Nina Schwarz, Daniel Kahlenberg, Dagmar Haase and Ralf Seppelt (2012)

ABMland - a Tool for Agent-Based Model Development on Urban Land Use Change

Journal of Artificial Societies and Social Simulation **15** (2) 8
<http://jasss.soc.surrey.ac.uk/15/2/8.html>

Received: 23-May-2011 Accepted: 24-Aug-2011 Published: 31-Mar-2012

Abstract

Modelling urban land use change can foster understanding of underlying processes and is increasingly realized using agent-based models (ABM) as they allow for explicitly coding land management decisions. However, urban land use change is the result of interactions of a variety of individuals as well as organisations. Thus, simulation models on urban land use need to include a diversity of agent types which in turn leads to complex interactions and coding processes. This paper presents the new ABMland tool which can help in this process: It is software for developing agent-based models for urban land use change within a spatially explicit and joint environment. ABMland allows for implementing agent-based models and parallel model development while simplifying the coding process. Six major agent types are already included as coupled models: residents, planners, infrastructure providers, businesses, developers and lobbyists. Their interactions are pre-defined and ensure valid communication during the simulation. The software is implemented in Java building upon Repast Simphony and other libraries.

Keywords:
Agent-Based Modelling, Urban, Land Use, Repast

Introduction

1.1 Although more than half of the population worldwide now lives in urban areas and consumes land and other resources, urbanisation is scarcely understood. Modelling urban land use change can foster understanding and is increasingly done with agent-based models (ABM) as they allow for coding land management decisions (Matthews et al. 2007; Parker et al. 2003; Schwarz et al. 2010; Wu & Silva 2010). In ABMs, individual agents act as autonomous entities towards realising own goals and interact with their environment and other agents. Some applications encompass a large number of agent objects that do differ in their attributes, but not in the behaviour rules, such as households (Haase et al. 2010). More complex models comprise of different agents like buyers and sellers of land with different decision rules (Filatova et al. 2009). Agents can either control land use change directly or indirectly (Parker et al. 2008). Such complex simulations are necessary for simulating urban land use change as urban sprawl as well as urban shrinkage are influenced by decisions of different groups of individuals and institutions, e.g. households, urban planners or developers (Schwarz et al. 2010). What is more, decision processes are structured differently in different regions.

1.2 Such complex simulations are very likely not developed by a single investigator, but a group of researchers, probably from different institutions (Cioffi-Revilla 2010). The joint design and implementation of programming code by different modellers or groups can be fostered by toolkits that facilitate the coding process and the coupling of the single models.

1.3 A variety of toolkits for implementing ABMs already exists (Berryman 2008; Railsback et al. 2006; Nikolai & Madey 2009; Robertson 2005; Tobias & Hofmann 2004), for instance Repast (Argonne National Laboratory 2007), Netlogo (Wilensky 1999), Mason (Luke et al. 2005) or Swarm (Minar et al. 1996). None of them explicitly provides a framework for coupling different ABMs. Simulation models in general can on the one hand be coupled using existing generic solutions, e.g. Geonamica (Hurkens et al. 2008), openMI (http://www.openmi.org) or Object Modeling System OMS (http://javaforge.com/project/oms). On the other hand, project-specific solutions for coupling ABMs are tailor-made for the simulation in question (e.g. Barthel et al. 2007). We describe such a new tool that provides the infrastructure to couple different ABM for urban land use change. In the current version, six ABMs are included: residents, businesses, planners, developers, infrastructure providers, and lobbyists.

Software features

2.1 **Scheduling.** The single ABMs follow a common scheduling that distinguishes initialisation, data exchange, and computation. Data exchange and computation run for each time step, while initialisation takes place only at the beginning of a simulation. A controller to store and update commonly used data, such as land use and prices, is implemented. Its computing cycle is executed before the single ABMs.

2.2 **Interfaces**. Specified data types (e.g. "household number") make sure that only a pre-defined data type ("integer") within upper and lower boundaries (0 to 10,000) is communicated.

