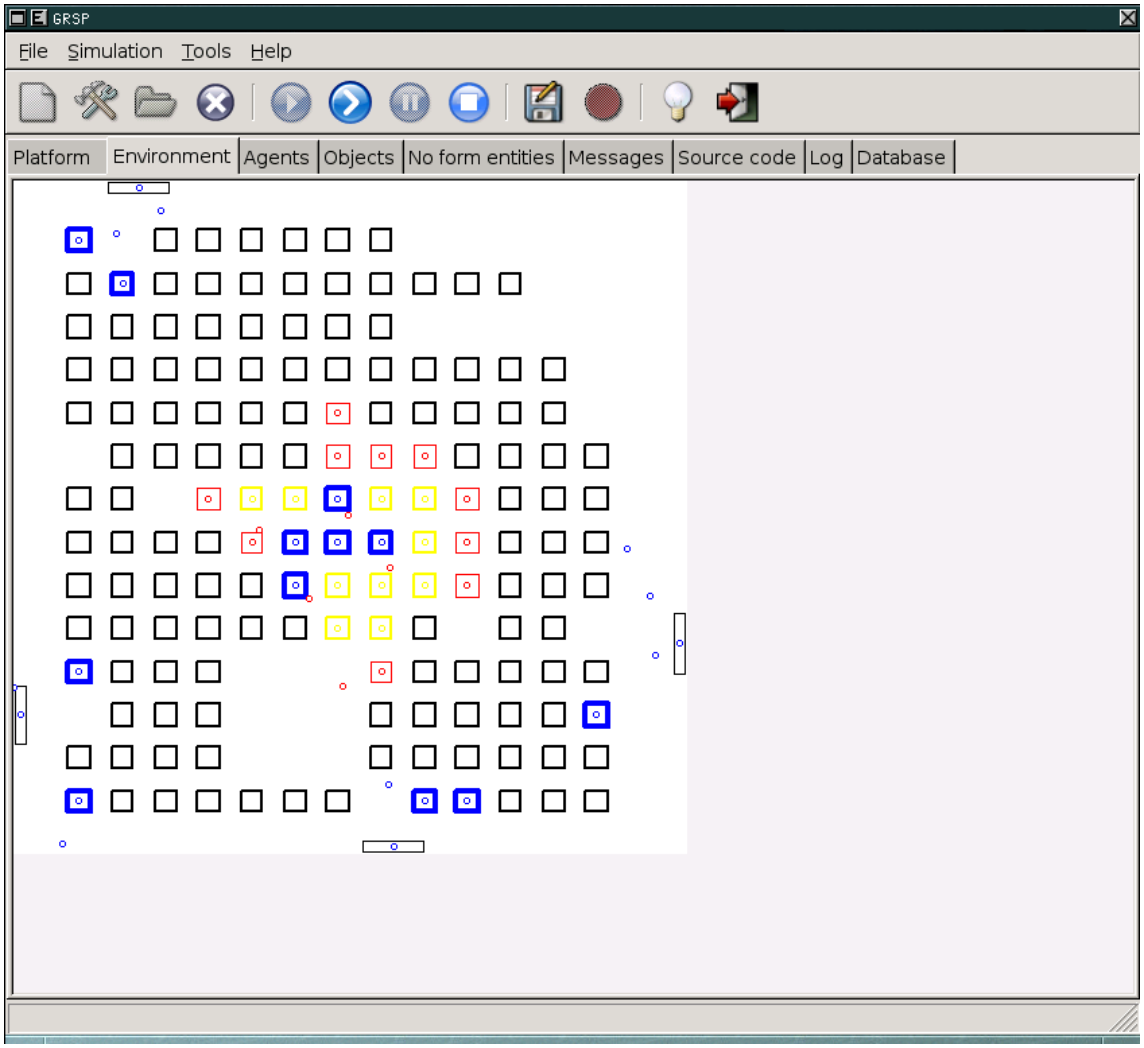

Figura B.2: Interface para visualização da forma e do posicionamento dos agentes móveis e fixos.

