SIMULATING INNOVATION ADOPTION BEHAVIOR: LESSONS LEARNED FOR MODELERS AND PROGRAMMERS

Christian Stummer[(a)], Elmar Kiesling[(b)]

[(a)]Department of Business Administration and Economics, Bielefeld University, Germany
[(b)]Institute of Software Technology and Interactive Systems, Vienna University of Technology, Austria

[(a)]christian.stummer@uni-bielefeld.de, [(b)]elmar.kiesling@tuwien.ac.at

ABSTRACT
To adopt or not to adopt an innovation is a question that is ultimately answered by individual (prospective) customers. Their behavior is of practical relevance because it drives the market success of new products or services and it also constitutes an interesting area for academic research. In the course of a research project at the University of Vienna we have developed an agent-based simulation to investigate this topic. During the initial months, we reviewed numerous tools (i.e., frameworks and modeling environments) for this purpose. In this paper we share experiences we made in this respect as well as later on when implementing the simulation tool.

Keywords: agent-based modeling, innovation adoption behavior, frameworks and modeling environments, lessons learned

1. INTRODUCTION

The prosperity and long-term survival of many firms hinge on their ability to systematically develop new products and introduce them into market successfully. Both challenges require considerable amounts of resources, which is why practitioners have a strong interest in the projection of an innovation's potential market diffusion. Agent-based simulation can capture the complex diffusion process of an innovation on the macro-level as a result of relatively simple micro-level interactions between heterogeneous individuals (who, for example, exchange information within their social network through word-of-mouth); for a recent review of agent-based diffusion models confer Kiesling et al. (2012). This notion was the starting point for a research project on quantitatively simulating and modeling the diffusion of innovations ("QuaSiMoDI"). The endeavor was financed by the Austrian Research Fund and ran from 2008 to 2011.

The simulation model that resulted from the project contributes to innovation diffusion research in that it covers all phases of the purchasing process (ranging from receiving initial information to post-purchase product experiences), takes into account initial adoption as well as repeat purchases, allows for several suppliers, accounts for temporal as well as spatial aspects, and considers heterogeneous consumer preferences with respect to multiple product attributes.

Furthermore, emphasis has been placed on illustrating the applicability of our work by referring to a real product (i.e., a second generation fuel made of biomass that is currently under development at the Vienna University of Technology) for a particular market (i.e., Austria). Results therefore may also be useful for practitioners interested in this particular technology, because simulation experiments for several scenarios (each with its own strategy for price, communication, and roll-out) were based on real data. Also policy-makers could benefit from such simulations that enable them to assess the impact of diverse (e.g., fiscal) measures to further the diffusion of biofuels and contribute to environmental objectives. For descriptions of previous versions of the simulation approach see Kiesling et al. (2009), Kiesling et al. (2010), and Günther et al. (2011).

In addition to researchers and practitioners, a third group of stakeholders may benefit from experiences gained in the QuaSiMoDI project, namely modelers and programmers who are about to embark on a similar project. It is particularly them this paper is targeting at.

In the remainder we will therefore elaborate on modeling and implementation issues. First, we provide an overview of alternative frameworks that have been considered as a basis for the implementation of our agent-based simulation model (Section 2). Next, we will describe the platform and tools actually used in the implementation of QuaSiMoDI (Section 3). Then the architecture of our software implementation is outlined (Section 4). Finally, we mention some general lessons learned from the research project that may be of value for modelers and programmers starting a similar endeavor (Section 5).

2. SURVEY OF AGENT-BASED SIMULATION TOOLS

In recent years, the incursion of agent-based approaches in many scientific disciplines has entailed the development of increasingly sophisticated software-platforms for agent-based modeling and simulation. Today, a modeler selecting a platform for the implementation of an agent-based model is therefore

faced with an abundant range of programming languages, libraries, frameworks, and modeling environments to choose from. In the following, we outline the basic types of available options.

2.1. Programming languages

Of course, it is feasible to implement agent-based models with "plain" general purpose programming languages rather than relying on specialized software tools. Early agent-based models were typically implemented independently following this approach (Gilbert 2002). Today, programming the whole simulation software "from scratch" still appears to be a relatively common approach, even though it leads to duplication of efforts and forces modelers working on different models to repeatedly implement the same basic functionality and algorithms. This process is error-prone, may lead to code that is not easily accessible, and impedes verification of the implementation. Object-oriented languages such as Java or C++ are typically used because core concepts like encapsulation, inheritance, and abstraction fit the agent-based modeling paradigm well. Types of agents are implemented as classes; particular agents are instances (i.e., objects) of these classes that have an internal state: agents' interactions with one another and their environment are implemented as methods of the agent classes.

