Software Architecture Document (SAD)

Appendix D

Contents

Figures

Tables

1 INTRODUCTION

This document constitutes one of the deliverables of the Inception Phase. According to the Terms of Reference (TOR) /1/ the document shall focus on the overall system architecture, which will be subject to client approval before proceeding further into the subsequent development cycles.

1.1 Scope

The scope of the software architecture document is to provide a design and technical specification of the overall system architecture of the NB DSS system. This implies that the document does not provide a detailed analysis or design of the components constituting the system.

Software architecture is a term that conveys different connotations for different people. Some people might picture a very detailed blue print of the inner mechanisms of a system others very high-level descriptions of the technologies and concepts ("patterns") applied. This software document aims at following the definitions provided by the Software Engineering Institute (SEI).

> *Software architecture is a sketchy map of the system. Software architecture describes the coarse grain components (usually describes the computation) of the system. The connectors between these components describe the communication, which are explicit and pictured in a relatively detailed way. In the implementation phase, the coarse components are refined into "actual components", e.g., classes and objects. In the object-oriented field, the connectors are usually implemented as interfaces.*
>
> *Wahab Ahmen, www.sei.cmu.edu/architecture/start/community.cfm*

Given this definition, the document demonstrates to the Client how the system will be decomposed into components and how these components interact with each other to provide the required functionality outlined in the Software Requirement Specification (SRS) /2/. The design of the individual components will be conducted at the beginning of the respective cycle for each release of the NB DSS system. However, in order to accommodate NBI's request for fairly detailed information concerning model integration certain aspects have been described in a level of detail similar to a implementation design.

1.2 How to read this document

This document aims at describing, at a high level, how the system defined in the SRS /2/ will be designed. The reader therefore will gain a better understanding of the descriptions in this document if the SRS is read beforehand.

The SRS focussed on the functional and non-functional requirements stated in the /1/ and on the "4+1" use cases, see /3/. Based on this information the SRS established (see /2/) functional components – or logical groups of functionality. The software architecture presented in this document extends the functional components into software components.

The document presents the software architecture in the following order:

- Chapter 1 Introduction, this chapter provides a brief introduction including scope of the document and list of terminology.
- Chapter 2 Viewpoint outlines the viewpoints addressed in the document, i.e. the way the descriptions and discussions of the system are organized.
- Chapter 3 Architectural Overview provides an overview of the system and identifies the software components that will constitute the system.
- Chapter 4 Software components describes how the functional and non-functional requirements, the "4+1" use cases and the functional components established in /1/ have been used to defined the software components
- Chapters 5 through 7 provide the architecture descriptions for the following 4 viewpoints – logical view, system use, implementation and infrastructure aspects.

The document makes use of various types of diagrams in order to best possible convey the description of the system. Some of these diagrams are formatted according to the UML specifications and such diagrams are marked with a "UML" indication in the figure caption. The use of UML is briefly described in Section 1.5.

The document refers at several places to documents referenced in Chapter 1.3. Whenever text is directly quoted from these documents, it will be shown in italics.

A document reference is made using a /n/ notation where the 'n' indicates the reference number. I.e. /1/ refers to the Terms of Reference.

1.3 References

/1/ The Contract entered between DHI and NBI. /DHI and NBI, May 2009

/2/ The Software Requirement Specification (Inception Report, Appendix C). /DHI, December 2009

/3/ "Essential Technical Diagrams and key information for assessing design" as communicated to DHI during the Cairo workshop.

/4/ The "4+1" Use cases as communicated to DHI in e-mail on the 10th of September 2009.

/5/ Design Patterns for Relational Databases by Eugenia Stathopoulou, Panos Vassiliadis, University of Ioannina.

1.4 Terminology

The Terminology used in this document is listed in Table 1.1 below.

Table 1.1 Terminology

Name	Description
Actor	In UML an actor specifies a role played by a user or any other system that interacts with the subject.
Adapter	In the context of NB DSS a piece of software that makes it possible for the DSS Front-end and Model Tools to integrate. The name originates from the software engineering Adapter pattern. The Adapter pattern translates one interface of a component into a compatible one (see e.g http://en.wikipedia.org/wiki/Adapter_pattern). In this case part of the model tool interface is translated into an interface that the NB DSS can leverage
BLOB	Binary Large Object. In a database BLOB fields can be added, changed, and deleted. However, they cannot be searched and manipulated with standard database commands.
Catchment	A catchment is an area where water is collected by the natural landscape. In a catchment, all rain and run-off water eventually flows to a creek, river, lake or ocean, or into the groundwater system.
CBA	Cost-benefit analysis.
Component, functional	A logical grouping of related functionality.
Component, software	An identifiable part of a larger program. In software engineering a component is a reusable program building block that can be combined with other components in the same or other computers in a distributed network to form an application.
CRUD	Create, Read, Update and Delete – the canonical database interaction types.
DAO	Data Access Object, typically referring to a programming pattern for interaction with a database. See http://en.wikipedia.org/wiki/Data_access_object
DSS	Decision Support System.
DSS Front-end	The custom Windows application developed to the NB DSS. Serves as the front-end (entry point) of the NB DSS.
Ensemble	In this context an Ensemble is equivalent to a group of time series typically used as inputs for an ensemble simulation
Ensemble modelling	Refer to applying mathematical model with ensembles of time series as input
Ensemble scenario	Refer to defining scenarios using ensembles of time series as input
GUI	Graphical user interface, synonymous with UI.
Hydraulic Model, Hydrological Model, Water balance and allocation model	Mathematical models with a more specific scientific focus (see also Mathematical Model)
Hydro Objects	Hydro objects are entities related to modelling of water related processes, e.g. reservoirs and irrigation schemes.

Name	Description
IDE-style user interface	A UI style where multiple child windows reside under a single parent window. Child windows are dockable and collapsible and can be tabbed and resized.
Interface	A protocol or interface is what or how unrelated objects use to communicate with each other.
IS	Information System.
Layer	A logical part of an application providing a set of specific functionalities.
Leaf use case	The term leaf use case is usen in Chapter 3 to identify the most specific use cases defined in the “4+1” use cases.
Management Scenario	Describes the present or possible future conditions resulting from alternative water resources management and development strategies or changed climatic conditions (See also Scenario)
Mathematical Model	A set of mathematical expressions and logical statements combined in order to simulate certain characteristics of the natural system. The “Model” in this document refers to a Mathematical Model. A Mathematical model may also consist of a number of linked models.
MCA	Multi-criteria analysis.
Member	A function or method in a software class or interface description.
Model Setup	A model setup is a definition of all data required for a simulation, including input data, configuration of the physical infrastructure, management strategies, such as reservoir operation rules, and all output specifications.
Model Tool	A generalised mathematical model such as MIKE11 and MIKE BASIN. May include proprietary as well as public domain systems.
Model Tool Engine	The engine is considered the full suite of executables and Dynamic Link Libraries (dll) controlling the simulation of system behaviour, including pre- and post-processing of Model Data and parameters and solving of the underlying mathematical equations.
Model Tool User Interface (UI)	The Model Tool UI gives access to the functionality of the Model Tool Engine.
Modelling System	A suite of Model Tools.
Modelling System UI	The Modelling System UI gives access to the functionality of the Modelling System.
Module	A module takes care of a specific well defined functionality within the DSS front-end. An example of a module is the Timeseries Manager
NB DSS	The complete DSS system delivered to the Nile Basin Initiative.
Parameter	A parameter is a quantity characterizing a (physical or conceptual) property of a system. Within the context of hydrologic mathematical models, examples of a parameter are hydraulic conductivity, channel roughness/resistance, storage delay time, and time of concentration. A parameter may or may not be constant in time.
Pivoting pattern	This pattern describes a flexible way of modelling attribute-pair values in a database schema, see /5/
PostGIS	Extension to PostgreSQL for handling spatial data.

Name	Description
PostgreSQL	Object-oriented relational database management system.
PROJ4	Cartographic Projections library
Reference Scenario	Describes past conditions and is used for comparing impacts of present and future water resources development and management strategies.
Scenario	A Model Setup or a set of linked models designed to analyse a specific combination of water resources development strategies and water resources management strategies. A scenario is typically used for simulation of conditions for ground water, surface water, water quality, water allocation etc. under a given combination of water resources development strategies, water resources management strategies and climatic conditions. Two principally different scenarios exist, namely *reference scenarios* and *management scenarios*.
Schema pattern	The Schema pattern is data model pattern where an entity is modelled through a type identifier and an aggregated field. The aggregated field can be validated through a type specific schema stored elsewhere in the database. This pattern can be used for modelling entities that not necessarily are known at system construction time. It makes it possible to extend a live system with new types of entities. The pattern is not an officially known pattern; but a pattern established by the Consultant
Simulation	A time-varying description of certain behaviour of the natural system as computed by the mathematical model. A simulation will produce outputs referred to as Simulation Results.
SRTEXT	An OpenGIS Consortium standard for representing of a spatial reference system, based on WKT
Study	A logical grouping of data, typically stored within a dedicated data compartment.
System	Any structure, device, scheme or procedure, real or abstract, that interrelates in a given time reference, an input, cause, or stimulus, of matter, energy, or information and an output, effect, or response of information, energy or matter.
The natural system	The entirety of the socio-economic, environmental/ecological and water resources (hydrological) sub-systems and includes the hydrological cycle or parts of it as we currently conceive it.
ThinkGeo	In this document shorthand for the MapSuite Desktop product from the Company called ThinkGeo. The product is map control for rendering of spatial data.
Tier	Hardware where a specific layer is deployed.
UI	User Interface.
UI object	A software object (instance of a class) residing in the user interface part of the software code.
UML	Unified Modelling Language – A language for the specification, visualization, construction, and documentation of the components of software systems.

Name	Description
Use case	A use case is a description of how an actor will use a software code. It describes a task or a series of tasks that an actor will accomplish using the software, and includes the responses of the software to the actions.
Variable	A variable is a characteristic of the natural system that may be measured and that may assume different numerical values at different times. It can be a series of inputs and outputs from the model, but also a description of conditions in some component of the model (state of the system).
Water Resources Development Strategy	A set of water management options that include structural changes to the existing system such as the construction or modification of reservoirs and irrigation schemes.
Water Resources Management Strategy	A set of water management options that include operational changes to the system such as changes of operation rules for reservoirs.
WKB	Well-known binary, an OpenGIS Consortium standard for GIS geometry specification
WKT	Well-known text, an OpenGIS Consortium standard for GIS geometry specification

1.5 Use of Diagrams

In this document, the notation describing the architecture is based on UML component diagrams, UML sequence diagrams, conceptual data model diagrams expressed as UML class diagrams and Gane-Sarson data flow diagrams. These diagrams will be further detailed into UML class diagrams during the analysis and design stages of the three planned development cycles.

1.5.1 UML Component Diagrams

This document uses UML 2.0 compliant *component diagrams* to describe the various NB DSS components. The component diagrams strive to illustrate a high-level logical view of the NB DSS by showing the structural relationships (interactions) between the various NB DSS components.

A component defines its behaviour in terms of *provided* and *required* interfaces. The *provided interfaces* are services offered to other components of the system, shown using the "lollipop" symbol (see Figure 1.1). *Required interfaces* are services that the component expects from its environment (i.e., other components that it interacts with), shown using the "socket" symbol (see Figure 1.1).

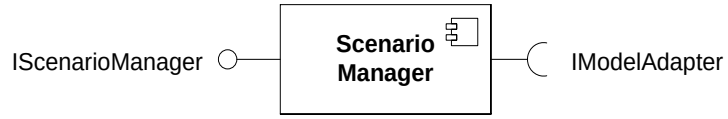

Figure 1.1 Required and provided interfaces (UML)

Components can be connected by usage dependencies. A usage dependency is a relationship where one component requires another component for its full implementation. A usage dependency is shown as a dashed arrow with an arrowhead pointing from the dependent component to the one of which it is dependent (see Figure 1.2).

Figure 1.2 Usage dependencies (UML)

1.5.2 UML Sequence Diagrams

The document uses UML 2.0 compliant *sequence diagrams* to describe the interactions and dataflow *between* components. These diagrams illustrate the interactions between components in the sequential order that those interactions occur. Time passes from top to bottom. The interaction starts near the top of the diagram and ends at the bottom.

In a sequence diagram, a component is displayed as an annotated rectangle. Below the component, its lifeline extends for as long as the target exists. The lifeline is displayed as a vertical dashed line.

When a target sends a message to another target, it is shown as an arrow between their lifelines. A closed and filled arrowhead signifies that the message is sent synchronously whilst an open arrowhead is used to indicate that a message is sent asynchronously. Return values are shown using a dashed arrow from receiver to sender.

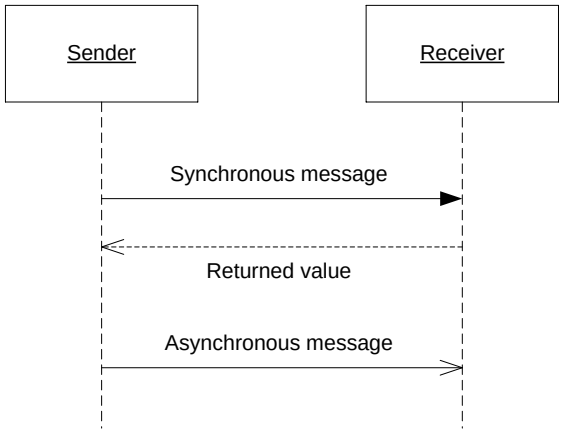

Figure 1.3 Sequence diagrams (UML)

1.5.3 Gane-Sarson Data Flow Diagrams

The document uses compliant Gane-Sarson diagrams to express data flows between processes in the system. These diagrams show the storage, exchange, and alteration of data and resources throughout the diagram.

A Gane-Sarson diagram is built from just 4 stereotypes as shown below in Figure 1.4

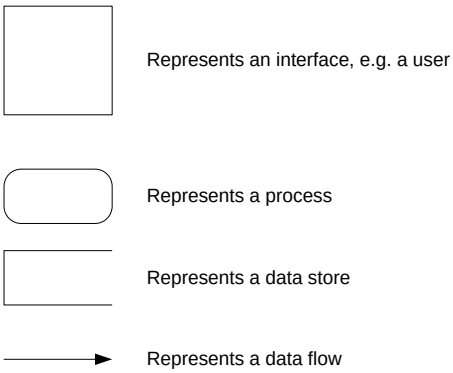

Figure 1.4 Gane-Sarson diagram stereotypes

A self-explanatory example is shown in Figure 1.5.

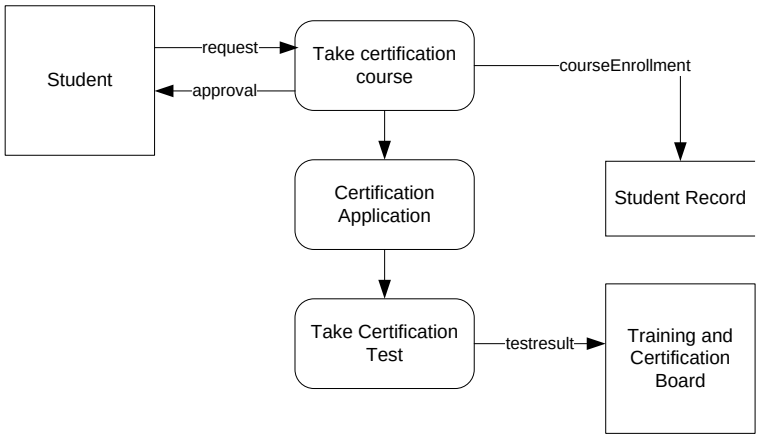

Figure 1.5 Sample Gane-Sarson data flow diagram

2 VIEWPOINTS

This document presents the proposed NB DSS software architecture from the following four viewpoints:

1. The ***Logical Viewpoint***, which is a logical representation of the key components within the software system and their interactions. It provides overview only and does not describe the presented components in more detail.

2. The ***System Use Viewpoint***, which addresses the system from an end-user point of view. This viewpoint responds to the questions: How will the interaction with the system be, how can it be integrated with other systems – or new systems – within an organisation?

3. The ***Implementation Viewpoint,*** which addresses the system from a software developer point of view. It responds to the question: What technologies are used, what applications tiers exist, what client-server model is being used etc.?

4. The ***Infrastructure Viewpoint***, which defines the way the actual software programs (processes) are instantiated and deployed on the physical hardware and how the processes communicate with each other.

Compared with the well-known 4+1 software architecture approach, the logical viewpoint correspond to the "4+1" logical viewpoint, the implementation view corresponds to the "4+1" implementation view[1], the infrastructure viewpoint corresponds to a combination of the "4+1" deployment and process view points. The System Use view point as included in the SRS does not have a corresponding view in the "4+1" architecture approach. The "4+1" Scenarios view point corresponds to the SRS document.

[1] This viewpoint is strictly speaking not part of an overall architecture and is not represented in great detail in this document. It will be addressed further in the detailed analysis and design phase.

3 ARCHITECTURE OVERVIEW

This chapter explains the Consultants vision for the NB DSS and – at the same time – defines the major guiding architectural principles. Later chapters present the different aspects of the architecture in more detail.

3.1 Vision

DSS is an abbreviation for Decision Support System which in many ways is synonymous with business intelligence. Wikipedia has the following definition of business intelligence.

> *Refers to skills, technologies, applications and practices used to help a business acquire a better understanding of its commercial context. Business intelligence may also refer to the collected information itself and the knowledge developed from this information.*
>
> *BI applications provide historical, current, and predictive views of business operations. Often these applications use data gathered into a data warehouse or a data mart. Occasionally the applications work from operational data. Common functions of business intelligence applications are reporting, OLAP, analytics, data mining, business performance management, benchmarks, and predictive analysis.*
>
> */ c.f. wikipedia*

Keywords in this definition are:

- Acquire a better understanding
- Knowledge developed from collected information
- Provide predictive view
- Predictive analysis

The NB DSS system will provide historical, current, and predictive views of business operations using data collected into its database. It will have functionality that makes it possible to analyse the data, to perform what-if analyses through its modelling capabilities and to condense and aggregate its outputs into reports for later decision making.

3.2 Architecture overview

Figure 3.1 defines the NB DSS in terms of highest-level interactions.

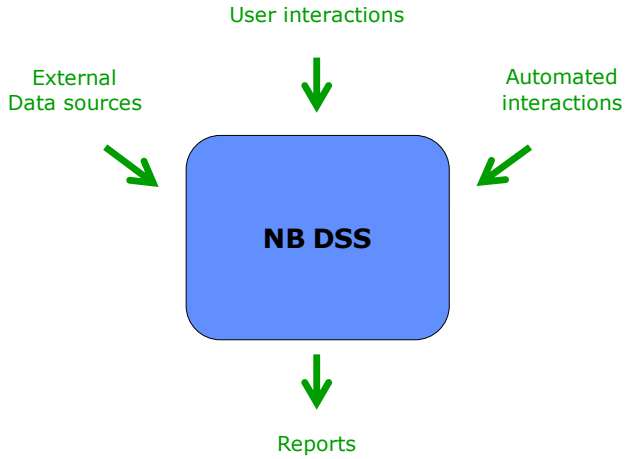

Figure 3.1 NB-DSS Highest-level interactions

At this level the system is a black-box supporting the following types of interactions:

- User interactions – representing a user working with the system, typically for preparing and analysing data, setting up and running scenarios
- External data sources – representing loading of data into the NB DSS database
- Automated interactions – representing the use of the system as a programming platform, e.g. scripting or coding against the public programmable interfaces provided by the system

Output from the system typically is reported in various formats.

At a level deeper the system reveals itself as an integration platform consisting of the following 3 high-level components (Figure 3.2).

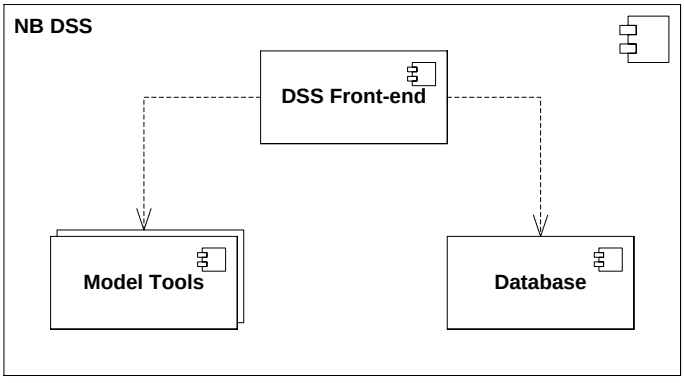

Figure 3.2 NB DSS – first level break-down (UML)

The figure identifies three top-level components: The DSS Front-end, the model tools and the DSS database. The guiding architectural principles in designing these components and their interactions are briefly described below.

It is the Consultants experience that flexibility with respect to model tools is very important. Model tools are becoming increasingly advanced and multi-disciplinary and a

DSS system needs to be able to integrate with many different kinds of such tools, not all of which are known at the time of the initial system development.

> *Guiding architectural principle: Establish well-defined interfaces for integrating model tools with the DSS Front-end.*

The DSS Front-end is where the Information Management part of the system is situated, i.e. the capabilities for processing and analysing data, for creating scenarios, performing multi-criteria analysis etc. A proper DSS Front-end should be designed it such a way that it can be extended with new functionality over time by adding new components to the live system. This includes, for example, adding functionality for displaying simulated results on top of Google Maps or integrating the NB DSS with AutoCAD for coupling infrastructure data with structural design drawings.

> *Guiding architectural principle: Establish well-defined interfaces and protocols for DSS Front-end modules. Distributing DSS Front-end functionality amongst the software components should strictly adhere to the separation of concern patterns – similarly with the component internals.*

The database in the NB DSS system is a data warehouse, i.e. a consolidation of all applicable water resources data related to the mission of the Nile Basin Initiative. This also implies that, over time, the database will grow very large. In order to prevent users of the system from 'drowning' in data, the database should provide logical views of the data entities that are of particular interest for the specific work situation.

> *Guiding architectural principle: Apply the data warehouse principle of providing (logically or physically) data mart[2]s for specific, subject oriented data designed to answer specific questions to a specific set of users.*

Users of the system will have varying roles in their interactions with the system. One user might be responsible for the modelling part in one study, the data part in another, reviewer in a third and study manager in a fourth. I.e. control of access to data can easily become unmanageable if this has to be based on a direct mapping between user and data entities.

> *Guiding architectural principle: Base access control on work roles seen in relation to specific studies.*

[2] For the NB DSS the term Study is used

4 SOFTWARE COMPONENTS

The SRS /2/ identifies numerous required functional components, i.e., a logical grouping of functionality. These functional components must be converted into actual software components for the system design and implementation phases.

This chapter explains how the functional components established in /2/ are converted into software component.

4.1 Establishing Software Components

Establishing software components from requirements and use cases is not a linear process. It requires iteration between design and sometimes even implementation and analysis of requirements and use cases.

As part of this task, requirements and use cases were analyzed and converted into functional components, experiences from previous projects were converted into patterns and candidate designs were established. The candidate designs were evaluated against the requirements and the use cases – and new iterations were performed to derive a final design.

Table 4.1 below shows how use cases and functional components were used to establish the software components described in this chapter.

Note the use cases identified in /2/ and listed here for reference are subject for change and clarification during the detailed analysis and design stages of the development cycles.

Table 4.1 Use cases, functional and software components[3]

Use case (id and name)	Activities included	Functionality component	Software component
001: Import time series	Import time series including geo-referencing.	Time series	Timeseries Manager
002: Use time series tool	Process a time series (into another time series, a number or other output).	Time series	DSS Tools
003: Visualise time series	Plot and/or tabulate. Access meta information.	Time series	Timeseries Manager
004: Use table tool	Create tabular structure – either as tool output or manually.	Tables	DSS Tools
005: Publish in report	Insert content (text, table, plot, GIS) in a report structure. Manually or automatic.	Reporting	Report Manager
006: Visualise GIS layer	Select and display GIS layer on map – define symbology to use.	GIS	GIS Manager Map Control

[3] The use cases are maintained in a database and the numbers used in Table 4.1 are the primary id in the database table. This is the reason why the numbers are not in ascending order and have wholes.