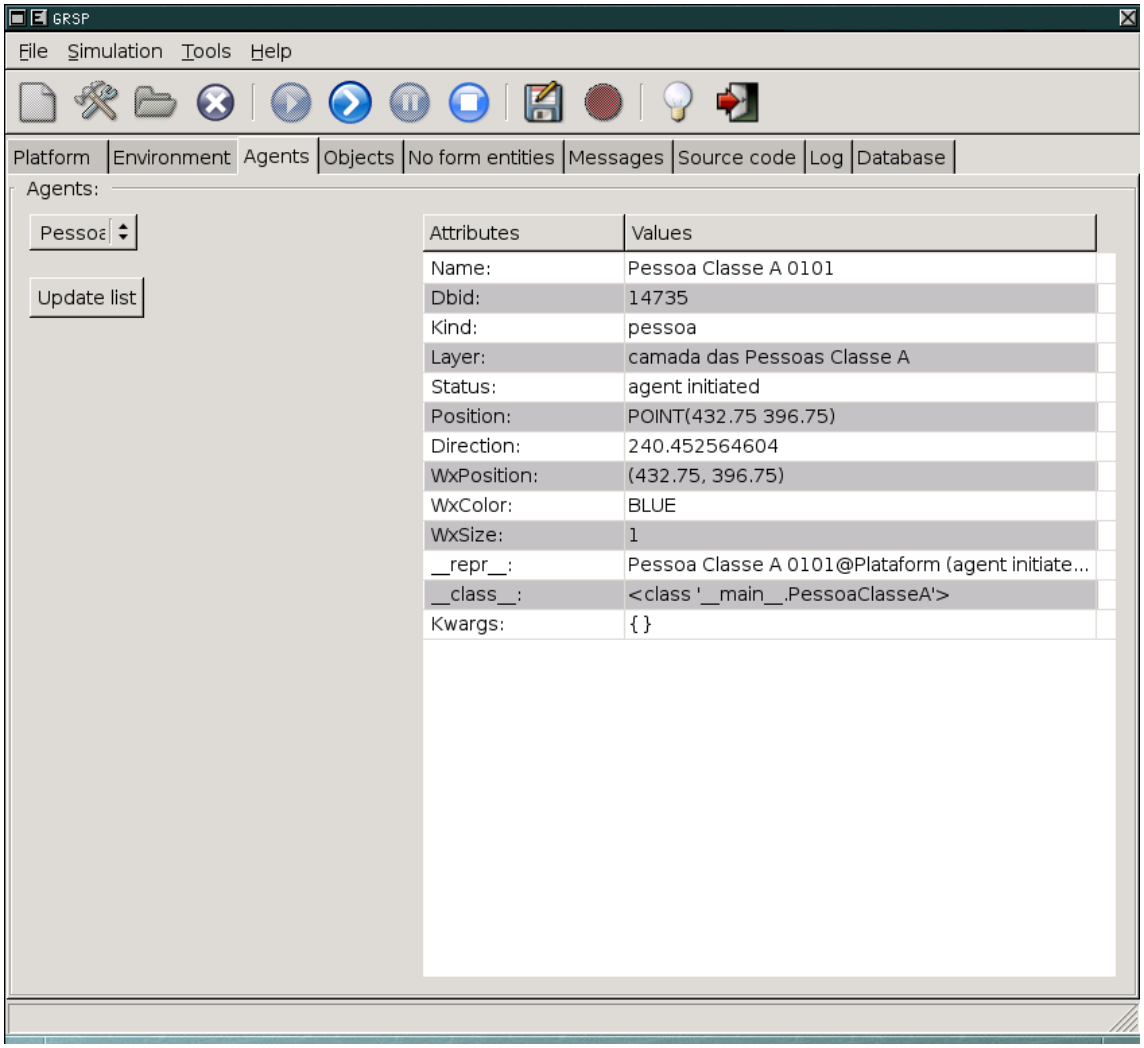

Figura B.3: Interface para visualização dos atributos dos agentes móveis. Existem interfaces semelhantes para os agentes fixos e as para as entidades sem forma.

