International Environmental Modelling and Software Society (iEMSs)
2012 International Congress on Environmental Modelling and Software
Managing Resources of a Limited Planet, Sixth Biennial Meeting, Leipzig, Germany
R. Seppelt, A.A. Voinov, S. Lange, D. Bankamp (Eds.)
http://www.iemss.org/society/index.php/iemss-2012-proceedings

Model representation, parameter calibration and parallel computing – the JAMS approach

Sven Kralisch and Christian Fischer

Department of Geoinformatics, Hydrology and Modelling,
Friedrich Schiller University, Jena, Germany
(sven.kralisch@uni-jena.de)

Abstract: To tackle problems of integrated environmental management, flexible and powerful simulation models are needed in order to analyse the current state of natural systems or to project their future dynamics under given development scenarios. Beyond the mere simulation of physical processes, accompanying tasks like model calibration or optimization for specific hardware platforms are usually requested here. The Jena Adaptable Modelling System (JAMS) is an open-source software platform that has been especially designed to address the demands of a process-based environmental model development and various aspects of model application. This paper gives an overview of JAMS and its underlying concepts and shows how its explicit representation of model structure and modelling entities can support parameter calibration and parallel computing.

***Keywords**:* Environmental modelling; modelling frameworks; model calibration; parallel computing.

1 INTRODUCTION

Software frameworks and accompanying standards that allow for an easy implementation and linking of integrated environmental models have gained increasing attention during the last decade, both from model developers and users. The systems that have emerged range from pure interface solutions (i.e. focusing more on coupling already existing models) to simulation frameworks that also cover the creation of problem-tailored simulation components along given requirements. Reviews and comparisons of currently available environmental modelling frameworks and their underlying techniques can be found in Argent (2005), Rizzoli et al. (2008) and Jagers (2010).
The *Jena Adaptable Modelling System* (JAMS)[1] is a simulation framework developed with a thematic priority on hydrological processes (Kralisch and Krause, 2006; Kralisch et al., 2007). Its focus is not on the coupling of existing environmental models but on the creation of problem-tailored models from well-defined process simulation components. These components simulate e.g. interception, potential evapotranspiration or soil temperature with conceptual or physically based algorithms. Making use of the Java Native Access (JNA) library[2], JAMS components may even wrap existing functionality offered via native shared libraries that were compiled e.g. from C/C++ or Fortran code. With regard to the spatio-temporal domain, JAMS aims to simulate environmental processes at discrete points in time and/or space. Such systems, often referred to as timed event system, are widely-used by many distributed hydrological models applied in current practice.

[1] http://jams.uni-jena.de
[2] https://github.com/twall/jna

JAMS is a JAVA-based framework, developed under the GNU Lesser General Public License. Depending on its application purpose it can be used in different execution environments (e.g. desktop and server based). In addition to functions for the creation and application of models, JAMS offers various software tools that cover frequently requested tasks in the environmental modelling context (e.g. for model calibration, parameter analysis and result visualization).
The next section will give an overview of the JAMS architecture and its main concepts. Section 3 is dedicated to describe how these support parameter calibration and parallel processing of JAMS models.

2 JAMS CONCEPTS

2.1 System architecture

The JAMS framework is structured into three main sections, i.e. (i) the core library, (ii) the runtime system, and (iii) the base component repository (Figure 1). The core library contains the API definition and different global support functions, e.g. for loading models or for model data I/O during simulation. The application of the models is managed by the runtime system, which is responsible for loading and executing JAMS models and provides additional services for event logging or profiling. The base component repository offers standard functions often used in environmental simulation models, e.g. providing geospatial processing capabilities. The core library provides interfaces and data types both for the creation of modelling components and the runtime system. The latter can create and execute a JAMS model by accessing the component repository and a model definition, which is defined by a XML document.