2.3 **Space**. One or more grids can be used to allocate agents or land use. The software provides helper classes for importing and exporting basic spatial data.

2.4 **Default behaviour**. For each ABM, the "standard" behaviour needs to be coded in the respective default model implementation. This mechanism ensures that the coupled simulation can be executed even if concrete implementations are not available for all models. The standard behaviour might simply encompass static values that should be exported in every time step or include more dynamic responses. As soon as the concrete model implementation extends the default model implementation, the standard behaviour is not used, but the fully implemented version itself.

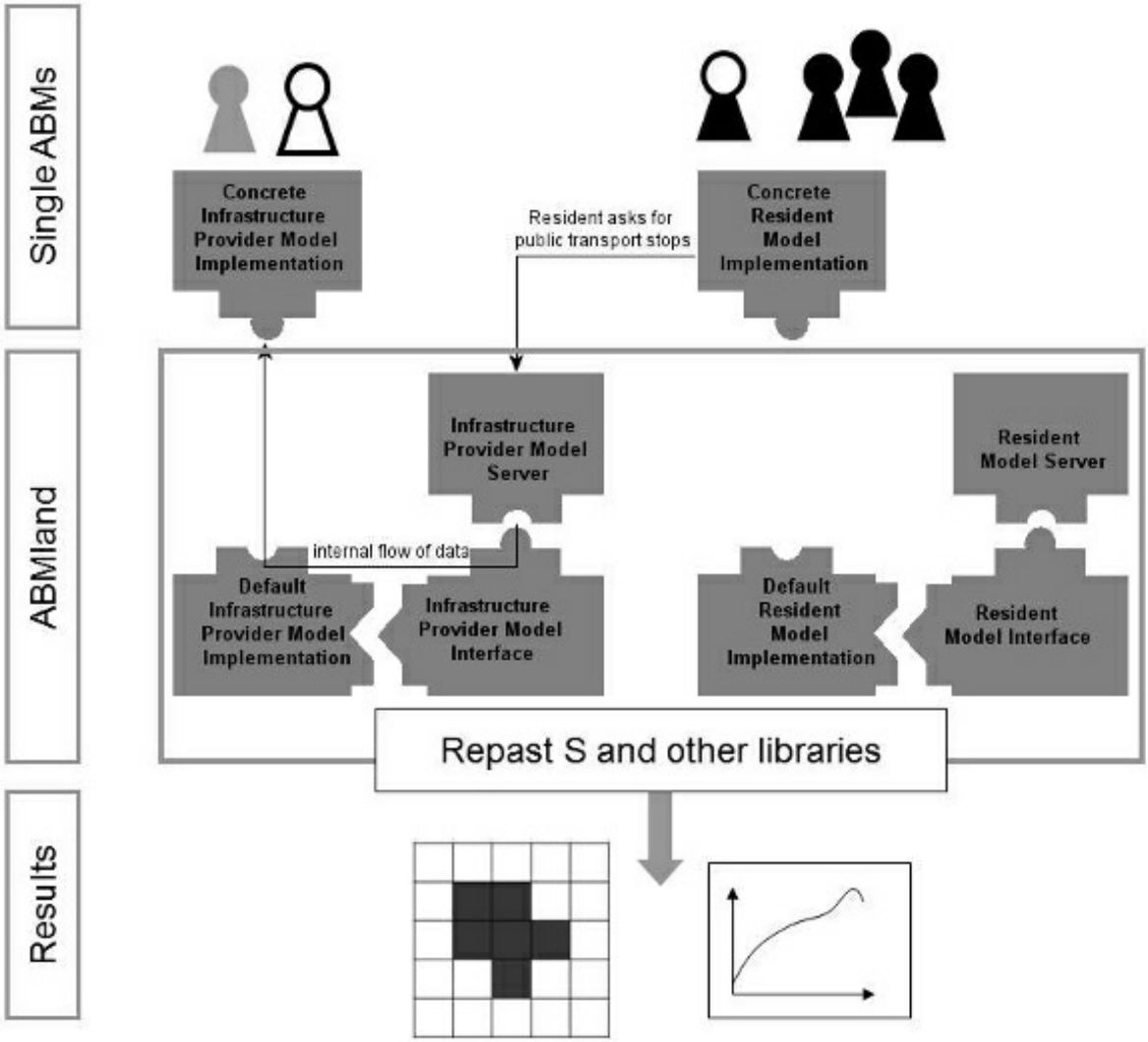

Figure 1. An overview of ABMland. Implementations of single ABMs (here: infrastructure providers and resident) communicate indirectly via server, interface and default model implementation as provided by the software. Results are exported as maps or aggregated data

Using ABMland

3.1 The ABMland tool consists of JPF (Java Plugin Framework) plugins that build upon Repast Simphony and other libraries in form of regular jar files. Additionally to the ABMland framework, a toy model is provided to exemplary show the usage of the framework. This toy example of infrastructure providers and residents indicates where meaningful parameters and methods should be included.

3.2 **Coding models and agents**. Various interactions have been implemented between the models (and therefore indirectly

between their agents). A number of specific data types are already pre-defined. Thus, coding the concrete models is straightforward: The methods to export data needed by other agents are already given, and each model only needs three more methods: `init()`, `getData()` and `compute()` with annotations to ensure the correct scheduling. Modellers are free to design their agents and additional classes as they like, as long as they do not interfere with the scheduling and use the model's scheduled classes to call all other methods.

3.3 **Changing default behaviours**. In the default implementation of each model, pre-defined values for each of the export methods are already implemented to ensure that some interaction takes place even if the full model has not been finalised yet by the modeller. For instance, planning agents in the default implementation randomly permit or deny proposals to change land use. To change this, the export methods in the respective default implementation have to be adapted.

