Advances in Geosciences, 4, 45–50, 2005
SRef-ID: 1680-7359/adgeo/2005-4-45
European Geosciences Union

Advances in Geosciences

DIVA: an iterative method for building modular integrated models

J. Hinkel

Potsdam Institute for Climate Impact Research GPO Box 601203 14412 Potsdam, Germany

Received: 1 August 2004 – Revised: 1 November 2004 – Accepted: 15 November 2004 – Published: 9 August 2005

Abstract. Integrated modelling of global environmental change impacts faces the challenge that knowledge from the domains of Natural and Social Science must be integrated. This is complicated by often incompatible terminology and the fact that the interactions between subsystems are usually not fully understood at the start of the project. While a modular modelling approach is necessary to address these challenges, it is not sufficient. The remaining question is how the modelled system shall be cut down into modules. While no generic answer can be given to this question, communication tools can be provided to support the process of modularisation and integration. Along those lines of thought a method for building modular integrated models was developed within the EU project DINAS-COAST and applied to construct a first model, which assesses the vulnerability of the worlds coasts to climate change and sea-level-rise. The method focuses on the development of a common language and offers domain experts an intuitive interface to code their knowledge in form of modules. However, instead of rigorously defining interfaces between the subsystems at the projects beginning, an iterative model development process is defined and tools to facilitate communication and collaboration are provided. This flexible approach has the advantage that increased understanding about subsystem interactions, gained during the projects lifetime, can immediately be reflected in the model.

1 Introduction

This paper presents a method for building modular integrated models. The method was developed and first applied within the EU project DINASCOAST (Dynamic and Interactive Assessment of National, Regional, and Global Vulnerability of Coastal Zones to Climate Change and Sea-Level Rise, www.dinas-coast.net. The aim of the three-year project is to develop a dynamic, interactive, and flexible tool that will enable its users to quantitatively assess coastal vulnerability to sea-level rise and explore possible adaptation strategies. Underlying are various climatic and socio-economic scenarios and adaptation policies on national, regional, and global scales covering all coastal nations. This tool is called DIVA, Dynamic and Interactive Vulnerability Assessment, and is centred around an integrated model.

DINAS-COAST was motivated by apparent limitations of previous global vulnerability assessments (Hoozemans et al., 1993; Baarse, 1995), including: the obsolescence of underlying data sources and the static, one-scenario approach. To overcome these limitations DINAS-COAST combines data, scenarios, and assessment models into an integrated tool, and makes it available to a broad community of end-users on a CD-ROM.

For the development of such a tool expert knowledge from the domains of Natural and Social Science must be integrated, calling for a modular approach to model development. Individual partners independently develop modules representing coastal sub-systems which are then "plugged" together to form one integrated model. While a modular modelling approach is necessary to address these challenges, it is not sufficient. The remaining question is how the modelled system shall be cut down into modules. While no generic answer can be given to this question, communicational and organisational tools can be provided to support the process of modularisation and integration.

Facing these challenges the DIVA method for modular integrated modelling was created. The method organises the development process and facilitates communication and cooperation. The actual DIVA tool is currently being built using this method. While the DIVA tool is specific to DINAS-COAST, the DIVA method can be reused in other contexts with similar requirements.

This paper first analysis the DINAS-COAST requirements as perceived from the perspective of model integration and software development (Sect. 2), then explicates some concepts of the modelling process needed for the following

Correspondence to: J. Hinkel
(hinkel@pik-potsdam.de)

discussions (Sect. 3). Section 4 explores the space of solutions to the requirements and Sect. 5 presents the DIVA method as a possible answer. Finally Sects. 6 and 7 list some limitations and conclusions, respectively.

2 Requirement analysis

The development of an integrated model faces several challenges. Knowledge from the domains of Natural and Social Science must be integrated. This is complicated by the often incompatible terminology, differing model types and modelling styles, and also by the fact that domain experts are distributed over various institutes worldwide. Frequent project meetings are not possible. Most of the model development must be coordinated via email, web-sites, and telephone calls.

