Towards a Generic Tool for River Basin Management

IT framework report
Elbe River Basin
feasibility study - phase 4

The Elbe River Basin, Andrees Handatlas, 4.Auflage, Leipzig, 1904

BfG  USF
Institute of Environmental Systems Research
University of Osnabrück
Towards a Generic Tool for River-basin Management

Elbe-DSS feasibility study phase 4 —
IT framework report

Bernhard Hahn
Guy Engelen, Lic.

Research Institute for Knowledge Systems BV
P.O. Box 463
6200 AL Maastricht
The Netherlands

Tel.: +31 (43) 3883322
Fax: +31 (43) 3253155
WWW: www.riks.nl

Dr. Jürgen Berlekamp
Prof. Dr. Michael Matthies

Universität Osnabrück
Institut für Umwelt systemforschung
49069 Osnabrück
Germany

Tel.: +49 (541) 9692572
Fax: +49 (541) 9692599
WWW: www.usf.uni-osnabrueck.de

— Version 1.0 —

Bundesanstalt für Gewässerkunde
Koblenz
1 Contents

2 Summary

3 Introduction

3.1 Context of the IT framework report

3.2 Results from previous phases of the feasibility study

3.2.1 Problem Definition Phase

3.2.2 System description and inventory of models and data

3.2.3 Linking the three modules

3.3 Objectives of the IT framework report

4 Requirements Analysis

4.1 Requirements from previous phases of the feasibility study

4.1.1 Problem definition study

4.1.2 The qualitative system diagram

4.1.3 Recommendations for a pilot DSS

4.2 Types and roles of users of an IRBM-DSS

4.3 Scope and functions of an IRBM-DSS

4.3.1 Analysis

4.3.2 Communication

4.3.3 Library (knowledge base)

4.3.4 Management

4.3.5 Learning

4.4 Generality and flexibility

4.4.1 Reusability

4.4.2 Extensibility

4.4.3 The modeling gap

4.4.4 Technical requirements for generality

5 System Architecture

5.1 System architecture alternatives
5.2 Model integration ........................................................................................................ 30
  5.2.1 Approaches ........................................................................................................ 30
  5.2.2 Evaluation Criteria ............................................................................................ 30
  5.2.3 4 possible architectures ................................................................................... 34
5.3 Data integration ........................................................................................................ 38
5.4 Tool integration ........................................................................................................ 39
5.5 Component-based development ............................................................................. 41
5.6 Object-oriented application frameworks .................................................................. 43
  5.6.1 What is an (object-oriented) application framework? ........................................ 44
  5.6.2 What are the benefits of application frameworks? ............................................ 45
  5.6.3 Framework classifications .................................................................................. 45
  5.6.4 Geonamica® — a DSS Generator .................................................................... 46
  5.6.5 Other examples of enterprise application frameworks ..................................... 47
5.7 Integration of GIS functionality .............................................................................. 47
  5.7.1 Implementation of all needed GIS functionality in the DSS .............................. 47
  5.7.2 Implementation of the DSS as an application in a GIS framework .................... 48
  5.7.3 DSS architecture with separate GIS components layer .................................... 48
6  System Design ............................................................................................................. 50
  6.1 Spatial scales .......................................................................................................... 50
  6.1.1 Multiple scale cellular automata ........................................................................ 50
  6.2 Temporal scales ..................................................................................................... 53
  6.2.1 Simulation control ............................................................................................ 53
7  The DSS Development Process .................................................................................. 54
  7.1 Lifecycle planning ................................................................................................. 54
  7.2 Changing requirements – hitting a moving target ............................................... 54
  7.3 DSDM / SOMA ..................................................................................................... 55
8  Conclusions .................................................................................................................. 56
  8.1 10 critical management success factors for an IRBM-DSS pilot: ......................... 56
  8.2 10 critical technical success factors an IRBM-DSS pilot: .................................... 57
1  Abstract ...................................................................................................................... 60
2  Introduction ................................................................................................................ 60
3 Methodological Approach........................................................................................................... 61

3.1 The EFEDA project.................................................................................................................. 61

3.2 The ERMES Project.................................................................................................................. 62

3.3 The ModMED Project.............................................................................................................. 62

3.4 The ARCHAEOMEDES Project............................................................................................... 62

4 MODULUS: a Spatial Modeling Tool for Integrated Environmental Policy-making
63

4.1 Choice of the Pilot Regions...................................................................................................... 63

4.2 Policy makers............................................................................................................................ 64

4.3 Policy problems, policy levers, and policy indicators.............................................................. 64

5 ‘Research Models’ versus ‘Models to support Policy Making’ .............................................. 65

5.1 End-use integration................................................................................................................... 65

5.2 Scientific integration.................................................................................................................. 67

5.3 Technical Integration................................................................................................................. 68

6 The MODULUS model.............................................................................................................. 70

7 Using the MODULUS model and Decision Support System.................................................. 75

1 The GREATER software concept............................................................................................... 78
2 Summary

The IT framework report presents the results of phase 4 of the feasibility study ‘Towards a Generic Tool for River Basin Management’. The feasibility study has researched the possibilities for the development of an Integrated River Basin Management Decision Support System (IRBM-DSS), and gives recommendations for a pilot version of such a system.

While the reports of the previous phases of the study tried to find answers to the most important what questions, the primary concern of this report are the how questions.

Chapter 4 analyses functional and non-functional requirements for the IRBM-DSS; these were derived from the project meetings and the reports of the previous phases. The main functions of the DSS: Analysis, Communication, Library, Management and Learning are described together with typical use cases. Five groups of DSS users: Policy makers, Analysts, Scientists, Public organizations and Public are defined by their roles in the decision making process. Because of its far-reaching technical consequences, special attention is paid to the requirement that the tool should be ‘generic’.

Chapter 5 presents four alternative system architectures that differ in their approach to model integration. The alternatives are evaluated against a set of 15 criteria. It is recommended to build an IRBM-DSS based on a single integrated dynamic model with access to the original scientific models. This is the most expensive solution, but it fulfills best the user profiles and the intended uses of the DSS. Four classes of tools that need to be included in the DSS: Input, Output, Exploration and Evaluation are described and their relevance with respect to the main functions of the DSS is given. A summary of Component based development (CBD) and object-oriented application frameworks is given. These techniques form the technological basis for the Geonamica® DSS-generator, which was successfully applied in the development of several DSS with similar requirements as the Elbe-DSS project.

Chapter 6 focuses on a selection of the most challenging engineering problems for the Elbe project, foremost the proper handling of vastly different spatial and temporal scales of the various sub-models. Multiple scale cellular automata and a discrete event simulation controller are presented as possible solutions.

Chapter 7 gives an outline of the DSS development process. For the development of the Elbe-DSS pilot we recommend to adopt a development process that is suitable for rapid prototyping and incremental, iterative object-oriented development.
3 Introduction

Over the last decades river basin management has become increasingly complex. Increasing demands of society regarding use and protection of water bodies, new views and strategies towards (the making of) policy for river basin management call for a multidisciplinary approach for river basin management. Since methodologies and tools for such a multidisciplinary approach are not readily available, the Bundesanstalt für Gewässerkunde (BfG) has initiated the project ‘Towards a generic tool for river basin management’. The ultimate goal of the project is to develop a generic tool, which helps the water manager(s) to formulate policy for river basin management and to take appropriate measures to realize policy objectives.

3.1 Context of the IT framework report

In November 1999 a feasibility study has started to explore the possibilities for the development of this generic tool as a Decision Support System (DSS). A DSS can be defined as a computer based instrument developed to support the policy making process. In a DSS a structured approach towards river basin management is combined with eminent Information Technology, leading to an instrument that facilitates the processing, analysis and presentation of information. A DSS helps the end-user of the DSS to discern which information is relevant at any given time in the policy making process. With this information the end-user can enhance the quality of the different actions that are to be taken in the policy making process. On the one hand these are actions with respect to the contents of the policy like problem analysis, forecasting of future contexts, design and screening of alternatives, impact assessment and comparing and ranking alternatives. On the other hand these involve more process-like actions like communication, interactive policy-making, etc.

In order to focus the feasibility study, use will be made of the data and science for the river Elbe. The Elbe is one of the largest rivers in Central Europe, with a catchment of approximately 148,000 km², which stretches over Germany, Czech Republic, Austria and Poland (see chapter 2 for more details).

The feasibility study will result in recommendations for a pilot study focusing on some selected topics, which could be considered for implementation in a period of three years following the feasibility study. The feasibility study consists of five phases:

1. Problem definition
2. Qualitative system design (for the DSS)
3. Available data and science related to the system design
4. Informatics framework for the DSS
5. Covering note

This report focuses on phase 4, the informatics framework for the DSS. For the other phases separate reports have been written or will be written. The feasibility study is
carried out by order of the BfG by the University of Osnabrück, the Research Institute for Knowledge Systems, the University of Twente, and Infram.

3.2 Results from previous phases of the feasibility study

3.2.1 Problem Definition Phase

The problem definition phase formed the starting point for the design of a DSS. The purpose of this phase was to identify the so-called end-users of the system (i.e. persons or institutes who can be identified as problem-owner), make an inventory of relevant problems, determine the objectives to be achieved, identify tentative measures, and determine the spatial, temporal, economic and other boundaries of the system. In short: the problem definition delineates the scope of the study.

During the problem definition phase it was not possible to identify a single problem owner for the Elbe catchment. Instead a number of potential decision-making institutes, each having their own problems, objectives and measures, were identified. Although there is a wide spectrum of specific problems a number of problems mentioned by representatives of different problem owners could be worth including in the prototype model are:

- how to improve the socio-economic use of the river basin (shipping, tourism, fisheries, agriculture, etc.);
- how to provide a sustainable level of flood protection;
- how to reach a sustainable improvement of the physical, chemical and biological state of the Elbe and its tributaries;
- how to increase the ecological value of the river and the floodplains in the Elbe river basin.

A management objective is a desired state of the system that the decision makers want to achieve. The objectives are closely related to the problems. Achievement of the objectives is measured by means of (usually quantitative) criteria. For the design of the DSS the objectives are of particular importance as they determine which information the model should provide to its users. Referring to the identified problems the following main objectives can be discerned:

- improvement of socio-economic use:
- improvement of navigability of the Elbe river; maintain/improve agricultural yield; growth of tourism & recreation; improve conditions for fisheries;
- flood protection:
- reduction of the risk of flooding;
- improvement of physical, chemical and biological state of the Elbe and its tributaries and increase of the ecological value of the river and floodplains:
• river and ground water quality;
  soil quality of the riverbed and the floodplains;
  improve ecological functions of the catchment area.

• improve ecological functions of river and banks;

• improve ecological functions of flood plains;

One of the main functions of the DSS is to link the measures that can be implemented to solve the problems to the objectives. Suggestions for promising measures can be made by the end-users themselves, or the team of researchers designing the model. Although the measures are tentative their selection should be made with care. There is no point in analyzing measures that are too expensive or unacceptable for other reasons. Furthermore, one should be aware of the models and data that are needed to analyze the consequences of the selected measures. For the prototype model the following measures are selected:

• waste-water treatment;
• reduction of point- and non-point-source pollution;
• improving agricultural practice;
• groin modification and replacement;
• dike shifting, lowering of flood plains (space for the river);
• adding material to the river bed;
• reduction of buildings and other treasuries.

3.2.2 System description and inventory of models and data
The purpose of the system description and qualitative design is to provide a conceptual framework for the linkage of management measures to the objectives (see 3.2.1). A detailed schematization for the Elbe system is given in the general system description (Figure 2 of the Report concerning the ‘Qualitative system description & inventory of models and data’). The system description is based on the tentative results of the problem definition study. For practical reasons, and in view of the example function of the prototype, a selection must be made of key functions and processes, which are sufficient to demonstrate the added value of an integrated model. Although the diagram is designed for the Elbe system, the approach followed is generic and can be applied to river-basin management in general.

3.2.2.1 Modular structure of the DSS
One of the problems faced in the design of the prototype is that models and data are being collected at a variety of different spatial and temporal scales, ranging from 1000 km² sub-catchments for the landuse to 1 m grids for habitat modeling in floodplains. In the former case data are collected for the complete catchment, whereas in the latter case the study sites are limited to a few km². The main question is then how these different models and data can be incorporated in a common framework of analysis. Figure 3 of the Report ‘Qualitative system description & inventory of models and data’ shows how the problem could be approached in the prototype DSS.
A distinction is made between three levels of analysis, which exist in the research projects of the Elbe Ecology Program. At the highest level of analysis (gray box Catchment Module) we find the processes, which are being studied at the scale of the complete Elbe catchment of 150,000 km². Here we find the models describing the impact of landuse and hydrology on diffuse (nutrient) runoff as well as the impact of industrial or urban point discharges. At this scale level the time horizon is long (25-100 years), and the spatial and temporal resolution low (100-1000 km² and time steps of months or years).

At the second level of analysis (gray box labeled River Module) we find the models pertaining to the Elbe River of 700-800 km in length. This includes, for example, models describing the navigation condition, flood risk, and water quality. Although a variety of models can be used for these purposes a one-dimensional model would be more appropriate for the prototype DSS. For the river module the spatial and temporal detail will be in the order of 100 m-10 km, and weeks to years, depending on the type of processes studied (bed-level changes will require less temporal resolution than flood-level predictions).

At the third level of analysis (River Section Module) we find the most detailed models that describe the impacts of river engineering measures such as dike shifting and the habitat conditions for different species in the river, its banks, and the floodplains. At this scale the level of spatial and temporal detail will be in the order of 10-50 m. This module could be developed for a well-chosen example section of the Elbe river of 10-100 km, which would be representative for the Elbe river in general, and for which the data and models are available or can be collected in the time frame of the pilot study.

Preferably the three modules should be linked top-down, by selecting output variables of the higher-level modules that form input variables for the lower-level modules. For example, the total nitrogen load calculated for the catchment can be used as input for a 1D water quality module for the Elbe river. In turn the water levels or flooding frequencies calculated in the river module can be used in the ecological habitat models at the third level of analysis.

3.2.3 Linking the three modules
The three modules should not function independently, but must be linked in some way. In general integration can take place in two ways. Top-down integration is the more obvious approach and means that processes at a higher scale will influence the system at a lower scale. Sometimes the top-down integration will have consequences for the type of models needed. The incorporation of water quality in Module 2, for example, requires a pathway model for pollutant transport in Module 1.

A different approach is to integrate the three modules in a bottom-up way. For the Elbe DSS bottom-up integration could mean that local processes have an impact on a meso (river) and even macro (catchment) scale of analysis. The question however is, whether the processes included in the DSS for the Elbe (such as the habitat models) have a bottom-up influence.

At least these two top-down interactions between the modules can be discerned:
• pollutant load of Module 1 as input for water quality model in Module 2;
• discharge, water quality, and water level calculated in Module 2 as input for the habitat models in Module 3.

3.3 Objectives of the IT framework report
The first objective of the IT framework report is to analyze the technical feasibility and requirements for an Integral River Basin Management Decision Support System (IRBM-DSS). Based on the results of the previous phases of the feasibility study the IT framework report shows the ways to realize the requirements from those phases. This includes the scope and functions the DSS will have.

The second and main objective of the report is to propose a possible system architecture of IRBM-DSS. To achieve this, concepts for model integration as well as for data integration existing software-technical approaches (e.g. component based development) are evaluated concerning the usability for the IRBM-DSS. Selected aspects of technical system design (e.g. working with temporal and spatial scales) are discussed.

Finally the appropriate software development process is described. This includes the estimation of risks by developing a DSS system.

4 Requirements Analysis

4.1 Requirements from previous phases of the feasibility study
4.1.1 Problem definition study
As one result of the problem definition phase a set of possible end-users are identified. Together with the results of the end-user meetings held in Koblenz (Feb.) and Osnabrück (May) the following end-users for the pilot-DSS could be identified:

1. BfG
2. ARGE
3. IKSE
4. Länder

The problem definition study also showed that the possible functions of a DSS-system for Integrated Water Basin Management could be ordered regarding their significance for the selected end-users: the most important function are the analysis capabilities of the DSS. An also very important function is communication between end-users as well as with the public. Another significant function will be management capabilities. Especially for the BfG this will include knowledge management, e.g. to see what domains the different available models cover and if there are blank spots. Finally database functions and the function of the DSS as a learning tool could be identified.

The problem definition study emphasized the importance of handling space and time in the DSS. Time scales and spatial scales mainly depend on which decision processes have to be supported by the DSS. Concerning the spatial scales three different scales
(catchment area, river and flood plains, river sections) could be identified. The corresponding time resolutions on which the problem relevant processes are treated will depend on the available models. Therefore a more detailed analysis has to be carried out when the results of the feasibility study are all available (at the beginning of the pilot DSS).

In general it is clear that the large amount of information generated by an integral DSS calls for ways to aggregate, prioritize and present (e.g. maps) information to support the decision-maker to determine priorities and help to make a choice between feasible alternatives. Accordingly this requires appropriate methods and tools. Most common in this field are (spatial) multi-criteria evaluation tools and methods. The problem definition report raised the question whether such decision-making models should be part of the DSS, because in general it calls for a lot of discussion and is relatively little used. It is therefore recommended to determine the need for these models in the first stage of the development of the pilot DSS. Nevertheless with regard to the IT-framework this option of additional tools has to be taken into consideration.

4.1.2 The qualitative system diagram
The qualitative system description proposed a modular structure of the DSS in order to overcome the problem of scale differences. The system diagram is composed of three modules (catchment, river, river section) that represent the three levels of analysis. The relevant processes of each module that have to be modeled in the DSS have been associated with existing models and data available from Elbe Ecology or Elbe2000. The inventory of models and data showed that the situation is very heterogeneous and ranges from already (more or less) usable data and models at the appropriate spatial-temporal scale to those processes for which models and data are not immediately available. Between these extremes in most cases the relevant models and data have to be adapted in some way. In general this is important for the system architecture of the DSS. The necessity of adaptation of existing models for integration into the DSS implies specific techniques for model integration as described in Chapter 5.1. Indeed it may be necessary and reasonable to re-implement a model if this is the most effective way of integration.

