Extensão da Abordagem de Desenvolvimento Dirigido a Modelos de Simulações com Agentes (MDD4ABMS) para Suportar Portabilidade de Simulações

Fernando Santos[1], **Rafael Tenfen**[1]

[1] Departamento de Engenharia de Software
Universidade do Estado de Santa Catarina (UDESC)

fernando.santos@udesc.br, rafaeltenfen.rt@gmail.com

Abstract. *This paper presents an extension of the model-driven development for agent-based modeling and simulation approach (MDD4ABMS) to support portability of simulations. The extension consists of an automatic code generator for the Repast simulation platform. With this extension, the same simulation model can be ported to both NetLogo and Repast. A case study was performed considering the* SugarsScape *simulation. The results obtained with the execution of the generated NetLogo and Repast simulations were similar, showing evidence of the viability of portability of simulations specified with MDD4ABMS.*

Resumo. *Este artigo apresenta uma extensão da abordagem de desenvolvimento dirigido a modelos de simulações com agentes (MDD4ABMS) para suportar a portabilidade de simulações. A extensão consiste de um gerador automático de código para a plataforma de simulação Repast. Com isto, um mesmo modelo de simulação pode ser portado tanto para NetLogo quanto para Repast. Um estudo de caso foi realizado considerando a simulação* SugarScape. *Os resultados obtidos com a execução das simulações Repast e NetLogo geradas foram similares, evidenciando a viabilidade da portabilidade de simulações especificadas com a abordagem MDD4ABMS.*

1. Introdução

Simulações computacionais têm sido cada vez mais utilizadas como ferramenta para estudar fenômenos e apoiar a tomada de decisão. Por exemplo, simulações de tráfego auxiliam no projeto de infraestruturas de transporte, e simulações de evacuação auxiliam no projeto de estádios onde espectadores devem evacuar o local rapidamente em emergências.

Simulações com agentes fazem uso de agentes simulados para reproduzir e investigar estes fenômenos [Klügl e Bazzan 2012]. Neste paradigma de simulação é possível focar nos indivíduos (agentes) e nos efeitos que o comportamento e as interações destes indivíduos podem causar no sistema — por exemplo, efeitos do isolamento de indivíduos infectados durante a propagação de uma epidemia.

Existem várias plataformas para desenvolver simulações com agentes, como por exemplo NetLogo, Repast e MASON [Klügl e Bazzan 2012]. Cada uma destas plataformas ou adota uma linguagem de programação própria, ou utiliza alguma linguagem de programação existente (ex. Java). A demanda por este conhecimento de lógica de programação, aliada a ausência de ferramentas fáceis de usar e que forneçam blocos

de construção produtivos para o desenvolvimento de simulações, podem desencorajar a adoção do paradigma de simulações com agentes [Macal 2016].

Recentemente foi proposta uma abordagem dirigida a modelos para desenvolver simulações com agentes, denominada MDD4ABMS [Santos et al. 2018]. Esta abordagem permite modelar simulações em um nível mais alto de abstração, através elementos recorrentemente usados em simulações com agentes, e sem a necessidade de programação. Além de uma linguagem e ferramenta de modelagem, a MDD4ABMS disponibiliza geração automática do código fonte da simulação, para que o projetista possa executar a simulação modelada. Entretanto, atualmente a MDD4ABMS suporta geração de código apenas para a plataforma de simulação NetLogo. Sendo assim, apesar de que modelos especificados com a abordagem MDD4ABMS serem independentes de plataforma, atualmente a simulação é portável somente para NetLogo.

Este artigo apresenta uma extensão da MDD4ABMS para suportar a portabilidade das simulações. A extensão consiste de um gerador de código para a plataforma Repast. Com isto, um mesmo modelo de simulação pode ser portado tanto para NetLogo quanto para Repast, permitindo que desenvolvedores com alguma experiência em Repast também possam executar as simulações especificadas através da MDD4ABMS. Um estudo de caso foi realizado considerando a simulação *SugarScape*, que foi especificada através da MDD4ABMS e teve o código fonte NetLogo e Repast gerados automaticamente. Os resultados obtidos com a execução destas simulações foram similares, evidenciando a viabilidade da portabilidade de simulações especificadas com MDD4ABMS.