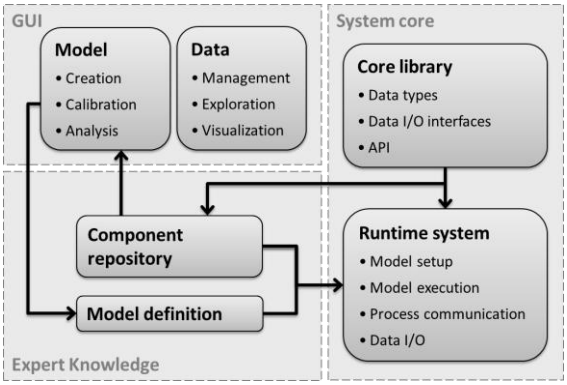

Figure 1. JAMS framework elements

2.2 Model building blocks

The main building block of any JAMS model is named *component*. A component is a JAVA class that implements some given interface defining different methods (*init*, *run* and *cleanup*). They have to be executed at according runtime stages that every JAMS model iterates through. Communication with the framework and other components is handled by arbitrary public attributes that fulfill two conditions: (i) they are of a valid JAMS data type and (ii) they are marked by special JAVA annotations, i.e. syntactic meta-information defining their I/O type (read, write), default values, physical unit and boundaries (if numeric), and their purpose by means of a short text. This information is used both by the runtime system to setup and interlink attributes and by the graphical user interface (GUI) that provides support during model design.
JAMS components can serve a variety of different purposes. The most important one is the simulation of environmental processes. An example is the calculation of potential evapotranspiration (PET), taking wind speed, temperature, humidity, radiation and elevation as input and calculating PET as output, e.g. according to after Penman-Montheith. Other components typically found in JAMS models implement data I/O or statistical analyses. They are applied e.g. for reading basic data like land use and soil type attributes or for aggregating process simulation results over spatial model entities in a region of interest. The source code complexity of a JAMS process simulation component can range from less than 50 logical executable lines of code (SLOC-L) (e.g. for calculating radiation input) up to more than 500 SLOC-L (e.g. for more complex snow simulation algorithms).

A special type of components is a GUI component. In addition to standard components they feature a graphical panel that serves as a container for arbitrary GUI elements. Added to a JAMS model, GUI components can be used to create graphical output during model runtime, e.g. for showing simulation results by utilizing chart and mapping libraries.
Components in JAMS are stateless objects, meaning that all processing information has to be provided via attributes at each invocation of the component's run method. Therefore, all state information (e.g. spatial attribute values) must be stored outside of the component.

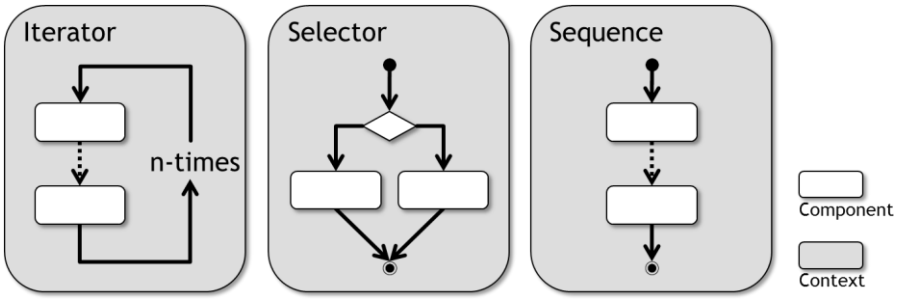

Figure 2. Context examples

Although generally possible, temporal or spatial iteration is not meant to be done inside of components. Instead, this is subject of the second type of model building blocks, called *contexts*. A JAMS context is a special, compound-type component that can nest other components and contexts, named children. It serves three purposes: (i) it controls the execution of its children; (ii) it controls the data exchange between its children, and (iii) serves as storage for its children's states. Depending on the purpose of a given context, it might invoke its children multiple times over a number of iterations (Figure 2 left), only once if some predefined condition is satisfied (Figure 2 center) or only once in a sequence (Figure 2 right). The invocation of its children is controlled in a context's run method, leaving full control over the way they are executed in the hands of the context developer. Due to this fact, contexts can address a wide variety of tasks, covering for example the application of different submodels depending on input data and user requirements or the calibration of models. As contexts are specialized components they can be nested in each other, allowing the creation of complex component hierarchies and execution control structures.
Spatial and temporal iteration is provided by the *spatial context* and the *temporal context*, iterating over a set of spatial model entities or time steps respectively and invoking their children for each of them. A typical application of both contexts is shown in Figure 3, with a spatial context nested inside of a temporal context. Using this setup, conceptual models that simulate environmental processes at discrete points in space-time can easily be represented in JAMS.

