The hydrological modelling system J2000 - knowledge core for JAMS

[1]**Krause, Peter** **and** [1]**Kralisch, Sven**

[1]Department for Geoinformatics, Hydrology and Modelling
Friedrich-Schiller-University Jena, Löbdergraben 32, 07743 Jena, Germany. E-Mail: p.krause@uni-jena.de

Keywords: *J2000, JAMS, Modular Modeling Framework, Hydrological Modelling*

EXTENDED ABSTRACT

Hydrological model development and application for research projects is most often a cycle of model selection, model application, model adaptation and enhancement. The reason for the adaptation of an existing one or the development of a new model is that most of the conceptual hydrological models have been developed for a specific test catchment, scale and problem focus. The transfer to other catchments, other scales or other problems very often reveals some systematic drawbacks in the model's concept, its data and parameter needs or its capability to reflect the problem the user is interested in. This often leads to an extension of the selected model or in the worst case to model rejection. Additionally, new process knowledge becomes continuously available because of the very active research carried out in catchment hydrology world wide. From time to time this forces model updates so that the models always reflect the state of the art.

In general it can be stated that the perfect model does not exist yet and will probably never exist in the future. Therefore a more future-proof approach would be to think of hydrological modelling systems as toolboxes which should help model developers and applicators in the adaptation of existing models for their specific needs. To assist such a purpose some fundamental considerations have to be addressed during the development and implementation. The J2000 (J2K) modelling system which is presented in this paper, takes these into account by its object-oriented modular approach.

J2K provides the process knowledge core for the Jena Adaptable Modelling System (JAMS) currently under development, which can be considered as a generalized Java framework for the application of environmental model components. The development of JAMS was driven by some disadvantages the J2K currently has. This is first of all a lack of performance which is a follow up of the increasing flexibility, functionality and the more dynamic concept realised during the continuous development of J2K. It has to be noted that such a decrease in performance is common due to an increase in flexibility and functionality. Nevertheless we are convinced that considerations of how software performance can be increased are not only of interest for shorter model execution times. Moreover it forces the development of lean and straightforward concepts and implementations of system and process components, which is a very important prerequisite for future proof software development.

In the paper a short overview of the current J2000 modelling system's design and components and the basic approach of JAMS will be given. A short description of an application from a mesoscale catchment in Germany will complete the paper.

1 THE J2000 MODELLING SYSTEM

The development of the J2000 modelling system started in 1997 as process oriented hydrological modelling in large river basins. The reason for the development of a new system was that the distribution concept of the Hydrological Response Units (Flügel 1995), which was selected for the project work, had not been adapted to large scale basins at this time. The first version of the modelling system, implemented using C++, was successfully applied in three large subbasins (Mulde, Unstrut and Schwarze Elster) of the river Elbe in Germany (Krause, 2001). A thorough review which was carried out after the first applications, revealed that the J2000's systematic concept was not flexible enough to use the system in different environments or for different purposes or scales without further development. In particular, the separation between the system core and the process modules was not perfect in the first version. These limitations led to a new implementation of the whole system in Java. The new development was partly influenced by the Modular Modelling System MMS (Leavesley et al. 1996) and the Object Modeling System OMS (Ahuja et al. 2005). On the one hand, it was designed as a modelling framework for hydrological purposes and is therefore much more constrained than the OMS, which is implementing a much wider perspective. On the other hand, it is more open and flexible compared to MMS, which was developed as a platform for the Precipitation Runoff Modeling System (PRMS) (Leavesley et al. 1983).

The system layout of the J2000 is implemented in such a way that the system core and the more generic components are strictly separated from the knowledge part, which is assembled by the process modules. The interaction and communication between the system and the process modules is managed by the run class of the system which acts as a translator between the system and the processes.

The core components implement tools and methods for data in- and output, a graphical user interface and generic methods for data regionalisation, statistical analysis and visualisation. Additionally, a spatial context is provided by basic classes for spatial objects like distribution units, sub-catchments, the whole catchment and river reaches. The different spatial objects which are needed to set up a model are initialized and parameterised based on object parameter files containing all relevant attributes for describing the objects and their topology. The actual adding of spatial objects to an existing model is task of the system core which takes care of all necessary steps.

The system is complemented by the hydrological process modules implementing different parts of the hydrological cycle like evapotranspiration, interception, snow and soil water assessment, groundwater and flood routing, which form the knowledge part of the system.

The experiences made during the development of J2000 and and also OMS was providing the baseline for JAMS.