For conceptual reasons the three modules should not function independently, but must be linked in some way. Integration can take place in two ways: top-down integration or bottom-up integration. Independently from the selected way of integration it needs techniques for coupling of existing models and data. Due to the fact that existing models of very different types have to be coupled together with probably new or re-implemented models this will be more than the matter of defining capable data interfaces. In fact it requires a methodology of flexible and modular model integration as described in Chapter 5.3.

Questions of water quality will be the basis for linking the results of the Elbe 2000 program with those of Elbe Ecology in the DSS as the Qualitative System Description Report pointed out. Water quality data were collected during the Elbe 2000 program in large volumes as well as they are routinely collected by the ARGE Elbe until now. While the data from the ARGE Elbe are already available via the Internet a more sophisticated and GIS-based system (ELBIS) that integrates the ARGE data with the Elbe 2000 data will start in the second half of 2000. If water quality will be one of the objectives of the DSS it will be necessary to use actual data from these databases in
the DSS, e.g. for parameterization of models or validation of modeling results. Due to
the fact that relevant data are administered at different sites it is suggested to use IT-
techniques for integration of heterogeneous and shared data bases as described later
on in Chapter 5.3.

4.1.3 Recommendations for a pilot DSS
As pointed out in the Appendix of the Qualitative System Description Report it is
recommended that the design of a pilot DSS should focus on one or more of the
following four management objectives:

- Water quality
- Flood risk
- Navigability
- Floodplain ecology

From this selection Water quality is the most important objective against the
background of the new EC water framework directive and will be the focus of most
decision makers in water basin management.

At the end three alternatives for the DSS are possible, where the choice will depend
on interests of the decision makers, on financial and personal resources as well as on
time to be invested. The three alternatives are:

1. A DSS based on a selection of models taken mainly from Elbe
   Ecology and related sources and using Elbe 2000 data for water
   quality questions
2. A limited design using only simple models available at BfG
3. A more sophisticated prototype using advanced models

The main functions of the pilot DSS will be:

- analysis in case of comparison of management alternatives
- communication
- library function

The general recommendation is to start with option 1 and go back to option 2 if there
are problems. This will lead to a combination of models from Elbe Ecology and BfG.
In the long term the model may be revised to accommodate models of option 3.

Taking the above points into consideration and with the focus on the water quality
alternative based on selected models from the Elbe Ecology the following technical
issues have to be taken into account from the information technology point of view:

- the existing models that can be used from Elbe Ecology are very
different concerning their internal complexity and their spatial and
temporal scale. Especially the models for calculating the diffuse
inputs into the river system spread from mechanistical models focusing on very local processes to models that calculate impacts on a very large scale. Here it is necessary to adapt the scales and the model complexity. It is highly recommended to avoid too complex models in order to achieve a fast and robust DSS.

- As the evaluation of existing models showed, for most of the models it seems to be necessary to use advanced techniques of model integration (see 5.5 and 5.6) and sometimes a complete re-implementation may be necessary.

- The DSS will have to deal with either raster data as well as vector data, the integrated models will need both types of geodata.

- The prototype DSS should be an open system in the case of data integration what means that it should be realised to use and get access to existing databases. This includes monitoring data mainly from Elbe 2000 and the ARGE Elbe, that provide their data via Internet as well as hydrological data provided by the BfG.

### 4.2 Types and roles of users of a IRBM-DSS

The possibly heterogeneous group of users of an IRBM-DSS can be differentiated in two ways at least: first of all, the role of the users and their function in the decision making process is important. Secondly from a more practical point of view the users experience and fitness of working with computer-based tools is of particular importance especially in respect to the IT-framework. Regarding the different roles of users the following grouping/arrangement can be made:

1. Decision makers and water managers (e.g. federal offices, WSV): usage of the DSS for analysis and balancing of possible alternatives.

2. Officials in charge (Sachbearbeiter) in planning processes (water authorities and agencies, water associations): working with the DSS for putting the objectives from general planning into practice by defining and selecting real measures.

3. Scientific experienced users for water-relevant sub-tasks (researchers, BfG, ARGE/GKSS): more or less permanent work with the DSS for varying tasks.

4. Associations, public organizations, stakeholders (e.g. nature conservation, fishery, industry, agriculture and other): usage of the DSS to find positions (Meinungsbildung) and discussion with other groups.

5. Public: information about possible alternatives

6. Concerning the fitness of the users with IT-tools no definite assignment of the above mentioned groups can be made. Although a more or less general correlation of the users role with his computer knowledge can be done (e.g. scientific users may be
already more familiar with GIS or modeling than the public) the real IT-competence is mainly dominated by individual abilities.

4.3 Scope and functions of a IRBM-DSS
The scope and functions of an IRBM-DSS mainly depend on the demand of information of the end-users. As they use information from the DSS in their decision making processes the DSS should satisfy the demands of the end-users as far as possible. But there may be limitations from lacks of knowledge and data about the processes that relate measures to the objectives.

The following points describe the main functions the DSS will have. In addition selected typical use cases are shown which include the type of user and its role in the river basin management process.

4.3.1 Analysis
A complex integral model provides a holistic representation of the system, by explicitly defining the linkages between the natural system and the socio-economic system. In general analysis functions require a description of the system at the appropriate levels of spatial detail and temporal scales. In practice analysis skills are one of the most important functions of the DSS. Analysis capabilities will be important at different points/levels of the DSS system. First of all the user may want to analyze the current state of the river basin. This includes an inventory of all relevant functions of the river basin (e.g. socio-economical and ecological functions) for the desired decision process. Secondly analysis functions are necessary for evaluation of the effects and impacts of the measures on the river basin. The calculated projected state of the river basin has to be compared to the current state as well as to the desired state. By comparing projected and desired state the user will be able to decide if further measures are necessary to reach the desired state.

Typical use case:
The International Commission for the Protection of the Elbe river (IKSE) wants to improve the flood protection strategy of the whole Elbe catchment. As another user the relevant state authorities are also interested in this objective too, because they are the top authorities for the implementation of the measures. Due to this problem the analysis functions will be used to estimate to current state of the flooding in the catchment, e.g. by analysis of flooding risk frequencies. Here the flooding risk may not be the only relevant question itself. Beyond this, it may be necessary to analyze what socio-economic functions are affected by flooding and where the hot spots of economic damages are. Hence measures for flood protection will be focused on areas with the highest expected damages. The analysis tools of the DSS for instance can estimate the effects of such measures like dike shifting. At the end an advanced analysis may compare the amount of money investigated in the measures with the economic loss when flooding occurs.

4.3.2 Communication
An integral IRBM-DSS can facilitate communication between policy makers and stakeholders in participative planning efforts. Interactive simulation of the integral model shows the stakeholders how their different views on the system are related to each other. Transparency of the system guarantees that the stakeholders recognize
their domain explicitly represented in the system. Transparency and user friendliness thus are the key factors for the system to function as a mediating device in a collaborative planning context. Part of the user friendliness is the responsiveness and speed of the system, which are particular important in brainstorm-like sessions, where one wants to explore different scenarios during a discussion.

As several existing systems already showed (e.g. WadBOS, EnvironmentExplorer) communication is one of the most important functions of a DSS. Because in integrated water basin management heterogeneous parties and actors are involved communication is at least the most central part of the DSS. Beyond this type of communication information to the public is a possible function of the DSS too.

Typical use case:

Groyne modification is an important issue at the moment and is related to ecology, shipping and fisheries. Therefore communication between nature organizations, WSD, BAW and BfG about the various effects of groyne modification is necessary.

To improve the river water quality by reduction of pollutant impacts from non-point and point sources several measures are possible where different groups are involved like the industry (production of chemicals), agriculture (impacts via land-use practice and usage of chemicals) and local authorities or water agencies (waste water treatment). To enable discussions about the effects of possible measures between these involved groups the communication function of the DSS is important. For instance the DSS can be used to estimate the effects of improved wastewater treatment to show the importance of this measure. Otherwise the impacts of land-use changes or agricultural practice may be demonstrated by the DSS. This can stimulate exchange of arguments concerning the estimated effects and efficiency of measures towards a generally accepted strategy. Thereby the DSS acts as an instrument that is accepted by all of the above named parties.

Tools:

- storage, import and export of complete scenarios for exchange between stakeholders or lobbies;
- reporting tools to save scenarios together with comments and arguments;
- tools to geo-reference remarks and create commented maps.

4.3.3 Library (knowledge base)
An IRBM-DSS based on an integral model can serve as a knowledge management infrastructure. It gathers, orders and links existing knowledge about a system and therefore can fulfill the function of a dynamic library. It may reveal knowledge gaps, and thereby give impetus to further research and data collection. Through the IRBM-DSS knowledge about a system becomes available in operational form. The IRBM-DSS can be a common infrastructure for storage and transfer of the knowledge for participating organizations and possibly the general public.

As already shown in the qualitative system description integrated river basin management needs to incorporate a wide range of knowledge and data. In particular
for an IRBM-DSS of the Elbe this include data, models and knowledge from Elbe Ecology, Elbe 2000 and related research projects. For this reason library functions of the DSS will be helpful - not for all but for some users for instance for federal agencies. The library functions help the users to structure the current knowledge and available data of the river basin.

Typical use case:

The BfG may use the library function of the DSS to get an overview of present models for a special process on a definite scale. As a result the library function of the DSS gives a selection of current available models at the given scale.

Tools:

- Masks and/or wizards for search/selection in the DSS model and data database
- Map based tools for geographically referenced selection of available models and data

4.3.4 Management
Management is a function of the DSS that is important for the users that have to evaluate general decisions and turn them into realizable measures. From the set of possible measures they have to select those that fit best to the objectives. Of course financial aspects must be taken into consideration and therefore have to be evaluated.

Typical use case:

The Länder may use the management function of the DSS to evaluate and handle different possible alternatives of measures. For example in the case of flood protection alternatives may be building of retention areas by dike shifting or raising the dike levels. What are the costs of these measures and what effects will these measures have on the river and floodplain ecology, the landuse near the river, the shipping and fishery?

Tools:

- multi-criteria evaluation tools

4.3.5 Learning
A DSS cannot only be used for analysis, communication or other of the already above mentioned functions but also for learning purposes. Primarily this means learning about the linkage of processes, natural and user functions, which build a complex network of the system with multiple interdependencies. Even if experts are familiar with the dependencies in their special field of interest they may use the DSS for learning about the linkage to unknown functions.

Typical use case:

A water manager in general knows the effects of river deepening or dike shifting on flooding risks that can be calculated with hydraulic models. Normally he will not know how these measures affect the ecology of the river and the floodplains. To get a
better understanding here he can use the DSS to learn about the interactions of river morphology, groynes, dike shifting and for instance ecological functionalities, habitat conditions and species compositions.

Tools:

- Tools to show the dependencies of processes and functions based on the systems description
- Identification tools to highlight the qualitative impacts of interventions in the systems diagram

4.4 Generality and flexibility

This study’s objective is to evaluate the feasibility of the development of a generic tool for decision support in integral river basin management.

Before heading into requirements engineering and systems analysis for the IRBM-DSS, we think it is essential to define more precisely what it means to state that the system should be ‘generic’. Such a definition is needed because generality and flexibility may be built into software systems in a number of different ways and at various degrees.

Choices in this area not only have a huge impact on the development effort for the IRBM-DSS, but also affect the entire product lifecycle and even can make a difference in what is perceived as ‘the product’. In this sense generality and flexibility could be seen as meta-requirements, because they define conditions for the development of other functional and non-functional requirements of the system.

Within the context of this feasibility study and the development of a pilot version of the IRBM-DSS, our discussion of generality and flexibility issues is related to the following questions:

1. What do we want to demonstrate with the pilot?
2. How should we allocate our resources to various development activities?
3. What is an appropriate development process?

Naturally the pilot study is more biased towards demonstrating what is usable and technologically possible for selected problem areas of river basin management systems, rather than working out a flexible but abstract architecture that could form the basis for future DSS applications in this domain.

However, from projects like WadBOS and EnvironmentExplorer, which have survived the prototype phase of their lifecycle and are now in use for real world policy support, we have learned how important it is to put effort in a flexible and generic architecture as early as possible.

Despite better intentions in the beginning, prototypes are rarely thrown away. Once users get enthusiast about the system, the distinction between ‘prototype’ and ‘product’ starts to vanish and the prototype/product is incrementally improved. It is
then the flexible and generic architecture that makes it possible to deal with the
frequent requests to add, change or replace models, indicators or other tools in the
system while maintaining high quality and keeping the costs for this continuous
change under control.

Thus depending on the answers to the above questions, we should define the
dimensions as well as the level of generality at which we want to aim with our system
architecture. With such a definition at hand, we are better equipped to design a system
framework, which provides useful spots for variability and extension.

In the following paragraphs we will discuss the various options open to us to build
generality and flexibility into our system, the economics of generality in terms of
costs and benefits, as well as the relation between various kinds of system generality
and user roles. In particular reusability and extensibility, two important aspects of
generic systems, will be discussed.

4.4.1 Reusability
With respect to generality the problem definition report mentions one important
requirement:

... an IRBM-DSS developed for the Elbe should be applicable to other large river
systems as well.

Obviously this requirement is concerned with reusability. As we will soon see,
achieving reusability is an ambitious goal for an IRBM-DSS, both from a scientific
(with respect to the models) and a software engineering point of view. The remainder
of this paragraph will focus solely on the software engineering aspects of reusability
within the context of the development of an IRBM-DSS. This does not mean that the
scientific aspects of (model) reusability are less important or less difficult to deal
with. These problems however are dealt with in a dedicated section of this feasibility
report.

In software engineering reusability neither happens automatically nor is choosing an
object-oriented development method sufficient to obtain reusable system components.
Reusability does not come ‘for free’. It requires a deeper conceptual and technical
analysis and additional software engineering effort, compared to what is necessary for
a perfectly working one-time solution for the same problem. Therefore developing for
reusability should be seen as a long-term investment and, as with any investment, it is
a calculated risk. The risk factor is, to weigh the chances that the system or some of its
components actually will be reused against the extra effort that is required to create a
more generic reusable solution.

In order to take into account the reusability requirement for the IRBM-DSS systems
architecture, it is helpful to formulate and answer the following questions:

1. What are the reusable artifacts?
2. Who is reusing what?
3. When does reuse take place?
4.4.1.1 What are the reusable artifacts?

Here follows a list of possible reusable artifacts that will be generated during the development phase of the IRBM-DSS (adapted from [D’Souza 99]):

- System as a whole (one large executable object: black-box component)
- Scientific models with or without software implementation
- Compiled code, executable objects (black-box component)
- Source code; classes, methods (white-box component)
- Designs and models: collaborations, frameworks, patterns
- Plans, strategies, algorithms and architectural rules
- Interface specifications
- User interfaces

Note that nearly all artifacts being produced in the system development process have the potential to become reusable assets. Which ones are most appropriate is not only a technical choice, but also depends on organizational and economic aspects of the project.

In [D’Souza 99] some helpful rules are given for making a rational choice:

**Reuse Law 1**

Don’t reuse implementation code unless you intend to reuse the specification as well. Otherwise, a revised version of the implementation will break your code.

**Reuse Lemmas**

(1) If you reuse a specification, try a component based approach: Implement against the interface and defer binding to the implementation.

(2) Reuse of specifications leads to reuse of implementations. In particular whenever you implement standardized interfaces, whether domain-specific or for infrastructure services, you enable the reuse of all other implementations that follow those standards.

(3) Successful reuse needs decent specifications.

(4) If you can componentize your problem domain descriptions themselves and reuse domain models, you greatly enhance your position to reuse interface specifications and implementations downstream.

Reusing more abstract artifacts like interfaces, specifications, (collaboration) patterns and frameworks yields the largest benefit, but these artifacts are also the most difficult to produce from scratch. Generally, the easier way is to evolve from the concrete towards the abstract:

Suppose we are working with an object-oriented development environment and have represented our relevant domain concepts as classes and objects. We try to identify common behavior in the classes
and factor it out into a set of *base classes*, which gives developers the benefit of code reuse by inheritance. Often we will discover, that many typical operations and interactions between these base classes can be specified without providing any particular implementation. Such a specification is called an interface. Interfaces bring us one step further towards reusability, because they can be defined independently of any class hierarchy. Base classes, interfaces and collaboration patterns together form the basis for domain specific object-oriented application frameworks (see section 5.6). Application frameworks offer huge reuse potential and can be seen as major asset within a knowledge management infrastructure for a given domain.

### 4.4.1.2 Who is reusing what?

The list of potential reusable artifacts given in 4.4.1.1 contains items of various levels of abstraction, with some of them being more accessible to software engineers, while others being of greater interest for scientists, domain experts or end-users. The choice of what should be reused is therefore related to who is reusing something.

Since developing for reuse is an investment, this raises the question, which participants in the development of an IRBM-DSS do want to commit them to this investment. Who owns, administers, and takes the responsibility for further development of the resulting library of reusable IRBM-DSS components? In many cases this is the organization or firm that develops the software, on the other hand it also could be the end-user organization or a consortium of end-user organizations and software firms.

<table>
<thead>
<tr>
<th></th>
<th>Software engineer</th>
<th>Modeler</th>
<th>Domain expert / Analyst</th>
<th>Policy Maker / End user</th>
</tr>
</thead>
<tbody>
<tr>
<td>System as a whole</td>
<td>++</td>
<td>O</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>Scientific models</td>
<td>O</td>
<td>++</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Compiled code</td>
<td>++</td>
<td>++</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Source code</td>
<td>++</td>
<td>O</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>Designs and models</td>
<td>++</td>
<td>++</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Plans, strategies</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>O</td>
</tr>
</tbody>
</table>
Table 1 gives an overview of the reuse potential of various artifacts with respect to several classes of users. As reuse lemmas (3) and (4) state, the most benefit is generated from reuse as far upstream as possible in the development process, because this way chances for reuse further downstream are maximized. The question about who might want to invest in the development of a library of reusable IRBM-DSS components can be reformulated as the question whether the development of an IRBM-DSS (prototype) should be technology driven or user driven.