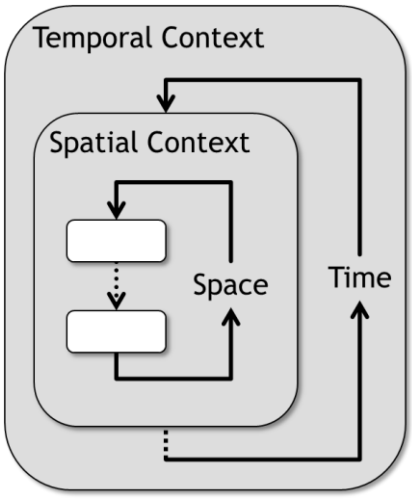

Figure 3. Nesting of temporal and spatial contexts

2.3 Component data exchange

Data exchange between components is managed by their surrounding contexts. For this purpose, each context features at least one state object which allows the storage of arbitrary JAMS data. These data are stored in data slots that can be dynamically added and removed as needed. By connecting their attributes to the

same data slots, children can easily exchange data during model runtime (Figure 4).
Sometimes it is necessary that a context can store more than one state, as it is the case with spatial contexts. In this case, each spatial model entity is reflected by a separate state, with its spatial attributes (e.g. elevation or slope) stored in data slots. While iterating over its spatial model entities, the spatial context can restore an entity's state prior the invocation of its children and store it afterwards. This way, a context can provide access to different sets of attributes and thus maintain data exchange between its children over varying spatial model entities (Figure 4).

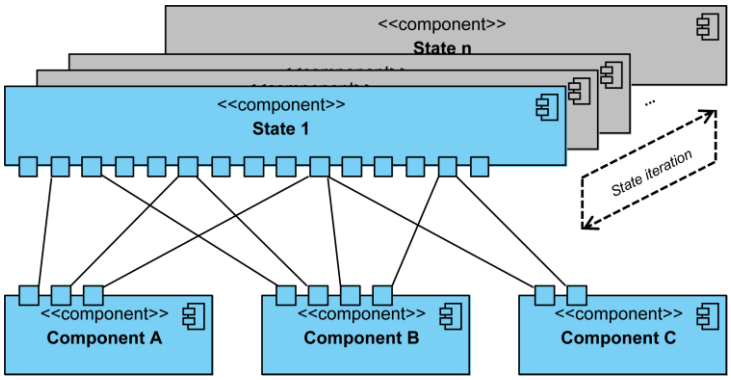

Figure 4. Data exchange between components using state objects

The same approach is applied for other context types where it is necessary to store and restore the state of their children (i.e. their input and output data) during execution. If there is no need to retain the state, the context uses only one state object, overwriting the data in its data slots in the case of repeated children invocation.

2.3 Runtime behaviour

A JAMS model is a special context called *model context*, which is a sequence-type context as shown in Figure 2 (right). Within this outermost context, other components and contexts can be arranged and nested as needed. The execution of the model is started by iterating through the init, run and cleanup stages of the model context. This in turn will start the invocation of its children according to a generalized activity scheme as shown in Figure 5. In the context's init stage, its own init method is invoked first before it will iterate over its children and invoke their init methods accordingly. During the run stage, the context will perform the following activity sequence depending on its behavior (e.g. in an iterated, conditional or sequential fashion): (i) restore the current state, (ii) iterate over all children and invoke their run stage, and (iii) save the current state. After the run stage has been finished, the context enters its cleanup stage, invoking the cleanup stages of its children first and its own at the end. This means that in contrast to other frameworks, e.g. as based on the Discrete Event System Specification (DEVS) (Zeigler, 2000), the invocation of

