Universidade Federal de Minas Gerais

Instituto de Geociências

Programa de Pós-Graduação

Mestrado em Análise e Modelagem de Sistemas Ambientais

Tiago França Melo de Lima

TerraME GIMS – Uma interface gráfica para a descrição de modelos ambientais para a plataforma TerraME

Belo Horizonte

2010

Tiago França Melo de Lima

TerraME GIMS – Uma interface gráfica para a descrição de modelos ambientais para a plataforma TerraME

Dissertação apresentada ao Programa de Pós Graduação em Análise e Modelagem de Sistemas Ambientais da Universidade Federal de Minas Gerais como requisito para a obtenção do título de mestre em Análise e Modelagem de Sistemas Ambientais.

Orientador: Sergio Donizete Faria

Co-orientador: Tiago Garcia de Senna Carneiro

Belo Horizonte
Instituto de Geociências da UFMG
2010

Dissertação defendida e aprovada, em 09 de julho de 2010, pela Banca Examinadora constituída pelos professores:

Prof. Dr. Sérgio Donizete Faria

Prof. Dr. Tiago Garcia de Senna Carneiro

Prof. Dr. Britaldo Silveira Soares Filho

Prof. Dr. Clodoveu Augusto Davis Júnior

"É graça divina começar bem.
Graça maior persistir na caminhada certa.
Mas a graça das graças é não desistir nunca."
Dom Hélder Câmara

Dedico este trabalho a todos que contribuíram
para que ele acontecesse.

AGRADECIMENTOS

Agradeço a todos que ajudaram na realização deste projeto.

À Fundação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), pelo auxilio financeiro.

À Universidade Federal de Minas Gerais (UFMG), pela oportunidade.

Aos professores da UFMG e colegas da Pós-Graduação pela contribuição em minha formação.

Aos meus orientadores prof. Dr. Sergio Donizete Faria e prof. Dr. Tiago Garcia de Senna Carneiro, pela amizade, orientação, apoio, dedicação, confiança e paciência ao longo desta jornada.

À equipe do TerraLab.

Aos irmãos masmorranos, pela torcida e apoio.

À minha namorada Raquel, pelo carinho, motivação e dedicação.

À minha família.

Aos meus pais e irmãos, amo vocês.

RESUMO

A intensificação das mudanças ambientais causadas pelas atividades antrópicas tem levado ao desenvolvimento e utilização de diversas plataformas de *software* para modelagem e simulação de processos e fenômenos antrópicos e biofísicos e suas interações. Dentre estas plataformas de *software* tem-se o *TerraME*, um ambiente para modelagem e simulação de processos espaciais dinâmicos. Ele oferece, através de uma linguagem de programação de alto nível também denominada *TerraME*, uma base conceitual e serviços que permitem a seus usuários representar estruturas de dados e regras que irão definir o comportamento do modelo. Contudo, a utilização direta de uma linguagem de programação é um fator limitante à ampla utilização deste ambiente, principalmente considerando que seus principais usuários são pesquisadores e profissionais com formações diversas (biólogos, engenheiros, ecólogos, economistas, sociólogos e outros) e que muitas vezes não possuem conhecimentos básicos sobre algoritmos e técnicas de programação. Sendo assim, faz-se necessário um nível maior de abstração, permitindo que os usuários se concentrem na resolução dos problemas pertencentes ao domínio de aplicação dos modelos e não nos problemas envolvidos na representação computacional destes. A partir disto, este trabalho apresenta o desenvolvimento do *TerraME Graphical Interface for Modeling and Simulation – TerraME GIMS*, uma interface gráfica de usuário que fornece os recursos necessários à modelagem e simulação de sistemas ambientais utilizando o ambiente *TerraME* por meio de metáforas visuais que representem graficamente o modelo.

ABSTRACT

The intensification of environmental changes caused by human activities has led to the development and use of several software platforms for modeling and simulation of anthropogenic and biophysical processes and phenomena and their interactions. Among these software platforms there is *TerraME*, an environment for modeling and simulation of dynamic spatial processes. Through a high-level programming language, also called TerraME, it offers a conceptual basis and services that allow its users to represent data structures and rules that define the model's behavior. However, the direct use of a programming language can be a limiting factor to the widespread use of this environment, especially given that its main users are researchers and professionals with different backgrounds (biologists, engineers, ecologists, economists, sociologists), who often lack basic knowledge about algorithms and programming techniques. Therefore, a higher level of abstraction is necessary, allowing users to focus on solving the problems pertaining to the scope of the models and not get involved in problems of its computational representation. This work presents the development of the *TerraME Graphical Interface for Modeling and Simulation – TerraME GIMS*, a graphical user interface that provides the required resources for modeling and simulation of environmental systems using the *TerraME* platform through visual metaphors that graphically represent the model.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE SIGLAS E ABREVIATURAS

API - *Application Programming Interface*

CSR - Centro de Sensoriamento Remoto

EMF - *Eclipse Modeling Framework*

GEF - *Graphical Editor Framework*

GUI - *Graphical User Interface*

IDE - *Integrated Development Environment*

INPE - Instituto Nacional de Pesquisas Espaciais

Repast - *Recursive Porous Agent Simulation Toolkit*

ROAD - *Repast Organization for Architecture and Development*

SDG - *Swarm Development Group*

SIG - Sistema de Informações Geográficas

TerraME - *TerraLib Modeling Environment*

TerraME GIMS - *TerraME Graphical Interface for Modeling and Simulation*

UFMG - Universidade Federal de Minas Gerais

UFOP - Universidade Federal de Ouro Preto

URL - *Uniform Resource Locator*

1 INTRODUÇÃO

A intensificação das mudanças ambientais causadas por processos antrópicos exige respostas cada vez mais rápidas por parte dos tomadores de decisão (governantes, administradores, pesquisadores etc.). Estes atores, por sua vez, precisam de informações e ferramentas que permitam obter uma maior compreensão acerca do funcionamento dos sistemas terrestres e que forneçam subsídios para auxiliar suas tomadas de decisões, visando intervenções de menor impacto ao meio ambiente.

Os sistemas terrestres compreendem a interação dos sistemas sócio-econômicos (de origem antrópica: sistema de uso do solo, sistema de transporte urbano, sistema financeiro) com os sistemas biofísicos (de origem natural: sistema climático, ecossistemas aquáticos e terrestres), a qual é um processo complexo e seu entendimento exige o trabalho de uma equipe multidisciplinar, envolvendo especialistas e profissionais das mais diversas áreas do conhecimento.

A modelagem, que consiste na construção de representações simplificadas da realidade (modelos), é uma técnica que permite definir problemas e conceitos de forma mais clara, além de fornecer um meio para análise de dados e comunicação de resultados (TURNER et al., 2001). Por sua vez, a simulação computacional, ou seja, a execução de um modelo por meio de sistemas de computação, tem sido utilizada em trabalhos de investigação científica para lidar com problemas de natureza complexa e/ou quando a solução apresenta um custo muito elevado ou mesmo é impossível de ser obtida por meio de experimentos (BRATLEY et al., 1987).

Em geral, fenômenos do sistema terrestre, tais como o processo de mudança de uso e de cobertura da terra, possuem uma natureza inerentemente complexa, envolvendo diversas variáveis e dinâmicas, o que exige a utilização de técnicas e ferramentas de modelagem e simulação para estudar, compreender e representar o funcionamento dos diversos processos (antrópicos e biofísicos) que constituem o fenômeno e suas interações.

Neste contexto, diversas plataformas de *software* têm surgido e sido utilizadas para modelar e simular processos antrópicos, biofísicos e suas interações, dentre as quais podem ser citadas *NetLogo* (WILENSKY, 1999), *Swarm* (MINAR et al., 1996), *Repast* (COLLIER, 2002), *Vensim* (VENSIM, 2009), *Dinamica EGO* (RODRIGUES et al.,

2007), e *TerraME* (CARNEIRO, 2006). Muitos são os aspectos que diferenciam estas plataformas, tais como a base teórica a partir da qual foram desenvolvidas e as funcionalidades que oferecem aos usuários (maiores detalhes sobre estas plataformas são apresentadas no Capítulo 3).

O *TerraME – TerraLib Modeling Environment* (CARNEIRO, 2006), objeto de estudo nesta pesquisa, é uma plataforma de modelagem e simulação de processos ambientais que permite construir modelos espaciais dinâmicos integrados a um Sistema de Informações Geográficas (SIG). Esta plataforma, componente da família de soluções *TerraLib* (CÂMARA et al., 2000), oferece uma linguagem de programação de alto nível, também chamada *TerraME*, que permite ao modelador (usuário) representar as estruturas de dados e as regras que regerão o comportamento dos modelos de forma mais clara e eficiente quando comparada a linguagens de programação de uso geral, como *C++* (STROUSTRUP, 1994) ou *Java* (GOSLING et al., 1996). Maiores detalhes sobre esta plataforma serão apresentados no Capítulo 4.

Diversos trabalhos já utilizaram o *TerraME* como plataforma de modelagem e simulação. Andrade et al. (2009) desenvolveram um modelo para jogos espaciais, apresentando resultados que demonstram como a mobilidade afeta o equilíbrio de Nash. Um modelo de simulação de padrões de incêndio para o Parque Nacional das Emas (GO – Brasil) foi desenvolvido e apresentado por Almeida et al. (2008). Moreira et al (2008) fizeram uma análise sobre as relações espaciais entre objetos geográficos em diferentes escalas e implementaram através do *TerraLib* dois tipos de relações – hierárquica e baseada em redes, sendo a plataforma *TerraME* utilizada para a construção dos modelos. A plataforma também foi utilizada por Pimenta et al. (2008) em um estudo de caso onde foi analisado como a existência de diferentes regras de uso do território afetam a dinâmica de paisagem no nível regional. Carneiro et al. (2008) apresentam o modelo computacional para representação do conceito de espaço geográfico implementado pela plataforma *TerraME*, denominado *Irregular Cellular Space*, que oferece suporte ao desenvolvimento de modelos dinâmicos espacialmente explícitos integrados a bancos de dados geográficos.

No entanto, apesar da linguagem de modelagem *TerraME* facilitar a representação de modelos dinâmicos espaciais integrados a um SIG, a assimilação dos conceitos e das construções existentes na linguagem ainda é uma tarefa que apresenta um elevado grau

de dificuldade de aprendizagem devido à complexidade inerente aos conceitos que ela implementa e aos fenômenos ambientais aos quais se aplica. Profissionais e pesquisadores não familiarizados com algoritmos e técnicas de programação encontram dificuldades na sua utilização, conforme foi constatado em avaliações realizadas por participantes de cursos e apresentações sobre a plataforma *TerraME* (CÂMARA et al., 2007; CÂMARA et al., 2008). Este é exatamente o caso da maioria dos especialistas que se ocupam do estudo de sistemas ambientais e das interações entre os processos antrópicos e biofísicos: geógrafos, ecólogos, biólogos, antropólogos, sociólogos, economistas etc. Entretanto, são esses os profissionais mais interessados e envolvidos no desenvolvimento de modelos ambientais e que detêm o conhecimento a respeito do domínio de aplicação destes modelos.

Desta maneira, o uso direto e obrigatório de uma linguagem de programação para representação de modelos ambientais constitui a principal barreira para a utilização da plataforma *TerraME*. Assim, uma ferramenta em que são abstraídos detalhes computacionais de implementação no processo de desenvolvimento de modelos ambientais deve encontrar respaldo na comunidade de modeladores e permitir que estes se concentrem nos aspectos relacionados ao domínio do problema.

Para permitir que os usuários do *TerraME* se concentrem na resolução de problemas pertencentes ao domínio de aplicação dos modelos, não se ocupando com muitos dos problemas envolvidos na sua representação computacional, um novo e mais alto nível de abstração é necessário. A representação dos modelos através de componentes gráficos, tais como diagramas, ao invés de algoritmos, poderá tornar mais intuitivo e eficiente o processo de modelagem e simulação de sistemas ambientais, aumentando a produtividade dos atuais usuários do ambiente *TerraME* e diminuindo sobremaneira as dificuldades e o tempo de aprendizagem para os novos usuários.

A construção de modelos para a plataforma *TerraME* pode ser feita por meio da utilização de qualquer editor de textos, sendo necessárias para a execução do modelo a instalação do interpretador *TerraME* e a configuração do editor para invocar o interpretador e então executar o modelo. As limitações apresentadas pela utilização de editores não específicos na construção de modelos para o ambiente *TerraME*, além da dificuldade de muitos usuários em descrever estes modelos na forma de algoritmos, dificultam sua utilização.

Neste sentido, a disponibilização de um ambiente de desenvolvimento integrado específico para a plataforma *TerraME*, que ofereça funcionalidades para a criação e visualização dos modelos, faz-se necessária. Um ambiente desta natureza tornará mais ágil o processo de desenvolvimento e análise de modelos, aumentará a troca de informações e colaboração entre os usuários, uma vez que estes utilizarão o mesmo ambiente de desenvolvimento, além de tornar mais intuitivo o processo de construção de modelos para sistemas ambientais.

1.1 Objetivos

Este trabalho tem como objetivo geral o desenvolvimento de um ambiente de desenvolvimento integrado específico para a plataforma *TerraME*, que ofereça funcionalidades para a criação e visualização dos modelos.

E como objetivo específico o desenvolvimento do *TerraME GIMS – TerraME Graphical Interface for Modeling and Simulation*, uma interface gráfica de usuário (Graphical User Interface – GUI) para a construção de modelos dinâmicos espaço-temporais e simulação baseados no ambiente *TerraME*.

1.2 Estrutura do documento

Este trabalho é constituído por mais seis capítulos, organizados conforme descrito a seguir:

- Capítulo 2 – Modelagem e Simulação de Sistemas Ambientais: apresenta uma breve revisão da literatura acerca da modelagem e simulação de sistemas ambientais.

- Capítulo 3 – Plataformas de Modelagem e Simulação: apresenta algumas das plataformas (*NetLogo, Swarm, Repast, Vensim, Dinamica EGO*) para a construção e simulação de modelos ambientais;

- Capítulo 4 – A plataforma de Modelagem *TerraME*: apresenta a plataforma para construção de modelos e simulação de fenômenos espaciais dinâmicos *TerraME*;

- Capítulo 5 – Desenvolvimento do *TerraME GIMS*: apresenta a construção do *TerraME GIMS*, uma interface gráfica para a descrição de modelos para a plataforma *TerraME*;

- Capítulo 6 – Aplicação do *TerraME GIMS* na construção de um modelo do ciclo hidrológico: descreve a metodologia utilizada para avaliar o *TerraME GIMS*, e apresenta os resultados obtidos a partir do desenvolvimento do trabalho e a análise destes resultados;

- Capítulo 7 – Conclusão: apresenta as conclusões do trabalho.

2 MODELAGEM E SIMULAÇÃO DE SISTEMAS AMBIENTAIS

Neste capítulo é apresentada uma revisão da literatura sobre alguns dos conceitos necessários ao entendimento e desenvolvimento deste trabalho, tais como modelagem e simulação, sistemas espaciais dinâmicos, processo de modelagem.

2.1 Modelagem e simulação

Os fenômenos do mundo real são em geral multifacetados, interligados e de difícil compreensão, de tal forma que para lidar com tais fenômenos é necessária uma visão ampla e geral dos mesmos, abstraindo detalhes e considerando o conjunto de características que dê sustentação ao fenômeno em relação ao objeto em estudo (HANNON e RUTH, 2001). O processo de modelagem consiste justamente em produzir representações da estrutura e/ou funcionamento de um sistema[1] com o objetivo de melhor compreender a realidade observada. Portanto, um modelo pode ser definido como uma representação simplificada, uma abstração da realidade (RENNÓ e SOARES, 2007; TURNER et al., 2001; HANNON e RUTH, 2001). Segundo Novaes (1981), esta representação pode ser feita por meio de diferentes linguagens, tais como matemática, lógica, icônica, gráfica etc., e se basear em uma ou mais teorias.

Esta compreensão dos fenômenos através da modelagem permite, por meio de análises, estimar o comportamento futuro do sistema em estudo, ou seja, é possível analisar propriedades e comportamento do sistema de uma maneira prática, fornecendo ao modelo insumos (entrada) e observando as respectivas respostas (saída) (BRATLEY et al., 1987). Este procedimento é conhecido como a simulação de um sistema, o que corresponde à execução de um modelo matemático-computacional que o representa (MARIA, 1997).

A construção de modelos é útil em diversas áreas do conhecimento uma vez que: ajuda a definir problemas e conceitos de forma mais precisa e clara; fornece um meio para análise de dados e comunicação de resultados; e permite, através de simulações, fazer predições acerca do comportamento futuro do objeto de estudo (TURNER et al., 2001). Além disto, a modelagem e a simulação são extremamente úteis quando a solução de

[1] Sistema é um conjunto de partes ou elementos independentes, denominados subsistemas, que interconectados formam uma unidade ou todo, ou seja, o sistema.

problemas possui um custo muito elevado ou é até mesmo impossível de ser obtida por meio de experimentos, e também para lidar com problemas complexos ou do tipo “caixa-preta”, ou seja, quando não se tem uma clara compreensão acerca das propriedades e comportamentos internos do objeto de estudo (BRATLEY et al., 1987).

Um aspecto também importante é a necessidade de se utilizar uma abordagem e uma equipe de trabalho multidisciplinar, envolvendo especialistas em diversas áreas do conhecimento, para lidar com alguns problemas de natureza complexa. Por exemplo, a criação de um modelo para o processo de desmatamento da Floresta Amazônica deveria envolver biólogos, ecólogos, sociólogos, economistas, geógrafos, tecnólogos etc.

Desta forma, a representação de um modelo, realizada por meio de uma linguagem de modelagem, deve ser de comum entendimento para todas as pessoas envolvidas no processo de modelagem, sendo específica o suficiente para facilitar a representação do conhecimento que cada um tem a respeito do sistema, e geral o suficiente para não restringir a representação do modelo a um domínio de aplicação específico. Ademais, o desenvolvimento de modelos para simulação computacional exige que a linguagem utilizada seja livre de ambigüidades para permitir a representação e execução destes por um computador. A escolha da linguagem a ser utilizada para a construção de modelos é, portanto, um fator crucial para o sucesso de um projeto de modelagem que envolva a colaboração de profissionais com diferentes formações.

Os modelos podem ser classificados com base em diferentes aspectos: tipo de variáveis utilizadas (estocásticos ou determinísticos); tipo de relações entre essas variáveis (empíricos ou baseados em processos); forma de representação dos dados (discretos ou contínuos); existência ou não de relações espaciais (pontuais ou distribuídos); existência de dependência temporal (estáticos ou dinâmicos); dinâmica dos processos (tempo contínuo ou discreto) (RENNÓ e SOARES, 2007; TURNER et al., 2001).

A modelagem e simulação de sistemas têm sido utilizadas nos mais diversos campos científicos na busca de um melhor entendimento do objeto de estudo e maior compreensão da realidade observada. O estudo de fenômenos dos sistemas ambientais, devido à sua natureza inerentemente complexa, exige a utilização de técnicas e ferramentas de modelagem e simulação para representar e estudar o funcionamento dos processos antrópicos e biofísicos e suas interações, a ser discutido a seguir.

2.2 Modelagem de Sistemas Ambientais

De acordo com Nobre (2008, p. 3), "[...] Sistema Terrestre é uma maneira de enxergar a Terra com todos os seus elementos vivos e não-vivos, com vários compartimentos, componentes inter-relacionados, interligados, interativos".

Em geral, os fenômenos dos sistemas ambientais que constituem o sistema terrestre são complexos e, segundo Steyaert (1993), podem incluir comportamentos não-lineares, propriedades e comportamentos estocásticos, sobre múltiplas escalas de espaço e tempo. Assim, apesar de muitas vezes ser possível um entendimento qualitativo a respeito do fenômeno, um entendimento quantitativo pode ser limitado. Ainda conforme o autor, a possibilidade de expressar o fenômeno por meio de um conjunto de equações matemáticas pode não existir ou mesmo ser tão complexo que exige simplificações para sua utilização. Steyaert (1993) destaca ainda que estas equações constituem uma aproximação desenvolvida pelo modelador para explicar estes processos, e que é importante reconhecer que um modelo ambiental é, no melhor caso, apenas uma representação dos fenômenos do mundo real.

É praticamente impossível considerar, ao mesmo tempo e com a mesma intensidade, todos os aspectos da realidade ambiental. Toda particularização a ser introduzida no processo de modelagem deve ser feita levando-se em conta o caráter sintético inerente aos modelos ambientais. Esses são sínteses, que se resolvem segundo a expressão espacial das entidades envolvidas, ou seja, sua distribuição territorial. Como sínteses, constituem-se em uma visão de conjunto, elucidativa do jogo integrado dos fatores físicos, bióticos e socioeconômicos responsáveis pela realidade ambiental. Porém, não podem, ao mesmo tempo, conter todos os aspectos dessa realidade. (SILVA, 2007)

A modelagem, portanto, mais que uma ferramenta útil, é essencial para o estudo de sistemas ambientais, pois além de contribuir para a compreensão dos fenômenos em estudo, permite a realização de predições (simulações) que torna possível, a partir do desenvolvimento de vários cenários (entradas), a avaliação de impactos ambientais (saídas) causados por processos diversos como a construção de rodovias, a criação de parques e reservas, a legislação de uso e ocupação do solo etc.

Construir modelos para descrever fenômenos ambientais geralmente implica em representar processos dinâmicos e que possuem componentes (propriedades ou

comportamentos) com localização espacial. Segundo Pedrosa e Câmara (2007), um dos grandes desafios da Ciência da Informação Espacial é justamente o desenvolvimento de técnicas e abstrações que sejam capazes de representar adequadamente fenômenos espaço-temporais dinâmicos.

Um modelo é espacial quando variáveis, entradas ou processos possuem localizações espaciais explícitas representadas no modelo. Modelos espaciais são úteis quando a heterogeneidade dos recursos e processos é necessária para representar e predizer apropriadamente a dinâmica do sistema (TURNER et al., 2001).

Um modelo espacial dinâmico descreve a evolução de padrões espaciais de um sistema ao longo do tempo (PEDROSA e CÂMARA, 2007). Burrough (1998) define modelo espacial dinâmico, numa tradução livre, como sendo uma representação matemática de um processo do mundo real em que uma localização na superfície terrestre muda em resposta a variações nas forças direcionadoras.

Conforme Couclelis (1997 citado por PEDROSA e CÂMARA, 2007), para modelar processos espaciais dinâmicos deve ser possível representar: o espaço como uma entidade não homogênea; as vizinhanças como relações não estacionárias; as regras de transição como regras não universais; a variação do tempo como um processo regular ou irregular; o sistema como um ambiente aberto a influências externas.

De acordo com Turner et al. (2001) os modelos devem ser vistos como ferramentas ou métodos para se alcançar um fim e não como metas. Ainda segundo o autor, uma vez que o conhecimento é incompleto, suposições são sempre necessárias, e a maioria dos modelos é, portanto, empregada para explorar as conseqüências das hipóteses quanto à estrutura e/ou dinâmica do sistema em estudo. Essas hipóteses podem também evoluir, na medida em que evolui a compreensão acerca do fenômeno em estudo, no processo de construção dos modelos.

2.3 Processo de modelagem

A construção de um modelo pode ser realizada seguindo diferentes metodologias. No entanto, Hannon e Ruth (2001) identificam um conjunto de atividades gerais, ilustradas na Figura 1, que são freqüentemente seguidas: estímulos gerados por eventos reais são traduzidos em uma questão (ou conjunto de questões) sobre os eventos observados e os

processos que levaram à ocorrência destes; elementos chave dos processos e observações são identificados e formam uma versão abstrata dos eventos reais; as variáveis que descrevem estes eventos e os relacionamentos entre estas são identificados e a estrutura do modelo é estabelecida; com base nos resultados apresentados pela execução do modelo, podem ser elaboradas conclusões e fornecidas predições sobre eventos ainda a serem experimentados ou observados; e por fim, estas conclusões e predições são comparadas com os eventos reais e podem conduzir e/ou indicar a incorreção do modelo, sua aceitação, ou ainda sua revisão.

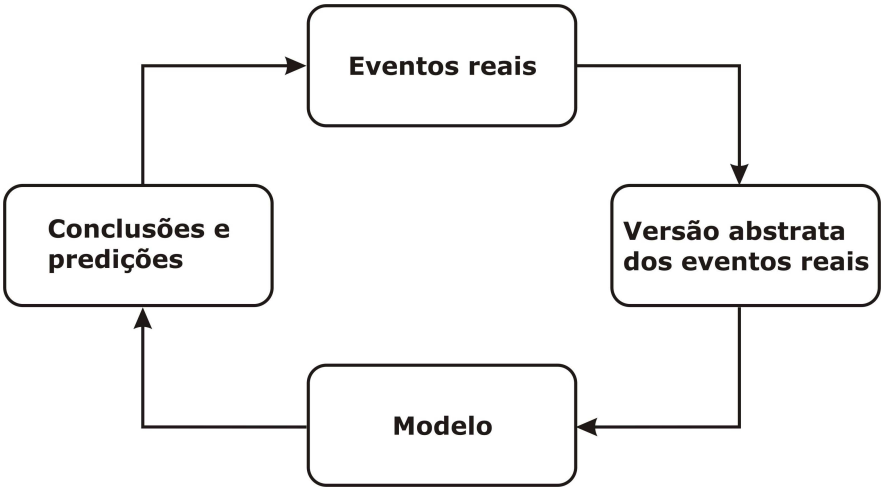


Figura 1 – Processo cíclico de desenvolvimento de modelos.
Fonte: Hannon e Ruth (2001)

Segundo Turner et al. (2001), o processo de construção de modelos inclui os seguintes passos: (1) definição do problema; (2) desenvolvimento do modelo conceitual; (3) escolha do tipo de modelo; (4) desenvolvimento do modelo através da escrita de equações matemáticas e relacionamentos; (5) implementação computacional incluindo verificação e documentação do código; (6) estimação dos parâmetros e calibração se necessário; (7) avaliação do modelo através da comparação com observações empíricas e realização de análises de sensitividade ou incerteza; (8) utilização do modelo para experimentos e predições.

O processo de modelagem de fenômenos espaciais dinâmicos, de acordo com Carneiro (2006), ocorre de forma iterativa e incremental, conforme ilustrado na Figura 2, e compreende as seguintes fases: (a) desenvolvimento da base de dados; (b) desenvolvimento do modelo; (c) calibração, verificação e validação do modelo; (d) execução e visualização do modelo e análise de relatórios; (e) projeção de cenários. As atividades de desenvolver modelos para sistemas ambientais e desenvolver *software*

baseado no modelo espiral, apresentado por Boehm (1988), são essencialmente similares, e envolvem as seguintes "macro-atividades": concepção, projeto, implementação e testes.

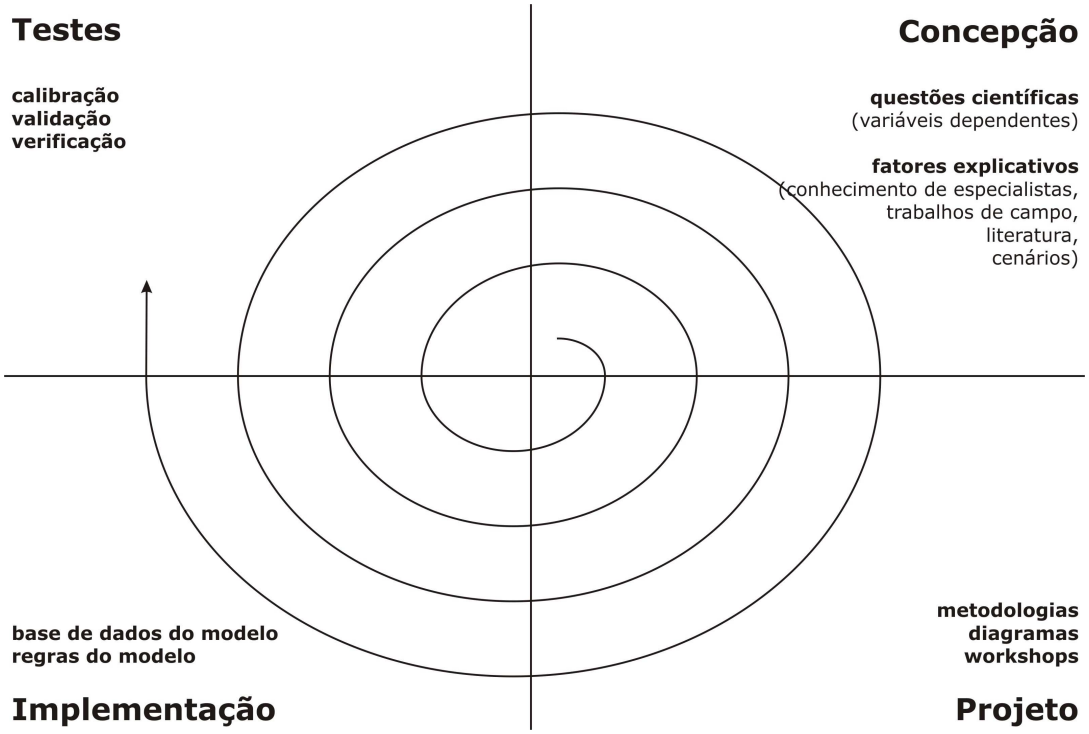


Figura 2 – Processo iterativo e incremental de desenvolvimento de modelos.
Fonte: Carneiro (2006)

De acordo com Hannon e Ruth (2001) e a partir das definições apresentadas nesta seção é possível perceber que a modelagem é um processo "interminável", onde o modelo é construído, revisado, comparado e alterado, e a cada ciclo é ampliada e melhorada a compreensão sobre a realidade observada. Apesar de em geral ser um processo iterativo e incremental, a construção de modelos pode apresentar particularidades e diferenças, por exemplo, em função da metodologia e plataforma de modelagem escolhidas.