The authors experience from similar development projects is that policy support DSS projects tend to start technology driven and then, if successfully applied to real policy problems, tend to evolve towards a more user centered development process. The maturing process of the DSS goes along with a transfer of responsibilities from the developers to the user community. This pattern can be used as a guide when we have to decide how much and which type of reusability we want to build into the (pilot) system.

4.4.1.3 When does reuse take place?

In reuse oriented DSS development, two parallel and interrelated activities take place:

- **Product development:**
  Within product development we can further distinguish (1) **user product development** and (2) **developer product development**.

  **User product development** deals with the design and implementation of specific DSS applications. With a library of reusable assets already existing, this is where we harvest from our earlier work, by assembling new applications from generic frameworks and components.

  **Developer product development** deals with the design and implementation of the generic framework. Geonamica® (see paragraph 5.6.4) together with its software development kit (SDK) is an example of such a developer product.

- **Library development:**
  Library development creates reusable components by generalizing, certifying and enhancing artifacts that were developed in product development.

Product and library development are coordinated processes. A full description of the techniques available that allow managing and coordinating these processes in a
collaborative effort of several participating organizations is beyond the scope of this report. The history of similar projects has shown, that forming an organization or a consortium in which developers as well as end-users are represented seems to be a good approach. The benefits of early and active end user involvement in both development processes cannot be overestimated.

In the specific context of the IRBM-DSS, our advice concerning reusability is to be rather modest on this criterion, certainly in the early development phases of the system. Choices relative to the architecture and system design should maximize the level of reusability, but it should not become the main thrust of the project (which it easily could). The first versions of the system will need to be evaluated on their ‘usability’, rather than on their ‘re-usability’. Given the fact that it is not as yet very clear who will be the main end-user of the system, what his technical skills will be, how he will use the system, and what the main theme of the system will be, reusability should be maximized at the level of the Software engineer and the modeler first. They will benefit most and always, even from the most concrete artifacts discussed.

Our discussion of reusability initiated from the requirement that an IRBM-DSS developed for the Elbe should be applicable to other large river systems as well. In our view the best way to achieve this kind of system level reusability is to use/develop an object-oriented application framework (see section 5.6) for model based spatial DSS in the domain of integral river basin management. Since the development of such a framework is a huge investment, we suggest evaluating whether already existing (commercial) application frameworks for integral spatial DSS (e.g. Geonamica® see section 5.6.4) meet our requirements and can be used as a basis for the development of IRBM-DSS. Such an evaluation could be part of the requirements engineering phase of the pilot study.

4.4.2 Extensibility

For IRBM-DSS extensibility is a requirement, the importance of which can hardly be overestimated. This type of application exists in a continuously evolving environment of user requirements. As stated earlier, experience with similar systems has shown that from the moment an organization starts to use the system for real policy support, users issue frequent requests to add or change functionality. This is mostly so for policy support systems, because the policy context within which these systems are used changes rapidly, continuously and in ways that cannot be anticipated from the beginning of the development process. For the IRBM-DSS this is not different. It is not possible to know in advance all the problems an analyst or policy maker will want to explore in a field as broad and complex as integral river basin management. This holds even more if the system will be used to support participatory policy design.

During the DSS life cycle, system level extensibility makes cost-effective handling of user requests for new or altered functionality possible. During DSS development component level extensibility is required when we want to reuse an already existing component, that implements only part of what we need. As long as there is scientific progress, the models that form the simulation kernel of the system will evolve, thus the models themselves should be extendable and / or exchangeable parts.

Extensibility can be achieved with state of the art development techniques like component based development (CBD) and object-oriented application frameworks.
The technical aspects of these techniques are described in more detail in section 5.5 and 5.6.

4.4.3 The modeling gap

Model based software tools for decision support in planning and policy-making always face the problem of the modeling gap. The modeling gap is the distance between the concepts the end-user uses to describe the problem at hand and the concepts directly expressed by the software tools he uses. Since the modeling gap is closely related to the abstraction level of the various DSS tools we discuss it here under the aspect of the ‘generality’ requirement of the system.

Increasing the generality of the tool in most cases also widens the modeling gap and therefore might decrease the usability of the tool for users that work at a high level of abstraction (e.g. policy makers). Rather than trying to develop a single tool that bridges the whole modeling gap and suits every class of user, we believe that it is better to develop generic as well as specific tools at different levels of abstraction, servicing different roles and tasks in the process of DSS development, use and maintenance. Together these tools then form a generic DSS framework for integral river basin management. The modeling gap and the various roles and tasks of end-users are critical factors to consider when conceiving a ‘generic’ framework architecture for the tool.

Table 2 gives an overview of several classes of users, their roles in a decision process supported by integral models, and the software tools that correspond to the abstraction level of their activity.

| User              | Role                                                          | Tools                                                                 |
|-------------------|---------------------------------------------------------------|                                                                      |
| Policy Maker      | Uses interactive graphical DSS like WadBOS or Environment Explorer for policy exercises | Interactive graphical IRBM-DSS                                      |
| Domain expert / Analyst | Provides domain knowledge in natural language, heuristics, rules, databases, mathematical formulas… | Visual IRBM-DSS shell, with customizable algorithms, rules and run-time model configuration |
| Modeler           | Defines sub-models in mathematical formulas, codes model building blocks in C++ or other general purpose programming language | Model Building Blocks (MBBs), COM interfaces, C++ class library,    |
| Software engineer | Builds integral model from model building blocks and implements graphical interactive DSS. Defines | General purpose object-oriented programming language (C++),         |
Table 2: A role-oriented classification of IRBM-DSS users with the corresponding tools

<table>
<thead>
<tr>
<th>Role</th>
<th>Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modeler</td>
<td>Core components, COM interfaces and application and framework classes</td>
</tr>
<tr>
<td>Domain expert</td>
<td>Models based on pluggable components</td>
</tr>
<tr>
<td>Policy maker</td>
<td>Interactive IRBM-DSS</td>
</tr>
</tbody>
</table>

Note that in Table 2 the tasks at the highest level of abstraction (policy maker) require the most specific, thus least general tools and vice versa. Also note the tool reuse hierarchy in the table. On every level the corresponding tools are used to build artifacts that serve as tools on the next higher level.

The modeler uses model components, COM interfaces and application and framework classes to build complete (sub) models. The domain expert / analyst may use these models in a visual development environment to develop or maintain a specific DSS application, foremost by assembling and configuring prefabricated components. Finally the policy maker uses the interactive IRBM-DSS.

Of course in real-life projects the neatly separated roles shown in Table 2 will overlap most of the time and often individuals will be active in more than one role. However the general approach to provide a hierarchy of reusable artifacts on various levels of abstraction, corresponding to different human roles in the decision making process, results in a long-term component-based development strategy for a more flexible and general IRBM-DSS framework, that can be customized for various projects.

To our knowledge, there exist at this moment, and in the domain of integrated watershed management, no tools to service the Domain expert or Policy maker as described in Table 2. During the development of the IRBM-DSS we advice to take on board the general concern to service each of the users of the DSS with the appropriate tools, however, to also be realistic as to the feasibility of this aim and concentrate on the needs of the Modeler and Domain expert first and most.

4.4.4 Technical requirements for generality
What does generality mean for the technical architecture of the IRBM-DSS? As we will see, it sets a rather ambitious goal, which is only achievable by fulfilling, at least partly, a set of more modest generality requirements:

1. **Data**
   First of all, to be applicable to more than one area, an IRBM-DSS must be generic with respect to data. It must be capable of loading and saving maps in various formats. Measurement data perhaps even connections to real-time monitoring data feeds, as well as different policy scenarios should be editable at run-time or at least not require a recompilation of the application.

2. **Models**
   When applied to a new river basin, it is very likely, that some if not most components of the integral model as well as its spatial and temporal resolution will be different. Therefore the architecture of the integral model should be modular and based on *model building blocks* that function as pluggable components. Just like chips on an electronic circuit board, alternative blocks may replace model...
building blocks, as long as they adhere to the same interface specifications.

3. **Interfaces**

Interfaces between components of a system often are the most stable parts of a system. However flexible interfaces allow for much easier integration of already existing models, where changing the interface is often not an option.

Maximum flexibility can be achieved with a component architecture that can discover interfaces at run-time, however a considerable price in terms of efficiency has to be paid to achieve this level of flexibility.

4. **Functions**

Spatial DSS based on integral models may be used for very different tasks:

- Analysis of policy options
- Exploration and learning about the dynamics of a complex system
- Management of a complex environment
- The DSS may serve as a library knowledge management tool
- The DSS may serve as a communication medium, which is especially useful in participatory approaches to policy design
- For every task the DSS provides specialized tools. When applying the DSS to a new region, it is likely that the relative importance of the different tasks as well as the specifications of the tools needed to fulfill them is subject to change. Therefore, what was said about the models in a generic IRBM-DSS also holds for the tools, they should be interchangeable parts of the system.

5. **Context**

Last but not least, the context in which an IRBM-DSS is used is likely to change when it is applied to a new region. Things that may be different include: methods of distribution, operating systems and computer platforms, maintenance policies, organizational aspects, and most important, the role of the end user may be defined very differently.

Each of the items in the above requirements list corresponds to a level of generality to the system. Together they span a scale of generality and we should decide about the appropriate point for our product on that scale. To that effect, we believe it is useful to describe briefly the products at both extremes of the scale.

On the least generic end of the scale, we find a DSS running on fixed data and based on a fixed model, with static interfaces, or even a monolithic model with no internal interfaces at all, a predefined set of functions and tools that cannot be extended. Of
course, this system would only function well in its predefined context. One might think that such a DSS would be useless, or at least be a waste of investment, but actually to our best knowledge, quite a number of (successful) DSS fall within this category.

On the opposite end of the generality scale, we can define a product that is not a DSS in itself, rather a software component framework and development environment for the domain of integral river basin management. Such a product then enables a quick implementation of an IRBM-DSS as part of diverse projects.

When moving on the scale towards more generality, the granular size of reusable components decreases, while the effort needed to implement a concrete solution increases.

Because the level of generality has considerable impact on the software development effort, we have to decide how much generality we want to put into the system architecture for the IRBM-DSS in the long and the short run (the pilot study). In particular for the pilot study we might want to give up some generality in order to get a more complete feature set. Further (financial) support of the project to a large extend will depend on how convincing the prototype is, while generality only pays off in the long run, when more systems of the same type are implemented. To come up with useful generalizations, we must ask ourselves, or better, ask our DSS-users, which aspects of the system they expect to change and which they believe to be rather stable. Unfortunately, these questions are notoriously difficult to answer, especially in the relatively new domain of integral modeling, where few examples of systems with a complete lifecycle exist.
5 System Architecture

5.1 System architecture alternatives
Decision Support Systems (parts of this text are adapted from [Engelen et al., 1993] and [Engelen, 2000b]) are principally made up of four components (See Figure 5-1): (1) a user interface enabling easy interaction between the user and the system, (2) a data base containing the raw and processed data of the domain and the area at study; (3) a model base with relevant models of the decision domain, and (4) a tool base with the methods, analytical techniques, and software instruments required to work in an effective manner with the domain models and the data.

All four components have a complex internal structure, but for the moment we will focus our discussion on their role and internal interaction. Note that the four-component view does not necessarily match with the user’s perception of the DSS, because the user interface layer may present things in a way that differs from the internal structure of the system.

The user interface:
Decision-support systems are intended for use by high-level decision makers to solve ill-structured problems. Although often specialized in their domain, these decision- and policy-makers may be unfamiliar with information science and technology. The user interface, the vehicle of interaction between the user and the computer, takes into account differences in the cognitive styles and relative knowledge of the users. It is designed to hide the complexities of the internal computer system without hampering its flexibility and provides insight into the structure of the mathematical models, methods, variables, parameters and processes, the underlying theoretical assumptions, the boundary conditions and other constraints. It allows the user to address the different components of the DSS (tools, data, models, etc.), translates the user input into appropriate computer instructions, and reports back the results of the computations. To provide maximal user-friendliness, state-of-the art interactive graphic techniques are being applied extensively.
Thus, the most important, but also the most difficult task of the user interface is to hide the full technical complexity of the system and at the same time to provide structured access to the system. In WadBOS [Huizing 98] this was achieved by introducing several user access levels (modes) corresponding to the various user profiles that were described in section 4.4.3 as well as an alternative front-end tailored for policy-makers (the so-called Policy Wizard).

**The database:**
A thorough database serves as an input medium for the data required in the models used. It is filled with information that is appropriate to the management or policy issues dealt with in the DSS. There is a growing trend to store spatial, social and ecological data in GIS databases. Consequently, a good interface linking GIS and the DSS is of utmost importance for policy-making and planning. Particular solutions will be discussed in section 5.3.

**The tool base:**
The convenience, richness and scope of a DSS is primarily determined by the spectrum of tools and models available from the tool base and the model base. Typically decision methods, statistical and operations research techniques, as well a tools to describe, portray, compare, rank, and evaluate different policy alternatives are part of the tool base. More basic, but just as essential, are simpler tools such as editors, and devices to represent output in its truly multi dimensional and dynamic nature. Tools that are considered of immediate relevance to the IRBM-DSS will be discussed in section 5.4.

**The model base:**
Finally, most essential in representing the decision domain are the domain-specific (simulation) models capable of grasping the complexities of the system and the problems at hand. The elements in the model base are of a formal nature, and exclude decision-making solely based on common sense or intuition. Elaborate model bases will contain both mathematical and rule-based techniques, often playing complementary roles in the decision-making process. In the context of the IRBM-DSS the models available for use in the DSS are available in a distributed manner from different authors and institutions. This fact, the end-use requirements and a set of performance criteria should be essential considerations in the selection of an architecture for the IRBM-DSS (see section 5.2)

The basic functional components can be integrated into a DSS application in various ways. Four alternatives will be discussed in section 5.2. They differ foremost in tightness of model integration. The alternatives are evaluated on the basis of 15 performance criteria including among others the development cost and various usability aspects.
5.2 Model integration

5.2.1 Approaches
As discussed in section 5.1 a core element in every DSS is the model base. This is not different in the IRBM-DSS. The very technical character of ‘watershed management’ itself and the addition of the characteristic ‘Integrated’ to this requires a very rich model base with models operating at different spatial and temporal scales and in very different domains.

Moreover, the IRBM-DSS will be developed on the basis of existing modeling material available from a great number of organizations distributed over Germany and the Czech Republic. This modeling material has been developed primarily for research purposes and will need to be made available in an instrument intended for policy purposes. Chapter 5 of the paper in Appendix A explains at some length the difficulties that are involved in adapting and coupling this research material for policy-making purposes. We will not repeat this here; rather refer the interested reader to the appendix.

A thorough analysis of the available modeling material has been carried out. A description of the available models has been given and a suggestion for a first selection and integration of compatible models at three different levels of spatial detail has been presented in the dedicated chapter. We will focus our attention on the consequences the above has for the development process and the architecture of the system.

We believe that essentially four different types of architecture could potentially be used to build the IRBM-DSS. They differ from one another mostly in terms of:

- their policy support, policy relevance, and performance;
- the development and maintenance costs;
- the expected technical and organizational difficulties involved in the development of the DSS.

These characteristics have been captured in 15 selection and evaluation criteria. We will first give a brief working definition of each criterion and next will discuss the merits of each of the four architectures.

5.2.2 Evaluation Criteria
We retained 15 criteria to evaluate the different architectures. Other lists would probably have been possible and other interpretations of the terminology and definitions could have been used. However, we believe that this list is sufficient to demonstrate the differences between the architectures and their relevance for the usage and the end-user of the IRBM-DSS.

Policy relevance
Policy relevance refers to the way in which the system provides immediate support for the policy end-user. It is about how well it is tailored to his needs, his skills, and his method of working. In this study we have defined the policy user as a person knowledgeable about
the policy domain, but less proficient in computer usage in general. Further, this person is a high-level user, meaning also that he does not necessarily have the time to work his way through lengthy computer procedures and routines.

Policy relevance is also important with respect to the various models and processes that are represented in the DSS. The policy relevant processes are a (possibly small) subset of all scientifically interesting processes in the problem domain. Generally a process becomes policy relevant if the policy maker has some direct or indirect measure to influence the process.

**Explorative learning**

Explorative learning refers to the ease with which the user can learn about (a part of) a problem by means of the system. Learning will only take place if the user understands causes and effects in the system. Hence, the system will need to be highly transparent and user friendly. The level of complexity of the system should be within the limits the learner can handle, and the system needs to be responsive, meaning that an input of the user should result in an output by the system in a manner that makes intuitive sense to the learner. Moreover, the system should produce output that is as instructive and as concise as possible: geographical output in the form of maps, time series as time charts, etc. Further, the user-interface should be uniform for as much of the components of the system as possible and the system is best equipped with a model that represents all the processes at the same levels of abstraction and detail.

**User friendliness**

User friendliness of the system refers very much to the ease with which the system can be used by its intended end-user. As little as possible time should be lost in executing tasks that are not immediately relevant to the problems for which it is developed. User friendliness is among others obtained if the system has a well designed, intuitive and uniform user interface which is set-up according to guidelines pertaining to the operating system and platform on which the system (at least its front-end) runs. Also, a system is most user-friendly if it is equipped with an appropriate and easy to manipulate set of tools required for carrying out the analytical tasks.

**Transparency**

Transparency refers to the tractability of the results generated by the system as well as the documentation of the different tasks carried out by the system. The more the system will carry out its tasks in a manner that makes intuitive sense to the end-user, the more transparent it will be. The more models and tools of the system are opened up and documented, the more transparent it will be. Black box systems lack transparency.
**Interactivity**

Interactivity refers to the ease with which the end-user can interact with the system. What proportion of the tasks can the user carry out directly and via the user-interface of the system without having to fall back on other analytical instruments? What tools are available in the system to support the user in carrying out these analytical tasks during a session with the system? How much effort then is involved in carrying out the tasks, and what kind of maneuvers are required on behalf of the user? How much of this can he do on the spot as he pleases and without having to refer to other software or other instruments?

