20th International Congress on Modelling and Simulation, Adelaide, Australia, 1–6 December 2013
www.mssanz.org.au/modsim2013

Integrated Land Systems Modelling and Optimisation

A. Herzig [a] and D. Rutledge [b]

[a] *Landcare Research New Zealand Ltd, Private Bag 11052, Palmerston North 4442, New Zealand*
[b] *Landcare Research New Zealand Ltd, Private Bag 3127, Hamilton 3240, New Zealand*
Email: herziga@landcareresearch.co.nz

Abstract: The challenge of sustainable land management is to understand the involved ecosystems, their spatial variability, and the impact of human activities, and to 'configure' the landscape in the best possible way for sustaining natural resources and maximising ecosystem services. We introduce the Land Use Management Support System, which addresses system understanding by its environmental integrated modelling framework (EIMF) and land-use configuration by its multi-objective spatial optimisation component. LUMASS is built on free and open source software libraries and mainly targeted at processing raster data sets. The LUMASS EIMF is based on the Orfeo Tool Box (OTB) and the Insight Segmentation and Registration Toolkit (ITK). OTB/ITK provide advanced functionality for processing big n-dimensional raster data sets, such as sequential and multi-threaded processing. The LUMASS optimisation component is based on the mixed integer linear programming solver lp_solve and provides the optimisation of areal resource allocations. LUMASS supports read and write access of data sets via the GDAL/OGR library as well as storage and retrieval of big multi-dimensional raster data by rasdaman. Furthermore, rasdaman's Petascope component provides access to raster data via OGC-compliant web services, such as WCS, WCPS, WPS, and WMS. One of the main objectives of LUMASS is to allow non-programmers to develop spatially explicit system dynamics models. For that purpose, we developed a user interface for the interactive visual (i.e. icon-based) development of LUMASS models. The visualisation of spatial (raster and vector) data, attribute tables, and charts, is implemented based on Qt widgets provided as part of the Visualisation Toolkit (VTK). To support LUMASS model and optimisation applications on cluster environments, we also provide a LUMASS command line version ('lumassengine'). This also allows LUMASS to be run as a backend engine in distributed service-oriented architectures as well as being integrated into other component-based EIMF (e.g. OMS3, OpenMI), subject to the development of appropriate wrapper classes. The LUMASS EIMF is based on the ITK processing pipeline architecture and its fundamental components: data and algorithms (i.e. processes). They are augmented by iterable components that provide the capability to build dynamic and hierarchical processing pipelines. An iterable component may either host a single process component, or a chain (pipeline) of other model components to build a hierarchical structure of processing pipelines. Furthermore, iterable components allow the hosted components to be executed a number of times, thereby changing process parameter values with each iteration. Additionally, each model component is assigned to a user-specified time level. Execution order and data flow starts at the highest time level and descends to lower time levels. This allows users to implement models operating on different time scales and/or to control execution order for component initialisation purposes. Selected model components can be saved as XML-based files and re-used in other modelling exercises. For each component, general and component-specific properties, including the assigned values, are stored to build a complete blueprint of a model and its parameterisation. Used in conjunction with a version control system, such as git, it facilitates model management and governance, especially for revised model implementations. To enable the future development of different model views, e.g. as part of a web-application, as well as to enable the 'lumassengine' command line application, the graphical model representation as displayed in the user interface, is stored in a binary file format, and is completely separated from the model structure and parameterisation.

The free and open source Land Use Management Support System integrates spatial land systems modelling and multi-objective spatial optimisation. Its integrated modelling framework allows non-programmers to develop spatially explicit, hierarchical, and dynamic land system models using a visual programming environment. LUMASS makes integrated modelling available to modellers without requiring any programming skills. It supports the development of models operating on big data sets and different temporal scales, and supports the exchange and collaboration among modellers by facilitating the management, governance, and re-use of model components.