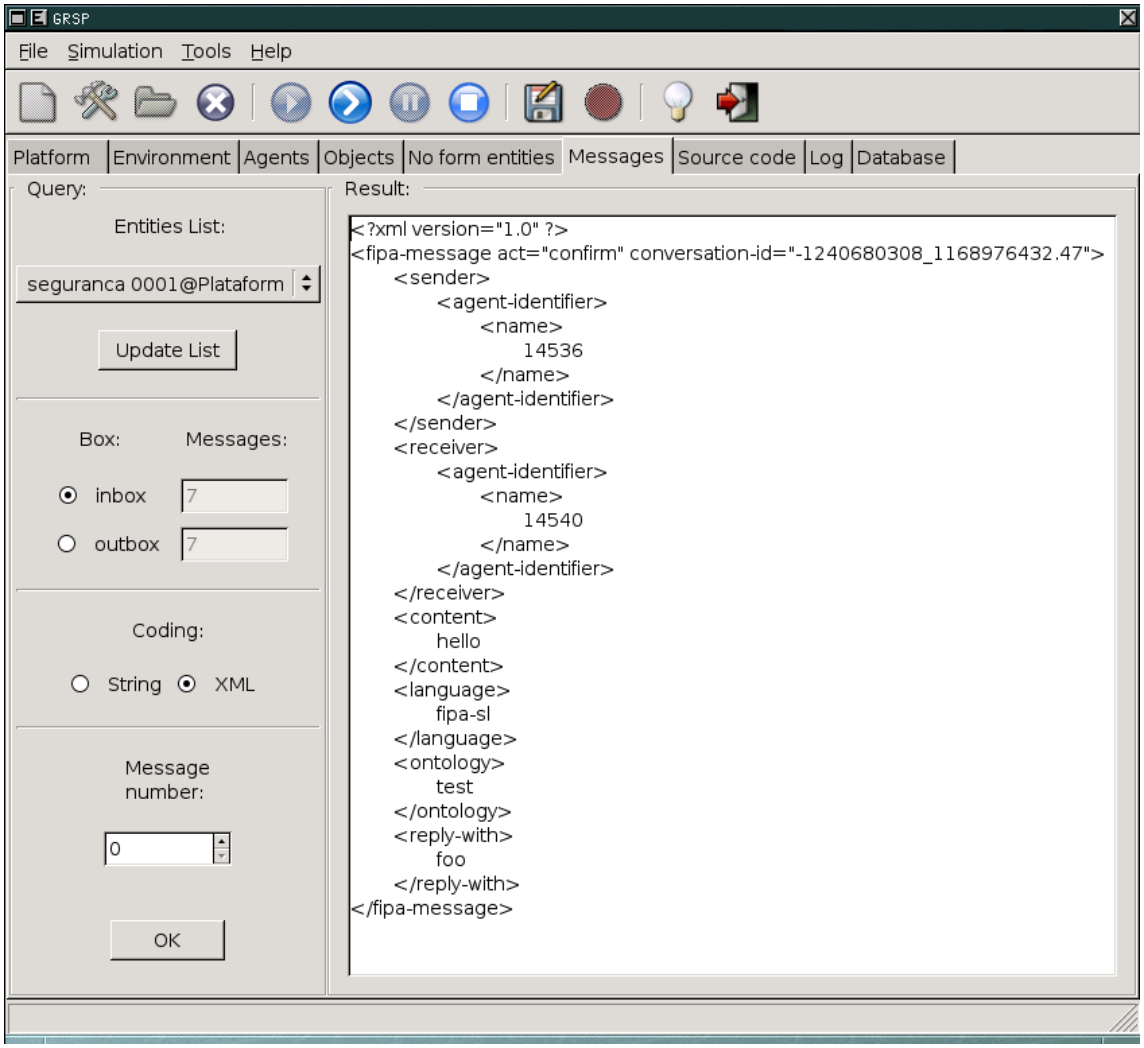

Figura B.4: Interface para consulta e visualização das mensagens trocadas entre as entidades (caixas de entrada e saída).