**Integration**

Integration refers to the level of model and tool integration attained in the system itself. The different models in a system can be loosely coupled only, or they can be very strongly coupled. Coupling refers to complexity of the models: the number of state variables that are exchanged among the models in the system. Integration also refers to the way in which the tools fit the functional and analytical requirements of the models.

**Flexibility**

Refers to the notion discussed in section 4.4. Flexibility refers to the ease with which the system can be adapted or changed for tackling other problems, or for similar problems in another region or context.

**Correctness**

Refers to the quality of the output generated by the system. The level of correctness of the system will mostly depend on the quality of the models used in the system and by the way these models are coupled into a single integrated model. It is a difficult scientific notion, which is strongly linked to uncertainty, predictability, the complexity of the problem modeled, etc. In the context of this analysis, we look at correctness partly as the loss of information from the original representation of the process by means of a model.

**Completeness**

Completeness refers to the proportion of relevant domain processes that are generally represented by the models and the tools of the system at a sufficient level of detail. Completeness differs from abstraction level in the sense that a complete system does not need to consist of models that are fully coupled, nor do these models need to run at the same levels of detail, same temporal scales, same set of state variables, etc.

**Abstraction level**

The abstraction level refers to the level of detail with which the system represents the decision domain. The level of detail attained in the system should be appropriate and relevant to the kind of problems that need to be solved by the end-user. In this study the end-user is a policy maker rather than a researcher, hence the models in the system should
be evaluated on their policy relevance rather than their research relevance.

**Performance**
Refers to the speed with which the system, its models and tools generates results that are immediately relevant to the end-user. Performance therefore is relative to the platform and the machine, which is typically available to the end-user. A system that runs fast on a standard machine and on a widely available platform, such as MS Windows, is said to have a high performance.

**Development cost**
Refers to the costs required to build a running version of the final version of the system. Development costs include not only the software implementation, but also the preparatory work involved in functional and technical design of the system. It does not include the maintenance costs.

**Maintenance cost**
What are the costs involved in maintaining and upgrading the system? In the maintenance costs we include the costs to adapt the system to the changing needs of the end-users as well as the software and hardware standards.

**Collaboration**
Collaboration refers to the potential for the distributed development, maintenance, and use of the system. This quality is dealt with in section 5.5

**Implementation (level of difficulty)**
Implementation refers to the technical difficulties that need to be solved by those involved in the construction of the system. This includes the difficulties in the practical realization of the architecture and the functional components of the system: its model base, its tool base, its databases, and its user interface. Generally speaking the higher the level of difficulty the more risks are involved that the production of the system will meet with a lot of technical and organizational difficulties.
5.2.3 4 possible architectures

In Figure 5-2 a graphical representation of the 4 architectures is given. For each solution, the user is represented at the left. He interacts with the IRBM-DSS via a user-interface and has access to an integrated model. The integrated models consist of different component sub-models available from the model base of the DSS. On the right of the diagram is shown how precisely the sub-models are integrated.

The solutions can be synthesized as described in the following sections. Table 5-1 contains a kind of score table showing the weak and strong points of the four solutions.

5.2.3.1 Access to loose and distributed models

In this solution, the integration of the models is only weakly developed. Most sub-models are available to the user as they are and in a near to original form. Only the most essential adaptations have been carried out for making them more useful in the context of the IRBM-DSS. The adaptations to the models, their maintenance and their management remains very much with the owners and original developers. The models basically reside on the computer of the owner, usually together with the data required to run them, hence the burden to store and run the models resides with the owners equally well. When the DSS is used, it will make use of the sub-model either via a direct network access, or via the owner as an intermediate. In the former case, synchronous usage is possible, while in the latter, the usage will always be asynchronous. Rather than running the model directly, the owner will be requested to run it on his machine and return the results to the DSS end-user for further analysis.
This solution is not user-friendly. In the worst case it requires complicated procedures to do even the simplest things. It is also very slow, in that it might take a long time to get model results. It has also some advantages. Providing that a sufficient set of procedures and protocols are being adhered to, a distributed development is possible. The development and maintenance costs are kept rather low, and the models are updated by those knowledgeable.

We believe this solution is to be selected if access to very complicated models is required, e.g. models that would not run easily on the platform of the end-user. Models too, that are used for technical tasks and not so much for explorative or learning purposes. For the IRBM-DSS it would only be an acceptable solution if the access to the sub-models would not require human agents acting as interfaces and if the DSS could simplify greatly the transfer of data from one sub-model to the other.

In summary:

- Low development cost
- User-friendliness low
- Thin knowledge management shell to link and reference various research models
- Distributed architecture
- Distributed development and maintenance
- Weak integration
- Human and/or software interface agents
- Asynchronous
- Internet based client-server architecture

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Distributed KM shell</th>
<th>Components (weakly integrated)</th>
<th>Systems model</th>
<th>Systems model &amp; Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policy relevance</td>
<td>--</td>
<td>+</td>
<td>++</td>
<td>+</td>
</tr>
<tr>
<td>Collaboration</td>
<td>--</td>
<td>++</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>Explorative learning</td>
<td>--</td>
<td>+</td>
<td>++</td>
<td>+</td>
</tr>
<tr>
<td>User friendliness</td>
<td></td>
<td>+</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>Transparency</td>
<td>-</td>
<td>+</td>
<td>++</td>
<td>+</td>
</tr>
<tr>
<td>Integration</td>
<td>--</td>
<td>+</td>
<td>++</td>
<td>+</td>
</tr>
<tr>
<td>Flexibility</td>
<td>++</td>
<td>+</td>
<td>-</td>
<td>--</td>
</tr>
<tr>
<td>Correctness</td>
<td>++</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Criterion</td>
<td>Distributed KM shell</td>
<td>Components (weakly integrated)</td>
<td>Systems model</td>
<td>Systems model &amp; Detail</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>----------------------</td>
<td>--------------------------------</td>
<td>---------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>Performance</td>
<td>++</td>
<td>+</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>Development cost</td>
<td>++</td>
<td>+</td>
<td>-</td>
<td>--</td>
</tr>
<tr>
<td>Maintenance cost</td>
<td>++</td>
<td>+</td>
<td>-</td>
<td>++</td>
</tr>
<tr>
<td>Completeness</td>
<td>++</td>
<td>+</td>
<td>-</td>
<td>++</td>
</tr>
<tr>
<td>Abstraction level</td>
<td>--</td>
<td>-</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>Implementation (level of difficulty)</td>
<td>++</td>
<td>+</td>
<td>-</td>
<td>--</td>
</tr>
</tbody>
</table>

Table 5-1: Four architectures evaluated against 15 criteria

5.2.3.2 Existing models coupled into a single system

This solution differs from the previous one in that the integrated model consists of sub-models that are more tightly coupled and reside on the machine of the end user. This results in a system that is user friendlier. It is more suitable for explorative use and learning; yet, it enables rather technical usage as the constituting models will usually be moderately changed versions of the originals. The performance of this system will be better as well.

But, the development and maintenance costs will be medium, as models will need to be partly rebuilt and re-implemented to fit the integration scheme. Also, the system needs to be equipped with tools and techniques to run models operating at different geographical and temporal scales simultaneously. Providing that the system is built on the basis of state-of-the-art component technology (see section 5.5), it will be possible to re-use sub-models written in different programming languages. The latter will also enable a distributed development, and allow for the owners of the original models to participate actively in the design and implementation of the ‘light’ versions of their (sub) models. If this kind of collaborative effort can be organized and kept alive, then the DSS will be strongly anchored in the relevant research fields.

This solution combines a high degree of integration with acceptable development and maintenance costs. It is to be advised for the development of the IRBM-DSS if analysts and policy-makers will use the system alike. It will become more interesting and feasible if a good collaboration with the original model developers can be guaranteed.

In summary:

- Medium development and maintenance costs
- User-friendliness medium
- Standalone application
• Linked (adapted) research models
• Medium integration
• Various spatial and temporal scales
• Component and wrapper technology
• Programming language independence
• Collaboration between domain experts of various disciplines

5.2.3.3 Reformulation of existing models into a systems model

The core element in this architecture is the integrated model. It is fully tailored to the precise role of the DSS and the needs of the end-user. Hence, a clear problem definition an in depth user requirements analysis and a precise user profile are essential elements to decide on the precise depth and the extent of the integrated model and its constituting sub-models. More than in the previous two solutions, this integrated model is very strongly coupled. The model is a truly complex model. It is so by design, and, as each of the sub-models are (re-)developed and (re-)implemented for this purpose it is also materialized technically. Once realized, this system is user-friendly and, it usually will have a high performance. More than in the previous two cases, the system will represent all relevant processes at the same level of abstraction, namely the level demanded by the end-user. The processes will be modeled at an appropriate level, which does necessarily produce the most detailed or most accurate results possible.

Because of the effort spend in the design and implementation it is also a medium to high cost solution. Also the maintenance costs can be very high. This will certainly be the case if the use of the system changes. Development and maintenance of this system can be kept in the hands of a small group of modelers and system developers. There is no real need for distributed development as the near complete model is build from scratch. However, with a view to keep the maintenance costs within limits, it is advisable to choose a component-based technology to implement this solution.

This solution is to be recommended if the IRBM-DSS is used by high-level policy user and for explorative integrated assessment exercises mostly, because it enables the development of a well-balanced, transparent system. For the same reasons this solution supports explorative learning very well.

In summary:

• Medium to high costs
• User-friendliness high
• End-user involvement in development high
• Interactivity is high
• Highly integrated DSS for policy support
• Policy-model developed from adapted or rebuild research models
• Knowledge acquisition phase to fill gaps (missing links) with newly developed models
• Standalone application

5.2.3.4 Single systems model with access to original models

One of the disadvantages of the previous solution is the fact that the integrated model lacks the accuracy to perform detailed calculations on what would be considered less important components of the real world system represented. It would for example be very difficult to include beetle dynamics, observed in a very particular river...
ecosystem, in an integrated model covering the complete Elbe watershed. This level of detail would be difficult to attain because in most of the watershed the data required would not be available. The policy maker might not find this a problem, but an analyst using the system, an ecologist for instance, might have a different view on this. As a solution to this problem, an architecture could be chosen representing the watershed by means of an integrated model as explained in the previous solution, but which in addition permits access to more precise, original models for a selected set of processes. The latter models can be run on particular relevant spots in the watershed for which the data are available. Moreover, the output of the detailed models could be exchanged with the integrated model, and visa versa, thus permitting a more complete analysis of a problem.

This architecture combines the advantages of the second (5.2.3.2) and the third (5.2.3.3) solution presented. It is not to wonder that it is the most expensive of all, both in terms of development and maintenance costs. But it has major advantages too. It is a user-friendly solution, providing maximum accuracy, completeness, interactivity, and flexibility. Also, the policy relevance is high, while the analytical user can still make use of the system effectively.

This solution is to be recommended for the IRBM-DSS. This is mostly so because of the vastly different spatial and temporal scales, which need to be covered in the system. Also, the requirements on behalf of the intended end-users are rather different.

In summary:

- High development costs
- User-friendliness max
- End-user involvement in development maximum
- Various spatial and temporal scales
- Standalone or distributed
- Policy-model developed from adapted research models and integration of full-size models
- Knowledge acquisition phase to fill gaps (missing links) with newly developed models

Providing that model integration is a major obstacle in the development of the IRBM-DSS, and based on the information gathered so far, it would seem that the most appropriate architecture is the last one described. It combines the advantages of a stand-alone system running on the machine of the end-user and those of a strongly integrated model. The development of the system could in part be implemented in a distributed manner, and thus involve the scientists and developers of the original research models at best. The suggestions made in the modeling chapter relative to the nature of the integrated model are fully compatible with this conclusion.

5.3 Data integration
The integration of existing data into the DSS system is particularly important. Not only because the integrated models need the 'right' data in order to run successfully, but also because it should be possible to compare and validate the (model) calculations against monitoring data and real world observations.
Huge hydrological or hydro-morphological databases already exist (e.g. at the BfG) as well as databases of land use or statistical data and should be used for the DSS. Concerning the water quality aspects, huge databases exists as results of former research activities (e.g. the Elbe 2000 program) or permanent measuring campaigns (e.g. from the ARGE Elbe). The data from Elbe 2000 will be useful for comparison with modeling results.

All the heterogeneous databases named above reside on different hosts and are administered by different organizations. In general this situation of shared data storage and maintenance is most practical and assures the most actual data with best data quality. For this reason this structure should be used in the DSS as well.

Generally the following three types of data integration can be distinguished as the main alternatives:

**Stand-alone database:** at the moment this is the most common way of data integration. The database is directly part of the DSS system or the database management system is external but resides on the same host. Data updates are realized by changing database files and are only possible by the way of editing or uploads of new files.

**Offline-connection to external database:** The DSS opens the possibility to get a connection (typically by internet) to external databases for updating the data uses into the DSS. The DSS relevant data aren’t stored anymore in the DSS system itself or a database on the same host. This alternative is the best way of data integration if large external databases with relevant data exists that should be used for data update but the update is not necessary in very short periods (e.g. in the order of hours). In case of a IRBM-DSS for the Elbe this is the prior alternative to use the several existing databases for updating properties in a suitable way.

**Online-connection to external database:** This alternative requires a direct and permanent connection to an external database management system. In practice this restricts the usage of the DSS to sites were an Internet connection is available. Otherwise it possible to shift the whole database part of the DSS to external hosts. Indeed problems may occur if the external database is not available. For DSS systems this concept of highly distributed resources should be preferred if always actual data are necessary for the simulations. In the case of an IRBM-DSS this is not the preferred alternative.

### 5.4 Tool integration

The tool base is the component that usually gets the least attention in the DSS. Many authors will either consider it to be an integral part of the user interface. Others still will consider it to be part of the model base. Both views in our opinion are wrong and tools should be treated with particular attention because they decide to a large extent on the usability and effectiveness of the DSS. Indeed, in a well-designed DSS the tools are the gnomes that will carry out the many small technical tasks in the background of the system. The user will hardly be aware that he is using a tool when he is editing a parameter or viewing a variable, but without them, the most sophisticated model would be totally useless. Tools are among the most robust elements in a DSS; hence they can be re-used easily in applications in combination with vastly different model bases and accessed by different types of user interfaces.
We will not dwell extensively on the specifications of the many different tools that are required in a fully blown DSS. Instead we present in Table 5-2 a list of tools that we consider at the least useful in the context of the IRBM-DSS. Most of these tools are available from the DSS applications, which we have developed in the past. We order the tools relative to the type of task they carry out in the DSS, and distinguish between Input tools, Output tools, Exploration tools, Evaluation Tools, to end with a number of tools worth developing for incorporation in a DSS aimed at integrated river basin management.

Among the **Input tools** there are the typical editors required to change single numbers, or series of numbers in a textual or graphical manner. As part of the latter, the table editor, which enables entering 2-D relations as a curve and the map editors are very powerful instruments.

The **Output tools** take care of the difficult task to present massive amounts of data generated by the DSS is as concise and precise a manner possible. The developer of the DSS will make decisions on the kind of tool to use. To present spatial and dynamic data dynamic maps are essential, and so are recorders and players of animations, because they enable the user to take more time to analyze the results generated.

**Exploration tools** enable the user to interactively perform searches in the solution space. The possibility to generate scenarios, produce map overlays, perform map comparisons, and carry out sensitivity analysis puts great analytical power in the hands of the end-user.

**Evaluation tools** will help the user decide on a ‘best’ solution. Techniques such as Multi Criteria Analysis, Score Tables and the like are wanted. But, rather than running the risk that the ‘best’ solution is never attained when series of ‘what-if’ analyses are carried out, the user might be much helped with some from of goal seeking and optimization techniques.

Finally, and in the particular context of integrated watershed management, we could think of a couple of instruments that would enable carrying out tasks that otherwise would be difficult to perform. We refer here to very specific interventions such as groin modification and dike shifting. But also more generic operations like cell-to-network and network-to cell conversion, or cell aggregation and cell disaggregation seem like very useful instruments.