Use case (id and name)	Activities included	Functionality component	Software component
007: Inspect GIS layers	Access properties of features, including meta data.	GIS Meta data	GIS Manager Meta Data Manager
008: Filter time series	Make filter to select one or more time series from the full available list.	Time series	Timeseries Manager
009: Inspect time series	Access properties, meta data and data to assess content of a time series.	Time series Meta data	Timeseries Manager Meta Data Manager
010: Use rainfall runoff model	Calculate runoff from precipitation, evaporation etc. Special case of “Run Scenario”.	Scenarios Model Tools	Scenario Manager Model Tools
011: Setup MIKE BASIN	Define a MIKE BASIN setup in the model tool, with all inputs, parameters and outputs defined. Usually involves calibration of parameters by running with historical inputs and comparing outputs with historical measurements. (Special case of “setup model in model tool”).	Model Tools	Model Tools
013: Calibrate MIKE BASIN	Sub-set of “Setup MIKE BASIN”.	Model Tools	Model Tools
014: Register model	Define attributes in the NB DSS for the system to “know” a model setup, i.e. inputs, outputs, Hydro Objects.	Scenarios	Scenario Manager
015: Create scenario	Define the actual inputs to use for a simulation and the outputs to extract from the simulation.	Scenarios	Scenario Manager Model Linker
016: Run scenario	Prepare the setup and inputs, execute the model tool and retrieve the outputs from a simulation.	Scenarios	Scenario Manager Model linker
018: Visualise tables	Present information in tabular form.	Tables	Table Manager
019: Create study	Define the data compartment used to encapsulate information related to a specific investigation/study.	Studies	Study Manager
020: Manage study	Edit properties of the study. Set user permissions to study data.	Studies	Study Manager
021: Import GIS data	Import GIS data from external sources (different formats) into the DSS database.	GIS	GIS Manager

Use case (id and name)	Activities included	Functionality component	Software component
021: Visualize GIS data	The same as “Visualize GIS layer”.	GIS	GIS Manager
023: Edit GIS data	Edit properties of GIS layer and features. Edit meta data for GIS layer.	GIS	GIS Manager
024: Setup semi-distributed rainfall-runoff model (MIKE SHE light)	Define a MIKE SHE Light setup in the model tool, with all inputs, parameters and outputs defined. Usually involves calibration of parameters by running with historical inputs and comparing outputs with historical measurements. (Special case of “setup model in model tool”).	Model Tools	Model Tools
025: Register MIKE BASIN model	Define attributes in the NB DSS for the system to “know” a MIKE BASIN model setup, i.e. inputs, outputs, Hydro objects. (Special case of “register model”).	Scenarios	Scenario Manager
026: Use GIS tool	Perform GIS processing of data in a map (e.g. query by feature, area calculations).	GIS	DSS Tools
027: Create linked model	Define links (output to input) between sequentially run models.	Linked models	Scenario Manager Model Linker
028: Calibrate model	Adjust model parameters to make the model fit historical conditions with known inputs and output.	Model Tools	Model Tools
032: Clone and modify MIKE BASIN model and scenario	Create a copy of all DSS database definitions for a MIKE BASIN model and associated scenarios. Edit the copied model setup in the MIKE BASIN model tool. Re-register the model setup and correct/adjust definitions in the database reflecting the differences following the edit. Highlight/adjust/correct scenario definitions to make them match the new setup. (Special version of “Clone and modify model and scenario”).	Scenarios Model Tools	Scenario Manager Model Tools
034: Setup MIKE 11 model	Define a MIKE 11 setup in the model tool, with all inputs, parameters and outputs defined. Usually also involve calibration of parameters by running with historical inputs and comparing outputs with historical measurements. (special case of “setup model in model tool”).	Model Tools	Model Tools

Use case (id and name)	Activities included	Functionality component	Software component
036: Register MIKE 11 model	Define attributes in the NB DSS for the system to "know" a MIKE 11 model setup, i.e. inputs, outputs, HydroObjects. (Special case of "register model").	Scenarios	Scenario Manager
038: Define indicator	Process of establishing the characteristics of a variable to enter a further analysis. The variable may be a result of a GIS tool, a time series tool or a macro calculation. {This definition is performed outside the NB DSS by subject matter experts}	Indicator	Analysis Manager
039: Use MCA indicator tool	Establish definition of indicators including references to and functions on underlying data.	Indicator	Analysis Manager
040: Run MCA	Access definitions, properties and functionality to perform a Multi Criteria Analysis (MCA).	MCA	MCA
041: Take decision	The process of the user assessing available information and making a decision based upon that.	*User action*	
042: Use CBA	Perform Cost-Benefit Analysis (CBA), define indicators, perform data conversions.	CBA	CBA
043: Plan scenarios	Process of identifying the conditions to apply to the model and/or the modification to the models.	*User actions*	
044: Clone and modify model and scenario	Create a copy of all DSS database definitions for a model setup and associated scenarios. Edit the copied model setup in the model tool. Re-register the model setup and correct/adjust definitions in the database reflecting the differences following the edit. Highlight/adjust/correct scenario definitions to make the match the new setup.	Scenarios	Scenario Manager
045: Setup ensemble scenario	Define a scenario for running a model setup using an ensemble of input time series and performing post-processing.	Ensembles Scenarios	Timeseries Manager Scenario Manager Ensemble Modeler

Use case (id and name)	Activities included	Functionality component	Software component
046: Run ensemble scenario	Run the model setup and apply the ensemble of time series. Execute defined post-processing.	Ensembles Scenarios	Timeseries Manager Scenario Manager Ensemble Modeler
047: Compare scenarios	Select (manually or automatically) comparable information from two executed scenarios and presenting the output.	Scenarios	Scenario Manager Timeseries Manager
048: Export study	Export all data in a study to files in defined exchange format.	Studies	Study Manager
050: Import Study	Import exported study data into the DSS database, restoring the original structure.	Studies	Study Manager
051: Use GIS catchment delineation tool	Use advanced GIS tool to make catchment delineation.	GIS	GIS Manager
052: Edit time series	Modify time series data in database.	Time series	Timeseries Manager
053: Assess data availability	Evaluate whether available data are sufficient for the purpose at hand.	Time series GIS Tables Meta data	Timeseries Manager GIS Manager Table Manager Meta Data Manager
054: Define scenario	Same as “Plan scenario”.	*User actions*	
055: Setup CBA	Configure CBA tool with respect to parameters and links to data.	CBA	CBA
056: Define optimization scenario	Special case of “define Scenario” dealing with planning how to use optimization with a model setup.	Optimizations	Scenario Manager Optimizer
057: Create optimization scenario	Configure a scenario in the NB DSS to run a model with respect to optimization (inputs, objectives, constraints, methods).	Optimizations	Scenario Manager Optimizer
058: Run optimization scenario	Run the model setup and apply the optimization scenario definitions and data.	Optimizations	Scenario Manager Optimizer
060: Establish understanding of model tool	Study and understand the data structures of and available interfaces to data of a model setup/model tool.	*User actions*	

Use case (id and name)	Activities included	Functionality component	Software component
061: Design model tool adapter	Write specification for a model tool adapter (what information it be able to share/exchange with the DSS Front-end).	User actions	
062: Analyse model tool inter-communication options	Understand the capabilities of the possibilities for model-to-model communication.	User actions	
062:Analyse of IModelAdapter	Understand the capabilities of the interface to model adapters.	User actions	
064: Code model tool adapter	Program and unit test code.	User actions	
065: Change model tool in order to enable model tool to model tool communication	Place and execute change request s to 3rd part modelling software in order to match NB DSS integration requirements.	User actions	
066: Test model registration	Verify if the use of IModelAdapter for configuration works with an actual model setup.	Scenarios Model Tools	Scenario Manager Model Tools
068: Test create scenario	Verify that the DSS Front-end can use the registration details for creating a scenario from the model setup.	Scenarios	Scenario Manager
070: Test run scenario	Verify that execution of a model setup with scenario definitions is possible – and successful.	Scenarios	Scenario Manager
071: Test model linkage	Verify that model linkage is possible with the new adapter	Linked models	Scenario Manager
072: Test model inter-communication	Check if programmed model tool changes meet expectations with respect to time-step control of model linkage.	Linked models	Scenario Manager
073: Import table	Create table structure and Import tabular data for storage in database.	Tables	Table Manager
074: Create time series	Establish a time series with properties, data and meta data.	Time series	Timeseries Manager
075: Create hydro object	Link tabular data with a hydro object.	Hydro objects	Hydro Object Manager
076: Use soil erosion tool	Use advanced time series tool to calculate soil erosion.	Time series	DSS Tools
077: Link time series to feature	Create a geo-reference for a time series.	GIS	GIS Manager
078: Edit hydro object	Edit a hydro object instance.	Hydro objects	Hydro Object Manager

Use case (id and name)	Activities included	Functionality component	Software component
079: Use demand calculator tool	Use advanced time series tool to demands and loads.	Time series	DSS Tools
080: Use ensemble generator tool	Use advanced time series tool to establish an ensemble of time series.	Ensembles	DSS Tools
081: Setup MCA	Define parameters and data links in MCA tool.	MCA	MCA
082: Calibrate MIKE 11 model	Sub-set of "Setup MIKE 11", defining the parameters of a model to fit historical events.	Model Tools	Model Tools
085: Edit model	The process of modifying the model setup in the model tool.	Model Tools	Model Tools
086: Run CBA	Start the CBA and access definitions, properties and functionality via the UI.	CBA	CBA
088: Export time series data	Export time series data from the NB DSS database to the file system	Time series	Timeseries Manager
089: Export GIS data	Export GIS data from the NB DSS database to the file system	GIS	GIS Manager

In addition to the software components listed in Table 4.1, the following components were identified during the iterations between analysis and design.

- Shell
- Application
- Model tool adapter
- Database
- DSS Proxy
- Script Manager
- System administration

All software components will be described in the following section.

4.2 Software Component Descriptions

Table 4.2 describes the software components identified during the software analysis and design phase:

Table 4.2 Software component descriptions

Software Component	Description
Analysis Manager	The Analysis Manager provides a framework for analysis tools like MCA and CBS. Common facilities like Indicators reside in this component.
Application	The application component is the top level functional component as it via the modules represent en entry point for functionality in the system.
CBA	The CBA component encapsulates the CBA analysis tool
Database	The Database component implements the physical database
DSS Proxy	The DSS Proxy component implements a piece of the NB DSS allowing distribution of batch tasks.
DSS Tools	The DSS Tools is a common component including all implemented tools (time series tools, GIS tools, analysis tools etc.) as well as a framework for further expansion of the system with respect to tools.
Ensemble Modeller	This component assists the Scenario Manager in definition scenarios using time series ensembles
GIS Manager	GIS Manager provides GIS layer management functionality but primarily handling of data access for GIS data. Visualisation of GIS data is in the hands of the Map Component and GIS tools are in the DSS Tools component
Hydro Object Manager	The Hydro Object Manger controls creation, editing, searching for and deleting of Hydro Objects.
Map Component	The Map Component implements the visualisation of GIS data (currently ThinkGeo) in a common manner
MCA	The MCA component encapsulates the advanced analysis tool, MCA
Meta data Manager	The Meta Data Manager defines a common framework for handling meta data across entities in the system. A Meta data browser also resides here.
Model Linker	This component assists the Scenario Manager in handling definition and execution of linked models and scenarios
Model Tool Adapter	The Model Tool Adapter component assist the Model Tools component in encapsulating model specific logic, making model tools “pluggable” into the NB DSS
Model Tools	The Model Tools component contains the actual model tools in the system(whether known or unknown at time of delivery)
Optimizer	This component assists the Scenario Manager in handling definitions and execution of optimization scenarios
Report Manager	The Report Manager allows a user to create and handle documents and pieces of documents.
Scenario Manager	The Scenario Manager provides functionality to register models, define (and edit) scenarios, execute and compare scenarios. Subcomponents will provide specialized logic where required (e.g. Optimizer)
Script Manager	The Script Manager controls creation, storage and execution of scripts wherever they are used in the application.

Software Component	Description
Shell	The Shell component provides the framework for the application to interact with the Modules. It ties application, user interface and module functionality together. It contains the executables of the system.
Study Manager	The Study Manager gives an interface for the user to administer data and user permissions for logical data compartments.
System Administration	This component is used for system administration tasks like adding new users to the system and changing access control lists.
Table Manager	The Table manager contains logic to support handling of tabular data (i.e. rating curves) in a manner allowing users to specify layout and content and the application to refer and operate on the data.
Timeseries Manager	The Timeseries Manager provides time series management functionality. This includes time series data access, logic related to time series and visualisation of time series. Time series tools are NOT part of this component but part of the DSS Tools component.

4.3 Software Components and Use Case Mapping

Table 4.3 shows the mapping of software components and use cases. Not all components have been derived from or touched by use cases. Some are identified from looking at the functional requirements; others arise from a need to isolate certain functionality in the application design.

Table 4.3 Software components and use case mapping

Software Component	Use Case
Analysis Manager	Define indicator
	Use indicator tool
Application	
CBA	Run CBA
	Setup CBA
	Use CBA
DSS Proxy	
DSS Tools	Use GIS catchment delineation tool
	Use GIS tool
	Use table tool
	Use time series tool
Ensemble Modeller	Run ensemble scenario
	Setup ensemble scenario
GIS Manager	Assess data availability
	Edit GIS data

Software Component	Use Case
	Import GIS data
	Link time series to feature
	Visualize GIS data
Hydro Object Manager	Create hydro object
	Edit hydro object
Map Component	
MCA	Run MCA
	Setup MCA
Meta Data Manager	Assess data availability
	Inspect GIS layers
	Inspect time series
Model Linker	Create linked model
	Create scenario
	Run scenario
	Test model inter-communication
	Test model linkage
Model Tools	Calibrate MIKE 11
	Calibrate MIKE BASIN
	Calibrate model
	Clone and modify MIKE BASIN model and scenario
	Edit model
	Setup MIKE 11 model
	Setup MIKE BASIN
	Setup semi-distributed rainfall-runoff model (MIKE SHE light)
	Test model registration
	Use rainfall runoff model
Model Tool Adapter	
Optimizer	Create optimization scenario
	Define optimization scenario
	Run optimization scenario
Report Manager	Publish in report
Scenario Manager	Clone and modify MIKE BASIN model and scenario
	Clone and modify model and scenario
	Clone DSS model setup and scenario
	Compare scenarios

Software Component	Use Case
	Create linked model
	Create optimization scenario
	Create scenario
	Define optimization scenario
	Modify DSS scenario
	Register MIKE 11 model
	Register MIKE BASIN model
	Register model
	Run ensemble scenario
	Run optimization scenario
	Run scenario
	Setup ensemble scenario
	Test create scenario
	Test model inter-communication
	Test model linkage
	Test model registration
	Test run scenario
	Use rainfall runoff model
Script Manager	
Shell	
Study Manager	Create study
	Export study
	Import Study
	Manage study
System Administration	
Table Manager	Assess data availability
	Import tables
	Visualise tables
Timeseries Manager	Assess data availability
	Create time series
	Edit time series
	Filter time series
	Import time series
	Inspect GIS layers

Software Component	Use Case
	Inspect time series
	Run ensemble scenario
	Setup ensemble scenario
	Use demand calculator tool
	Use ensemble generator tool
	Use soil erosion tool
	Visualise time series

4.4 Software Components and Requirement Mapping

Table 4.4 shows the mapping between software component and requirements. The Comments column in the table is used to identify issues that have to be resolved during the detail analysis and design phases.

Table 4.4 Software components and requirement mapping

Software Component	Requirement	
Shell	2.1.1	Graphical User Interface (GUI)
	2.1.1.1	Interactive, fully menu driven graphical, hyperlinked
	2.1.1.1 (1)	The GUI shall be as powerful and interactive as possible for creating, locating and inter-connecting the various components.
	2.1.1.1 (2)	The GUI shall allow data entry by users with features to add, modify and delete data according to predefined user privileges
	2.1.1.1 (3)	The GUI shall have a menu structure which is always evident to the user. The system shall provide clear feedback regarding the...
	2.1.1.2	Multi-language support (English, French)
	2.1.1.3	On-line, context-sensitive help facility with an online hierarchical & cross-linked help system in HTML
	2.1.1.4 (1)	The GUI shall be consistent with regard to screen layouts, messages, text and graphic positions etc. to ease learning of the ...
	2.1.1.4 (2)	The GUI shall have readable text that is easy to understand, even for non-native speakers.
	2.1.1.4 (3)	The GUI shall have readable text that is easy to understand, even for

Software Component	Requirement	
		non-native speakers.
	2.1.1.4 (4)	The GUI shall have the ability to make adaptations due to regional differences in line with the platform and infrastructure s...
	2.1.1.5	Error/action messages for wrong entries
Application	2.1.2.7 (1)	Batch data processing features: Batch mode requirements
	2.1.2.7 (2)	Batch data processing features: Macro/scripting
	2.1.2.7 (3)	Batch data processing features: Storage and management of coded procedures
	2.1.2.7 (4)	Batch data processing features: Traceability of outputs
	2.1.2.8 (1)	The DMI shall ensure seamless communication/integration between modules/models of the DSS.
	2.1.2.8 (2)	The DMI shall be the integration base for module-module and module-database communication and interaction for all tools, mode...
	2.1.2.8 (3)	In addition, the data management interface (DMI) shall support plug-ins of modules that meet the requirements of the communic...
	2.1.7.6	Support for user defined process representation/algorithms
	2.2.1.3	Modular implementation
	2.2.1.3 (2)	Layered architectures, modular design, well defined interfaces, object-orientation and component-based development
	2.2.1.7	Highly cohesive and loosely coupled design
Timeseries Manager	2.1.3(1)	IMS: Pre- and Post-Processors and Data Analysis Tools - visualization tools
	2.1.3(2)	IMS: Pre- and Post-Processors and Data Analysis Tools - charting functionality
	2.1.3(3)	IMS: Pre- and Post-Processors and Data Analysis Tools - multiple selections
	2.1.3(4)	IMS: Pre- and Post-Processors and Data Analysis Tools - customization

Software Component	Requirement	
		of predefined
	2.1.3(5)	IMS: Pre- and Post-Processors and Data Analysis Tools - Charts in IMS
	2.1.3(6)	IMS: Pre- and Post-Processors and Data Analysis Tools - export interface
GIS Manager	2.1.4.1	IMS: Pre- and Post-Processors and Data Analysis Tools - export interface
	2.1.4.2	OGC compatibility and compliance
	2.1.4.2 (1)	OGC compatibility and compliance: Import / Export of spatial information
	2.1.4.2 (2)	OGC compatibility and compliance: Projections
	2.1.4.3	Support of spatial data pre-processing for model inputs
	2.1.4.3 (1)	Support of spatial data pre-processing for model inputs: Process geo-referenced vector and raster data
	2.1.4.3 (2)	Support of spatial data pre-processing for model inputs: Pan, zoom and select
Table Manager		
Hydro Object Manager	2.1.5.2	Basic set of pre-defined node types
	2.1.5.2 (1)	start or input nodes
	2.1.5.2 (2)	Demand nodes
	2.1.5.2 (3)	Structural components
	2.1.5.2 (4)	Control nodes
	2.1.5.2 (5)	Geometry nodes
	2.1.5.2 (6)	Aquifers
	2.1.5.2 (7)	End nodes
Scenario Manager	2.1.5.10	Model nesting, hierarchical linkage of networks
	2.1.6.1	Model scenario management
	2.1.6.3	Direct scenario comparison
	2.1.6.4	Simulation based optimization
	2.1.6.5	Sensitivity analysis
	2.1.6.6	Stochastic modelling (error distribution)
	2.1.6.6 (3)	The stochastically generated sequences shall be used for deterministic applications

Software Component	Requirement	
Analysis Manager	2.1.5.12	Tools for converting model outputs into desired criteria using user-defined methods
	2.1.5.19	Yield/reliability analysis for reservoirs and catchments
	2.1.8	Multi-Criteria-Analysis (MCA) Tools
	2.1.8.3	Automatic model linkage
Report Manager	2.1.2.4	User defined report generation
	2.1.2.4 (1)	Report configuration
	2.1.2.4 (2)	IMS shall enable the user to customize predefined report templates
	2.1.2.4 (3)	All reports shall be stored and managed in the IMS.
	2.1.2.4 (4)	Export of all reports to standard applications such as office software packages/formats shall be possible.
Script Manager	2.1.7.6(1)	Support for user defined process representation/algorithms
	2.1.7.6(2)	Support for user defined process representation/algorithms
	2.1.7.6(3)	Support for user defined process representation/algorithms
	2.1.7.6(4)	Support for user defined process representation/algorithms
	2.1.7.6(5)	Support for user defined process representation/algorithms
Study Manager		
Meta Data Manager	2.1.2.3	Standard META data model
	2.1.2.3 (1)	Standard META data model: General requirements
	2.1.2.3 (2)	Standard META data model: Main data types
	2.1.2.3 (3)	Standard META data model: Authorization and user access
	2.1.2.3 (4)	Standard META data model: Business process
Ensemble Modeller	2.1.6.6	Stochastic modelling (error distribution)
	2.1.6.6 (3)	The stochastically generated sequences shall be used for deterministic applications
Model Linker	2.1.5.10	Model nesting, hierarchical linkage of networks
Optimizer	2.1.6.4	Model nesting, hierarchical linkage

Software Component	Requirement	
		of networks
MCA	2.1.5.12	Tools for converting model outputs into desired criteria using user-defined methods
	2.1.8.1	Multiple MCA methods
	2.1.8.2	User defined open list of criteria
	2.1.8.3	Automatic model linkage
CBA	2.1.5.11	Economic analysis of scenarios (CBA)
Database	2.1.2.1	Standard RDMS with database level application clustering feature
	2.1.2.1 (1)	The DBMS shall store all relevant basin data as well as all data to manage and run the NB DSS
	2.1.2.1 (2)	The DBMS shall support data definition and manipulation according to state of the art standards for data base managements sys...
	2.1.2.1 (3)	Database corruption and losses
	2.1.2.1 (4)	DBMS data type requirements
	2.1.2.1 (5)	DBMS geo-referenced spatial information requirements
	2.1.2.1 (6)	DBMS shall efficiently operate on voluminous time series data
	2.1.2.1 (7)	IMS access to DBMS - interfaces to different programming languages and other third-party applications.
	2.1.2.1 (8)	DBMS synchronization requirements
	2.2	Non-functional requirements
	2.2.1.9	Embedded backup tools and backup strategy
Model Tools	2.1.5	Dynamic Water Budget and Allocation Model
	2.1.5 (1)	Dynamic Water Budget and Allocation Model - multiple years
	2.1.5 (2)	Dynamic Water Budget and Allocation Model - surface+groundwater
	2.1.5 (3.1)	Dynamic Water Budget and Allocation Model - water allocation per user
	2.1.5 (3.2)	Dynamic Water Budget and Allocation Model - source priority per user
	2.1.5 (3.3)	Dynamic Water Budget and Allocation Model - operation rules

Software Component	Requirement	
	2.1.5 (3.4)	Dynamic Water Budget and Allocation Model - groundwater management rules
	2.1.5 (3.5)	Dynamic Water Budget and Allocation Model - operation rules of the diversion structures
	2.1.5 (3.6)	Dynamic Water Budget and Allocation Model - water allocation based on targets
	2.1.5 (3.7)	Dynamic Water Budget and Allocation Model - proportional water allocation
	2.1.5 (3.8)	Dynamic Water Budget and Allocation Model - re-use of drainage water
	2.1.5 (4)	Dynamic Water Budget and Allocation Model - explicit transfer scheme
	2.1.5.1	Data driven, user specified, interactive network configuration
	2.1.5.13	Lateral inflow, lateral catchments, floodplain representation
	2.1.5.14	Hydropower production
	2.1.5.15	Hydraulic model (1D)
	2.1.5.16	Sediment transport in river networks and reservoir siltation
	2.1.5.17	Multiple routing methods (data dependent) (optional)
	2.1.5.18	Open (user defined) list of node types
	2.1.5.2	Basic set of pre-defined node types
	2.1.5.2 (1)	start or input nodes
	2.1.5.2 (2)	Demand nodes
	2.1.5.2 (3)	Structural components
	2.1.5.2 (4)	Control nodes
	2.1.5.2 (5)	Geometry nodes
	2.1.5.2 (6)	Aquifers
	2.1.5.2 (7)	End nodes
	2.1.5.3	Geo-referenced network geometry
	2.1.5.4	Explicit routing of flow
	2.1.5.5	Explicit routing of flow
	2.1.5.6	Explicit mass budget, error statistics
	2.1.5.7	Explicit groundwater representation and coupling

Software Component	Requirement	
	2.1.5.8	Multiple reservoirs, including hydro-power generation
	2.1.5.9	Variable reach geometry
	2.1.6.2	Embedded calibration methods with error statistics
	2.1.6.2 (1)	The system shall provide embedded calibration tools
	2.1.6.2 (2)	The calibration tools shall support calibration of any part of the basin under consideration
	2.1.6.2 (3)	The system shall provide appropriate objective functions
	2.1.6.2 (4)	The calibration tools shall enable the users to determine model parameter sets
	2.1.7.1	Rainfall-runoff models (lumped, semi-distributed)
	2.1.7.2	Irrigation water demand estimation, crop production model
	2.1.7.3	Water quality model (DO/BOD, conservative, first order decay
	2.1.7.4	Catchment erosion process modelling
	2.1.7.5	Multiple evapotranspiration estimation methods
	2.1.7.6	Support for user defined process representation/algorithms
	2.1.7.7	Support for user defined process representation/algorithms
	2.1.7.8	Rainfall-runoff model: fully distributed (optional)
Model Tool adapter	2.2.1.3	Modular implementation
	2.2.1.3 (2)	Layered architectures, modular design, well defined interfaces, object-orientation and component-based development
DSS Proxy		
DSS Tools	2.1.4.3 (3)	Support of spatial data pre-processing for model inputs: Catchment delineation
	2.1.4.4	Spatial analysis, interpolation (GIS links)
	2.1.4.4 (1a)	Spatial analysis, interpolation (GIS links): Geo-processing: Intersection
	2.1.4.4 (1b)	Spatial analysis, interpolation (GIS