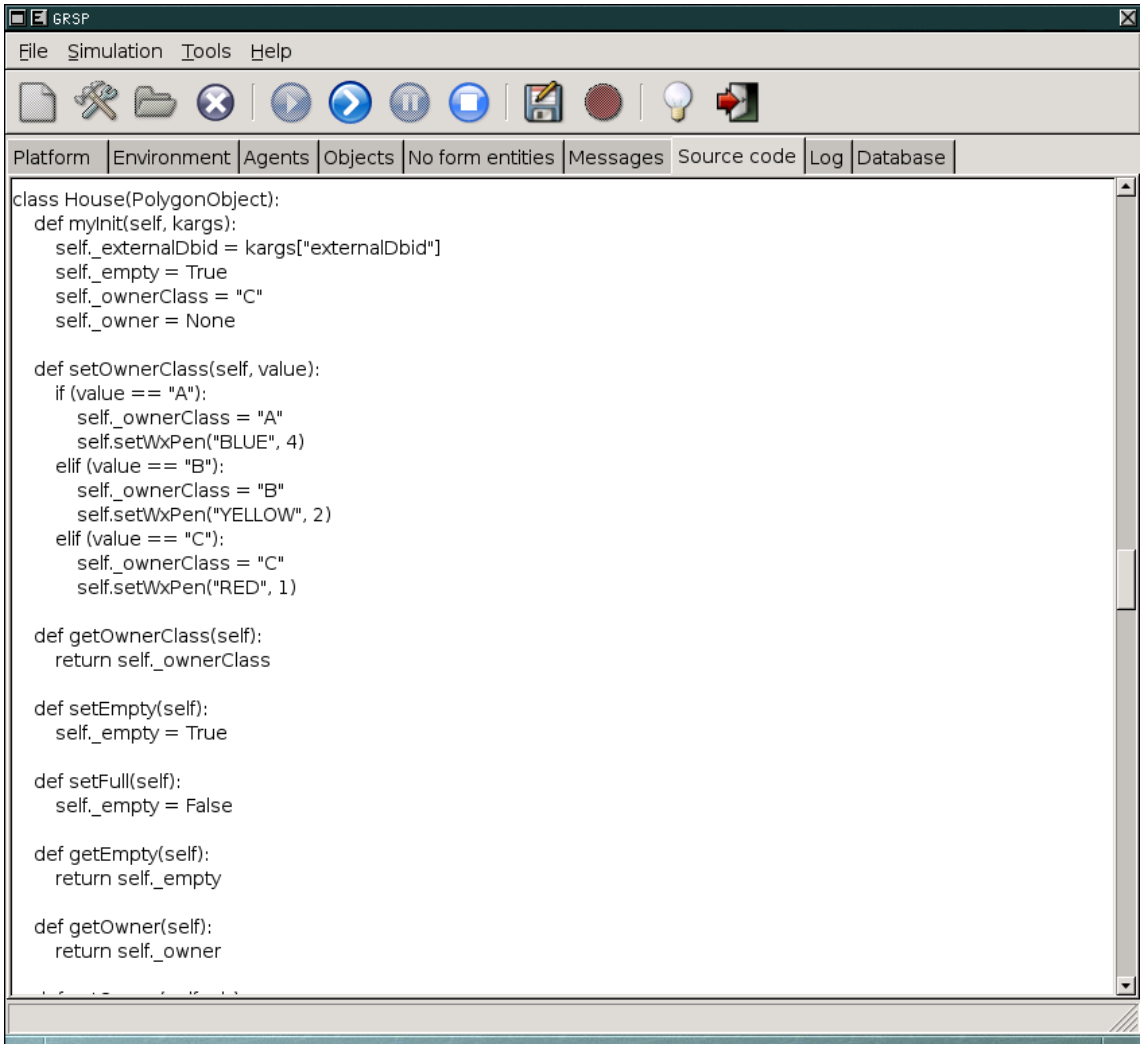

Figura B.5: Interface para edição e visualização do arquivo Python de descrição da simulação.

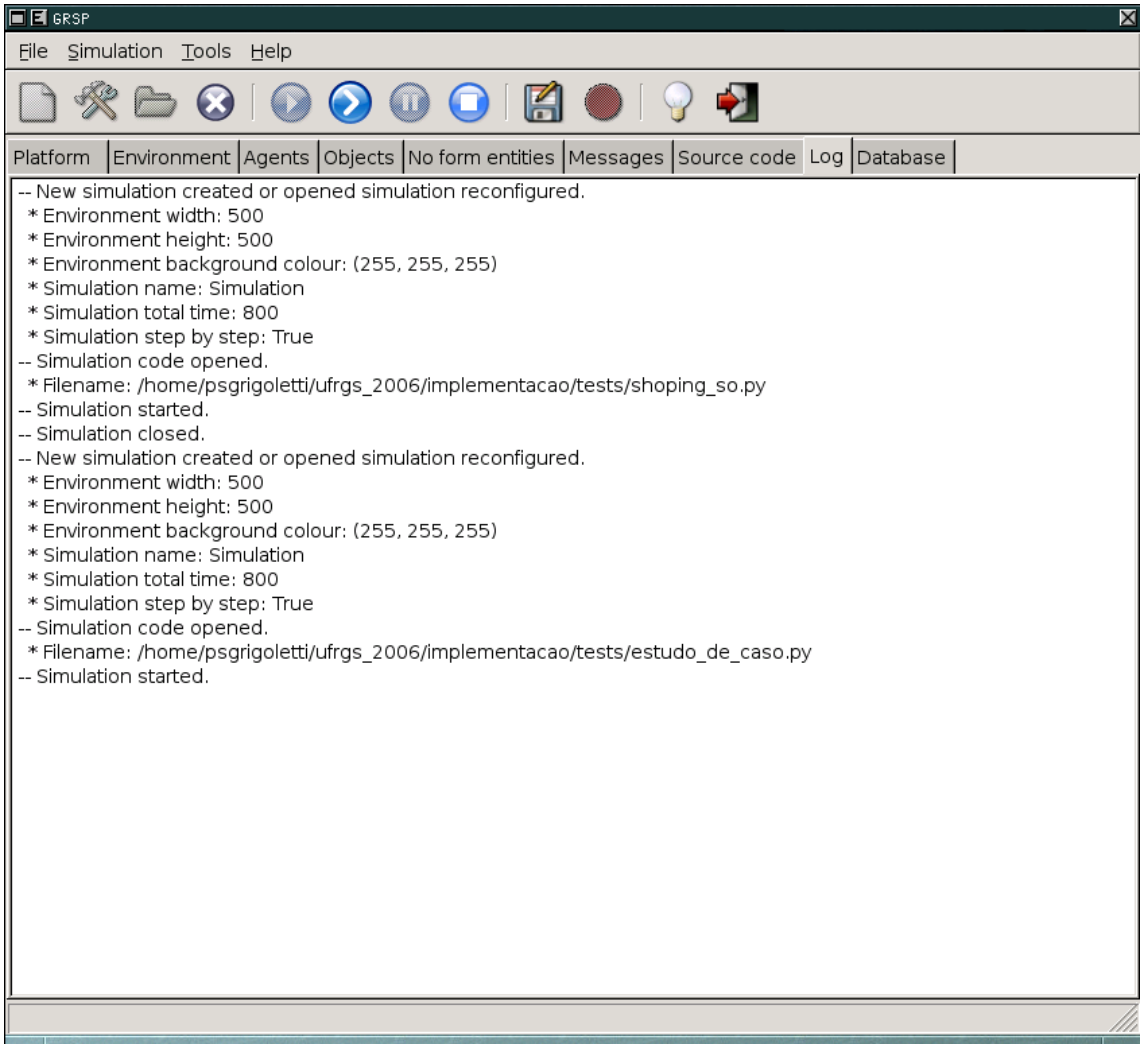

Figura B.6: Interface para visualização do arquivo de log da simulação.

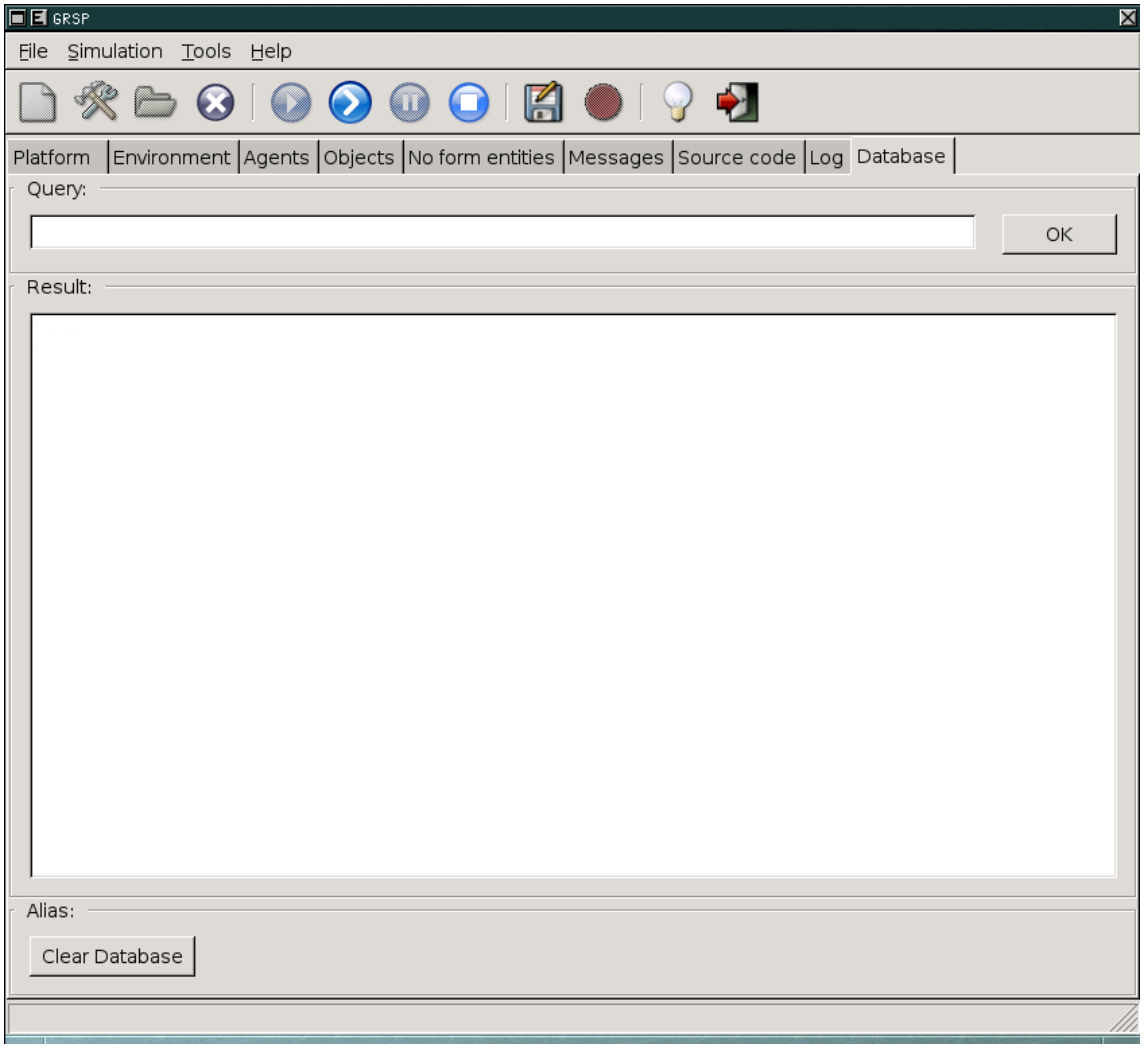

Figura B.7: Interface para realização de consultas SQL sobre o BDG interno.

APÊNDICE C ARQUIVO PYTHON

```
#-------------------------------------------------------------
# Arquivo Python referente ao Estudo de caso 02
#-------------------------------------------------------------
# Definição do tipo Limite

class Limite(LineFixedAgent):
  def run(self):
    while 1:
      yield True

#-------------------------------------------------------------
# Inserção de 4 entidades do tipo Limite

li001 = Limite("Limite 001", "Limite")