<table>
<thead>
<tr>
<th>Tools</th>
<th>Analysis</th>
<th>Communication</th>
<th>Learning</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input tools</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>text editor</td>
<td>++</td>
<td>+</td>
<td>+++</td>
<td>+++</td>
</tr>
<tr>
<td>value editor</td>
<td>+++</td>
<td>+</td>
<td>+++</td>
<td>+++</td>
</tr>
<tr>
<td>series editor</td>
<td>+++</td>
<td>+</td>
<td>+++</td>
<td>+++</td>
</tr>
<tr>
<td>table editor</td>
<td>+++</td>
<td>++</td>
<td>+++</td>
<td>+++</td>
</tr>
<tr>
<td>function editor</td>
<td>++</td>
<td>+</td>
<td>+</td>
<td>+++</td>
</tr>
<tr>
<td>network (points and lines) editor</td>
<td>++++</td>
<td>+</td>
<td>+++</td>
<td>+++</td>
</tr>
<tr>
<td>2D map editor</td>
<td>++++</td>
<td>++</td>
<td>+++</td>
<td>+++</td>
</tr>
<tr>
<td><strong>Output tools</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>on-line documentation</td>
<td>++++</td>
<td>+++</td>
<td>+++</td>
<td>+++</td>
</tr>
<tr>
<td>on-line help</td>
<td>++</td>
<td>+</td>
<td>+++</td>
<td>+++</td>
</tr>
<tr>
<td>Tools</td>
<td>Analysis</td>
<td>Communication</td>
<td>Learning</td>
<td>Library</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------</td>
<td>---------------</td>
<td>----------</td>
<td>---------</td>
</tr>
<tr>
<td>time graphs</td>
<td>++++</td>
<td>+</td>
<td>++++</td>
<td>+</td>
</tr>
<tr>
<td>dynamic maps (2D)</td>
<td>++++</td>
<td>++</td>
<td>++++</td>
<td>+</td>
</tr>
<tr>
<td>dynamic maps (network)</td>
<td>++++</td>
<td>++</td>
<td>++++</td>
<td>+</td>
</tr>
<tr>
<td>3D representation</td>
<td>++</td>
<td>+++</td>
<td>++++</td>
<td>+</td>
</tr>
<tr>
<td>Animation</td>
<td>++</td>
<td>++++</td>
<td>++++</td>
<td>+</td>
</tr>
<tr>
<td>tracing tool</td>
<td>+++</td>
<td>++</td>
<td>++++</td>
<td>+</td>
</tr>
<tr>
<td>Exploration tools</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>POLICY-wizard</td>
<td>+</td>
<td>+++</td>
<td>+++</td>
<td>+</td>
</tr>
<tr>
<td>OVERLAY-Tool</td>
<td>++++</td>
<td>+++</td>
<td>+++</td>
<td>+</td>
</tr>
<tr>
<td>ANALYSE-Tool</td>
<td>++++</td>
<td>+++</td>
<td>+++</td>
<td>+</td>
</tr>
<tr>
<td>SCENARIO-Tool</td>
<td>++</td>
<td>+++</td>
<td>++</td>
<td>+</td>
</tr>
<tr>
<td>MONTE CARLO-Tool</td>
<td>+++</td>
<td>++</td>
<td>+++</td>
<td>+</td>
</tr>
<tr>
<td>Evaluation tools</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCORE-Table-Tool</td>
<td>++++</td>
<td>+++</td>
<td>++++</td>
<td>+</td>
</tr>
<tr>
<td>EVALUATE-Tool (MCA)</td>
<td>+++</td>
<td>+++</td>
<td>++++</td>
<td>+</td>
</tr>
<tr>
<td>Optimization</td>
<td>++</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Goal seeking</td>
<td>+++</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>IRBM-specific</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dike shifting</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>+</td>
</tr>
<tr>
<td>Groin modification</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>+</td>
</tr>
<tr>
<td>Cell aggregation / cell disaggregation</td>
<td>+++</td>
<td>+</td>
<td>++</td>
<td>+</td>
</tr>
<tr>
<td>Cell-to-network / network-to-cell conversion</td>
<td>+++</td>
<td>+</td>
<td>++</td>
<td>+</td>
</tr>
</tbody>
</table>

Table 5-2: A list of tools and their usefulness to different users

In Table 5-2 we also value the importance of the tools relative to the potential uses or functions of the Decision Support System. From the table it is clear that the analyst will be serviced best by as complete a set of tools. He will need both the very down to earth instruments to enter data, to read the input and to analyze the output. When the main purpose of the DSS is communication, then the need for pertinent output instruments is very essential. The learner needs a bit of everything, his usage of the system is not fundamentally different from the analyst’s. The main difference is that the analyst will want the more sophisticated instruments in support of his analysis, while the learner is foremost interested in the documentation systems and the simpler input and output tools. Finally the library function of the system requires the facilities to quickly enter, update and retrieve the information stored in the library. A good documentation system is paramount.

5.5 Component-based development
In section 4.4 we have argued that our system should fulfill certain requirements of generality and flexibility. In section 5.2 we have referred a couple of times to distributed development and re-use of existing models. In this section we will dwell more on these issues and describe how state of the art software technologies enable to develop information systems that to a large extent use and integrate existing parts. The integration effort does not stop with the delivery of the first prototype but is a constant activity during the whole lifecycle of the system. To our knowledge the best
The basic idea of CBD is old, simple and powerful: to build a complex whole, try to assemble it from less complex parts, which you might already have. Organizing things in whole-part hierarchies is a basic human strategy to deal with complexity and its application can be seen in nearly every modern product of mechanical or electrical engineering. To be able to assemble something useful from parts, the parts need to fit together. This is where interfaces come into play. Interfaces can be seen as a specification or contract to which parts must adhere to be compatible with each other. A good metaphor for an interface from the physical world is the plug with its corresponding socket. CBD lets software engineers build complex systems in a way that is very similar to how their hardware colleagues design a new car or a computer motherboard: by assembling compatible parts.

A convenient 1-page overview of CBD is given in the summary of a recent paper by Alan Wills [Wills 00b]:

The basic principle of CBD is polymorphism, or pluggability: variant behavior is produced by parameterizing & reconfiguring components, which themselves have fixed designs. The same principle applies on any scale. There are a variety of technologies by which the components can be composed.

Components should preferably be designed in kits. A kit is a coherent set of components, with a minimal set of interfaces.

There are three principal activities in component-based development, requiring different balances of maintainability and cost:

- Kit architecture: defining common models and interfaces
- Component development
- Product assembly from components

Interfaces defined as lists of function calls, are too low-level for component-based design. It is better to design with more abstract connectors, which include different kinds of transfer or transaction. The choice of connectors used in a particular kit is part of its architecture; the ‘wiring’ may support some of them directly.

Each component may have its own partial view of the complete business model. The design method must include ways of mapping these views.

Components may be distributed across different machines. In this case, the design method must include patterns that deal with the possibility of links and machines going out of service, and treat links as potential bottlenecks.

In order to achieve good component based development:

- The structure of the software producing organization must be such as to resource the three development levels (architecture, components, products).
• The teams must be appropriately skilled in an appropriate design method that includes appropriate patterns for component and distributed development; including the notation and means to define connectors, to define components, and to build products from components.

• Select appropriate tools to support the design method; and select tools and platforms to support the architecture.

For a more thorough introduction into CBD we recommend the excellent papers by Alan Wills [Wills 00a] and [Wills 00b]. A lot of useful and up-to-date information on CBD can also be found at: http://www.cbd-hq.com/.

The value of CBD for DSS development in general and of component technologies COM and ActiveX specifically was demonstrated in the MODULUS project (see Appendix A). With model integration and reuse being the main themes of MODULUS, RIKS developed a wrapper technology based on ActiveX and COM, that made it possible to integrate models from various sources and implemented in various programming languages.

Component technology also can help to enhance the flexibility of the DSS by decoupling its subcomponents (e.g. tools, models, GIS access, user interface…) and introducing interfaces as their only way to exchange information among each other. For example separating the user interface layer from the simulation kernel yields a much more flexible architecture. A stand-alone version and an Internet version of the DSS could then be built with identical simulation kernels but different front-ends. The simulation kernel would not even need to know to which kind of front-end it is connected. Of course the same holds for the various tools needed in the DSS. By ensuring that all information exchange between the tool and its environment goes via its component interface, the tools internal structure is completely hidden.

Catalysis™ is a mature and industry tested CBD methodology and is fully described in [D'Souza 99]. Catalysis™ is based on object oriented analysis and design principles, but goes much further than classic OO by including explicit representations for patterns, frameworks, design by contract, interfaces, connectors, rule sets, a process model etc… Catalysis™ uses the Unified Modeling Language (UML) as its graphical formalism. For a short-term IRBM-DSS prototype project, depending on budget and project run-time, it might be worthwhile to consider a ‘light’ version of Catalysis™, to prevent too much method overhead.

5.6 Object-oriented application frameworks

In the previous paragraph, we discussed component-based development and how this technique may be applied in the DSS development to achieve a more flexible and modular system architecture. Application frameworks are built upon object / component technology and go even further in achieving reusability. While component libraries could be described as providing ‘bottom-up’ reuse, object-oriented application frameworks provide ‘top-down’ reuse. An object-oriented application framework for an IRBM-DSS provides a generic application template for this type of DSS. It would provide interfaces and extension points for the developer to ‘fill-in’ the application specific models, tools and data. It therefore allows saving substantial
development costs for extensions to an existing application or subsequent DSS applications.

This section will give an introduction on what object-oriented application frameworks are and how they are used for the development of DSS. Section 5.6.4 describes Geonamica®, an example of such a framework, which we recommend to use for the development of the Elbe-DSS. For in depth information about component and framework development we recommend [D’Souza 99] and [Fayad 99].

5.6.1 What is an (object-oriented) application framework?
Like many other concepts of modern software engineering (e.g. patterns) the notion of a ‘framework’ has its roots in architectural design. Timber frame or steel frame building constructions are the basis for many low cost building designs of high quality. Framework users build specific solutions, benefiting from reuse of a generic architecture of collaboration patterns and building blocks of proven design.

A framework is a reusable design of a system that describes how the system is decomposed into a set of interacting components. The framework describes the component interfaces as well as the collaboration patterns between the components by means of special purpose textual (IDL, OCL) or graphical (UML) languages. Besides an abstract design specification, application frameworks typically provide an implementation in the form of a semi-complete skeletal application that can be specialized to produce custom applications.

Object-oriented frameworks are designed and implemented as a collection of collaborating abstract classes. They make use of the three basic principles of object technology [Graham 98]:

- **Encapsulation** – data and processes are combined in objects and hidden behind an interface.
- **Polymorphism** – is the ability to use the same expression to denote different operations.
- **Inheritance** – implements the idea of classification and represents a special case of structural inter-relationship between a group of objects.

Essentially all these are basic human strategies to manage complexity; this is why object-oriented models often are perceived as more ‘natural’ compared to other approaches.

The recent focus of attention on component technologies like COM+, CORBA and Enterprise Java Beans (EJB) as well as component based development (CBD) in general, has triggered a lot of discussions about what ‘objects’ and ‘components’ have in common and what distinguishes them. Most of this discussion is outside the scope of our discussion of frameworks in this text. Where appropriate differences between ‘objects’ and ‘components’ will be explained. In our view component technology is largely built upon the foundation of object technology and shares with it the above-mentioned basic principles. See [Wills 00b] for a more detailed discussion of this topic.
5.6.2 What are the benefits of application frameworks?
The primary benefits of application frameworks stem from two types of reuse: design reuse and implementation reuse. The observation that core concepts and components and their interactions within a domain are relatively stable, has led to the notion of ‘design patterns’. By delivering a useful set of patterns as a documented design (design reuse), as well as a partial solution in form of a skeleton application (implementation reuse), a framework may save a lot of costs for rediscovery and reinvention.

In [Fayad 99] Fayat et. al. describe the following benefits of application frameworks:

- **Modularity** – Frameworks enhance modularity by encapsulating volatile implementation details behind stable interfaces. […]
- **Reusability** – The stable interfaces provided by frameworks enhance reusability by defining generic components that can be reapplied to create new applications. […]
- **Extensibility** – A framework enhances extensibility by providing ‘hook methods’ that allow applications to extend its stable interfaces. […]
- **Inversion of control** – The runtime architecture of a framework is characterized by an inversion of control. This architecture enables canonical application processing steps to be customized by event handler objects that are invoked via the framework’s reactive dispatching mechanism. […]

5.6.3 Framework classifications

In [Fayad 99] object-oriented application frameworks are classified by their scope and by the techniques they offer for extension:

**5.6.3.1 Framework classification by scope**

- **System infrastructure frameworks** simplify the development of portable and efficient system infrastructures […]. They are primarily used internally within a software organization.
- **Middleware integration frameworks** are commonly used to integrate distributed applications and components.
- **Enterprise application frameworks** address broad application domains […] and are the cornerstone of enterprise business activities.

Geonamica® is an example of an enterprise application framework for DSS in the domain of complex spatial policy problems. Geonamica® also has some features of a middleware integration framework in the sense that it supports the integration of external models with component technology.

**5.6.3.2 Framework classification by extension technique**

- **White-box frameworks** rely heavily on OO language features like inheritance, dynamic binding, templates (C++) or generic classes (Eiffel) in order to achieve extensibility. Existing functionality is reused and extended by (1) inheriting from framework base classes and (2) overriding pre-defined hook methods and using patterns like the Template Method [Gamma 95].
Black-box frameworks support extensibility by defining interfaces (IDL, Java) for components (COM, JavaBeans, CORBA) that can be plugged into the framework via object composition. Existing functionality is reused by (1) defining components that conform to a particular interface and (2) integrating these components into the framework using patterns like Strategy and Functor [Gamma 95].

Geonamica® is an example of a white-box framework. White-box frameworks are much more widely used than black-box frameworks. This is partly because they are conceptually easier to develop and partly because (programming language independent) interface description languages only recently became available to the software engineering community.

Despite their wide use, with respect to extensibility, white-box frameworks have some considerable disadvantages compared to black-box frameworks, especially when the planned extension involves the integration with other frameworks. White-box frameworks require the application developer to have deep knowledge about their internal structure, especially the inheritance relationships. For example, application developers that want to reuse functionality from the Geonamica® library, need to inherit this functionality from the Geonamica® base classes and need to override their virtual methods.

In contrast with the former, in black-box frameworks reuse is realized by composing new objects from predefined framework components and by implementing framework defined interfaces in application objects, in order to enable these objects to participate in framework defined collaboration patterns. This form of reuse is more flexible, since it is not coupled to a specific inheritance hierarchy.

5.6.4 Geonamica® — a DSS Generator

Geonamica® is an object-oriented application framework, developed by RIKS. It is specially tailored for developing Spatial Decision Support Systems featuring integral dynamic models as their core element. Examples for interactive SDSS built with Geonamica® are among others: RamCO, SimLucia [Engelen et al. 98a], WadBOS [Huizing 98], Environment Explorer (LOV) [Engelen et al. 98b], MODULUS [Engelen et al. 2000a], and MURBANDY.

It is the support for dynamic modeling and simulation that distinguishes Geonamica® from other planning tools, in particular from those based on standard GIS technology. In particular the need for dynamic modeling on high-resolution data lead to the implementation of highly efficient computational techniques and algorithms. Geonamica® supports the development and execution of integrated models consisting of sub-models running at multiple spatial and temporal resolutions. Typically it will combine system dynamics models and cellular models for this purpose. In particular use is made of spatial interaction based models, different kinds of cellular automata models, multi agent or other kinds of rule-based models. But also the visualization of modeling results and the support of an iterative and interactive working method has been given special attention.

Geonamica® runs on the PC platform and manipulates primarily grid data. The available internal memory of the host machine is the only limiting factor in the size of the grids and the amount of spatial variables the system can handle. Grids with 1 million cells are processed without problems. For calculations involving limited
spatial interactions, the calculation and refreshment of this kind of grids will be in the order $10^{-2}$ seconds on a state of the art PC.

*Geonamica*® is not only equipped with a series of fast computational routines, but it also includes an important amount of analytical tools, visualization tools, and input, import, export and output tools. It is equipped with a number of cartographic tools, in particular map editors and display tools for 1D network and 2-D map objects. Also it supports interactive map comparison and analysis as well as interactive overlay-analysis.

5.6.5 Other examples of enterprise application frameworks

- OpenGIS
- Standaard Raamwerk Water (SRW)
- Framework Integral Water-management (FIW)

5.7 Integration of GIS functionality

A DSS system for integrated river basin modeling strongly requires capabilities to work with geo-referenced data so GIS functionality is needed. In general the following three main alternatives are possible:

1. Implementation of all needed GIS functionality in the DSS
2. Implementation of the DSS as an application in a commercial GIS framework
3. Building of a separate GIS layer in the DSS architecture, which is implemented with commercial GIS components

5.7.1 Implementation of all needed GIS functionality in the DSS

In principal it is possible to implement all needed GIS functionality in the DSS itself.

Advantages:

- self implementation of only those GIS-functions that are needed will lead to a lean system,
- the implementation can be highly adapted on the available models and data. This will increase the systems performance,
- no external GIS-software necessary, so that there are no additional license costs for the DSS.

Disadvantages and risks:

- implementation of GIS-functions may be very expensive depending on the extent and amount of needed functionalities and the needed data structures. The implementation costs may be reasonable if only a reduced set of functions is needed (e.g. standard raster data processing). Otherwise it will be too expensive to implement for a DSS if sophisticated functions of vector data processing (e.g.
complex analysis, spatial intersections or other) perhaps in combination with raster data processing are necessary.

- the GIS-functions of the system are fixed to the implemented set. If the users want additional functions they have to be re-implemented. This includes the situation that new models, which may need new GIS-functionalities, shall be integrated in the DSS.

5.7.2 Implementation of the DSS as an application in a GIS framework

Here two main alternatives exist: either the DSS is totally implemented within a (commercial) GIS system or it is coupled with the GIS in a direct or indirect way.

Advantages:

- all GIS functions of the commercial GIS framework can be used if necessary,
- the GIS-part of the DSS is open in general, what means that new and user-specific geo-data can be incorporated and used for further analysis,
- all other functions of the GIS-framework can be used, for instance for integration of external geo-data, mapping, report generation or other,
- possibly it is simple to realize different 'user modes' (see chapter 4.2). Expert users may use the DSS as an open system with the possibility of using GIS capabilities they are already familiar with.

Disadvantages and risks:

- the coupling of models and needed GIS functions may be difficult or perhaps unrealizable. Eventually the coupling will need a lot of software-technical overhead and therefore will lead to a system with less performance.
- normally the usage of commercial GIS frameworks cause license costs for each DSS implementation (e.g. approximately 3000 DM for ArcView®). This implicates that it will be impossible to distribute the DSS at no or only a little charge. As a result main potential user groups, first of all the public, will not have easy access to the DSS.
- if the DSS is only linked with the GIS possibly no uniform user interface will be available and there may be a 'Look and Feel' problem of the system. Probably performance problems may occur due to the use of the GIS-intrinsic macro language.

5.7.3 DSS architecture with separate GIS components layer

Commercial GIS components make it possible to provide a complete and tested set of GIS functions as a separate layer within the DSS software.
Advantages:

- a great range of GIS functions can be used, which are all well tested,
- the integration into the DSS system is flexible and efficient because only the needed GIS functions can be used and the performance of the system will be high,
- the GIS functions can be completely integrated under a consistent and uniform user interface,
- normally there will be no license costs for the GIS components for the developed software.