Somewhat less common approaches build agent-based simulations on top of computational mathematics systems such as Mathematica (Wolfram Inc. 2012) or Matlab (Math-Works 2012), procedural languages (e.g., StarLogo, cf. Resnick 1996), functional languages (Legéndi et al. 2009), or spreadsheet software (Macal and North 2007).

2.2. Libraries and toolkits

Specialized libraries and toolkits that provide dedicated facilities for agent-based simulation offer modelers a number of significant advantages over implementing a model from scratch. First, they provide standard mechanisms that are frequently required in agent-based modeling, such as scheduling, event handling, random number generation, network modeling, logging, visualization, and analysis. As a consequence, the resulting code can be more compact, accessible and easier to verify than custom implementations that involve large amounts of "boilerplate" code. By providing ready-made building blocks, standardized libraries can assist modelers and ideally save them time, effort, and energy.

2.3. Modeling environments

While libraries may assist modelers with only limited programming skills, they still require sufficient fluency in the underlying programming language. Modeling environments, by contrast, provide an entire graphical model building interface and allow modelers to assemble building blocks visually or with very limited syntax. They may therefore alleviate this limitation or require no programming at all. Such environments include, for example, Repast S (repast.sourceforge.net), StarLogo (education.mit.edu/starlogo), Eclipse Agent Modeling Framework (www.eclipse.org/amp), NetLogo (ccl.northwestern.edu/netlogo), and Anylogic (www.xjtek.com/anylogic). The main disadvantage of all-encompassing modeling environments is that they may impose assumptions upon the model and limit the modeler's ability to control detailed aspects of the simulation.

2.4. Prior Reviews

Several authors have reviewed available libraries and environments for agent-based simulation approaches in the past. In an early survey, Gilbert (2002) provide a brief overview of the toolkits available at that time and compare the state of development of software tools for agent-based simulation to the early stages of development of statistical software. Tobias and Hofmann (2004) evaluate free Java-libraries for social agent-based simulation, comparing nineteen different characteristics across the four platforms taken into account, and conclude that the Repast environment (North, Collier, and Vos 2006) was the most advanced of the libraries at the time of the review. Railsback et al. (2006) review four main platforms (NetLogo, Mason, Repast, Swarm) and compare them by implementing a template "StupidModel" at various levels of sophistication in each of them. In total, they discuss sixteen intentionally simplified template models, and provide full specifications for all of them. Isaac (2011) refines these template models and provides implementations in Python, which the authors deem highly readable and more compact than implementations in other languages. Castle and Crooks (2006) examine eight simulation platforms, focusing particularly on evaluating geospatial capabilities. The most extensive survey to date was conducted by Nikolai and Madey (2009). The authors compare five characteristics of 53 toolkits, viz. programming language, operating system support, type of license, primary domain for which the toolkit is intended, and types of support available to the user.

We can conclude this section by asserting that several powerful tools are available to the model builder today. Table 1 summarizes the main contenders considered for the implementation of our innovation diffusion model.

3. PLATFORM AND TOOLS IN QUASIMODI

Several criteria were considered in the selection of tools for the implementation of QuaSiMoDI. First, because the simulation was deployed on a high-performance computing cluster, a platform-independent solution that could be run on various operating systems (Windows, Mac OS X, Linux) was required. Java-based frameworks offer significant advantages in this respect, because the resulting simulation program is portable and can easily be deployed on any computing platform without recompiling the code. Furthermore, the