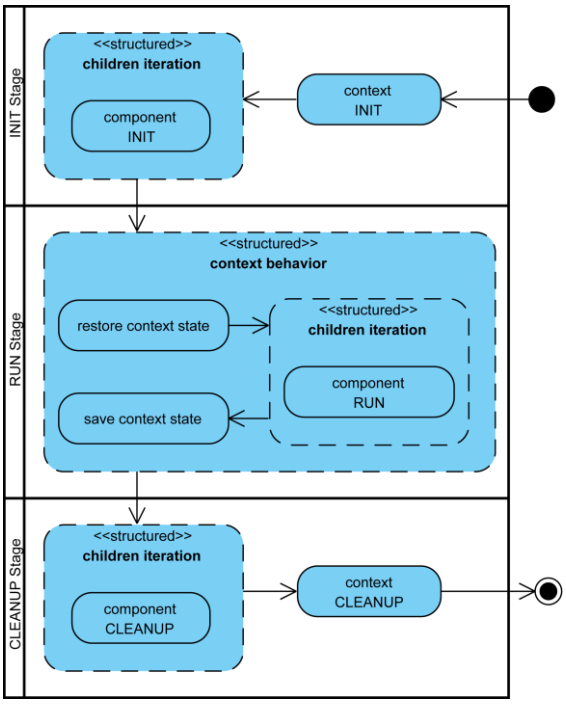

Figure 5. Context runtime activities

JAMS components is not autonomously triggered by external events, but explicitly by the surrounding context.
Using this approach, an entire JAMS model (i.e. model context) can easily be reused and combined with or nested in other components. A typical use case is model analysis and calibration, where a model context is added as a child to another context which allows the automated sampling of parameters of the internal model.

3 MODEL OPTIMIZATION

3.1 Use case J2000

Applying the pattern outlined in Figure 3, environmental models that simulate spatially distributed processes over time can be created easily. An example is the J2000 hydrological model (Krause, 2002) which has been successfully applied in a large number of catchment studies covering the lower meso- to lower macro-scale (Krause & Flügel 2005; Krause et al. 2006). J2000 simulates the water balance of hydrological catchments based on their spatial decomposition into hydrological response units (HRUs) (Flügel, 1996) for daily and sub-daily time steps. At each simulated time step, different runoff components are calculated for each HRU, followed by a spatial routing of the lateral runoff components from HRU to HRU or HRU to stream segment respectively.
Figure 6 shows the generic layout of J2000 in JAMS. The tasks executed during a simulation with J2000 can be sketched as follows:

1. The model is initialized by reading spatial input data, i.e. a set of HRUs including information about their land-use, soil and geological parameters and a set of stream entities as a basis for streamflow simulation. In addition, the sets of HRUs and stream segments are analyzed regarding their flow topology and ordered in such a way that an element is processed only after its contributors have been processed.
2. At each time step, the needed time series input data are read and steps 3 to 5 are executed.
3. For each HRU, a set of local input data is calculated (e.g. by spatial interpolation of climate data) and various hydrological processes are simulated (e.g. ET, interception and infiltration). Lateral runoff components are routed to HRUs and stream segments.
4. For each stream segment, streamflow is simulated.
5. Simulated data are aggregated and output.