While the requirements listed above are common to integrated modelling, some special challenges needed to be addressed in DINAS-COAST. Due to lack of an appropriate data source the model had to be developed simultaneously with its proper world-wide database (see Vafeidis et al., 2003). The interactions between sub-systems were not fully understood at the start of the project; instead, such understanding is a major result of the project itself. Both circumstances necessitated a flexible model design that accounts for the incorporation of new knowledge in form of data, algorithms, or sub-system interactions at any stage during the development process. Finally the DINAS-COAST model, together with the database, is meant to be made available to a broad community of end-users, such as scientist, politicians and coastal-planers. This calls for an easy-to-use graphical user interface and an efficient model.

3 Modelling process

This Section explicates four concepts involved in the modelling process that are needed for the following discussions.

1. **Ontology:** The modelling process starts with some concepts that we have at our disposal to perceive the world. It is good practise, especially in integrated modelling, to make this basic conceptualisation explicit. An explicit specification of a conceptualisation shall be called ontology.

2. **Mathematical problem:** Based on the ontology a mathematical problem is formulated. For example one might have a system of differential equations and be interested in knowing its evolution over time (initial value problem).

3. **Algorithm:** Since in most cases the mathematical problem cannot be solved analytically, it's solution must be approximated by applying numerical methods. The result of this step is the numerical solution or the algorithm.

4. **Computer model:** The last step considered here is the implementation of the algorithm in a programming language. This step yields the executable computer model.

4 Modular integrated modelling

An integrated model is composed of various submodels. It is evident that such a model, like other complex software, should be built in a modular rather than monolithic fashion: all contributers provide their knowledge about sub-systems in form of self-contained components (modules).

While modularity is a necessary answer for integrated modelling, it is not sufficient. Among others, four questions need to be addressed:

1. At which stage of the modelling process shall the integration take place?

2. What are the modules' interfaces or how shall the system be decomposed into sub-systems?

3. Which technology or software shall be used?

4. How shall the process of model integration be organised?

The following Subsections explore possible answers to these questions and motivate the decisions taken in the case of DINAS-COAST.

4.1 Integration level

The first question which arises in integrated modelling is at which stage of the modelling process the integration shall take place. Clearly, model integration has to start with a common ontology. Any attempt without a common conceptualisation of the system to be modelled is likely to fail. The remaining question is whether to integrate mathematical problems, algorithms, or executable computer models.

From an idealistic point of view models should be integrated at the level of the mathematical problems. Having a complete mathematical formulation of the system allows for careful selection of appropriate numerical methods and leads to stable and efficient algorithms. In praxis this route is seldom taken. Reasons for that are: the existence of legacy computer models; the need for a lot of cooperation at an early stage of the project; unclear linkages between sub-systems and that it is uncommon to "think" about integrated modelling in terms of mathematical problem specifications rather than algorithms and computer programs.

From a pragmatic point of view it makes sense to integrate existing computer models. Legacy models, in which a lot of development time was invested, can then be reused. The flip-side of the coin is that the coupling of computer models involves a lot of technical issues, due to the heterogeneity in platforms, computer languages, compilers and data structures involved. A further disadvantage of this approach is that due to the absence of a complete specification of the

mathematical problem it often remains unclear whether the numerics of the coupled computer models adequately represent the problem.

In the case of DINAS-COAST an intermediate approach was taken: the models were integrated at the level of the algorithms. Thus the project partners were free to solve their mathematical problem individually, but then had to implement the algorithms as modules in a common programming language. This route could be taken, because there were no legacy models to include.

4.2 Module interfaces

An elementary question of any modular approach to integrated modelling is how the modelled system shall be decomposed into sub-systems or, phrased differently: What are the modules' interfaces?

An efficient way of developing an integrated model would be to define specialised interfaces between the modules: Each module has its proper interface, specific to the subsystem it represents. That way, the data-flow between the modules is fixed with the definition of the interfaces. The development process would then be straight forward: At the beginning of the project the interfaces are defined, then the developers program their modules concurrently in accordance with the interface specification. At the end of the project the whole model is plugged together.