Software Component	Requirement	
		links): Geo-processing: Union
	2.1.4.4 (1c)	Spatial analysis, interpolation (GIS links): Geo-processing: Proximity analysis
	2.1.4.4 (2a)	Spatial analysis, interpolation (GIS links): Spatial interpolation: Nearest neighbour
	2.1.4.4 (2b)	Spatial analysis, interpolation (GIS links): Spatial interpolation: Inverse distance
	2.1.4.4 (2c)	Spatial analysis, interpolation (GIS links): Spatial interpolation: Moving polynomials
	2.1.4.4 (2d)	Spatial analysis, interpolation (GIS links): Spatial interpolation: Kriging
	2.1.3.1	Embedded statistical tools
	2.1.3.1 (1a)	Embedded Statistical Tools: Descriptive statistics: Minimum
	2.1.3.1 (1b)	Embedded Statistical Tools: Descriptive statistics: Maximum
	2.1.3.1 (1c)	Embedded Statistical Tools: Descriptive statistics: Mean
	2.1.3.1 (1d)	Embedded Statistical Tools: Descriptive statistics: Median
	2.1.3.1 (1e)	Embedded Statistical Tools: Descriptive statistics: Mode
	2.1.3.1 (1f)	Embedded Statistical Tools: Descriptive statistics: Standard deviation
	2.1.3.1 (1g)	Embedded Statistical Tools: Descriptive statistics: Skewness
	2.1.3.1 (1h)	Embedded Statistical Tools: Descriptive statistics: kurtosis
	2.1.3.1 (1i)	Embedded Statistical Tools: Descriptive statistics: Empirical frequency
	2.1.3.1 (1j)	Embedded Statistical Tools: Descriptive statistics: Cumulative frequency distributions
	2.1.3.1 (2)	Embedded statistical tools: Statistical tests
	2.1.3.1 (2a)	Embedded statistical tools: Statistical tests: Stationarity
	2.1.3.1 (2b)	Embedded statistical tools: Statistical tests: Homogeneity
	2.1.3.1 (2c)	Embedded statistical tools: Statisti-

Software Component	Requirement	
		cal tests: Randomness
	2.1.3.1 (3)	Embedded statistical tools: Double mass analysis
	2.1.3.1 (4)	Embedded statistical tools: Frequency distributions
	2.1.3.1 (4a)	Embedded statistical tools: Frequency distributions: Common distributions
	2.1.3.1 (4b)	Embedded statistical tools: Frequency distributions: Parameter estimation
	2.1.3.2	Time series analysis tools
	2.1.3.2 (1)	Time series analysis tools: Auto and cross correlation analysis
	2.1.3.2 (2)	Time series analysis tools: Duration curves
	2.1.3.2 (3)	Time series analysis tools: Regression analysis
	2.1.3.2 (4)	Time series analysis tools: Generation of long time-series.
	2.1.3.2 (5)	Time series analysis tools: Projection of future demands.
	2.1.3.3	Data treatment and quality assurance tools
	2.1.3.3 (1)	Data Treatment and Quality Assurance Tools: Data Validation Tools
	2.1.3.3 (1a)	Data Treatment and Quality Assurance Tools: Data Validation Tools: Physical or numerical limits
	2.1.3.3 (1b)	Data Treatment and Quality Assurance Tools: Data Validation Tools: rate of rise and fall
	2.1.3.3 (1c)	Data Treatment and Quality Assurance Tools: Data Validation Tools: Comparison against observed behavior
	2.1.3.3 (1d)	Data Treatment and Quality Assurance Tools: Data Validation Tools: Comparison against related variables
	2.1.3.3 (1e)	Data Treatment and Quality Assurance Tools: Data Validation Tools: Comparison between adjacent stations
	2.1.3.3 (1f)	Data Treatment and Quality Assurance Tools: Data Validation Tools: General comparison between dif-

Software Component	Requirement	
		ferent variables
	2.1.3.3 (2)	Data Treatment and Quality Assurance Tools: Graphical Screening Methods
	2.1.3.3 (2a)	Data Treatment and Quality Assurance Tools: Graphical Screening Methods: Maps
	2.1.3.3 (2b)	Data Treatment and Quality Assurance Tools: Graphical Screening Methods: Tables
	2.1.3.3 (2b.1)	Data Treatment and Quality Assurance Tools: Graphical Screening Methods: Tables: All station information
	2.1.3.3 (2b.2)	Data Treatment and Quality Assurance Tools: Graphical Screening Methods: Tables: Time interval
	2.1.3.3 (2b.3)	Data Treatment and Quality Assurance Tools: Graphical Screening Methods: Tables: Display statistics
	2.1.3.3 (2c)	Data Treatment and Quality Assurance Tools: Graphical Screening Methods: Test on extremes
	2.1.3.3 (2c.1)	Data Treatment and Quality Assurance Tools: Graphical Screening Methods: Test on extremes: upper/lower limits
	2.1.3.3 (2c.2)	Data Treatment and Quality Assurance Tools: Graphical Screening Methods: Test on extremes: limits in rise and fall
	2.1.3.3 (2c.3)	Data Treatment and Quality Assurance Tools: Graphical Screening Methods: Test on extremes: User defined rules
	2.1.3.3 (2d)	Data Treatment and Quality Assurance Tools: Graphical Screening Methods: Inspection of temporal variation
	2.1.3.3 (2d.1)	Data Treatment and Quality Assurance Tools: Graphical Screening Methods: Inspection of temporal variation: TS Plot
	2.1.3.3 (2d.2)	Data Treatment and Quality Assurance Tools: Graphical Screening Methods: Inspection of temporal variation: lag/shift
	2.1.3.3 (2d.3)	Data Treatment and Quality Assurance Tools: Graphical Screening

Software Component	Requirement	
		Methods: Inspection of temporal variation: Residuals and movi...
	2.1.3.3 (2e)	Data Treatment and Quality Assurance Tools: Graphical Screening Methods: Inspection of longitudinal/spatial variation
	2.1.3.3 (2e.1)	Data Treatment and Quality Assurance Tools: Graphical Screening Methods: Inspection of longitudinal/spatial variation: Displa...
	2.1.3.3 (2e.2)	Data Treatment and Quality Assurance Tools: Graphical Screening Methods: Inspection of longitudinal/spatial variation: Map pr...
	2.1.3.3 (2e.3)	Data Treatment and Quality Assurance Tools: Graphical Screening Methods: Inspection of longitudinal/spatial variation: Spatia...
	2.1.3.4 (1)	dvanced statistical methods: Multivariate analysis and regression (optional)
	2.1.6.6 (1)	The system shall be able to generate climate data as well as stream flow data
	2.1.6.6 (2)	The system shall support the stochastic generation of data for single site and multi site
Map Component	2.1.4.3 (1)	Support of spatial data preprocessing for model inputs: Process geo-referenced vector and raster data
	2.1.4.3 (2)	Support of spatial data preprocessing for model inputs: Pan, zoom and select
System Administration	2.1.2 (4)	For systematic tracing of results the IMS shall support logging of all processes in the DSS and interactions of the user with the DSS
	2.1.2.5	Support access to system performance statistics
	2.2.4.*	General Administrative

Note from Table 4.4 that

- Three software components have no associated requirements. These are the Table Manager, Study Manager and DSS Proxy – all software components that comes out of the use cases and not the requirements in /1/. Requirements for these should be established during the detailed analysis and design stages.

- Five software components – Model linker, Optimiser, Ensemble modeller, Model tool adapter and CBS - are backed by just 1 or 2 requirements. These are all potential complicated components that need further analysis before being detailed further. Requirements for these should also be further elaborated during the detailed analysis and design stages

- Otherwise requirements are well distributed over the software components.

5 VIEWPOINT: LOGICAL VIEW

This chapter describes the overall architecture in terms of software components and their interactions.

5.1 Components

The software components are derived from the functional components established in SRS /2 / and the use cases /4/. This has been explained in the chapter 4. In the following the term *component* is used for software components.

Components are considered autonomous, encapsulated units that provide one or more interfaces. Components can be reused – and even replaced – within the NB DSS.

Figure 5.1 shows a diagram of all the identified components. The rest of this Chapter will dig into the details with these components and their interactions.

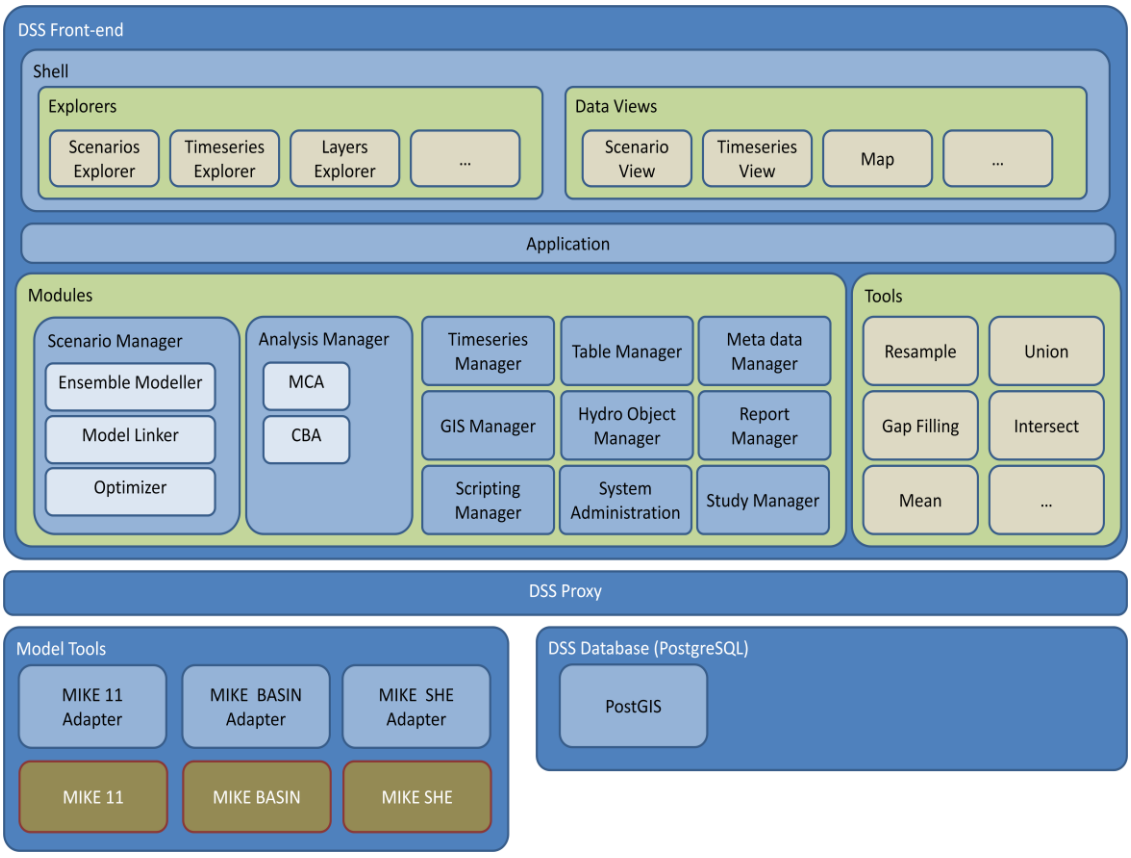

Figure 5.1 NB DSS Components

5.1.1 The Helicopter View

Seen from the helicopter view, the NB DSS can be considered as a system built of three components as shown in Figure 5.2.

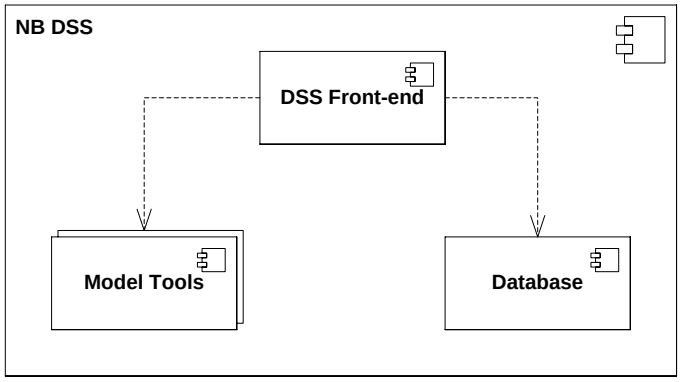

Figure 5.2 NB DSS - helicopter view of components (UML)

The *DSS Front-end* component is a Windows application providing the DSS-specific functionality. The DSS Front-end is built from "scratch" during the NBI project.

The *Database* component is a RDBMS prepared for handling all types of DSS data – such as GIS (spatial) data, time series data, metadata, hydro objects and scenario data.

The *Model Tools* component is a collection of generalized mathematical models such as the DHI proprietary models MIKE 11 and MIKE BASIN – but can also be public domain models or other proprietary models. The Model Tools are "off-the-shelf" software products and as such not directly part of the DSS Front-end.

In the following, each of these three main components will be described in more details.

5.1.2 The DSS Front-end

In this section the components of the DSS Front-end will be described. The DSS Front-end is the "starting-point" for most interactions with the NB DSS. It has two overall components: a *Shell* component and an *Application* component as shown in Figure 5.3.

The purpose of the Shell component is to host and display User Interface (UI) elements. The Application component provides the business functionality. For details on these two components, see sections 5.1.2.1 and 5.1.2.2.

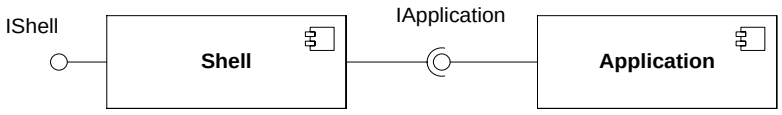

Figure 5.3 Shell and Application components (UML)

The DSS Front-end has a modular architecture supporting the idea of "separation of concerns". A number of *Modules* each takes care of a specific part of functionality within the DSS Front-end. An example of a module is the Time Series Manager, taking care of all functionality regarding time series. Each module is implementing business functionality, data access and a data model.

5.1.2.1 The Application Component

The Application component provides the business functionality of the DSS Front-end by hosting all available *Modules* and *Tools* as shown in Figure 5.4.

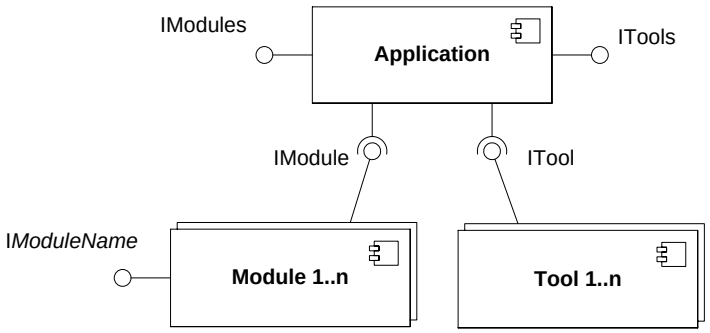

Figure 5.4 Application, Modules and Tools components (UML)

Each module implements the IModule interface. During start-up of the DSS Front-end (the Shell component), the Application component scans the installation folder for available modules (assemblies implementing IModule), instantiates the modules and adds them to the list of modules (IModules – see Figure 5.5).

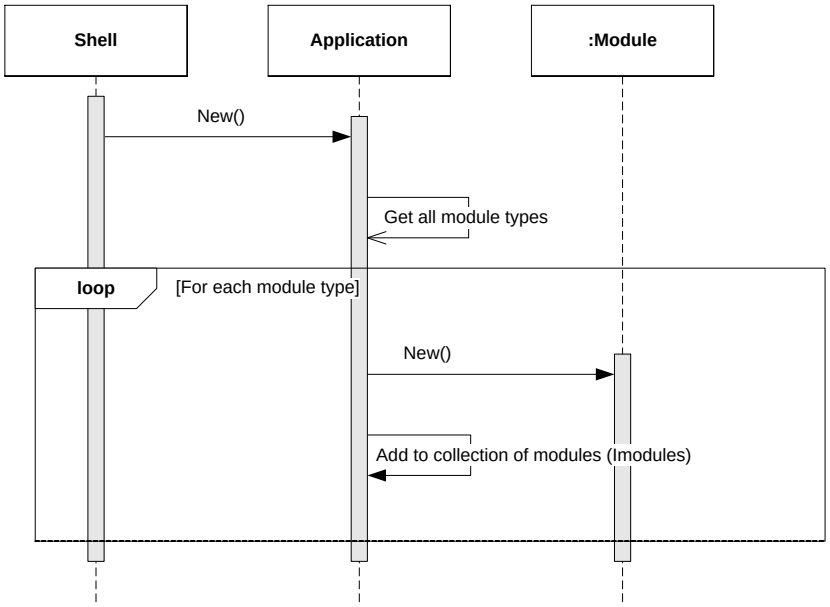

Figure 5.5 Discovering and enabling Modules (UML)

For more details on the modules, see Section 5.1.2.3.

Another group of components hosted by the Application component are the *Tool* components. Tool components are normally lightweight components with much more limited and specific functionality than the modules. An example of a tool is the Standard Deviation Tool, calculating the standard deviation of a time series.

All tools implement the ITool interface, and as the modules, they also have the ability to register themselves during installation. During start-up of the DSS Front-end, the Application component scans a configuration for available Tools and adds them to the list of available tools (ITools) – similar to the sequence for modules shown in Figure 5.5.

For more details on tool components, see Section 5.1.2.4.

All mentioned interfaces are briefly described in Table 5.1.

Table 5.1 Application component interfaces

Interface	Description
IApplication	Interface providing access to the Application members. Examples of properties in the IApplication interface are: • Modules (implementation of IModules) • Tools (implementation of ITools) etc.
IModules	A collection of all modules registered on the machine. Examples of methods in the IModules interface are: • Add() • Remove() • Contains() etc.
ITools	A collection of all tools registered on the machine. Examples of methods in the ITools interface are: • Add() • Remove() • Contains() • GetToolsOfType() etc.
IModule	The interface each module must implement to plug into the Application component. Examples of properties in the IModule interface are: • Name • Description • Enabled etc.
I*ModuleName* (e.g., ITimeseries-Manager)	Interface providing the business functionality of the module, *Module-Name*. For examples of methods in these interfaces, see section 5.1.2.3.
ITool	The interface each tool must implement. The interface provides information about on which object types the tool can execute and contains methods for defining input, executing the tool and delivering output. Examples of methods in the ITool interface are: • SupportsType() • Execute() etc.

5.1.2.2 The Shell Component

The *Shell* component is hosting and displaying the UI components. Three basic types of UI components exist:

- Explorer windows
- Data view windows
- Map windows

The explorer windows serve as the entrance point for exploring, searching and filtering the relevant data for a particular module. The data view windows are used to visualise and edit data and normally include UI controls such as charts or tables. Map windows are used to display GIS data.

Each UI component implements one of the three corresponding interfaces IExplorer-Window, IDataViewWindow or IMapWindow[1] as shown in Figure 5.6.

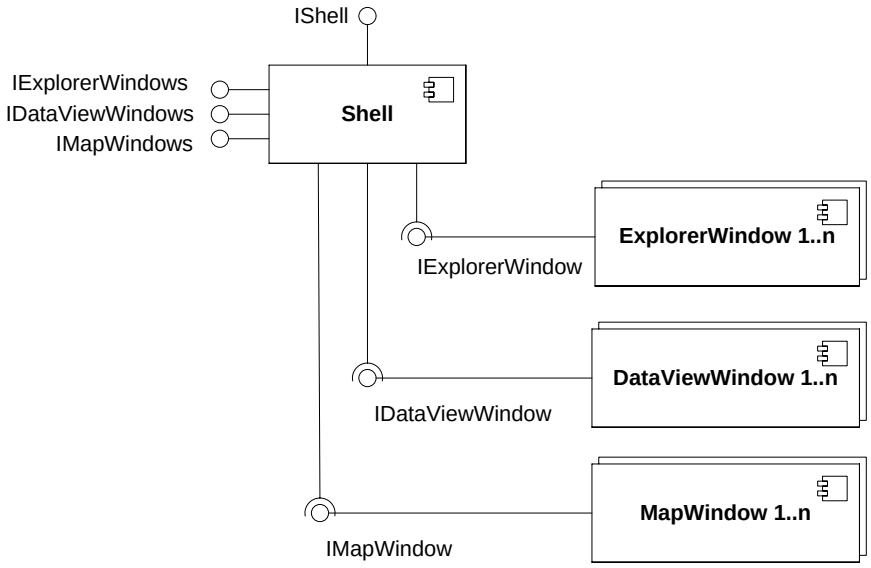

Figure 5.6 Shell and UI components (UML)

All explorer windows implement the IExplorerWindow interface... During start-up the Shell component scans the installation folder for available explorer windows, instantiates them, adds them to the list of explorer windows (IExplorerWindows) and finally displays them (see Figure 5.7).

[1] IMapWindow is actually a descendant of IDataViewWindow. In other words, a map window is a special type of data view window.

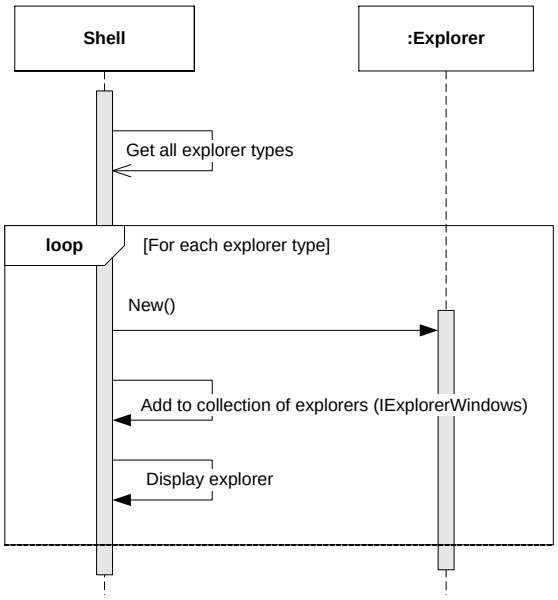

Figure 5.7 Discovering and enabling explorer windows (UML)

All data view windows implement the IDataViewWindow interface.

The map windows are special data view windows that in addition to the IDataView-Window interface implement a IMapWindow interface. This enables replacing an existing implementation of a Map window (ex. ThinkGeo) with another one.

Table 5.2 briefly describes the mentioned interfaces.

Table 5.2 Shell component interfaces

Interface	Description
IShell	Interface provided by the shell component.. Examples of properties in the IShell interface are: • State • ProgressbarVisible • ExplorerWindows (implementeation of IExplorerWindows) • DataViewWindows (implementation of IDataViewWindows) • MapWindows (implementation of IMapWindows) etc.
IExplorerWindows	A collection of explorer windows currently available in the Shell. Examples of methods in the IExplorerWindows are: • Add() • Remove() • Hide() • Show() • Select() • Contains() etc.

Interface	Description
IDataViewWindows	A collection of data view windows currently available in the Shell. This interface has similar methods as IExplorerWindows.
IMapWindows	An array of map windows (DataView windows implementing the IMap-Window interface) currently available in the Shell.
IExplorerWindow	The interface that all explorer windows has to implement to plug into the Shell component. Examples of properties in the IExplorerWindow interface are: • Control • Caption • Name • Enabled etc.
IDataViewWindow	The interface that all data view windows implement. This interface has similar properties as the IExplorerWindow interface
IMapWindow	Derives from IDataViewWindow. Map windows are used for visualization of GIS (spatial) data. Examples of methods in the interface are: • AddLayer() • RemoveLayer() • HideLayer() • ShowLayer() • MoveLayerUp() • MoveLayerDown() etc.

For more details on the Shell UI, see Section 6.2.

5.1.2.3 The Modules

Modules are a specialised type of components as they provide specific subset of the full functionality of the NB DSS system. The available modules are automatically detected during start-up of the DSS-Front-end as described in Figure 5.5. Typically, a module implements all functionality regarding a well-defined category of data. For example the Timeseries Manager module handles all functionality regarding time series data, and the Scenario Manager handles all functionality regarding scenario data. The business logic of a module is exposed via public interfaces. If and when a module needs functionality provided by another module, access to this functionality is achieved through that modules public interface.

Each of the individual modules is using a 2-layered architecture with distinct layers for business logic and data access as shown conceptually in Figure 5.8. A layered architecture has a number of benefits, e.g. the application becomes easier to test, adding new functionality becomes easier and other applications will be able to reuse functionality exposed by the layers. The latter is important for a proper scripting interface.