li001.setWxPen("BLACK", 1)
plataform.registerFixedAgent(li001, li001.run, "Camada dos Limites",
                             "LINESTRING(0 0, 0 500)")

li002 = Limite("Limite 002", "Limite")
li002.setWxPen("BLACK", 1)
plataform.registerFixedAgent(li002, li002.run, "Camada dos Limites",
                             "LINESTRING(0 500, 500 500)")

li003 = Limite("Limite 003", "Limite")
li003.setWxPen("BLACK", 1)
plataform.registerFixedAgent(li003, li003.run, "Camada dos Limites",
                             "LINESTRING(500 500, 500 0)")

li004 = Limite("Limite 004", "Limite")
li004.setWxPen("BLACK", 1)
plataform.registerFixedAgent(li004, li004.run, "Camada dos Limites",
                             "LINESTRING(500 0, 0 0)")

#-------------------------------------------------------------
# Definição do tipo Quadra

class Quadra(PolygonFixedAgent):
  def myInit(self, kargs):
    self._externalDbid = kargs["externalDbid"]

  def setPosition(self, position):
    self._position = position
    self._plataform.getDatabase().updateFixedAgentPosition(self._dbid,
                                                           position)
    self.updateWxPosition()

  def setPositionOnGDB(self, position):
    self._setPositionOnExternalDatabase_("ruas03", "geometria",
                                         position, "id")

  def getPosition(self):
    return self._position
```

```
  def getPositionFromGDB(self):
    return self._getPositionFromExternalDatabase_("ruas03",
                                                   "geometria", "id")

  def run(self):
    self.setPosition(self.getPositionFromGDB())
    while 1:
      yield True