Table 1: Selected agent-based simulation frameworks

Framework	Website	Language(s)	License	Reviewed in
AnyLogic (Garifullin, Borshchev, and Popkov 2007)	www.xjtek.com/	UML-RT; Java	Proprietary	Castle and Crooks (2006); Nikolai and Madey (2009)
Ascape (Parker, 2001; Inchiosa, 2002)	ascape.sourceforge.net	Java	BSD	Gilbert (2002); Nikolai and Madey (2009)
MASON (Luke et al., 2004)	www.cs.gmu.edu/~eclab/projects/mason/	Java	Academic free, open source	Castle and Crooks (2006); Railsback et al. (2006); Nikolai and Madey (2009)
NetLogo (Tisue and Wilensky, 2004)	ccl.northwestern.edu/netlogo/	NetLogo language	Freeware, not open source	Castle and Crooks (2006); Railsback et al. (2006); Nikolai and Madey (2009)
RePast (v 1-3) (North, Collier, and Vos, 2006)	repast.sourceforge.net/repast_3/index.html	Java (RepastJ), Python (RepastPy), C++, .net (Repast.net)	BSD	Gilbert (2002); Tobias and Hofmann (2004); Castle and Crooks (2006); Railsback et al. (2006); Nikolai and Madey (2009)
Repast S (North et al., 2005)	repast.sourceforge.net/repast_simphony.html	Java, Groovy	BSD	Nikolai and Madey (2009)
StarLogo	education.mit.edu/starlogo	StarLogo language	Freeware, not open source	Gilbert (2002); Castle and Crooks (2006)
Swarm (Minar et al., 1996)	www.swarm.org	Objective C, Java	GPL	Gilbert (2002); Tobias and Hofmann (2004); Castle and Crooks (2006); Nikolai and Madey (2009)

simulation returns consistent results irrespective of the underlying computing architecture, which is by no means guaranteed when natively compiled code is used. Moreover, almost all available Java-based frameworks can be easily complemented with any of the wide array of software libraries available for the Java programming language. As Java is the main programming language for many frameworks (e.g., 42% of the frameworks reviewed by Nikolai and Madey 2009), the number of available options fulfilling the first criterion is large.

Second, the continuous time approach we chose for QuaSiMoDI requires appropriate discrete event mechanisms, i.e., means for maintaining and processing a list of scheduled events. (Note that in such an approach new events can be scheduled for any (future) point on a continuous timeline, which is why scheduling mechanisms for events within the same "time period" as in discrete time approaches become dispensable.) Because most frameworks are based on a discrete time approach and unfold their full potential only in a discrete time setting, the number of candidate platforms was significantly reduced when this requirement was taken into account.

From the remaining options, we finally chose MASON (Luke et al. 2004), a fast discrete-event multi-agent simulation core written in Java that also provides a fast Mersenne Twister (Matsumoto and Nishimura 1998) implementation for pseudo-random number generation. MASON is open source, lightweight, and can be run without a graphical user interface or visualization on a headless server. It also provides checkpointing capabilities and allows for simulation runs to be dynamically migrated across platforms.

The simulation was implemented in Java SE6 using several additional libraries and tools as summarized in Table 2. The list includes a number of standard tools, specialized Java libraries that provide functionality required in the simulation, and common tools for statistical analysis of results and automation of the simulation process as outlined in the following sections.

3.1. Basic Java tools

The first group of tools used in the implementation consists of Apache Maven, Apache Commons and Apache Log4j, XStream and jUnit. We used Apache Maven to manage builds and dependencies of the various Java libraries.

Verification of micro-level mechanisms is crucial in agent-based simulations, because implementation errors cannot easily be detected and traced in the simulation's emergent macro-level output. We therefore conducted extensive unit tests of all major model components and mechanisms on the micro-level with jUnit.

Table 2: Platform, libraries, and tools used

Component	Website	Purpose
Java SE6	java.sun.com	Implementation of the simulation
MASON	www.cs.gmu.edu/~eclab/projects/mason	Agent-based simulation core
Apache Maven	maven.apache.org	Build management
jUnit	www.junit.org/	Unit and integration testing
CERN Colt library	acs.lbl.gov/software/colt	Probability distributions, statistics
JUNG	jung.sourceforge.net	social network generation and visualization
GeoTools GIS toolkit	geotools.codehaus.org	Geospatial model, shapefile reading, distance calculations
Apache Commons, Log4j	www.apache.org	Utility classes, logging of output and simulation results
XStream	xstream.codehaus.org	XML deserialization for parameter and configuration files
Perl	www.perl.org	Automation of parameter sweeps and analysis process
Gnu R	www.r-project.org	Analysis of results; graphs

The recording of detailed information results in the generation of a considerable amount of data. Therefore, a flexible logging facility that provides mechanisms to selectively activate or deactivate output at runtime and that is executed in a separate thread that is independent of the main simulation program can provide significant performance benefits (particularly on multi-core computers). Apache Log4j fulfills these requirements and was used to produce both comma separated output for analysis and optional human readable textual log files.

Finally, we aimed for a highly generic and versatile simulation that is fully configurable at runtime. To this end, all model inputs as well as the configuration of parameters can be performed through human-readable XML files. XStream, a fast XML serializer and deserializer, was used to read these XML files and import parameters into the simulation.