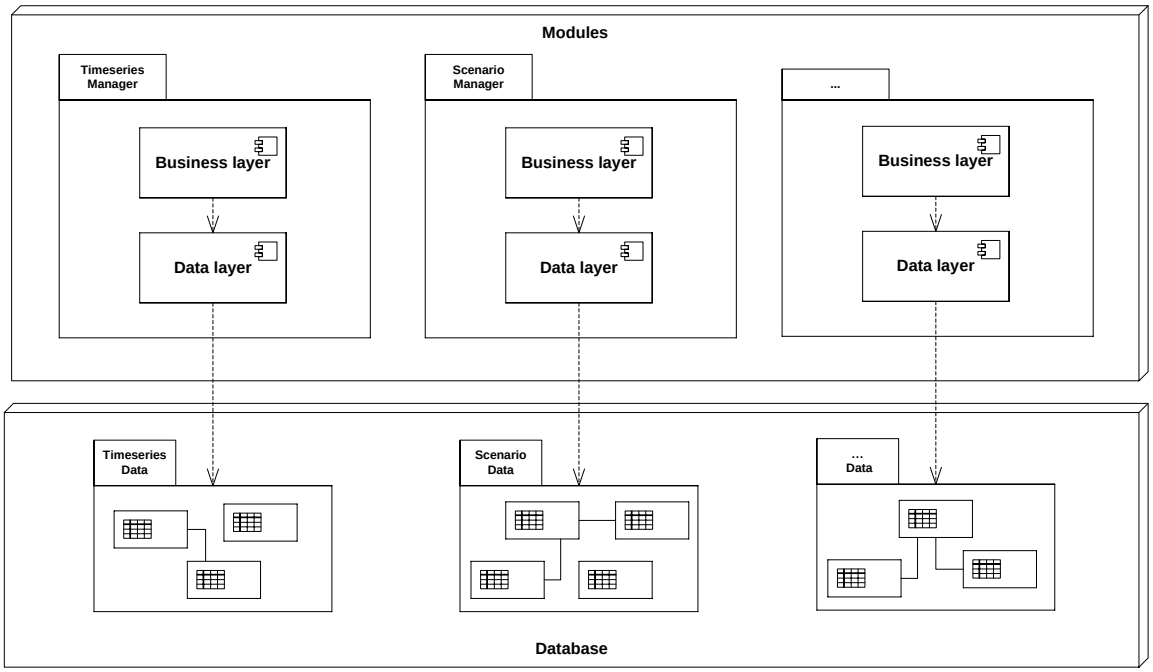

Figure 5.8 Module architecture

Note in Figure 5.8

- Each module is responsible for a part of the full data model. E.g. the Timeseries Manager handles the time series part of the full data model. If other modules need access to time series data they will interact with the Timeseries Manager

- The data model provides full database managed relationships, i.e. an entity belonging to one part of the data model can be used as a foreign key in another part. I.e. the separation shown at the Database layer in Figure 5.8 is conceptual.

This is a strictly layered architecture, meaning that each layer interacts with only the layers directly below and above itself. Below is a short description of the responsibilities of the three layers:

Business Layer
The business layer implements the IModule interface. Furthermore, it provides the business functionality of the module itself through the I*ModuleName* interface (ex. ITimeSeries). For details see Section 5.1.2.1.

Data Layer
The data layer is responsible for creating, reading, updating and deleting data for a particular module. For details see Section 5.1.4.5.

Table 5.3 lists the identified modules.

Table 5.3 Modules

Module	Description

Module	Description
Timeseries Manager	The Timeseries Manager holds all the logic for handling time seriesTimeseries Manager. The module also serves other modules and components with time series via the time series business layer of logic. It's public interface (ITimeseriesManager) includes methods like: • GetTimeseries() • GetAllTimeseriesAttributes() • GetAssociatedFeatures() • RemoveTimeseries() • UpdateTimeseries() etc.
GIS Manager	The GIS Manager holds all the logic for handling GIS data. The module also serves other modules and components with spatial data via the business logic layer of the module. It's public interface (IGisManager) includes methods like: • GetFeatureLayerNames() • GetFeatureLayer() • Update() • Merge() • RefreshDssFeatureLayer() etc.
Scenario Manager	The Scenario Manager deals with all aspects of modelling in the NB DSS in relation to the models integrated via model adapters: setup, registration in the NB DSS, definition of scenarios in the NB DSS, execution scenarios and post-processing of modelling results, including scenario comparison. The Scenario Manager utilizes other components like the Ensemble Modeller, the Optimizer, and the Model Linker to provide functionality for advanced model execution. It's public interface (IScenarioManager) includes methods like: • Create() • Update() • Delete() • GetNewModelSetup() • GetInitialConditionforModel() • GetInputTimeSeriesforModel() • GetOutputTimeSeriesofModel() • GetOutputDataofModel() etc.

Module	Description
Analysis Manager	The Analysis Manager provides functionality for doing advanced analysis of data. It uses separate components like MCA and CBA to provide specialized functionality while providing common functionality like handling indicators. Access to time series, simulations, tables etc. makes use of the respective other modules in charge of these types of data. Members of the public interface could include: • GetAllAnalyses() • GetAnalysisById() • RemoveAnalysis() • GetAnalysisTool() etc.
Table Manager	The Table Manager module provides functionality to define, store and access tabular information, e.g. rating curves and lists. Members of the public interface could include: • GetAllTables() • GetTableById() • GetTableCell() etc.
Hydro Object Manager	The Hydro Object Manager provides functionality to define, store and retrieve Hydro Objects as well as functionality to be associated with Hydro Objects displayed on a map. Other components (e.g., the Scenario Manager) utilize Hydro Objects. Members of the public interface could include: • GetAllHydroObjects() • GetHydroObjectTypes() • GetHydroObjectFromModel() etc.
Report Manager	The Report Manager provides functionality for creation, storing, display and printing of reports.
System Administration	The System Administration Module provides functionality to administer users and groups, defines studies, export/import data, and perform other system administration relevant tasks. Members of the public interface could include: • AddUser() • RemoveUser() • AssignUserToStudy() etc.
Scripting Manager	Scripting functionality – and the ability to store, to retrieve and to run script within the NB DSS is provided by the Scripting Manager.

Module	Description
Metadata Manager	The Meta Data Manager provides all modules and components with facilities to create, store and find meta data information on entities. The UI of the module includes a data explorer for meta data. Members of the public interface could include: • GetMetadataForEntity() • GetMetadataForParentEntity() • GetMetadataForChildEntity() • AddMetaData() etc.

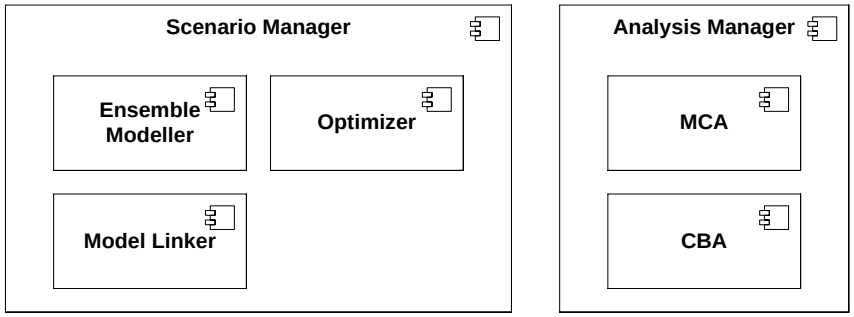

Figure 5.9 Sub-components (UML)

The Scenario Manager module and Analysis Manager modules are separated into more than one component, as seen in Figure 5.9. These sub-components are listed in Table 5.4.

Table 5.4 Module sub-components

Module sub-component	Description
Ensemble Modeller	The Ensemble Modeller component encapsulates the functionality required by the Scenario Manager to define and execute ensemble scenarios. This includes handling of ensemble time series for input and output as well as post-processing of outputs (e.g. aggregation, statistics). For a detailed description, see Section 5.2.1.
Optimizer	The Optimizer component provides configuration and execution functionality for the Scenario Manager to handle simulation based optimization scenarios. For a detailed description, see Section 5.2.3.
Model Linker	The Model Linker component assists the Scenario Manager in defining scenarios on multiple models as well as executing these linked models. For a detailed description, see Section 5.2.2.
MCA	The MCA component provides the UI, business and data access functionality with respect to multi-criteria analysis. The MCA component is a sub-component of the Analysis Manager.

Module sub-component	Description
CBA	The CBA is similar the MCA only encapsulating the Cost Benefit Analysis. The CBA component is a sub-component of the Analysis Manager.

5.1.2.4 The UI Components

The shell component is hosting and displaying UI components – as explained in Section 5.1.2.2 and Section 6.2.

The UI components are organized according to the modules they mainly serve – e.g. Explorers and Data Views for the Timeseries Manager, Explorers and Data Views for the GIS Manager and so on. They, however, also collaborate on the module level in order to provide coherent end-to-end functionality for the users. E.g. the Timeseries Explorer makes use of both the Timeseries Manager module and the GIS Manager module for associating time series with GIS features. Similarly, the Scenario Manager will - when creating a new scenario - use business functionality from both the Scenario Manager module and the Timeseries Manager module when providing the user a list of relevant input time series to choose from.

The use of modules from the UI layer is depicted in Figure 5.10.

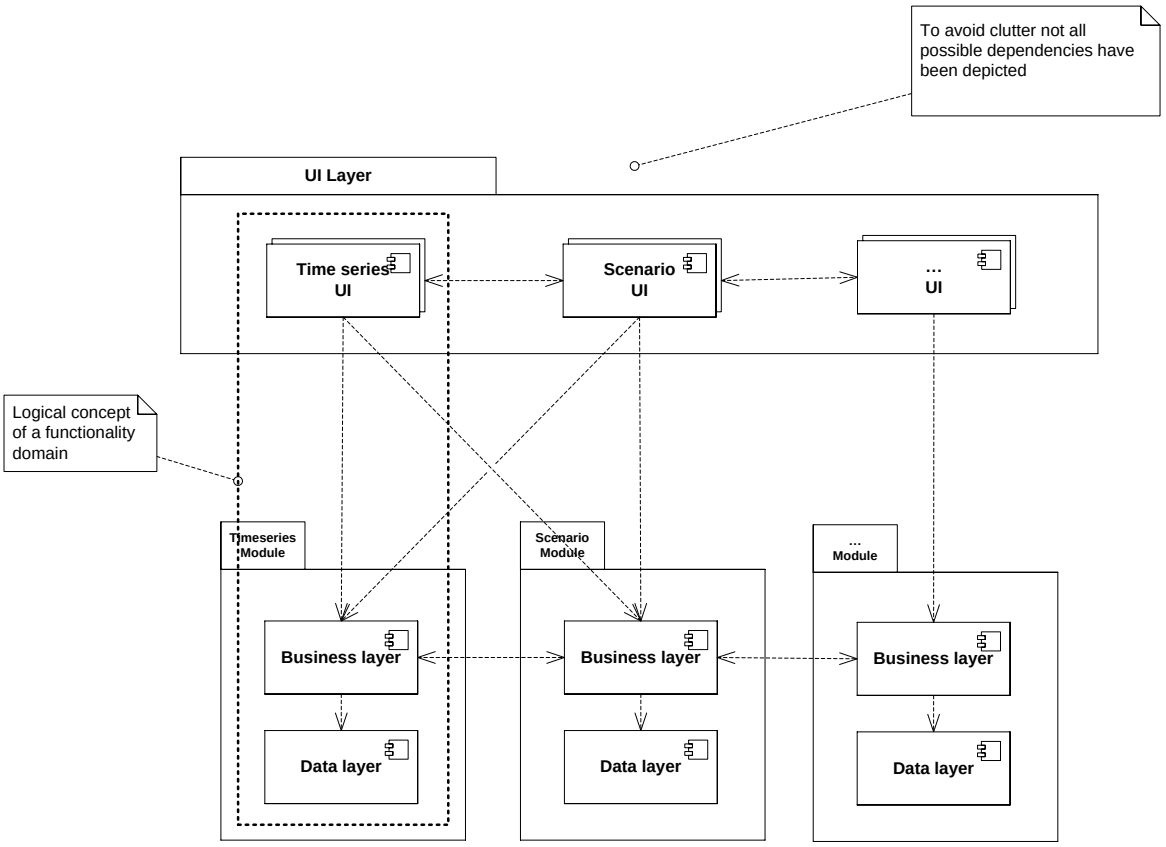

Figure 5.10 UI components and modules (UML)

Note from Figure 5.10.

- The combination of a specific module and its primary related UI components – explorers and data views - is called a functionality domain (hereafter just domain). E.g. the Time series domain constitutes the Timeseries Manager module and the corresponding Timeseries UI components.

- All UI components can make use of all modules.

- UI components can interact across manager boundaries.

There is a slight difference between interactions within the boundary of a manager and interactions that occur between managers. The former happens through direct component references while the latter happens through a factory mechanism. This is depicted in Figure 5.11

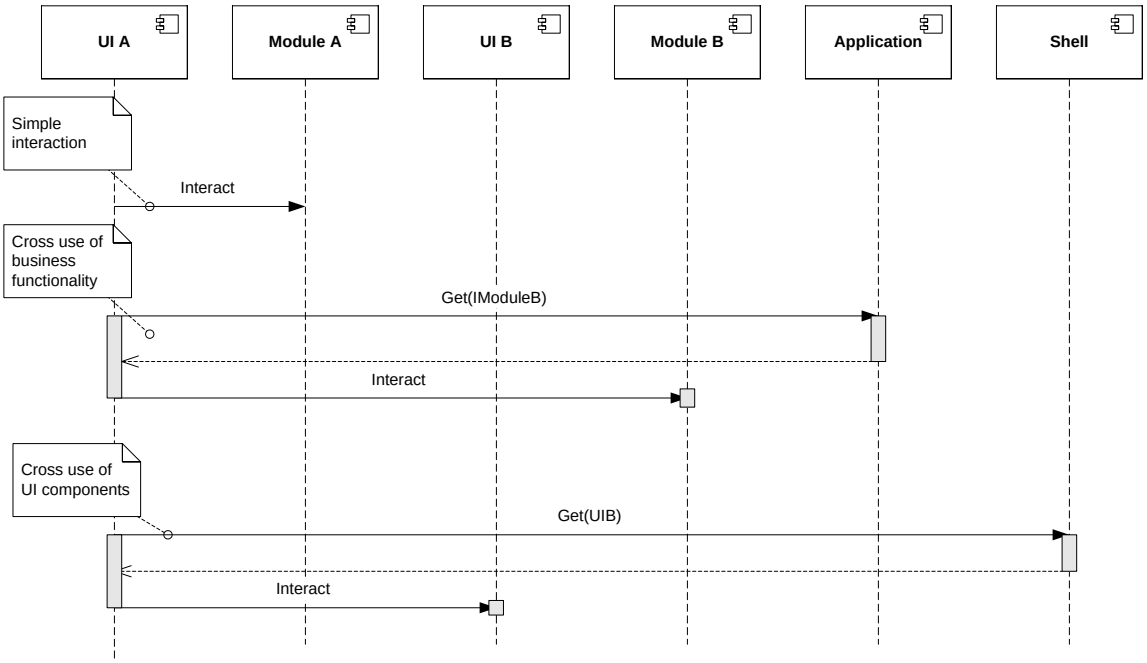

Figure 5.11 Cross-manager component referencing (simplified) (UML)

Note from the figure:

- A UI component can directly interact with the module within its own domain, but when interacting with a module in another domain, it will use the singleton Application component as factory. This provides for a loosely coupled system.

- UI components from different domains can also make use of each other through a factory mechanism. E.g. a Timeseries UI component will – when interacting with a GIS UI component – use the Shell as a factory for getting a reference to the object implementing a specified interface. This provides for making a loosely coupled system coherent.

- All object definitions occur through interfaces defined at a generic level where by all domains know the interface for all business objects (the module APIs). The loose coupling allows for different UI components as well as scripting using the business object at module level.

- •All module interfaces are defined at a generic level, and are thus available across domains. The loose coupling allows for different UI components, as well as scripting, to access modules in all domains.

5.1.2.5 The Tool Components

Tools target specific functionality areas, such as time series tools or GIS tools. From a software architecture perspective, Tool components are "cross cutting" components that are not associated with a particular module, but can be used by any module. Furthermore, tool components are normally lightweight, with much more limited and specific functionality than the modules.

All tools require some specific input data, have an execute method and deliver some output. For example the time series *Resample* tool expects a time series as input and delivers a re-sampled time series as output. Each tool implements the ITool interface, and similar to the modules, has the ability to register itself during installation.

The toolbox uses the .NET reflection mechanism to establish a knowledge of the types of data that the individual tools consumes and produces. This makes it possible to build sequences of tool executions where output from one tool is automatically made available as input for the next tool in the sequence. The toolbox will include specialized tools for displaying and persisting output.

This architecture provides a mechanism for developing custom tools catering to specific needs within an organisation or a project, while being seamlessly integrated with the NB DSS.

5.1.3 The Model Tools

In this section the Model Tool components and the interaction with the Front-end component will be described. The Model Tools are a collection of "off-the-shelf" software products integrated in the DSS Front-end using the Adapter pattern.

Most modelling tools - and certainly the DHI MIKE model tools - have a *UI* component and an *engine* component. The engine controls the model execution – i.e. solves the mathematical equations and processes. The UI is used to configure the model – that is creating the model setup.

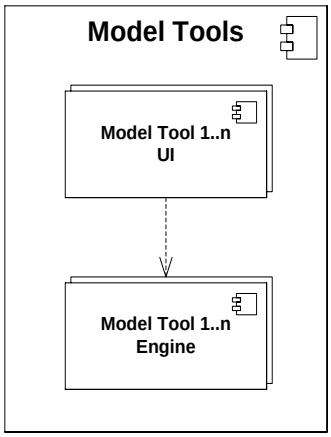

Figure 5.12 The Model Tools (UML)

The following section describes the use of model adapters. The description is generic and not biased towards the MIKE modelling tools.

5.1.3.1 The Model Adapters

Interaction between the DSS Front-end and the Model Tools is performed using the Adapters design pattern. Each Model Tool component includes a specific *model adapter* component that implements two interfaces: *IConfigAdapter* controlling the model setup registration and *IRuntimeAdapter* controlling the preparation and execution of a model simulation (see Figure 5.13).

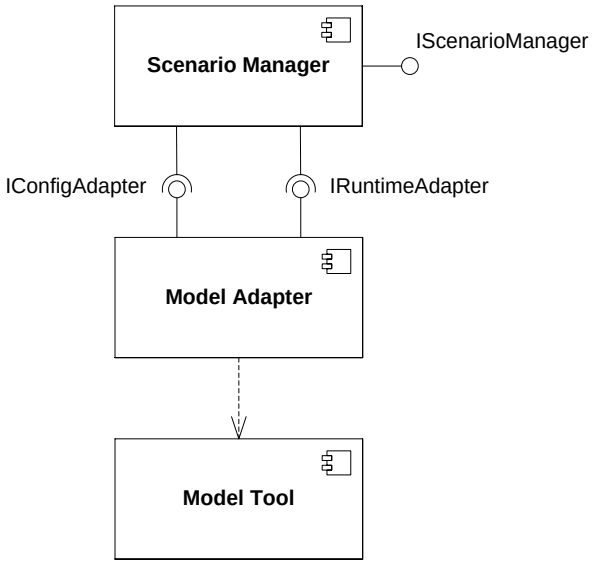

Figure 5.13 The model adapters (UML)

Adapters for the relevant DHI MIKE models will be developed as part of the NB DSS project, but the proposed architecture also enables integration with other model tools.

5.1.3.2 Model Setup Registration (IConfigAdapter)

Model setup registration in the NB DSS is performed using the IConfigAdapter interface of the model adapter as shown in Figure 5.14.

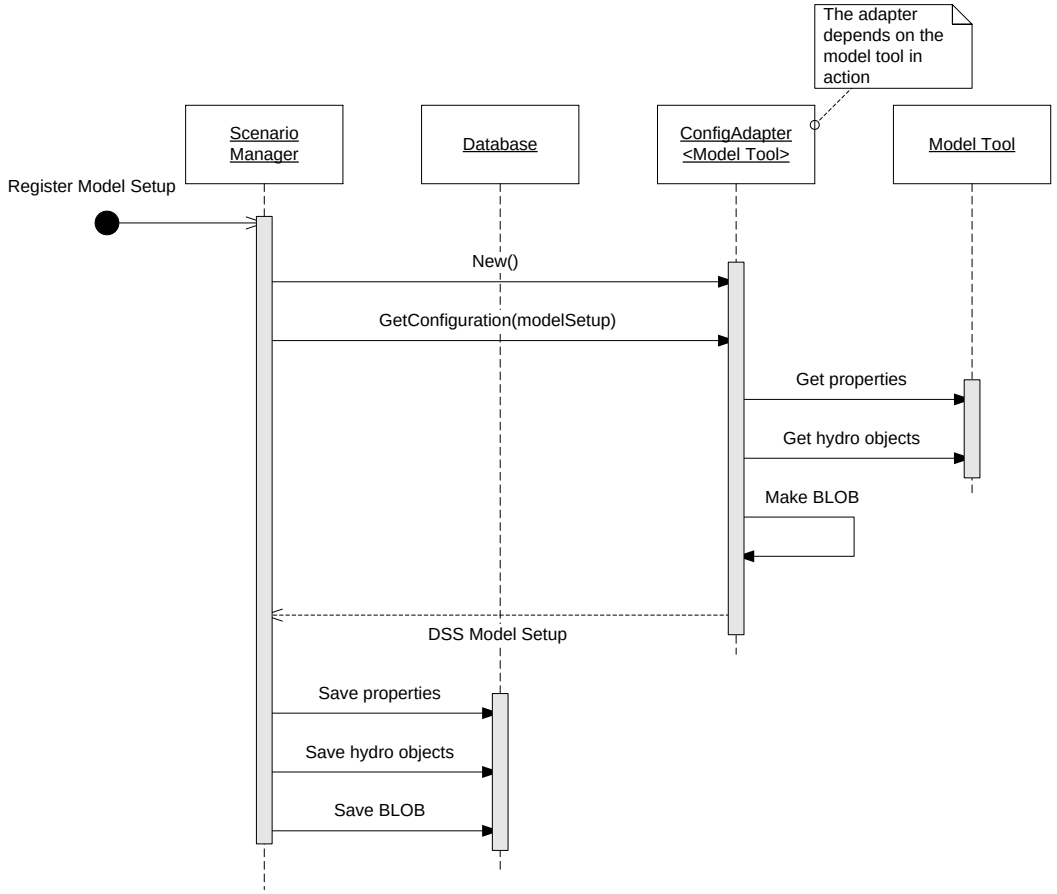

Figure 5.14 Register Model Setup (UML)

In this sequence,

1. The DSS Front-end component (The Scenario Manager) launches the model adapter and calls the GetConfiguration() method.
2. The model adapter understands how to communicate with the Model Tool in order to interpret the model setup and returns a data structure with the model setup information to the Scenario Manager.
3. The Scenario Manager stores the model setup in the database.

The *DSS Model Setup* returned from the adapter is the data structure representing the model setup in the NB DSS. This data structure will be defined in an interface (IModel-Setup) and will consist of the following groups of data:

- *BLOB*. A binary representation of the full model setup in the database. This is used to re-create the model setup when editing or running it with the Model Tools.
- *Properties*. These are the definition in the database allowing the system to run a Model Setup. They comprise:
    - Base data such as name, Model Tool reference, description etc.
    - A set of time series data or parameters required to run a simulation. This data will be stored explicitly in the data base. For instance:
        - Model input and output time-series.
        - Parameters such as start and end of simulation.
        - References to log-file for examining the Model Tool performance.
- *Hydro Objects*. Hydro objects are model setup features such as hydraulic structures, catchments and river reaches. An agreed upon set of Hydro Objects will be available in the DSS Front-end and stored explicitly in the database.

The model setup in the NB DSS is depicted as data flow in Figure 5.15 below.