No Capítulo 3 são apresentadas algumas plataformas de modelagem e simulação baseadas em diferentes teorias e que oferecem conseqüentemente diferentes funcionalidades a seus usuários, além de utilizar metodologias diferentes para a construção de modelos.

3 PLATAFORMAS DE MODELAGEM E SIMULAÇÃO

Neste capítulo são apresentadas as plataformas de modelagem *NetLogo*, *Swarm*, *Repast*, *Vensim* e *Dinamica EGO*.

3.1 Introdução

Na medida em que os processos do sistema terrestre a serem modelados se tornam mais complexos, devido à necessidade de representar múltiplas escalas, as diferentes interações entre os processos, a necessidade de considerar processos antrópicos e a maior disponibilidade de dados, faz-se necessária a utilização de diferentes técnicas e ferramentas. Isso é o que vem acontecendo atualmente, diversas técnicas de modelagem e plataformas de software têm sido utilizadas no desenvolvimento de modelos ambientais e nas simulações destes modelos. No entanto, de acordo com Soares-Filho et al. (2002), embora os modelos ambientais estejam se tornando cada vez mais complexos e híbridos para serem classificados definitivamente em uma única categoria, o paradigma dominante de modelagem, em geral, é baseado em indivíduos (*individual-based*), processos *(process-based*) ou autômatos celulares orientados ao espaço (*space-oriented cellular automata*).

Dentre as diversas opções de ferramentas de *software* baseadas nestes três paradigmas para o desenvolvimento e a simulação de modelos ambientais, são apresentadas a seguir as seguintes plataformas: *NetLogo* (WILENSKY, 1999), *Swarm* (SWARM, 1999) e *RePast* (ROAD, 2009), como exemplos de plataforma para modelagem baseada em indivíduos; *Vensim* (VENTANA, 2009) e *Dinamica-EGO* (SOARES-FILHO, 2009), como exemplos de plataformas para modelagem baseada em processos. Como exemplo de plataforma para modelagem baseada em autômatos celulares o *TerraME*, por servir de base para este trabalho, será apresentada com maiores detalhes no Capítulo 4.

3.2 *NetLogo*

O *NetLogo* (WILENSKY, 1999), projetado para fins de educação e pesquisa, é um ambiente para modelagem e simulação de fenômenos naturais e sociais através de uma linguagem de programação multi-agente (TISUE e WILENSKY, 2004).

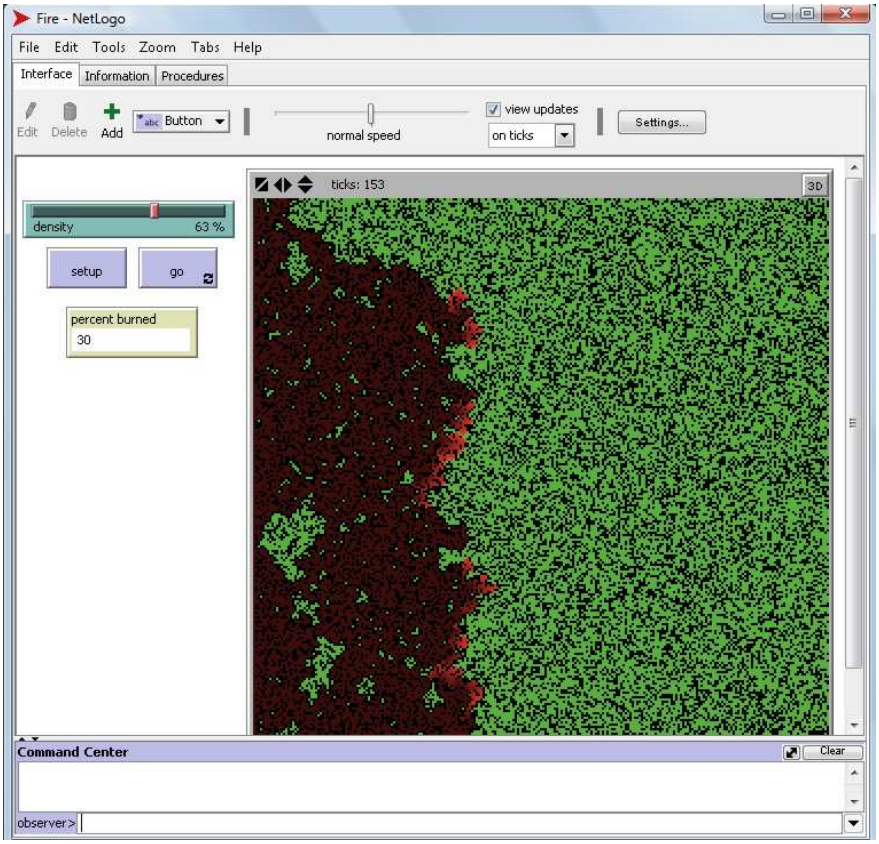
Desenvolvido em *Java* como uma aplicação *standalone*, é mantido pelo *Center for Connected Learning and Computer-Based Modeling*[2] e distribuído gratuitamente (*freeware*), o que permite aos usuários fazer o *download* do ambiente e construir modelos sem qualquer tipo de restrição. A documentação deste ambiente, bem como um conjunto de tutoriais e uma vasta coleção de modelos estão disponíveis aos usuários no *website* (http://ccl.northwestern.edu/netlogo) (TISUE e WILENSKY, 2004).

Originado a partir de uma combinação das linguagens *Logo* (PAPERT, 1980) e *StarLisp* (LASSER e OMOHUNDRO, 1986), seguindo a filosofia de facilidade de uso da linguagem *Logo* e herdando da linguagem *StarLisp* a concorrência e agentes múltiplos, o *NetLogo* possui seu *design* baseado no ambiente *StarLogoT* (WILENSKY, 1997b). Através do *NetLogo* é possível criar agentes móveis, chamados *turtles*, que se movem através de um *grid* de *patches*, os quais são também agentes programáveis. Os agentes podem interagir uns com os outros e executar múltiplas tarefas concorrentemente (TISUE e WILENSKY, 2004).

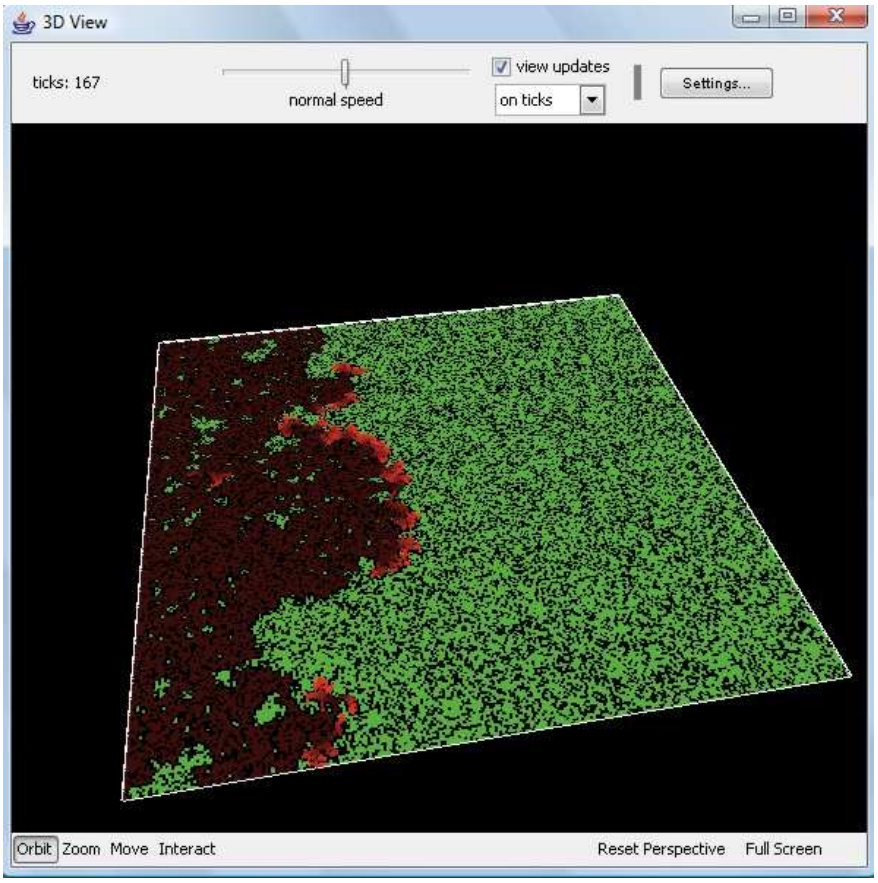
O *NetLogo* é completamente programável através de sua linguagem de programação de alto nível, de estrutura simples, derivada da linguagem *Logo*. O ambiente oferece uma série de facilidades e recursos, tais como: estrutura de linguagem simples; construção de agregados, redes e grafos de agentes; visualização 2D e 3D do modelo; controle de velocidade de simulação; monitores que permitem inspecionar e controlar os agentes. O ambiente é simples o suficiente para permitir que estudantes e professores possam facilmente executar simulações ou mesmo construir suas próprias simulações. E é avançado o suficiente para servir como uma poderosa ferramenta para pesquisadores de diversas áreas (WILENSKY, 1999).

A interface gráfica de usuário do *NetLogo* é voltada para a visualização e execução (simulação) do modelo, como ilustrado na Figura 3. Na Figura 3(a) pode-se observar a interface de interação com o usuário para a configuração e execução de um modelo de espalhamento de fogo (WILENSKY, 1997), contendo *widgets* (componentes gráficos de interface) que permitem iniciar e alterar a velocidade da simulação e configurar parâmetros do modelo; e na Figura 3(b) pode-se observar o resultado da execução do modelo em uma representação tridimensional.

[2] http://ccl.sesp.northwestern.edu/



(a)



(b)

Figura 3 – Interface de simulação do *NetLogo*: (a) configuração e execução da simulação, (b) visualização tridimensional da simulação.
Fonte: WILENSKY (1997)

A interface do *NetLogo* permite visualizar e editar o modelo diretamente a partir do seu código fonte, conforme ilustrado na Figura 4.

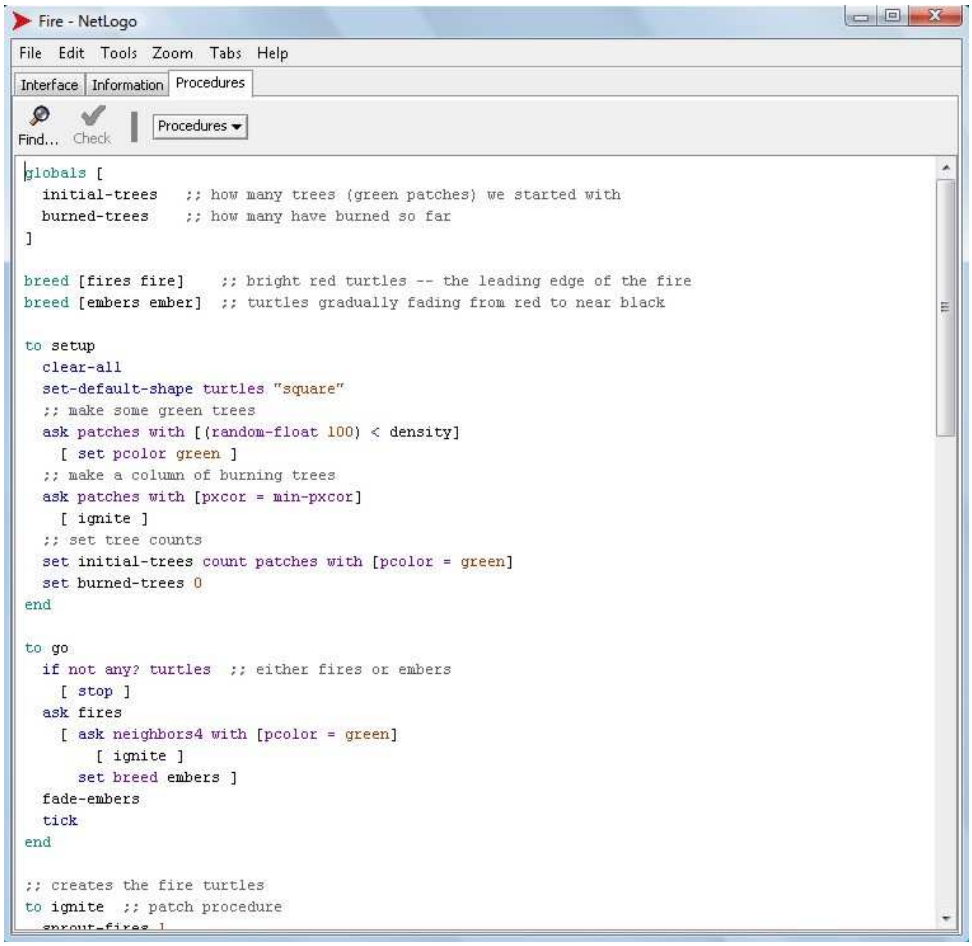


Figura 4 – Interface de visualização do modelo (código-fonte) do *NetLogo*.
Fonte: WILENSKY (1997)

O ambiente *NetLogo*, apesar de apresentar uma interface gráfica intuitiva, que permite aos usuários rapidamente serem capazes de executar e interagir com a execução de modelos previamente construídos, como o modelo de propagação de fogo, não oferece os mesmos recursos para a etapa de construção de modelos. O processo de construção de modelos no *NetLogo* é direcionado para a construção da interface de simulação do modelo, conforme ilustrado na Figura 5. Desta forma, o modelador é induzido a construir o modelo a partir da sua interface de execução (simulação), definindo por fim o comportamento associado a cada elemento da interface, usando diretamente a linguagem de programação *NetLogo*. A plataforma não oferece recursos e facilidades específicas para a construção do modelo propriamente dito, tendo o modelador que especificar diretamente através de uma linguagem de programação, ainda que de alto nível, as estruturas e regras que definem o comportamento do modelo.

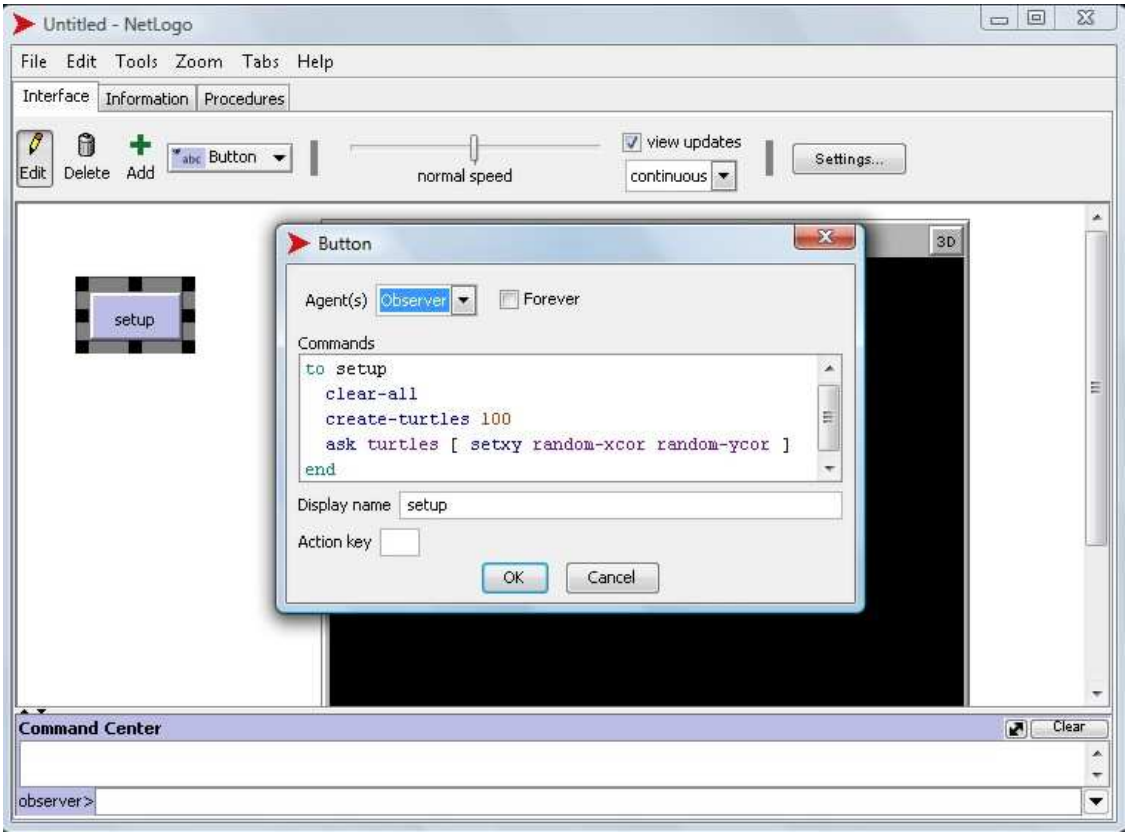


Figura 5 – Interface de modelagem do *NetLogo*.
Fonte: WILENSKY (1999)

O *NetLogo* vem sendo utilizado para a construção de modelos em diversas áreas do conhecimento, incluindo as ciências naturais, tais como a física e a biologia, e as ciências sociais, tais como a economia, a história e a psicologia. Wilensky (2001) apresenta dois exemplos de utilização do *NetLogo*: o modelo *Wolf/Sheep Predation*, um modelo presa/predador de lobos e carneiros; e o modelo *Gas-in-a-Box*, que simula uma "caixa" contendo moléculas de gás que colidem entre si e com a caixa e demonstrando o processo de perda de energia destas. Han et al. (2008) apresentam a utilização de um modelo de mudanças climáticas através do *NetLogo* para a educação ambiental. Filatova e van der Veen (2007) utilizam a modelagem baseada em agentes através do *NetLogo* para estudar a mudança de uso do solo em cidades costeiras da Holanda.

3.3 *Swarm*

O *Swarm* é um pacote de *software* para simulação multi-agentes de sistemas complexos. Originalmente desenvolvido no *Santa Fe Institute*[3], é atualmente mantido pelo *Swarm*

[3] http://www.santafe.edu/

Development Group[4] (SDG). Foi uma das primeiras bibliotecas para modelagem baseada em agentes e está disponível sob os termos de licenciamento GNU (SWARM, 1999; MINAR et al., 1996).

O *Swarm* fornece, a partir de um conjunto de bibliotecas, um *framework* conceitual para projetar, construir e conduzir experimentos em modelagem baseada em agentes. A plataforma fornece bibliotecas orientadas a objetos de componentes reutilizáveis para construção e análise de modelos, exibição de controle de experimentos. Fornece ainda bibliotecas para gerenciar agentes, estruturas espaciais para seu ambiente, suas atividades e a agregação destas atividades e a análise dos resultados. Por meio da biblioteca os usuários podem construir modelos e simulações baseadas em agentes utilizando as linguagens de programação *Objective-C* (uma extensão da linguagem C) ou *Java* (SWARM, 1999; MINAR et al., 1996).

A unidade básica de simulação do *framework* do *Swarm* é chamado de *swarm*. O *swarm* representa um modelo completo, composto por agentes e por um escalonador de eventos (representação do tempo). O *Swarm* suporta a modelagem hierárquica, onde agentes podem ser compostos por *swarm*s (enxames) de outros agentes em estruturas aninhadas, sendo o comportamento do agente de mais alto nível definido pelos fenômenos emergentes dos agentes interiores ao seu *swarm* (SWARM, 1999; MINAR et al., 1996).

A plataforma possui uma vasta comunidade de usuários e desenvolvedores, que compartilham idéias, *softwares* e experiências. A comunidade conta ainda com a *SwarmFest*[5], uma conferência anual internacional sobre modelagem baseada em agentes, que não se restringe apenas aos usuários da plataforma (SWARM, 1999).

Esta plataforma apresenta um elevado grau de dificuldade de aprendizagem, sendo necessário ao usuário possuir experiência em *Java* ou *Objective C*, ser familiarizado com a metodologia de orientação a objetos e ser capaz de aprender os códigos da biblioteca do *Swarm* (RODRIGUEZ-AGUILAR et al., 2001).

A plataforma não oferece recursos específicos para auxiliar os usuários na etapa de modelagem, ou seja, os usuários devem construir seus modelos diretamente a partir da linguagem de programação *Java ou Objective C*.

[4] http://www.swarm.org/index.php/Swarm_Development_Group
[5] http://www.swarm.org/index.php/Swarm:_SwarmFest

Exemplos de utilização desta plataforma podem ser encontrados em: Pitt et al. (2003) e Conner et al. (2008) – modelagem de dinâmica de populações; Luna e Stefansson, (2000) e Luna e Perrone (2001). – na área de economia e finanças.

3.4 *Repast*

O *Recursive Porous Agent Simulation Toolkit* (*Repast*), criado na Universidade de Chicago e atualmente mantido pela *Repast Organization for Architecture and Development*[6] (ROAD), é um *toolkit* (conjunto de ferramentas) gratuito e de código aberto (*open source*) para simulação de modelos baseados em agentes. O *Repast* utiliza muito dos conceitos do *Swarm*, fornecendo uma biblioteca de objetos para criação, execução, visualização e coleta de dados de simulações baseadas em agentes, se diferenciando pela implementação em múltiplas linguagens e por embutir funcionalidades adaptativas como algoritmos genéticos e regressão. Porém, ao contrário do *Swarm*, é disponibilizado em implementações "nativas" para as plataformas *Java* e *Microsoft .NET*. O foco do *Repast* é a modelagem de comportamento social, mas não se limita a isso (REPAST3, 2009; ROAD, 2009; COLLIER, 2002; COLLIER et al. 2003).

O *Repast 3* pode ser visto como uma especificação para serviços ou funções para modelagem baseada em agentes, sendo atualmente disponibilizado em três implementações: *Repast for Java* (*Repast J*); *Repast for the Microsoft .Net Framework* (*Repast .Net*); e *Repast for Phyton Scripting* (*Repast Py*). As implementações se diferem com relação à plataforma base e à linguagem para desenvolvimento dos modelos, sendo todas elas constituídas do mesmo núcleo de serviços (*core services*) que constitui o sistema *Repast* (REPAST3, 2009; ROAD, 2009). Maiores detalhes podem ser encontrados na documentação Repast3 (2009), ROAD (2009) e em North et al. (2006).

A plataforma oferece diversos recursos e serviços, tais como: vários *templates* e exemplos de agentes; é completamente orientada a objetos; um escalonador de eventos discretos que suporta tanto operações seqüenciais quanto paralelas; ferramentas gráficas e de registro para os resultados de simulações; um *framework* para simulações de Monte Carlo; permite acessar e modificar as propriedades e equações de comportamento dos agentes e as propriedades do modelo em tempo de execução; bibliotecas para algoritmos genéticos, redes neurais, geração de números aleatórios; ferramentas de apoio a

[6] http://repast.sourceforge.net/

modelagem de redes sociais; suporte à integração com sistemas de informações geográficas (SIG); é completamente implementado em várias linguagens de programação, incluindo Java e C#; os modelos podem ser desenvolvidos em diversas linguagens incluindo *Java*, *C#*, *Managed C++*, *Visual Basic .Net*, *Managed Lisp*, *Managed Prolog*, e *Python Scripting*; está disponível para diversas plataformas, tanto para computadores pessoais quanto para *clusters* de computação científica em larga escala, incluindo *Windows*, *Mac OS* e *Linux* (REPAST3, 2009; ROAD, 2009).

A plataforma compreende uma simulação como uma máquina de estados, em que cada estado é constituído pelo conjunto dos estados de seus componentes. Estes componentes podem ser divididos em: infraestrutura (os vários mecanismos que executam a simulação, exibem e coletam dados etc.) e representação (o que o modelador constrói, o modelo de simulação propriamente dito). Qualquer mudança nos estados dos componentes de infraestrutura e dos componentes de representação ocorre através de um objeto *Schedule* (escalonador), ou seja, em suma o *Repast* permite construir uma simulação como uma máquina de estados em que todas as mudanças ocorrem através de um escalonador (REPAST3, 2009; ROAD, 2009; COLLIER et al. 2003).

O *Repast Simphony* (*Repast S*) estende as funcionalidades do *Repast* fornecendo uma nova abordagem para o desenvolvimento e execução de simulações (HOWE et al., 2006). O *Repast S Runtime*, uma extensão *Java* pura, foi projetado para fornecer novas funcionalidades e recursos para a família *Repast*, incluindo características/funcionalidades avançadas para armazenamento, visualização e ativação comportamental de agentes, bem como novas facilidades para análise e apresentação de dados, conforme ilustrado na Figura 6 (NORTH et al., 2005a). O ambiente de desenvolvimento do *Repast S* (*Repast S Development Environment*) inclui características/funcionalidades para a especificação comportamental dos agentes e construção de modelos dinâmicos. A plataforma *Repast S* não substitui as ferramentas existentes no *Repast*, mas as complementa (NORTH et al., 2005b).

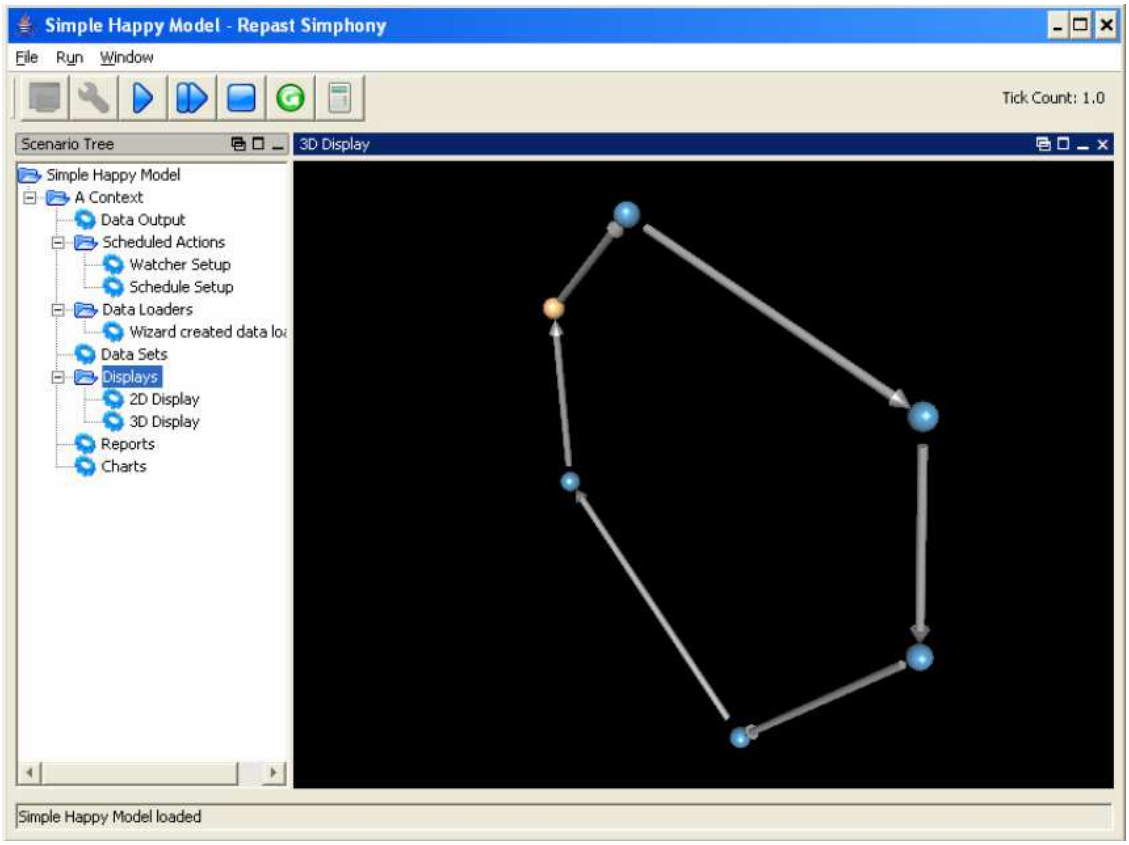


Figura 6 – Interface de simulação do *Repast S*.
Fonte: North et al. (2005a)

O *Repast S*, para fazer a associação e ligação entre as diversas peças dos modelos, utiliza dois tipos principais de configuração de modelos e simulação: descritor do modelo (*model descriptor*) e descritor do cenário (*scenario descriptor*). O descritor do modelo define o que "pode estar" (*what can be*) em um modelo, como os tipos permitidos de agentes, as relações permitidas entre os agentes e as informações a serem observadas. O descritor do cenário define o que "realmente é" (*actually is*) o modelo, tais como fontes de dados de agentes, visualizações e *logging*. Os descritores do modelo devem ser criados em tempo de desenvolvimento do modelo, enquanto é esperado que os descritores do cenário sejam criados em tempo de execução. O *Repast S Development Environment* fornece tanto um *wizard*[7] para a criação e um editor "apontar e clicar" (*point-and-click editor*) para a modificação dos descritores do modelo, e o *Repast S Runtime Environment* inclui um painel "apontar e clicar" (*point-and-click panel*) para a criação e manutenção dos descritores do cenário. Desta forma, o *Repast S* permite aos usuários definir diversos aspectos do modelo como a especificação de

[7] *Wizard* é um padrão de projeto de software utilizado em interfaces gráficas do usuário para prover um meio simples de realizar tarefas através de uma seqüência de passos.

configurações e comportamento dos agentes através da interação com *wizards* e editores de diagramas, conforme ilustrado na Figura 7 (NORTH et al., 2005b).