Keywords: *Spatially explicit system dynamics, Land systems modelling, Spatial optimisation, Integrated modelling framework, LUMASS*

1. INTRODUCTION

The challenge of sustainable land management is to understand the involved ecosystems, their spatial variability, and the impact of human activities, and to 'configure' the landscape in the best possible way for sustaining natural resources and maximising ecosystem services. System understanding and impact assessment is commonly addressed by spatially explicit biophysical models comprising different science domains (Argent, 2004), whereas land-use patterns are optimised using spatial multi-objective optimisation tools (e.g. Ausseil et al., 2012). We introduce the Land Use Management Support System (LUMASS), comprising an environmental integrated modelling framework (EIMF, Rizzoli et al., 2008) as well as a multi-objective spatial optimisation tool (Herzig et al., 2013). The complexity of coupled human-natural systems requires purpose-built models and does not allow for generic abstraction. Therefore, new models will always have to be built. Furthermore, the same system or process can be modelled in different ways, reflecting different conceptualisations of a particular process (e.g. stochastic vs. deterministic), the availability of data, or just the personal preference of the modeller. In this context, it is desirable to use a modelling framework, which supports the flexible development, re-design, and testing of system, sub-system, and process implementations. Many environmental modelling frameworks have been developed for this purpose. To get an overview of available frameworks, basic concepts and applications as documented in the literature, we refer readers to the following articles: Argent (2004), Argent et al. (2006), Rizzoli et al. (2008), Jagers (2010), Granell et al. (2013), Laniak et al. (2013), and Kelly et al. (2013). The majority of EIMF is targeted at software developers. This is especially the case for frameworks that support the development of "coupled component models" (Kelly et al., 2013). For example, TIME (Rahman et al., 2003), ModCom (Hilyer et al., 2003), JAMS (Kralisch and Krause, 2006), OpenMI (OATC, 2010), and OMS3 (David et al., 2013) require knowledge of a general purpose programming language (e.g. Java, C/C++) to develop basic model components, such as the implementation of a particular process. Even more sophisticated programming skills are required to build components that can deal with multiple scales in space and time as well as big data sets. System dynamics modelling environments (e.g. STELLA, www.iseesystems.com, VENSIM, www.vensim.com) do not require programming skills to develop complex system models, but often only provide limited spatial modelling capabilities (Kelly et al., 2013). Therefore, they have been extended by third-party software packages (e.g. SME, Maxwell et al., 2004), to allow for spatially explicit system dynamics modelling (Kelly et al., 2013). However, this creates dependence on closed source software systems, which cannot be adapted or extended if required. In this paper, we introduce a free and open source EIMF as part of the Land Use Management Support System (LUMASS) (Herzig, 2008a). The main objective of the LUMASS EIMF is to allow non-programmers to develop spatially explicit hierarchical dynamic land system models for system understanding and integrated impact assessments. Modelling results may be used as input to the LUMASS spatial optimisation component (Herzig, 2008b) to maximise land-use performance and ecosystem services (Ausseil et al., 2012, Herzig et al., 2013). In the remainder of this paper, we describe the LUMASS architecture (Section 2), provide an overview of the LUMASS EIMF (Section 3), outline the sample implementation of a simple dynamic soil water budget model (Section 4), discuss the presented modelling framework (Section 5), and conclude (Section 6).

2. LUMASS ARCHITECTURE

An earlier version of LUMASS, based on proprietary GIS software, is discussed by Herzig (2008a,b). In this paper, we introduce a revised and improved system exclusively built on free and open source software (Figure 1). Each underlying library is supported by a large and active community of users and developers, which ensures future development, testing, and bug fixing. The main programming language used is C++ and all libraries have been, or are currently used on multiple platforms and potentially allow a future version of LUMASS to be compiled on different platforms.

2.1. Data Processing

LUMASS provides two main processing components, an EIMF and a multi-objective spatial optimisation component. The EIMF is primarily targeted at processing raster data sets to support the development of cellular automata-based models (e.g. White et al., 1997), and the processing of remote-sensing images and digital elevation models, and to facilitate overlay operations (e.g. map algebra), as well as sequential and parallel processing of large data sets. The implementation of the LUMASS EIMF is based on the Orfeo Tool Box (OTB) (OTB Development Team, 2013) and the Insight Segmentation and Registration Toolkit (ITK) (Ibanez et al., 2005). ITK provides advanced functionality for processing big n-dimensional raster data sets, such as sequential and multi-threaded processing, provided by its processing pipeline architecture. OTB extends ITK by specific capabilities and algorithms for processing high-resolution remote-sensing images. OTB and ITK significantly facilitate the implementation of core processing components for the LUMASS

EIMF (Section 3). The LUMASS optimisation component is based on the mixed integer linear programming solver lp_solve (Berkelaar et al., 2004) and provides the spatial optimisation of areal resource allocations (Herzig, 2008b, Herzig et al., 2013).