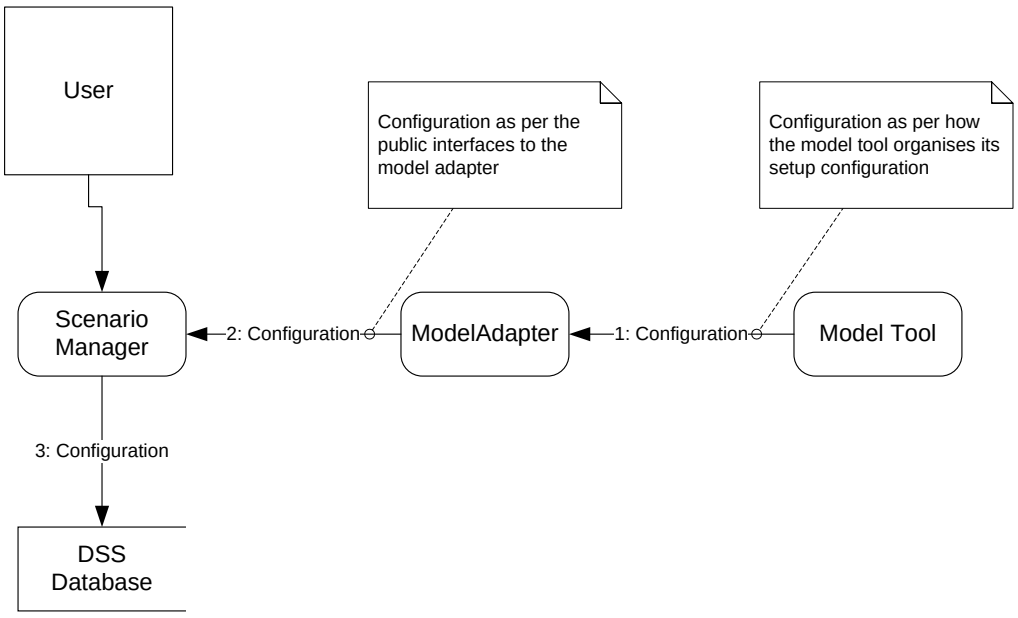

Figure 5.15 Data flow when registering a model (Gane-Sarson)

Note from the figure:

- The flow from the Model Tool to the Model Adapter is based on an agreement between the 2 components
- The flow from the Model Adapter to the Scenario Manager is based on the public IConfigAdapter interface, which uses the DSS Model Setup data structure to exchange model configuration information

The type of interaction between the model adapter and the model tool – i.e. how the information is exchanged between the model adapter and the model tool - depends mainly on the capabilities of the model tool. If the model tool provides an API for getting and setting properties of a model, then the model adapter will likely make use of such an API. In the case where a model tool does not provide such an API, the model tool will have to directly process the input files constituting the model, e.g. through text processing.

Model adapters for MIKE models will most likely make use of the latter approach because few of them provides an API sufficiently rich for getting and setting all relevant properties of the models.

When registering a model, the user will be presented a Model registration wizard. This wizard will guide the user through the registration process which constitutes of selection of model tool, selection of input file(s) constituting the model – either by browsing the file system or communication through the model tool – and finally mapping between input time series and GIS features.

The NB DSS allows the user to edit the topology of a model through the use of the model tool UI. This happens through an export of the model to the file system and subsequent launch of the model tool. After having edited the topology, the user can update the model stored in the database by re-register the model. The re-registration is similar to registration but more light-weight because the database already contains information about the model.

5.1.3.3 Scenario Simulation (IRuntimeAdapter)

Execution of a scenario simulation in the NB DSS involves preparing a model setup for simulation, executing the Model Tool and extracting selected simulation outputs to be stored in the database. This is done using the IRuntimeAdapter interface of the model adapter as illustrated in Figure 5.16.

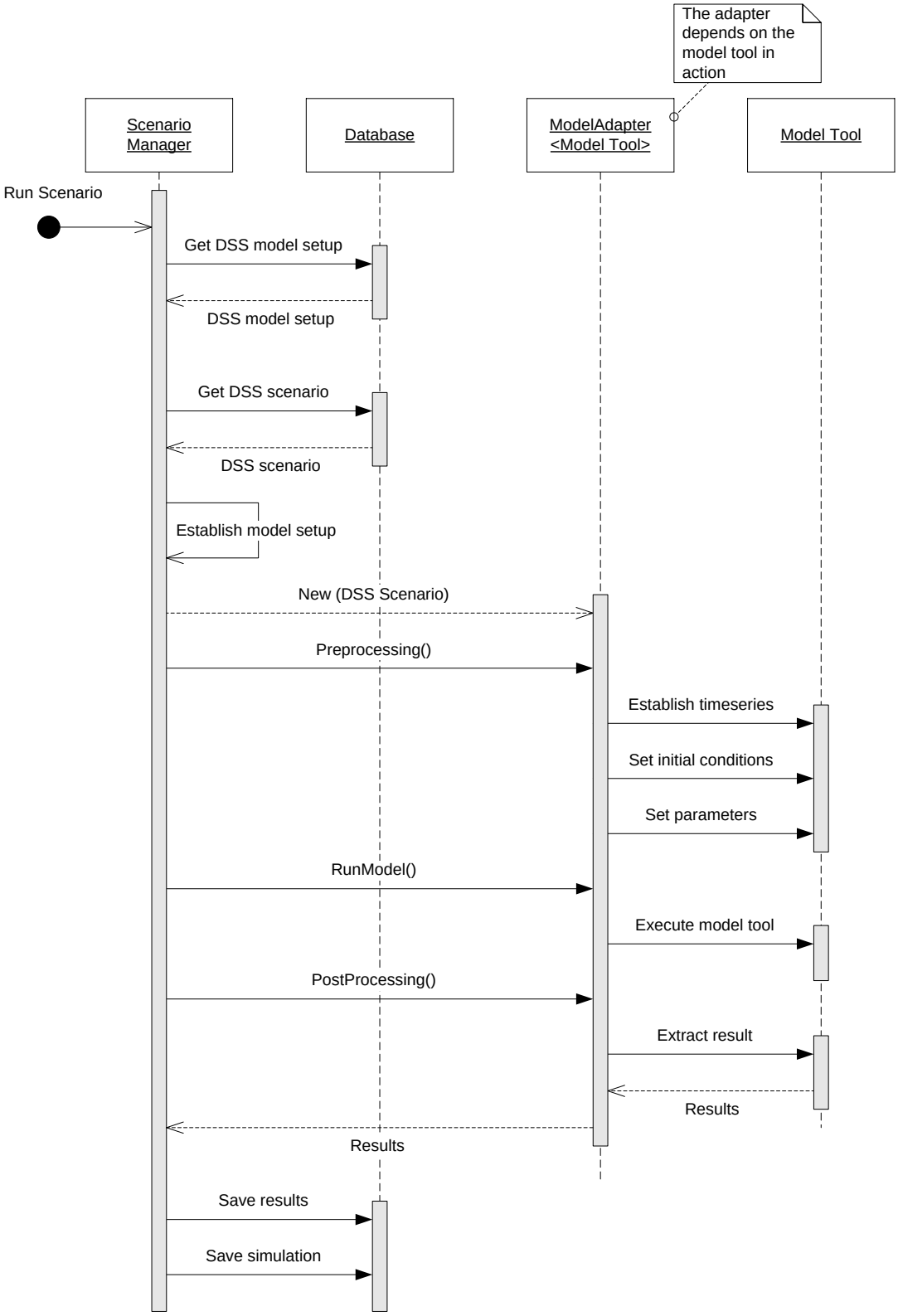

Figure 5.16 Run Scenario (UML)

The figure shows that

1. The DSS Front-end (the Scenario Manager) retrieves the model setup and the variations to the model setup (DSS scenario) from the database.
2. The Model Adapter – on behalf of the Scenario Manager - establishes the model setup (unpacking the BLOB and establishing a directory structure for the model tool and adapter to work within), the assumption being that the model tool is file based and exchange of data via the adapter is done at file level. This happens by having the Scenario Manager launching the model adapter with the selected DSS scenario and calls the Preprocessing() method.
3. The model adapter knows how to establish the input time series, the initial conditions and the input-parameters of the variations (the DSS scenario) in the simulation folder.
4. The RunModel() method is called on the model adapter which knows how to execute the Model Tool.
5. The PostProcessing() method is called on the model adapter which knows how to extract the model results and to write them to the simulation folder.
6. The PostProcessing() method is called from the model adapter to instruct it to extract the model results from the proprietary format of the model tool. The Scenario Manager may further post-process the results as defined in the scenario (not shown in figure)

The execution of a scenario within the NB DSS is depicted as a data flow in Figure 5.17.

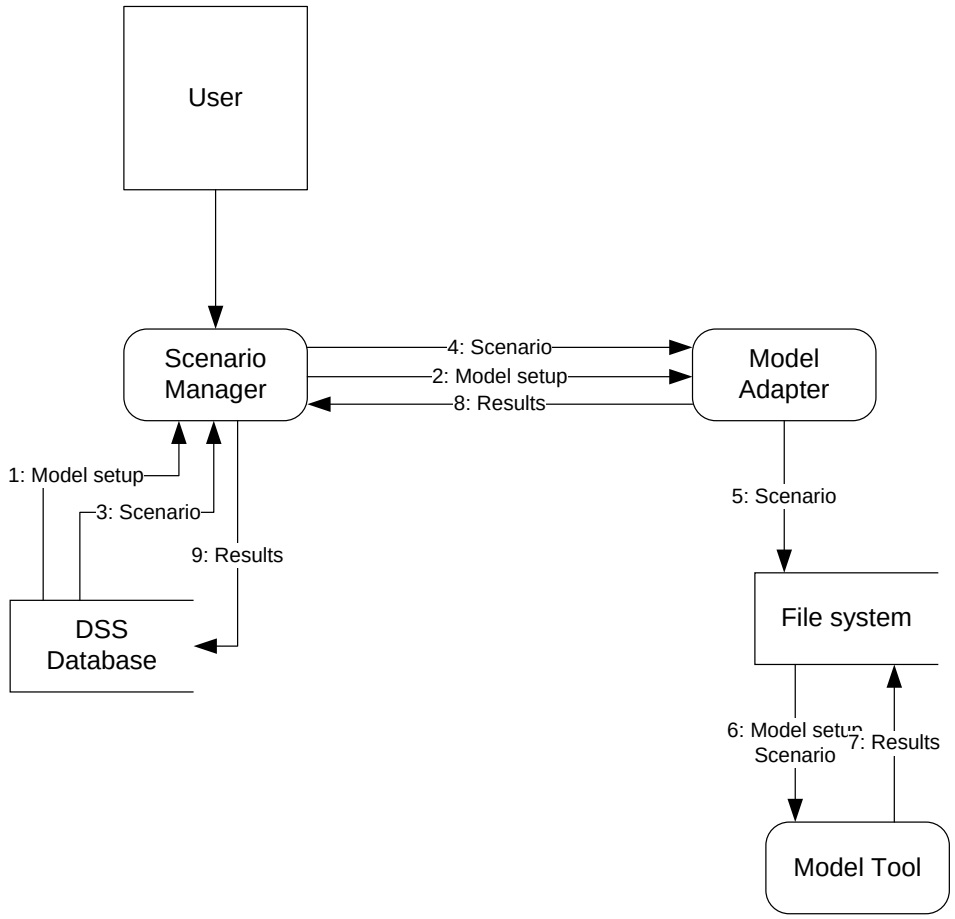

Figure 5.17 Run scenario data flow (Gane-Sarson)

Note from the figure:

- The flow from the Scenario Manager to the file system goes through the Model Adapter, i.e. the NB DSS does not itself interact with the file system. Communication consists of unpacking a BLOB[1], containing the model configuration and loaded from the DSS Database, to file system folder.

- The flow from the Model Adapter to the file system follows the custom agreement between the Model Adapter and the Model Tool

- The flow from the Scenario Manager to the Model Adapter uses the public IRunAdapter interface

Table 5.5 below briefly describes the mentioned interfaces:

Table 5.5 Model Tool components interfaces

Interface	Description

[1] Could be implemented as a zip-folder

Interface	Description
IRuntimeAdapter	Interface with methods for executing the Model Tool. Has methods such as: • PreProcessing() – for preparing the model setup by for instance convert input time series into model tool proprietary format, prepare initial conditions into model tool format, and alter model tool setup with respectt to simulation period setting. • RunModel() – for executing and controlling the model tool and verifying success • PostProcessing() – for retrieving outputs from the model tool proprietary format to be returned to the DSS
IConfigAdapter	Interface with methods for retrieving the necessary information for establishing a DSS model setup. Has public methods like: • GetConfiguration() – parsing a model setup (by files or by using an API if available) to retrieve all relevant configuration data to be used by the DSS: input time series, initial conditions, potential output time series, other output data, log files • GetModelSetup() – returns to the DSS a byte stream of data (probably a zip-file) with all data (files) required for enabling restore of the model setup for later editing and execution. • RestoreModelSetup() – will restore the model setup to the form required to edit the content with the Model tool
IModelSetup	The data structure representing a model setup in the NB DSS. It has properties like: • InitialCondition – adapter defined structure (bytestream) for exchangin initial conditions to/from the model • ListOfTimeSeries – each time series representing a model input time series. A time series include a full specification of the name, type and unit and location • ListOfHydroObjects – each Hydro Object a representation of parameters in the model setup • ListOfOutputTimeSeries – each representing selected outputs from a simulation

5.1.3.4 Linking of models

NB DSS supports dynamic linking – time step by time step - of models through the adapter pattern. A dynamically linked model is a model that makes use of 2 or more model tools that exchange data during the simulation. This in contrast to sequentially linked models where one model tool simulation provides the input for the next model tool simulation

OpenMI is a standard for how to enable such model tool integration – also across model tools from different vendors. A model tool is OpenMI compliant if it just implements one OpenMI integration interface; but one interface is typically not sufficient in order to communicate all kinds of data among the integrated model tools. The MIKE tools do not all have full support for OpenMI (MIKE11 and MIKE SHE are OpenMI compliant while MIKE BASIN is not). Although two OpenMI models may be linked, it is by no

means a trivial task. It typically requires a substantial effort and expert knowledge about the internal part of the model tool including the numerical solution of the governing equations.

Seen from the NB DSS point of view there is no difference between a dynamically linked model and a "single" model. In both cases there has to exist an adapter that the NB DSS can use in order to read and write model setups, run simulations etc.

In the case of a dynamically linked model, the model adapter needs to communicate with more than one model tool. This can be implemented in the model adapter by aggregating the model adapters for the model tools involved in the linked model. I.e. the adapter for the linked model makes use of the adapters for the individual model tools.

5.1.3.5 Model adapter pattern

The model adapter pattern as described in the previous sections is the most common way for a DSS system to interact with model tools. It is used in products like MIKE FloodWatch by DHI, FEWS by Deltares and HEC-WAT by US Army Corps of Engineers.

The advantages of the patterns are

- Virtually all type of model tools can be integrated with the ND DSS. As also requested in requirement 2.2.1.3
- Seen from an architectural perspective there is a clear separation of concerns between the NB DSS and the modelling tools
- Model tool developments (enhancements) can be leveraged from NB DSS without having to change to the NB DSS system

Potential disadvantages comprise

- The user will have to be accustomed to more than one user interface, i.e. that of the NB DSS and those of the native modelling tools
- The user will in some cases experience a slightly more complicated work process than if the model tool UI was directly incorporated in the NB DSS.

The NB DSS supports different types of simulations. All of these are well supported through the adapter approach

- Optimisation – The IConfigAdapter interface of the model adapter provides methods that the NB BSS system can use in order to obtain the list of parameters that can be used in order to perform optimization. This is discussed in more details in Section 5.2.3
- Ensembles – The IRuntimeAdapter interface of the model adapter provides optional methods that the NB DSS system can use to optimize the performance of the whole system when running ensemble-based simulation. E.g. limiting the system overhead by restoring models to the file system. This is discussed in more details in Section 5.2.1

- Linked models – Seen from the NB DSS this is merely execution of models in sequence, i.e. the role of the adapter is no different from the standard case. Linking of models is discussed further in Section 5.2.2

- Rule-based simulation (e.g. controlling the simulation from a script) – The NB DSS scripting interface provides methods for restoring models stored in the database to the file system and launching model simulations. I.e. the adapter approach does not negatively affect rule-based simulation.

The only alternative to the adapter approach is either limiting the model tools to those having an API for setting up and running model or directly embedding the model tool's UI in the NB DSS. The latter approach would not only be costly but also affect the future development. Development, testing and maintenance overhead would become very high as a significant amount of coordination is required towards model tool vendors.

With respect to the MIKE model tools as of today the only alternative to the adapter approach is UI embedding. None of the model tools provide an API rich enough for the NB DSS to use as a substitute for the adapters.

5.1.4 The Database

The *Database* component is a RDBMS prepared for handling all types of DSS data – including GIS (spatial) data, time series data and scenario/model data.

5.1.4.1 Data Categories

Each Module is responsible for handling its own well-defined category of data. For example the Timeseries Manager component handles all time series data, and the Scenario Manager component handles all scenario data. This means, that if the Scenario Manager has to retrieve some time series values – for example if it wants to run a scenario simulation – it cannot retrieve these data directly from the database, but has to ask the business services of the Timeseries Manager to retrieve this data as shown in Figure 5.18.

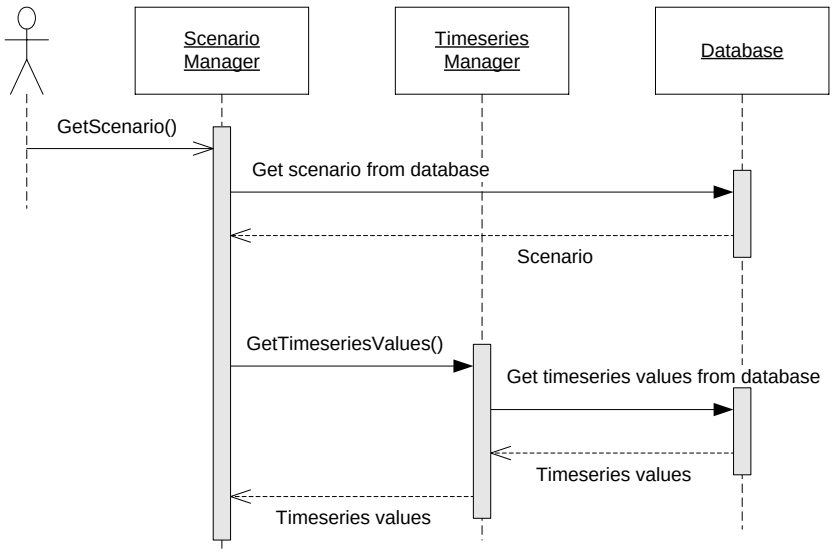

Figure 5.18 Accessing data from another module (UML)

Identified data categories are listed in Table 5.6.

Table 5.6 Data categories

Data Category	Description	Handled by Module
GIS data	Also called spatial data. The spatial data is a collection of feature classes and rasters. Examples of feature classes are countries, cities, rivers etc.	GIS Manager
Time series data	Time series are sequences of measurements in time. Examples of time series are discharge, water level, rainfall etc.	Timeseries Manager
Model data	All data associated with a model setup, i.e. the basic simulation properties, the BLOB, and the Hydro Objects.	Scenario Manager
Scenario data	All the data related to the definition of scenarios, i.e. definitions of linkages between a model setup and the input and output data involved in model execution.	Scenario Manager
Simulation data	All the data related to an executed simulation (references to model and scenarios, input time series, output times series, other input and output data, initial conditions)	Scenario Manager
Configuration data	Analysis components will store configuration data for performance indicators as well as references to configured analysis tools, e.g. MCA and CBA. Each tool will provide its own data model for configuration data	Analysis Manager
		MCA
		CBA
	Hydro Objects are considered configuration data which are not linked to any special functional components (although used by at least GIS and Scenario Manager). All definitions for Hydro Objects are stored in the database.	Hydro Objects Manager
	Tabular data managed by the Table Manager may be used in several places in the other components and modules. All data associated with a tabular structure managed by the Table Manager are stored here.	Table Manager
Reporting data	Documents and reporting configuration data	Report Manager
Scripting data	Scripts	Scripting Manager
Meta data	Variable descriptive data for different entities in the system and time-varying history information describing the life-line of these entities.	Metadata Manager
System data	Configuration information on system installation components (e.g. computers), users, groups, studies and otherwise non-domain specific information.	System Administration

5.1.4.2 Data Entities

Drilling down from the high-level data categories, the data can be described as data *entities* in a conceptual data model. In this section the most important data entities are described into more details.

GIS features

The GIS data is a collection of feature classes (countries, cities, rivers etc.). Each feature is characterized by having an attribute of type geometry. In the database, the geometry attribute will be handled by a specific PostGIS compliant geometry data type.

PostGIS uses a system table named `geometry_columns` to store the meta-data associated with the geometry columns in the database. PostGIS automatically creates this table. The `geometry_columns` table provides housekeeping information about geometry columns in the database, and is used by NB DSS to gather a list of geometry layers in the database. This implies that PostGIS (like most other GIS systems) operates with a flexible physical data model implying that new feature classes result in new physical table in the database. This is illustrated in Figure 5.19

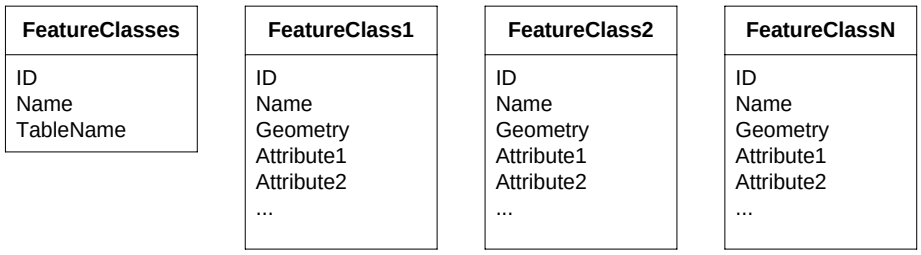

Figure 5.19 Conceptual data model for GIS features (UML)

This design is a challenge for the logical design as the classes encapsulating the data store vary with time, or rather must be dynamic with respect to properties. I.e. it is difficult to make an object relational mapping of feature classes.

Furthermore, the ability to establish different feature classes dynamically makes it difficult to establish a common standard for how a certain type of features is described and named, which may lead to confusion and lack of transparency on the part of the user.

An alternative design of the GIS data model could be based on a separation of geometries and attributes. E.g. by creating one table per base line geometry (point, line, polygon and later on raster) and then either associate set of feature attributes with the geometry or vice versa. This would imply that the DSS Front-end Data Access Layer should create the data access object using an object creational pattern like abstract factory.

The latter design probably could enforce a stronger control on the data added to the database - and later synchronized among installations. It might also help in reducing the size of the database. The down-side is a more complex design and implementation mainly because the support provided by PostGIS cannot be leveraged.

The final design on the GIS data model shall be taken during the detailed analysis and design stage in development cycle 1.

Time series

Time series data are separated into time series descriptions in the Timeseries table and the actual data in the TimeseriesValues table. The conceptual data model is depicted in Figure 5.20.

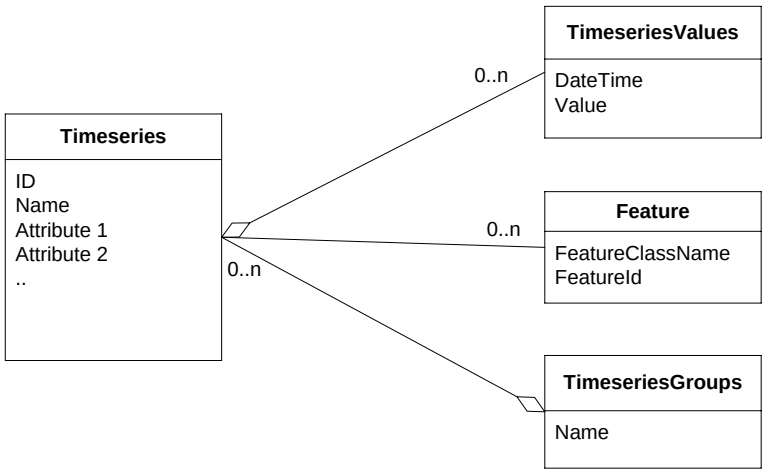

Figure 5.20 Conceptual data model for Time series (UML)

Note from the figure:

- A time series can be associated with one or more features, i.e. linked to GIS entities
- Time series can be grouped together, e.g. to support ensembles of time series

Because of the potentially large amounts of data, time series data need special considerations and solutions compared to other data. This implies that time series values – as opposed to the time series properties - have to be "lazy loaded", i.e., they are only read from the database when they are needed for visualization or processing.

Scenarios

Scenarios represents the linkage between a model setup and the input and output data involved in model execution. Scenarios have a reference to a model setup, and to one or more time series and time series groups.

Figure 5.21 depicts the conceptual data model for Scenarios.

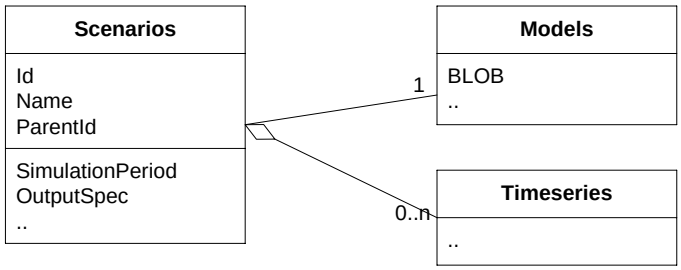

Figure 5.21 Conceptual data model for Scenarios (UML)

Note from the figure:

- The ParentId attribute in the Scenarios entity allows the UI to display scenarios – or rather their names – in a hierarchical way.
- The scenario is associated with a model, i.e. the entity holding the model setup data
- The scenario can be associated with time series
- The lower compartment of the scenario entity contains specific scenario settings

Meta data

Meta data exists for (almost) all entities in the system. A common component exists to handle the Meta data requirements for all other components. This will tie Meta data to entities via a central Entity table common for all components.