3.2. Specialized libraries

A number of specialized libraries were necessary for particular aspects of the model. First, the model incorporates probability distributions in many places. The CERN Colt library (more precisely, functionality provided in the cern.jet package) was therefore a valuable resource that allowed for a very generic implementation without "hardcoding" any distributions into the code. The resulting simulation tool allows modelers to select from various types of distributions for specific simulation scenarios at runtime through configuration of XML parameter files.

Next, the Java Universal Network/Graph Framework (JUNG) was used for visualizing, reading, writing, and analyzing the social networks. This library also provides implementations of some of the generative network algorithms.

Finally, we used GeoTools GIS toolkit to implement the geospatial model and read population density data in ESRI shapefile format.

3.3. Tools for analysis and automation

Gnu R was used extensively to analyze and plot data. Bash and Perl scripts came into play for automating the simulation process, the discretization of data, and the analysis as well as plotting of results.

4. IMPLEMENTATION

Major design objectives for the implementation of the simulation included (i) reproducible results, (ii) provision of a flexible parameterization mechanism, (iii) no "hardcoding" of parameter values in the program code, and (iv) scalability and support for parallelization.

The first objective was achieved by initializing the random number generators in the simulation with random seeds from a configuration file. Integration tests were performed regularly during the implementation process to ensure that simulation runs with the same parameter sets and seeds always yield identical results.

The second and third objectives were achieved by means of a convenient parameterization mechanism based on a number of separate XML files, each of which configures particular aspects of the model. Major advantages of this method are that the parameter files are human-readable, can be easily edited, and that they can be validated against XML Schemas (XSD). The partitioning into separate files allows for their reuse in multiple scenarios and avoids redundancy. In order to simulate the diffusion of an innovation at varying price levels, for example, the same set of parameter files can be used for all price levels, with the sole exception of the pricing policy file. A single line that points to the pricing policy to use in the simulation has to be edited in a configuration file that binds the parameter set together (run.xml). The left-hand side of Figure 1 illustrates the configuration files and their relations.

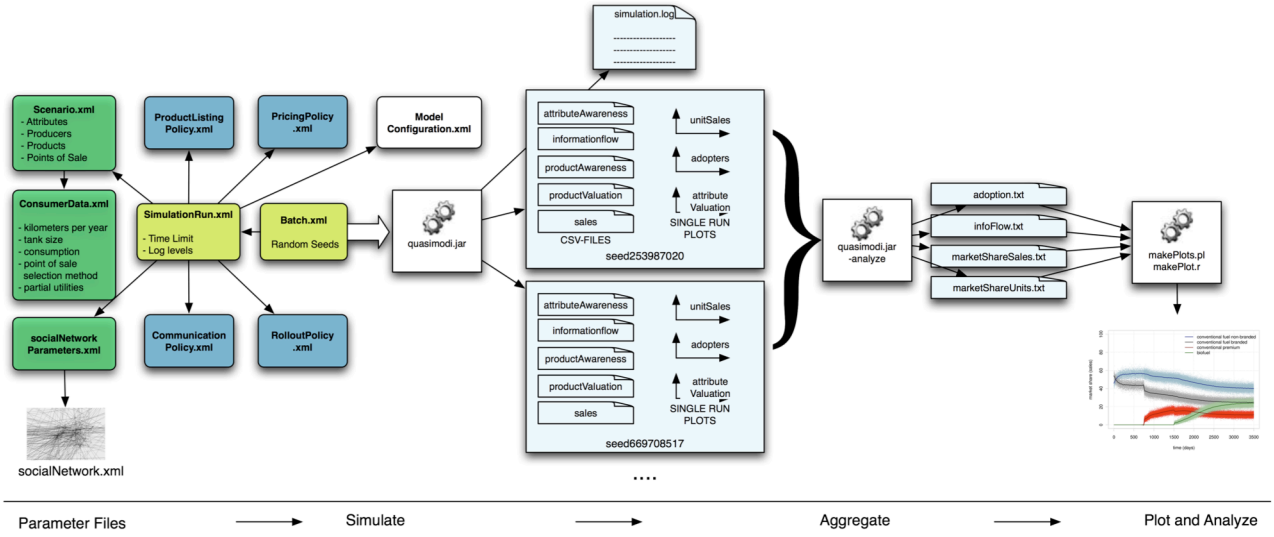

Figure 1: Architecture and simulation workflow

The fourth objective was achieved by dividing the steps in the simulation process into distinct program modules. Rather than optimizing for parallelization within individual replications (i.e., use of multiple processing cores to process events in a simulation run), we designed the simulation tool in a way that a set of runs with varying random seeds can be performed in parallel on multiple cores or computing nodes and results can then easily be collected, aggregated, and analyzed in a separate step. In particular, the following four distinct steps are performed for each simulation scenario, as illustrated in Figure 1: (i) Modeling of the scenario to simulate in a number of configuration files, (ii) simulation of the scenario for the number of replications specified, (iii) discretization and aggregation of results of individual simulation runs, and (iv) plotting and analysis of results.