3.4 **Additional features**. The ABMland tool also provides helper classes for input and output. The tool provides methods to import ASCII raster data as well as configuration files in XML format and some spreadsheet formats (csv, xls). Simulation results can currently be exported as log files or csv files. Finally, the ABMland tool also encompasses additional utilities to use Repast Simphony functions.

3.5 **Adapting the tool**. The ABMland tool can be adapted to (a) include other models and/or (b) data types. For (a), users have to

- modify the model.score file of their Repast Simphony simulation project,
- for each new model implement a default implementation, a server and an interface,
- add two context classes (for model and agent).

To include new data types, they need to be added with annotated boundaries into the respective package.

Concluding remarks

4.1 We described a tool to implement a coupled simulation consisting of six ABMs to model urban land use change. Using the ABMland software, one can easily experiment with different versions of the ABM because data exchanges are fixed. For instance, an infrastructure provider model representing central European planning regimes can be easily exchanged with an infrastructure provider model developed for Northern America. As a trade-off these models may be inconsistent in terms of their representation of reality. However, the ABMland software provides the opportunity to develop coupled ABM on urban land use change as single components can be implemented with a lot of freedom. Furthermore, the software can be adapted to include other models and/or data types. It not only provides an implementation to couple specific ABMs on urban land use change, but also gives the opportunity to adapt the general coupling principles for other applications.

Acknowledgements

This work is part of the PLUREL Integrated Project (Peri-urban Land Use Relationships) funded by the European Commission, Directorate-General for Research, under the 6th Framework Programme (project reference: 36921). The authors would like to thank Dave Murray-Rust for helpful discussions and bug reports as well as the colleagues of the PLUREL project for their constructive inputs. Furthermore, thanks go to Volker Grimm for reviewing an earlier version of the manuscript.