Disadvantages and risks:

- software-technical problems by integrating the different DSS components with the GIS components may occur,
- the costs of the development licenses are considerable (e.g. 25.000 DM for ArcIMS or MapObjects)
6 System Design

Rather than presenting a complete system design, which is outside the scope of this report, this section will outline innovative solutions to a selection of the most challenging engineering problems that we could derive from the problem definition report and the outline of the qualitative model.

To the most important distinguishing properties of the three modules of the qualitative model for the Elbe IRBM-DSS belong their various spatial and temporal scales. If we choose an integrated DSS architecture, the system has to deal with these scales simultaneously and in an integrated way. Integrated in this respect means, that processes that operate on different spatial or temporal scales can exchange information and influence each other.

Related to this problem, but not further explored in this report is the question of how to integrate models with different dimensionality.

6.1 Spatial scales

This section describes a Layered Cellular Automata (LCA) as a technique to handle multiple spatial scales in integral dynamic models. An integration of this approach into the Geonamica® framework is under development, an initial prototype is already available.

6.1.1 Multiple scale cellular automata

The processes we want to model and simulate within the IRBM-DSS applications differ vastly in temporal as well as in spatial scale. This holds even more, if we don’t want to exclude individual based ecological models from our framework.

Imagine we want to include an individual based model of some species of beetle into the integral model of the IRBM-DSS. The IRBM-DSS might cover an area of several thousand square kilometers with cells of $500 \times 500$ meters, which is typical to model land-use dynamics. On the other hand the behavior of the beetles might depend significantly on the properties and dynamics of an area of several square meters in their direct neighborhood.

Due to memory and execution speed constraints as well as lack of (high resolution) data, it is not an option to let the process with the smallest spatial scale define the spatial scale for the whole application. The previous example shows clearly that in order to build powerful integral models with various spatial scales we need a hierarchical representation of space.

In his landmark paper [Wolfram 86], Wolfram introduces the concept of multiple scale cellular automata and notes:

> Many kinds of complex systems can be considered as bases for engineering. Conventional engineering suggests some principles to follow. The most important is the principle of modularity. The components of a system should be arranged in some form of hierarchy. […]
Multiple scale cellular automata incorporate modularity, and need not to be homogenous.

Multiple scale cellular automata can have multiple time scales, multiple spatial scales or both. In [Wolfram 86] Wolfram gives several examples for multiple scale cellular automata, including one that operates on two time scales. The ‘slow’ automaton thereby controls the rules of the ‘fast’ automaton. The fast automaton completes several generations within one update cycle of the slow automaton and selects its rules according to the cell states in the slow automaton.

6.1.1.1 The structure of multiple scale cellular automata

The ecological modeling technique described in [Gronewold 98b] uses a 3-dimensional hierarchical structure, to define a Layered Cellular Automaton (LCA). A LCA is constructed by iterative refinement of 2-dimensional cellular automata, which yields a set of connected layers. In a LCA every cell on layer \( x \) can have exactly one parent cell in layer \( x - 1 \) and an arbitrary number of child cells in layer \( x + 1 \). These cells are called the ‘vertical neighbors’ of the cell in layer \( x \).

![Figure 6-1: Refinement and vertical neighborhood relations in a LCA](image)

Given layer \( x \) in a LCA, layer \( x + 1 \) is constructed by an operation called ‘refinement’ [Gronewold 98b]:

Let \( a \times b \) be the number of cells in level \( x \). Let \( C_x \) be a cell in level \( x \) and \( C_{x+1} \) the vertical neighbor in level \( x + 1 \).

- After the refinement, the level \( x + 1 \) has got \( k_{x+1} \cdot a \cdot l_{x+1} \cdot b \) cells, and the cell \( C_{x+1} \) is divided into a regular grid of \( k_{x+1} \cdot l_{x+1} \) cells \( C_{x+1}(i, j) \), \( i \in \{1, \ldots, k_{x+1}\}, j \in \{1, \ldots, l_{x+1}\} \).
- Let \( R \) be the neighboring relation between the cells on level \( x \). Each cell on level \( x \) has got the additional neighbors \( C_{x-1} \) (i.e. the vertical neighbor in level \( x - 1 \), if existing), and all \( C_{x+1}(i, j) \) (i.e. all cells of the refined vertical neighbor \( C_{x+1} \), if existing).

Figure 6-1 shows the structure of a LCA, which is very similar to a quadtree. In the definition of the refinement operation, as given in by Gronewold, it is assumed that all layers of the LCA cover the same area. However the main reason why we want to use LCA in integral modeling is that we need to represent space with variable resolution, in order to integrate models that operate on different spatial scales. This is not only due to technical reasons like limitation of memory and processing power but also the availability of models and data may vary considerably across the modeled region.
In real-world applications of integral modeling, models that operate at a higher resolution often cover a sub-area within the total area that is covered by the integral model. For example, we might want to embed a very detailed ecological model as a small patch within a much larger river basin model. A detailed model of a city could be embedded in a more coarse grained model of a larger region.

Therefore, the definition of refinement as given above, must be extended with the notion, that the refinement operation can be applied to a selection of cells, instead of applying it to the automaton as a whole. Figure 6-2 shows this situation: here a high-resolution model is embedded as a patch in a low-resolution model.

![Figure 6-2: Refinement of a selection of cells](image)

Note that with this extended definition of refinement, the resulting LCA may no longer be homogenous. In all but the lowest layer we now have two different kinds of cells, depending on whether or not they have vertical neighbors on a lower layer.

A complication introduced by selective refinement is that it is more difficult to define useful boundary conditions for the automata on the refined layers. Some constraints on the shape of the selection might help here. The simplest case is to restrict the selection to be a rectangle; boundary conditions then can be handled in the same way as it is done on the highest layer. A selection with a polygonal shape, as shown in Figure 6-2, is more complicated. While with a rectangular shape cells can only have 3, 5 or 8 direct horizontal neighbors, with a polygonal shape every combination from 1 to 8 neighbors is possible.

### 6.1.1.2 The dynamics of multiple scale cellular automata

The state of a cell $C_x$ on layer $x$ in a LCA depends on the state of the cells in its horizontal neighborhood and on the state of its single parent cell $C_{x-1}$ on layer $x-1$, and the state of its $k \times l$ child cells $C_{x+1}$ on layer $x+1$.

The dynamics of a LCA can be used to model complex interactions between large-scale and small-scale processes. For example, a cell $C_x$ on layer $x$ could use totalistic rules in order to aggregate information from its child cells on a higher resolution layer $x+1$. On the other hand, the value of $C_x$'s single parent cell $C_{x-1}$ on layer $x-1$ could be used to select from a set the rules $C_x$ uses with respect to its horizontal neighbors.
6.2 Temporal scales

6.2.1 Simulation control
The standard Geonamica® simulation engine is a discrete-time simulation controller. All states of all objects are updated periodically using equidistant time-intervals. The main advantage of this approach is its simplicity. Basically the simulation engine is a running clock, which for each time-step tells every simulation object to update its state.

In the MODULUS project [Engelen et al. 99b], where the simulation included several processes on different spatial and temporal scales, this approach proved too limited. With discrete-time simulation, to simulate several processes on different time scales, the process with the smallest time scale defines the unit of time. This can become very inefficient, because the update() methods of the software objects that represent the slower processes get called very often, only to find out that there is nothing to do.

To overcome the limitations of fixed scale discrete-time simulation, application frameworks for ecological modeling (e.g. ECO$\text{\textsc{sim}}$, SWARM), have simulation control mechanisms that follow the approach of object-oriented discrete-event simulation.

Lorek describes this approach in [Lorek 99] as follows:

---

...all [simulation] objects thereby inherit the ability to generate and consume events. Events consist of an object that takes notice of the event, a timestamp indicating the virtual time when the event will take place, and some action (local method of the object) that will be performed after the event has taken place. The simulation engine deals with all the events being scheduled by simulation objects.

---

Extending the Geonamica simulation engine with a limited form of discrete event simulation solved this problem in MODULUS. In this approach the simulation engine, basically is an event scheduler. Simulation objects register events with the simulation engine; the engine manages all the events in a dynamic priority queue and ‘fires’ them when appropriate. The control of the simulation is much more distributed compared to the centralized control in discrete-time simulation. Simulation objects are the instances that specify which events will happen, who will perceive them and how to act upon them.

Given this mechanism, events can also be seen as a form of communication between different agents in the system. For example a predator that enters a certain region generates a corresponding event and thereby informs its prey about its presence. The prey on the other hand knows how to deal with such an event and might generate a ‘flee’ or a ‘hide’ event.

Note that it is always possible to simulate discrete-time simulation with a discrete-event simulation engine just by scheduling a sequence of time-equidistant ‘update’ events. From the point of view of the simulation objects, this can be done nearly transparently, in some cases a thin wrapper will be needed, which does little more than receiving the ‘update’ event and calling the objects update() method.
7 The DSS Development Process

Every project that aims to develop a complex product, requiring substantial financial and/or human resources, should define and organize its activities in some kind of process based upon a methodology. This more or less formal process is what separates engineering from art. Despite the name ‘Software Engineering’, our young discipline has just begun to develop from art to engineering and this holds even more for the development of decision support systems.

A full-fledged, theoretically sound, development methodology for model-based spatial decision support systems is not yet available. Instead of that, in this section we will present what we have found to be ‘best practices’ in DSS development together with selected aspects of a process model for component based and object oriented systems development.

7.1 Lifecycle planning

Most DSS development proposals contain a description of a more or less formal development process or life cycle model. In most cases this is some variation of the well-known waterfall model. The waterfall model describes the software development process as a linear sequence of non-overlapping development activities: (1) Software Concept, (2) Requirements Analysis, (3) Architectural Design, (4) Technical Design, (5) Coding and Debugging, (6) Testing, Delivery and Maintenance.

The waterfall model has been successfully applied in projects where the requirements are well understood from the beginning. However, we don’t recommend this model for DSS development because it has only weak support for user involvement and iteration in the system development process. Iteration in the waterfall model tends to generate a lot of rework and therefore is extremely expensive. Furthermore the waterfall model does not define a process for reuse management and is proven to be less suitable for object-oriented and component-based development.

In DSS development, requirements are seldom well understood at the beginning of the project; therefore early user involvement and a collaborative and co-operative approach between all stakeholders throughout the project are critical success factors. Such an approach is much better suited by an incremental development method that elicits requirements in joint workshops and delivers working prototypes at regular intervals. There are several object-oriented development methods with these properties, in particular we recommend SOMA (see section 7.3).

7.2 Changing requirements – hitting a moving target

One factor that distinguishes software development from many other engineering disciplines is the relative ambiguity and instability of the product requirements. This general property of software development projects holds even more for DSS development.

Ambiguity stems from the fact, that users usually express product requirements more or less informally in natural language.

Users usually start to recognize and articulate what they really want, at the time when developers provide them with the first working prototypes of the product. From then
on requirements will never stop to change and to evolve together with the users organization or business. Our experience with systems like WadBOS or EnvironmentExplorer is, that in systems for policy support requirements change even faster and more frequently than in other types of DSS.

General acceptance of the fact that in most projects it is impossible to eliminate requirements ambiguity and continuous change has triggered the search for alternatives for the classic waterfall lifecycle model.

7.3 DSDM / SOMA

For the development of the ElbeDSS we recommend to adopt a development process that is suitable for incremental and iterative object-oriented development. We will give a short summary of the SOMA life-cycle model, which falls in this category. The reader who requires more information is referred to [Graham 94] and [Graham 98].

SOMA is an object-oriented development life cycle model that extends and refines the well-known Dynamic System Development Method (DSDM) [Stapleton 97]. DSDM starts with feasibility and business studies, which are followed by three iterative and overlapping phases: functional model iteration, design & build iteration, implementation.

The SOMA method defines projects as a network of activities that have dependencies but no explicit sequence. Each activity produces a tested result / deliverable. All transitions between activities are guarded by pre-conditions, which is why the model is called a contract driven life cycle model. Management control over the project is ensured by time-box planning that sets rigid limits to prototype iterations.

The main build time-box consists of a number of nested and iterated activities:

- Prototyping
  - Rapid object-oriented analysis, design and programming
  - Testing
- User review
- Consolidation, documentation, identification of reusable components
8 Conclusions

Based on the problem definition for the Elbe IRBM-DSS, in the IT-framework part of the feasibility study, we analyzed the key requirements for such a system and presented an outline of the system architecture. To conclude the IT-framework part, we provide a list of critical success factors, for the development of a pilot version of the Elbe-DSS, assuming an architecture that is similar to WadBOS / MODULUS. This list is derived from our experience with similar development projects, as well as from the requirements discussed in this report.

8.1 10 critical management success factors for an IRBM-DSS pilot:

1. Highly motivated end-users, with both, a visionary as well as pragmatic attitude towards the domain at which the DSS is targeted.

2. Highly motivated development team, with a broad interest in the application domain, DSS development, formal analysis and knowledge representation methods…

3. A small group of highly motivated software engineers with outstanding skills in knowledge engineering, software architecture, user interface design, object-oriented development environments, distributed systems, standards…

4. A small group of highly skilled modelers, with experience in combining various spatial and temporal scales in one model. Furthermore the modelers should have a broad interest for the application domain and should be able to take a pragmatic attitude as well as to achieve compromises, when they need to discuss solutions to technical problems with the software engineers.

5. A DSS architect, perhaps the most difficult role in the process. Like a building architect, this generalist is responsible for the overall vision of the product and must be able to professionally communicate with all participating specialists and stakeholders.

6. A project manager. For small projects this role is sometimes taken by the DSS architect. The project manager should have experience in managing interdisciplinary projects with participants coming from scientific, technical as well as public administration backgrounds.

7. Sufficient time and budget to build a high quality first prototype of the system. A successful prototype will further increase the end-user commitment and eventually will trigger further investments in the DSS development.

8. Early and ongoing end-user involvement in the development.

9. Respect for the role and knowledge of participants from other disciplines than your own.
10. Willingness and ability to take a calculated risk by putting substantial effort in the development of a highly innovative product.

### 8.2 10 critical technical success factors an IRBM-DSS pilot:

1. Use object technology and component based development.

2. Use existing application frameworks for spatial decision support systems.

3. Integrate GIS and database functionality as a component layer.

4. Separate the front-end (user interface) from the underlying tool-, model- and database with clean interfaces.

5. Keep in mind that re-implementation of existing scientific models is sometimes a more efficient and cost-effective ‘integration’ solution compared to extensive ‘wrapping’.

6. Techniques for handling various spatial and temporal scales simultaneously are new. They should be implemented early in the pilot phase, to allow some experimentation.

7. Provide templates and standard interfaces for sub-models (model building blocks).

8. Use standard data formats and protocols for inter-application information exchange (e.g. XML).

9. Test the behavior of the integral model under realistic conditions early and often.

10. For the pilot project maintain a realistic balance between generality and feature completeness.
References


[Engelen et al. 98a] Engelen G., White R., Uljee I., Vulnerability Assessment of Low-Lying Coastal Areas and Small Islands to Climate Change and Sea Level Rise – Phase 2: Case Study St. Lucia, Report to the United nations Environment Program, Caribbean Regional Coordinating Unit, Kingston, Jamaica. RIKS publication, 1998


[Gamma 95] ............... Gamma, E., Helm, R., Johnson, R., Vlissides, J., Design patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Massachusetts, 1995

[Graham 94] ............... Graham, I., Migrating To Object Technology Addison-Wesley, 1994


[Wills 00a] ......... Wills, A., *Frameworks and component based development*,
http://www.trireme.com/trireme/papers/fworks.ps

[Wills 00b] .......... Wills, A., *Component based development*,

Physica 22D, 385-399, 1986
1 Abstract

An important amount of new knowledge and material has been obtained from the many research projects carried out in the EU-DG12 Environment and Climate programme. However, little effort has gone into making this scientific material available as part of practical planning or management tools for public policy makers at the regional level. In a two-year project, MODULUS aims to develop a generic spatial Decision Support System for integrated environmental policy-making at the regional level. Models from past or ongoing EU-projects are integrated that represent the physical, economic and social aspects of land degradation and desertification in Northern Mediterranean watersheds. The individual models operate at very different temporal and geographical scales. At the most detailed temporal scale, processes taking few minutes are represented, and, at the most detailed geographical scale dynamic models running on top of raster-GIS layers are implemented. The MODULUS DSS is developed as a very interactive, transparent and (geo)graphical instrument. It runs on PC machines under Windows NT and makes extensive use of Active X and COM component technology. In order to demonstrate its generic applicability the MODULUS DSS is applied to two case regions: the Argolidas (Greece) and the Marina Baixa (Spain). In the presentation we will dwell on the many lessons learned in trying to bridge the gap between fundamental research and integrated policy making.

2 Introduction

In the past decade, as part of its successive Framework Programmes, the EU has sponsored major research efforts in the domain of land degradation and desertification in Mediterranean watersheds. This research has generated large amounts of data, methodologies and models that have been instrumental in getting a much better
understanding of physical and human causes and effects of these problems in Southern Europe. Based on the work carried out, many of the research projects made ‘scientifically based’ suggestions and recommendations, on ways to slow down, stop or reverse the process of land degradation. However few of the measures proposed found their way to the policy makers and got to be implemented. Hence, from a practical policy making point of view, little use was made of the studies carried out. This is in part due to the fact that much of the research was carried out for scientific reasons and with the purpose of better understanding the processes causing the problems. This type of research tends to be very sectorial and in depth, rather than integral and multi-faceted in nature. It may produce outputs, which are extremely valuable in their narrow disciplines, but too specific and disconnected for the policy maker who needs a broader view on the problems that need a solution. With a view to boast the policy use of material developed for scientific purposes, the MODULUS project poses the following scientific question: Can existing research material, obtained from different complementary research projects, be integrated and made useful to policy makers?