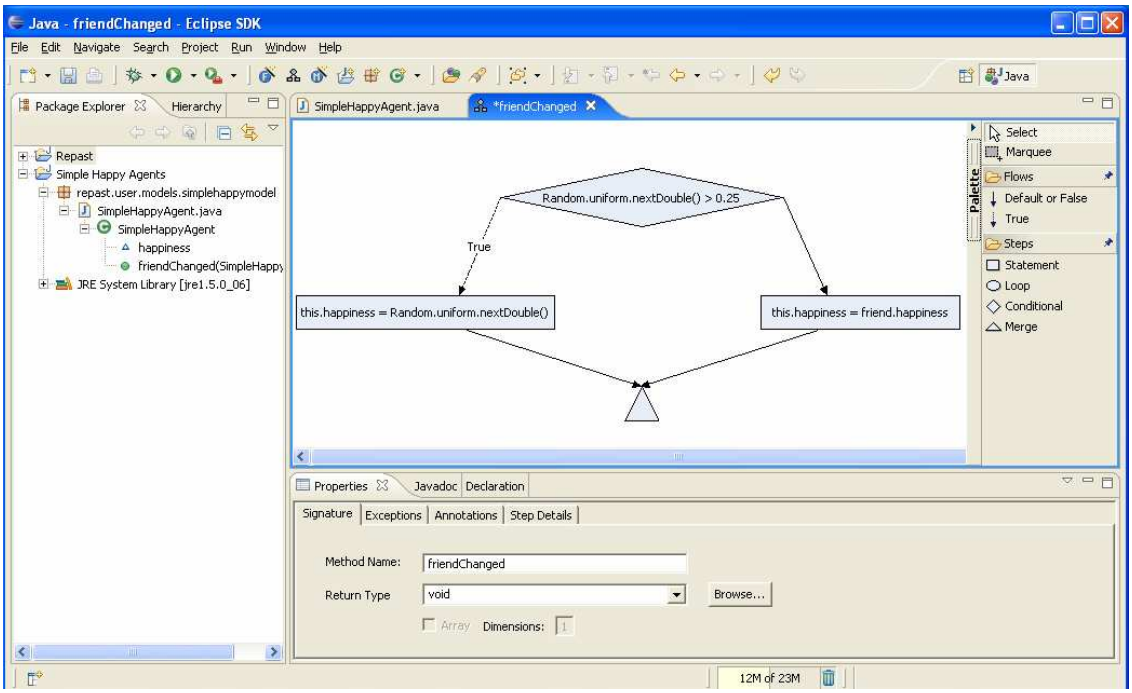


Figura 7 – Interface de modelagem do *Repast S*.
Fonte: North et al., (2005b)

Enquanto as versões anteriores exigiam dos usuários a configuração de um ambiente de desenvolvimento, o *Repast S* fornece um ambiente de desenvolvimento integrado (IDE) pré-configurado baseado na plataforma Eclipse (http://www.eclipse.org/), que oferece diversas facilidades para a construção e execução de modelos e não exige experiência prévia de programação para se construir um modelo (NORTH et al., 2007). Tatara et al. (2006) e North et al. (2007) apresentam tutoriais introduzindo como desenvolver modelos utilizando a plataforma.

Conforme apresentado em North et al. (2006), o *Repast* vem sendo utilizado em diversas áreas para modelar e simular diversos processos, como por exemplo para modelar: sistemas de produção e consumo; o desenvolvimento de padrões sociais e comportamentos como a emergência de normas sociais e leis, formação de movimentos nacionalistas. Crooks (2007) apresenta a utilização da plataforma no desenvolvimento de modelos para simulação de sistemas espaciais.

3.5 *Vensim*

O *Vensim* é uma plataforma de modelagem que permite idealizar, documentar, simular, analisar e otimizar modelos para sistemas dinâmicos. Desenvolvida pela *Ventana Systems Inc*., foi projetada para tornar mais fácil o aprendizado da dinâmica de sistemas, fornecendo um modo simples e flexível de construir modelos a partir de diagramas causais ou diagramas de fluxo. A plataforma é distribuída em opções de licença de uso gratuita e comercial, sendo oferecida gratuitamente a configuração Vensim PLE (*for Personal Learning Edition*) para uso educacional (VENTANA, 2009; VENSIM, 2009). A plataforma não oferece recursos para a construção de modelos espacialmente explícitos, sendo necessária neste caso a utilização e integração com outras ferramentas.

Através da conexão de palavras e setas, as relações entre as variáveis do sistema são inseridas e armazenadas como conexões de causalidade. É possível analisar o modelo durante todo o processo de construção, observando as causas e usos das variáveis e os *feedbacks* envolvendo as variáveis. Ao se construir um modelo que pode ser simulado, a plataforma permite explorar completamente o comportamento do modelo (VENTANA, 2009; VENSIM, 2009).

A plataforma oferece uma interface gráfica a partir da qual o usuário pode construir e interagir com a execução do modelo, conforme ilustrado na Figura 8 (para o modelo presa/predador). O *Vensim* possui componentes gráficos para representar os elementos necessários para construção de modelos de sistemas dinâmicos, tais como: estoques, variáveis, fluxos, taxas, equações, entrada (*input*) e saída (*output*) de dados; e ferramentas de análise, tais como: “árvores causais” (*causes tree*), geração de tabelas e gráficos (VENTANA, 2009; VENSIM, 2009).

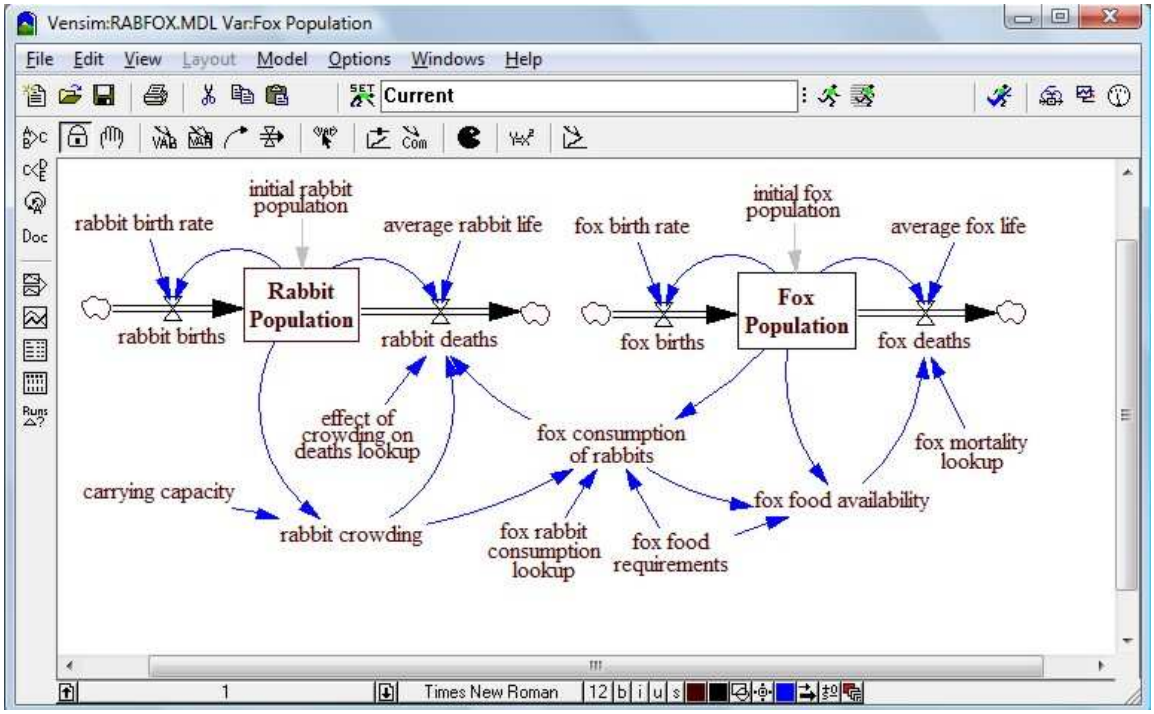


Figura 8 – Interface de modelagem e simulação do *Vensim*.
Fonte: Vensim (2009)

A plataforma vem sendo utilizada na construção de modelos de diversos fenômenos ambientais, tais como sucessão de vegetação em um ecossistema (MARTINEAU e SAUGIER, 2007); ondas de cheia em canais (SIMÕES et al., 2009); desmatamento e emissão de gases que contribuem para o efeito estufa (FEARNSIDE et al., 2009); gestão sustentável de uso da terra (YU et al., 2003); comportamento complexo de sistemas hidrológicos (KAHN et al., 2009).

3.6 *Dinamica* EGO

O *Dinamica* é uma plataforma para a construção e simulação de modelos de dinâmica de paisagem espacialmente explícitos, baseada em autômatos celulares e regras/funções de transição que permite, a partir de probabilidades de transição, identificar a dinâmica de formação e evolução de padrões espaciais do fenômeno – objeto de estudo. O *software Dinamica*, originalmente implementado na linguagem de programação C++, para sistemas Windows 32 bits, disponibilizava aos usuários uma interface gráfica a partir da qual os parâmetros do modelo podiam ser configurados (SOARES-FILHO et al., 2002; SOARES-FILHO, 2009).

A plataforma, desenvolvida e mantida pelo Centro de Sensoriamento Remoto (CSR), do Instituto de Geociências da Universidade Federal de Minas Gerais, é atualmente

disponibilizada gratuitamente para *download* em sua nova versão, denominada *Dinamica* EGO (do inglês, *Environment for Geoprocessing Objects*). Nesta nova versão o *Dinamica* foi totalmente reescrito, sendo seu núcleo, que é responsável pela criação e execução dos modelos, implementado em C++, enquanto a interface gráfica para modelagem foi desenvolvida em *Java*.

A construção de modelos no ambiente de modelagem *Dinamica EGO* pode ser feita de forma visual a partir da interação do usuário com a interface gráfica do ambiente, através da criação e conexão de componentes (operadores) denominados *functores*. Conforme Rodrigues et al. (2007), um *functor* pode ser entendido como um processo que atua sobre um conjunto de dados de entrada sobre o qual é aplicado um número finito de operações, produzindo como saída um novo conjunto de dados. A plataforma oferece *functores* para diversas tarefas e funcionalidades, tais como operadores de álgebra de mapas, operadores de análise espacial, métodos de calibração e validação. Também são disponibilizados operadores de grupo, denominados *containers*, os quais, ainda segundo Rodrigues et al. (2007), agrupam e determinam um comportamento para o conjunto de operadores nele contidos. São exemplos de *containers*: *Repeat*, executa iterativamente o sub-modelo nele contido; *Block*, agrupa *functors*; e *Region*, permite determinar uma região especifica do mapa a ser "afetada" pelas operações. Os *functores* e *containers* são interligados por meio de *ports* (portos), que permitem definir o conjunto e tipos de dados de entrada e saída de cada um deles.

Os modelos são construídos graficamente por meio do procedimento de arrastar e conectar *functores* e *containers*, conforme ilustrado na Figura 9, formando um diagrama que permite determinar e visualizar a estrutura seqüencial de execução e o fluxo de dados. Os modelos criados graficamente são armazenados de forma textual em *scripts* (SOARES-FILHO, 2009; RODRIGUES et al., 2007).

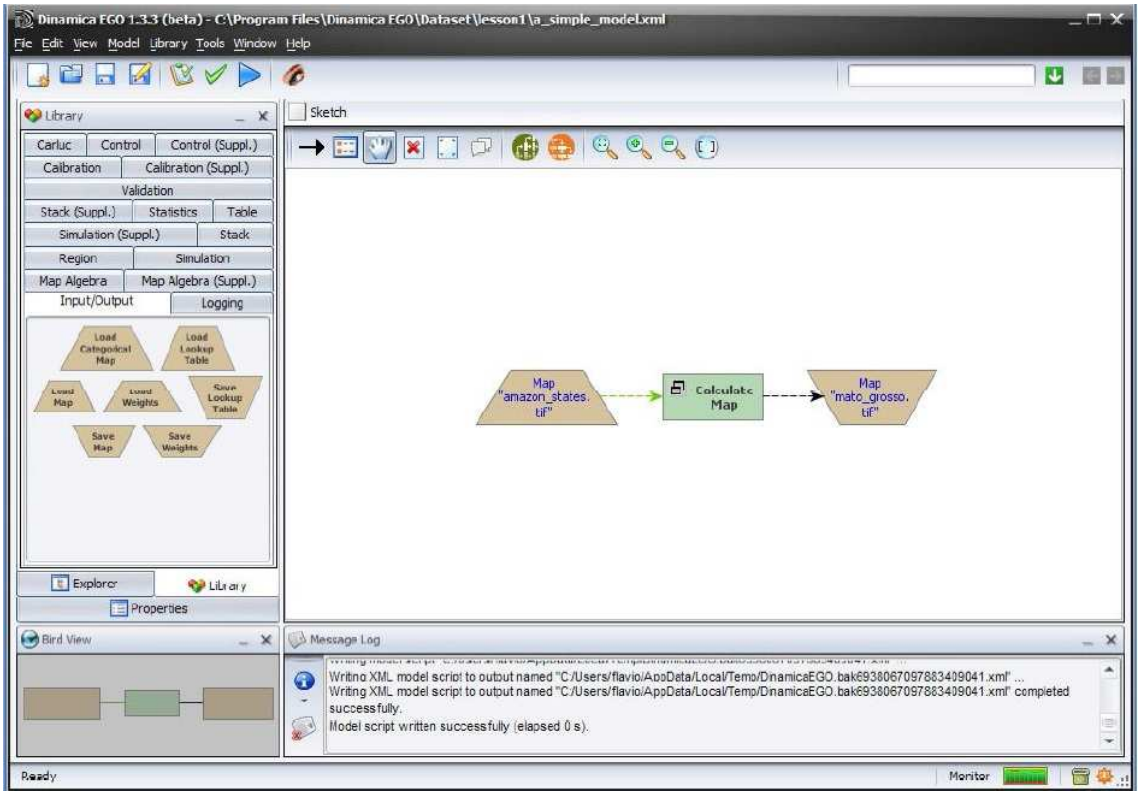


Figura 9 – Interface de modelagem e simulação do *Dinamica* EGO.
Fonte: Soares-Filho et al. (2009)

O *Dinamica EGO* vem sendo usado em diversos estudos de fenômenos ambientais, tais como: modelagem de expansão urbana e dinâmica intra-urbana (ALMEIDA et al., 2003; GODOY e SOARES-FILHO, 2007); mudança de uso e de cobertura da terra (*Land Use and Land Cover Change – LUCC*) (SOARES-FILHO et al., 2002; SOARES-FILHO et al., 2004); propagação de fogo (SILVESTRINI et al., 2009).

4 A PLATAFORMA DE MODELAGEM *TerraME*

Este capítulo apresenta a plataforma *TerraME*, um ambiente de modelagem e simulação de sistemas espaciais dinâmicos que oferece estruturas de dados e serviços para a construção e simulação de modelos ambientais, o modelo conceitual implementado pela plataforma, e o processo de construção de modelos através da linguagem de programação *TerraME*.

4.1 A plataforma *TerraME*

O *TerraME* (*TerraLib Modelling Environment*) (CARNEIRO, 2006) é um componente da família de soluções *TerraLib* (CÂMARA et al., 2000) para a implementação e simulação de modelos ambientais que envolvam a representação explícita do espaço. Desenvolvido a partir do trabalho de doutorado de Carneiro (2006), no Instituto Nacional de Pesquisas Espaciais (INPE), concluído em 2006, é disponibilizado para *download* e utilização a partir do *website* (http://www.terrame.org/). Em contínuo desenvolvimento desde então, atualmente é mantido pelo *TerraLAB* (Laboratório para Modelagem e Simulação de Sistemas Terrestres), uma parceria entre o INPE e a Universidade Federal de Ouro Preto (UFOP), que possui como missão o projeto e desenvolvimento de uma plataforma livre de modelagem e simulação de modelos ambientais espacialmente explícitos, a Plataforma *TerraME* (CARNEIRO et al., 2009).

O *TerraME* provê mecanismos que permitem a fácil representação e a eficiente simulação de modelos espaciais dinâmicos integrados a um sistema de informações geográficas. Os componentes de sua arquitetura de *software*, a ser apresentada a seguir, oferecem serviços específicos a usuários com diferentes níveis de conhecimento em algoritmos e técnicas de programação. Usuários experientes podem implementar modelos utilizando diretamente o *framework* de modelagem *TerraME* através da linguagem de programação *C++*, enquanto aqueles que possuem apenas o conhecimento básico sobre algoritmos e modelagem computacional podem utilizar a linguagem de programação de alto nível *TerraME Modeling Language* – uma extensão da linguagem de programação *LUA* (IERUSALIMSCHY et al., 1996), que permite a fácil escrita, leitura e alteração dos modelos (CARNEIRO, 2006).

4.2 Arquitetura do *TerraME*

O *TerraME* foi construído baseado na arquitetura em camadas, onde as camadas inferiores fornecem funcionalidades sobre as quais as camadas superiores são implementadas, conforme mostrado na Figura 10.

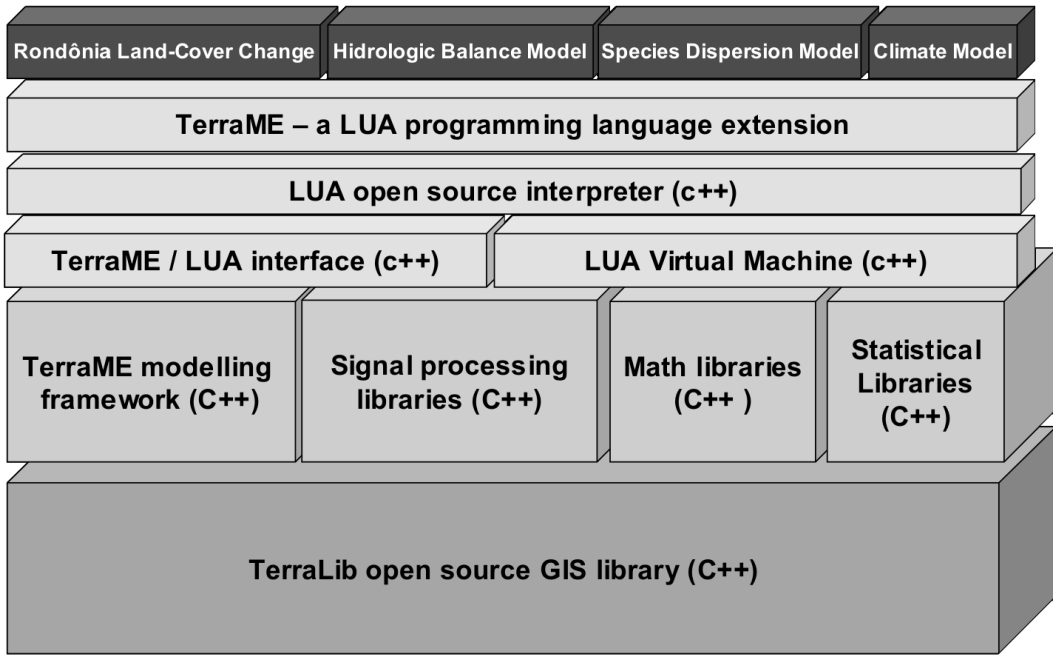


Figura 10 – Arquitetura da plataforma *TerraME*.
Fonte: Carneiro (2006)

Conforme definido por Carneiro (2006), na camada inferior, a biblioteca *TerraLib* fornece serviços para gerenciamento e análise de dados espaço-temporais. Na segunda camada, o *TerraME Modeling Framework* fornece serviços para a simulação, calibração e validação de modelos que podem ser utilizados através da linguagem *C++*. Nesta camada, o modelador tem acesso a um poderoso conjunto de serviços para modelagem e simulação (por exemplo, nesse nível não há restrição quanto ao tipo de representação espacial utilizada para modelar objetos espaciais – as células podem ser *pixels* em uma imagem, triângulos em um modelo digital de elevação, ou pontos, linhas e polígonos em um mapa vetorial). Contudo, sua interface de programação de aplicativos (ou API, do inglês *Application Programming Interface*) apresenta uma sintaxe complexa e, por isso, de difícil utilização para usuários que não possuem sólidos conhecimentos em técnicas de programação e na linguagem *C++*. A camada seguinte é aquela formada pelo interpretador e pelo ambiente de execução da linguagem de modelagem *TerraME*, que estende a linguagem de programação *LUA* pela inclusão de novos tipos de dados,

especialmente projetados para a modelagem espacial dinâmica, e de serviços para simulação e avaliação de modelos. Sobre as demais, está a camada de aplicação, composta por modelos ambientais desenvolvidos pelos usuários da arquitetura. Desta forma, a construção de modelos na plataforma pode ser feita a partir das linguagens de programação C++ e *TerraME*.

Neste trabalho é construída uma nova camada para a arquitetura, entre a camada do interpretador *TerraME* e a da aplicação, utilizando um nível maior de abstração, procurando tornar mais fácil a utilização do ambiente para usuários não experientes em programação. Maiores detalhes sobre esta camada são apresentados no Capítulo 5.

A linguagem de programação *TerraME* é uma extensão da linguagem LUA, sendo implementados tipos de dados específicos para a modelagem ambiental conforme apresentado a seguir.

4.3 A linguagem de modelagem *TerraME*

A linguagem de programação *TerraME Modeling Language*, uma extensão da linguagem LUA, permite a representação de modelos espaciais dinâmicos a partir dos seguintes tipos de dados:

- *Environment*: utilizado para representar o conceito de ambiente (escala) e permitir o desenvolvimento de modelos que considerem múltiplas escalas;

- *CellularSpace*, *Cell*, *Neighborhood*: utilizados para representar o espaço, suas propriedades e relações topológicas;

- *Agent*, *Automaton*, *State*, *Jump*, *Flow* e *Trajectory*: utilizados para representar o comportamento de sistemas – processos ou atores que dinamicamente alteram as propriedades do espaço e interagem entre si;

- *Timer*, *Event* e *Message*: utilizados para representar o tempo e definir o momento e a ordem na qual os eventos serão executados.

A seguir é apresentado o processo de modelagem utilizando a linguagem *TerraME*.

4.4 Modelagem utilizando o *TerraME*

O processo de desenvolvimento de modelos na plataforma *TerraME* por meio da linguagem de programação *TerraME* é descrito a seguir, sendo necessários um editor de textos, o interpretador *TerraME* e um SIG associado a uma base de dados, conforme ilustrado na Figura 11. O modelador pode implementar o modelo através da linguagem *TerraME*, utilizando por exemplo como editor de textos a plataforma *Eclipse* (http://www.eclipse.org/), juntamente com o *plug-in* para a linguagem LUA LuaEclipse (http://luaeclipse.luaforge.net/), que oferece funcionalidades tais como *syntax highlight* e identificação de erros no código. Uma vez implementado, o modelo poderá então ser executado através do interpretador *TerraME*, que por sua vez poderá fazer uso de uma base de dados geográficos por meio da biblioteca *TerraLib*, para ler e armazenar informações, e por fim os resultados poderão ser visualizados em um SIG, como por exemplo, o *TerraView*.

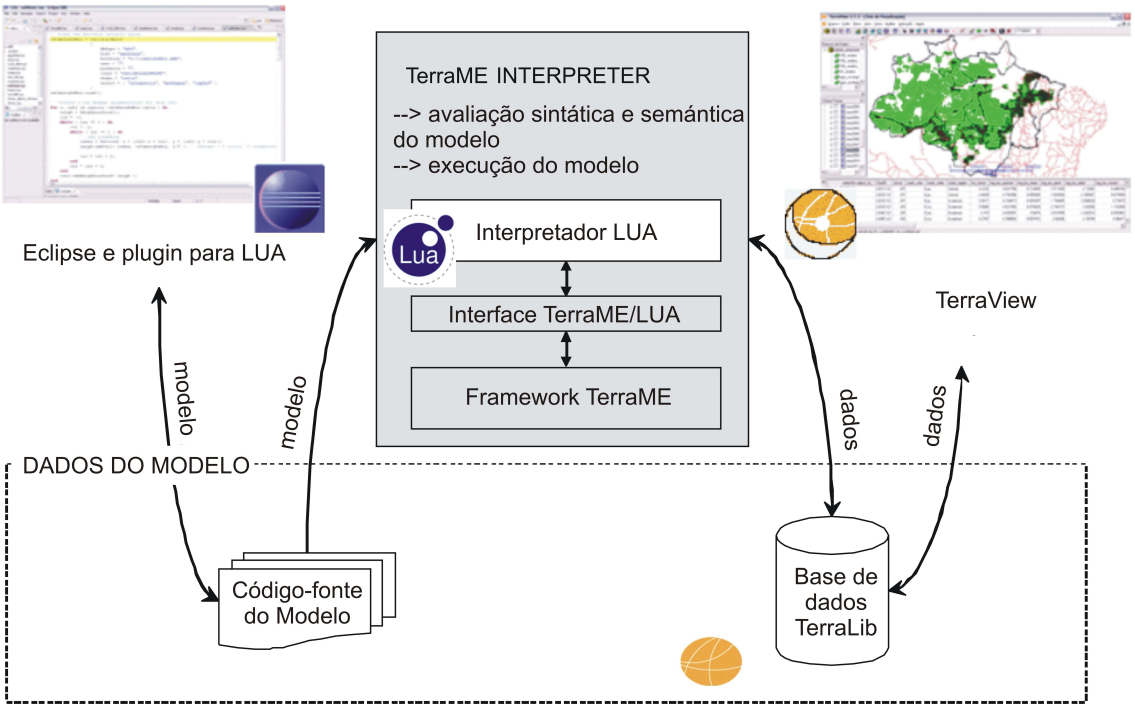


Figura 11 – Ambiente de desenvolvimento da plataforma *TerraME*.
Fonte: Carneiro (2006)

A representação de um fenômeno na forma de um modelo na plataforma *TerraME* é feita a partir da representação da dinâmica do comportamento deste fenômeno no tempo e no espaço. Conforme a concepção conceitual implementada pelo *TerraME*, um ambiente na Terra, de acordo com o conceito de escala (GIBSON et al., 2000), pode ser

representado por um ambiente sintético (virtual) onde entidades analíticas (regras) alteram as propriedades do espaço no tempo, como ilustrado na Figura 12.

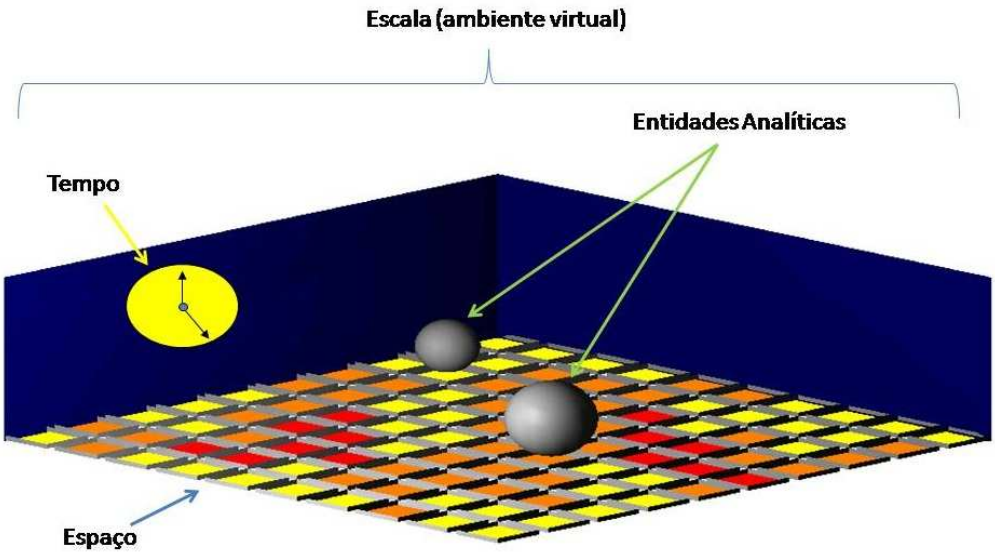


Figura 12 – O conceito de escala implementado como um ambiente em *TerraME*.
Fonte: adaptado de Carneiro (2006)

O *TerraME* oferece estruturas de dados e funcionalidades para a representação dos modelos espacial, temporal e comportamental, além de permitir a construção de modelos em múltiplas escalas, conforme será apresentado a seguir.