Meta data in the NB DSS consists of a list of key-value pairs associated with an entity. The key is defined in the Meta data system itself to allow for a consistent, language dependent terminology across the system. The values are for (relatively) simple types, such as text, numbers, date and time, references to other entities in the database and enumeration (for classification purposes). The conceptual data model for the definition part of the Meta data system is shown in Figure 5.22. Allowed values for enumerations are either kept in a separate table or de-normalised into an info field with the key definition as shown.

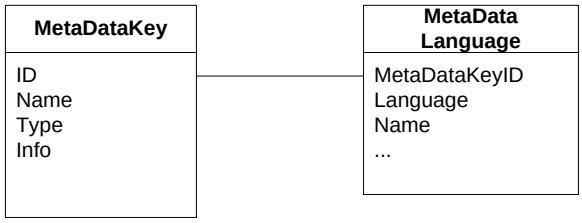

Figure 5.22 Conceptual data model for Meta data definition (UML)

Meta data can potentially be both static and dynamic description of data entities. Dynamic descriptions evolve over time. To support this, the meta data model has been split into two parts: a static part and a dynamic part, see Figure 5.23.

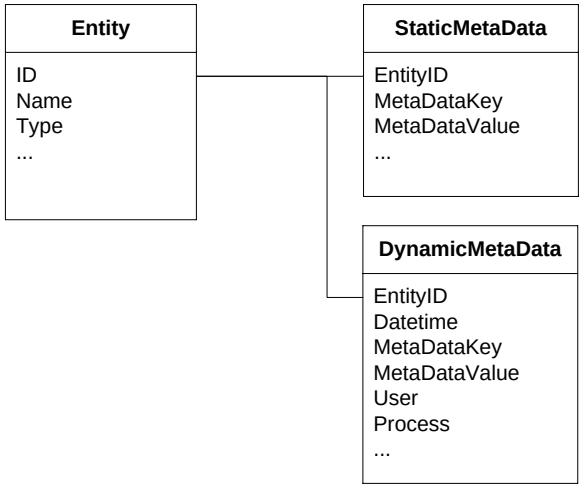

Figure 5.23 Conceptual data model for Meta data (UML)

The above design is similar to the pivoting design used for managing key-value pairs described in /5/. It is recommended to use this pattern when "(i) performance requirements impose it, (ii) schema modifications are rare, and (iii) the application code is appropriately stored and documented in such a way that maintenance is guided from an organized repository". The NB DSS will use Meta data across the entire application in a consistent manner with possibilities to extend keys and functionality (for display etc.), which match these arguments.

Hydro Objects

Hydro Objects represent model specific representations of information on various types of physical objects such as reservoirs and flow structures. Different modelling tools have their own custom way of representing these layouts. The NB DSS system supports interfacing with new modelling tools over time which implies that the representation of hydro objects cannot be bound to a fully static relational data model.

Figure 5.24 and Figure 5.25 depict two possible ways of implementing a dynamic data model for Hydro Objects.

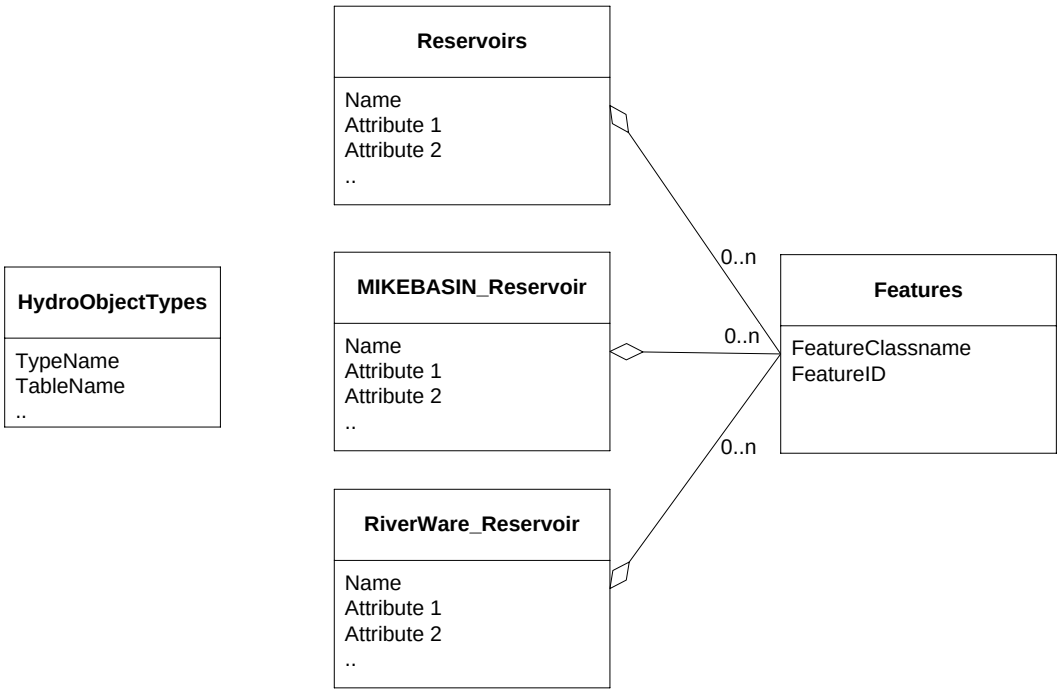

Figure 5.24 Conceptual Hydro object data model based on the GIS pattern (UML)

The solution shown in Figure 5.24 is modelled after a data pattern typically used in GIS solutions, i.e. creating new tables for new types of features (feature class), in this case a table for each new type of Hydro Object.

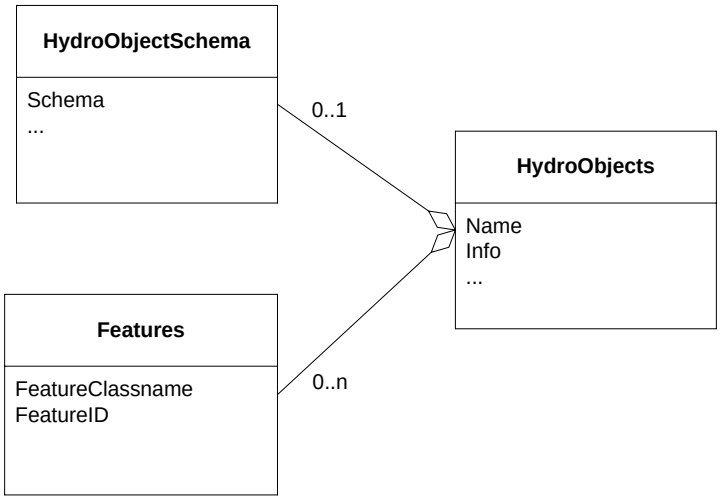

Figure 5.25 Conceptual Hydro object data model based on a Schema pattern (UML)

Figure 5.25 shows a solution where all hydro objects are stored within a single table. Here the actual hydro object properties are stored as an aggregated data field, here named Info. This field could be an XML string, a persisted .NET object or a reference to a table[1]. The Schema validates the content of the Info field.

[1] Table in this context does not refer to a database table; but to the logical table entity managed by the Table Manager, see the sub-section about Tables in Section 5.1.4.2

The Feature table in Figure 5.24 and Figure 5.25 allows Hydro Objects to be linked to GIS features.

The final decision regarding the Hydro Objects design will be made during the detailed analysis and design phase.

MCA

The configuration of a MCA is stored in the database with internal configuration parameters as well as definitions of indicators, references to time series, and data tables.

As with Hydro Objects the data model to use for the MCA can be very simple with respect to tables (e.g. the Schema pattern) or more explicit with tables for MCA properties and parameters and direct references to time series, tables, indicators.

In favour of an explicit data model speaks the fact that all MCA configurations follow a similar pattern: making a matrix of values and weights with references to internal or external performance indicators and criteria.

The fact that the MCA tool may evolve over time favours the Schema pattern, allowing encapsulation of the data model of the MCA tool and the possibility to extend the functionality by simply deploying a new version of the tool without having to alter the database schema itself.

Final decision on the design should be made during the detailed analysis and design phase.

CBA

CBA carries many similarities with MCA: It is complex tool in itself, uses similar references to input data and keeps similar internal parameters as part of the configuration. As such it should employ the same pattern for storing CBA configuration in the dataset as does the MCA tool.

Tables[1]

Tables are pieces of data – typically scalars - to be stored and used by other processes. Each Table - including its content - can be referenced by other components through the public interface of the Table Manager module, e.g. getting a whole Table or a single value within a table. Typically Tables are used to collect groups of related single values, e.g. indicators, as a transfer mechanism when using output from one tool as input for another tool or for storing semi-structured data.

Logically speaking the Tables are similar to a spreadsheet in the sense that the user can group related information in one matrix; e.g. indicators calculated on basis of all output time series from a scenario simulation or indicators calculated on the same output time series calculated in a number of different scenarios.

The physical data model behind Tables can be established in different ways

- Through a static data model supporting scalar values or

[1] Not to be confused with database tables

- By storing a group of scalar as an entity (as a Matrix) using the Schema pattern which could be mapped to a object model through an abstract factory mechanism.

The static data model is depicted in Figure 5.26.

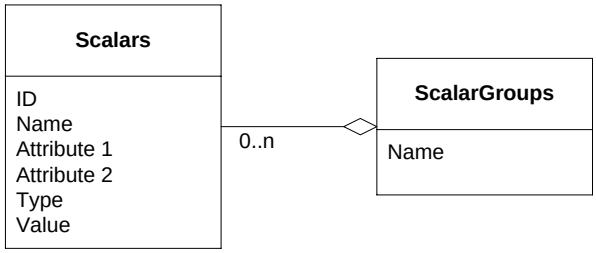

Figure 5.26 Table data based on a static data model (UML)

Here the grouping of the values is established through the ScalarGroups entity.

The Schema pattern based data model is depicted in Figure 5.27.

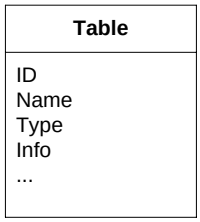

Figure 5.27 Table data based on a Schema pattern data model (UML)

Here the matrix of scalars is stored as en entity itself in the Info field. The Table Manager uses the Type field to serialise and de-serialise the matrix, e.g. through a schema or directly by the data of the Tables object. This solution has the advantage of supporting truly unstructured data, e.g. reservoir operational rules directly as supplied by reservoir management organisation and it also provides for storing of new types data relations without having to alter the underlying database structure.

Final decision on the design will be made during the detailed analysis and design phase.

5.1.4.3 Data Compartments

The total database schema will be partitioned into distinct data compartments. Each data compartment will on a one-to-one basis support specific studies as conceptually depicted in Figure 5.28.

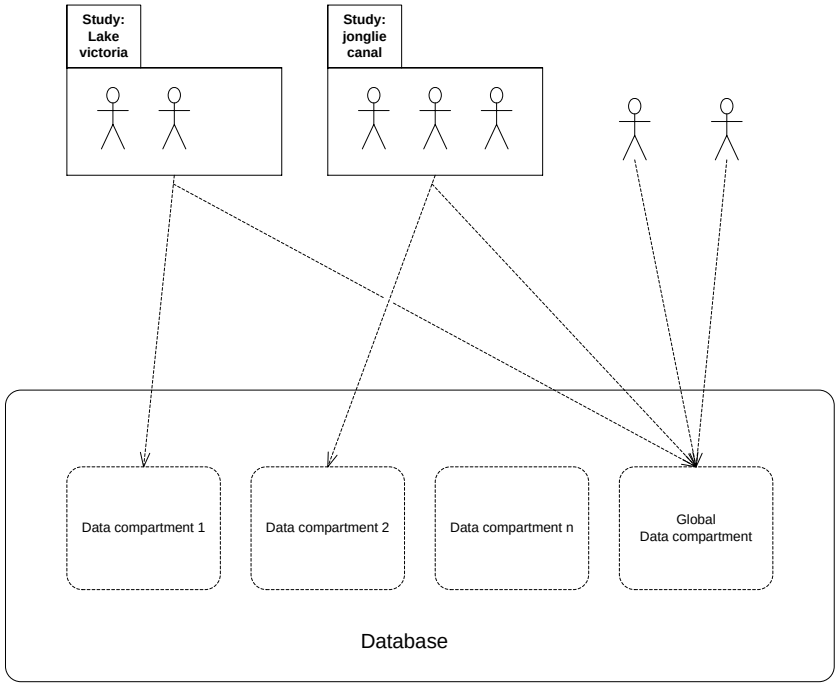

Figure 5.28 Data compartments and studies

Note from the figure:

- Users that are not associated with a study; i.e. have not selected a study when logging on to the system, will only have access[1] to data within the global data compartment.
- Users that are associated with a study; i.e. have selected a study when logging on to the system, will have access[1] to the data compartment associated with the study and the global data compartment.

One compartment is a single security realm. I.e. user John is treated different when logging in to different data compartments. E.g. John can be Study Lead in the Victoria lake study and Study Reviewer in the Jonglie canal study.

The advantages of the above model are simplified access control settings and a simpler role model compared to a solution where all data entities had to be configured independently.

Partitioning of the database into compartments can be done in different ways, e.g. physical partitioning where separate table space is created per study or as a logical portioning where the partitioning happens based on either a data entity attribute or on a meta data setting (pivoting data model pattern) Figure 5.29 and Figure 5.30 below conceptually illustrate how the logically based partitioning could take place.

[1] The type of access, e.g. whether updates are allowed, depends on the role of the user within the compartments

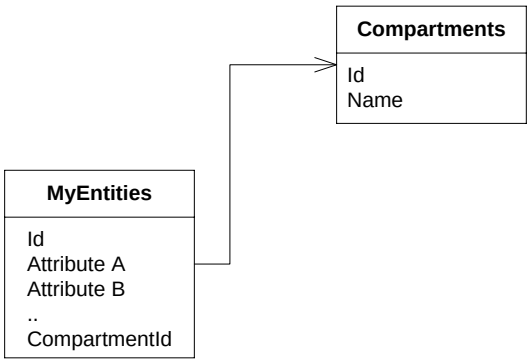

Figure 5.29 Logically based partitioning (UML)

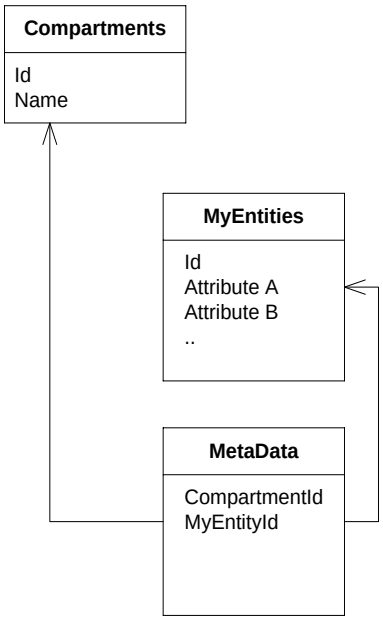

Figure 5.30 Logically based partitioning based under the Pivoting data model pattern (UML)

Note from the figures:

- The design in Figure 5.29 is based on a foreign key to the Compartments table
- The design Figure 5.30 is based on a Pivoting pattern defined in /5/.

The selection of the final design will be made during the detailed analysis and design period.

5.1.4.4 Data Integrity

It is likely that multiple versions of the database will be distributed for offline use to multiple parties – ex. multiple regional offices. In such a case, the various databases will occasionally have to be merged. Therefore, it is not feasible to let the database make auto-generated sequences of primary keys. Instead the Global Unique Identifier (GUID) data type will be used as the primary key of database tables.

The RDBMS will be used for maintaining data integrity. For example unique indexes, foreign key relationships, cascading delete options and other constraints will be used to maintain data integrity.

5.1.4.5 Data Access Pattern

The communication between the DSS Front-end and the Database is done in the individual modules in a data access layer using the *Data Access Object* (DAO) pattern. In this pattern, a DAO component defines an interface to persistence operations (CRUD)[1] and finder methods) relating to a particular entity.

The data access layer of each module includes a number of *DAO components* implementing the generic interface IDAO<Entity>, each handling the CRUD operations for a particular entity. In addition to the methods in the IDAO<Entity> interface, each DAO component can implement additional methods in a DAO-specific interface (I*EntityName*DAO).

Each *entity* component implements the IEntity interface. Furthermore, in the entity-specific interface I*EntityName*, they will provide all the properties of that particular entity. See Figure 5.31.

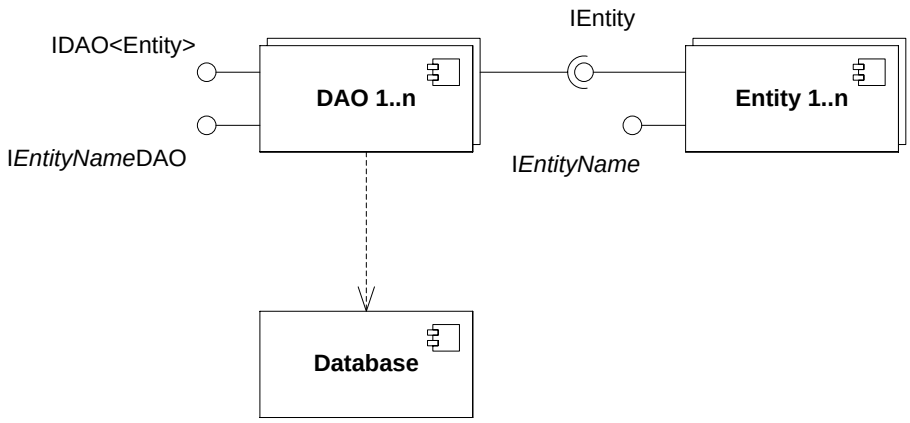

Figure 5.31 Data access through the DAO pattern (UML)

In Table 5.7, all the mentioned interfaces are shortly described.

Table 5.7 Data access interfaces

Interface	Description
IEntity	Interface that all Entity components implement. Includes at least an ID property of type GUID.
I*EntityName* (e.g. ITimeseries)	Entity-specific interface with all the properties of that particular entity.
IDAO<Entity>	A generic interface that all DAO components implement. The interface comprises methods such as Create(), Get(), GetAll(), Update() and Delete() for handling the basic CRUD operations for a particular Entity component.

[1] The acronym CRUD refers to the four basic operations in persistent storage: Create, Read, Update and Delete

Interface	Description
I*EntityName*DAO (e.g. ITimeseries-DAO)	The DAO-specific interface with supplementary methods not in the generic IDAO<Entity>. For example, ITimeseries-DAO.GetTimeseriesValues().

5.2 Modelling Components

This section gives provides details about the modelling components Ensemble Modeller, Model linker and Optimizer.

5.2.1 Ensemble Modeller Component

An ensemble scenario is identical with a standard scenario except that the input time series are not single time series but groups of time series – or ensembles. I.e. the boundary conditions are not specified as a single time series but through an ensemble.

Management of ensemble based scenarios is handled in concert by the following 2 software components:

1. The Timeseries Manager which provides functionality through tools or import for establishing ensembles. See the description of time series in Section 5.1.4.2.

2. The Ensemble modeller which provides functionality for looping over ensembles and execute the simple time series based scenarios and subsequent to the simulation add output time series to output ensembles

3. The Scenario Manager which provides functionality for managing scenarios.

Figure 5.32 below depicts the flow for creating an ensemble based scenario.

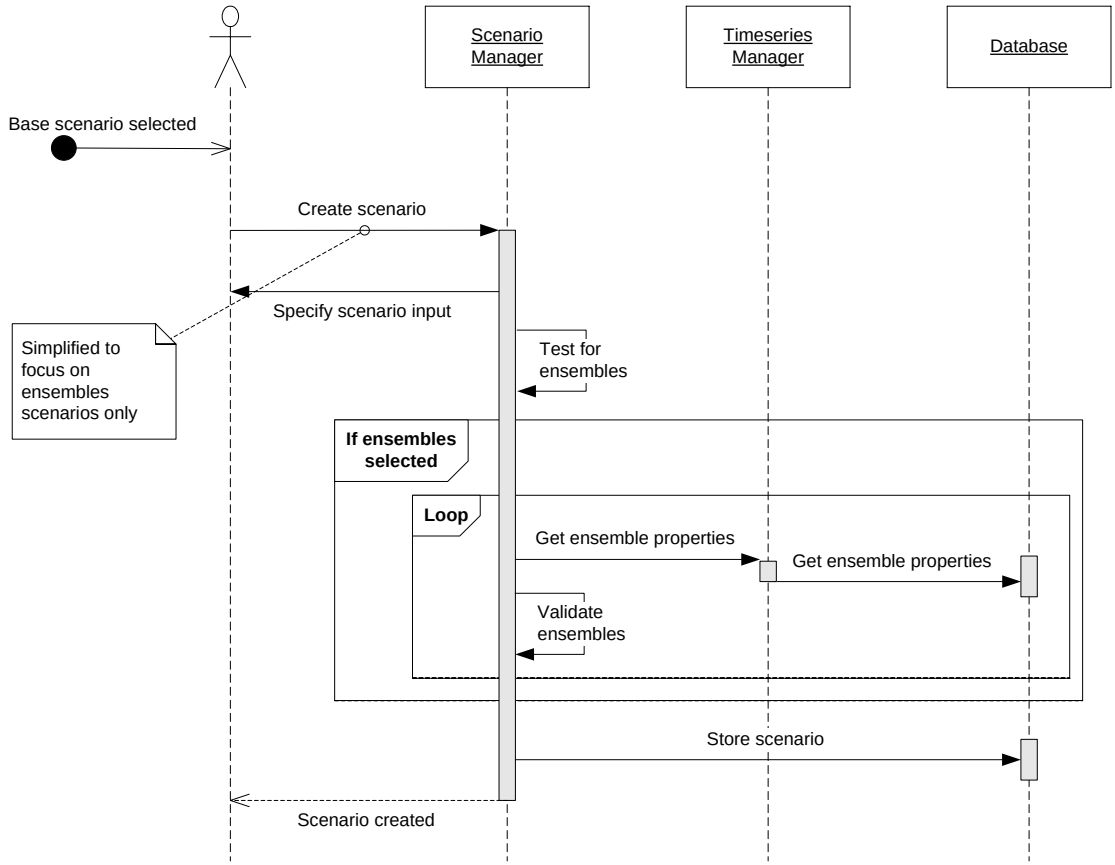

Figure 5.32 Creating ensemble scenario (UML)

Comments to the figure:

- The Scenario Manager prompts the user to specify the input time series when creating a scenario – and in fact offers three possibilities for doing so:
    - Selection of a single time series
    - Selection of an ensemble
    - Selection of an output time series or ensembles from another scenario.
- The Scenario Manager loops over the specified scenarios and performs a validation of the specified ensembles. E.g. using multiple ensemble requires that each ensemble has the same number of entries.

The Ensemble modeller is also involved in running an ensemble based scenario as shown below in Figure 5.33.