3 Methodological Approach

The methodological approach of MODULUS is clearly focused on the integration of research material that is or will become available from other EU Environment Programme Research Projects. The guiding principle in this integration effort is the fact that the resulting model should be useful for environmental decision and policy making in the Northern Mediterranean generally and in two pilot regions in particular. To that end MODULUS principally builds upon the research results obtained in 4 on-going or past projects: EFEDA, ERMES, ModMED and ARCHAEOMEDES, and to a lesser extent MEDALUS. These projects have been selected among those carried out in the DG 12 Climate and Environment Programme because of the complementarity of the research carried out.

3.1 The EFEDA project

The EFEDA project (See for example: Burke et al., 1998) examined the interaction between the types of land surface and hydrological change associated with desertification and meso-scale climatic impacts. EFEDA developed methods and models to investigate the interaction between the land surface and climate processes within the context of changing surface properties. One of the main outputs of research was the PATTERN ecosystem model, developed to investigate the impact of climatic variability and climatic change on surface and subsurface hydrology and plant ecology (Mulligan, 1996, 1998a). The model is a tightly coupled hydrology and plant growth model developed for semi-arid environments. It incorporates all of the major hydrological fluxes as well as ecological processes of germination, growth, biomass partitioning, death and competition for up to three plant functional types at any one time. It includes a rainfall, storm and weather generator in addition to the tightly coupled hydrology and growth model that forms its core. The model was originally designed as a cellular slope model applied at the 100m$^2$ scale. It was later coupled with a GIS and applied to the whole Guadiana catchment (Castilla La Mancha, central Spain) for analysis of the impact of land use and climatic change on groundwater recharge.
3.2 The ERMES Project

As part of the ERMES project (See for example; Oxley et al., 1998) multi-scalar models have been developed concerning the effect of changing land-use patterns on vegetation cover, erosion risks, water run-off and infiltration, changes in ground water and channel flows, and evapotranspiration. The models developed capture the effects of various processes of water flow and storage as a function of biological activity that operate in the system. These are very small-scale processes involving the water storage capacity and permeability of the soils as a function of the vegetation cover, slope, soil type, aspect and detailed spatial and temporal pattern of rainfall. This allows to represent at more aggregate levels the behavior of successive scales of sub-basins within a catchment, and to represent the complex impacts of land-use on the channel flows at local and large scales, as well as on the recharge rates for ground water, and the stability and fertility of soils within the catchment.

3.3 The ModMED Project

Although the focus of the ModMED project (See for example: Legg et al., 1998; ModMED, 1998) is on the study and modeling of natural vegetation dynamics; hence on the biological and ecological processes characterizing land covered by freely colonizing and growing plant species, there is awareness that the space available to natural vegetation enabling the recovery of spontaneous plant cover is largely dependent on socio-economic dynamics. Where human pressure is increasing, loss of biodiversity and the complete destruction of habitats occur, but, where old types of land-use practices are abandoned, new re-colonization and succession takes place restoring the dominance by shrubs and eventually forest species. In turn, this can lead to the loss of some ancient communities of grass- and shrub-land vegetation with a high biodiversity and conservation value. Although these processes are increasingly understood, the timing of the related landscape changes and the biological mechanisms behind such changes (species dissemination, establishment and competition) still need to be studied to a more satisfactory extent. ModMED addresses these problems by integrating three different levels of ecosystem analysis: individual plant, plant community, and landscape. A modeling environment has been developed consisting of hierarchically nested modules operating at different spatial and temporal scales.

3.4 The ARCHAEOMEDES Project

The ARCHAEOMEDES Project (See for example: Leeuw, 1998) investigated how the changing socio-natural dynamics of Southern Europe (urbanisation, agro-industry, infrastructure) relate to the problems of degradation and desertification in the area. Its central themes were: (1) the definition of the various levels of structuration which drive the dynamics involved, (2) the investigation of the ways in which the dynamics at these various levels articulate, (3) the development of decision-support models of these dynamics which facilitate the investigation of alternative scenarios for the future, and (4) the development of ways in which to map these dynamics in geographical time-space. The project used a combination of fieldwork, analysis, interpretation and modelling focussed on the relationship between the social dynamics responsible for perception, decision-making and action and the natural dynamics which sometimes are subject to human action and at other times trigger and constrain it. The phenomena are investigated at four spatial scales, each representing the
interaction between two levels of structuration ranging from the European to the individual scale.

From the above short project descriptions, it may be clear that there is both complementarity and overlap between the projects selected: the complementarity should permit to come to an integral model covering the essential physical, ecological, economic and social processes related to degradation, while the overlap should permit selecting the most appropriate and compatible model-components among the alternatives available.

4 MODULUS: a Spatial Modeling Tool for Integrated Environmental Policy-making

4.1 Choice of the Pilot Regions
MODULUS is to develop models and a Decision Support System with a high level of generic applicability in the Northern Mediterranean region. Applying the system to two pilot regions will test the adaptability and transferability of the system. Early in the project, the Marina Baixa (Spain) and the Argolid (Greece) were selected based on the following scientific and pragmatic considerations:

1. **Policy relevance.** MODULUS promotes a dynamic and integral approach to water management, desertification and land degradation. From the many sites where EU Environment Programme projects have been carried out, we selected two urbanised coastal watersheds where physical, natural, and socio-economic processes have been studied. Sites also, were the consequences of human practices (crop rotation schemes, irrigation, abandonment of agricultural land, return of natural vegetation cover, tourism and urbanisation) on the aquifer (depletion, pollution, salt intrusion) and on slope dynamics has been documented and modelled.

2. **Data availability.** MODULUS does not include an intensive data acquisition programme, rather it should work to the extent possible with existing data. For both regions selected sufficient high quality data, including GIS data, are readily available to validate and run the integrated models.

3. **Model availability.** MODULUS integrates existing models, methods and knowledge. It allows for the reformulation (aggregation and simplification) of existing models. But, as little as possible new models should be developed. In both sites ERMES and ARCHAEOMEDES have carried out combined research and model development in an effort to pool up to date understanding of the linked natural and socio-economic dynamics. EFEDA and ModMED have been involved in predominantly natural dynamics, and have, as a consequence, been able to come up with more easily ‘portable’ insights and models.
4.2 Policy makers

The development of DSS systems is an expensive and time consuming exercise, only to be undertaken if real problems exist that require a lasting surveillance of the system and regular interventions in order to bring it back on course. The only way to assure that the DSS will be effectively used for policy and planning exercises is to involve the end-users during its development. From the start, they should get the feeling that the end-product is useful in solving their problems in ways that make intuitive sense to them (see for example: Holtzman, 1989). In MODULUS the intended end-users are regional planners and policy makers, defined as: high-level technicians actively involved in the design and evaluation of regional public policies. They perform policy work of a formal/analytic nature in support of the administrator or politically appointed person whose role it is to implement policies.

MODULUS intended to involve the end-users in the project right from the start, but, we ran into two problems, which forced us into rethinking the practical implementation of this aim:

1. In both pilot regions selected, as would have been the case in most other pilot regions, the past scientific work has been carried out for research purposes and not for policy making strictly speaking. Hence, little attention has been paid to involve local or regional policy makers in the work. As a result, MODULUS needed to find and contact its own end-users and convince them of the usefulness of yet another research project. This process took more time and effort than expected.

2. In some northern EU-countries a long tradition exists of involving scientifically trained technicians in the policy preparation phase. Other countries are still in a phase of setting up the institutional frameworks within which these people are or will become active. As a consequence, and in order not to slow down the technical work, we decided to define temporarily a ‘virtual’ policy maker and ‘typical’ policy problems, to be replaced by real policy makers and their policy problems at a later stage. At the same time extra effort was put into the search for ‘life’ policy makers.

4.3 Policy problems, policy levers, and policy indicators

From the fieldwork carried out in the pilot regions as part of ARCHAEOMEDES and ERMES, a preliminary list of policy issues --policy problems, policy levers, policy indicators and policy criteria-- has been established. This list has been helpful in selecting and adapting the models considered useful for integration and in focusing the discussions with the actual policy makers. However, the policy issues mentioned in the preliminary lists very strongly focus on the short term: the actual economic activities, production practices, and immediate water management problems. It is precisely the strength of model-based systems to explore the decision space and search for development alternatives that are beyond the immediate concerns and imagination of stakeholders, politicians and planners. Although most of these alternatives might turn out to be totally unacceptable or undesirable, some will be worth further analysis and evaluation, and all of them are calculated on the basis of a coherent set of assumptions, represented in the same equations and rules of the models. Hence, if we work from the assumption that ‘good’ socio-environmental
policies are to be evaluated from a broader perspective and that they have to increase the level of sustainability of the region, then, we should define indicators and criteria that fit in broad categories including for instance: environmental quality, human welfare, resource availability and cost of policy implementation.

5 ‘Research Models’ versus ‘Models to support Policy Making’

Despite the fact that the terms ‘integrated’ or ‘integral’ model are widespread in the scientific literature, and despite the fact that the use of integrated models is strongly advocated in ‘disciplines’ such as Integrated Assessment (see for example: Gough et al., 1998), very few recipes or procedures for model integration are available from the literature. Hence, model integration seems more an art than a science at this moment.

The integration of models is clearly a multi-criteria and multi-objective problem. We believe that problems need to be solved that deal with the end-use, scientific, and technical aspects of the integration. Although we treat them here separately, it is clearly understood that this sub-division is rather artificial:

5.1 End-use integration

In his review of the models developed as part of projects in the field of desertification under EU framework III and IV, hence, including all of the projects concerned in MODULUS, Mulligan (1998b) discusses a number of important differences between ‘Research’ models and ‘Policy models’. Both are foremost tools developed to simplify reality in order to understand it better. The former are developed to push ahead scientific understanding. They are process oriented, and are developed to build, test and extend research hypotheses. The latter are output oriented and meant to explore, understand, and anticipate the consequences of policy interventions in complex ‘real world’ systems. As to their use there should not be a difference between both types of models, and policy makers should be able to work with the most up to date knowledge about the way systems work. However, there are a number of practical considerations that differentiate them. Mulligan mentions among others:

- Research models are mostly complicated models which are computationally demanding and hence are much more time consuming than what is wanted for interactive, explorative policy exercises;

- Research models are often developed to understand processes that take place at geographical and temporal scales that are not relevant to policy makers. Policy makers will typically be interested in time scales expressed in years, not in seconds or centuries, and spatial scales covering a typical political, administrative, or management unit;

- Research models are typically data demanding. Often they will require field data specifically collected to run the model. Policy models can usually not afford this time and resource consuming luxury and should run on the basis of existing data material;

- Research models are models for experts, generating a type of output that is of immediate interest to the expert. The output is not
compatible with the language and concepts that are of concern to the policy maker. Often, minor adaptations to the output generated by the models and a state of the art graphical user interface could bridge part of this gap, but they are rarely being applied;

- Research models are difficult to validate. This reflects their level of sophistication. Validation is a prerequisite for models if they are to be used in a policy context.

This list reinforces the point made earlier that research models are not automatically usable for policy purposes, and that often important adaptations need to be made to research models in order to use them for policy purposes. For MODULUS, the key end-user requirements of the Decision Support System and its integral model can be summarised as follows (see Chapter 4 by Mulligan and Reaney in Engelen et al., 1999):

(a) *All processes.* The MODULUS model must adequately represent all of the important processes necessary to provide the required output;

(b) *Scientifically proven.* The process descriptions within the MODULUS model should be well known and scientifically proven. It is better to have a well understood, proven but crude process description than an innovative but poorly documented and less proven description. The model results have to be robust, reliable and accurate;

(c) *Scale.* The MODULUS model must operate at a regional scale and must provide information at a sufficient level of spatial detail (resolution) to reflect the scale of variation in the most important environmental and human variables;

(d) *Time horizon.* The MODULUS model must be a dynamic model, operating at time scales and temporal resolutions which are relevant to the policy end-user. It should realistically represent the autonomous dynamics of the system modelled as well as the time scales involved in the policy preparation and implementation phases;

(e) *Routine data.* The MODULUS model must be sufficiently simple to run from routinely measured data. Routinely available data may include data collected by government or intergovernmental agencies such as the EU;

(f) *Scenario based.* The MODULUS model should provide easy to understand scenarios which the user can be taken through. These may be for environmental changes, anthropic impacts, and management options;

(g) *Output centered.* The MODULUS model must be output centered. It will be judged mostly upon the quality of its output and less upon its scientific or technical innovative character. It should provide appropriate results using indicators or variables that directly interface with the policy implementation process rather than more abstract scientific or technical variables;

(h) *Interactive.* The MODULUS model must be fast, responsive and interactive and should cater for a very short attention span. A response time of 15-60 minutes per simulation-run covering a period of 20-30
years should be aimed for. Clever models, fast algorithms, and efficient code will be required to achieve this.

The key trade-offs are between accuracy (of the data and of the model process representation) and simplicity (of models and of data). The model must have sufficient spatial and temporal detail and sufficient model complexity to accurately represent the processes but must achieve this over large areas in a fast and responsive manner with a minimum of data. From the above, it will be clear that this is not automatically achieved on the basis of research models, rather that important adaptations to the models are required before they are effectively integrated. In this respect, MODULUS has developed solutions at three levels: straightforward integration when the model represents the process adequately and efficiently, and when the interactions with other component models is possible; models are adapted if only minor repairs or reformulations to the model, its algorithms or code are required to have it perform its tasks more appropriately; finally rebuilding is considered when the model need major repair and adaptation in order for it to fit in the modeling scheme.

5.2 Scientific integration

The four projects described earlier in this paper where deliberately selected on the basis of the complementarity of the research carried out. And, from a preliminary analysis it was concluded that the potential for integration was real. From the 4 projects models were available in different stages of development: some models where fully finished and had been validated and tested against real world data, while others were still in an early development phase. The models were evaluated on their conceptual and technical merits as well as their scientific novelty. A ‘typical’ scientific evaluation would also have considered the performance of the models in terms of realistically representing the processes for which they are developed, and their capacity of generating validatable output. However, most of the models available from the 4 projects where not sufficiently operational to permit the latter type of analysis.

As a result most of the evaluation has been focussed on the role models could fulfil as component sub-models in the integrated context of the MODULUS model. The following criteria where taken into consideration for the selection and evaluation:

- **Time scales and temporal dynamics.** Only dynamic models are considered. Models have to span a strategic time horizon (10-20 years) and operate at appropriate (simulation) time steps reflecting the real world processes and decision-making time frame (1day-1year). With a view of simplifying or aggregating the model, the effect of increasing or decreasing the time step on the performance of the model is a criterion;

- **Spatial resolution and spatial dynamics.** Only spatial models or models that can be spatialised are considered. Models have to be applicable to a relatively large regional entity and operate at an appropriate spatial resolution reflecting realistically the real world processes, the spatial variability across the region, and the individual geographical entities subject to decision and policy making (1ha-1000km²). With a view of simplifying or aggregating
the model, the effect of increasing or decreasing the spatial resolution on the performance of the model is a criterion;

- **Compatibility of scientific paradigms.** Models are considered that from a scientific/operational point of view can be integrated. Thus, the basic assumptions and constraints on which the models are developed are evaluated. Most of the models available in MODULUS are spatial, dynamic, non-equilibrium models that are solved by means of simulation. Hence, little problems with clashing scientific paradigms were detected;

- **Models that fit the total integration scheme.** Models are considered that fulfil a task within the MODULUS integration scheme which is not dealt with by any other (sub-)model. They compute a subset of the total set of state-variables and exchange the necessary information among one another at the right temporal and spatial scale during the calculation;

- **Level of sophistication.** The models considered are in most cases simplified version of ‘the ultimate’ or ‘the best available’ models. In order to fit in the integrated scheme, and to be at the right level of abstraction, models need to be simplified and need to be stripped of details that are not directly relevant to the problems at stake. The value of the integral model is as good as the weakest element in the web of linked sub-models. Hence, it is better to improve this weakest element rather than to add details to the other sub-models.

The analysis of the models against the selection criteria lead us to conclude that an integral MODULUS model consisting of the models mentioned, would be a grid based model running at a spatial resolution of 1 ha (100 by 100 meters) and at a temporal resolution in the order of 1 week to 1 month. The output generated with this model would suffice for most relevant policy questions in both case regions. A spatial resolution of 1 ha would be appropriate for the majority of the processes represented. A large amount of GIS data are available at this resolution, and it allows for the inclusion of models running on irregular (administrative) areas if the borders of these areas are redrawn to coincide with the edges of cells. The errors thus made are minimal. As for the temporal resolution, the choice of a monthly or weekly time-step is not appropriate for a number of the models. In particular, KCL’s PATTERN model (see for more details Chapter 4 by Mulligan and Reaney in Engelen et al., 1999) requires a much finer time step (minutes or hours). As a result, the decision was made to develop a model running at an hourly time step. While the simulation is stepping through time, sub-models are invoked as required. Information that needs to be exchanged is aggregated over days, weeks or months as required.

### 5.3 Technical Integration

As most research models nowadays are also computer models, the problem of technical integration is very much a hard- and software problem. From a computer science point of view, integration of models has become very much a problem of software component integration. Software components are pieces of software that are designed for re-use: ‘a coherent package of software artifacts that can be
independently developed and delivered as a unit and that can be composed, unchanged, with the other components to build something larger’ (D’Souza and Wills, 1999). The ideal software-component is platform independent and can be plugged into a software system like a plug into a socket.

Clearly, the typical user of a modelling shell or Decision Support System (DSS) would be served best if he could compose, exchange and re-arrange sub-models as easily as Lego building blocks and develop his model from a set of exchangeable and interchangeable Model Building Blocks (MBB). Such Model Building Blocks can be more or less complete models varying from simple mathematical operators, such as Multipliers or Adders, to rather sophisticated and nearly complete models consisting of coupled mathematical equations performing a number of sophisticated calculations. A Model Building Block represents a part of a model: an action or process. MBB’s may simply represent sources of information (i.e. entered from file or by the user), other will transform information as it passes through them, and still other will simply serve to communicate, in a synthetic manner, the outputs of the model to the user.