4.4.1 *TerraME* e modelagem multi-escala

Um dos requisitos que ferramentas para modelagem e simulação de fenômenos do sistema terrestre devem implementar é permitir a representação de múltiplas escalas. Segundo Gibson et al. (2000) escala é um conceito genérico que inclui as dimensões espaciais, temporais e comportamentais usadas para mensurar um fenômeno. O *TerraME* implementa este conceito através da definição do tipo de dados *Environment* (Seção 4.3), que representa um ambiente constituído de um modelo comportamental (constituído pelas as regras que definem atores e processos de mudança em um ambiente), um modelo espacial (constituído pela representação do espaço e de seus atributos) e um modelo temporal (constituído pelos eventos que definem a dinâmica do ambiente, ou seja, sua evolução ao longo do tempo). Desta forma, a construção de modelos de múltiplas escalas é possível a partir da composição de escalas aninhadas, como ilustrado na Figura 13. Assim, é possível criar representações espaciais, temporais e comportamentais em múltiplas escalas.

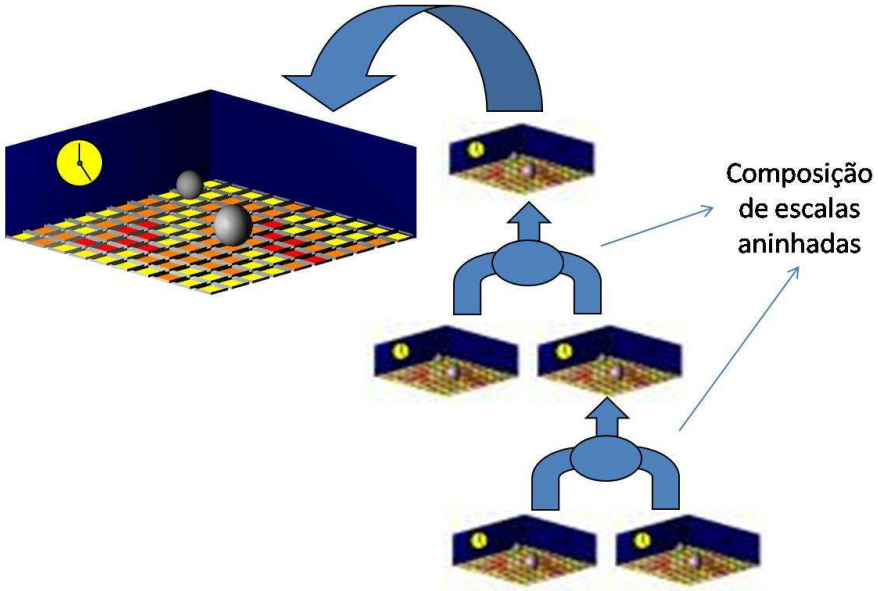


Figura 13 – Modelagem de múltiplas escalas na plataforma *TerraME*.
Fonte: adaptado de Carneiro (2006)

4.4.2 *TerraME* e o modelo espacial

A plataforma *TerraME* possui serviços que permitem a integração com bases de dados geográficos, sendo possível a leitura e armazenamento de informações a serem utilizadas no modelo, conforme ilustrado na **Erro! Fonte de referência não encontrada.**. Uma vez construída uma base de dados fazendo uso de um SIG, é possível construir o modelo espacial (constituído de propriedades do espaço e relações de vizinhança) a partir de um conjunto de planos de informação (*layers*) armazenados, por exemplo, como grades de células.

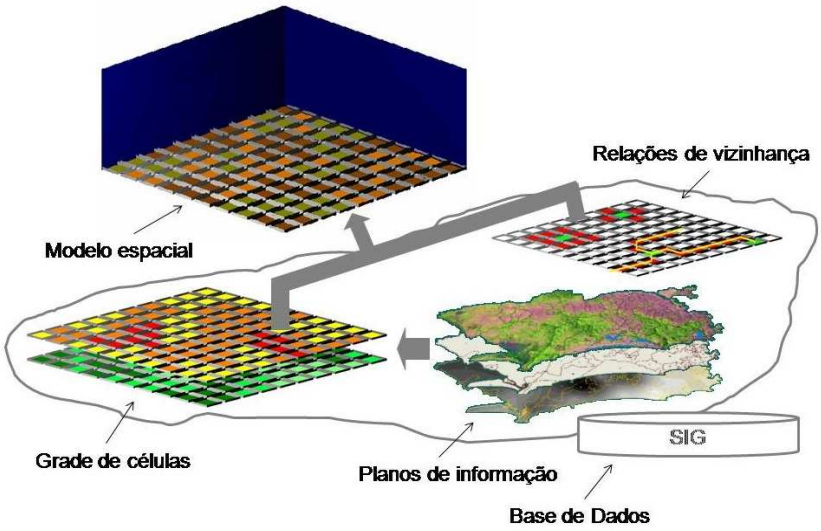


Figura 14 – Modelo espacial integrado a um SIG.
Fonte: adaptado de Carneiro (2006)

Na Figura 15 é apresentado um trecho de código onde o modelo espacial é carregado a partir de uma base de dados especificada. Através do tipo de dados *CellularSpace* (Seção 4.3) é possível representar o modelo espacial, que é constituído de diversas células – tipo de dados *Cell* (Seção 4.3). Vizinhanças podem ser definidas e especificadas através do tipo de dados *Neighboorhood* (Seção 4.3).

```
-- Loads a TerraLib cellular space
csCabecaDeBoi = CellularSpace {
    dbType = "ADO",
    host = "localhost",
    database = "CabecaDeBoi.mdb",
    user = "",
    password = "",
    layer = "cellsLobo90x90",
    theme = "cells",
    select = { "heigh", "capInf" },
    where = "mask <> 'noData'"
}
csCabecaDeBoi:load();
```

Figura 15 – Definindo um modelo espacial na linguagem de modelagem TerraME.
Fonte: Carneiro e Câmara (2009)

4.4.3 *TerraME* e o modelo temporal

A execução de um modelo (simulação) no *TerraME* ocorre a partir do modelo temporal especificado, conforme ilustrado na **Erro! Fonte de referência não encontrada.**, por meio de um escalonador de eventos. Desta forma, o modelo temporal é constituído de pares de eventos e mensagens, que serão por sua vez alocados de forma seqüencial no escalonador conforme o tempo, período e prioridade, definidos para cada evento (Carneiro, 2006).

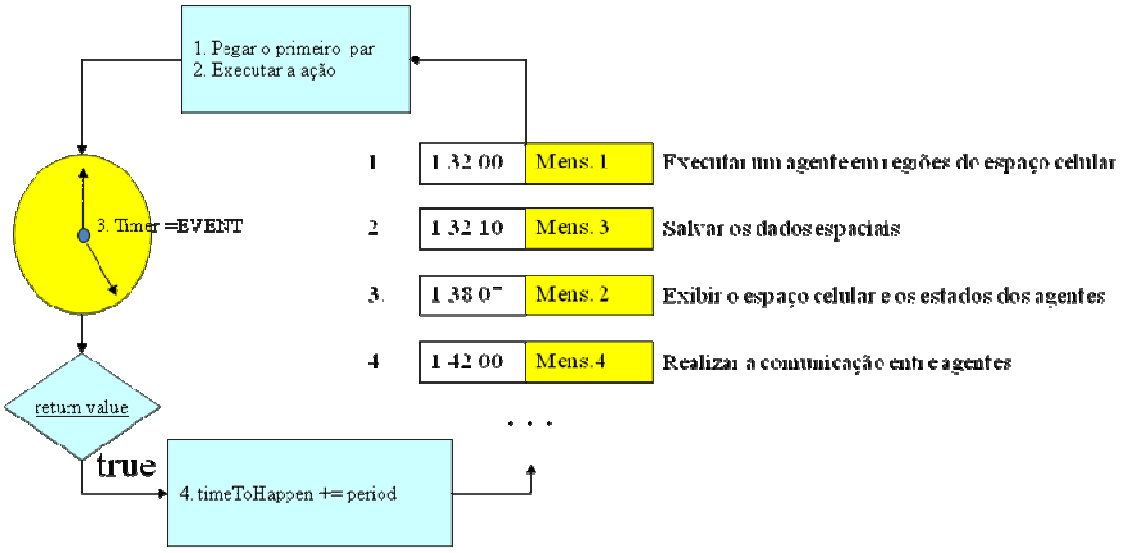


Figura 16 – Modelo temporal em *TerraME* implementado por meio de um escalonador de eventos.
Fonte: Carneiro e Câmara (2009)

Uma vez definidas pelo modelador as diversas escalas que irão constituir o modelo (Figura 17(a)), o modelo temporal é então construído pela plataforma a partir do modelo temporal de cada uma das escalas de forma aninhada conforme estas (Figura 17(b)). Desta forma, é possível construir modelos temporais de múltiplas escalas a partir da composição aninhada destas.

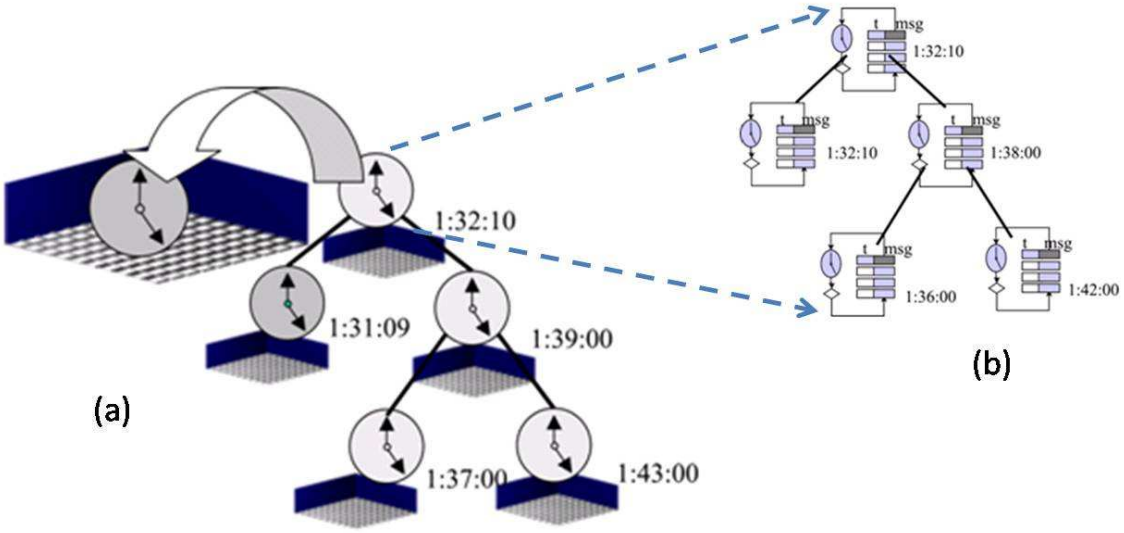


Figura 17 – Modelo temporal em múltiplas escalas: (a) composição de escalas; (b) composição do modelo temporal de cada escala.
Fonte: Carneiro (2006)

O modelo temporal, ilustrado pelos trechos de código da Figura 18, é construído a partir do tipo de dados *Timer*, que por sua vez é constituído de uma série de pares (*Pair*)

evento/mensagem (*Event/Message*). Para cada evento são definidos os seguintes atributos: o tempo em que o mesmo irá ocorrer (*time*), o período (*period*) e sua prioridade (*priority*). A mensagem por sua vez define, por meio de uma função, quais ações serão executadas pelo evento.

```
time = Timer{
      Pair{
        Event{ ... },
        Message{ ... }
      },
      Pair{
        Event{ ... },
        Message{ ... }
      },
      ...
      Pair{
        Event{ ... },
        Message{ ... }
      }
}
```

```
Event{ time = 1985, period = 1, priority = -1 }
```

```
Message{
      function( event )
        print(event:getTime());
        rain:execute( event );
        print("Rained");
        return false;
      end
}
```

Figura 18 – Definindo um modelo temporal na linguagem de modelagem TerraME.
Fonte: Carneiro (2006)

4.4.4 *TerraME* e o modelo comportamental

Modelos comportamentais no *TerraME* podem ser definidos através de agentes (tipo de dados *Agent*) ou autômatos (tipo de dados *Automaton*). Em um agente todas as células compartilham o mesmo estado, ou seja, temos um estado global, enquanto nos autômatos, cada célula possui seu próprio estado, conforme ilustrado na Figura 19. O comportamento do modelo pode ser representado por meio de uma máquina de estados, ou seja, é constituído de estados (tipo de dados *State*) e regras de transição de estados (tipo de dados *Jump*). Além disto, é possível definir também um comportamento contínuo a ser executado em um determinado estado, através do tipo de dados *Flow*. Desta forma, é possível descrever tanto comportamentos discretos quanto contínuos em um modelo comportamental.

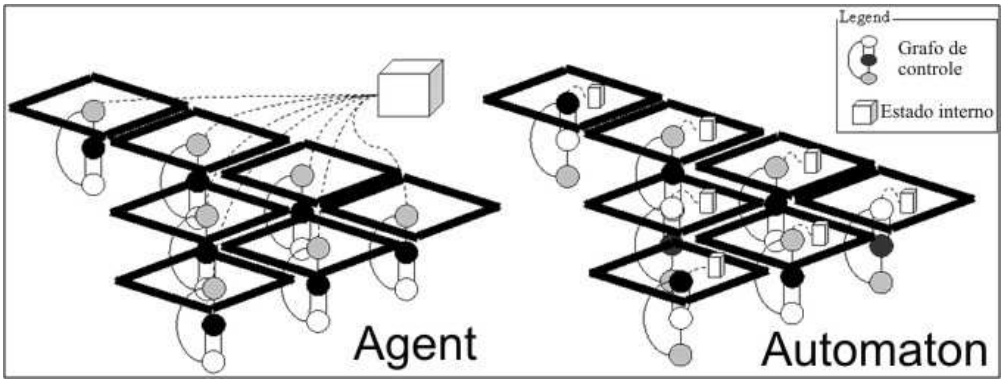


Figura 19 – Modelo comportamental em *TerraME* definido por meio de agentes (mesmo estado interno em cada célula) e autômatos (cada células armazena seu próprio estado interno).
Fonte: Carneiro (2006)

4.5 Aplicações da plataforma *TerraME*

As estruturas de dados e serviços da plataforma *TerraME* para modelagem ambiental, apresentadas ao longo deste capítulo, permitem construir modelos para representar fenômenos ambientais em diversas áreas de aplicação.

A plataforma tem sido utilizada, por exemplo, para o desenvolvimento de modelos de mudança de uso e de cobertura da terra (*Land Use and Land Cover Change – LUCC*) para a Amazônia brasileira (AGUIAR et al. 2007; MOREIRA et al. 2009; PIMENTA et al., 2008), simulação de propagação de fogo em reservas nacionais (ALMEIDA et al. 2008b), simulações sociais (ANDRADE et al. 2009), ecologia de populações (LANA, 2009; GONTIJO, 2009), simulação de processos hidrológicos em áreas urbanas (PEREIRA, 2008).

Apesar da plataforma estar sendo utilizada em diversas áreas na construção de modelos para vários fenômenos ambientais, os conceitos que ela implementa, a complexidade dos fenômenos aos quais se aplica, e a utilização direta de uma linguagem de programação, são barreiras para usuários não experientes em programação. Este fato pode ser constatado em avaliações realizadas por participantes de cursos e apresentações sobre o ambiente *TerraME* (CÂMARA et al., 2007; CÂMARA et al., 2008) e também pela não utilização nos trabalhos acima citados de grande parte dos conceitos implementados pela plataforma. Assim, é esperado que o uso do *TerraME GIMS*, apresentado no próximo capítulo, para a construção de modelos, auxilie os usuários na utilização adequada dos conceitos e recursos fornecidos pela plataforma *TerraME*.

5 DESENVOLVIMENTO DO *TerraME GIMS*

Neste capítulo é apresentada a estratégia adotada para a implementação do *TerraME GIMS* a partir da utilização da plataforma *Eclipse*, as etapas de desenvolvimento, a arquitetura do *software* e a interface gráfica implementada.

5.1 A plataforma *Eclipse*: um ambiente de desenvolvimento integrado extensível

O desenvolvimento de um sistema de computação ou de um modelo computacional para representar fenômenos naturais ou antrópicos são atividades essencialmente similares. Tanto o sistema de computação quanto o modelo precisam ter seus requisitos identificados, suas arquiteturas e estruturas especificadas e seus comportamentos implementados na forma de regras expressas em alguma linguagem de programação. É comum estes dois tipos de projetos envolverem uma equipe de desenvolvimento multidisciplinar e numerosa, algumas vezes dispersa geograficamente. Desta maneira, os atuais ambientes de desenvolvimento integrado (ou IDE, do inglês *Integrated Development Environment*) amplamente utilizados na indústria de desenvolvimento de *software* podem ser também utilizados para o desenvolvimento de modelos ambientais.

Um IDE é um ambiente cujas características e funcionalidades buscam agilizar o processo de desenvolvimento de *software*. Em geral, ambientes desta natureza permitem a representação de projetos de *software* através de diagramas, fornecendo funcionalidades que automatizam o processo de modelagem e implementação do *software*, gerando automaticamente o código-fonte da aplicação a partir de diagramas. Além disto, também podem ser disponibilizados serviços para organizar e facilitar a colaboração da equipe de desenvolvimento.

No entanto, o desenvolvimento de um ambiente desta natureza é uma tarefa extremamente dispendiosa. Então, uma alternativa ao desenvolvimento de um IDE totalmente novo para o ambiente *TerraME* é a reutilização de um IDE já existente. Neste trabalho, a plataforma *Eclipse* [8] é utilizada como estratégia para implementação do *TerraME GIMS*.

[8] http://www.eclipse.org

O *Eclipse* é um IDE de domínio público desenvolvido como um conjunto de *frameworks* de *software*. Estes *frameworks* podem ser reutilizados para o desenvolvimento de novas aplicações que demandem o desenvolvimento de um IDE que torne eficiente a colaboração entre membros da equipe de desenvolvimento e que permita a especificação do sistema por meio de diagramas e a geração automática de código (RIVIERES e WIEGAND, 2004; RIVIERES e BEATON, 2006).

Uma grande vantagem em se utilizar a plataforma *Eclipse* é sua elevada capacidade de integração, obtida por meio de uma arquitetura baseada em *plug-ins*. Desenvolver uma aplicação sobre a plataforma *Eclipse* permite que ela seja integrada a outras aplicações também desenvolvidas sobre a plataforma. Assim, a plataforma *Eclipse* é construída num mecanismo de descobrir, integrar, e executar módulos chamados *plug-ins* (RIVIERES e BEATON, 2006).

Os principais componentes do *Eclipse Software Development Kit* (*Eclipse* SDK) e sua arquitetura baseada em *plug-ins* são ilustrados na Figura 20. O *Eclipse* SDK inclui a plataforma *Eclipse* além de duas ferramentas úteis ao desenvolvimento de *plug-ins*: a *Java Development Tools* (JDT), que implementa um ambiente de desenvolvimento *Java*; e a *Plug-in Development Environment* (PDE) que adiciona ferramentas específicas para o desenvolvimento de *plug-ins* e extensões. Novas aplicações são desenvolvidas estendendo-se o sistema através de *plug-ins* (ECLIPSE, 2009).

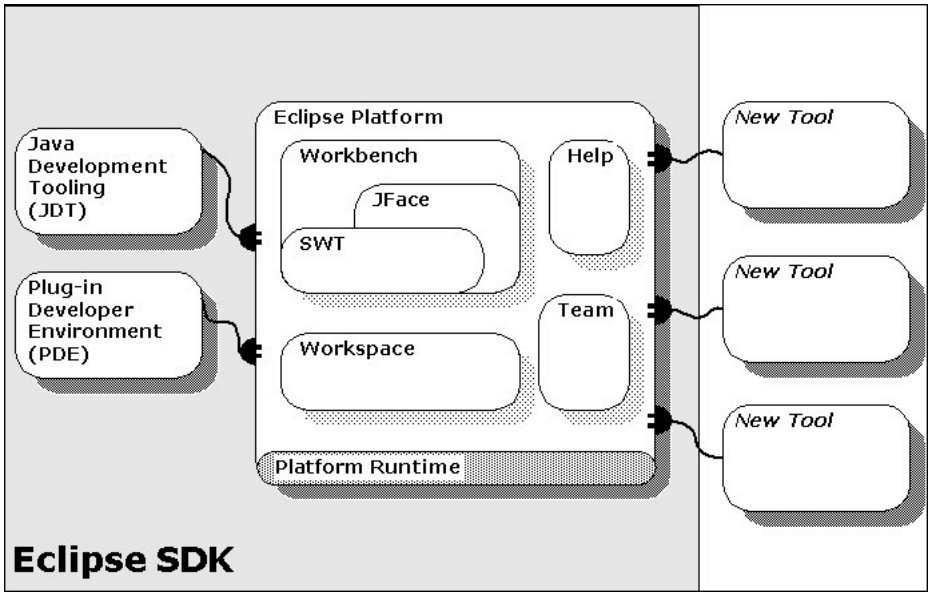


Figura 20 – Arquitetura baseada em *plug-ins* da plataforma *Eclipse* SDK.
Fonte: Rivieres e Beaton (2006)

Na Figura 21 é mostrada a arquitetura da plataforma *Eclipse* baseada em camadas. Na primeira camada temos a *Java Virtual Machine* (JVM), ou Máquina Virtual *Java*, sobre a qual é executada a plataforma *Eclipse*. A camada seguinte, *Platform*, constitui a base da plataforma e suas funcionalidades. Acima desta temos a JDT, oferecendo suporte ao desenvolvimento *Java*, e por último a PDE, constituindo o ambiente de desenvolvimento de *plug-ins*, que permite estender a plataforma acrescentando a ela funcionalidades e ferramentas ou mesmo criar aplicações completas.

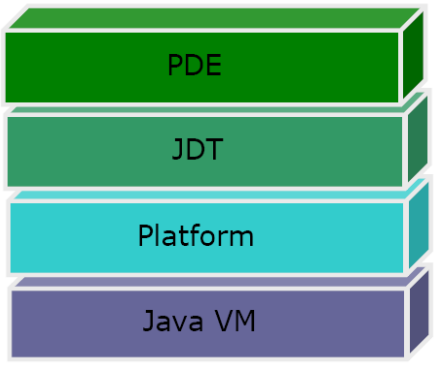


Figura 21 – Arquitetura em camadas da plataforma *Eclipse*.
Fonte: TecComm (2009)

Um *plug-in* é a menor unidade funcional da plataforma *Eclipse* passível de ser desenvolvida e distribuída separadamente. Exceto por um pequeno *kernel* conhecido como *Platform Runtime*, toda a funcionalidade da plataforma *Eclipse* é disponibilizada por meio de *plug-ins* (RIVIERES e BEATON, 2006). Os *plug-ins* são codificados na linguagem de programação *Java* e são distribuídos na forma de bibliotecas de classes e objetos encapsulados em um *Java Archive* (JAR).

A interface gráfica do *Eclipse* é organizada a partir do *workbench*, que corresponde ao seu ambiente de desenvolvimento e busca obter a integração de ferramentas por meio do fornecimento de um meio comum para a criação, gerenciamento e navegação dos recursos do espaço de trabalho (*workspace*). Uma janela do *workbench* é formada a partir de uma ou mais perspectivas (*perspective*), que por sua vez contém *views*, *editors* e *control* (que aparecem na forma de menus e barras de ferramentas), conforme ilustrado na **Erro! Fonte de referência não encontrada.**.

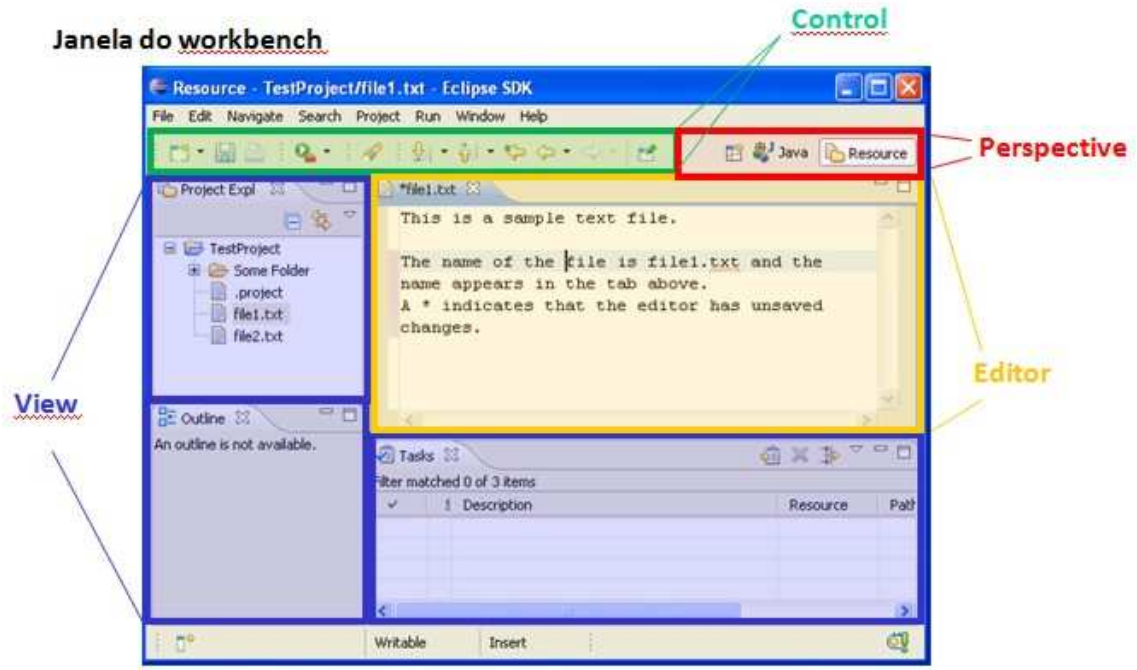


Figura 22 – Janela do Eclipse *workbench*.

Para o desenvolvimento do *TerraME GIMS* são utilizados ainda dois *frameworks* da plataforma *Eclipse*: o *Eclipse Modeling Framework* e o *Graphical Editing Framework*; os quais são apresentados a seguir.

5.1.1 *Eclipse Modeling Framework*

O *Eclipse Modeling Framework* (EMF) é um *framework* para modelagem e um instrumento de geração de código (*code generation facility*) que permite construir ferramentas e outras aplicações baseadas em um modelo de dados estruturado (ECLIPSE EMF, 2005). O EMF faz parte da arquitetura MDA (do inglês *Model Driven Architecture*), cuja base é o desenvolvimento de aplicações com foco no modelo (MORE et al., 2003). O *framework* fornece, a partir de especificações de modelos escritas em XMI (XML *Metadata Interchange*), ferramentas e *runtime* para a geração de um conjunto de classes *Java* correspondente ao modelo, classes adaptadoras que permitem sua visualização e edição, além de um editor básico. Novos modelos são especificados através de interfaces *Java*, documentos XML (*Extensible Markup Language*) ou diagramas UML (*Unified Modeling Language*) e podem então importados para o *framework*. Além disso, o EMF fornece serviços para interoperabilidade entre as aplicações cujo desenvolvimento foi baseado nele (ECLIPSE EMF, 2009).

Três elementos fundamentais compõem o *framework*:

- EMF – é o núcleo do *framework*, que inclui a base para o meta-modelo *Ecore*, que permite descrever modelos, e o suporte de sua execução (com notificação de mudanças, suporte a persistência com serialização XMI, e API para manipulação de objetos EMF);

- EMF.*Edit* – inclui classes genéricas reutilizáveis que permitem a construção de editores para os modelos EMF;

- EMF.*Codegen* – é o instrumento de geração de código EMF; gera código que permite construir editores para modelos EMF e inclui uma interface gráfica, a partir da qual podem ser especificadas opções de customização e invocados os geradores de código do editor.

O EMF, em sua integração modelo/geração de código, suporta operações de criação, restauração, atualização e remoção, além de suportar restrições de cardinalidade (*cardinality constraints*), relacionamentos complexos e estruturas de herança, definições de contenção (*containment definition*), e um conjunto de descrições de atributos, integrando o modelo ao código gerado (POWELL, 2004).

5.1.2 *Graphical Editing Framework*

O *Graphical Editing Framework* (GEF) é um *framework* que permite a criação de ambientes de edição gráfica a partir de modelos. Baseado na biblioteca SWT (do inglês *Standard Widget Toolkit*), ele consiste de dois *plug-ins*:

- *org.eclipse.draw2d* – fornece um conjunto de ferramentas de *layout* e renderização para visualização de gráficos;

- *org.eclipse.gef* – fornece ferramentas como seleção, criação e conexão; dois tipos de *viewers* (*graphical viewer* e *tree viewer*); um *framework* de controle (*Controller*) que permite mapear modelos em visões; suporte a comandos desfazer/refazer (*undo*/*redo*).