5. GENERAL LESSONS LEARNED

General lessons learned in the course of the QuaSiMoDI project that go beyond the selection of suitable frameworks and modeling environments can be roughly divided in four groups. Firstly, there is a need for a sound empirical foundation; "just" setting up an agent-based model and to implement the corresponding simulation tool is no longer sufficient in order to make some valuable contribution to the field. Instead it is essential (and strongly demanded by many reviewers) to mirror micro-level factors and processes from real markets. In the QuaSiMoDI application case we therefore organized a focus group for criteria identification, performed a conjoint analysis for consumers' preference elicitation, and did additional empirical social research with a standardized questionnaire in order to secure additional information on the structure of the underlying social network and the communication behavior of (potential) customers. All in all, these activities have cost several months of work (and also financial resources for the market research institution that provided us access to their representative panel), but, in retrospective, it was worth the effort.

Secondly, it turned out that both the structure of the social (communication) network and corresponding parameters (e.g., concerning communication frequency) have considerable impact on simulation results. This raises a number of promising topics for further research, e.g., investigating stylized social network characteristics that are prevalent in different types of (consumer) markets. Unless sufficient evidence is available in literature, we strongly recommend placing particular emphasis on empirical data acquisition in this respect.

Validation of simulation results forms a third challenge that has to be mastered. For QuaSiMoDI we performed (i) a conceptual validation for which, as an example, we grounded our innovation decision-process on Roger's (1962) well-established framework, (ii) an internal validation with extensive unit and integration testing, (iii) a micro-level external validation for calibration (e.g., with a check for implausible values for tank size or inconsistent preference values from the conjoint analysis) as well as for verification (e.g., whether the micro-level output concerning choice of gas station is consistent with reported behavior or whether agents' communication behavior reflects assumptions), (iv) a macro-level external validation for which we performed a face validation with experts and also compared simulation results with data for market diffusion of premium (fossil) fuels from Germany, and (v) a cross-model validation for which we replicated stylized facts formalized in the model by Bass (1969).

Finally, it was essential to gather a team of experts with complementary competences. For the QuaSiMoDI project they came from the fields of innovation management, marketing, organization studies, sociology, operations research, and IT. On a personal note, working in such an interdisciplinary team made "tons of fun" (as a former colleague from the University of Texas would have phrased it) and certainly has been among the most appealing aspects in pursuing this endeavor.

ACKNOWLEDGMENTS

In the course of the QuaSiMoDI project we obviously did not just review platforms and implement the simulation tool, but also perform many more tasks (such as properly modeling the purchase decision or information transfer, setting up the social network, or testing the tool with data from a real-world application). This has been joint work together with several others, most prominently with Markus Günther, Rudolf Vetschera, and Lea Wakolbinger who have considerably contributed to the QuaSiMoDI research endeavor for which we are just grateful. Finally, we thank the Austrian Science Fund (FWF) for financial support by grant No. P20136-G14.

REFERENCES

Bass, F., 1969. A new product growth model for consumer durables. *Management Science*, 15 (5), 215-227.

Castle, C.J., Crooks, A.T., 2006. *Principles and concepts of agent-based modelling for developing geospatial simulations*. Working paper 110. Centre for Advanced Spatial Analysis, University College London.

Garifullin, M., Borshchev, A., Popkov, T., 2007. Using AnyLogic and agent-based approach to model consumer market. *Proceedings of the 6th EUROSIM Congress on Modelling and Simulation*, pp. 1-5. Sept. 9-13, Ljubljana (Slovenia).

Gilbert, N., 2002. Platforms and methods for agent-based modeling. *Proceedings of the National Academy of Sciences*, 99, 7197-7198.

Günther, M., Stummer, C., Wakolbinger, L.M., Wildpaner, M., 2011. An agent-based simulation approach for the new product diffusion of a novel biomass fuel. *Journal of the Operational Research Society*, 62 (1), 12-20.