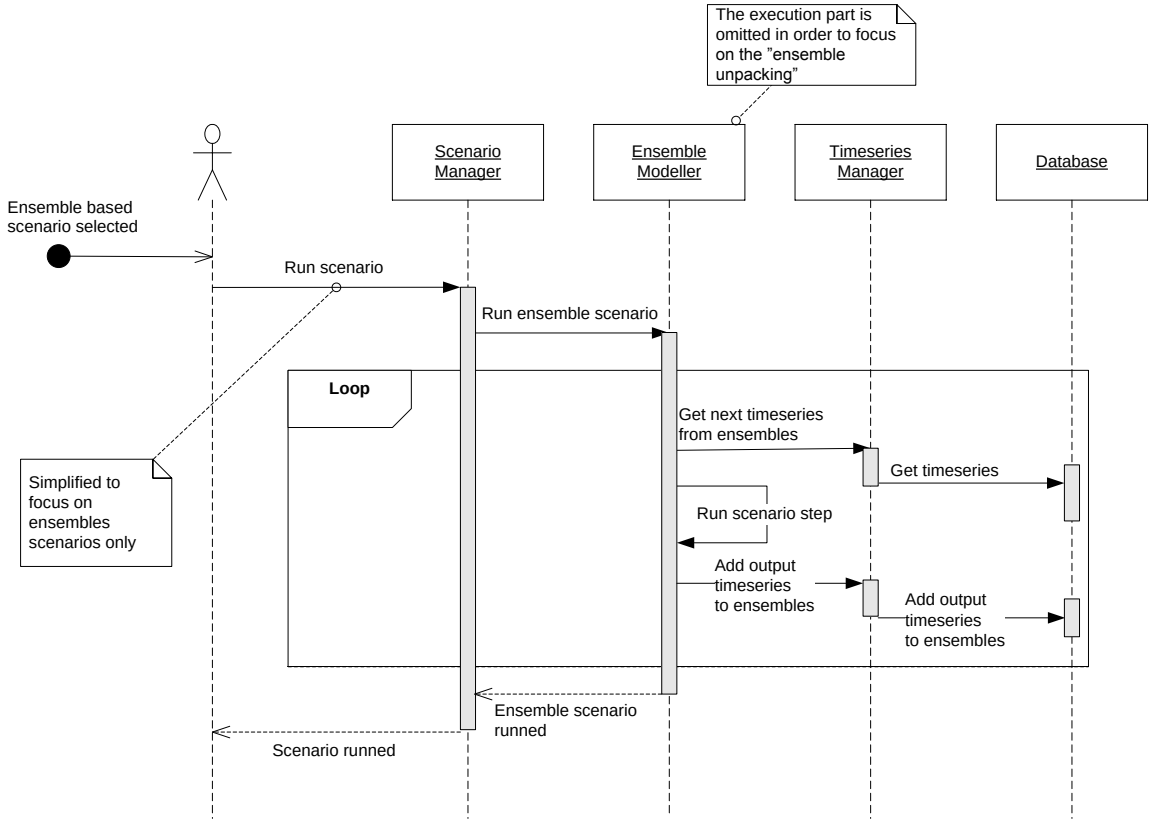

Figure 5.33 Run ensemble scenario (UML)

Note from the figure

- The Scenario Manager defers running of ensemble scenarios to the Ensemble modeller. This performing a loop over all the time series in the ensemble(s). For each time series in the ensemble(s) the Ensemble modeller will run a time series based scenario. The Ensemble modeller manages an optimised scenario simulation but not unpacking model and setup information for each step.

- The output time series will all be stored in the database and grouped into output ensembles

5.2.2 Model Linker Component

A scenario that includes a linked model involves automatic simulation of a number of individual models in a sequence. One model simulation generates output time series that automatically is fed as input time series into the next model simulation in the chain.. An example could be: A rainfall runoff model that generates catchment runoff time series for use in the water allocation model that again generates discharge time series that are fed to a fully hydrodynamic model.

Compared with the "normal" scenario management, a scenario that includes linked models involves one additional step:

1. Looping over the model simulations in the chain

This step is managed by the Model Linker component together with the Scenario Manager.

Defining a scenario that includes linked models is similar to creating a new scenario based on a single model. It involves specifying input time series for each *input point* – or boundary condition - in the combined (linked) model. This amounts to the all input points in the participating single models minus the ones defined as transfer points that gets data from a previous model in the sequence.

Running a scenario based on a linked model will execute the models in sequence with boundaries and transfer as defined. The output will represent all the participating models. This is depicted in

This is depicted in Figure 5.34

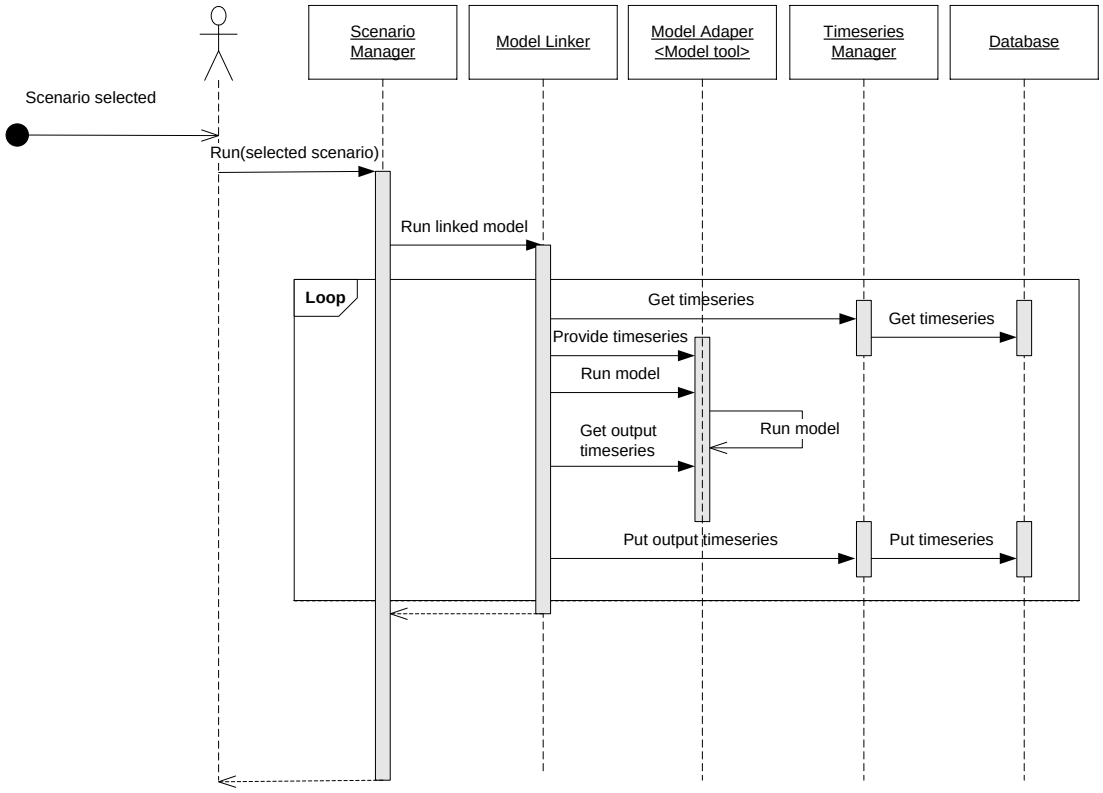

Figure 5.34 Running a scenario based on a linked model (UML)

Comments to the figure

- The 'Run linked model' method will loop over the sequence of models in the linked model
- Each loop will be a 'single' model run. The Model Linker will provide time series from the database and/or from the previous model in the sequence.
- The Model Linker will retrieve output time series both for database storage as well as for input to sub-sequent models in the linked model.

5.2.3 Optimizer Component

Simulation based optimisation involves running a model setup through a number of cycles and for each cycle do the following

a. Calculate variable parameters (to be optimised)

b. Prepare the model setup
   - Prepare input time series
   - Prepare initial conditions
   - Set the optimisation parameters

c. Run the model

d. Aggregate and evaluate outputs

e. Calculate objectives

f. Assess objectives with respect to optimisation targets

g. Stop if OK, otherwise restart at a)

Management of optimisation based scenarios is handled by the following 3 software components:

1. The Timeseries Manager which provides time series information during configuration and execution of scenarios
2. The Optimiser which is a sub component to the Scenario Manager component. This component provides functionality for defining scenarios based on optimisation. The Optimiser modeller also manages the looping and evaluation of objectives when running the optimisation scenarios.
3. The Scenario Manager which provides its standard functionality for managing scenarios.

Figure 5.35 below depicts the flow for creating an optimisation scenario.

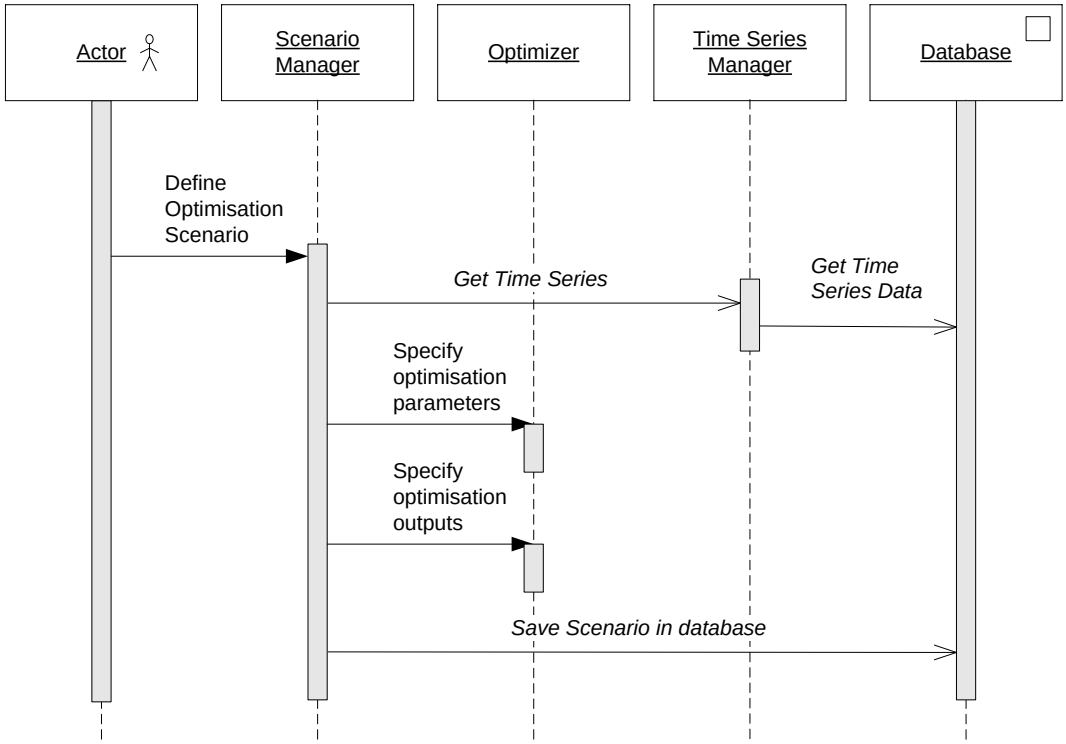

Figure 5.35 Creating an Optimisation scenario (UML)

Note from the figure:

- Get Time Series activity is carried out for as many time series as is required to configure the scenario.
- The Optimiser is used for specifying the additional properties of the optimisation scenario.

When the Scenario Manager executes the optimisation scenario the loop described above must be followed. Execution will still take place using adapters (they are the only ones knowing the proprietary formats and layouts of the model setup). Hence the adapter interface must support setting of parameters as specified by the Scenario Manager.

The Optimiser is also involved in running an optimisation scenario as shown below in Figure 5.36.

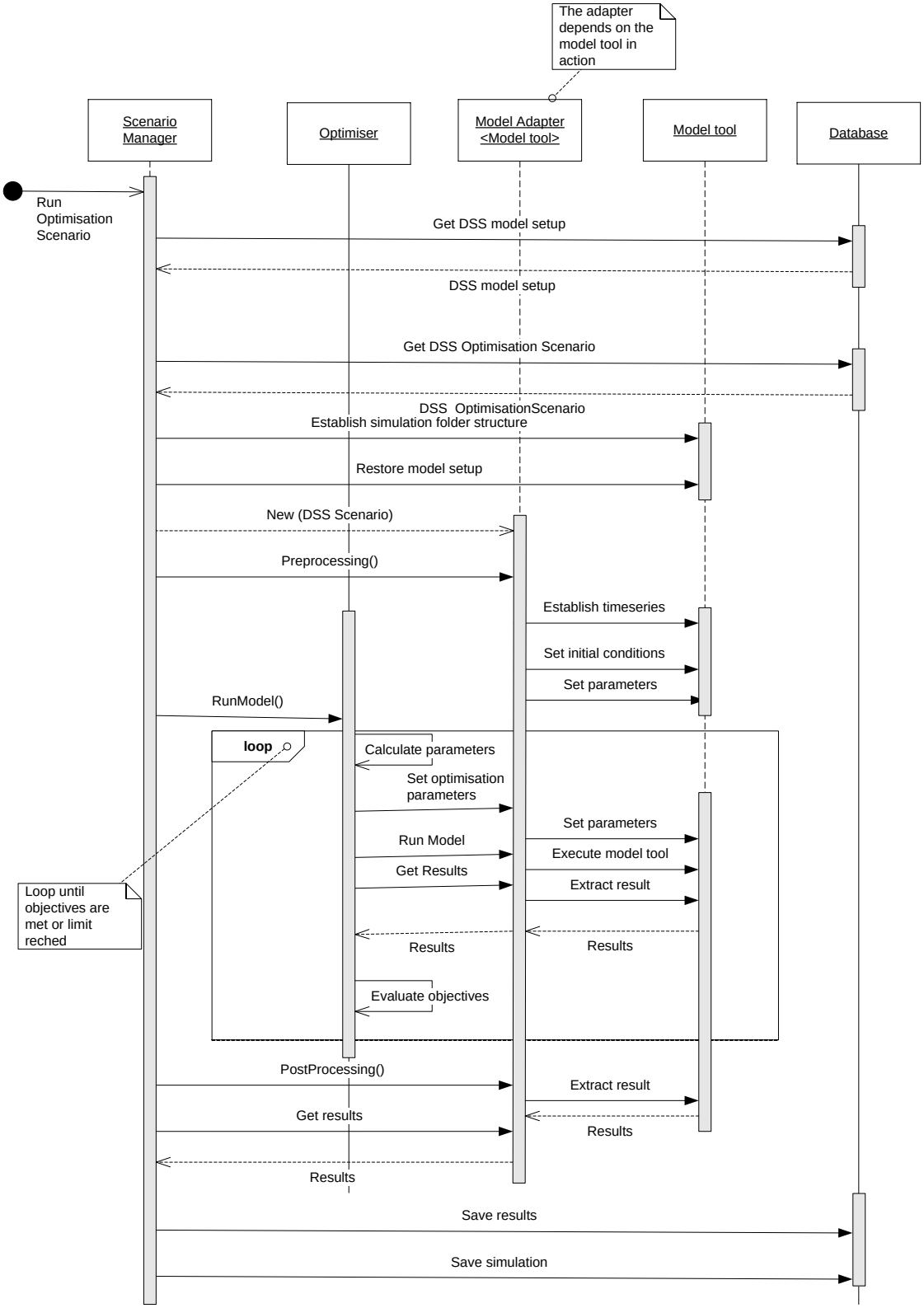

Figure 5.36 Run optimisation scenario (UML)

Note from the figure

- The Optimiser and Scenario Manager collaborate in order to run optimisation scenarios. Optimiser controls the loops while objectives are still to be evaluated. The Scenario Manager controls the overall execution and interaction with the database.

- The execution of the model via the adapter is controlled by the optimiser allowing multiple model executions during execution of only one scenario

- The post processing will return the optimum result

5.2.3.1 Implications

An optimisation tool can do the above in a general manner, but it requires the model(s) involved in the execution to be prepared for use by the tool in order for it to change the values of selected parameters (Ex. AUTOCAL not knowing the proprietary format applies a simple scheme of replacing text strings in setup files). As a result of this the model setup(s) can only be used for the optimization problem at hand.

The aim of the NB DSS is to provide general optimisation functionality where the parameter ranges, objective functions, and criteria are specified in the database irrespective of the model setup(s) used in the optimisation.

The choice of wrapping models with adapters (see Section 5.1.3.1) as the means of integration into the NB DSS for both configuration and execution then implies that the adapter be capable of providing:

- The knowledge on available optimisation parameters in the model setup

- Facilities to set parameters into the model setup during execution

Since the adapter provides all data describing a model setup during registration, the notion of optimisation parameters must be added to the definition of the IConfigAdapter IModelSetup interfaces presented in Section 5.1.3.2. Similarly the IRuntimeAdapter interface shall include methods to set parameters as well as retrieve the optimal parameters at the end of the optimisation exercise.

The data model of the Model Setup and Scenario in the NB DSS shall contain properties describing the possible optimisation parameters (Model Setup) as well as the configuration of the use of them (Scenario). The Scenario Manager shall enable the user to configure the optimisation properties of the Scenario.

5.3 Component Interactions

Components interact with respect to both functionality and data. This section exemplifies how the identified software components interact with respect to fulfilling the functionality of the original user stories.

Selected sections of the user stories have been chosen in order to highlight some of the key functionalities with respect to modelling:

- Model Setup, Scenario, run Scenario and post-process output (UC-01: Identify causes of declining Lake Victoria water level, steps 4-7).

The use cases steps cover the whole cycle from creating a new model setup, making it available in the NB DSS, defining scenarios, running them and evaluating the outputs.

- Scenarios with Linked models (UC-02: Select best option for Jongeli Canal, step IV)

  This use case step describes how to run a linked model from the NB DSS

- MCA (UC-03: Determine Preferred Cascade and first dam of EN, step V)

  Setting up and conducting an MCA analysis is described in this use case step

- Ensemble modelling (UC-03: Determine Preferred Cascade and first dam of EN, steps 2.3, 4.2, 4.5 and 4.6)

  The selected steps from UC-03 describe the use of ensemble modelling. The intermediate steps relate to other activities which are largely independent of the selected steps.

- Optimization (UC-04: Select best investment option for NEL region, steps 8-13)

  This part of the use case shows how to perform optimization.

A sequence diagram for each of the above samples of activities illustrates how the software components interact to provide the required functionality and how data are exchanged.

The diagrams show that the generalized use cases derived during the use case analysis are not sufficient to describe all activities in detail. Hence, some activities in the diagram are potential additional use cases to be further elaborated during the detail analysis and design period.

The sequence diagrams use the following notation:

- The steps from the User Stories are drawn as "Found" messages from outside which the user must perform
- Activities matching existing defined use cases have text in normal font
- New activities (potential use cases) have text in italics
- Actors are – apart from the user – the defined software components and the database.
- Return messages are not included in order to focus on the activities rather than the data flow.

5.3.1 Run model setup

Table 5.8 below is the extract from User Story 1 that has been converted to a sequence diagram in Figure 5.37. The “Generalize use case” column in the figure is the generalised use case derived from the Use case analysis, see SRS /2/.

Table 5.8 Selected steps from UC-01

Actor	Workflow	Generalized use case
Modeller	**4. Model setup – Lake Victoria (monthly time step)** a) Configure a reservoir water balance model setup for lake Victoria b) Calibrate model with observed inflow and outflow time series	4a: Setup MIKE BASIN
		4b: Calibrate MIKE BASIN
Modeller	**5. Define scenarios:** a) Scenario 1: Lake outflow governed by agreed curve b) Scenario 2: Lake outflow governed by natural outflow	5a: Register model Create scenario
		5b: Create scenario
Modeller	**6. Simulate scenarios 1 and 2**	6: Run scenario
Modeller	**7. Post-process scenario run results** a) Determine the trends in lake water level series of observed lake water level and the two scenarios b) Determine estimated lake level decline for scenarios 1 and 2 and compare with observed decline	7a: Use time series tool
		7b: Visualise time series

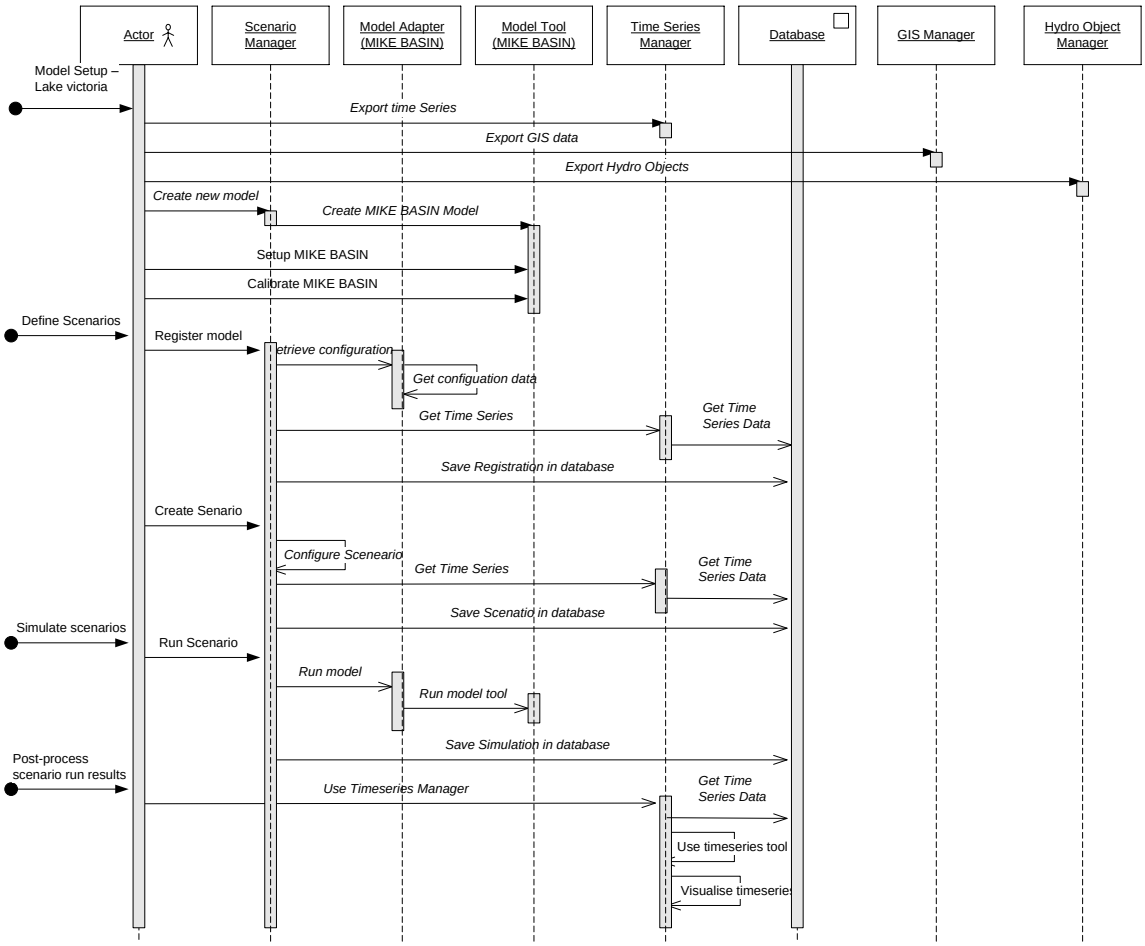

Figure 5.37 Setting up and running the Lake Victoria model (UML)

Note from the figure that

- Export of data to be used for modelling is carried out prior to creating the model in the model tool
- Some actions may be required by the Scenario Manager to establish the environment for the user in order to setup and calibrate a new mode
- The "Register model" and "Create Scenarios" activities in the Scenario Manger involve activities by several other components
- The Scenario Manager runs a model tool via the "Model Adapter"
- Post-processing is to a large degree a standard activity performed by the user via the Timeseries Manager.

5.3.2 *Linked models*

The selected steps from UC-02 focusing on establishing and linking models are depicted in the diagram in Figure 5.38.