O GEF utiliza a arquitetura MVC (*Model-View-Controller*), que permite aplicar mudanças ao modelo a partir do *view*. O GEF é completamente neutro em relação a aplicações e, através dele, é possível criar qualquer editor WYSIWYG (*What You See Is*

What You Get). Na Figura 23 pode-se observar o modelo de funcionamento do *framework.*

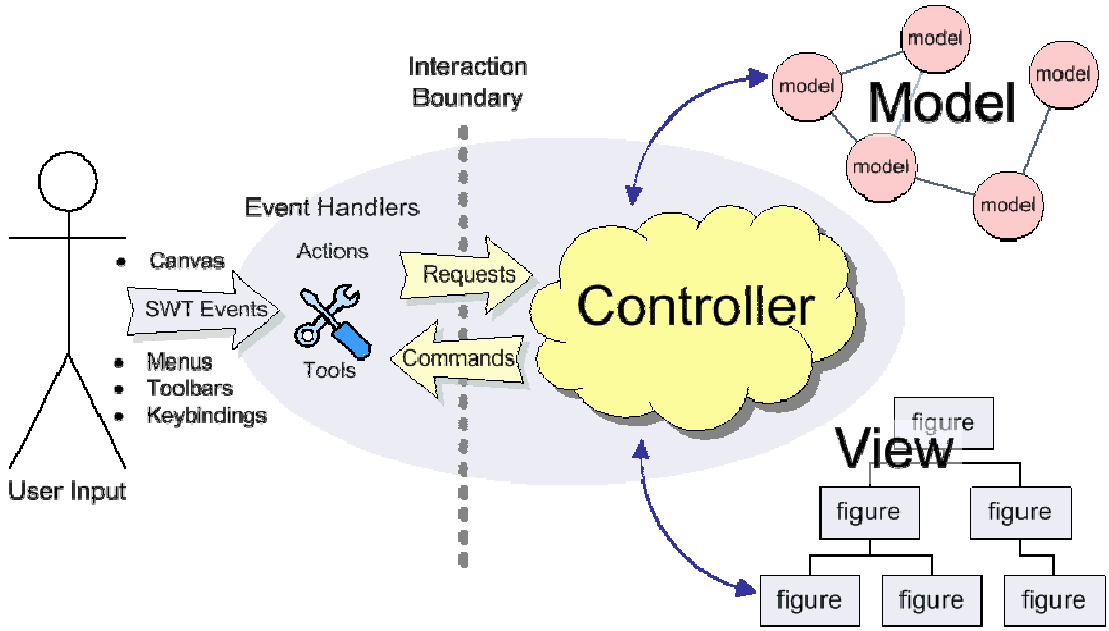


Figura 23 – Visão geral do padrão MVC implementado pelo GEF.
Fonte: Eclipse (2009)

As regras do MVC são aplicadas ao GEF da seguinte forma:

- *model*: o modelo deve possuir algum tipo de mecanismo de notificação, de forma que este possa ser modificado pelas ações do usuário;

- *view*: objetos que são visíveis ao usuário, sendo que tanto figuras quanto elementos em árvore podem ser elementos *view;*

- *controller*: chamado de *EditPart*, faz a ligação entre o *model* e o *view*.

5.2 Metodologia de desenvolvimento

A implementação do *TerraME GIMS* foi realizada de forma incremental e iterativa, em espiral, com prototipação de *releases*, sendo obtida a cada ciclo uma nova versão documentada do *software*. Foram realizados três ciclos de desenvolvimento, sendo cada um constituído de quatro etapas: iniciação, elaboração, construção e transição. Na iniciação foram realizadas atividades de levantamento dos requisitos; na etapa de elaboração foi feita a modelagem destes requisitos; na fase de construção foi realizada a implementação; e na fase de transição, foram realizados os testes e validação dos

requisitos, e atividades de preparação para o próximo ciclo. Os resultados iniciais do desenvolvimento foram apresentados em Lima et al. (2008).

O escopo do *software TerraME GIMS* consiste em uma interface gráfica para a construção de modelos espaciais dinâmicos para a plataforma *TerraME*. Desta forma, o *TerraME GIMS* permite construir graficamente modelos na forma de diagramas a partir da interação com componentes de sua interface, sendo o código *TerraME* correspondente a estes gerado automaticamente. Portanto, todos os tipos de dados para modelagem da plataforma *TerraME* apresentados na Seção 4.3, com exceção do tipo de dados *Cell*, possuem uma representação gráfica e podem ser criados e editados a partir do editor gráfico da interface. Desta forma, é possível criar a estrutura do código *TerraME* correspondente ao modelo, ou seja, sua parte estática, de forma gráfica. Não faz parte do escopo deste trabalho a representação e construção de forma gráfica do código correspondente à parte dinâmica do modelo, ou seja, as regras que definem a sua execução e são constituídas de expressões, comandos, e estruturas de controle da linguagem de programação tais como *for* e *if*. Baseado na plataforma *Eclipse*, o *TerraME GIMS* será distribuído como um conjunto de *plug-ins* para esta.

5.2.1 Arquitetura do *TerraME GIMS*

Duas alternativas de desenvolvimento foram analisadas para a implementação do *TerraME GIMS*. A primeira alternativa consiste no desenvolvimento completo do aplicativo, sua interface gráfica e integração com o *TerraME*. Esta foi desconsiderada devido à elevada complexidade do sistema, alto custo e necessidade de maior prazo e de uma equipe de desenvolvimento.

A segunda alternativa consiste na implementação a partir de *frameworks* ou bibliotecas já existentes. Esta alternativa se mostrou viável e a plataforma *Eclipse* (apresentada na Seção 5.2), por ser amplamente extensível, permitir a integração com diversas ferramentas e oferecer um vasto conjunto de *frameworks* de *software* de domínio público, foi escolhida para servir de base para o desenvolvimento do *TerraME GIMS*.

Para acelerar o processo de desenvolvimento do *TerraME GIMS*, este foi desenvolvido na forma de um conjunto de *plug-ins* para a plataforma *Eclipse*. Os *frameworks* EMF e GEF dessa plataforma foram estendidos para permitir a geração de código *TerraME* a partir de diagramas.

Desta forma, o *TerraME GIMS*, em conformidade com as plataformas sobre as quais é desenvolvido (*TerraME* e *Eclipse*), é baseado numa arquitetura em camadas. Na arquitetura em camadas, as camadas inferiores oferecem serviços a serem utilizados pelas camadas superiores para implementação de seus serviços. O *TerraME GIMS* compõe uma nova camada, entre o *TerraME* e o usuário final, sendo a plataforma *Eclipse* uma camada intermediária entre o *TerraME* e o *TerraME GIMS*. Desta forma, não há restrição para a criação de modelos diretamente sobre a linguagem *TerraME* quando é utilizado o *TerraME GIMS*.

A arquitetura é apresentada na **Erro! Fonte de referência não encontrada.**, incluindo as camadas da arquitetura *TerraME* (apresentada na Seção 4.2 e ilustrada na Figura 10) e as camadas da plataforma *Eclipse*. Acima da camada correspondente à plataforma *Eclipse* estão os *frameworks* EMF e GMF, utilizados para implementar funcionalidades relacionadas à criação e edição de modelos e de edição de componentes gráficos, respectivamente. A última camada é a camada de aplicação, onde se encontram os modelos ambientais a serem desenvolvidos pelos usuários finais do *TerraME GIMS*.

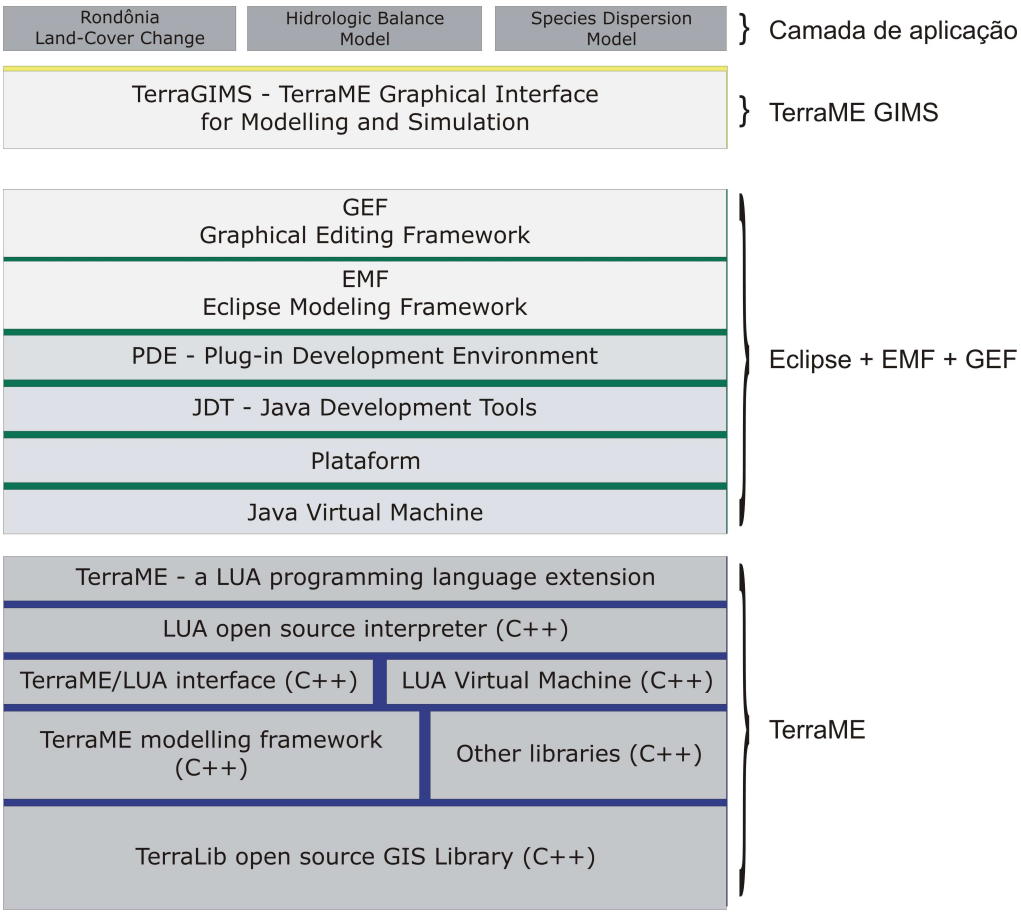


Figura 24 – Arquitetura em camadas do *TerraME GIMS*.

5.3 *TerraME GIMS*

Conforme apresentado neste capítulo, o *TerraME GIMS* é implementado como um conjunto de *plug-ins* que são executados sobre a plataforma *Eclipse*, adicionando funcionalidades para a construção e visualização de modelos para a plataforma *TerraME* a partir de componentes de interface gráfica, tais como diagramas, caixas de texto, árvores etc. Desta forma, a interface gráfica, ilustrada na **Erro! Fonte de referência não encontrada.**, está em conformidade com os padrões de interface gráfica da plataforma *Eclipse* (**Erro! Fonte de referência não encontrada.**). Não faz parte do escopo do trabalho o estudo semiótico e desenvolvimento dos signos (De SOUZA, 1993; De SOUZA, 2001; PEIRCE, 1931) que compõem a interface gráfica. Desta forma, foram utilizados como ícones para os elementos os signos já desenvolvidos no trabalho de Carneiro (2006).

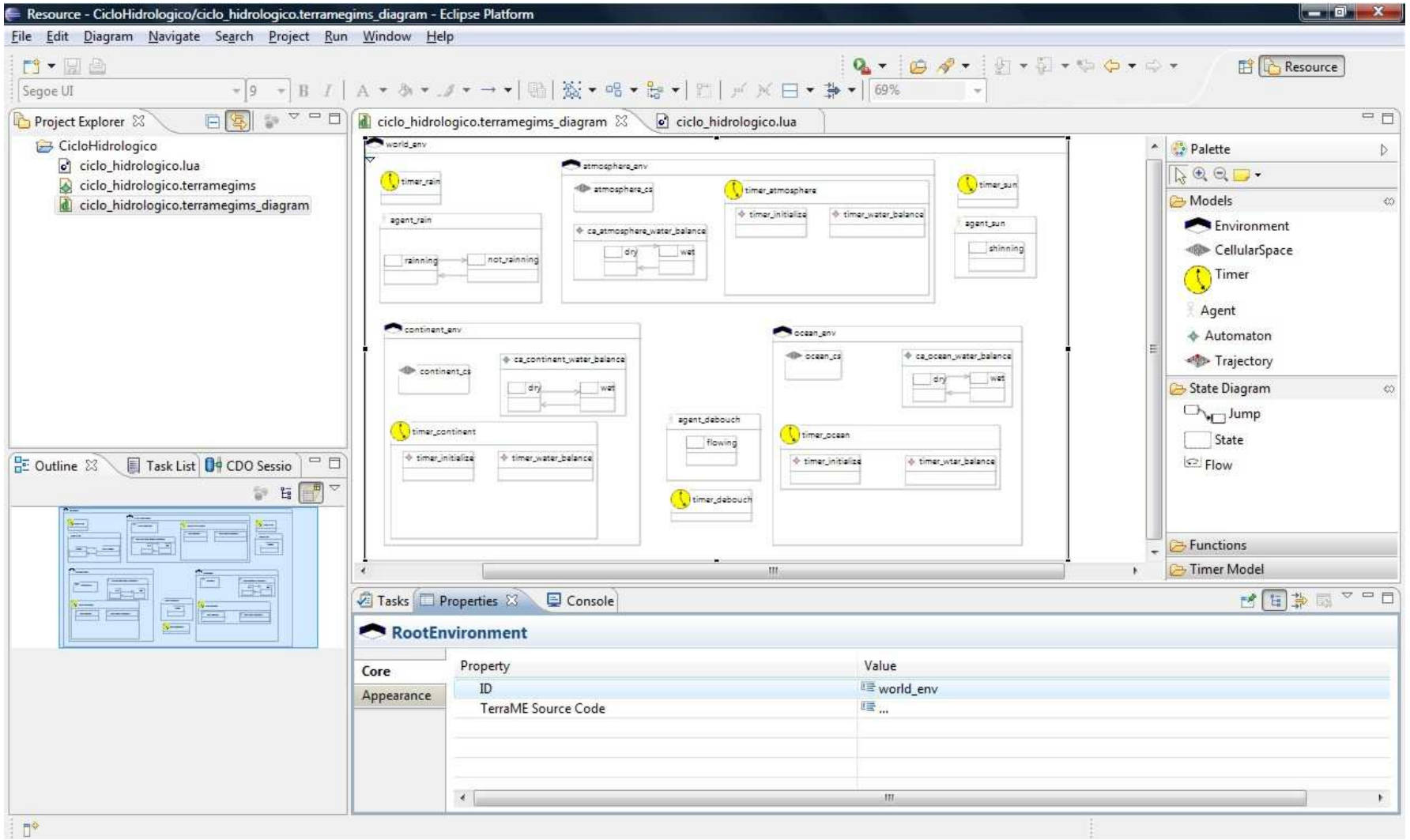


Figura 25 – Visão geral da interface gráfica do *TerraME GIMS*.

A construção de modelos na plataforma *TerraME* utilizando o *TerraME GIMS* é feita a partir das *views Project Explorer*, *Outline* e *Properties*, do editor gráfico de arquivos *TerraME GIMS*, conforme ilustrado na Figura 26.

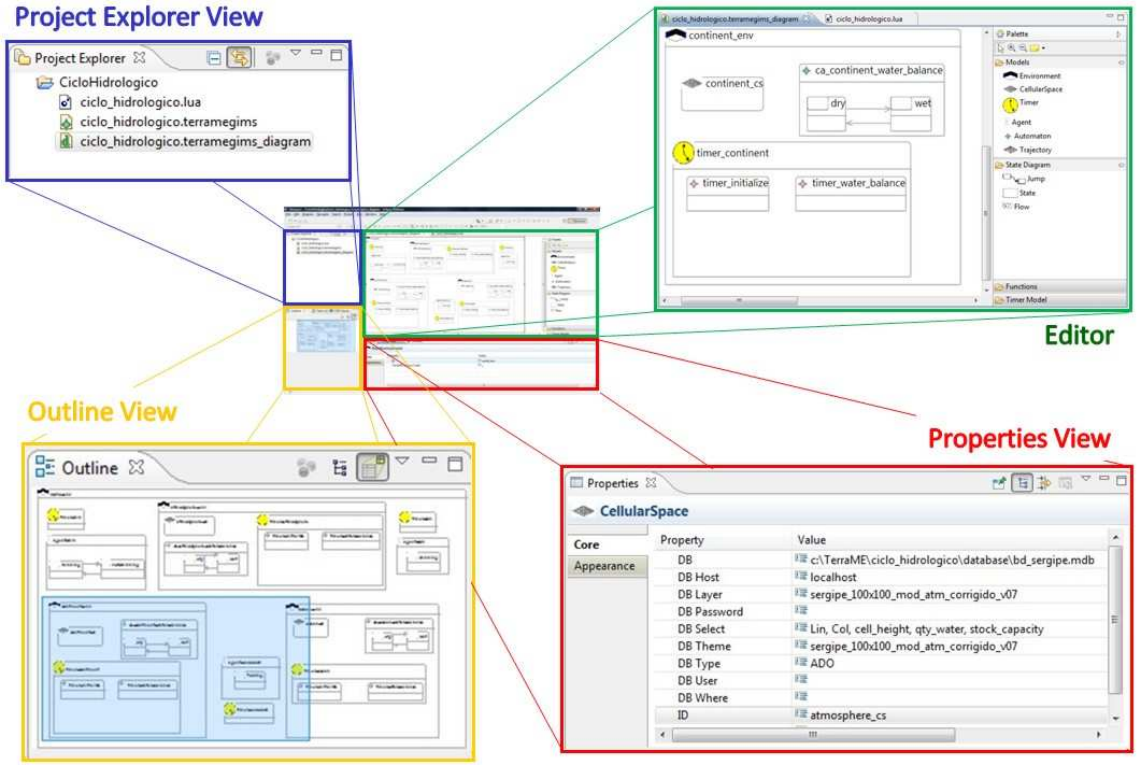


Figura 26 – Projeto da interface gráfica do *TerraME GIMS*.

O *TerraME GIMS Editor* é um editor gráfico que permite construir modelos ambientais para a plataforma *TerraME* a partir de diagramas. Na "paleta" (*Palette*) são apresentados os tipos de dados disponíveis para a construção do modelo *TerraME*, apresentados na Seção 4.3, tais como como *Environment*, *CellularSpace*, *Timer*, *Agent*, *Automaton*, e o usuário pode então compor o modelo a partir destes tipos de dados e visualizá-lo na forma de um diagrama, conforme ilustrado na **Erro! Fonte de referência não encontrada.**. Este tipo de visualização facilita a construção e a identificação dos relacionamentos de composição entre os elementos que constituem o modelo. O *Editor* também permite exibir e ocultar elementos do modelo.

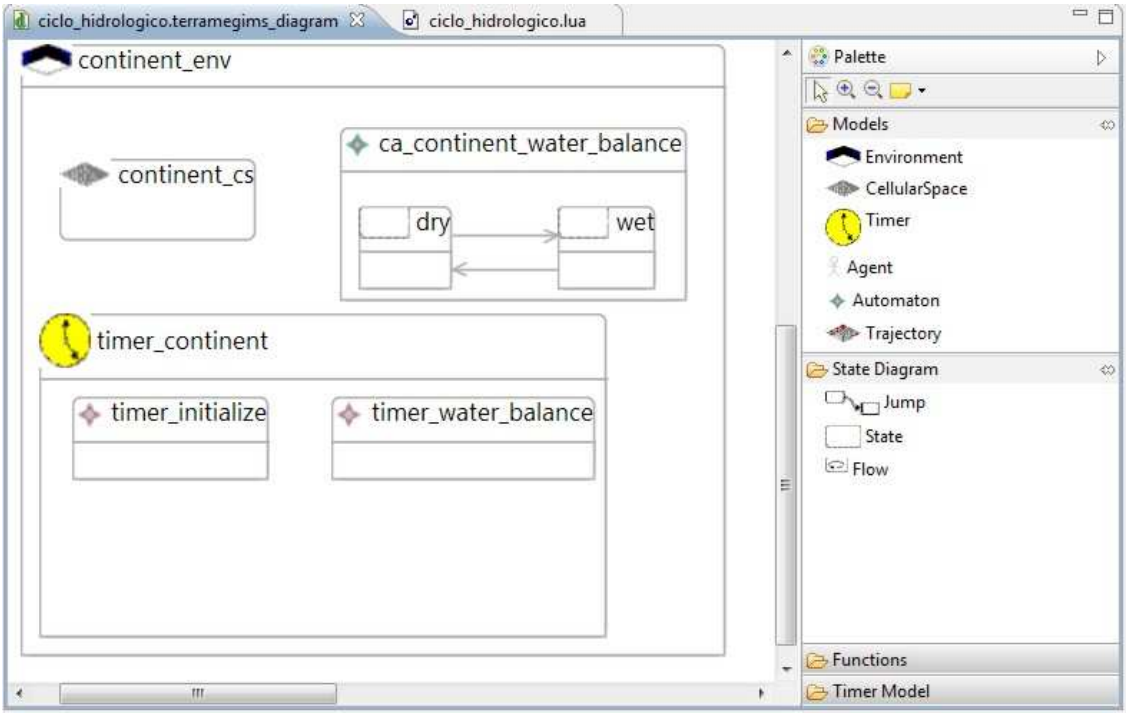


Figura 27 – Interface gráfica do *TerraME GIMS*.

No *Eclipse* os arquivos são criados a partir de um projeto, que consiste em um diretório onde recursos (arquivos e diretórios) podem ser criados e armazenados. Da mesma forma, ao utilizar o *TerraME GIMS* para se criar um arquivo para um modelo ambiental para a plataforma deve-se criar inicialmente um projeto *TerraME GIMS*. A visualização e manipulação dos arquivos que constituem um projeto é feita a partir do *Project Explorer View*, que permite também visualizar o modelo na estrutura de uma árvore, como ilustrado na **Erro! Fonte de referência não encontrada.**.

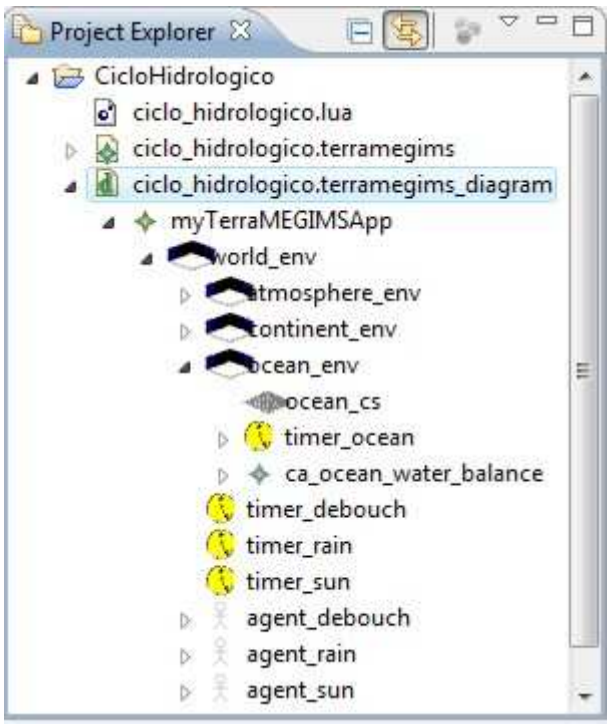


Figura 28 – Interface gráfica do *TerraME GIMS*: *Project Explorer view*.

Na medida em que o modelo é construído e constituído de um grande número de elementos, a visualização deste a partir do *Editor* pode se tornar uma tarefa difícil. Neste sentido, para auxiliar os usuários é apresentada uma visão geral no *Outline view*, que permite navegar pelo modelo apresentado no *Editor*, como ilustrado na **Erro! Fonte de referência não encontrada.**.

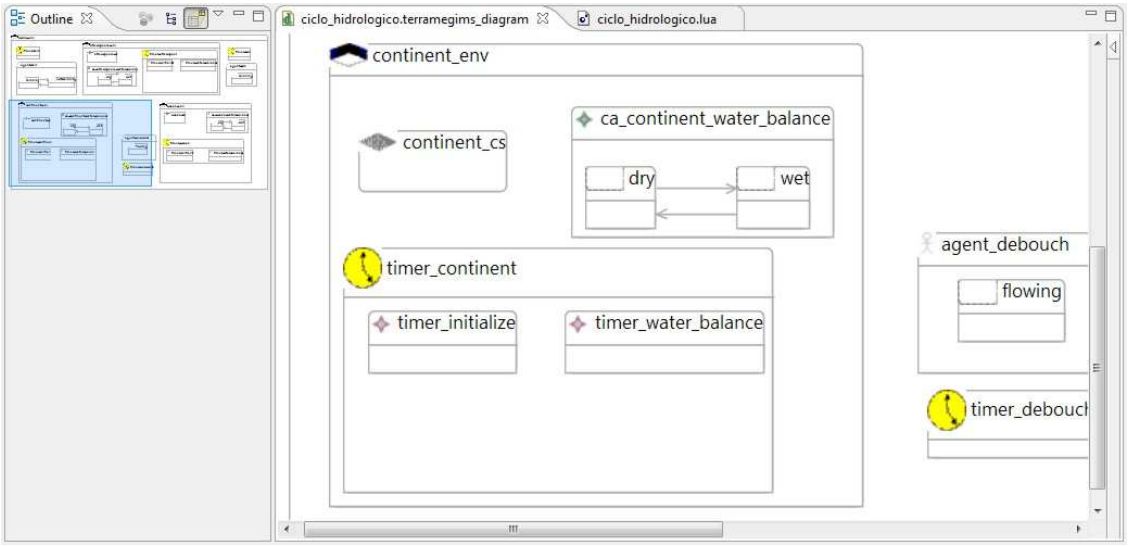


Figura 29 – Interface gráfica do *TerraME GIMS*: *Outline view*.

As propriedades dos elementos que constituem o modelo podem ser editadas a partir da interface gráfica. O *Properties view*, conforme ilustrado na **Erro! Fonte de referência**

não encontrada., apresenta e permite editar as propriedades de um elemento selecionado no *Editor*.

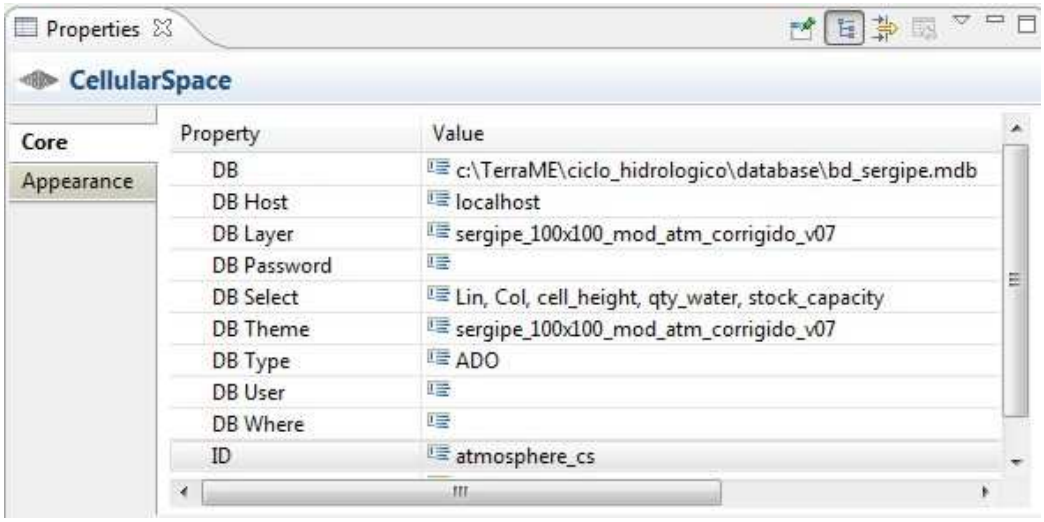


Figura 30 – Interface gráfica do *TerraME GIMS*: *Properties view*.

Uma vez construído o modelo a partir do editor gráfico e especificadas as propriedades dos elementos que o constituem, o código *TerraME* correspondente pode então ser gerado e o modelo simulado por meio de sua execução pelo interpretador *TerraME*.

Para demonstrar a aplicação do *TerraME GIMS* no desenvolvimento de modelos para a plataforma *TerraME* e validar o código gerado construído a partir da modelagem visual, conforme apresentado neste capítulo, um modelo pedagógico utilizando os principais recursos de modelagem do *TerraME* é apresentado no Capítulo 6.