O restante deste artigo está organizado da seguinte forma. A seção 2 apresenta a fundamentação teórica sobre simulações com agentes e desenvolvimento dirigido a modelos. A seção 3 apresenta a extensão desenvolvida neste artigo para suportar a portabilidade de simulações. Em seguida, a seção 4 descreve o estudo de caso realizado. Por fim, a seção 5 apresenta as conclusões e trabalhos futuros.

2. Fundamentação Teórica

2.1. Simulações com Agentes

Agentes, são sistemas inteligentes com duas características importantes: possuem autonomia para decidir por si próprios sobre o que necessitam fazer a fim de satisfazer seus objetivos; e são capazes de interagir com outros agentes, não apenas através de trocas de dados mas também de se envolver com práticas sociais humanas diárias, como negociação, coordenação e cooperação. Agentes podem tomar decisões a partir de deduções lógicas, modo que assemelha ao raciocínio humano, ou através da combinação de dedução e outro mecanismo de tomada de decisão [Wooldridge 2009].

Simulações com agentes é um paradigma de simulação que usa agentes simulados para reproduzir, entender e predizer fenômenos [Klügl e Bazzan 2012]. O paradigma de simulações com agentes oferece benefícios em relação a outros paradigmas de simulação. Um dos benefícios é a possibilidade de analisar e observar o fenômeno em estudo em dois níveis: microscópico e macroscópico. No nível microscópico pode-se estudar diferentes estratégias de tomada de decisão dos agentes. Pode-se também considerar, de forma natural, a heterogeneidade dos agentes, já que este paradigma de simulação foca nos indivíduos. Já no nível macroscópico pode-se estudar os resultados gerados a partir das

ações e interações dos agentes entre si e com o ambiente [Klügl 2008]. Por fim, os agentes podem incorporar técnicas de inteligência artificial (por exemplo aprendizagem, redes neurais) para aprimorar comportamentos e decisões. Segundo [Klügl e Bazzan 2012], para desenvolver uma simulação com agentes é necessário especificar três elementos: um conjunto de agentes autônomos; as interações entre os agentes e ambiente, responsáveis por produzir a saída geral do sistema; e o ambiente simulado, que contém todos os demais elementos de simulação, como recursos e outros objetos sem comportamento ativo.

A Figura 1 apresenta um exemplo de simulação de propagação de doenças desenvolvida na plataforma de simulação com agentes NetLogo. Nesta simulação há dois tipos de agentes: nativos e imigrantes. A propagação da doença ocorre a partir das interações entre estes agentes, podendo ser por contato ou proximidade. Os parâmetros de transmissão e recuperação da doença dependem do tipo de agente (heterogeneidade), sendo que os nativos estão mais suscetíveis a adquirir e propagar a doença. O ambiente é especificado como uma grade por onde os agentes podem se movimentar. Ao usar simulações com agentes para estudar o fenômeno da propagação de doenças é possível considerar, de forma simples, a questão da regionalidade das interações de transmissão da doença. Isto porque as interações ocorrem em função da localização dos agentes, logo, agentes que estão muito distantes (ex. em comunidades regionais diferentes) tendem a não interagirem. Esta vantagem do uso de simulações com agentes para simular a propagação de doenças já foi apontada na literatura [Eisinger e Thulke 2008]. A partir da saída da simulação (gráficos e visualização dos agentes) pode-se observar variações ao longo do tempo na população de agentes suscetíveis, infectados, e recuperados da doença.