2 JAMS

JAMS (Jena Adaptable Modelling System) has been developed as a generalized Java framework for the application of J2000 model components. In order to identify basic demands for such a framework J2000 has undergone a detailed system analysis. Here we especially focused on the representation of the temporal and spatial domain within J2000. Emphasis was also placed on data exchange between model components in order to not only provide flexibility but also maximum performance. As a result of this analysis we identified the following requirements to be addressed by JAMS:

1) The *representation of multiple, nested temporal contexts* is a basic precondition for the incorporation of model components that need to be applied in *different temporal resolutions* (e.g. daily and hourly mode),

2) Accounting for *multiple, nested spatial contexts* is a prerequisite for the integration of model components that consider *different spatial resolutions* (e.g. HRUs and subcatchments) and *discretization types* (e.g. raster cells or polygons).

3) The ability to represent *spatial model entities as compound objects* – each of them holding a set of user-defined attributes – is a precondition to incorporate model components that are based on arbitrary spatial contexts.

4) Data exchange between model components has to be realized in a *high-performance manner* in order to overcome a major drawback of component based modelling systems compared to conventional simulation models. Moreover, this data exchange has to be implemented in a *flexible* manner that allows the mapping of arbitrary outputs of component A to arbitrary inputs of a subsequent component B if data types are compatible.

All of the abovementioned requirements have been met by the JAMS architecture which is currently being implemented. For data exchange between model components special glue classes are being generated during runtime. Instances of these classes can then be used in order to access the data attributes of providing and requesting model components.

The control of component execution and data exchange is exercised by the JAMS runtime environment. This environment is parameterised by an XML-based user-defined configuration that specifies the model structure and corresponding metadata.

3 J2000 SYSTEM CORE

The system core of J2000 is assembled by packages, classes and methods which are common and generic for different model setups and process implementation. Because of their stronger technical meaning they are generally not of major interest for a hydrologist and it is not likely that they have to be adapted when new hydrological process knowledge becomes available. The functionality that a model developer needs to integrate his or her own process modules is provided by few documented methods.

The core provides packages for the processing of time series data, for in- and output of parameter files, data and results, for initialisation, parameterisation and processing of spatial objects, and various tools for statistical analysis, regionalisation, physical calculations, geographical transformation etc. In addition, specific GUI components for the creation and display of different views (e.g. diagrams, tables) of model data and for user interaction and feedback are part of the core package.

3.1 J2kRun class

The J2kRun class is responsible for the execution of existing models. The run-class gets the relevant information (e.g. start and end date) from a user dialogue and provides a spatial iterator nested in a temporal iterator. The ”run method” of each process module is called for each spatial unit and time step by these iterators. In addition the run-class is responsible to produce output in form of text files or pass the data to the visualisation tools to create diagrams of the modelling results.

3.2 The data package

The data package of the system core provides classes and methods for the instantiation and provision of time series and other data necessary to setup a model. For processing of time series data the class ”j2kStationDataSet” is once instantiated for each type of input data (e.g. temperature, wind speed, observed runoff). The class itself instantiates several classes of the ”j2kStation” class depending on the number of stations for which input data is available. Each j2kStationDataSet is described by attributes which contain information about the type of input data, its temporal resolution, the start and end data and the value for missing data entries. The instances of the j2kStation class are described by additional attributes like station names, geographical coordinates and elevation of the measuring point. Such information is stored together with the actual data in simple ASCII files, which are read in during the instantiation.

The ”j2kParameterSet” class is responsible for implementing and processing of other data like describing parameters of landuse classes or soil types. This class reads in attributes and parameters from ASCII files and provides interfaces for the retrieval and manipulation of such data inside a model.

3.3 The io package

The io package assembles classes and methods responsible for all physical file in- and output of models. The package provides classes for file related issues when a new model is setup as well as classes for writing and loading of existing models. When a model is saved the whole set up including processes and data is serialized to one object on the hard disc. In addition a restore file is created which contains a meta-description of the current model. The restore file is helpful when the deserialisation of an existent model fails for some reason or the user likes to change a specific model setup from outside the system. For rebuilding a model based on the restore file a special class exists in the io package.

Besides the methods described above the io package contains classes to create table views for the various data and attributes of a model. Such classes comprise a specific table model class and a table viewer for the user friendly display of tables.