6 APLICAÇÃO DO *TerraME GIMS* NA CONSTRUÇÃO DE UM MODELO DO CICLO HIDROLÓGICO

Neste capítulo é apresentada a avaliação do *TerraME GIMS* a partir da sua aplicação na construção de um modelo. Para esta avaliação é utilizado um modelo "pedagógico" do ciclo hidrológico, desenvolvido na plataforma *TerraME*. O modelo foi inicialmente desenvolvido utilizando diretamente a linguagem de programação *TerraME*, e posteriormente uma nova versão do modelo foi construída a partir da interface gráfica disponibilizada pelo *TerraME GIMS*.

6.1 Metodologia de Avaliação

Uma vez implementado o *TerraME GIMS*, a primeira etapa da avaliação deve ter como foco verificar a utilidade e "corretude" (*correctness*) do mesmo. Para isto, um modelo pedagógico foi construído com o intuito de demonstrar os recursos oferecidos pela plataforma *TerraME* para a modelagem e simulação de sistemas dinâmicos espacialmente explícitos. Desta forma, este modelo é representativo quanto à utilização das estruturas de dados (Seção 4.3) e serviços disponíveis aos usuários da plataforma, tais como: integração do modelo com uma base de dados geográficos; descrição de fenômenos ambientais a partir de modelos espacial, temporal e comportamental; representação do comportamento do fenômeno de forma discreta e contínua; modelagem em múltiplas escalas (espaciais, temporais e comportamentais). Uma vez desenvolvido o modelo diretamente a partir da linguagem de programação *TerraME* fazendo uso das estruturas de dados e serviços oferecidas pela plataforma, uma nova versão do mesmo foi desenvolvida a partir da interface gráfica *TerraME GIMS*, e o código *TerraME* correspondente foi gerado e validado a partir da comparação com o código desenvolvido utilizando diretamente a linguagem de programação *TerraME*.

6.1.1 Definição do problema: o ciclo hidrológico

O ciclo hidrológico (ou ciclo da água) é o fenômeno global de circulação da água entre a superfície terrestre e a atmosfera, e pode ser considerado um sistema fechado em nível global (SILVEIRA, 2003; WARD e STANLEY, 2004).

O ciclo hidrológico ocorre na superfície terrestre, na atmosfera, e na interação entre estes, conforme ilustrado na Figura 31. Uma fase do ciclo acontece na superfície

terrestre, que abrange os continentes e os oceanos, onde a circulação da água ocorre no interior e na superfície dos solos e rochas, nos oceanos e nos seres vivos. Na atmosfera, a camada da troposfera, que contém quase a totalidade da umidade atmosférica, engloba a maior parte dos fenômenos meteorológicos, enquanto a camada da estratosfera tem um papel importante por conter a camada de ozônio (reguladora da radiação solar que atinge a superfície terrestre e que é a principal fonte de energia do ciclo). A água que circula na atmosfera constitui, portanto, outra fase do ciclo hidrológico. Por fim, fechando o ciclo, tem-se o intercâmbio entre a circulação de água na superfície terrestre e na atmosfera, que ocorre em dois sentidos: 1) no sentido superfície-atmosfera, onde a transferência da água ocorre fundamentalmente na forma de vapor, em decorrência dos fenômenos de evaporação e transpiração; 2) no sentido atmosfera-superfície, onde a transferência de água pode ocorrer nos três estados físicos, sendo mais significativos em escala global os fenômenos de precipitação de chuva e de neve (SILVEIRA, 2003).

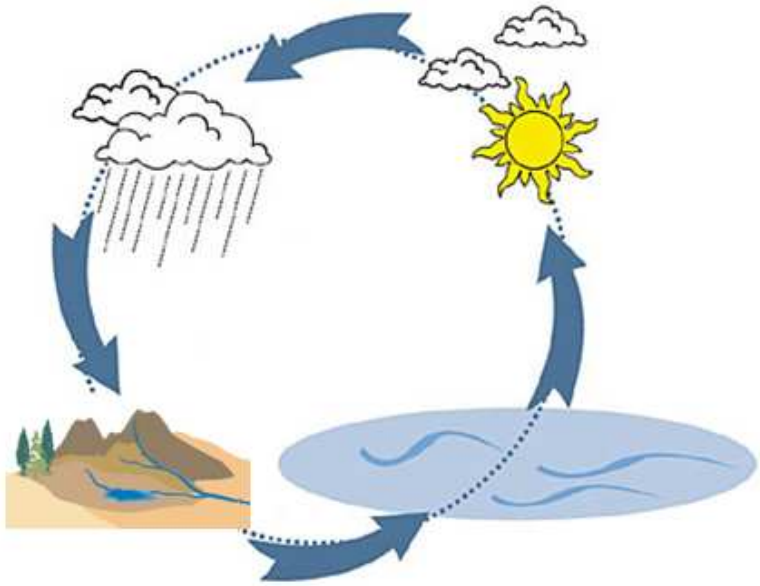


Figura 31 – Visão simplificada do ciclo hidrológico.
Fonte: ilustração adaptada de
(http://bbel.uol.com.br/upload_2009/conteudo/agua_vida_1.jpg). Acesso: Julho/2010.

6.1.2 Modelagem conceitual do ciclo hidrológico

Um modelo pedagógico do ciclo hidrológico, descrito nesta seção, é construído para exemplificar a utilização das estruturas de dados e serviços fornecidos pela plataforma *TerraME* para a modelagem e simulação de fenômenos ambientais.

As seguintes etapas, definidas a partir daquelas propostas por Hannon e Ruth (2001), Turner et al. (2001) e Carneiro (2006) e apresentadas na Seção 2.3, constituem o processo de desenvolvimento do modelo: 1) definição do problema; 2) desenvolvimento do modelo conceitual; 3) construção da base de dados; 4) implementação computacional do modelo; 5) calibração e validação do modelo; 6) simulação (execução) e análise dos resultados.

A primeira etapa, definição do problema, consiste na construção de um modelo pedagógico demonstrando a utilização da plataforma *TerraME* que permita visualizar o ciclo hidrológico através dos fluxos da água na superfície (continente e oceano) e na atmosfera e entre estes.

A segunda etapa consiste na definição do modelo conceitual para representar o problema. Conforme apresentado na seção 6.1.1, a interação no ciclo hidrológico entre a atmosfera e a superfície (formada por continentes e oceanos) ocorre a partir do fluxo de água entre estes, que são representados no modelo pelos fenômenos de evaporação (fluxo de água da superfície para a atmosfera), precipitação (fluxo de água da atmosfera para a superfície) e desaguamento (fluxo de água do continente para o oceano). Para tornar o modelo mais simples, atendendo ao seu propósito pedagógico de demonstrar a utilização das funcionalidades da plataforma *TerraME*, a superfície é modelada de forma separada em continente e oceano, de tal forma que o processo de evaporação ocorrerá somente no oceano e o processo de precipitação ocorrerá apenas no continente. O fluxo de água também ocorre internamente na atmosfera, continente e oceano, sendo considerados respectivamente os processos de convecção, escoamento superficial, e corrente superficial para representar estes fenômenos. O modelo conceitual é ilustrado na Figura 32.

Figura 32 – Modelo conceitual do modelo do ciclo hidrológico.
Fonte: ilustração adaptada de (http://www.dca.ufcg.edu.br/vapordagua/agua01.jpg).
Acesso: Julho/2010.

A terceira etapa da construção do modelo consiste na construção da base de dados a ser utilizada. Para isto é utilizada uma imagem SRTM do estado de Sergipe (Folha SC-24-Z-B)[9], a qual é recortada com o objetivo de obter uma região de tamanho menor e com áreas de continente e oceano, conforme ilustrado na Figura 33

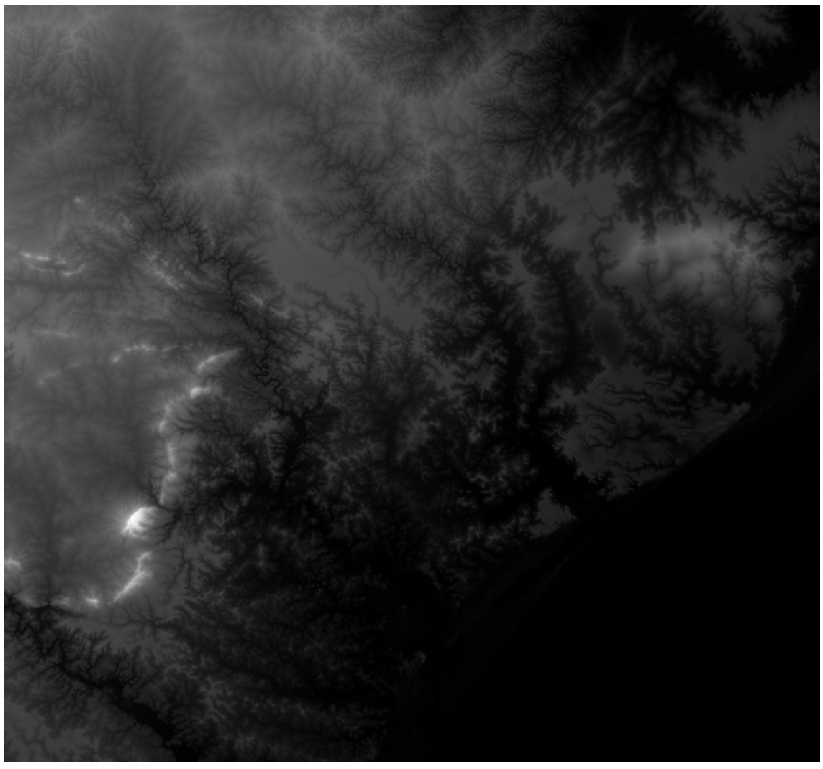
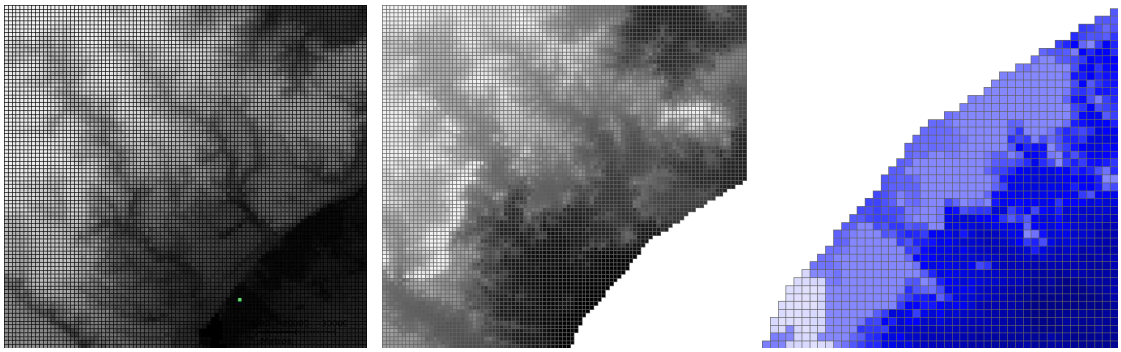


Figura 33 – Imagem SRTM da região de estudo (Folha SC-24-Z-B).

[9] A imagem foi obtida a partir do site “Brasil em Relevo” (http://www.relevobr.cnpm.embrapa.br/)

Utilizando-se o SIG *TerraView 3.2*, o banco de dados a ser utilizado é construído. São geradas a partir do recorte da imagem SRTM grades de células regulares para a atmosfera, continente e oceano, como ilustrado na Figura 34, onde tons de cinza e azul mais claros indicam maior altitude e menor profundidade respectivamente, enquanto tons mais escuros indicam menor altitude e maior profundidade.



(a) (b) (c)

Figura 34 – Vistas do *TerraView* contendo grades de células: (a) atmosfera, (b) continente, (c) oceano.

As próximas etapas (implementação computacional do modelo, calibração e validação do modelo, e simulação (execução) e análise dos resultados) são apresentadas na Seção 6.1.3. Na Seção 6.1.4 uma nova versão do modelo é desenvolvida, tendo como base as etapas anteriores conforme descrito nesta seção e a etapa de implementação computacional do modelo realizada por meio da utilização da interface gráfica do *TerraME GIMS*.

6.1.3 Modelagem utilizando a linguagem de programação *TerraME*

O *TerraME* oferece a estrutura de dados *Environment*, que permite a construção aninhada dos elementos conceituais do modelo: mundo, continente, oceano, atmosfera. Cada um destes elementos é implementado como um *Environment*, como ilustrado na Figura 35. Um modelo espacial, temporal e comportamental é então implementado para cada um dos *environments*.

```
world_env = Environment {
    id = "world_env",

    atmosphere_env = Environment {
        id = "atmosphere_env"
    },

    continent_env = Environment {
        id = "continent_env"
    },

    ocean_env = Environment {
        id = "ocean_env"
    }
}
```

Figura 35 – Definição dos ambientes na linguagem de modelagem *TerraME*.

O modelo espacial é construído através de uma conexão com a base de dados criada anteriormente (Seção 6.1.2), onde são carregados dados de quantidade de água, capacidade de armazenamento de água e altitude (profundidade para o oceano) das células, para cada um dos espaços celulares do modelo, como mostrado na Figura 36.

```
atmosphere_cs =     CellularSpace {
    dbType      =   "ADO",
    host        =   "localhost",
    database    =   DATABASE_PATH.."\\bd_sergipe.mdb",
    layer       =   "sergipe_100x100_mod_atm_corrigido_v07",
    theme       =   "sergipe_100x100_mod_atm_corrigido_v07",
    select      =   { "Lin", "Col", "cell_height", "qty_water", "stock_capacity" }
}
atmosphere_cs:load();

continent_cs =  CellularSpace {
    dbType      =   "ADO",
    host        =   "localhost",
    database    =   DATABASE_PATH.."\\bd_sergipe.mdb",
    layer       =   "sergipe_100x100_mod_con_corrigido_v07",
    theme       =   "sergipe_100x100_mod_con_corrigido_v07",
    select      =   { "Lin", "Col", "cell_height", "qty_water", "inf_capacity" }
}
continent_cs:load();

ocean_cs =  CellularSpace {
    dbType      =   "ADO",
    host        =   "localhost",
    database    =   DATABASE_PATH.."\\bd_sergipe.mdb",
    layer       =   "sergipe_100x100_mod_oce_corrigido_v07",
    theme       =   "sergipe_100x100_mod_oce_corrigido_v07",
    select      =   { "Lin", "Col", "cell_depth", "qty_water", "stock_capacity" }
}
ocean_cs:load();
```

Figura 36 – Definição dos modelos espaciais (espaços celulares) do modelo na linguagem de modelagem *TerraME*.

O fluxo de água também ocorre internamente no continente, no oceano e na atmosfera devido a diversos fenômenos. Estes fluxos foram implementados no modelo a partir de uma representação dos processos de escoamento superficial e infiltração; correntes

superficiais; e convecção, respectivamente. Outra abstração realizada, com o intuito de simplificar o modelo, é considerar apenas dados de altimetria como parâmetro para determinar a direção do fluxo da água.

No continente, o fluxo de água é modelado através dos processos de escoamento superficial e de infiltração. O modelo comportamental para representar este processo é implementado utilizando um autômato celular (*cellular_automaton_continent_water_balance*) constituído de dois estados (*State*): um estado para solo insaturado (*dry*), e um estado para solo saturado (*wet*), e duas regras de transição entre estes estados, que descrevem o comportamento discreto do modelo. O comportamento contínuo é descrito por meio de duas regras *Flow*: a primeira descreve a infiltração da água no solo, na segunda ocorre a drenagem da água excedente para as células vizinhas de menor altitude. Na Figura 30 é mostrado o código da implementação do modelo comportamental do continente.

```
-- cellular automaton for the overland flow process
cellular_automaton_continent_water_balance = Automaton {
    id = "cellular_automaton_continent_water_balance",
    it = Trajectory{ continent_cs, function( cell ) return true; end },
    State {
        id = "dry",
        Jump {
            function( event, agent, cell )  return (cell.qty_water > cell.inf_capacity);    end,
            target = "wet"
        },
        Flow {
            function(event, agent, cell)
                cell.qty_water = cell.past.qty_water + cell.past.runoff;
                cell.runoff = 0;
            end
        }
    },
    State {
        id = "wet",
        Jump {
            function( event, agent, cell )  return (cell.qty_water <= cell.inf_capacity);     end,
            target = "dry"
        },
        Flow {
            function(event, agent, cell)
                -- count the number of lower neighbor cells
                local count_lower_neigh = 0;
                ForEachNeighbor(
                    cell,
                    function(cell, neigh)
                        if (cell ~= neigh) and (neigh.cell_height <= cell.cell_height) then
                            count_lower_neigh = count_lower_neigh + 1;
                        end
                    end
                );
                cell.qty_lower_neigh = count_lower_neigh;

                cell.surplus = cell.past.qty_water - cell.inf_capacity;
                if( cell.surplus < 0) then cell.surplus = 0; end
                -- process: send water to the lower neighbor cells
                if( cell.qty_lower_neigh > 0 ) then
                    local runoff = cell.surplus / cell.qty_lower_neigh;
                    ForEachNeighbor(
                        cell,
                        function(cell, neigh)
                            if (cell ~= neigh) and (neigh.cell_height <= cell.cell_height) then
                                neigh.runoff = neigh.runoff + runoff;
                            end
                            return true;
                        end
                    );
                    cell.qty_water = cell.inf_capacity
                    cell.surplus = 0;
                end;
            end
        }
    }
}
```

Figura 37 – Definição do modelo comportamental do ambiente continente (processo de escoamento superficial) na linguagem de modelagem *TerraME*.

No oceano, a entrada de água ocorre por meio do “desaguamento” da água do continente em regiões litorâneas (regiões de fronteiras entre continente e oceano), e o fluxo de água segue no sentido da praia rumo ao oceano (de pequenas para grandes profundidades). Desta forma, o modelo comportamental é semelhante ao do continente, sendo também implementado utilizando-se um autômato celular

(*cellular_automaton_ocean_water_balance*) constituído de regras que fazem a distribuição da água em cada célula para as células vizinhas de maior profundidade.

Um dos processos responsáveis pela movimentação da água na atmosfera é a convecção, que está relacionada com a temperatura do ar. Uma vez que a temperatura apresenta relação direta com a altitude, e no modelo em questão não são necessárias informações precisas de temperatura, optou-se por utilizar a correlação entre estas. Desta forma, na atmosfera o fluxo da água (na forma de vapor de água) segue no sentido de regiões de menor altitude para aquelas de maior altitude. Novamente, para modelar o processo de fluxo da água internamente em um ambiente, foi utilizado um autômato celular (*cellular_automaton_atmosphere_water_balance*).

Os modelos temporais dos *environments* do continente (*continent_env*), da atmosfera (*atmosphere_env*) e do oceano (*ocean_env*) são semelhantes, sendo constituídos de dois pares evento/mensagem (*Event/Message*), conforme ilustrado na Figura 38 para o continente, onde: (i) o primeiro evento ocorre apenas uma vez, quando são inicializados os parâmetros do espaço celular, tais como quantidade e capacidade de armazenamento de água de cada célula; (ii) o segundo evento ocorre a cada período de tempo da simulação, quando é feita a chamada para execução do autômato celular.

```
time_continent = Timer {
    id = "timer_continent",

    -- "time_prepare_cellular_space_con",
    Pair {
        Event { time = 0, period = 1, priority = -1000 },
        Message {
            function( event )
                -- initialize the cellular space attributes
                (...)
                continent_cs:synchronize();
                return false;
            end
        }
    },

    -- "time_water_balance_con",
    Pair {
        Event { time = SIMULATION_TIME_INITIAL, period = 1, priority = 0 },
        Message {
            function( event )
                continent_env.cellular_automaton_continent_water_balance:build();
                continent_env.cellular_automaton_continent_water_balance:setTrajectoryStatus(true);
                continent_env.cellular_automaton_continent_water_balance:execute(event);
                continent_cs:synchronize();
                return true;
            end
        }
    }
}
```

Figura 38 – Definição do modelo temporal do ambiente continente na linguagem de modelagem *TerraME*.

Os processos de evaporação, precipitação e desaguamento, por meio dos quais ocorrem os fluxos de água entre os *environments continent_env*, *ocean_env* e *atmosphere_env*, correspondem ao modelo comportamental do *environment world_env* e são implementados por meio de três agentes: *agent_sun*, *agent_rain*, *agent_debouch*, que correspondem aos processos de evaporação, precipitação e desaguamento, respectivamente.

Um dos agentes que constitui o modelo comportamental do *world_env* é o *agent_sun*, que representa o processo de evaporação. O agente, que tem como trajetória o oceano, ao ser executado percorre cada célula do espaço celular do oceano executando a evaporação de parte da água, ou seja, transferindo parte da água da célula do oceano para a célula correspondente da atmosfera, conforme apresentado na Figura 39.

```
agent_sun = Agent {
    id = "agent_sun",
    it = Trajectory{ ocean_cs, function( cell ) return true; end },
    State {
        id = "shining",
        Flow {
            function(event, agent, cell)
                -- each cell of ocean has only one neigh in atmosphere
                ForEachNeighbor(cell,
                    function(cell, neigh)
                        local qty_water_to_evaporate = math.floor(cell.past.qty_water * EVAPORATE_COEF);
                        neigh.qty_water = neigh.past.qty_water + qty_water_to_evaporate;
                        cell.qty_water = cell.past.qty_water - qty_water_to_evaporate;
                    end,
                    'inter_oce_atm'
                );
            end
        }
    },
}
```

Figura 39 – Definição do modelo comportamental do ambiente mundo (processo de evaporação) na linguagem de modelagem *TerraME*.

O agente *agent_rain*, que corresponde ao processo de precipitação, percorre o espaço celular da atmosfera (*atmosphere_env*) realizando a precipitação, ou seja, transferindo parte da água de cada célula da atmosfera para a respectiva célula do continente sempre que a primeira se encontrar em um estado de saturação, conforme ilustrado na Figura 40.

```
agent_rain = Agent {
    id = "ag_rain",
    it = Trajectory{ atmosphere_cs, function( cell ) return true; end },
    State {
        id = "raining",
        Flow {
            function(event, agent, cell)
                -- each cell of atmosphere has only one neigh in continent
                ForEachNeighbor(cell,
                    function(cell, neigh)
                        local qty_water_to_rain = math.floor(cell.past.qty_water * RAIN_COEF);
                        cell.qty_water = cell.past.qty_water - qty_water_to_rain;
                        neigh.qty_water = neigh.past.qty_water + qty_water_to_rain;
                    end,
                    'inter_atm_con'
                );
            end
        },
        Jump {
            function( event, agent, cell ) return (cell.qty_water < ATMOSPHERE_SATURATION_CONST); end,
            target = "not_raining"
        }
    },
    State {
        id = "not_raining",
        Flow {
            function(event, agent, cell) end
        },
        Jump {
            function( event, agent, cell ) return (cell.qty_water >= ATMOSPHERE_SATURATION_CONST);  end,
            target = "raining"
        }
    }
}
```

Figura 40 – Definição do modelo comportamental do ambiente mundo (processo de precipitação) na linguagem de modelagem *TerraME*.

Por sua vez, o agente *agent_debouch* corresponde ao processo de fluxo da água do continente para o oceano, onde parte da água de cada célula litorânea do continente é transferida para as células do oceano que lhe são vizinhas e de menor altitude, conforme ilustrado na Figura 41.

```
agent_debouch = Agent {
    id = "ag_debouch",
    it = Trajectory{ continent_cs, function( cell ) return true; end },
    State {
        id = "flowing",
        Flow {
            function(event, agent, cell)
                -- count the number of neighbor cells
                local count_neigh = 0;
                ForEachNeighbor(cell,
                    function(cell, neigh)
                        if (cell ~= neigh) then count_neigh = count_neigh + 1;  end
                    end,
                    'inter_con_oce'
                );
                cell.qty_coast_neigh = count_neigh;

                cell.qty_water_debouch = cell.past.qty_water * DEBOUCH_COEF;
                -- process: send water to neighboor cells
                if( cell.qty_coast_neigh > 0 ) then
                    local qty_water_to_debouch = cell.qty_water_debouch / cell.qty_coast_neigh;
                    ForEachNeighbor(
                        cell,
                        function(cell, neigh)
                            if (cell ~= neigh) then
                                cell.qty_water = cell.past.qty_water - qty_water_to_debouch;
                                neigh.qty_water = neigh.past.qty_water + qty_water_to_debouch;
                            end
                            return true;
                        end,
                        'inter_con_oce'
                    );
                end;
            end
        }
    }
}
```

Figura 41 – Definição do modelo comportamental do ambiente mundo (processo de desaguamento) na linguagem de modelagem *TerraME*.

O modelo temporal do *environment world_env* consiste de três pares evento/mensagem, onde são executados respectivamente o agente de evaporação (*agent_sun*), o agente de chuva (*agent_rain*) e o agente de desaguamento (*agent_debouch*), conforme ilustrado na Figura 42.

```
world_env = Environment {
    (...)
    time_sun = Timer {
        Pair {
            Event { time = SIMULATION_TIME_INITIAL, period = 1, priority = 0 },
            Message {
                function( event )
                    world_env.ag_sun:setTrajectoryStatus(true);
                    world_env.ag_sun:execute(event);
                    ocean_cs:synchronize(); atmosphere_cs:synchronize();
                    return true;
                end
            }
        }
    },
    time_rain = Timer {
        Pair {
            Event { time = SIMULATION_TIME_INITIAL, period = 1, priority = 0 },
            Message {
                function( event )
                    world_env.ag_rain:setTrajectoryStatus(true);
                    world_env.ag_rain:execute(event);
                    atmosphere_cs:synchronize(); continent_cs:synchronize();
                    return true;
                end
            }
        }
    },
    time_debouch = Timer {
        Pair {
            Event { time = SIMULATION_TIME_INITIAL, period = 1, priority = 0 },
            Message {
                function( event )
                    world_env.ag_debouch:setTrajectoryStatus(true);
                    world_env.ag_debouch:execute(event);
                    continent_cs:synchronize(); ocean_cs:synchronize();
                    return true;
                end
            }
        }
    }
}
```

Figura 42 – Definição do modelo temporal do ambiente mundo na linguagem de modelagem *TerraME*.

Uma vez implementado, o modelo é então executado, e os resultados desta simulação, que permitem visualizar como se dá o fluxo de água em cada um dos *environments* (continente, oceano e atmosfera), podem ser observados na Figura 43. O resultado permite observar a emergência de padrões a partir de regras simples, como no caso do comportamento dos autômatos celulares, que possuem como regra distribuir a água entre seus vizinhos, considerando apenas a diferença de altitude.