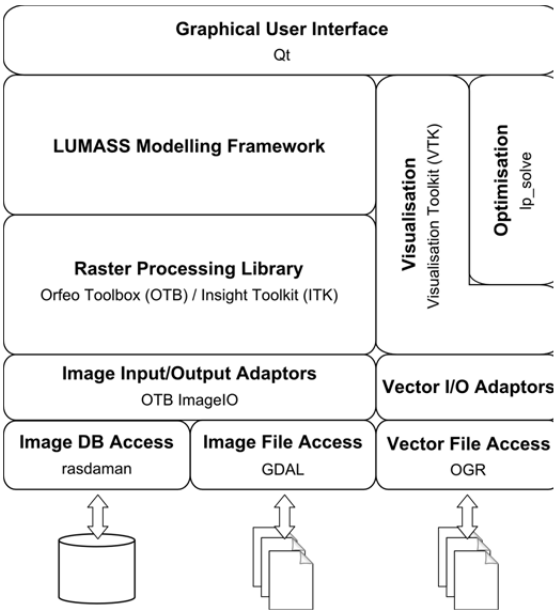

Figure 1. LUMASS architecture and underlying software libraries.

2.2. Data Storage and I/O

LUMASS supports read and write access to data base as well as file-based data sets. Raster and vector data stored in the most common two-dimensional data formats can be accessed via the GDAL/OGR library (GDAL, 2012). Storage and retrieval of multi-dimensional raster data sets is provided by rasdaman. Rasdaman is middleware and extends standard relational data bases with the capability to store and retrieve n-dimensional raster data sets (arrays) of unlimited size (Bauman et al., 1997). Rasdaman's Petascope component also provides web-based access to rasdaman image data via OGC®-compliant (OGC: Open Geospatial Consortium) web services, such as Web Coverage Service (WCS), Web Coverage Processing Service (WCPS), Web Processing Service (WPS), as well as Web Mapping Service (WMS). Specific input/output classes (adaptors), implemented as part of OTB/ITK and VTK, translate the data structures between the storage access layer and the actual data processing libraries.

2.3. User Interface and Visualisation

One of the main objectives of LUMASS is to allow non-programmers to develop spatially explicit system dynamics models. For that purpose, we developed a user interface for the interactive visual (i.e. icon-based) development of LUMASS models (Figure 4). For the implementation of the user interface, as well as the LUMASS EIMF classes (Figs 2 and 3), we used the Qt C++ programming framework (Qt Project, 2013). This provides sophisticated cross-platform user interface components (widgets) and C++ language extensions, such as the Meta-Object System. The latter allows object properties to be queried at run-time and enables dynamic graphical user interface components, e.g. for the configuration of model components. The visualisation of spatial (raster and vector) data, attribute tables, and charts, is implemented based on Qt widgets provided with VTK. Raster data sets, represented by OTB/ITK classes for input/output as well as processing, are mapped onto VTK classes for visualisation purposes. Both libraries provide specific import and export classes that allow for a seamless connection of the ITK processing pipeline with the VTK visualisation pipeline (Schroeder et al., 2006).

2.4. LUMASS Engine

To support scenario-based analyses or Monte Carlo simulations in the context of sensitivity or uncertainty assessments, we also provide a command line-based application ('lumassengine'). It runs geospatial models and optimisation scenarios based on LUMASS model and optimisation settings files, respectively. This not only allows LUMASS to be used on compute clusters, but also as a backend-processing engine in distributed service-oriented architectures. Furthermore, it facilitates the integration into other component-based EIMF, such as OMS3, or OpenMI, subject to the development of appropriate wrapper classes, which would allow LUMASS models to be linked with OMS3 or OpenMI components respectively.