Há várias plataformas para realizar simulações com agentes. Entre as principais estão: NetLogo, Repast e MASON [Klügl e Bazzan 2012]. O NetLogo foi projetado para

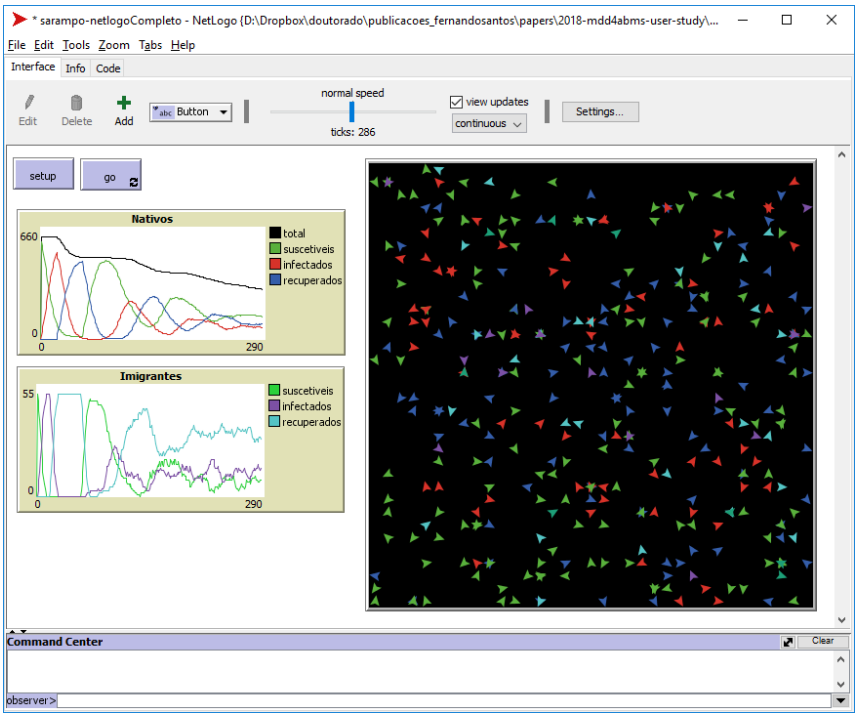

Figura 1. Exemplo de simulação com agentes: propagação de doenças.

o usuário final, tendo uma interface para trabalhar com os parâmetros da simulação, e uma extensa biblioteca de modelos existentes, prontos para executar. Além disto, NetLogo oferece uma linguagem de programação própria para desenvolver simulações. A plataforma Repast, por sua vez, permite desenvolver simulações utilizando programação Java. Ao usar Java, o desenvolvedor conta com uma biblioteca de classes comuns associadas à implementação de simulação baseada em agentes, com foco em ciências sociais, contando com ferramentas úteis, como análise de redes sociais. MASON também é uma plataforma baseada em Java que visa facilitar a programação de simulações em larga escala para ganhar desempenho, sendo muito atrativo para aplicações que demandam performance. Como pode-se verificar, todas estas plataformas demandam alguma habilidade de programação em diferentes linguagens. Isto pode desencorajar a adoção de simulações com agentes. Por exemplo, uma pessoa interessada em desenvolver uma simulação pode não conseguir por não possuir conhecimento de programação. Há casos em que o interessado pode já ter encontrado uma simulação pronta, e queira apenas estendê-la para avaliar novos comportamentos ou parâmetros. No entanto, mesmo se a pessoa já tem conhecimento em alguma plataforma de simulação, ela pode não conseguir estender e executar a simulação caso esteja desenvolvida em uma plataforma diferente daquela(s) que conhece.

2.2. Desenvolvimento Dirigido a Modelos de Simulações com Agentes

A engenharia de software fornece suporte para o desenvolvimento de software profissional por meio de técnicas de especificação, projeto e evolução de sistemas. Uma destas técnicas é o desenvolvimento dirigido a modelos (MDD—*model-driven development*). Em MDD procura-se encontrar abstrações específicas de domínio e disponibilizá-las para modelagem dos sistemas. O objetivo é construir modelos que sejam simultaneamente abstratos e formais. A abstração, nestes modelos, não significa imprecisão, mas compactação e redução à essência [Stahl et al. 2006]. Em MDD, o modelo de um sistema desempenha papel chave no processo de desenvolvimento, pois a partir dele o código fonte é gerado automaticamente. Essa geração automática de código-fonte melhora a eficiência, produtividade, confiabilidade, reusabilidade e portabilidade.