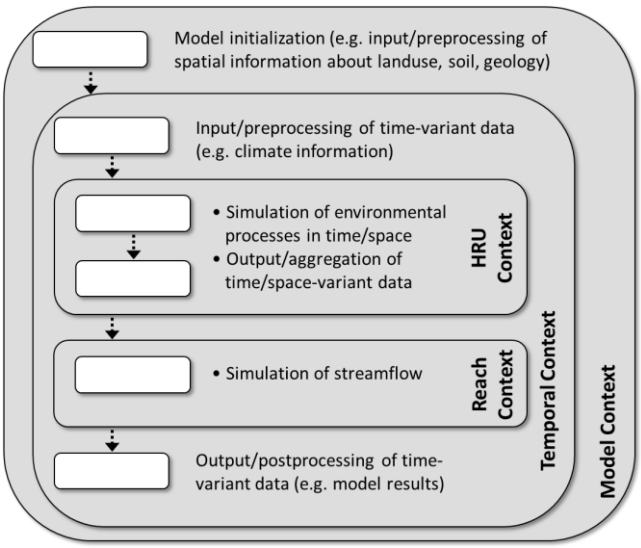

Figure 6. J2000 model layout in JAMS

3.2 Model parallelization

Spatially distributed models often have a high time and space complexity, i.e. depending on the number of spatial model entities and process detail they demand for large amounts of process memory and computation time. While memory consumption has become less constraining due to the availability of 64-bit operating systems and continuously dropping hardware costs, the runtime performance of environmental simulation models remains a crucial point. Thanks to multi-core architectures and cloud computing environments, the reason is not so much lacking CPU

power but model algorithms that are badly suited for concurrent processing. In general, two requirements must be addressed here which are often closely related:

1. The software platform used must offer support for the concurrent execution of processing routines.
2. The model's representation, i.e. its data and algorithms, must allow for an independent, parallel processing.

For modern software the first requirement is barely a problem as multithreading is a common paradigm supported by many current programming languages. Furthermore, software interfaces like MPI[3] offer standardized and well-tested support for distributed execution of simulation routines. The second requirement can become much more challenging as data and algorithms might induce strong interdependencies between different parts of the model. This problem can often be observed in distributed environmental models where energy and substances are exchanged between spatial model entities.

Due to its explicit representation of model structure and model entities, JAMS offers options to address both requirements. The concurrent execution of processing routines is achieved by isolating sub-models that can be processed independently from each other. A typical use case for this scenario is the calibration of simulation models. This can be done by embedding the model to be calibrated in a special JAMS context that controls the calibration process (Figure 7). The OPTAS optimization assistant (Fischer et al., 2009) uses this approach to setup parameter calibration procedures for any given JAMS model based on a variety of optimization methods and objective functions in a semi-automated way. In the setup shown in Figure 7, the model under calibration (right) can be executed concurrently for different samples of calibration parameters as interdependencies are not a problem here. The calibration context can be provided by the user as every other JAMS component, allowing calibration methods to be customized to high performance computing environments like computer grids if needed. Applying this approach, the GridGain[4] platform was successfully integrated and tested in OPTAS (Fischer et al., 2009).

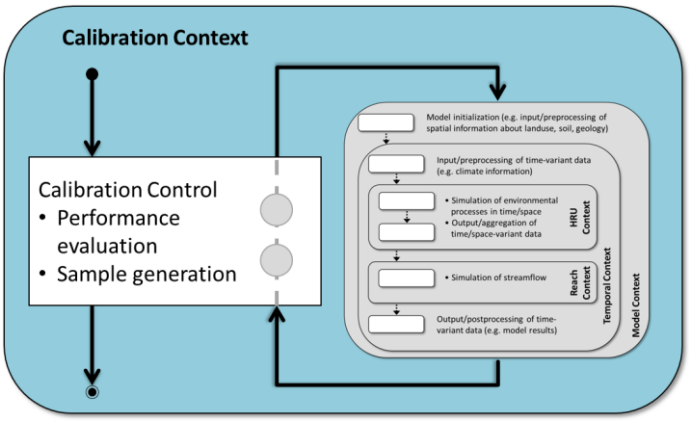

Figure 7. Model calibration in JAMS with calibration context (blue) and J2000 example model (grey)

Looking at an ever increasing amount of available environmental information and the associated growing complexity of simulation models, parallel processing has become interesting on a sub-model level, too. Taking into account that multi-core processing is available in virtually every computer nowadays, this step seems even more compelling. Models that simulate environmental processes in a spatially distributed way seem to be a good candidate for parallel computing, but as pointed out before, attention must be paid to interrelationships between their spatial model entities.