3. LUMASS MODELLING FRAMEWORK

3.1. Concept and Design

The fundamental building blocks of the ITK pipeline architecture, data and algorithms, are represented by the *NMItkDataObjectWrapper* and *NMProcess* classes respectively. They are wrapper classes around the corresponding template-based OTB/ITK classes and provide access to the underlying base class objects as well as information about the template parameters. This meta-data is represented by Qt properties and can be set and queried at run-time. *NMProcess* is an abstract class and only provides general public methods (e.g. *update()*) and properties, such as the list of inputs and data types. Concrete process implementations are derived from *NMProcess* and extend the list of properties by process specific parameters. Additional to the 'pipeline classes', the LUMASS modelling framework comprises a set of *NMModelComponent*-based classes (Figure 2) to manage data and process objects to allow for the implementation of hierarchical, dynamic models. The abstract base class provides common model component properties such as a description, the time level on which the object resides, as well as the list of input components. Dynamic pipelines and structural hierarchy (i.e. aggregation of model components) are provided by *NMIterableComponent*-derived classes. If only small data sets are being processed, users can avoid reading and writing data sets at the beginning and end of each pipeline, by using a data component object *(NMDataComponent)*. It explicitly allows model data to be kept in the main memory of the computer during model execution. Iterable components either host a single process object (henceforth called process component) or a chain of other model components (Figure 3), implemented as a doubly linked list. Each LUMASS model comprises at least one iterable component, henceforth referred to as the root component, which forms the basement of the hierarchical model structure representing time level zero. All additional components added to the model are hosted by the root component and are initially also placed on time level zero. The user can adjust the time level of individual components via the user interface. However, all components comprised by a particular processing pipeline have to share the same time level and host component. LUMASS ensures that the lowest time level assigned to a component is zero and that the time level of each individual model component is at least as high as the time level of its host component. Individual model components sharing the same host component can be aggregated, whereby LUMASS creates a new iterable component and places the selected model components inside it. Of the two different sub-classes of iterable components (Figure 3), only the sequential iteration component has been implemented yet. The number of iterations is represented as a component property of the class and can be specified at run-time. This enables the representation of time periods subdivided into a discrete number of intervals. The conditional iteration component rather represents a logical control structure, which could be used (once implemented), for example, to implement allocation procedures of land-use/land-cover change models. However, at each iteration step, process-components may assign new process parameters to its underlying process objects. Most process properties actually represent a list of parameters containing individual parameters for each iteration step, allowing for time-dependant process behaviour. Since this also includes process inputs, processing pipelines (i.e. their linkage) may change over time, for example to control process initialisation and finalisation (e.g. temporal or spatial data aggregation).

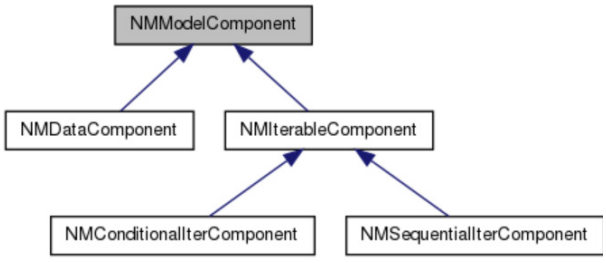

Figure 2. LUMASS model component classes (generated by doxygen 1.7.6.1; www.doxygen.org).

3.2. Model controller and model execution

Each LUMASS instance owns exactly one model controller instance, which manages an internal model register and controls the lifetime of all model component objects. The model controller also creates the root component of the LUMASS model and controls and monitors model execution and abortion. Each iterable model component can be executed individually to facilitate model testing without having to isolate the particular model component. However, model execution only applies to those components hosted by the particular iterable component. Hence, any required input data needs to be generated beforehand. Upon model execution, as long as no other model is currently running, the model controller calls *update()* on the user specified iter-

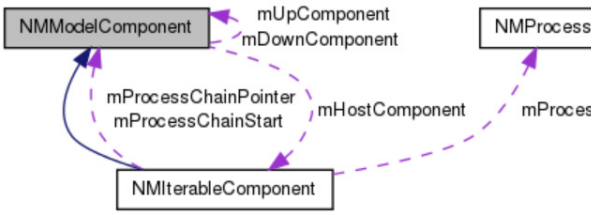

Figure 3. Model component collaboration diagram (generated by doxygen 1.7.6.1; www.doxygen.org).

able component. The component then identifies all processing pipelines on the individual time levels hosted by the component. The pipelines are then executed sequentially in the order of time levels, starting with the highest time level. Execution order of pipelines sharing the same time level is arbitrary; hence they must be independent of each other. If any of the required inputs to a particular component is not available, model execution aborts. Because of the sequential processing order of model components and the restriction of parallel processing to the data set level, dead locks are effectively avoided. Following the execution order, data flows from higher time levels to lower time levels. Therefore, higher time levels represent processes occurring in shorter intervals (e.g. daily processes), and lower time levels represent processes occurring in relatively longer intervals (e.g. monthly or annual processes). However, time levels may also be used to specifically control the execution order and data flow in the model, for example to control the initialisation of model components.