A MDD4ABMS (*model-driven development for agent-based modeling and simulation*) é uma abordagem MDD para desenvolver simulações com agentes [Santos et al. 2018]. Esta abordagem disponibiliza um metamodelo de simulações com agentes, e uma linguagem específica de domínio para modelar simulações com agentes. Ambos foram elaborados a partir de um processo de análise de domínio que buscou identificar os aspectos recorrentes de simulações com agentes para disponibilizá-los como elementos de modelagem. A MDD4ABMS conta com uma ferramenta de modelagem, onde os elementos estão disponíveis em uma paleta de componentes, e o projetista pode utilizá-los para especificar a simulação com agentes. Esta ferramenta também oferece geração automática de código para que o projetista possa executar a simulação e estudar os resultados. A Figura 2 apresenta o modelo de simulação de propagação de doenças explicado na seção anterior especificado através da abordagem MDD4ABMS.

Uma das limitações da MDD4ABMS é a geração de código unicamente para a plataforma de simulação NetLogo. Ou seja, apesar de que na MDD4ABMS a modelagem é independente de plataforma de simulação, atualmente a simulação é portável somente para NetLogo. Este artigo apresenta uma proposta para que as simulações especificadas com MDD4ABMS sejam portáveis para a plataforma Repast.

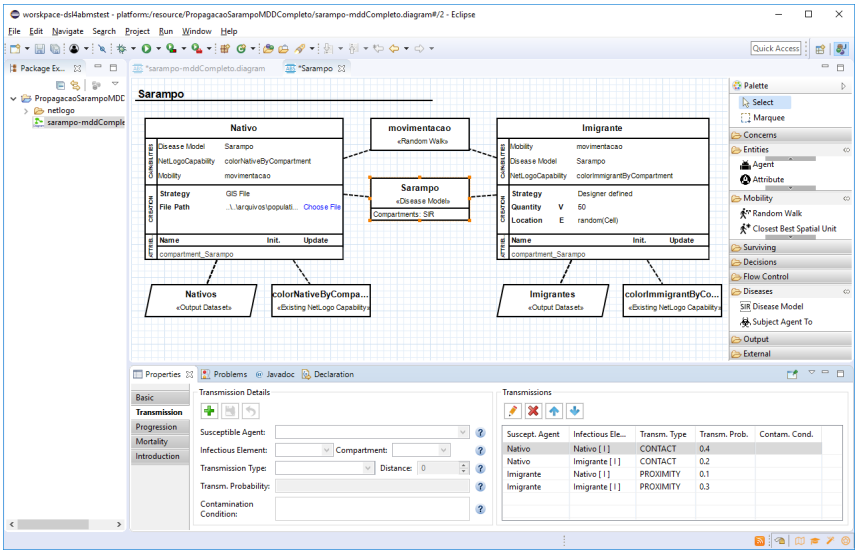

Figura 2. Simulação de propagação de doenças modelada em MDD4ABMS

3. Portabilidade de Simulações com Agentes em MDD4ABMS

Para oferecer portabilidade para a Repast, foi desenvolvido um gerador de código para esta plataforma. Este gerador é formado por regras de produção de código, que transformam os elementos do metamodelo MDD4ABMS (agentes, ambiente, etc) para elementos e blocos de comandos da plataforma Repast. As regras foram descritas utilizando Xpand,[1] uma linguagem para especificação de *templates* de geração de código.

Dentre as regras de geração de código desenvolvidas, destaca-se a que produz o código da classe Java de um agente a partir de um elemento *Agent* do metamodelo. A Figura 3 apresenta a definição desta regra. Inicialmente a regra gera os elementos Java *package* e *import*. Na linha 4 é gerada a declaração da classe Java para o agente, e nas linhas seguintes são gerados os atributos, o contrutor, e *getters*/*setters*. Na linha 8 é gerado o método *step*, que contém o comportamento do agente a ser invocado pela plataforma Repast a cada passo da simulação. O método *step*, por sua vez, invoca as habilidades especificadas para o agente, que são implementadas por métodos Java gerados pela regra Xpand ativada na linha 11. Outras regras são ativadas para completar o código gerado com métodos para movimentação do agente (identificando as melhores posições para movimento) e métodos para coletar dados do agente vinculados às saídas da simulação.