Inchiosa, M.E., 2002. Overcoming design and development challenges in agent-based modeling using ASCAPE. *Proceedings of the National Academy of Sciences*, 99 (90003), 7304-7308.

Isaac, A.G., 2011. The ABM template models: a reformulation with reference implementations. *Journal of Artificial Societies and Social Simulation*, 14 (2), 5.

Kiesling, E., Günther, M., Stummer, C., Wakolbinger, L.M. (2012) Agent-based simulation of innovation diffusion: A review. *Central European Journal of Operations Research*, 20 (2), 183-230.

Kiesling, E., Günther, M., Stummer, C., Wakolbinger, L.M., 2009. An agent-based simulation model for the market diffusion of a second generation biofuel. In: M.D. Rossetti M.D., R.R. Hill, B. Johansson, A. Dunkin, R.G. Ingalls R.G., eds. *Proceedings of the Winter Simulation Conference (WSC 2009)*, pp. 1474-1481. Dec. 13-16, Austin (Texas, USA).

Kiesling, E., Günther, M., Stummer, C., Vetschera, R., Wakolbinger, L.M., 2010. A spatial simulation model for the diffusion of a novel biofuel on the Austrian market. In: A. Bargiela, S.A. Ali, D. Crowley, E.J.H. Kerckhoffs, eds. *Proceedings of the 24th European Conference on Modelling and Simulation (ECMS 2010)*, pp. 41-49. June 1-4, Kuala Lumpur (Malaysia).

Legéndi, R., Gulyás, L., Bocsi, R., Máhr, T., 2009. Modeling autonomous adaptive agents with functional language for simulations. In: L. Seabra Lopes, N. Lau, P. Mariano, L.M. Rocha, eds. *Progress in Artificial Intelligence, Lecture Notes in Computer Science 5816*. Berlin: Springer, 449-460.

Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K., 2004. MASON: a new multi-agent simulation toolkit. *Proceedings of the 2004 SwarmFest Workshop*, pp. 1-8. May 9-11, Ann Arbor (Michigan, USA).

Macal, C., North, M., 2007. Agent-based modeling and simulation: desktop ABMS. In: S.G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J.D. Tew, R.R. Barton, eds. *Proceedings of the Winter Simulation Conference (WSC 2007)*, pp. 95-106. Dec. 9-12, Washington D.C. (USA).

MathWorks, 2012. Matlab. Available from: http:// www.mathworks.com [accessed 5 July 2012].

Matsumoto, M., Nishimura, T., 1998. Mersenne twister: a 623-dimensionally equidistributed uniform pseudorandom number generator. *ACM Transactions on Modeling and Computer Simulation*, 8 (1), 3-30.

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.

Nikolai, C., Madey, G., 2009. Tools of the trade: a survey of various agent based modeling platforms. *Journal of Artificial Societies and Social Simulation*, 12 (2), 2.

North, M.J., Collier, N.T., Vos, J.R., 2006. Experiences creating three implementations of the Repast agent modeling toolkit. *ACM Transactions on Modeling and Computer Simulation*, 16 (1), 11-25.

North, M.J., Howe, T.R., Collier, N.T., Vos, J.R., 2005. The 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*, pp. 151-158. Oct. 13-15, Argonne (Illinois, USA).

Parker, M.T., 2001. What is Ascape and why should you care? *Journal of Artificial Societies and Social Simulation*, 4 (1), 5.

Railsback, S.F., Lytinen, S.L., Jackson, S.K., 2006. Agent-based simulation platforms: review and development recommendations. *Simulation*, 82 (9), 609-623.

Resnick, M., 1996. StarLogo: an environment for decentralized modeling and decentralized thinking. In: *Conference companion on Human factors in computing systems: common ground*. New York, NY: ACM Press, 11-12.

Rogers, E.M., 1962. *Diffusion of innovations*. New York, NY: The Free Press.

Tisue, S., Wilensky, U., 2004. NetLogo: a simple environment for modeling complexity. *Proceedings of the International Conference on Complex Systems (ICCS 2004)*, pp. 1-10. May 16-21, Boston (Massachusetts, USA).

Tobias, R., Hofmann, C., 2004. Evaluation of free Java-libraries for social-scientific agent based simulation. *Journal of Artificial Societies and Social Simulation*, 7 (1), 6.

Wolfram Inc., 2012. Mathematica. Available from: http://www.wolfram.com [accessed 5 July 2012].