However, a distinguishing feature of interdisciplinary research is that interactions between subsystems are usually not fully understood at the start of the project. General interfaces that provide the freedom to define the data-flow between the modules during the course of the project are required. In the approach presented here all the modules have identical interfaces. They share a reference to the model's global state and are allowed to perform any read or write operation on it. Thus the actual data-flow between the modules is not fixed, offering the flexibility for taking advantage of the interdisciplinary learning process during the project's lifetime.

The generality of the interfaces has implications on the development process. While specialised interfaces would not require extensive collaboration between partners during module development, general interfaces do so. To organise the collaboration a rigorously defined iterative development process is introduced. Module development takes place in two phases: First, the modules are programmed individually with the freedom to read and write any property of the system's sate. In the second phase, the actual data-flow between the modules is analysed jointly. The two phases are iterated until a satisfactory result is achieved. A detailed description of the iterative development process is found in Sect. 5.2.

4.3 Technology

A wealth of methods and technologies from software engineering, like for example object-oriented programming or component technologies, are based on the concept of modularity. The necessity to build complex and integrated models has brought these techniques to the modelling communities and triggered the development of modelling frameworks.

Frameworks provide a conceptual frame, that is an abstract ontology for certain classes of the problems. Frames often support one (or several) modelling paradigms. For example an object-oriented framework for agent-based modelling might provide classes for agents, organisations, and environments. Models implemented in a framework use its basic concepts and specialise them further to their own needs.

An up-to-date overview of modelling frameworks developed within the environmental modelling community is given by Argent (2003). Most approaches, just like the one presented here, tackle model integration at the algorithmic level of the modelling process. Consequently, sub-models must be implemented in a framework-specific language. The route to integrate existing computer models implemented in different languages or on different platforms is taken by Leimbach and Jaeger (2004). Few approaches support model integration at the stage of the mathematical problem specification. Examples are the M software environment (Jos de Bruin, 1996) and the declarative modelling approach (e.g. Muetzelfeldt, 2004). Other mathematical approaches can be found within the the Decision Support Community. See Dolk (1993) for an introduction.

In the case of DINAS-COAST it was decided to develop a new framework. This was motivated by the will to provide the project partners with a very simple and efficient interface for expressing their knowledge. To this end the framework has to provide the "right" framing. If it frames too little a lot of coding needs to be done to express the specific problem. If it frames too much some aspects of the problem cannot be represented in the frame. A second motivation for developing something new was the aim to tightly couple the framework to tools supporting the actual process of integrated model development.

4.4 Organisation

Model integration is an organisationally challenging and communication intensive process. While there is a wealth of modelling frameworks framing model design, there is little framing communication and the process of model development. Model documentation and meta-data are first steps in the right direction (see for example Rahman et al., 2003).

The DIVA-Method emphasis and structures the process of integrated modelling. This necessity arose specifically from the requirement, that the model must be flexible to account for changes in interfaces, algorithms, and data-structures at any stage of model development.

5 The DIVA Method

The DIVA Method is a method for building modular integrated models by distributed partners. It consists of a conceptual frame (Sect. 5.1), an iterative development process (Sect. 5.2), a generic model (Sect. 5.3), and a build and

```
public class Country implements Feature {

    public float area;
    public int   population;
    public float gdp;
    public Region region;

    protected readParameters(DataInput in) {
       area       = in.readFloat();
    }

    protected readDrivers(DataInput in) {
       gdpc       = in.readFloat();
       population = in.readInt();
    }

}
```

Fig. 1. A generated Java feature class.