3.4 Space and time in J2000

Space is represented in J2K by spatial objects which carry hydrological information and, therefore have to be considered and processed during modelling. In practical terms, these might be raster cells, hydrological response units, sub-catchment and catchments, but also river reaches or point objects like lysimeters.

The spatial objects needed to setup a model are either parameterized by parameter files which contain attributes describing the entities or constructed from already existing spatial objects. For instance, a model implementing the distribution concept of hydrological response units (HRUs) is instantiating a set of such units by reading attributes (i.e. coordinates, area, elevation etc.) from a parameter file. The attributes of the HRU instances can then be used to instantiate and construct the catchment object.

Beside the temporal static attributes, the spatial objects can obtain temporal dynamic state variables from process modules when such are installed into the system. During model execution the state variables are changed by processes taking place in the process modules.

As J2K is a modelling system for continuous hydrological simulation it requires a temporal context. This context is provided by the J2kDate class which is extended from the GregorianCalendar class of Java. Instances of the J2kDate class provide robust and consistent time environments for the modelling. The class has getter and setter methods for the construction or update of J2kDate objects as well as transformation functions from calendar date to Julian days. Additional functions allow e.g. testing if an actual date object lies within specific constraints given by start and end dates, if a date is before or after another one, or how many days or hours a specific time span comprises.

3.5 Geographical tools and Data analysis

A basic package with geographical tools has been implemented which helps the user in the conversion of different projected data sets. Currently only conversion of Gauss-Krüger coordinates to latitude and longitude in decimal degree and vice versa are available but transformation functions for other projections like UTM can simply be added.

The analysis package of the J2000 system provides different classes for the quantification of model performance and a basic class for sensitivity analysis. Implemented performance measures are the Nash-Sutcliffe efficiency with normal and logarithmic values as well as a modified version which allows other powers than 2, linear regression with the coefficient of determination and its gradient, double sum analysis of simulated and observed variables and the index of agreement. A comprehensive description of all efficiency measures implemented in J2K can be found in Krause et al. (2005).

The J2kSensitivity class provides tools for quantifying parameter sensitivities by one-, two- and multi-dimensional analysis. The two-dimensional analysis can be used to detect parameter interdependencies. From a dialogue the user can choose two process modules and related process parameters. The parameter range which should be investigated is specified by defining upper and lower boundaries (Min, Max) and the resolution (Res.) of the parameter space for both variables. A two dimensional matrix is constructed from the boundaries and the resolution of the variables during the sensitivity analysis. Inside the matrix fields random variable pairs are defined by a Latin-Hypercube method. Then the model is run for all realisations of the variables and different performance measures are calculated. The output is given as text file, which contains the various realisations and the resulting performance measurements as well as 2D graphs. The visual interpretation of the output in form of 2D graphs shows not only the influence of different parameter values on different performance measures but also the dependencies of the two chosen parameter from each other.

One- to multi-dimensional sensitivity analysis can be performed by Monte-Carlo-Analysis combined with a Latin-Hypercube search algorithm. Here the user can select as many variables as he likes and can perform multiple runs with different random parameter combinations. The output is stored in single files which hold the parameter values, different efficiency measures and the resulting variables from each run.

3.6 Visualisation and Graphical User Interface

One of the predefined goals which were guiding the development of the J2000 modelling system was to make the system user-friendly and attractive. The idea was not only to produce something appealing but moreover to provide an easy to use tool for model application and development as well as to encourage other scientists to contribute to the system and process libraries. Part of this strategy was the provision of an attractive and intuitive graphical user interface (GUI) differing from those most often found in hydrological modelling software. In addition, some effort was also spent on the integration of Java packages for the graphical output of modelling results and input data in form of graphs and diagrams. Fig. 1 shows a screen shot of the J2000 main frame.

Plotting capabilities based on the JFreeChart package (www.jfree.org/jfreechart) were implemented into J2000 for the analysis of input data and model results. With these tools the user can freely configure different time series plots for each of the calculated variables with just a few mouse clicks. Furthermore, the graphs can be zoomed, altered, printed and exported. Figure 1 and 3 shows some examples of the plotting capabilities of J2000.

Most of the J2000 plots present integrated results for the entire catchment. To view the distribution of the hydrological conditions, spatially variable output can also be created. However, at the current state this is only possible as tabular data, which then can be visualised in a GIS. A graphical front-end renderer, which will allow the visualisation of GIS maps of distributed parameters and model results, is under development and will be implemented in the near future.