#-------------------------------------------------------------
# Inserção de n entidades do tipo Quadra.
# O valor de n é igual ao número de entidades existentes no BDG.

quadrasIds = plataform.getExternalDatabase().getDbidList("id",
                                                         "ruas03",
                                                         "id",
                                                         number="ALL",
                                                         order="ASC")
quadras = []
for i in range(len(quadrasIds)):
  exec("qu%d = Quadra('Quadra %0.4d', 'Quadra',
                      externalDbid=quadrasIds[i])" % (i,i))
  exec("qu%d.setWxPen('ROYALBLUE3', 2)" % (i))
  exec("plataform.registerFixedAgent(qu%d, qu%d.run,
                                     'Camada das Quadra',
                                     'POLYGON((0 0, 0 1, 1 1,
                                               1 0, 0 0))')" %(i,i))
  exec("quadras += [qu%d]" %(i))

#-------------------------------------------------------------
# Definição do tipo Policia

class Policia(PointMobileAgent):
  def run(self):
    self.randomRotate()
    while 1:
      self.wander()
      if(self.avoidObstacles(30, ["Quadra", "Presidio", "Limite"])):
        self.move(10)
      actualPerception = self._plataform.getPerception(self)[:]
      actualPerception = self._plataform.getEntitiesById(actualPerception)[:]
      ldao = 0
      for i in actualPerception:
        if (i.getKindName() == "Ladrao"):
          ldao += 1
          ldaoid = i.getDbid()
      if ldao == 1:
        self._plataform.getEntitiesById([ldaoid])[0].setPosition("POINT(250 270)")
        self._plataform.getEntitiesById([ldaoid])[0].preso = True
      elif ldao >= 3:
        for i in self._plataform.getEntitiesByKinds(["Delegacia"]):
          msg = "(confirm :sender (agent-identifier :name %s)" % (self._fipaName, str(i))
          msg+= ":receiver (set (agent-identifier :name %s))" % "string"
          msg+= ":content x :language fipa-sl :ontology test :reply-with foo)"
          self.sendMessage(msg)
      yield True

plataform.registerTemplate(Policia)
```

```
#-------------------------------------------------------------
# Definição do tipo Delegacia

class Delegacia(PolygonFixedAgent):
  def myInit(self, kargs):
    self._externalDbid = kargs["externalDbid"]

  def setPosition(self, position):
    self._position = position
    self._plataform.getDatabase().updateFixedAgentPosition(self._dbid,
                                                           position)
    self.updateWxPosition()

  def setPositionOnGDB(self, position):
    self._setPositionOnExternalDatabase_("delegacias03", "geometria",
                                         position, "id")

  def getPosition(self):
     return self._position

  def getPositionFromGDB(self):
     return self._getPositionFromExternalDatabase_("delegacias03",
                                                   "geometria", "id")

  def run(self):
    self.setPosition(self.getPositionFromGDB())
    yield True
        yield True
        while 1:
          vvv = 40
          color = "BLUE"
          if (self.existMessagesForMe()):
            self.readAllMessages()
            vvv = 80
            color = "GREEN"

          rand = self.getRandint(0,100)
          if (rand <= vvv):
            t = self._plataform.getTemplate("Policia")
            tempVar = t("Policia", "Policia")
            tempVar.setWxPen(color, 1)
            self._plataform.registerMobileAgent(tempVar, tempVar.run,
                                                "Camada dos Policiais",
                                                str(self.getCenterPoint()))
            kindsList = ["Quadra", "Presidio", "Delegacia",
                         "Limite", "Ladrao"]

            pg001 = Perception()
            pg001.setKinds(kindsList)
            pg001.setDistance(10)
            pg001.setExpression("PBK AND PBD")
            self._plataform.registerEntityPerception(pg001, [tempVar])
          yield True

#-------------------------------------------------------------
# Inserção de n entidades do tipo Delegacias.
# O valor de n é igual ao número de entidades existentes no BDG.

delegaciasIds = plataform.getExternalDatabase().getDbidList("id", "delegacias03",
                                                            "id", number="ALL",
                                                            order="ASC")
delegacias = []
for i in range(len(delegaciasIds)):
    exec("de%d = Delegacia('Delegacia %0.4d', 'Delegacia',
                           externalDbid=delegaciasIds[i])" % (i,i))
    exec("de%d.setWxPen('ORANGE', 1)" % (i))
    exec("plataform.registerFixedAgent(de%d, de%d.run, 'Camada das Delegacias',
                                       'POLYGON((0 0, 0 1, 1 1, 1 0, 0 0))')" %(i,i))
    exec("delegacias += [de%d]" %(i))
```

```
#-----------------------------------------------------------
# Definição do tipo Presidio

class Presidio(LineFixedAgent):
  def run(self):
    while 1:
      yield True