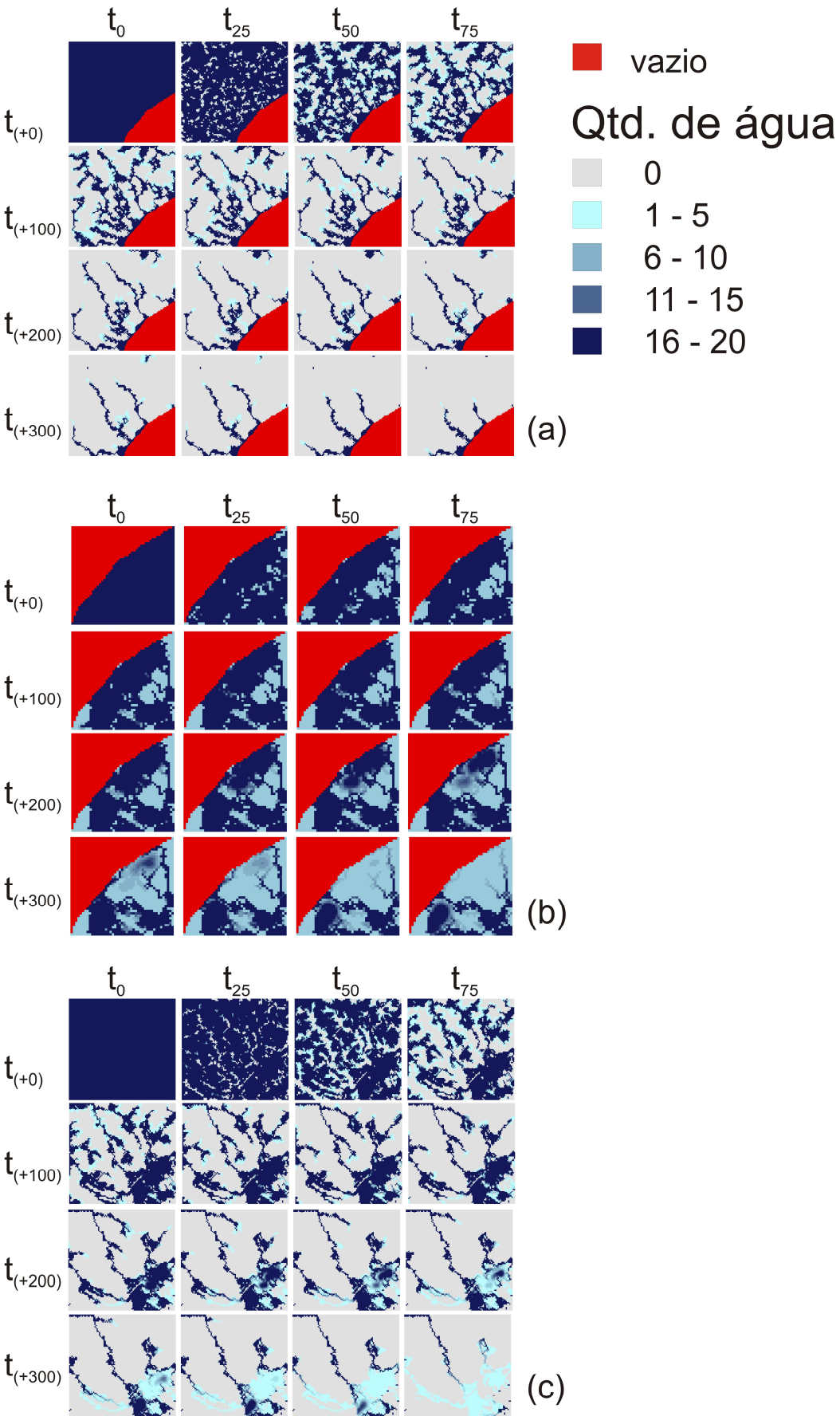


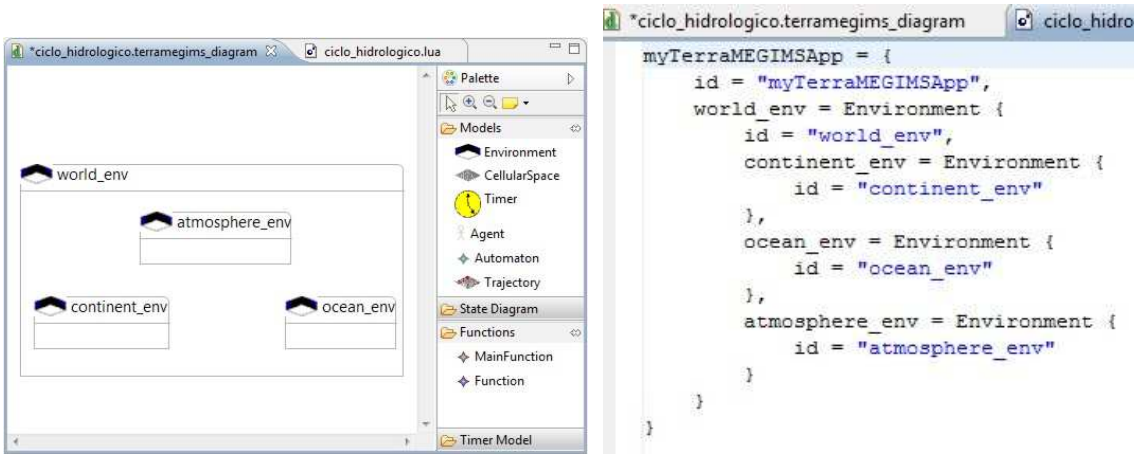
Figura 43 – Resultado da simulação do modelo: (a) continente; (b) oceano; (c) atmosfera

Uma vez implementado o modelo utilizando diretamente a linguagem *TerraME*, este é implementado novamente utilizando a interface gráfica do *TerraME GIMS*, como apresentado na Seção 6.1.4.

6.1.4 Modelagem utilizando o *TerraME GIMS*

Com o intuito de avaliar a geração de código *TerraME* a partir da interface gráfica disponibilizada pelo *TerraME GIMS* (apresentado no Capítulo 5 – Seção 3), o mesmo modelo do ciclo hidrológico é implementado novamente. Para isto, o primeiro passo necessário é a criação de um novo projeto e arquivo *TerraME GIMS* no *Eclipse*.

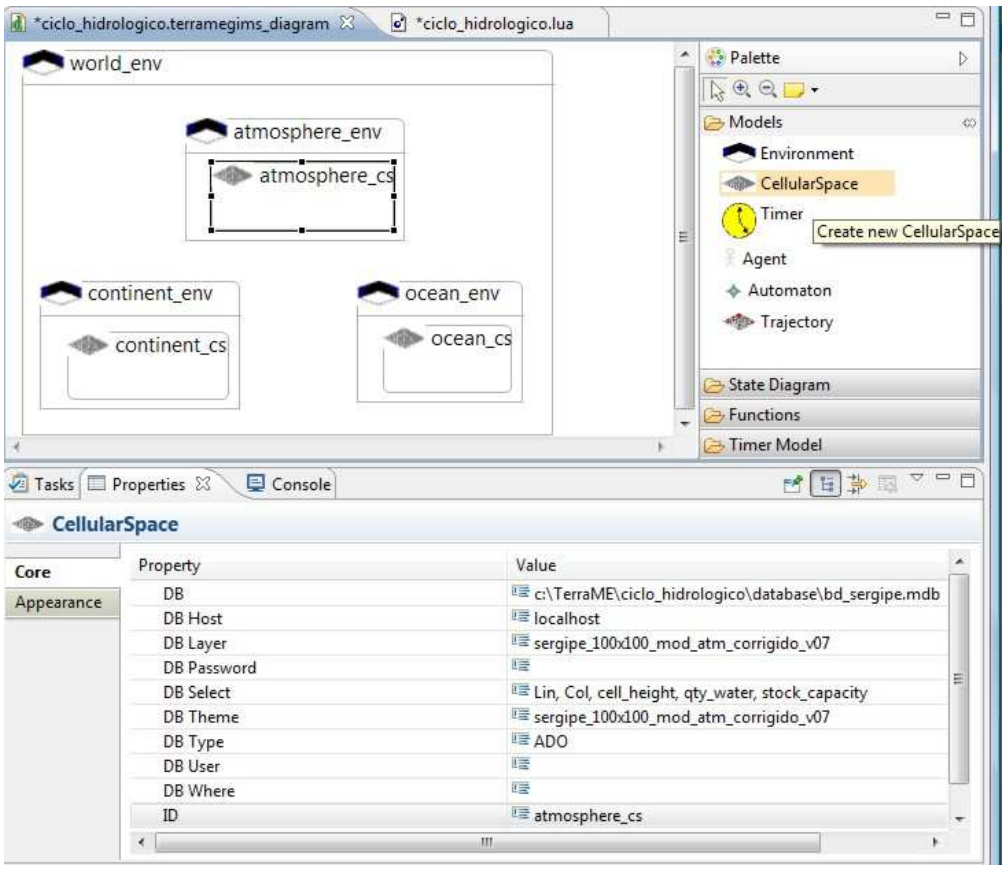
Um modelo em *TerraME GIMS* é desenvolvido a partir de uma aplicação *TerraME GIMS* (*TerraMEGIMSApp*), que pode ser constituída de funções e *environments*. O modelo do ciclo hidrológico, como mostrado na Figura 44 (a), pode ser construído graficamente a partir do *environment world_env*, que por sua vez contém os *environments continent_env*, *ocean_env*, e *atmosphere_env*, sendo então gerado o código *TerraME* correspondente, como pode ser observado na Figura 44 (b).



(a) (b)

Figura 44 – Definição dos ambientes utilizando o *TerraME GIMS*: (a) representação gráfica; (b) código *TerraME* correspondente.

De modo semelhante, o modelo espacial de cada um dos *environments* pode ser definido a partir da interação com componentes da interface gráfica de usuário e o código correspondente é gerado conforme os atributos definidos para o espaço celular correspondente, como apresentado na Figura 45.



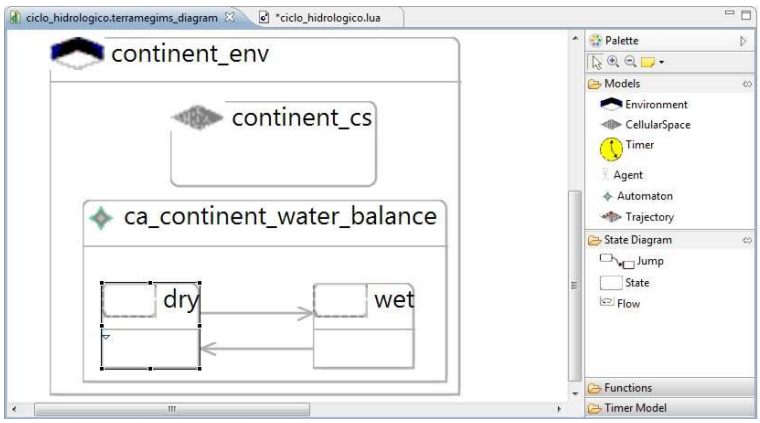
(a)

```
world_env = Environment {
    id = "world_env",
    continent_env = Environment {
        ...
    },
    ocean_env = Environment {
        ...
    },
    atmosphere_env = Environment {
        id = "atmosphere_env",
        atmosphere_cs = CellularSpace {
            id = "atmosphere_cs",
            dbType = "ADO",
            host = "localhost",
            database = "c:\TerraME\ciclo_hidrologico\database\bd_sergipe.mdb",
            layer = "sergipe_100x100_mod_atm_corrigido_v07",
            theme = "sergipe_100x100_mod_atm_corrigido_v07",
            select = { "Lin", " Col", " cell_height", " qty_water", " stock_capacity" }
        },
    }
    ...
}
```

(b)

Figura 45 – Definição do modelo espacial do ambiente atmosfera: (a) representação gráfica; (b) código *TerraME* correspondente.

Por sua vez, o modelo comportamental pode ser construído por meio de autômatos celulares (*Automaton*) e agentes (*Agent*). Estes podem ter sua estrutura definida através da representação gráfica de seus estados e regras de transição de estados, como apresentado na Figura 46.



(a)

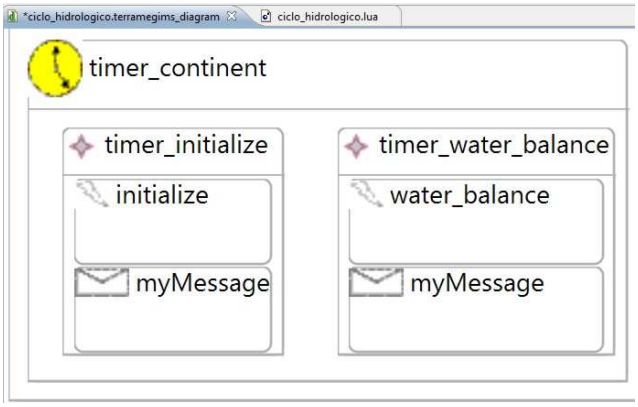
ciclo_hidrologico.terramegims_diagram | ciclo_hidrologico.lua

```
continent_env = Environment {
    id = "continent_env",
    continent_cs = CellularSpace {
        id = "continent_cs"
    },
    ca_continent_water_balance = Automaton {
        id = "ca_continent_water_balance",
        dry = State {
            id = "dry",
            myJump = Jump {
                id = "myJump",
                function(event,automaton,cell)
                end,
                target = "wet"
            }
        },
        wet = State {
            id = "wet",
            myJump = Jump {
                id = "myJump",
                function(event,automaton,cell)
                end,
                target = "dry"
            }
        }
    },
```

(b)

Figura 46 – Definição do modelo comportamental do ambiente continente (processo de escoamento superficial) através de um autômato celular: (a) representação gráfica; (b) código *TerraME* correspondente.

O modelo temporal também pode ser graficamente construído a partir da criação de pares evento/mensagem e o respectivo código *TerraME* gerado, como ilustrado na Figura 47.



(a)

ciclo_hidrologico.terramegims_diagram | ciclo_hidrologico.lua

```
timer_water_balance = Pair {
    id = "timer_water_balance",
    Event {
        id = "water_balance", time = "1.0", period = "1.0", priority = "0"
    },
    Message {
        id = "myMessage",
        function(event)
        print("-> " .. event:getTime() .. " - continent - time_water_balance_con");
        continent_env.con_ca_water_balance:build();
        continent_env.con_ca_water_balance:setTrajectoryStatus(true);
        continent_env.con_ca_water_balance:execute(event);
        continent_cs:synchronize();
        return true;
        end
    }
}
```

(b)

Figura 47 – Definição do modelo temporal do ambiente continente: (a) representação gráfica; (b) código *TerraME* correspondente.

A visualização gráfica dos diversos componentes que constituem o modelo, incluindo modelos espaciais, modelos temporais e modelos comportamentais, ajuda a melhor compreender a estrutura do modelo, além de tornar mais fácil sua construção, na medida em que este se torna mais complexo. Desta forma, é essencial que ferramentas para modelagem visual sejam flexíveis na apresentação dos elementos do modelo e permitam exibir e/ou ocultar aspectos conforme a necessidade do usuário. Como por exemplo, uma visão mais geral do modelo do ciclo hidrológico, pode ser desejada, apresentando os *environments* *continent_env*, *ocean_env* e *atmosphere_env* e os modelos comportamentais que descrevem os processos de fluxo de água entre estes, como apresentada na Figura 48.

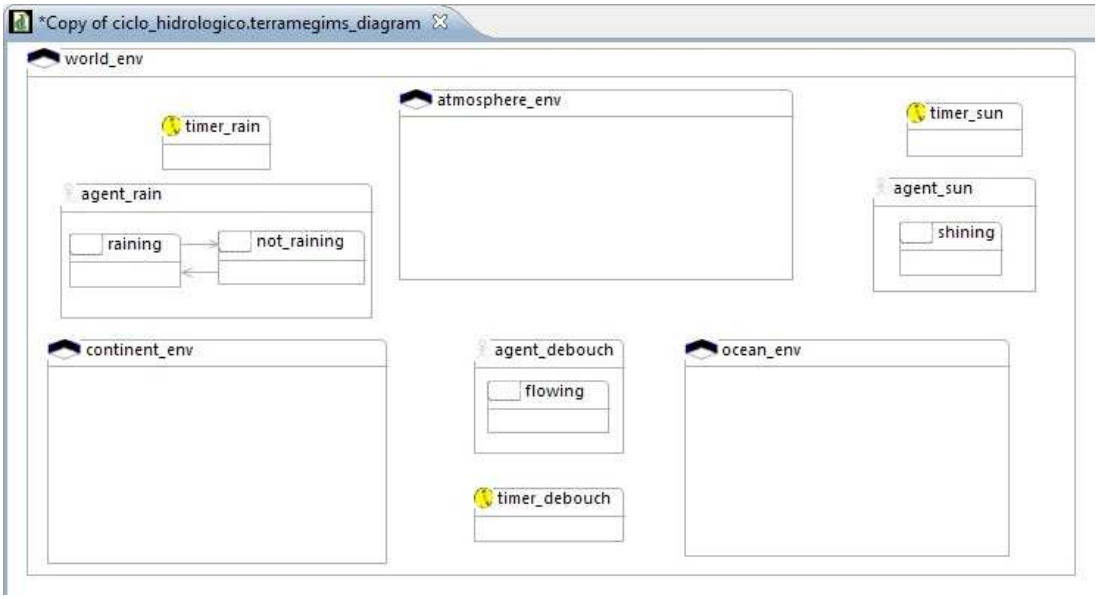


Figura 48 – Visão geral do ambiente mundo.

Uma visualização completa do modelo e de toda a sua estrutura também pode ser apresentada, como ilustrado na Figura 49.

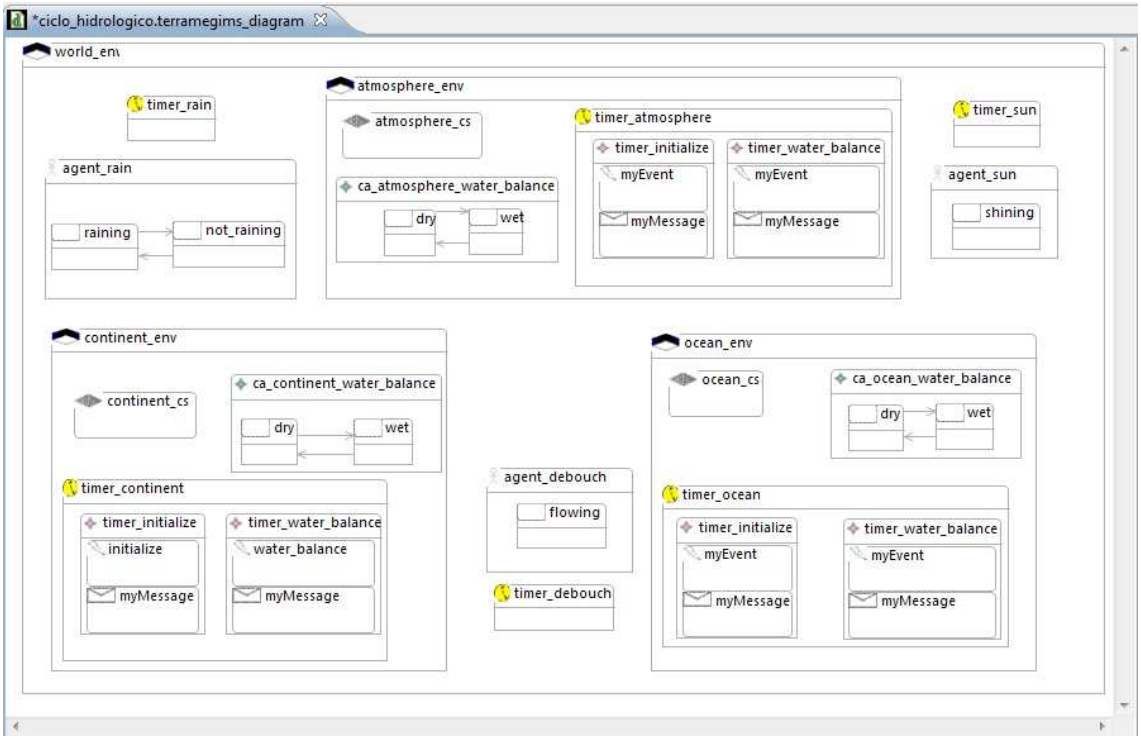


Figura 49 – Visão geral do modelo do ciclo hidrológico

6.2 Avaliação do *TerraME GIMS*

Para validar a interface gráfica foi construído o modelo pedagógico apresentado na Seção 6.1.1 para ilustrar as funcionalidades para modelagem de sistemas ambientais oferecidas pela plataforma *TerraME*. Inicialmente este modelo foi construído utilizando diretamente a linguagem de modelagem *TerraME*, e posteriormente foi implementado a partir da interface gráfica.

Uma vez que as regras que descrevem os processos modelados são as mesmas para as duas versões implementadas, utilizando diretamente a linguagem *TerraME* e por meio da interface gráfica, os resultados obtidos pela simulação também foram os mesmos, conforme apresentado na Figura 43 (Seção 6.1.3). Desta forma, pode-se afirmar que é possível construir toda a estrutura do modelo utilizando a interface gráfica do *TerraME GIMS*.

No entanto, têm-se algumas diferenças quanto à estrutura do código implementado utilizando a linguagem *TerraME* e do código gerado pela interface gráfica. A primeira delas é a falta de flexibilidade oferecida ao usuário no que se refere à organização do código, imposta pela própria natureza de ferramentas de geração automática de código – as quais apresentam certa rigidez quanto à personalização do código gerado. Exemplos disto podem ser observados a partir da utilização de indentação e quebras de linha para organização do código do modelo, que são personalizados conforme a preferência do modelador quando utilizada diretamente a linguagem de programação (Figura 50 (a)), mas apresentam uma estrutura padronizada quando gerados pela ferramenta (Figura 50(b)).

```
timer_ocean = Timer {
    id = "timer_ocean",

    Pair {
        Event { id = "myEvent", time = "0.0", period = "0.0", priority = "0" },
        Message {
            id = "myMessage",
            function(event)
                (...)
            end
        }
}
```

(a)

```
timer_ocean = Timer {
    id = "timer_ocean",
    timer_initialize = Pair {
        id = "timer_initialize",
        Event {
            id = "myEvent",
            time = "0.0",
            period = "0.0",
            priority = "0"
        },
        Message {
            id = "myMessage",
            function(event)
                (...)
            end
        }
}
```

(b)

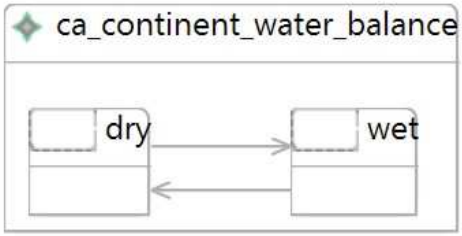
Figura 50 – Flexibilidade quanto à organização do código: (a) código do modelador, (b) código gerado pelo *TerraME GIMS*.

Outro exemplo de limitação relacionada à flexibilidade dos usuários na organização do código do modelo é a restrição quanto à divisão deste em módulos (arquivos) distintos. A atual versão do *TerraME GIMS* não permite customizar a geração de código de forma que o código correspondente a elementos específicos do modelo, como por exemplo *environments*, sejam criados em arquivos distintos.

No entanto, se por um lado a estrutura padronizada de geração de código pode limitar a liberdade do modelador quanto à organização deste, por outro, pode ser também vir a ser um grande benefício para aqueles usuários que não possuem grande experiência em programação e na adoção de práticas como a utilização de endentação e quebras de linha para organizar o código, que contribuem para a boa legibilidade e manutenibilidade do código.

Além disto, outra grande vantagem propiciada pela geração automática de código a partir da representação gráfica do modelo é a diminuição do esforço por parte dos usuários quanto a aspectos sintáticos da linguagem de programação, ou seja, permitir que estes se dediquem mais à tarefa de representar (modelar) os processos do objeto de estudo por meio das ferramentas oferecidas pela plataforma *TerraME* ao invés de se

preocuparem como a descrição computacional (estruturas de código *TerraME*) destas ferramentas e utilizá-las. Para construir, por exemplo, um autômato celular constituído de dois estados (seco e molhado) e duas regras de transição de estados, conforme apresentado na Figura 51, o único esforço necessário por parte do usuário e necessidade de utilização da linguagem de programação *TerraME* será para descrever a parte dinâmica do processo em questão, uma vez que toda a parte estática é gerada automaticamente pelo *TerraME GIMS*.



(a)

```
ca_ocean_water_balance = Automaton {
    id = "ca_ocean_water_balance",
    dry = State {
        id = "dry",
        myJump = Jump {
            id = "myJump",
            function(event,automaton,cell)
                -- código TerraME que o usuário deve implementar para representar o processo
            end,
            target = "wet"
        }
    },
    wet = State {
        id = "wet",
        myJump = Jump {
            id = "myJump",
            function(event,automaton,cell)
                -- código TerraME que o usuário deve implementar para representar o processo
            end,
            target = "dry"
        }
    }
}
```

(b)

Figura 51 – Modelagem visual da parte estática do modelo (a), e geração do código *TerraME* correspondente (b).

O maior benefício da ferramenta para os usuários é, no entanto, oferecer um meio de comunicação que permite transmitir e comunicar conhecimento a partir da representação visual do modelo, ou seja, um nível maior de abstração, que permite visualizar e identificar os elementos que constituem o modelo e os relacionamentos entre estes, de forma mais fácil e intuitiva se comparada à utilização direta de uma linguagem de programação, conforme apresentado na Figura 52.

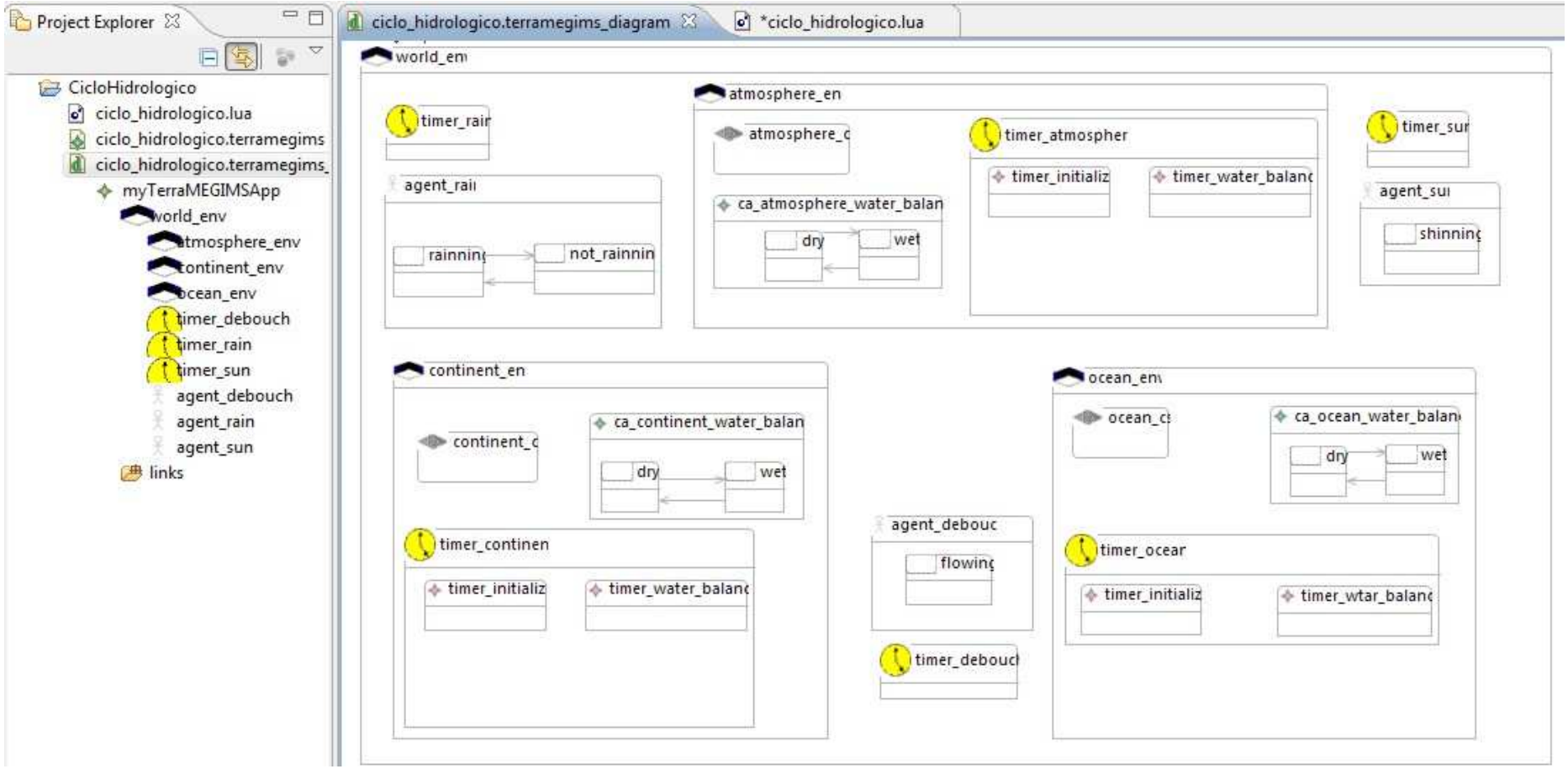


Figura 52 – Visualização dos elementos do modelo e respectivos relacionamentos em forma de árvore e de diagrama.

É possível verificar, analisando a construção de modelos diretamente a partir da linguagem de programação *TerraME* e por meio da interface gráfica do *TerraME GIMS*, que quanto maior a facilidade fornecida pela ferramenta maiores também serão as limitações em termos de flexibilidade para a construção de modelos. Também é possível indicar que quanto maior o nível de abstração utilizado para representação do modelo, ou seja, quanto mais próxima e intuitiva for para o usuário a linguagem adotada para representação do modelo, menor será o tempo necessário para aprendizado e utilização desta. Pode-se observar também que o nível maior de abstração a partir da representação gráfica dos elementos que constituem o modelo permite tornar mais próximos o modelo conceitual e sua implementação, favorecendo os modeladores a manter o foco e esforço em aspectos relacionados ao domínio do problema e na obtenção de uma visão comum em equipes multidisciplinares.

7 CONCLUSÕES

É crescente a demanda por modelos ambientais que permitam melhor compreender a interação entre processos biofísicos e antrópicos e forneçam subsídios para auxiliar pesquisadores, empresários e governantes no processo de tomada de decisões, em busca de intervenções de menor impacto ao meio ambiente. No entanto, a construção de modelos ambientais para esta finalidade é uma tarefa complexa, que exige a utilização de ferramentas computacionais de suporte e envolvimento de uma equipe multidisciplinar.

Apesar da existência de diversas plataformas de modelagem e simulação que permitem a construção de modelos ambientais, como aquelas apresentadas no Capítulo 3, e da utilização destas plataformas em diversos problemas desta natureza, estas apresentam limitações, devido à ausência de recursos necessários e específicos para a modelagem de processos ambientais (tais como modelagem em múltiplas escalas, representação contínua e discreta de processos, relações de vizinhança não isotrópicas) e/ou funcionalidades que auxiliem os usuários na modelagem (tais como construção e apresentação visual do modelo, apresentação visual da estrutura/organização do modelo).

A plataforma *TerraME* (Capítulo 4) foi projetada para atender aos requisitos específicos necessários a uma plataforma para modelagem ambiental, permitindo simular processos dinâmicos com representação espacial explícita. No entanto, apesar de oferecer serviços e estruturas de dados para modelagem a partir da linguagem de programação de alto nível *TerraME*, de mais fácil utilização quando comparada a outras linguagens de propósito geral, tais como C++ e Java, sua utilização ainda é um fator limitante para os potenciais usuários da plataforma, que em geral são especialistas no domínio do problema mas não possuem sólidos conhecimentos sobre algoritmos e técnicas de programação. Assim, a modelagem utilizando a plataforma *TerraME* exige atualmente dos usuários, além do esforço necessário para a representação do sistema em estudo na forma de um modelo, a preocupação quanto a aspectos relacionados à representação deste modelo na linguagem de programação *TerraME*, como por exemplo a adequação às regras sintáticas da linguagem.