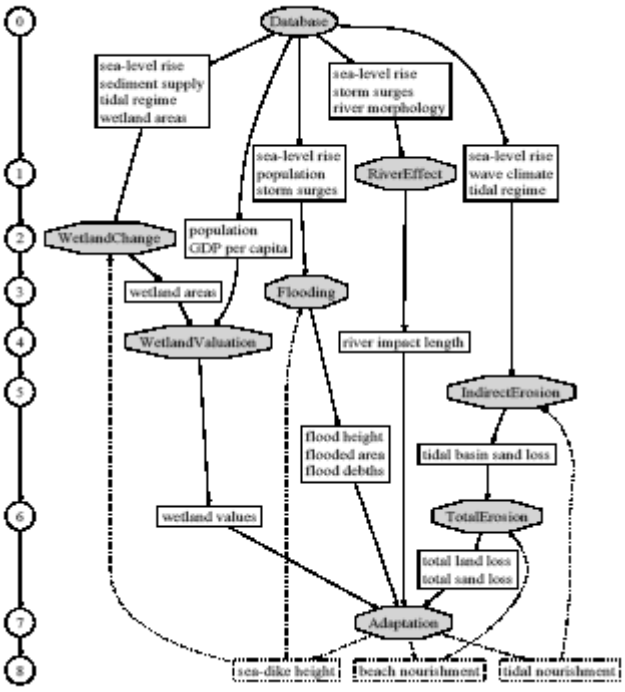

Fig. 2. Module linkages in the DINAS-COAST model. Ovals represent the modules, boxes represent data, the drawn through arrows represent the flow of data during one time step, and the dotted arrows represent the data fed into the next time step.

documentation tool (Sect. 5.4). The first two sub-sections describe the method from the point of view of the scientists developing the modules, while the last two sections deal with the technical implementation. The DIVA Method was designed to be generic and can be applied to problems with similar requirements as DINAS-COAST.

5.1 The frame

The DIVA Method provides, just like any other modelling framework, a conceptual frame for modelling. Only what can be expressed with the frame's concepts can be modelled by the DIVA Method. For modelling dynamical systems concepts for expressing static information about the system (data model) as well as concepts for representing the system's dynamics are needed.

The statics of the system is represented by a relational-data model consisting of geographic features, properties, and relations. The geographic features represent the real-world entities, like rivers or countries. Properties capture the quantitative information about the features; e.g. a country might have the property area or a river the property length. Finally, relations describe how the features are structured. For example the feature region might contain several country features.

The dynamics of the system is represented by first-order difference equations: the state of the system is a function of the state at the last time-step and the drivers. All properties of the features must be classified according to the role they play in the dynamics into the four categories: driver, state variable, diagnostic variable and parameter. For example the country's area would most likely be static, that is a parameter, while its population might be driving the model.

5.2 The development process

The first step of the development process consists in defining the model's ontology: Given the abstract frame the specific features, properties, and relations which constitute the modelled system must be specified. The main part of the ontology is the list of system properties which contains the property names, the features they belong to, their type (that is whether they are drivers, parameters, or variables), their data type (e.g. float, integer), and some other meta-information. The compilation of the ontology is a joint responsibility of the project consortium.

The ontology is then automatically translated into Java source code by the DIVA Build Tool (Sect. 5.4). For each feature one class is generated. The class contains public member fields for the feature's properties. Relations between the features are represented by class composition. Figure 1 shows generated code for a feature called country. The class has four public member fields: the first three hold the feature's properties (area, population, and gdp) and the last one points to the region the country belongs to.

In the next step the project partners code the algorithms. They express their knowledge about the dynamics of the system in form of difference equations written in Java and using the generated feature classes. Since now the model's ontology is hard-coded an algorithm will only compile if it is consistent with the ontology. Related algorithms are grouped into modules. Before a module is submitted for inclusion into the integrated model it is run and validated stand-alone.

The last step of the development process consists in the analysis of the modules, their linkages, and the validation of the complete model. Whenever a new version of a module is submitted the build tool automatically updates the project's

web-site, which offers documentation and the new model for download. Figure 2 shows a generated document which visualises the data-flow between the modules of the DINAS-COAST model. On the basis of this graph the developers analyse the interactions between the modules and decide which changes are to be made in the next iteration of the development process.