Para gerar o ambiente simulado, foram especificadas regras Xpand que geram código Java e XML.[2] Isto é necessário pois em Repast a especificação das dimensões do ambiente (linhas e colunas) é feita em XML, enquanto que os atributos das unidades espaciais são definidos em Java. Para os elementos do metamodelo que definem as saídas da simulação foram especificadas regras Xpand que geram código XML de acordo com a especificação adotada em Repast para definição de gráficos. Por fim, o gerador de código foi integrado na ferramenta de modelagem da abordagem MDD4ABMS, de forma que o projetista pode facilmente ativar a geração de código Repast.

[1] `http://www.eclipse.org/modeling/m2t/?project=xpand`
[2] Extensible Markup Language

```
1  «DEFINE AgentClasses (mm::Agent agent) FOR mm::AgentBasedSimulation»
2      «EXPAND getPackageName FOR this»
3      «EXPAND importsAgentClass ((Agent) agent) FOR this»
4      public class «EXPAND getClassName ((Agent) agent) FOR this» {
5        «EXPAND agentAttributes ((Agent) agent) FOR this»
6        «EXPAND agentConstructor ((Agent) agent) FOR this»
7        «EXPAND agentGettersSetters ((Agent) agent) FOR this»
8        «EXPAND agentStep ((Agent) agent) FOR this»
9        «EXPAND agentDie ((Agent) agent) FOR this»
10       «EXPAND randomAgentAtributte ((Agent) agent) FOR this»
11       «EXPAND createAgentCapabilities ((Agent) agent) FOR this»
12       «EXPAND bestSpotClass ((Agent) agent) FOR this»
13       «EXPAND getContentAgentExternalCapability ((Agent) agent) FOR this»
14       «EXPAND outPutDataSetMethods ((Agent) agent) FOR this»
15     }
16 «ENDDEFINE»
```

Figura 3. Regra Xpand para gerar a classe Java do agente Repast

4. Estudo de Caso

Para demonstrar a portabilidade desenvolvida nesse trabalho, foi gerado o código fonte Repast e NetLogo a partir de um mesmo modelo. A simulação selecionada é a *Sugarscape* [Epstein e Axtell 1996], onde os agentes estão em um ambiente simulado que contém recursos (açúcar) distribuído em seu espaço. Cada agente necessita coletar açúcar para transformar em energia e sobreviver, possuindo habilidades para se movimentar e retirar o açúcar do ambiente. A cada passo da simulação o agente identifica e se move para a posição contendo maior quantidade de açúcar, e morre quando sua energia decai a zero.

A Figura 4 apresenta o modelo do agente da simulação *Sugascape*, especificado através da linguagem de modelagem provida pela abordagem MDD4ABMS. O agente, denominado *Individual*, possui atributos para definir o alcance da visão, metabolismo, e a quantidade de açúcar (energia) que dispõe. Parâmetros são especificados para inicializar estes atributos e criar os agentes. O ciclo de vida do agente é especificado por uma habilidade de sobrevivência, e a sua movimentação por uma habilidade de mobilidade. A coleta de açúcar é especificada por uma habilidade de colheita de açúcar. Por fim, a especificação define saídas para observar a quantidade de agentes ao longo da simulação, a visão e metabolismo médios destes agentes, e a distribuição da riqueza (açúcar).