Applying this approach to JAMS and the J2000 model, only small changes have to be applied to the model layout, leaving the existing process components untouched. In a first step, the set of HRUs is partitioned into *n* HRU subsets, where *n* is the number of concurrent simulation processes. The challenge here is to find subsets that (i) can be processed independently and (ii) have a similar size. For hydrological models as J2000, the first requirement can be met by making sure that HRUs from the same sub-catchment are not distributed into different subsets.

[3] http://www.mcs.anl.gov/research/projects/mpi
[4] http://www.gridgain.com

While the second requirement is not mandatory, a better balanced partitioning leads to a higher speedup. Although strongly depending on the characteristics of the catchment (e.g. number of HRUs/sub-catchments) and the number of concurrent processes, experience has shown that a satisfying partitioning can be achieved in most cases. In a second step, *n* copies of the HRU context (cf. Figure 6) together with all its children are created and placed within a special context that replaces the HRU context (Figure 8). This *concurrency context* makes sure that each of the copies is provided with one of the previously created HRU subsets. The conversion of the model is done by JAMS in a fully automated way, leaving only the implementation of the concurrency context and the HRU partitioning component in the responsibility of the user.

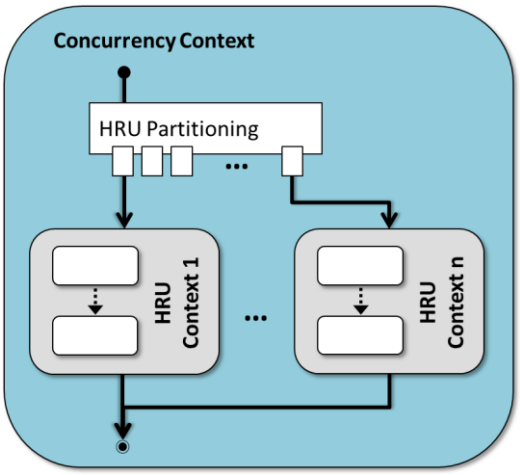

Figure 8. JAMS context for concurrent processing of HRUs

The presented approach was tested on a standard workstation computer with a current 4-core CPU. In order to consider the impact of the number of HRUs on the achieved speedup, two J2000 models for different catchments were tested. The first model was represented by 614 HRUs, the second model was about ten times larger with 6,242 HRUs. A time period of 4 years was simulated with both models, using simple JAVA multi-threading for concurrent processing.

Figure 9 shows the speedups that were achieved for both models using different numbers of CPU cores. Speedup is defined as the ratio between compute time using one core and compute time using multiple cores, i.e. a higher speedup means less compute time. The red dotted line marks the ideal speedup where the use of *n* cores results in *1/n* compute time. The data clearly show that J2000 is well suited for in-model parallel computing, even though only parts of the model were processed concurrently (i.e. all components within the spatial context). Regarding the fact that nearly all physical processes are simulated here which sums up to about 85% of the overall compute time (135s for the larger model), this result is not that much surprising.

With growing number of cores, the slopes of the speedup graphs decline which indicates that an increase of CPU cores will pay off only up to a certain point where the additional computational effort for parallel computing fully compensates the associated gain. As indicated in the graph, this point is reached earlier when using less HRUs which is explained with the worse ratio of fixed vs. parallelizable computational effort.

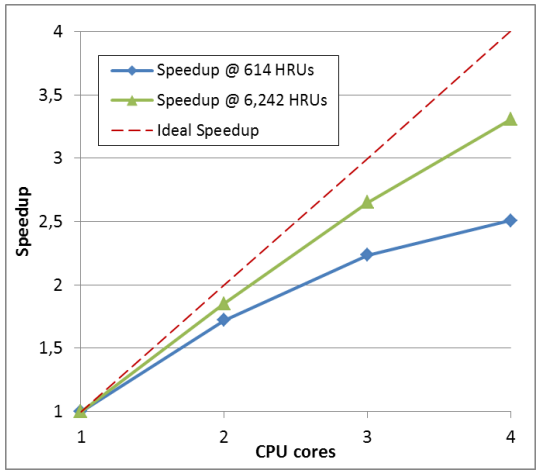

Figure 9. Speedup of J2000 models

4 SUMMARY & CONCLUSION

After introducing the main concepts of creating environmental models with JAMS, special attention was given to the adaptation of models to the requirements raised by model calibration and parallel computing. Due to the explicit representation of their structure and spatial modelling entities, JAMS allows to perform this task with-