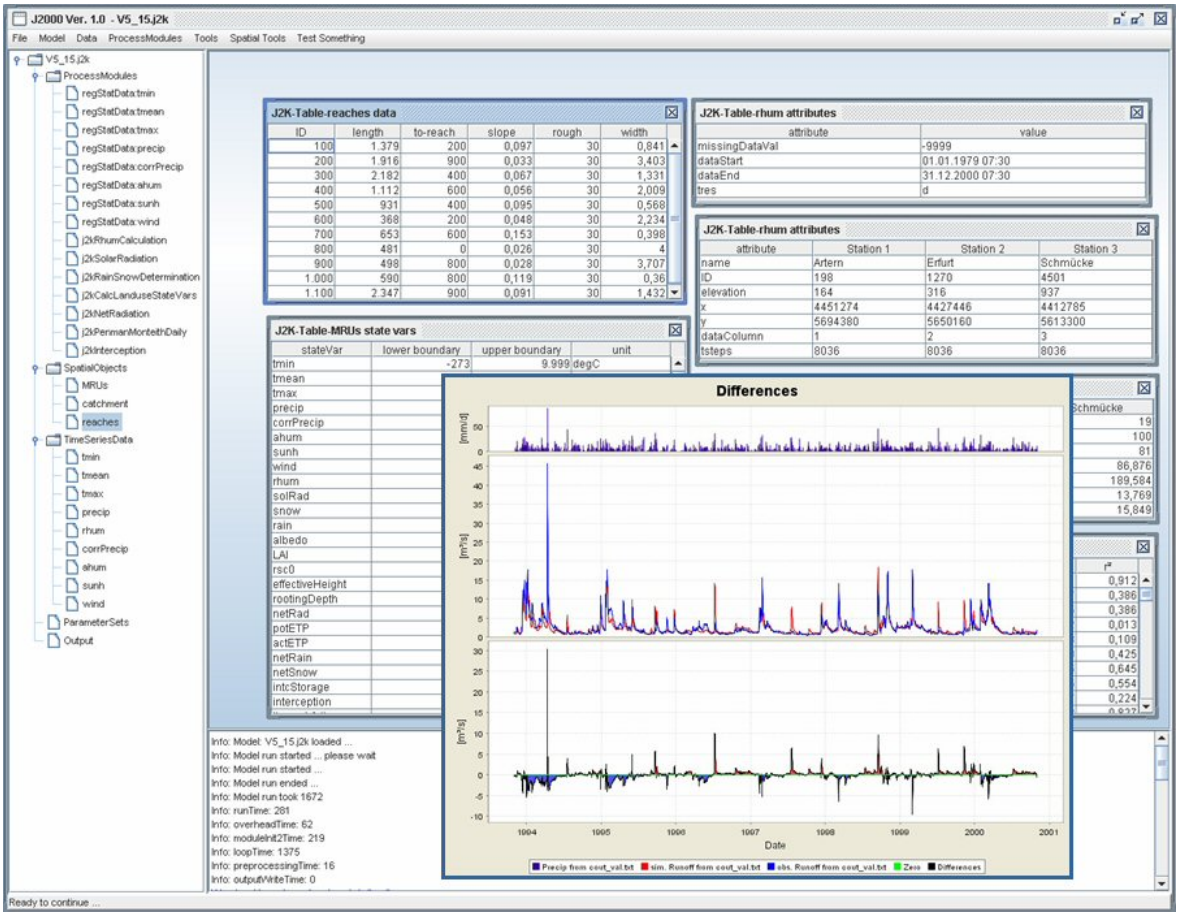

Figure 1. The graphical user interface of J2000, showing various views of data, attributes and results

4 PROCESS COMPONENTS

The system components described above were developed in such a way that the modelling system is as open as possible for various tasks, different model setups and user requirements. Even though the J2000 was developed as a hydrological modelling system, most of the generic components may be useful for modelling of other natural environments. The knowledge part in form of process components was strictly divided from the system part to keep the system open for various hydrological concepts and other purposes beyond.

For the J2K a complete set of process modules exists which are contains methods for regionalisation and correction of time series data and the single processes for distributed hydrological modelling, like potential ETP, interception, snow, soil water, ground water and reach routing. A detailed description of the processes can be found in Krause (2001 and 2002).

4.1 Process template

The process modules have to follow a template which implements the following four important methods: install, init, run, and cleanup.