Figure 3 summarises the work-flow of the development process. It also includes the database and the graphical user interface, which however are discussed elsewhere (see Vafeidis et al. (2003) and http://www.demis.nl/home/pages/products.htm). Knowledge about the modelled system enters the process via four categories: (i) the model's ontology; (ii) the modules, which express the functional relationships between the system properties; (iii) the data, expressing the actual state of the system and its possible futures in form of scenarios; (iv) the use-cases, which specify the end-user requirements. Those four categories are interrelated: new data may create the need to change existing algorithms or develop new ones with the consequent need to update the ontology. Once the knowledge has entered the development cycle most of the subsequent processes are automated. The development process can be iterated as many times as needed. At any stage a complete model is available. This approach allows for rapid-prototyping of new models and their incremental refinement until a satisfactory result is produced.

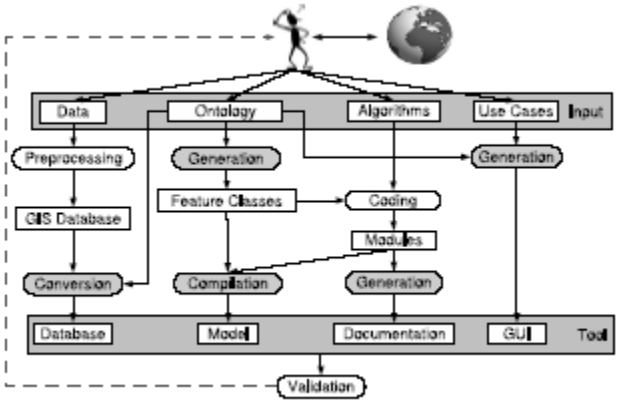

Fig. 3. The DIVA Development Process. Boxes denote deliverables, ovals denote processes, and shaded ovals denote automated processes.

5.3 The model

The integrated model consists of a generic kernel, a number of modules, and the feature classes. The modules and feature classes are problem specific and developed as described in the last section. All components, as well as the code generator, are completely implemented in Java and thus platform independent. The Build tool described in the next section includes some non Java components.

The kernel is responsible for data input, data output, and the time-loop. It dynamically creates the data structures according to the input data, sets the parameters, initialises the state variables, and reads the drivers. The kernel loads the modules at run-time and invokes them sequentially for each time-step. The modules' order of invocation is given in a configuration file. In the case of DINAS-COAST all modules operate on the same time-scale. The model, however, could be easily extended to support multiple time-scales.

Data input and output (I/O) is taken care of by two components: the feature classes and generic adapters. The feature classes handle the problem specific part of the I/O, that is when to read (write) which properties from (to) which data stream. This logic is specified by the ontology and then taken up by the code generator to produce methods for initialising the feature's parameters, reading its drivers, and outputting its state. For example, in Fig. 1 the feature country has two drivers: population and gdp. The code generator produces the method *readDrivers()*, which allows reading the drivers from a given data stream. This way model input and output is hard coded and efficient. While the generation of the feature code as described in the last section could be taken care of by any CASE (Computer-Aided Software Engineering) Tools, the generation of the input and output methods could not. The adapters handle the generic part of the I/O. They take care of reading and writing different file formats. Up to now, only one adapter for a self describing binary data format developed by Delft Hydraulics exists (see http://www.wldelft.nl/soft/tools/index.html). However, other formats could be easily implemented.

5.4 The build tool

A tool for building, testing, and documenting the model accompanies the development process. It takes the Java modules and the XML ontology as inputs and generates a website offering documents in various human- and computer-readably formats (HTML, XML, CSV and PDF). The documents include meta-information about the modules, the model, and the ontology, as well as documents used for the generation of the graphical user interface and input data files. Also included is a diagram that shows the data-flow through the system of modules (Fig. 2). The whole build and documentation process is fully automated: all documents are always consistent with the current model development status and available on the web.

The DIVA Build Tool is based on *Ant*, which is a platform independent build tool developed by the Apache Software Foundation (see http://ant.apache.org). It is written in Java, open-source, and easy to extended. The build processes and dependencies are specified in an XML language. A couple of other standard open-source tools were used and integrated via *Ant*: The graph visualisation tool *Graphviz* (http://www.research.att.com/sw/tools/graphviz) is deployed to visualise the data-flow between the modules. The XML parser *Xerces* and the XML processor *Xalan* (http://xml.apache.org) are used for XML parsing and processing. *Latex*, *Latex2html* and the postscript utilities *Ghostview* are used for the generation of the PDF and HTML documents. The DIVA

Build Tool is currently implemented on the *GNU/Linux* platform (http://www.gnu.org/gnu/the-gnu-project.html). However, since all the tools mentioned above are also available for a variety of other platforms, it could easily be ported.