Appendix

Software availability

Name of Software	ABMland
Programming language	Java
Packages used	fj 2.12; google-collections 1.0; j3d-core-utils 1.3.1; jai_core 1.1.3; javaRasters 0.0.1.2; junit 4.6; log4j 1.2.13; Repast Simphony 1.2.0; simple-xml 2.3.1
Developers	Daniel Kahlenberg
Contact address	Helmholtz Centre for Environmental Research - UFZ Department of Computational Landscape Ecology Permoserstrasse 15 04318 Leipzig Germany Email: abmland@ufz.de
Availability	Via http://www.ufz.de/abmland, Apache 2.0 license
Detailed Information	Technical documentation, see URL above

Static description of ABMland

A.1 For each agent type, exactly one model class is implemented that encompasses all its behaviour. This model class has three main tasks: to include this part of the simulation into the overall scheduling of the coupled simulation, to control its own agent(s), and to exchange data with all other models and the environment. Splitting up the overall control / data exchange and the agent has the following advantage: One entity is coupled to the rest of the system in terms of scheduling and data exchange (= the model), while one or more entities (= the agents) are controlled by the model. Thus, only the model has to fulfil a few requirements, while the agent is not visible for the other parts of the coupled simulation and can therefore be coded without restrictions (Figure A-1).

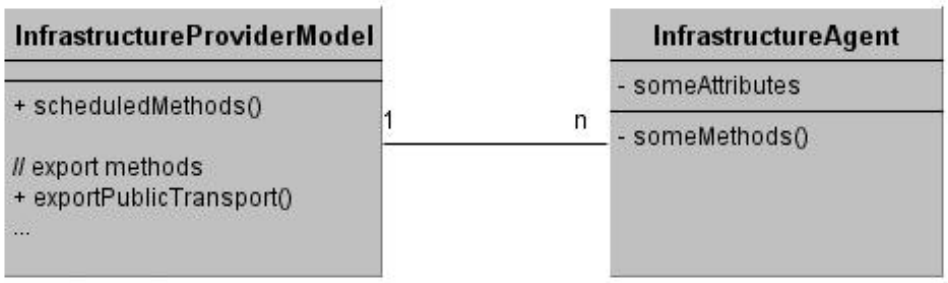

Figure A-1. The relationship of model and agent illustrated with the infrastructure provider model. In the infrastructure provider model, scheduled methods as well as methods exporting data are public and can thus be used from outside the class (indicated with '+'). Neither the infrastructure provider agent(s) nor its attributes and methods are visible (indicated with '-')

For one agent type (and thus for one model), the framework consists of a default implementation, a server and an interface. The interaction of these three constructs makes sure that at least the default implementation can deliver pre-defined values in a coupled simulation if one or more models are still under development (Figure 1 in the main text). Each model extends a default implementation providing dummy values in case no concrete implementation for a specific model is available for a coupled simulation. This default implementation extends an interface that lists all the methods the model uses to export data. Servers forward the data to a coupled model. The server gets the information via the interface and either receives the data from the concrete model implementation or from the default implementation, if there is no concrete implementation available.

A.2 Thus, communication between agents is indirect, i.e. their respective models exchange the information via servers. This mechanism enforces a model to implement the exact export methods. This is crucial for ensuring communication during the coupled simulation. Specified data types (e.g. "household number") make sure that only a pre-defined data type ("integer") within upper and lower boundaries (0 to 10,000) is communicated. Agents only receive the data that are provided by the interfaces, i.e. they do not know the source of the data. Thus, the information transported to the agents depends on the definition of the interfaces. If it is not sufficient for the ResidentAgent to know the mere location of a public transport stop because it also needs to know the specific bus line, then this information needs to be added in the interface. This is done by including a parameter to the definition of the data type which then states for instance the number of the bus line, in addition to the location of the bus stop. Controllers store and update commonly used data (land use and land prices). Spatially explicit data are wrapped into the Repast Simphony grid value layers.