The install() method is responsible for inserting the module correctly into the system's context during module installation. The method is exporting the module's state variables to the specific spatial objects and is responsible for assuring that the module receives the necessary input. During runtime the run() method is called for each time step and spatial unit. The run() method itself calls first init() to retrieve the actual system status, then it performs the process specific calculations and finally calls cleanup() to pass altered variables back into the system.

For instance, an interception module might need the actual leaf area index (LAI) together with daily rainfall and daily potential ETP of a distribution unit implemented as spatial object. With this information the module calculates maximum interception storage (depending on LAI) and reduces the daily rainfall to throughfall by subtracting interception. The intercepted precipitation can then partly or fully evaporate depending on the potential ETP.

4.2 Process module implementation and installation

For the implementation of process modules a parent j2kProcess class exists from which specific process implementations are derived. The parent class implements the install(), init() and cleanup() methods which can either be directly used or can be overwritten by the user if needed.

The simplified code shown in fig.2 should illustrate the layout and implementation of a J2000 process module. In this example a simple bucket module is shown which is extended from the parent class J2kProcesses. It has two local variables (precipitation and bucket) and a calibration parameter (maxStorage). When the module is instantiated and its constructor is called the variable maxStorage is mapped into the module's parameter hash-map and is therefore available for calibration or sensitivity analysis by the system or for user interaction.

In the install() method the module's "bucket" is added to each modelling unit as a state variable and initialised with the value 0. When the model is processed in a temporal and spatial context the run() method is called by the system, which is first of all invoking init(). In init() the module retrieves the actual precipitation of the current unit and the state of its bucket and maps them to the local variables. Then the process implementation of run()is executed where the precipitation is added to the bucket. When the bucket exceeds its maximum storage capacity the user is informed by a J2K information message and the bucket storage is set to zero again. Finally, run() invokes cleanup() in which the changed bucket content is passed back to the units bucket state variable. If the module contains calibration variables, like "maxStorage" which the developer wants to be accessible for tuning a parameter dialogue is added by the parent class j2kProcess to the module automatically.

```
public class j2kBucket extends j2kProcess{
   double precipitation;
   double bucket;
   double maxStorage;
   public j2kBucket(){
      this.addCalibrationParameter("maxStorage", 100);
   }
   public void install(spatialObject[] MRUs){
      for(int i = 0; i < MRU.length; i++){
        MRUs[i].addStateVar(„bucket, 0);
      }
   }
   public void init(spatialObject MRU){
     this.precipitation = MRU.getStateVarVal("precip");
     this.bucket = MRU. getStateVarVal("bucket");
   }
   public boolean run(spatialObject MRU){
      init(MRU);
      this.bucket = this.bucket + this.precipitation;
      if(this.bucket > this.maxStorage){
        j2kMainFrame.infoMsg
              ("Bucket is full and will be emptied now");
        this.bucket = 0;
      }
      cleanUp(MRU);
      return true;
   }
   public boolean cleanUp(spatialObject MRU){
      MRU.setStateVarVal("bucket", this.bucket);
      return true;
   }
```

Figure 2. Simple example of a J2000 process module.

5 APPLICATION OF J2000 IN THE MESOSCALE GERA CATCHMENT

The following section will give a very brief overview of an application of J2000 for a research project to show the potential for hydrological process modelling. A more comprehensive and critical discussion of the hydrological modelling in the Gera basin can be found in Krause & Flügel (2005).

The J2000 was applied in the mesoscale Gera basin in Germany for a research project carried out in scope of the implementation of the EU Water Framework Directive. The Gera basin has an area of 850 km^2 and is very heterogenous in terms of topography, landuse, soils and geology. The main goal was to simulate the vertical and lateral water transport processes as good as possible in a fully distributed way to add nitrate process dynamics during a second phase, which is currently in work. As distribution concept 13769 topological connected HRU polygons were delineated from GIS layers of topography, landuse, soil types and hydrogeological units. For the modelling, the entire catchment was divided into four subbasins, which were modelled separately in a nested catchment approach. A thorough model calibration was carried out for the most southern basin "Arnstadt" on four years of the eleven year time-series by a comparison of observed and simulated runoff at the subbasin outlet. Therefore, the calibration parameters were tuned until a Nash-Sutcliffe efficiency of > 0.7 was achieved. The model was then validated with the remaining seven years of the time series. In addition the spatial and temporal distribution of the runoff generation was inspected. After calibrating a good and plausible parameter set, the model was transferred to the remaining subbasins and slightly recalibrated until a reasonable model fit could be observed. The resulting efficiencies for the subbasins and the total basin are shown in table 1. A graphical view of the model results of subbasin Arnstadt is shown as the diagram in fig 1. The upper panel of the diagram shows the precipitation, the middle panel the observed

(blue line) and the simulated (red line) runoff and in the lower panel the differences between observed and simulated runoff is shown.