Além disto, uma vez que o processo de modelagem cada vez mais é feito por equipes multidisciplinares, outro ponto importante se deve à limitação na utilização de uma linguagem de programação como meio de representação e comunicação do conhecimento acerca do sistema. Para que a colaboração seja possível e eficaz, é fundamental que todos os envolvidos compreendam a mensagem, ou seja, tenham conhecimento da linguagem utilizada. Uma linguagem de fácil assimilação torna a comunicação mais eficiente.

Desta forma, para tornar efetiva a utilização de todos os serviços oferecidos pela plataforma *TerraME* e ampliar sua comunidade de usuários, é essencial oferecer aos usuários um nível maior de abstração para a construção de modelos.

Assim, o próximo estágio em termos de funcionalidades para a construção de modelos ambientais utilizando a plataforma *TerraME* consistia na disponibilização de uma interface visual que permitisse construir modelos a partir da interação com componentes de interface gráfica, tais como ícones, caixas de texto, menus, diagramas.

Neste contexto, este trabalho apresenta como principal contribuição o ambiente de modelagem visual para a plataforma *TerraME*, denominada *TerraME GIMS*. Este ambiente oferece uma interface gráfica que permite aos seus usuários, a partir da interação com componentes de sua interface, construir modelos espaciais dinâmicos para a plataforma *TerraME*. O código *TerraME* correspondente ao modelo é gerado automaticamente. Além de permitir a construção visual do modelo, também é possível visualizar sua estrutura tanto na forma de diagramas quanto na forma de árvore.

Dentre os possíveis desdobramentos desta pesquisa é possível destacar os seguintes: (i) desenvolver os signos que compõem a interface gráfica do *TerraME GIMS*; (ii) prover serviços direcionadas a usuários experientes como funcionalidades de engenharia reversa que permitam a geração automática da representação visual de modelos a partir do código *TerraME*; (iii) prover serviços de simulação interativa; (iv) realizar a avaliação da usabilidade da interface gráfica do *TerraME GIMS* para a construção de modelos espaciais dinâmicos.

BIBLIOGRAFIA

AGUIAR, A. P. D.; CÂMARA, G.; ESCADA, M. Spatial statistical analysis of land-use determinants in the Brazilian Amazonia: exploring intra-regional heterogeneity. **Ecological Modelling**, vol. 209, p. 169-188. 2007.

ALMEIDA, C. M.; BATTY, M.; MONTEIRO, A. M. V.; CÂMARA, G.; SOARES-FILHO, B. S.; CERQUEIRA, G. C.; PENNACHIN, C. L. Stochastic cellular automata modeling of urban land use dynamics: empirical development and estimation. **Computers, Environment and Urban Systems**. 2003. vol. 27, p. 481-509.

ALMEIDA, C. M.; CÂMARA, G.; MONTEIRO, A. M. VIEIRA; SOARES-FILHO, B. S.; CERQUEIRA, G. C. Modelos celulares de dinâmicas espaço-temporais: aplicações em estudos urbanísticos. In: MEIRELLES, M. S. P.; CÂMARA, G.; ALMEIDA C. M. (Org.). **Geomática: Modelos e aplicações ambientais**. 1ª Ed. Brasília, Embrapa Informação Tecnológica: 2007. cap. 9, p. 445-496.

ALMEIDA, R. M., MACAU, E. E. N., FRANCA, H., RAMOS, F. M., CARNEIRO, T. G. S. Simulando padrões de incêndio no Parque Nacional das Emas, Goiás, Brasil. In: SIMPÓSIO BRASILEIRO DE GEOINFORMÁTICA, 2008, Rio de Janeiro, RJ. **Anais do Simpósio Brasileiro de GeoInformática**, 2008. p.10-20.

ALMEIDA, R. M.; MACAU, E. E. N.; RAMOS, F. M.; FRANÇA, H. Modelo de propagação de fogo em incêndios florestais e a teoria de percolação. In: XXXI CONGRESSO NACIONAL DE MATEMÁTICA APLICADA E COMPUTACIONAL. **Anais do XXXI Congresso Nacional de Matemática Aplicada e Computacional**. Belém – PA. 2008b.

ANDRADE, P.R.; MONTEIRO, A. M. V.; CÂMARA, G.; SANDRI, S. Games on Cellular Spaces: How Mobility Affects Equilibrium. In: **Journal of Artificial Societies and Social Simulation (JASS)**. 2009.

BOEHM, B. W. A Spiral Model of Software Development and Enhancement. **Computer**. 1988. vol. 5, pp. 61-72.

BRATLEY, P.; FOX, L. B.; SCHRAGE, E. L. **A guide to Simulation**. 2ª Ed. Springer-Verlag. New York: 1987. Cap. 1.

BURROUGH, P. A. Dynamic Modelling and Geocomputation. In: LONGLEY, P. A.; BROOKS, S. M.; MCDONNELL, R.; MACMILLAN, B. **Geocomputation: A primer**. Chichester, UK, John Wiley & Sons Ltd. 1998. p. 165-191.

CÂMARA, G.; SOUZA, R. C. M.; PEDROSA, B. M.; VINHAS, L.; MONTEIRO, A. M. V.; PAIVA, J. A.; CARVALHO, M. T. M.; GATTASS, M.. TerraLib: Technology in support of GIS Innovation. In: **II Simpósio Brasileiro de Geoinformática**. 2000, São Paulo.

CÂMARA, G.; CARNEIRO, T. G. S.; BEYILACQUA, L. I Curso de Verão Geoma 2007 – Modelagem Dinâmica com TerraME. 2007. Disponível em: < http://www.dpi.inpe.br/geoma/curso_verao/index.php >. Acesso em: agosto de 2008.

CÂMARA, G.; CARNEIRO, AGUIAR, A. P. D.; CARNEIRO, T. G. S.; NETO, P. R.; FEITOSA, F. Course on Environmental Modeling – LUCC. 2008. Disponível em <http://www.dpi.inpe.br/cursos/environmental_modelling/index.html>

CARNEIRO, T. G. S. **Nested-CA: a foundation for multiscale modeling of land use and land change**. 2006. 29 p. (INPE -5522-TDI/519) Tese (Doutorado em Computação Aplicada) - Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, 2006.

CARNEIRO, T. G. S., MARETTO, E. V., CÂMARA, G. Irregular Cellular Spaces: Supporting Realistic Spatial Dynamic Modeling over Geographical Databases In: Simpósio Brasileiro de GeoInformática, Rio de Janeiro (RJ), 2008. **Anais do Simpósio Brasileiro de GeoInformática**. 2008.

CARNEIRO, T. G. S.; LIMA, T. F. M.; FARIA, S. D. TerraLAB – Using Free Software for Earth System Research and Free Software Development. In: Workshop de Software Livre, Porto Alegre (RS). 2009. **Anais do Workshop de Software Livre**. 2009.

CARNEIRO, T. G. S.; CÂMARA, G. A. **An Introduction to TerraME**. INPE/UFOP. versão 1.2. 2009. Disponível em: <www.terralab.ufop.br>

COLLIER, N. **Repast: An Extensible Framework for Agent Simulation**. 2002. Disponível em: <http://www.econ.iastate.edu/tesfatsi/RePastTutorial.Collier.pdf> Acesso: outubro de 2009.

COLLIER, N.; HOWE, T. R.; NORTH, M. J. Onward and Upward: The Transition to RePast 2.0. **Proceedings of the First Annual North American Association for**

Computational Social and Organizational Science Conference. Pittsburgh, PA, USA. 2003.

CONNER, M. M.; EBINGER, M. R.; KNOWLTON, F. F. Evaluating coyote management strategies using a spatially explicit individual-based, socially structured population model, **Ecological Modeling**, vol. 219, November 2008, pages 234-247, ISSN 0304-3800. 2008. Disponível em: <http://www.sciencedirect.com/science/article/B6VBS-4TNXB13-2/2/f52b7e8ce67f10eee60759f9be8d8479>

CROOKS, A. T. **The Repast Simulation/Modelling System for Geospatial Simulation**. UCL Centre from Advanced Spatial Analysis. UCL Working Papers Series. 2007. ISSN: 1467-1298. Disponível em: <http://www.casa.ucl.ac.uk/working_papers/paper123.pdf>

COUCLELIS, H. (1997) .From cellular automata to urban models: new principles for model development and implementation. **Environment and Planning B: planning and design**. London, vol. 24, p. 165-174.

De SOUZA, C.S. (1993) "The Semiotic Engineering of User Interface Languages". **International Journal of Man-Machine Studies**, Vol.39, 1993, pp.753-773.

De SOUZA, C. S., BARBOSA, S. D. J., PRATES, R. O. (2001) "A Semiotic Engineering Approach to User Interface Design". **Journal of Knowledge-Based Systems**, Vol.14, Issue 8, 2001, pp 461-465.

ECLIPSE. **Documentação do Eclipse**. Disponível em: <http://help.eclipse.org/help32/index.jsp>. Acesso:: junho/2009.

ECLIPSE EMF. **Eclipse Modeling Framework Project**. Disponível em: <http://www.eclipse.org/modeling/emf/>. Acesso em: junho/2009.

ECLIPSE EMF. **The Eclipse Modeling Framework (EMF) Overview**. 2005. Documentação do Eclipse. Disponível em: <http://help.eclipse.org/ganymede/index.jsp?topic=/org.eclipse.emf.doc/references/overview/EMF.html>. Acesso em: junho/2009.

FEARNSIDE, P. M.; GRAÇA, P. M. L.; KEIZER, E. W. H.; MALDONADO, F. D.; BARBOSA, R. I.; NOGUEIRA, E. M. Modelagem de desmatamento e emissões de

gases de efeito estufa na região sob influência da Rodovia Manaus-Porto Velho (BR-319). In: **Revista Brasileira de Meteorologia**. 2009.

FILATOVA, T.; van der VEEN, A. (2007). Scales in coastal land use: policy and individual decision-making (an economic perspective). Issues in Global Water System Research. Global Assessments: Bridging Scales and Linking to Policy. C. v. Bers, D. Petry and C. Pahl-Wostl. Bonn, The Global Water System Project. #2: 61-68.

GIBSON, C. C., E. OSTROM, et al. (2000). "The concept of scale and the human dimensions of global change: a survey." Ecological Economics 32(2): 217-239.

GODOY, M. M. G.; SOARES-FILHO, B. S. Modelagem da dinâmica intra-urbana no bairro Savassi, em Belo Horizonte. In: ALMEIDA, C. M.; CÂMARA, G.; VIEIRA, A. M. (org.). **Geoinformação em urbanismo: cidade real x cidade virtual**. São Paulo (SP). Ed. Oficina de Textos: 2007. p. 286-304.

GONTIJO, A. B. **Estudo e modelagem das dinâmicas estruturais de assembléias de formigas tropicais em diferentes escalas ecológicas**. Dissertação (Mestrado em Ecologia de Biomas Tropicais), Universidade Federal de Ouro Preto (UFOP), Ouro Preto. 2009.

GOSLING, J.; JOY, B.; STEELE, G. L. **The Java ™ Language Specification**. Addisson-Wesley, 1996.

HAN J.; PAN X.; XUE H. Sustainability education based on NetLogo modeling environment: Taking climate change simulation as an example. In: **Journal of Communication and Computer**. Sep. 2008, vol. 5, n. 9. ISSN: 1548-7709. Disponível em: < http://isoshu.cqvip.com/journal/default.aspx?url=/qk/88584X/200809/index.shtml>

HANNON, B.; RUTH, M. **Dynamic Modeling**. Springer-Verlag, New York. 2 Ed. ISBN: 978-0-387-98868-8. 2001.

HOWE, T. R.; COLLIER, N. T.; NORTH, M. J.; PARKER, M.T.; VOS, J. R. (2006). Containing Agents: Contexts, Projections and Agents. In: Sallach, D., Macal, C.M., and North, M. J. (eds.). **Proceedings of the Agent 2006 Conference on Social Agents: Results and Prospects**, University of Chicago and Argonne National Laboratory, Chicago, IL. Disponível em: <http://agent2007.anl.gov/2006procpdf/Agent_2006.pdf>

IERUSALIMSCHY, R.; FIGUEIREDO, L. H.; CELES, W. Lua – an extensible extension language, **Software: Practice & Experience**. n. 26. 1996. p. 635-652.

KHAN, S.; YUFENG, L.; AHMAD, A. Analysing complex behaviour of hydrological systems through a system dynamics approach. **Environmental Modelling & Software**. 2009. v. 24, p. 1363-1372.

LANA, R. M. **Modelos Dinâmicos Acoplados para a Simulação da Ecologia do vetor Aedes aegypti**. Dissertação (Mestrado em Ecologia de Biomas Tropicais), Universidade Federal de Ouro Preto (UFOP), Ouro Preto. 2009.

LASSER, C.; OMOHUNDRO, S. M. **The Essential Star-lisp Manual**. Thinking Machines Technical Report. 1986. Disponível em: <http://omohundro.files.wordpress.com/2009/03/omohundro86_the_essential_starlisp_manual.pdf>

LIMA, T. F. M.; CARNEIRO, T. G. S.; FARIA, S. D. Desenvolvimento de uma Plataforma Gráfica para a Descrição de Modelos de Sistemas Ambientais. In: SIMPÓSIO BRASILEIRO DE GEOINFORMÁTICA, 2008, Rio de Janeiro. **Anais do X Simpósio Brasileiro de Geoinformática**, 2008, p. 121-126.

LUNA, F.; STEFANSSON, B. Economic Simulations in Swarm: Agent-Based Modelling and Object Oriented Programming. Series: **Advances in Computational Economics**, vol. 14. Kluwer Academic Publishers, Norwell, MA, USA. 2000. ISBN: 978-0-7923-8665-0.

LUNA, FRANCESCO; PERRONE, A. **Agent-Based Methods in Economics and Finance: Simulations in Swarm**. Kluwer Academic Publishers, Dordrecht, Londres. 2001.

MARIA, A. Introduction to Modelling and Simulation. In: **Proccedings of the 1997 Winter Simulation Conference**. 1997.

MARTINEAU, Y.; SAUGIER, B. A process-based model of old field succession linking ecosystem and community ecology. **Ecological Modeling**, v. 204, March 2007, pages 399-419, ISSN 0304-3800. 2007.

MINAR, N.; BURKHART, R.; LANGTON, C.; ASKENAZI, M. 1996. **The Swarm simulation system: a toolkit for building multi-agent simulations**. Working Paper

96-06-042, Santa Fe Institute, Santa Fe. Disponível em: < http://www.swarm.org/archive/overview.ps>

MORE, B.; DEAN, D.; GERBER, A.; WAGENKNECHT, G.; VANDERHEYDEN, P. **Eclipse Development using the Graphical Editing Framework and the Eclipse Modeling Framework**. IBM Redbooks, 2003.

MOREIRA, E.; AGUIAR, A. P.; COSTA, S. S.; Câmara, G. Spatial relations across scales in land change models. In: X SIMPÓSIO BRASILEIRO DE GEOINFORMÁTICA. 2008, Rio de Janeiro, 2008. **Anais do X Simpósio Brasileiro de Geoinformática**, 2008.

MOREIRA, E.; COSTA, S.; AGUIAR, A. P.; CÂMARA, G.; CARNEIRO, T. G. S. Dynamic coupling of multiscale land change models. **Landscape Ecology**, vol. 24, n. 9, p. 1183-1194, ISSN: 1572-9761. 2009. Disponível em: <http://www.springerlink.com/content/nl452l1736u1141n/>

NOBRE, C. Ciência do sistema terrestre e a sustentabilidade da vida no planeta. **Revista Pesquisa FAPESP**, julho, 2008. Disponível em: <http://www.revistapesquisa.fapesp.br/pdf/revolucao_genomica/nobre.pdf>

NORTH, M. J.; HOWE, T. R.; COLLIER, N. T.; AND VOS, R. J. Repast Simphony Development Environment. In: Macal, C.M.; North, M. J.; Sallach, D. (eds.), **Proceedings of the Agent 2005 Conference on Generative Social Processes, Models and Mechanisms**. Argonne National Laboratory and The University of Chicago, Oct. 13-15. 2005a.

NORTH, M. J.; HOWE, T. R.; COLLIER, N. T.; VOS, R. J. Repast Simphony Runtime System. In: C.M. Macal; M. J. North; and D. Sallach (eds.), **Proceedings of the Agent 2005 Conference on Generative Social Processes, Models and Mechanisms**. Argonne National Laboratory and The University of Chicago, Oct. 13-15. 2005b.

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**. ACM (Janeiro): New York, NY. 2006.

NORTH, M. J.; TATARA, E.; COLLIER, N. T.; OZIK, J. (2007). Visual Agent-based Model Development with Repast Simphony. **Proceedings of the Agent 2007 Conference on Complex Interaction and Social Emergence**. Argonne National

Laboratory, Argonne, IL USA. 2007. Disponível em: <http://agent2007.anl.gov/2007pdf/Paper%2015%20--%20North%20Agent%202007_ALM_MJN_alm2_chngsaccpt_NEW.pdf>

NOVAES, A. G. (1981). **Modelos em planejamento urbano, regional e de transportes.** Edgard Blucher. São Paulo (SP).

PAPERT, S. **Mindstorms: Children, Computers, and Powerful Ideas**, Basic Books. 1980.

PEDROSA, B. M.; CÂMARA, G. Modelagem Dinâmica e Sistemas de Informações Geográficas. In: MEIRELLES, M. S. P.; CÂMARA, G.; ALMEIDA C. M. (Org.). **Geomática: Modelos e aplicações ambientais**. 1ª Ed. Brasília, Embrapa Informação Tecnológica, 2007. ISBN 978-85-7383-386-7. cap. 5, p. 235-280.

PEIRCE, C.S. (1931-1958). Collected Papers. Edição brasileira: Semiótica. São Paulo, Ed.Perspectiva (coleção estudo, n.46), 1977.

PEREIRA, L. M. Modelagem Hidrológica Dinâmica Distribuída para Estimativa do Escoamento Superficial em uma Microbacia Urbana. Dissertação (Mestrado em Sensoriamento Remoto), INPE, São José dos Campos. 2008.

PIMENTA, P.; COELHO, A.; COSTA, S. S.; MOREIRA, E ; AGUIAR, A. P.; CÂMARA, G.; ARAÚJO, R.; RIBEIRO, A . Land change modeling and institutional factors: heterogeneous rules of territory use in the Brazilian Amazonia. In: X SIMPÓSIO BRASILEIRO DE GEOINFORMÁTICA. 2008, Rio de Janeiro, 2008. **Anais do X Simpósio Brasileiro de Geoinformática**, 2008.

PITT, W. C.; BOX, P. W.; KNOWLTON, F. F. An individual-based model of canid populations: modeling territoriality and social structure. **Ecological Modeling**, vol. 166, pp. 109-121, ISSN 0304-3800. 2003. Disponível em: <http://www.sciencedirect.com/science/article/B6VBS-48Y6RCS-4/2/d912cd896a7ad92068d4870b74e65522>

POWELL, A. **Model with the Eclipse Modeling Framework**. 2004. Disponível em: <http://www-128.ibm.com/developerworks/library/os-ecemf1/>. Acesso: junho de 2009.

RENNÓ, C. D.; SOARES, J. V. Conceitos básicos de modelagem hidrológica. In: MEIRELLES, M. S. P.; CÂMARA, G.; ALMEIDA C. M. (org.). **Geomática: Modelos**

e aplicações ambientais. 1ª Ed. Brasília, Embrapa Informação Tecnológica, 2007. cap. 11, p. 529-556.

REPAST3. Repast 3. Repast Organization for Architecture and Design. Disponível em: <http://repast.sourceforge.net/repast_3/ >. Acesso: junho de 2009.

RIVIERES, J.; WIEGAND, J. Eclipse: A platform for integrating development tools. **IBM Systems Journal**, 43(2). 2004.

RIVIERES, J.; BEATON, W. **Eclipse Technical Platform Overview**. 2006. Disponível em: <http://www.eclipse.org/articles/Whitepaper-Platform-3.1/eclipse-platform-whitepaper.html>. Acesso: junho/2009.

ROAD. Repast Organization for Architecture and Design. Chicago, IL. 2009. Disponível em: <http://repast.sourceforge.net/>

RODRIGUES, H. O.; SOARES-FILHO, B. S.; COSTA, W. L. S. Dinamica EGO, uma plataforma para modelagem de sistemas ambientais. In: **Simpósio Brasileiro de Sensoriamento Remoto**, 13. Instituto Nacional de Pesquisas Espaciais. 2007.

RODRIGUEZ-AGUILAR, J. A.; PINYOL, F.; N., X.; LOPEZ-SANCHEZ, M. **State of the Art of Software tools for agent-based simulations**. Research Report D30.1. julho. 2001. Disponível em: <http://www.maia.ub.es/~maite/papers/State-of-the-artMAS.pdf>

SILVA, J. X. Geoprocessamento em Estudos Ambientais: Uma Perspectiva Sistêmica. In: MEIRELLES, M. S. P.; CÂMARA, G.; ALMEIDA C. M. (org.). **Geomática: Modelos e aplicações ambientais**. 1ª Ed. Brasília, Embrapa Informação Tecnológica, 2007. ISBN 978-85-7383-386-7. cap. 1, p. 21-53.

SILVEIRA, A. L. L. Ciclo Hidrológico e Bacia Hidrográfica. In: TUCCI, C. E. M. (Org.). **Hidrologia: Ciência e Aplicação**. 3ª Ed. Porto Alegre: Editora da UFRGS. 2003. cap. 2, p. 35-51.

SILVESTRINI, R. A.; SOARES-FILHO, B. S.; ALENCAR, A. A. C.; RODRIGUES, H. O.; ASSUNÇÃO, R. M.; MENDONZA, E. Modelo probabilístico de espalhamento de fogo: Aplicação para a região do Xingu. In: Simpósio Brasileiro de Sensoriamento Remoto, 2009, Natal. **Anais do XIV Simpósio Brasileiro de Sensoriamento Remoto**, Natal, Brasil, 25-30 abril, 2009, INPE, p. 5459-5466.

SIMÕES, A. L. A.; PINHEIRO, H. D.; GONÇALVES, J. C. S. I.; GIORGETTI, M. F.; PORTO, R. M. Estudo da propagação de ondas de cheia em canais por meio das

equações de Saint-Venant: uma abordagem indutiva com o auxílio de um software gratuito. In: **International Conference on Engineering and Computer Education**, 2009.

SOARES-FILHO, B. S.; CERQUEIRA, G. C.; PENNACHIN, C. L. DINAMICA – a stochastic cellular automata model designed to simulate the landscape dynamics in an Amazonian colonization frontier. **Ecological Modelling**. 2002. v. 154, n. 3, p. 217–235.

SOARES-FILHO, B. S.; ALENCAR, A.; NEPSTAD, D.; CERQUEIRA, G.; DIAZ, M. C. V.; RIVERO, S.; SOLÓRZANOS, L.; VOLL, E. Simulating the response of land-cover changes to Road paving and governance along a major Amazon highway: the Santarém-Cuiabá corridor. **Global Change Biology**. 2004. v. 10, n. 5, p. 745-764.

SOARES-FILHO, B. S. Dinamica Project. Disponível em: <http://www.csr.ufmg.br/dinamica/>. Acesso: outubro de 2009.

SOARES-FILHO, B. S., RODRIGUES, H. O., COSTA, W. L. Modeling Environmental Dynamics with Dinamica EGO. Centro de Sensoriamento Remoto / Universidade Federal de Minas Gerais. Belo Horizonte, Brasil. 2009. ISBN: 978-85-910119-0-2. Disponível em <http://www.csr.ufmg.br/dinamica>. Acesso: julho de 2010.

STEYAERT, L. T. A perspective on the state of environmental simulation modeling. In: GOODCHILD, M. F.; PARKS, B.O.; STEYAERT, L.T. **Environmental modeling with GIS**. New York: Oxford University Press, 1993. p. 16-30.

STROUSTRUP, B. **The Design and Evolution of C++**. Addison-Wesley. 1994.

SWARM. Swarm. Swarm Development Group, Santa Fe, New Mexico. Disponível em: <http://www.swarm.org/> . Acesso em: julho de 2009.

TATARA, E.; NORTH, M. J.; HOWE, T. R.; COLLIER, N. T.; VOS, J. R. An Introduction to Repast Modeling by Using a Simple Predator-Prey Example. **Proceedings of the Agent 2006 Conference on Social Agents: Results and Prospects.** Argonne National Laboratory, Argonne, IL USA. Disponível em: <http://www.agent2006.anl.gov/2006procpdf/Tatara_Agent_2006.pdf>

TECCOMM. Apresentação: "Visão Geral da Plataforma Eclipse. Disponível em: <http://server.teccomm.les.inf.puc-rio.br/SeminarioAcompanhamento/03_VisaoGeralPlataformaEclipse.pdf>. Acessado em: junho/2009.

TISUE, S.; WILENSKY, U. NetLogo: A Simple Environment for Modeling Complexity. In: **International Conference on Complex Systems**. Boston, MA, May 16-21, 2004.

TISUE, S.; WILENSKY, U. NetLogo: Design and implementation of a multi-agent modeling environment. In: **SwarmFest 2004**, Ann Arbor, MI, Swarm Development Group. 2004b.

TUCCI, C. E. M. Hidrologia: Ciência e Aplicação. In: TUCCI C. E. M. (Org.). **Hidrologia: ciência e aplicação**. 3ª Ed. Porto Alegre: Editora da UFRGS. 2002. v. 1, cap. 1, p. 25-33.

TUCCI, C. E. M. Interceptação. In: TUCCI C. E. M. (Org.). **Hidrologia: ciência e aplicação**. 3ª Ed. Porto Alegre: Editora da UFRGS. 2002b. v. 1, cap. 6, p. 243-252.

TUCCI, C. E. M. Fundamentos do escoamento não-permanente. In: TUCCI C. E. M. (Org.). **Hidrologia: ciência e aplicação**. 3ª Ed. Porto Alegre: Editora da UFRGS. 2002c. v. 1, cap. 10, p. 373-389.

TUCCI, C. E. M. Escoamento superficial. In: TUCCI C. E. M. (Org.). **Hidrologia: ciência e aplicação**. 3ª Ed. Porto Alegre: Editora da UFRGS. 2002d. v. 1, cap. 11, p. 391-441.

TUCCI, C. E. M. Escoamento em rios e reservatórios. In: TUCCI C. E. M. (Org.). **Hidrologia: ciência e aplicação**. 3ª Ed. Porto Alegre: Editora da UFRGS. 2002e. v. 1, cap. 12, p. 443-483.

TUCCI, C. E. M; BELTRAME, L., F. S. (2002). Evaporação e Evapotranspiração. In: TUCCI C. E. M. (Org.). **Hidrologia: ciência e aplicação**. 3ª Ed. Porto Alegre: Editora da UFRGS. v. 1, cap. 7, p. 254-287.

TURNER, M. G.; GARDNER, R. H.; O'NEILL, R. V. 2001. **Landscape Ecology in Theory and Practice: pattern and process**. 2001. Springer-Verlag. New York. Cap. 3.

VENTANA. Vensim software – linking systems thinking to powerful by dynamic models. Disponível em: <http://www.vensim.com/software.html>. Acesso em: nov/2009.

VENISM. **Vensim PLE User's Guide**. version 4. Ventana Software Inc. Hardard, MA, USA. 2009.

WARD, A. D.; Trimble, S. W. **Environmental Hydrology**. 2a Ed. ISBN 1-56670-616-5. Lewis Publishing, 2004.

WILENSKY, U. **NetLogo Fire Model**. Center for Connected Learning and Computer-Based Modeling. Northwestern University, Evanston, IL. 1997. Disponível em: <http://ccl.northwestern.edu/netlogo/models/Fire>.

WILENSKY, U. **StarLogoT**, Center for Connected Learning and Computer-Based Modeling. Northwestern University, Evanston, IL. 1997b. Disponível em: <http://ccl.northwestern.edu/cm/starlogot/>

WILENSKY, U. **NetLogo (computer software and user manual)**, Center for Connected Learning and Computer-Based Modeling, Northwestern University. 1999. Disponível em: <http://ccl.northwestern.edu/netlogo>

WILENSKY, U. 2001. Modeling Nature's Emergent Patterns with Muti-agent Languages. In: SENDOVA, E.; NEUWIRTH, E. (eds.), **Proccedings of the Eighth European Logo Conference**, Virtech, Linz, Austria, 2001. pp. 43-54.

YU, C.; CHEN, C.; LIN, C.; LIAW, S. Development of a system dynamics model for sustainable land use management. **Journal of the Chinese Institute of Engineers**. 2003. v. 26, n. 5, pp. 607-618.