3.3. Model Serialisation

Selected model components can be saved as XML-based files and re-used in other modelling exercises. For each component general and component specific properties including the assigned values are stored to build a complete blueprint of a model and its parameterisation. Used in conjunction with a version control system, such as git, it facilitates model management and governance, especially for revised model implementations. To enable the future development of different model views, e.g. as part of a web-application, as well as to enable the 'lumassengine' command line application, the graphical model representation as displayed in the user interface, is stored in a binary file format and is completely separated from the model structure and parameterisation.

4. SAMPLE MODEL IMPLEMENTATION

Figure 4 shows a screenshot of the sample implementation of a simple, spatially explicit single layer soil water budget model as represented in the graphical user interface of the modelling framework. The model comprises process components, depicted by the gears icon, data components, depicted by the coloured grid, and an iterable component depicted as blue pane. The model operates on three time scales representing the data initialisation phase on time level 2, the soil water budget and runoff calculation over a number of time steps on time level 1, and finally a phase in which the resulting data sets are written to the database. This implementation only uses small raster data sets, which can be kept in the main memory of the computer, as indicated by the use of data components. The iterable component on time level 1 calculates the actual soil water budget and the runoff, based on the initial data sets read in the initialisation phase, as well as based on the time series of maps representing precipitation and evapotranspiration. However, from the second iteration onwards, the map algebra component changes its set of input components. It replaces the initial soil water content data set with the actual soil water content calculated as result of the previous iteration. After the soil water budget and runoff at the end of the given time period has been calculated, the results are written into the data base (time level 0).

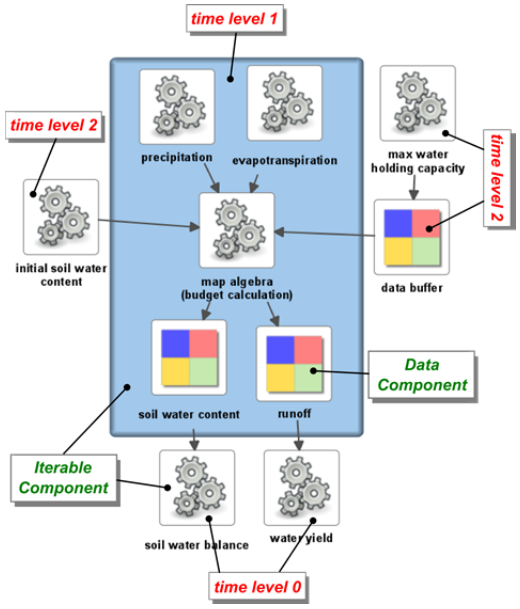

Figure 4. Screenshot of a simple dynamic soil water budget model as represented in the LUMASS model development interface (descriptions of component types and time levels are not part of the user interface representation).

5. DISCUSSION

The challenge of complex spatial systems modelling has been addressed by EIMF, such as TIME (Rahman et al., 2003), ModCom (Hilyer et al., 2003), JAMS (Kralisch and Krause, 2006), and OpenMI (OATC, 2010). However, the majority of frameworks are tailored to software developers (e.g. OMS3, David et al., 2013) rather than modellers. Voinov and Shugart (2013) point out that this type of systems is mostly concerned with the coupling and integration of software components. They hardly provide any out-of-the-box functionality to deal with complex spatial modelling issues, such as multiple spatial and temporal scales or big data sets. They are