6 Limitations

The flexibility of the iterative model development process comes at a price. The danger is that model development doesn't come to an end and not enough project time remains for model validation and application. Another drawback of this approach is that no complete specification of the mathematical problem needs to be formulated. This is common to all approaches which integrate models at the algorithmic or computational level. Unintended model dynamics can result from that and more efficient numerical solutions cannot be found.

While the performance of Java has increased significantly, there are still deficiencies compared to languages which are compiled to native binary code. Performance could have been increased by representing the data as arrays of simple types rather then arrays of (feature) classes. However, the primary goal was to make the interface for the module developers as intuitive as possible, rather than to optimise performance. Since all data for one time-step is kept in memory, the model's performance decreases significantly, if the data size exceeds the physical memory of the model's host computer.

7 Conclusions and outlook

The DIVA method is an innovative method for building modular integrated models by distributed partners. Unlike other integrated modelling frameworks it emphasises communication and the organisation of the development process. It provides scientists from different backgrounds with a way to harmonise their conceptualisations of the system to be modelled and an intuitive interface to express their knowledge about it. The process of model development is well defined and automatically documented. As a result, the status quo is constantly available on the web, providing a basis for efficient communication between project partners.

Within the project DINAS-COAST the DIVA method has been applied to build a tool for assessing the coastal vulnerability to sea-level rise. Meanwhile, application and improvement of both the DIVA tool and the DIVA method can go hand in hand. The global scientific and policy relevance of DIVA have already been recognised and collaboration on a range of initiatives is anticipated, including the EU ICZM (Integrated Coastal Zone Management) Strategy, and the new LOICZ (Land Ocean Interactions in the Coastal Zone) Science Plan. Improvements on the current DIVA tool could include a module for coral reefs and atolls, refining the adaptation module and increasing the spatial resolution of the analysis, thus increasing DIVA's usefulness to coastal management. In addition, it is conceivable to develop regional versions of the DIVA tool, such as a DIVA-Europe or a DIVA-India.

Edited by: P. Krause, S. Kralisch, and W. Flügel
Reviewed by: anonymous referees

References

Argent, R. M.: An overview of model integration for environmental applications – components, frameworks and semantics, Environmental Modelling & Software, 19, 3, 219–234, 2003.

Baarse, G.: Development of an operational tool for global vulnerability assessment, CZM Centre Publication, No. 3, 1995.

Dolk, D. R.: An introduction to model integration and integrated modeling environments, Decision Support Systems, 10, 3, 249–254, 1993.

Hoozemans, F., Marchand, M., and Pennekamp, H.: Sea Level Rise: A Global Vulnerability Assessment: Vulnerability Assessments for Population, Coastal Wetlands and Rice Production on a Global Scale, 2nd revised edition, Delft Hydraulics and Rijkswaterstaat, 1993.

Bruin, J. de, Vink, P. de, and van Wijk, J.: M – a visual simulation tool, Simulation in the Medical Sciences, The Society for Computer Simulation, San Diego, 181–186, 1996.

Leimbach, M. and Jaeger, C.: A modular approach to integrated assessment modelling, Environmental Modeling and Assessment, 9, 4, 207–220(14), 2005.

Muetzelfeldt, R.: Declarative modelling in ecological and environmental research, European Commission, Volume EUR 20918, 2004.

Rahman, J. M., Seaton, S. P., and Cudy, S. M.: Making frameworks more usable: using introspection and metadata to develop model processing tools, Environmental Modelling & Software, 19, 3, 275–284(10), 2003.

Vafeidis, A. T., Nicholls, R. J., and McFadden, L.: Developing a database for global vulnerability analysis of coastal zones: The dinas-coast project and the diva tool, Remote Sensing in Transition, 333–341, 2003.