![Diagram](image)

**Figure 1:** The MODULUS simulation engine works with standard COM ActiveX/MODULUS components. Each of the existing models is encapsulated in a ‘wrapper’ which makes them look like ActiveX/MODULUS software components. For more details see Chapter 5 by Engelen et al. in Engelen et al., 1999.

We realize that MODULUS will not develop the ultimate methodology or library containing a set of easily pluggable Model Building Blocks. More development time would be required to achieve this goal. But in MODULUS the question of re-use is posed: from different EC projects partners have their existing models, written as monolithic applications in whatever programming language they master. Rather than re-coding this material in yet another programming language to develop yet another monolithic application, MODULUS has chosen to integrate the material on the basis of a state of the art component technology. The constraints of time and budget, the availability of ready to use modeling material, as well as the objective to produce a running system applied to two case regions made us decide to attack the problem from a rather practical angle. The focus of the work carried out under the heading ‘technical integration’ therefore is on the evaluation of the usability of the existing component...
technologies, rather than on the development of new standards, functional specifications and technical designs.

6 The MODULUS model

Screening the available models against the End-use integration and Scientific integration criteria has allowed deciding on a scheme for an integral MODULUS model, consisting of component sub-models (see Figure 2). This integration scheme has the great merit of covering an important part of both the natural and the socio-economic system. From an initial analyses it was concluded that sufficient data are available in both pilot regions to run the integral model, and that the sub-models produce and exchange the appropriate information required in the integration scheme. This in itself is a remarkable result, since each of the models has been developed within different research contexts and with different purposes in mind. However, one should not interpret this to mean that model integration is a straightforward and easy process. On the contrary, it requires a very careful examination of every aspect of every sub-model that is affected by the integration. It may be clear that this is the work of specialists working in a team, and it is to wonder whether this process can ever be made into an operational procedure not requiring the involvement of the model developers.

Each of the models included in the scheme is dynamic and spatial. The typical spatial resolution at which the integral model runs is the 1ha grid. The role of the individual models in the connection scheme can be summarised as follows:

- **Weather:** (EFEDA, PATTERN Weather & Storms model, available as a C++ MBB). This model runs daily. It calculates for each day the time of sunrise and sunset and the average solar radiation map at the top of the atmosphere between sunrise and sunset. The average solar radiation is then corrected for the slope and aspect of each cell. The average temperature per cell is updated monthly. Further the model generates for the day a detailed time
series (expressed in minutes and bucket-tip times) for precipitation for the study areas based on data from at least 1 AWS weather station.

**Figure 3:** Air temperature in the Argolid at ground level. The mean monthly temperature at sea level is derived from a 30-year climate scenario. It is spatialised and corrected for altitude and local deviation from the mean temperature as measured by local weather stations.

- **Hydrology & Slope processes:** (EFEDA, PATTERN Hydrology & Slope processes model, available as a C++ MBB). This model runs daily, but integrates internally over bucket-tip times. This model deals with the soil hydraulic properties and calculates the water budget. It calculates the interception, infiltration, soil moisture, transpiration, soil evaporation, overland flow, surface recharge, and erosion.

**Figure 4:** Soil moisture in the Argolid in the winter (end of December)
• **Natural vegetation:** (ModMED, RBCLM2 Community model, available as a PROLOG MBB). This model runs once a month. It represents the processes of growth, succession and decline of the natural vegetation at the community level. It calculates the leaf area index, the vegetation cover fraction, and the rooting depth. The natural vegetation model is a rule based model, applied to each individual cell of the case regions. It is supplemented with a cellular seed diffusion model, which produces a seed biomass maps, which links the community level cells at the landscape level.

• **Crop Growth:** (EFEDA, PATTERN Plant Growth model, available as a C++ MBB). This model runs daily. It represents the processes of growth of commercial crops and calculates the leaf biomass, root biomass, leaf area index and the vegetation cover fraction.

• **Aquifer:** (ARCHAEOAMEDES, 2 versions of the aquifer model are retained: the Agricultural University of Athens-ModFlow model and the IERC-Aquifer model. The ModFlow model is available as a FORTRAN MBB, while the IERC model is available as a POWER BASIC MBB). Due to the very complex and discontinuous nature of the aquifer in the Marina Baixa, the aquifer model is only applied in the Argolid region. This model represents the depletion, recharge and pollution of the aquifer. It calculates the aquifer water height, salt concentration and the fluxes between cells. The ModFlow-aquifer model runs monthly and on a spatial resolution of 500 by 500 m. The IERC-Aquifer model is intended to run daily on a 1ha or 1km resolution.
Figure 6: The main watershed modelled in the Argolid and the location of the Aquifer within. The Aquifer is modelled by means of a ModFlow model at a 500 by 500m grid, which runs on a monthly basis. The volume of the aquifer is represented.

- **Catchment**: (ERMES, Catchment model, available as a POWER BASIC MBB) This model runs on a daily basis. It represents the river, canal, and water reservoir system, and the water quality of the surface water. It calculates the river flows per stream order, the sinkhole flows, the catchment recharge flows, and the river PO₄ and NO₃. The model runs on irregular shaped, natural defined areas –the catchments and sub-catchments.

Figure 7: Catchment stream orders for the Argolid watershed.

- **Crop type decision**: (ARCHAEOMEDES, Decision making model, available as a POWER BASIC MBB). This model runs on a yearly basis. It is a rule based model representing the crop-choices made by farmers as a function of changing physical, socio-economic and institutional conditions and circumstances. It is applied to each 1ha cell and calculates the crop type, crop water requirements, water source, presence of boreholes, borehole depth, pumping capacity, air mixer deployment and the total yearly long term exploitation costs.
• **Pumping**: (Extracted from work done in ARCHAEOMEDES by IERC, available as a POWER BASIC MBB). This model runs twice daily. It is a rule based model representing the farmers decision to switch on the water pumps and start the irrigation. It is applied to each 1ha cell and calculates the pump status, volume to be pumped, extraction from the canal, volume of frost water, frost water salt concentration, irrigation water volume, irrigation water salt concentration, and the total yearly short term exploitation costs.

![Figure 8: Crop types in the Argolid.](image)

![Figure 9: User-interface of the MODULUS system. The user gets access to the individual sub-models (Model Building Blocks) by means of the systems diagram (shown in bottom left). He can run simulations and select any combination of maps on the display.](image)
• **Land Use**: (GEONAMICA Constrained Cellular Automata model, available as a C++ MBB). This model runs yearly. It is a cellular automata based model which allocates in a detailed manner (1ha grid) the land claims resulting from demographic changes, as well as the dynamics in the agricultural and non-agricultural part of the economy. The allocation methodology will take into consideration the activity specific attractiveness of cells in terms of their suitability, zoning regulations and accessibility to the road transportation infrastructure.

The model presented heavily relies on GIS data. As an input it requires some 25 GIS layers (raster maps, mostly at 100 meter resolution), and it updates at every simulation time step some 50 output maps. All the output maps are simultaneously available to the user. Hence, during the simulation he can watch the evolution of the modelled region by means of any combination of the 50 mapped variables. Some of the output maps represent a final output variable of the integral model, but most maps are generated or updated by an MBB to serve as an input to another MBB.

Although it is expected that the integration will eventually lead to a **Scientifically acceptable model**, this does not mean that the **End-use usability** of the integral instrument is automatically guaranteed. The first tests performed showed that a single run of the integral model for the entire Argolid region, consisting of 239385 * 1 ha cells, took nearly 12 days. In this test the sub-models were running at the appropriate time step (1 minute – 1 year), for a period of 30 years, and at the spatial resolution, which is considered minimally required for the soil and slope models (namely the 100 m grid). It goes without saying that a model, which takes this much time to perform a single simulation run, is not a very practical tool for policy making. It looses all of its explorative capabilities as well as its role as a communication tool. Since then, a lot of effort has been put into reducing the execution time. A simulation run now takes some 2 hours.

## 7 Using the MODULUS model and Decision Support System

The use, role and usefulness of models and Decision Support Systems in policy making has been the subject of a rich scientific debate and literature, and extreme views have their advocates. In this paper we do not have the room to dwell on this discussion. However, inherent in the aims of the MODULUS project, is the somewhat positivistic view that the use of scientific models can improve the policy making process. More in particular, MODULUS adheres to the view that **better informed** policy makers are **better equipped** to make **better** policies that bring the systems they are to manage on a path towards sustainability. Thus, the prime role of the models and the Decision Support Systems is awareness building and education, rather than the decision-making act itself. The models therefore should give an adequate and truthful representation of the real world system, and the policy maker should be enabled to work with the models in a well-structured, well-guided and flexible manner. A well-designed **user-friendly interface** should enable to structure the policy exercises carried out with the model. The same interface should increase the transparency of the model and the DSS as much as possible: at any point in time, the user should have access to the background information required to understand the processes in the model he is working with and the numbers they generate. Without this information, the model
becomes a black box and no learning takes place. The user can try out policy interventions in his system (which he selects from a predefined Policy Options Window). He can test the robustness of the system and his policy interventions if the system is subjected to scenarios (which he can select from a predefined Scenarios Window). Finally, he can see how his policy interventions and/or scenarios in terms of indicators (which are shown in an Indicator Window) affect the system. This way the impact of interventions can be tested and tuned in an interactive session between the policy maker and the modelled system and catastrophes can perhaps be avoided in the real system. Such an explorative approach, we believe, contributes to the design of actions to pilot the system past the worst and hopefully towards the most desirable future possible.

Acknowledgements

This research is supported by the Commission of the European Union Directorate General XII Environment (IV) Framework Climatology and Natural Hazards Programme contract ENV4-CT97-0685 (DG12-EHKN).

The consortium carrying out the work described consists of:
Research Institute for Knowledge Systems b.v., P.O. Box 463, 6200 AL Maastricht, The Netherlands.
e-mail: guy@riks.nl. Guy Engelen (co-ordinator), Maarten van der Meulen, Bernhard Hahn.
Department of Geography, King’s College London, Strand, London WC2R 2LS, UK.
e-mail: mark.mulligan@kcl.ac.uk. John Thornes, Mark Mulligan, Sim Reaney.
International Ecotechnology Research Centre, Cranfield University, Cranfield Bedford MK43 0AL, UK.
e-mail: p.m.allen@cranfield.ac.uk. Peter M. Allen, Tim Oxley, Condelynia Blatsou, Macarena Mata-Porra, Spiros Kahrmanis, Panagiotis Giannoulopoulos.
Università di Napoli ‘Federico II’, Istituto di Botanica Generali e Sistemica, Facoltà di Agraria, Via Università 100, I-80055 Portici (Napoli), Italy. e-mail: mazzolen@cds.unina.it. Stefano Mazzoleni, Adele Cappola, Brian McIntosh.
The Spatial Modelling Centre on Human Dimensions of Environmental Change, Rymduaset, Osterleden 15, S-981 28 Kiruna, Sweden. e-mail: kultso@strix.its.uu.se. Einar Holm, Nick Winder.

References

Holtzman, ‘Intelligent Decision Systems’, Addison-Wesley, Reading, Massachusetts, USA, 1989
Legg C., Papanastasis V.P., Heathfield D., Arianoutsou M., Kelly A., Meutzelfeldt R.,
and Mazzoleni S., ‘Modelling the impact of grazing on vegetation in the
Mediterranean: the approach of the ModMED project.’ in: Papanastasis V.P. and
Peter D. (eds.) ‘Ecological basis of livestock grazing in Mediterranean ecosystems’,
Proceedings of the International Workshop held in Thessaloniki, Greece, pp.189-199,
1998
Journal of Modelling and Assessment. Submitted 1999
Mulligan M., ‘Modelling the geomorphological impact of climatic variability and extreme
events in a semi-arid environment’. Geomorphology, 24:1, p.59-89, 1998a
Mulligan M., ‘Modelling Desertification’, EU Concerted Action on Mediterranean
Desertification. Thematic Report, King’s College London, 1998b
Mulligan M., ‘Modelling the complexity of landscape response to climatic variability in
ModMED, ‘ModMED, Modelling Vegetation Dynamics and Degradation in
Mediterranean Ecosystems’, Final report, EU-DG12 (Contract ENV4-CT95-0139),
1998
Oxley T., Allen P., and Lemon M., ‘The integration of dynamic socio-economic and
biophysical models: A model of slope dynamics in the Marina Baixa’. IERC,
Cranfield. Final report for the ERMES II Project EU-DG12 (ENV4-CT95-0181),
1998
D’Souza D. F., Cameron Wills D., ‘Objects, Components, and Frameworks with UML.
The Catalysis Approach’, Addison-Wesley, Reading, Massachusetts, USA, 1999
Uljee I., Engelen G., and White R., ‘Rapid Assessment Module for Coastal Zone
Management. RamCo Demo Guide’, Coastal Zone Management Centre, National
White R. and Engelen G., ‘Cellular Automata as the Basis of Integrated Dynamic
B. GREAT-ER

1 The GREATER software concept

To show an example of data and model handling of water quality assessment and dealing with uncertainty of models and data a short summary of the GREAT-ER software concept is given here. A more detailed description can be found elsewhere [Matthies et al. 97], [Feijtel et al. 97], [Boeije et al. 97], [Boeije 99]. The GREAT-ER manual can be ordered on request [ECETOC 99].

The European project GREAT-ER (Geo-referenced Regional Exposure Assessment Tool for European Rivers) was launched and carried out as an international effort to develop and validate the basic software and data methodology for the geo-referenced exposure assessment of aquatic systems [Matthies et al. 97], [Feijtel et al. 97], [Boeije et al. 97]. Pilot study areas in the UK, Germany, Italy and Belgium were selected and spatial and non-spatial data sets for down-the-drain chemicals (Boron, LAS) and intermediates were collated and integrated on a Windows NT platform using the desktop GIS ArcView. Chemical emission and waste water pathway as well as hydrological flow data are processed to obtain a consistent spatial data set for the catchment under investigation.

In general GREAT-ER follows a modular approach (Fig 8.2.1). The whole river network is segmented into river stretches, which are related to geographical units (GU) or subcatchments. A hydrological model is used to estimate the magnitude of river flow and flow velocities at ungauged river reaches. Waste water pathway and chemical fate models are connected linked together to estimate the impact from land use and human activities from each GU. The implemented models (Fig 8.2.2) can run in different complexity modes depending on the available data on the chemical and environmental properties [Trapp and Matthies 98], [Boeije 99].

GREAT-ER was designed for and implemented on a Windows NT platform. ArcView® as a desktop GIS system was customized as a graphic user interface (GUI). The models used (Fig 8.2.2) are implemented in C and run outside the ArcView® system under the control of the GUI. This opens the possibility to update the model software or to integrate new models based on defined data and information interfaces. Regarding the classification of model and data integration and implemented GIS technology done in chapter 5 in the current GREAT-ER can be specified as a system model with local databases and GIS functionality of a commercial system.
Fig. 8.2.1: Modular approach of GREAT-ER.

Fig 8.2.2: simulation models integrated in GREAT-ER and their interdependencies.

Much additional functionality was integrated to allow a user-friendly operation of the system. Two modes of the GUI were developed, an ‘easy-to-use’ mode mainly for novices and occasional users. The ‘expert mode’ provides the complete set of spatial and non-spatial operations. The user defines a scenario by the input of spatial and non-spatial data, e.g. substance properties.

GREAT-ER calculates the concentrations in the wastewater effluents and the river segments and their probability distributions according to the input parameter variability and uncertainty. This possibility to deal with the variability and uncertainty of the input data (Fig 8.2.3) is very important. Long-term statistics of the hydrological flow are provided for the gauging stations. Moreover, for all input parameters a frequency distribution instead of a deterministic value can be used. With Monte-Carlo simulations the probability distribution of the resulting concentrations as a function of the input data variability and uncertainty is determined. Temporal probability distributions represent stochastic values for simulated concentrations in all contaminated segments. For each segment mean, variance and any percentile, e.g. the 90th percentile can be plotted and visualised. In terms of risk assessment, the
maximum concentrations are expected at the discharge sites. These concentrations at the beginning of a contaminated segment are called PEC\textsubscript{initial}. Another important information is the spatial distribution of the mean concentrations in the whole catchment, which is called PEC\textsubscript{catchment} \cite{Boeije et al. 2000}. It can be compared to the PEC\textsubscript{regional}, which is routinely used in the EU risk assessment scheme for the evaluation of background concentrations \cite{EC 96}.

Fig. 8.2.3: Stochastic simulation capabilities of GREAT-ER to deal with uncertainty and variability of input data.

The GREAT-ER software was developed and tested for various catchments in the UK (Yorkshire), Italy (Lambro), Germany (Itter, Rhine, Rur) \cite{Koormann et al. 98}, \cite{Schulze et al. 99} and Belgium (Rupel). Comprehensive monitoring studies over more than two years were carried out and over 2000 river water and over 600 waste water treatment effluent samples were analysed for the surfactant LAS and Boron. The measured concentration profiles from the monitoring programs were compared to the simulation results. The accuracy of the prediction is for most of the investigated catchments below a factor of three and shows the general applicability of GREAT-ER \cite{Matthies et al. 2000}. It is proposed to integrate more catchments located all over Europe and to extend the approach to other environmental media (e.g. soil, estuaries, air) and intermedia mass exchange processes (e.g. run-off, deposition).
Fig. 8.2.4: Mean simulated LAS concentration in the Calder (UK Yorkshire) catchment under current conditions (top) and under the scenario of improved technology (activated sludge plus primary settler) for three selected waste water treatment plants (bottom).

GREAT-ER is not a DSS system as this was not the purpose it was developed for. But it is already a tool for water quality management purposes and in this case it can be used to estimate various effects of changed conditions in the river basin by formulation of possible scenarios. For example the effects of changed emission rates (e.g. by reduced consumption of household chemicals) or improved wastewater treatment plants on the simulated concentrations in the river system can be calculated (Fig. 8.2.4). Under the boundary condition of limited capital this may help the water manager to select the highest effective measure concerning the reduction of water pollution.
References