very flexible and well suited to integrate legacy code, but they are only useful to modellers with substantial programming skills or to large (i.e. rich) projects and agencies that can afford to employ a team of software developers working alongside domain-specific modellers. Many modellers do not have such available resources, nor do they have sufficient programming skills to implement new algorithms and model components. The LUMASS EIMF is tailored to modellers to support the creative, visual development of spatially explicit complex system models. It provides most of the desirable features of modern component-based EIMF as given by Rizzoli et al. (2008). It currently does not support the integration of components developed in other EIMF; however, wrapper classes for components of other EIMF may be developed, e.g. for OMS3 and OpenMI, to make them available within the LUMASS EIMF. Likewise, wrappers around the 'lumassengine' can be developed to allow for the integration of LUMASS models in other EIMF. Unlike other frameworks (e.g. OMS3, ModCom, OpenMI), LUMASS explicitly supports the development of models operating on big data sets, provided by the support of the OTB/ITK pipeline architecture and a direct link to the rasdaman data base. Furthermore, the LUMASS EIMF implementation allows for the development of hierarchical dynamic models operating on different temporal scales, without requiring any additional programming by the modeller. Hence, LUMASS allows the modeller to focus on the domain specific aspects of model design, component implementation, and linkage, thus minimising the risk of developing 'integronsters' (Voinov and Shugart, 2013) through focusing too much on the software challenge, rather than on the modelling challenge (Voinov and Shugart, 2013).

6. CONCLUSIONS

The free and open source Land Use Management Support System integrates spatial land systems modelling and multi-objective spatial optimisation. Its integrated modelling framework allows non-programmers to develop spatially explicit hierarchical dynamic land system models using a visual programming environment. LUMASS makes integrated modelling available to modellers without requiring any programming skills. It supports the development of models operating on big data sets and different temporal scales, and also exchange and collaboration among modellers by facilitating the management, governance, and re-use of model components.

REFERENCES

Argent, R.M., (2004). An overview of model integration for environmental applications—components, frameworks and semantics. *Environmental Modelling & Software,* 19(3), 219–234.

Argent, R.M., Voinov, A., Maxwell, T., Cuddy, S.M., Rahman, J.M., Seaton, S., Vertessy, R.A., Braddock, R.D., (2006). Comparing modelling frameworks – A workshop approach. *Environmental Modelling & Software*, 21(7), 895–910.

Baumann, P., Furtado, P., Ritsch, R., (1997). Geo/environmental and medical data management in the RasDaMan System. In: Jarke, M., Carey, M.J., Dittrich, K.R., Lochovsky, F.H., Loucopoulos, P., Jeusfeld, M.A. (eds.), VLDB'97, Proceedings of 23rd International Conference on Very Large Data Bases, Athens, Greece.

Berkelaar, M., Eikland, K., Notebaert, P., (2004). lp_solve version 5.5 – open source (mixed-integer) linear programming system. http://groups.yahoo.com/group/lp_solve (accessed 06/08/2013).

David, O., Ascough, J.C., Lloyd, W., Gree, T.R., Rojas, K.W., Leavesley, G.H., Ahuja, L.R. (2013). A software engineering perspective on environmental modeling framework design: The Object Modeling System. *Environmental Modelling & Software*, 39, 201–213.

Hilyer, C., Bolte, J., van Evert, F., Lamaker, A. (2003). The ModCom modular simulation system. *European Journal of Agronomy*, 18(3), 333–343.

GDAL, (2012). GDAL - Geospatial Data Abstraction Library: Version 1.9.2, Open Source Geospatial Foundation, http://gdal.osgeo.org (accessed 06/08/2013).

Granell, C., Schade, S., Ostlaender, N. (2013). Seeing the forest through the trees: A review of integrated environmental modelling tools. *Computers, Environment and Urban Systems*, 41, 136–150.

Herzig, A. (2008a). Design and Implementation of the Land Use Management Support System (LUMASS). In: Sanchez-Marre, M., Bejar, J., Comas, J., Rizzoli, A., Guariso, G. (eds.), 2008 International Congress on Environmental Modelling and Software: Integrating Sciences and Information Technology for Environmental Assessment and Decision Making, Fourth Biennial Meeting, International Environmental Modelling and Software Society, Barcelona, Spain.

Herzig, A. (2008b). A GIS-based Module for the Multiobjective Optimization of Areal Resourc Allocation. In: Bernard, L., Friis-Christensen, A., Pundt, H., Compte, I. (eds.), Proceedings of the 11th AGILE International Conference on Geographic Information Science, Girona, Spain.

Herzig, A., Ausseil, A.-G.E., Dymond, J.R. (in press): Spatial Optimisation of Ecosystem Services. In Dymond, J.R. (ed.), Ecosystem Services in New Zealand – conditions and trends. Manaaki Whenua Press, Lincoln, New Zealand.