Table 1. Nash-Sutcliffe efficiencies for the three subbasins and the total Gera basin

subbasin	calibration	validation
Arnstadt	0.7090	0.8317
Apfelstädt	0.6304	0.7573
Wipfra	-0.0843	0.5279
Gera total	0.6654	0.7436

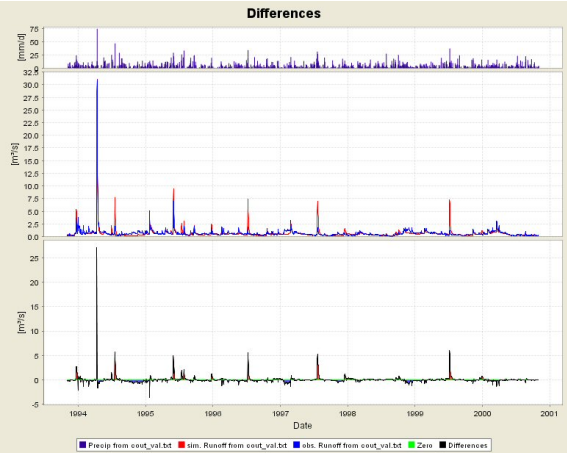

Figure 3. The systematical overprediction of peak flows in the subbasin Wipfra in the validation period 1994 - 2000.

The efficiencies summarized in table 1 show that the model was performing well in the subbasins Arnstadt and Apfelstädt, but produced bad results in the subbasin ”Wipfra”. The reason for the bad efficiencies in these subbasins is a flood retention dam in the upper part of the catchment, which is reducing flood peaks at the outlet significantly. In the current version of J2K the influences of such dams can not be simulated. Therefore, the model was overpredicting peak flows systematically. The modelling results for the validation period in subbasin Wipfra are shown in fig.3. The systematic overprediction can particularly be seen in the differences plot (lower panel).

6 CONCLUSIONS

The current version of the J2000 modelling system can be considered a flexible and easy to use model environment for hydrological purposes. Various applications in Germany and also South Africa showed that the modelling system is able to simulate the hydrological process dynamics fully distributed with reasonable quality. The model has been successfully applied in basins with areas from less than 2 km^2 up to more than 6000 km^2 which demonstrates in particular the multi-scale ability of the implemented process descriptions. During the development from the first version up to now the system became more and more flexible. Alas, the growing flexibility led to a decrease in model performance mostly because of the more flexible data exchange methods between single sub-components based on hash-maps, which results in frequent internal type casting that slows down the system. Experiences made by the transfer of the model to other basins revealed also that the implementation of the spatial and temporal context of the current system is sometimes not flexible enough. Here, a more common approach for providing variable temporal and spatial environments would be an advantage.

Such disadvantages led to the development of JAMS, which is mostly concentrating on performance issues but also intends to provide even more flexibility to the entire system. The multiple, nested temporal and spatial contexts provided by JAMS open up new and better opportunities for problem tailored model setup and application.

7 REFERENCES

Ahuja, L.R., Ascough II, J.C. & David, O.: Developing natural resource models using the object modeling system: feasibility and challenges, Advances in Geosciences, Vol. 4, pp 29-36, 2005.

Krause, P., Das hydrologische Modellsystem J2000 - Beschreibung und Anwendung in großen Flußgebieten (The hydrological modelling system J2000 - Documentation and application in large river basins); Schriften des Forschungszentrums Jülich, Reihe Umwelt/Environment, Band 29, 2001.

Krause, P., 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, p. 663-673, 2002.

Krause, P., Boyle, D.P. & Bäse, F., Comparison of different Efficiency Criteria for Hydrological Model Assessment, Advances in Geosciences (in print), 2005.

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

Leavesley, G.H., Lichty, R.W., Troutman, B.M., Saindon, L.G., Precipitation Runoff Modeling System: User’s manual, Water Resources Investigations 83-4238, USGS, Denver, Colorado, 1983.

Leavesley, G.H., Restrepo, P.J., Markstrom, S.L., Dixon, M., Stannard, L.G., The Modular Modeling System (MMS): User’s manual, Open File Report 96-151, USGS, Denver, Colorado, 1996.