out touching their processing components. After looking at parameter calibration methods, a generalized approach for model-internal concurrent processing was discussed. This approach uses standard JAMS components only, while existing process simulation components can be used without modification. The method was tested with two hydrological models featuring different numbers of modelling entities and proved to be more effective on the larger model. Ongoing work is focusing on fine-tuning the partitioning of spatial model entities and on the automated deployment and parallel processing in cloud computing environments.

ACKNOWLEDGEMENTS

The authors acknowledge the support of the German Ministry of Education and Research which has funded the JAMS development as part of the InnoProfile program (grant number 03IP514).

REFERENCES

Argent, R.M., 2005. A case study of environmental modelling and simulation using transplantable components. Environmental Modelling & Software 20, 1514–1523.

Fischer, C., Kralisch, S., Krause, P., Fink, M., Flügel, W.-A., 2009. Calibration of hydrological model parameters with the JAMS framework, in: Anderssen, R.S., Braddock, R.D., Newham, L.T.H. (Eds.), Proceedings of the 18th World IMACS Congress and MODSIM09 International Congress on Modelling and Simulation. Cairns, Australia, pp. 866–872.

Flügel, W.-A., 1996. Hydrological response units (HRU's) as modelling entities for hydrological river basin simulation and their methodological Potenzial for modelling complex environmental process systems - Results from the Sieg catchment. Die Erde 127, 43–62.

Jagers, H.R.A. (Bert), 2010. Linking Data, Models and Tools: An Overview, in: Swayne, D.A., Yang, W., Voinov, A.A., Rizzoli, A., Filatova, T. (Eds.), Proceedings of the iEMSs Firth Biennial Meeting: International Congress on Environmental Modelling and Software (iEMSs 2010). International Environmental Modelling and Software Society, Ottawa, Canada.

Kralisch, S., Krause, P., 2006. JAMS - A Framework for Natural Resource Model Development and Application, in: Voinov, A., Jakeman, A., Rizzoli, A.E. (Eds.), Proceedings of the iEMSs Third Biannual Meeting. Burlington, USA.

Kralisch, S., Krause, P., Fink, M., Fischer, C., Flügel, W.-A., 2007. Component based environmental modelling using the JAMS framework, in: Kulasiri, D., Oxley, L. (Eds.), Proceedings of the MODSIM 2007 International Congress on Modelling and Simulation. Christchurch, New Zealand.

Krause, P., 2002. Quantifying the impact of land use changes on the water balance of large catchments using the J2000 model. Physics and Chemistry of the Earth 27, 663–673.

Krause, P., Bende-Michl, U., Bäse, F., Fink, M., Flügel, W.-A., Pfennig, B., 2006. Multiscale Investigations in a Mesoscale Catchment – Hydrological Modelling in the Gera Catchment. Advances in Geosciences 9, 53–61.

Krause, P., Flügel, W.-A., 2005. Integrated research on the hydrological process dynamics from the Wilde Gera catchment in Germany, in: Headwater Control VI: Hydrology, Ecology and Water Resources in Headwaters. Presented at the IAHS Conference, Bergen.

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., 2008. Chapter Seven Integrated Modelling Frameworks for Environmental Assessment and Decision Support, in: Environmental Modelling, Software and Decision Support. Elsevier, pp. 101–118.

Zeigler, B., 2000. Theory of modeling and simulation : integrating discrete event and continuous complex dynamic systems, 2nd ed. ed. Academic Press, San Diego.