#-----------------------------------------------------------
# Inserção das entidades do tipo Presidio.

pr001 = Presidio("Presidio 001", "Presidio")
pr001.setWxPen("BLACK", 2)
plataform.registerFixedAgent(pr001, pr001.run, "Camada dos Presidios",
                             "LINESTRING(199.5 336.0, 217.5 249.0)")

pr002 = Presidio("Presidio 002", "Presidio")
pr002.setWxPen("BLACK", 2)
plataform.registerFixedAgent(pr002, pr002.run, "Camada dos Presidios",
                             "LINESTRING(217.5 249.0, 339.0 243.0)")

pr003 = Presidio("Presidio 003", "Presidio")
pr003.setWxPen("BLACK", 2)
plataform.registerFixedAgent(pr003, pr003.run, "Camada dos Presidios",
                             "LINESTRING(339.0 243.0, 322.5 313.5)")

pr004 = Presidio("Presidio 004", "Presidio")
pr004.setWxPen("BLACK", 2)
plataform.registerFixedAgent(pr004, pr004.run, "Camada dos Presidios",
                             "LINESTRING(322.5 313.5, 199.5 336.0)")

#-----------------------------------------------------------
# Definição do tipo Ladrao

class Ladrao(PointMobileAgent):
  def run(self):
    self.preso = False
    yield True
    self.randomRotate()
    while 1:
      self.wander()
      if(self.preso):
        klist = ["Presidio"]
      else:
        klist = ["Quadra", "Presidio", "Delegacia", "Limite"]
      if(self.avoidObstacles(30, klist)):
        self.move(10)
      yield True

plataform.registerTemplate(Ladrao)

#-----------------------------------------------------------
# Inserção das entidades do tipo Ladrao.

la000 = Ladrao("Ladrao 000", "Ladrao")
la000.setWxPen("BROWN3", 1)
plataform.registerMobileAgent(la000, la000.run, "Camada dos Ladroes",
                              "POINT(2 2)")

#-----------------------------------------------------------
# Definição da percepção do tipo Ladrao.

pg001 = Perception()
pg001.setKinds(["Quadra", "Presidio", "Delegacia", "Limite"])
pg001.setDistance(10)
pg001.setExpression("PBK AND PBD")
plataform.registerEntityPerception(pg001, [la000])
```

```
#-----------------------------------------------------------
# Definição do tipo Entrada1

class Entrada1(PointFixedAgent):
  def run(self):
    while 1:
      yield True
      yield True
      yield True
      rand = self.getRandint(0,100)
      if (rand <= 75):
        t = self._plataform.getTemplate("Ladrao")
        tempVar = t("Ladrao", "Ladrao")
        tempVar.setWxPen("BROWN3", 1)
        self._plataform.registerMobileAgent(tempVar, tempVar.run,
                                             "Camada dos Ladroes",
                                             str(self.getPosition()))
        kindsList = ["Quadra", "Presidio", "Delegacia", "Limite"]
        pg001 = Perception()
        pg001.setKinds(kindsList)
        pg001.setDistance(10)
        pg001.setExpression("PBK AND PBD")
        self._plataform.registerEntityPerception(pg001, [tempVar])

en001 = Entrada1("Entrada1 001", "Entrada")
en001.setWxPen("WHITE", 1)
plataform.registerFixedAgent(en001, en001.run, "Camada das Entradas",
                             "POINT(5 255)")

en002 = Entrada1("Entrada1 002", "Entrada")
en002.setWxPen("WHITE", 1)
plataform.registerFixedAgent(en002, en002.run, "Camada das Entradas",
                             "POINT(495 495)")

en003 = Entrada1("Entrada1 003", "Entrada")
en003.setWxPen("WHITE", 1)
plataform.registerFixedAgent(en003, en003.run, "Camada das Entradas",
                             "POINT(495 195)")

en004 = Entrada1("Entrada1 004", "Entrada")
en004.setWxPen("WHITE", 1)
plataform.registerFixedAgent(en004, en004.run, "Camada das Entradas",
                             "POINT(145 495)")
```

Figura C.1: Código-fonte do arquivo Python referente ao Estudo de Caso 02.