O código fonte NetLogo e Repast deste modelo foi gerado e executado para veri-

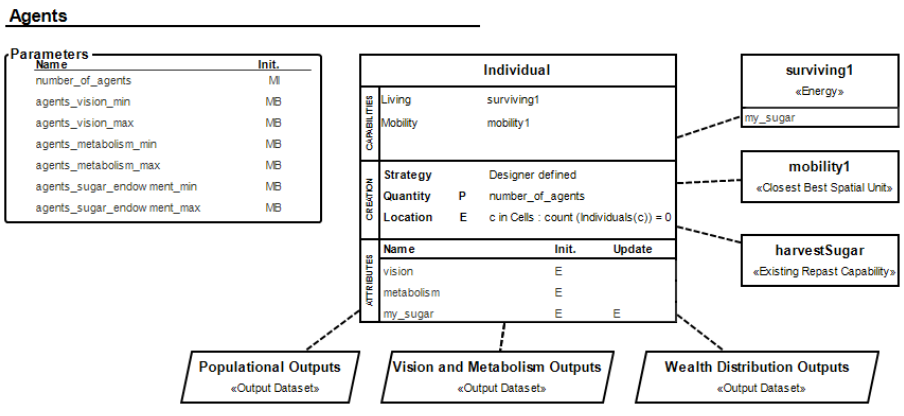

Figura 4. Especificação em MDD4ABMS do agente da simulação *Sugarscape*

ficar a similaridade dos resultados produzidos. A portabilidade será obtida com sucesso caso os resultados das simulações Repast e NetLogo sejam similares quando executados com os mesmos valores de parâmetros. As Figuras 6 e 5 apresentam o resultado da execução da simulação por 100 passos em Repast e NetLogo, respectivamente.

O resultado da execução evidencia a similaridade do comportamento dos agentes. Nos gráficos *populational output* verifica-se, nas duas plataformas, haver mortalidade dos agentes que não conseguem coletar açúcar suficiente para sobreviver no início da simulação. Também verifica-se, através dos gráficos de saída *vision and metabolism*, que os agentes sobreviventes são aqueles que possuem maior alcance de visão e baixo metabolismo. A distribuição de riqueza entre os agentes sobreviventes também é similar, como mostram os gráficos *wealth distribution*. Também observa-se similaridade no comportamento de movimentação dos agentes: nas duas plataformas eles se moveram para as posições do ambiente com maior concentração de açúcar. Por fim, em ambas as platafor-