Dynamic description of ABMland

A.3 For the scheduling of the activities in a coupled simulation, three main tasks are distinguished:

1. `init()`. All steps to start a simulation take place in the initialisation phase. This might refer to creating objects such as agents, reading in external data or configuration files and the like.
2. `getData()`. All models import data from other models in parallel. Only if every data import is finished, can the next phase start.
3. `compute()`. During the computation, models update their variables, e.g. by computing new states or by triggering the agents' decision processes. `getData()` and `compute()` are separated in order to ensure unique data imports. If these steps were not separated, then model A (resident) might import data (public transport stops) from model B (infrastructure provider) which is currently being altered in model B's `compute()` phase (new bus stops are built). If controllers are implemented as well, their `getData()` and `compute()` phase takes place before the models do theirs, so that the controllers can provide updated information in their `compute()` phase for the models (Figure A-2).

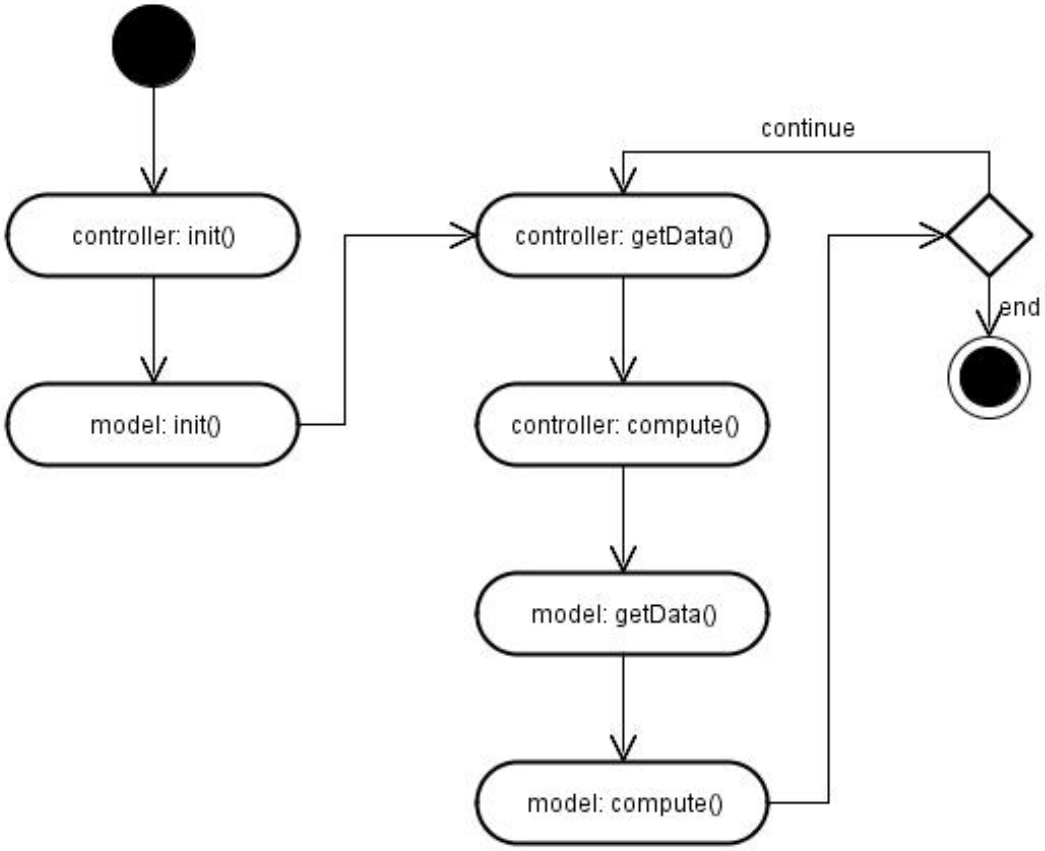

Figure A-2. Activity diagram for controllers and models

References

ARGONNE NATIONAL LABORATORY (2007) Repast Simphony. http://repast.sourceforge.net/index.html. Archived at: http://www.webcitation.org/5yd9xfEkJ

BARTHEL, R, Janisch, S, Schwarz, N, Trifkovic, A, Nickel, D, Schulz, C, Mauser W. (2008) An integrated modelling framework for simulating regional-scale actor responses to global change in the water domain. *Environmental Modelling & Software* 23, 1095-1121. [doi:10.1016/j.envsoft.2008.02.004]

BERRYMAN M (2008) *Review of Software Platforms for Agent Based Models*. Technical report. http://hdl.handle.net/1947/9306.