Ibanez, L., Schroeder, W., Ng, L., Cates, J., (2005). The ITK Software Guide, 2nd ed., Kitware, Inc., New York, USA. http://www.itk.org/ItkSoftwareGuide.pdf (accessed 06/08/2013).

Jagers, H.R.A. (2010). Linking, Data, Models and Tools: An Overview. In: Swayne, D.A., Yang, W., Voinov, A.A., Rizzoli, A., Filatova, T. (eds.), 2010 International Congress on Environmental Modelling and Software: Modelling for Environment's Sake, Fifth Biennial Meeting, International Environmental Modelling and Software Society, Ottawa, Canada.

Kelly (Letcher), R.A., Jakeman, A.J., Barreteau, O., Borsuk, M.E., ElSawah, S., Hamilton, S.H., Henriksen, H.J., Kuikka, S., Maier, H.R., Rizzoli, A.E., van Delden, H., Voinov, A.A., (2013). Selecting among five common modelling approaches for integrated environmental assessment and management. *Environmental Modelling & Software*, 47, 159–181.

Kralisch, S., Krause, P. (2006). JAMS – A Framework for Natural Resource Model Development and Application. In: Voinov, A., Jakeman, A.J., Rizzoli, A.E. (eds.), 2006 International Congress on Environmental Modelling and Software: Summit on Environmental Modelling and Software, Third Biennial Meeting, International Environmental Modelling and Software Society, Burlington, USA.

Laniak, G.F., Olchin, G., Goodall, J., Voinov, A., Hill, M., Glynn, P., Whelan, G., Geller, G., Quinn, N., Blind, M., Peckham, S., Reaney, S., Gaber, N., Kennedy, R., Hughes, A., (2013). Integrated environmental modeling: A vision and roadmap for the future. *Environmental Modelling & Software*, 39, 3–23.

Maxwell, T., Voinov, A., Costanza, R. (2004). Spatial Simulation Using the SME. In: Costanza, R., Voinov, A. (Eds.), Landscape Simulation Modeling, Springer, New York, USA.

OATC - The OpenMI Association Technical Committee (2010). The OpenMI "in a Nutshell" for the OpenMI (Version 2.0). https://sites.google.com/a/openmi.org/home/learning-more/OpenMIInaNutshell.pdf?attredirects=0 (accessed 28/07/2013).

OTB Development Team, (2013). The ORFEO Tool Box Software Guide – Updated for OTB-3.18. Centre National d'Etudes Spatiales (CNES), http://www.orfeo-toolbox.org/packages/OTBSoftwareGuide.pdf (accessed 06/08/2013).

Qt Project, (2013). Qt Project – cross-platform application and UI framework. http://qt-project.org/ (accessed 06/08/2013).

Rahman, J.M., Seaton, S.P., Perraud, J.M., Hotham, H., Verrelli, D.I., Coleman, J.R. (2003). It's TIME for a new environmental modelling framework. In: Post, D.A. (ed.) MODSIM 2003 International Congress on Modelling and Simulation. Modelling and Simulation Society of Australia and New Zealand. Townsville, Australia.

Rizzoli, A.E., Leavesley, G., Ascough II, J.C., Argent, R.M., Athanasiadis, I.N., Brilhante, V., Claeys, F.H.A., David, O., Donatelli, M., Gijsbers, P., Havlik, D., Kassahun, A., Krause, P., Quinn, N.W.T., Scholten, H., Sojda, R.S., Villa, F., (2008a). Integrated Modelling Frameworks for Environmental Assessment and Decision Support. In: Jakeman, A.J., Voinov, A.A., Rizzoli, A.E., Chen, S.H., (eds.) Environmental Modelling, Software and Decision Support: State of the Art and New Perspectives. Elsevier, Amsterdam, The Netherlands.

Schroeder, W., Martin, K., Lorensen, B., (2006). The visualization toolkit : an object-oriented approach to 3D graphics, 4th ed., Kitware, Inc., New York, USA.

Voinov, A., Shugart, H.H., (2013). "Integronsters", integral and integrated modeling. *Environmental Modelling & Software*, 39, 149–158.

White, R., Engelen, G., Uljee, I., (1997). The use of constrained cellular automata for high-resolution modelling of urban land-use dynamics. *Environment and Planning B: Planning and Design*, 24(3), 323–343.