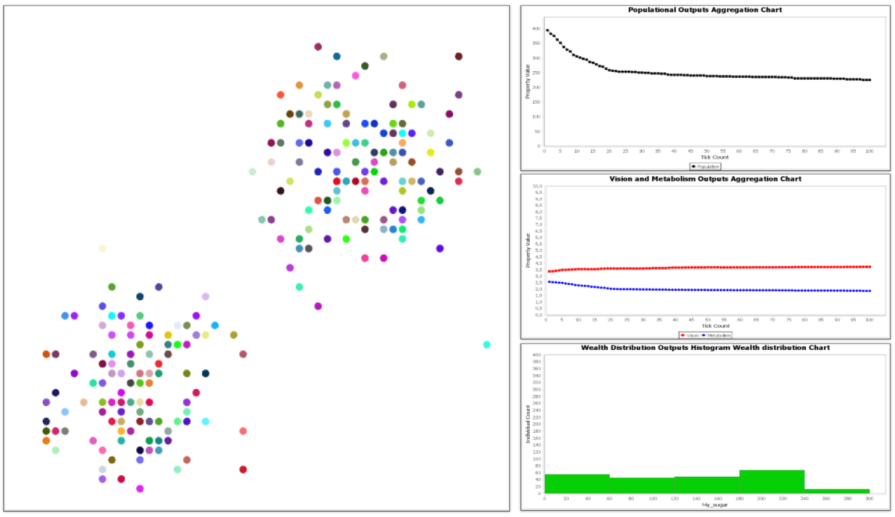

Figura 5. Simulação *Sugarscape* em Repast

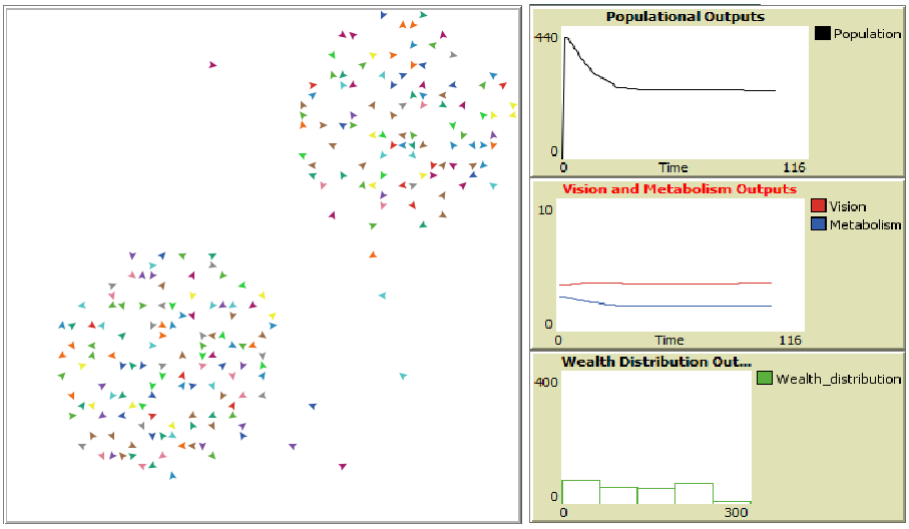

Figura 6. Simulação *Sugarscape* em NetLogo

mas, pode-se observar que os agentes foram criados no ambiente simulado (à esquerda nas figuras) e que os mesmos elementos gráficos foram criados para exibir as saídas da simulação (à direita nas figuras). Isto evidencia que elementos estruturais (ambiente, agentes, e saídas da simulação) são portados com sucesso para as duas plataformas.

De modo geral, os resultados obtidos com a execução da simulação nestas plataformas evidenciam que modelos de simulação contendo ambiente do tipo grade, com gráficos de saída, e agentes com as habilidades de sobrevivência, mobilidade, atualização de atributos, e habilidade definida externamente são portáveis para NetLogo e Repast.

5. Conclusão

Este trabalho estendeu a abordagem MDD4ABMS para suportar a portabilidade de modelos de simulação para duas plataformas: NetLogo e Repast. A portabilidade foi obtida por meio de um novo gerador de código para Repast. Essa portabilidade pode ser útil em meios acadêmicos e profissionais, pois oferece maior liberdade aos interessados em desenvolver simulações com agentes, fomentando a adoção do paradigma de simulações com agentes. Um estudo de caso foi conduzido considerando a simulação *Sugarscape*, e os resultados evidenciaram que, a partir de um mesmo modelo de simulação, os códigos fonte NetLogo e Repast produzem resultados similares.

Apesar da portabilidade para a plataforma Repast proposta nesse trabalho, o gerador de código desenvolvido suporta apenas ambientes simulados do tipo grade de duas dimensões. Portanto, uma sugestão de trabalho futuro é estender o gerador de código para tratar os outros tipos de ambiente previstos na abordagem MDD4ABMS. Além disso, a abordagem MDD4ABMS contempla outras habilidades dos agentes, tais como aprendizado, máquinas de estado, e habilidades referentes a propagação de doenças. Essas habilidades não foram tratadas neste trabalho, sendo outra sugestão de trabalho futuro.

Referências

Eisinger, D. e Thulke, H.-H. (2008). Spatial pattern formation facilitates eradication of infectious diseases. *Journal of Applied Ecology*, 45(2):415–423.

Epstein, J. e Axtell, R. (1996). *Growing Artificial Societies*. Brookings Institution Press, Washington D.C.

Klügl, F. (2008). A validation methodology for agent-based simulations. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 39–43. ACM.

Klügl, F. e Bazzan, A. L. (2012). Agent-based modeling and simulation. *AI Magazine*, 33(3):29–29.

Macal, C. M. (2016). Everything you need to know about agent-based modelling and simulation. *Journal of Simulation*, 10(2):144–156.

Santos, F., Nunes, I., e Bazzan, A. L. (2018). Model-driven agent-based simulation development: A modeling language and empirical evaluation in the adaptive traffic signal control domain. *Simulation Modelling Practice and Theory*, 83:162–187.

Stahl, T., Voelter, M., e Czarnecki, K. (2006). *Model-driven software development: technology, engineering, management*. John Wiley & Sons, Inc.

Wooldridge, M. (2009). *An introduction to multiagent systems*. John Wiley & Sons.