CIOFFI-REVILLA C (2010) A Methodology for Complex Social Simulations. *Journal of Artificial Societies and Social Simulation* 13(1), 7, http://jasss.soc.surrey.ac.uk/13/1/7.html.

FILATOVA T, Parker D, van der Veen A (2009) Agent-Based Urban Land Markets: Agent's Pricing Behavior, Land Prices and Urban Land Use Change. *Journal of Artificial Societies and Social Simulation* 12(1), 3 http://jasss.soc.surrey.ac.uk/12/1/3.html.

HAASE D, Lautenbach S, Seppelt R (2010) Applying social science concepts: modelling and simulating residential mobility in a shrinking city. *Environmental Modelling and Software* 25, 1225-1240. [doi:10.1016/j.envsoft.2010.04.009]

HURKENS J, Hahn B, van Delden H (2008) Using the GEONAMICA software environment for integrated dynamic spatial modelling. In: Sanchez-Marre, M., Comas, J., Rizzoli, A., Guariso, G. (Eds), *Integrating Sciences and Information Technology for Environmental Assessment and Decision Making*, International Congress on Environmental Modelling and Software 2008.

LUKE S, Cioffi-Revilla C, Panait L, Sullivan K, Balan G (2005) MASON: A Multi-Agent Simulation Environment. *Simulation: Transactions of the society for Modeling and Simulation International*. 82(7), 517-527.

MATTHEWS RB, Gilbert N, Roach A, Polhill JG, Gotts NM (2007) Agent-based land-use models: a review of applications. *Landscape Ecology*, 22, 1447-1459. [doi:10.1007/s10980-007-9135-1]

MINAR N, Burkhart R, Langton C and Askenazi M (1996) *The Swarm simulation system: a toolkit for building multi-agent simulations*. Technical report. http://www.santafe.edu/media/workingpapers/96-06-042.pdf.

NIKOLAI C and 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 http://jasss.soc.surrey.ac.uk/12/2/2.html.

PARKER DC, Manson SM, Janssen MA, Hoffmann MJ, Deadman P (2003) Multi-agent systems for the simulation of land-use and land-cover change: A review. *Annals of the Association of American Geographers* , 93(2), 314-337. [doi:10.1111/1467-8306.9302004]

PARKER DC, Brown D, Polhill JG, Manson SM, and P Deadman (2008) Illustrating a new 'conceptual design pattern' for agent-based models and land use via five case studies: the MR POTATOHEAD framework. In A. L. Paredes and C. H. Iglesias (eds.): *Agent-based Modelling in Natural Resource Management*, Universidad de Valladolid, Valladolid, Spain. 29-62.

RAILSBACK SF, Lytinen SL, Jackson SK (2006) Agent-based Simulation Platforms: Review and Development Recommendations. *Simulation*, 82(9), 609-623. [doi:10.1177/0037549706073695]

ROBERTSON DA (2005) Agent-Based Modeling Tookits NetLogo, RePast, and Swarm. *Academy of Management Learning and Education*, 4(4), 525-527. [doi:10.5465/AMLE.2005.19086798]

SCHWARZ N, Haase D, Seppelt R (2010) Omnipresent sprawl? A review of urban simulation models with respect to urban shrinkage. *Environment & Planning B* 37(2), 265-283. [doi:10.1068/b35087]

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 http://jasss.soc.surrey.ac.uk/7/1/6.html.

WILENSKY U (1999) *NetLogo*. http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL. Archived at http://www.webcitation.org/5yeJ2ozEQ.

WU N, Silva EA (2010) Artificial Intelligence Solutions for Urban Land Dynamics: A Review. *Journal of Planning Literature*, 24(3), 246-265. [doi:10.1177/0885412210361571]