Table 5.9 Selected steps from UC-02 regarding linked models

Actor	Activity	Generalized use case
Modeler	IV Scenario Configuration and Run	
	1. Copy and modify water spine model of baseline scenario and define different sets of parameters for inlet node of Jonglei Canal, such as: A Water abstraction rules for flows (e.g. as diversion flow as a function of another state variable of another hydro-object) B Water abstraction flows/levels (e.g. as time series)	IV.1.A Clone and modify MIKE BASIN model and scenario IV.1.B Clone and modify MIKE BASIN model and scenario
	2. Set-up and configure hydraulic model of Jonglei-Canal; use DEM and other data sets to determine cross-sections and longitudinal profile	IV.2 Use GIS tool Setup MIKE 11 model Register MIKE 11 model
	3. For each set of parameters define a scenario and link the set of models (rainfall-runoff and water spine) with the hydraulic model of Jonglei Canal (including definition of modelling sequence) – new linkage between water spine and hydraulic model through inlet and outlet node of Jonglei Canal	IV.3 Create linked model Create scenarios
System	4. Simulate each Jonglei-Canal scenario; run set of models (according to defined modelling sequences)	4: Run scenario

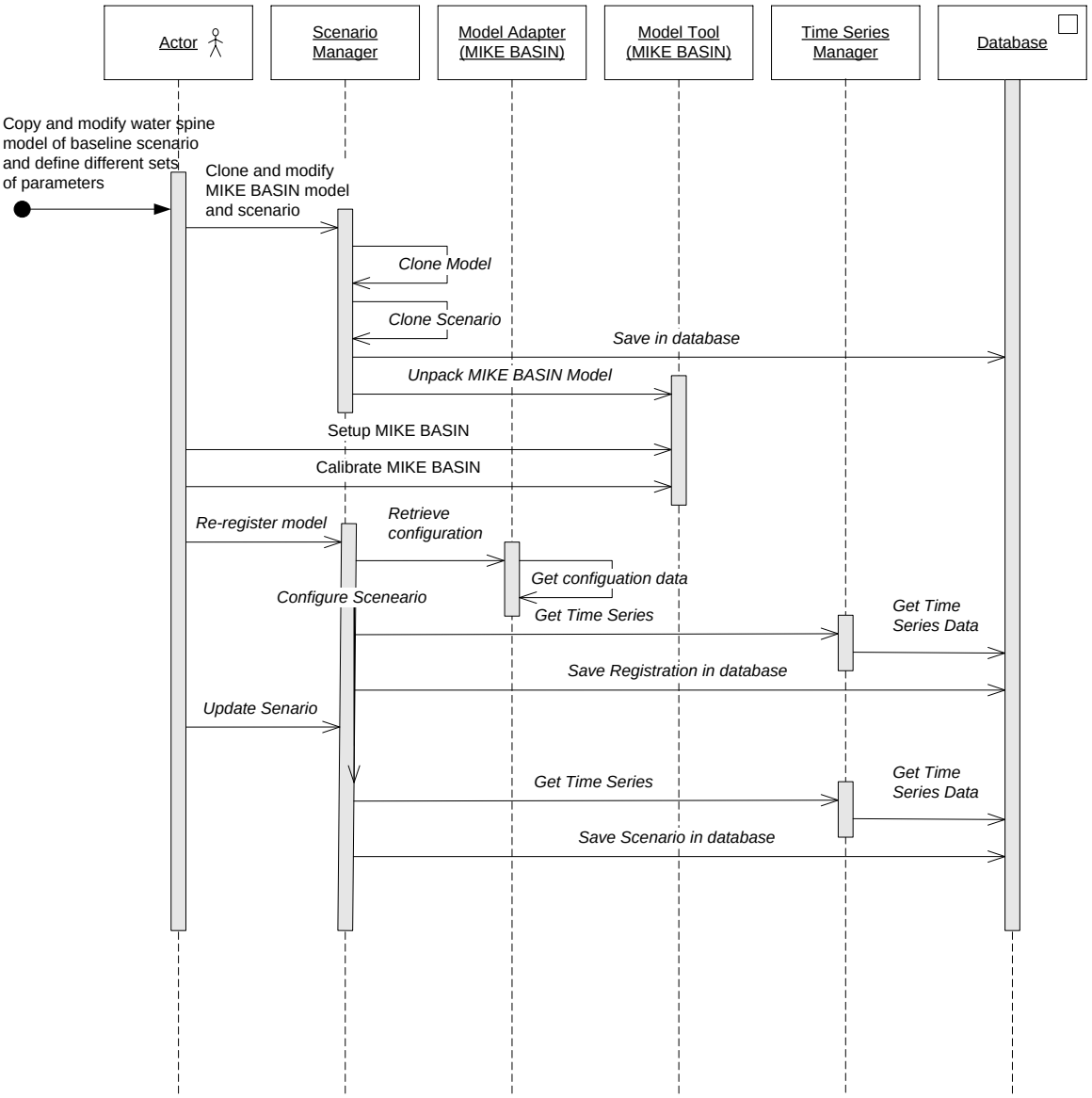

Figure 5.38 Creating alternative models and linking them, part 1/2 (UML)

Note from the figure that

- Cloning of a model setup and scenarios is a database activity.
- Editing of a setup is both setting up the model in the model tool and calibrating it
- After editing, the re-registration of the model setup updates the cloned configuration in the database

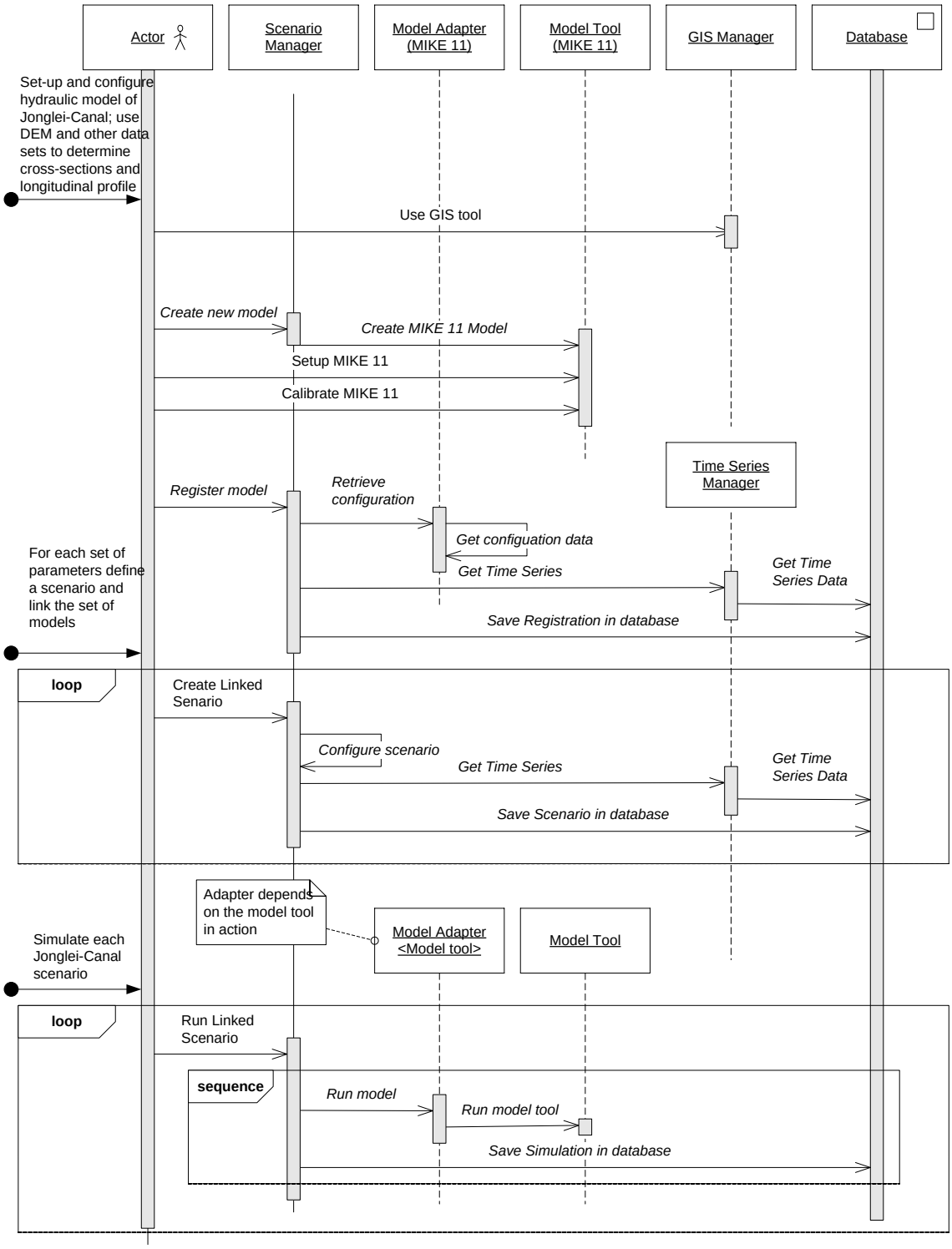

Figure 5.39 Creating alternative models and linking them, part 2/2 (UML)

Note from the figure that

- Setting up the HD model is similar to the water spine model: use of the model tool for setting up and configuration followed by registration into the NB DSS

database. Note: Cloning it is a similar to MIKE Basin as described in Figure 5.38

- The linking of the two models defines the transfer of time series (output to input) and the sequence in which to execute the model.

- A scenario definition for the linked model is similar to that of any other model, only using the combined conditions for the participating models.

- From a user perspective running the scenario with linked models is similar to running any other scenario. Internally Scenario Manager will perform the transfer (sequentially) of time series from output to input as defined in the linked model.

5.3.3 MCA

Table 5.10 shows the selected steps from UC-02 regarding use of MCA for analysis. Figure 5.40 depicts the activities in a diagram.

Table 5.10 Selected steps from UC-02 regarding MCA

Actor	Activity	Generalized use case
Modeller & Decision Maker	V Indicator Definition (and Calculation)	
	1. Review all information that was made available through the data pre-processing phase using the DSS GUI	V.1 Visualize GIS data Visualize time series
	2. Define relevant indicators for scenario comparison and MCA, such as A Total area reclaimed for agriculture B Extent of change in swamp area (permanent and seasonal) and its impacts on the livelihood of the community (such as decrease in livestock, grazing area, fishery production) C Impacts on the flora and fauna that exist in the swamp D Total benefit from conserved water (in this case through assessment in other use cases of the DSS)	V.2.A Define indicator V.2.B Define indicator V.2.C Define indicator V.2.D Define indicator
	3. Define relationships between model data/properties and the above indicators A Import and/or edit tables that represent relationships (in this case these relationships are derived outside the scope of the DSS) B Write scripts to formulate functional relationships and/or aggregations as appropriate	V.3.A Use MCA indicator tool V.3.B Use MCA indicator tool
System	4. Calculate indicators for all scenarios including baseline scenario (convert	V.4 Run MCA

Actor	Activity	Generalized use case
	model outputs to indicators as defined in indicator definition phase) and convert the criteria 5. Populate MCA with criteria as specified and defined in the indicator definition phase	V.5 Run MCA

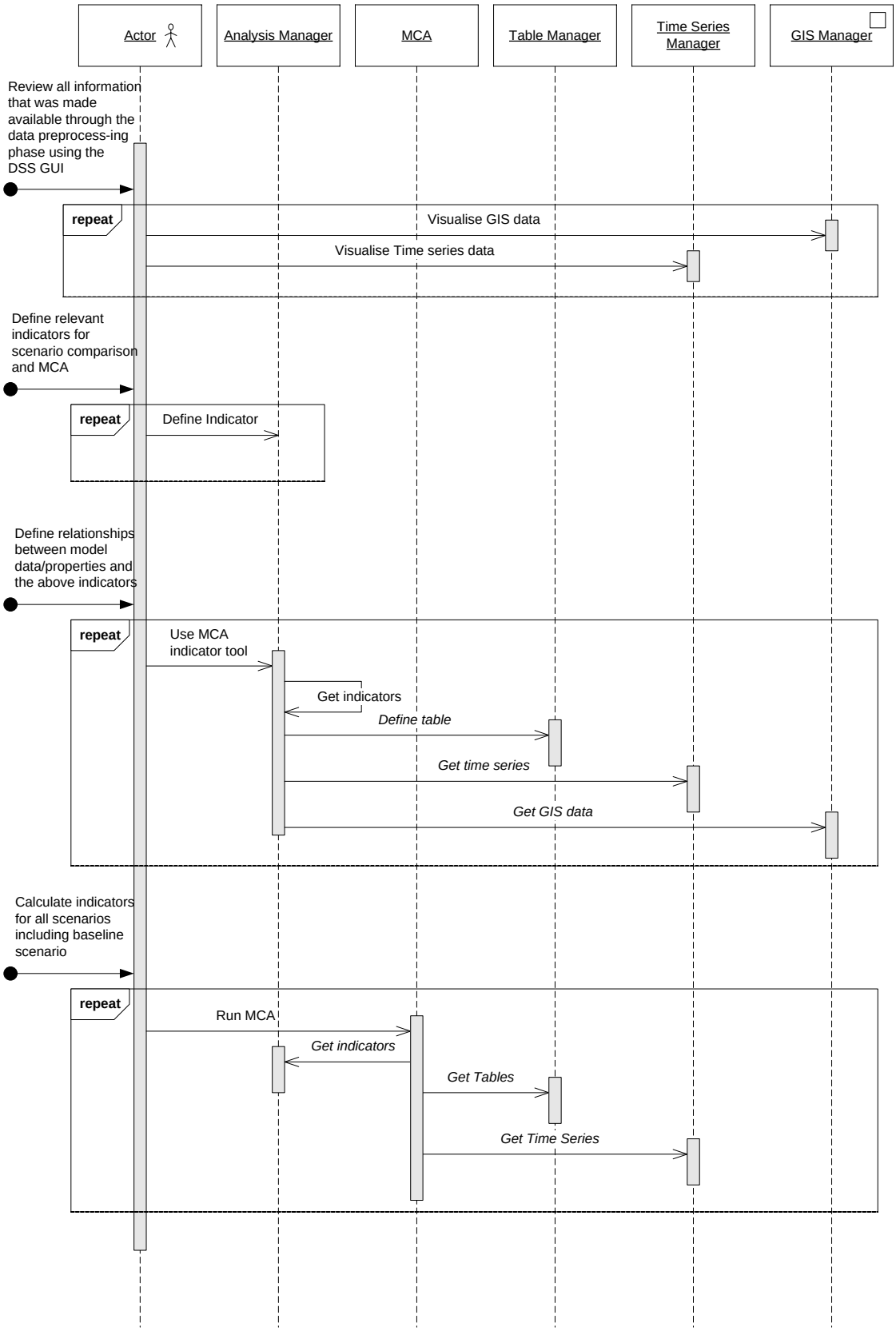

Figure 5.40 MCA analysis (UML)

5.3.4 Ensemble modelling

Table 5.11 Selected steps from UC-03 regarding ensemble modelling

Actor	Workflow	Generalized use case
Group of *Hydrologists, GIS experts* collaborating (including across country) on the study (including synchronization)	**2.3 Prepare data for study** a) Import all data not available within database (from spreadsheet, ascii files, manual entry, etc) b) Quality assure hydro-meteorological data	2.3.a Import time series Import GIS data Import tables
	c) Generate/prepare in-stream flow requirements at key control points d) Estimate/generate flow series for unguaged catchments	2.3.b Use time series tool Edit time series Create time series
		2.3.c Create time series
	e) Generate inflow Time Series (TS) at dam sites f) Prepare rainfall TS at dam sites and irrigation development sites	2.3.d Use rainfall runoff model Use time series tool
	g) Prepare TS of temperature, evaporation, etc at irrigation development sites	2.3.e Use rainfall runoff model Use time series tool
	h) Import/Determine flood damage zones characteristics (damage curves/tables) i) Import population distribution and density data	2.3.f Create time series Use time series tool
	j) Compile operation rules of existing reservoirs	2.3.g Create time series Use time series tool
	k) Generate/import sediment yields at key dam sites (could involve catchment erosion estimation) l) Prepare characteristics of dams (area-capacity curve, spillway rating curves, etc) m) Estimate water demands (such as irrigation sites, major urban centres) n) Generate ensemble of flow time series (daily time step)	2.3.h Import GIS data Import table
		2.3.i Import GIS data
		2.3.j Create hydro object
		2.3.k Use soil erosion tool Import time series Link time series to feature
		2.3.l Use hydroobject tool
		2.3.m Use demand calculator tool
		2.3.n Use ensemble generator tool

Actor	Workflow	Generalized use case
Modeller	**4.2 Link generated ensemble with model setup of each scenario**	4.2 Setup ensemble scenario
Modeller	**4.5 Simulate the sequencing scenarios** using the ensemble of input series	4.5 Run ensemble scenario
Modeller	**4.6 Analyse ensemble simulation run results and synthesize statistical outputs (mainly indicator values given under 4.3)**	4.6 Visualise time series Use time series tools Visualise tables

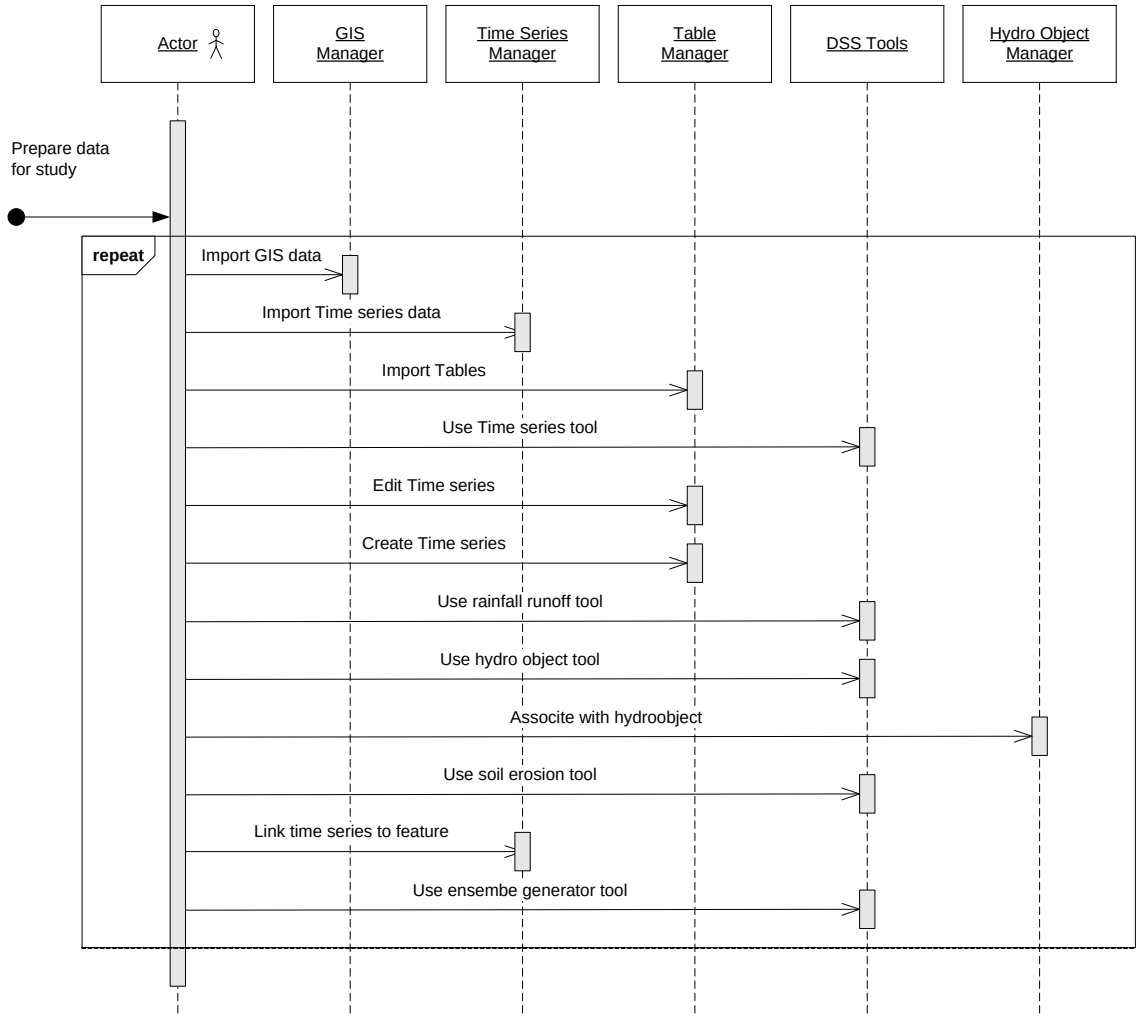

Figure 5.41 Ensemble modelling, part 1/2 (UML)

Note from the figure that

- Numerous tools and components are in play for preparing data
- The order of the use of tools lies with the user (indicated by “repeat”)

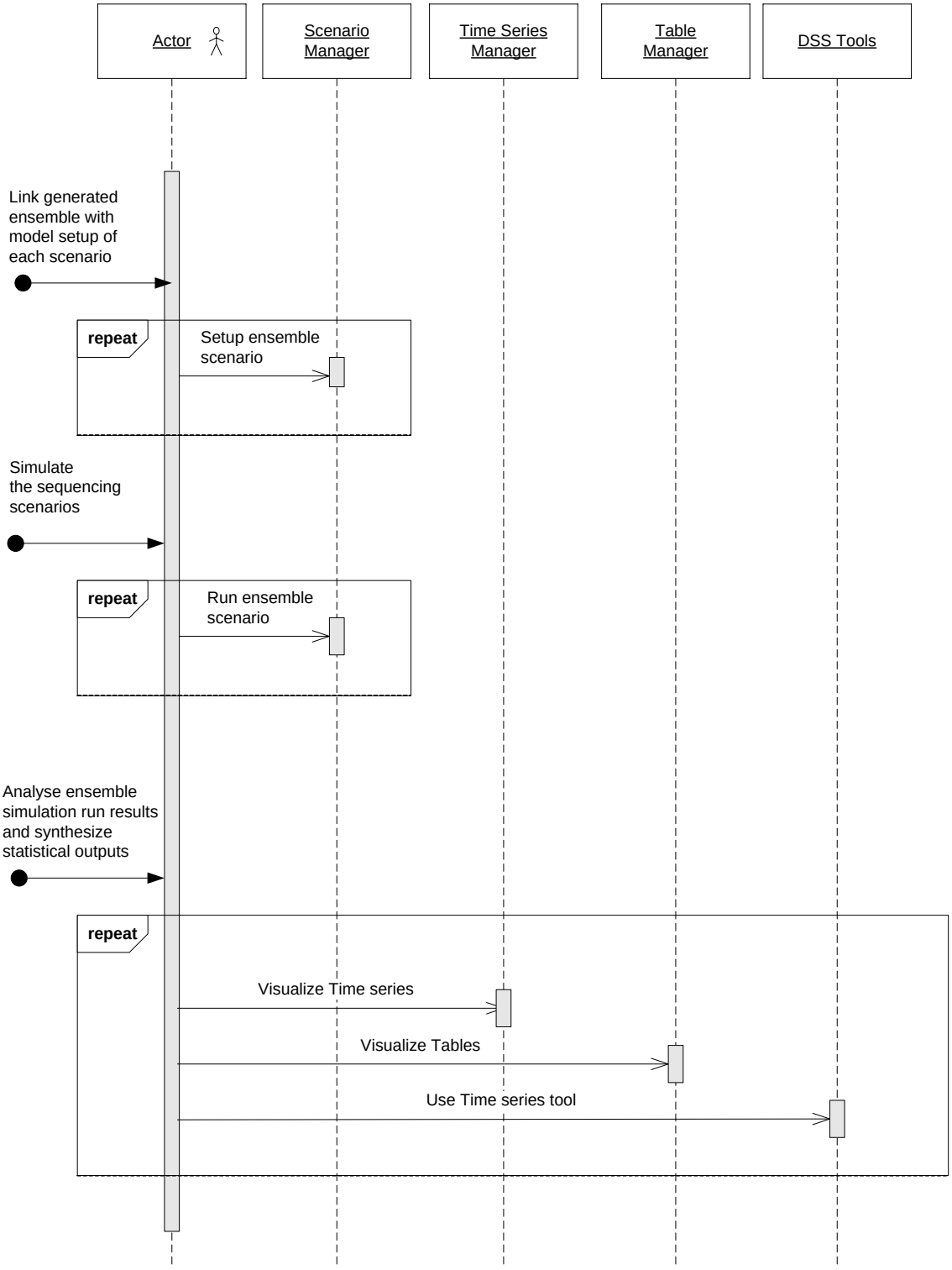

Figure 5.42 Ensemble modelling, part 2/2 (UML)

Note from the figure that

- Setting up and running ensemble scenarios is similar to running any other scenario, with the differences being the ensemble loop during simulation and that collection of results and statistics as well as post processing are additional sets of specifications to include with the scenario.

- The user can assess results by using the available tools and viewers following the simulations.

5.3.5 Optimization

Table 5.12 Selected steps from UC-04 regarding optimization

Actor	Work Flow	Generalized use case
Water resources economist /Modeler	8. Determine parameters for simulation based optimization including Objective functions and Constraints A. Define objectives I. Define objective function to Maximize revenue from Power production or Irrigation crop production. II. Define objective function to minimize cost. (the cost can be investment, environmental mitigation) B. Define constraints I. Minimum Lake Water level (Time series) II. Minimum downstream water release (Ecological Flow) (Time series) III. Acceptable water quality (In terms of water quality parameters like Nutrient level, BOD, sediment load etc.) IV. Minimum Economic and financial parameters (such as: IRR, B/C ratio, NPV) V. Maximum cost C. Determine/define method of optimizer.	Define optimization scenario Create optimization scenario
System	9. Optimization Maximize benefits/Minimize cost (impacts) with respect to pre defined objective functions and constraints. A. Run simulation for the selected scenario a number of times by varying the sizes of reservoirs and/or scales of irrigation developments and determine i TS of Water quality parameters including sediment load ii TS of Lake Victoria water level iii Flood Prone areas: extent of	Run optimize scenarioUse Indicator tool Setup CBA RunCBA

Actor	Work Flow	Generalized use case
	economic damage iv Power generation v Irrigated areas & consumptive use of water vi Affected tourist attraction sites. B. Determine costs and benefits for each of the simulation runs: (using specs of step 5) i Determine costs: Investment and running costs, economic losses due to flooding etc. ii Determine benefits: Revenue from power generation, irrigation development fisheries etc. C. Calculate economic and financial parameters (B/C, EIRR, FIRR, NPV) for every simulation	
Water resources economist /Modeler	10. Alternative selection. A. Select optimum alternative of the scenario under consideration. B. Sensitivity analysis with respect to discount rate and other parameters like delay in construction and change in cost of construction etc.	Take decision Setup CBA Run CBA
System	11. Convert Model outputs into user defined indicators for the optimum alternatives of each scenario. (using specs of step 6)	Setup MCA
Decision Maker	12. Run the MCA tools and Select the best option	Run MCA Take decision
Decision Maker and Modeler	13. Generate Report	Visualize time series Visualise tables Publish to report

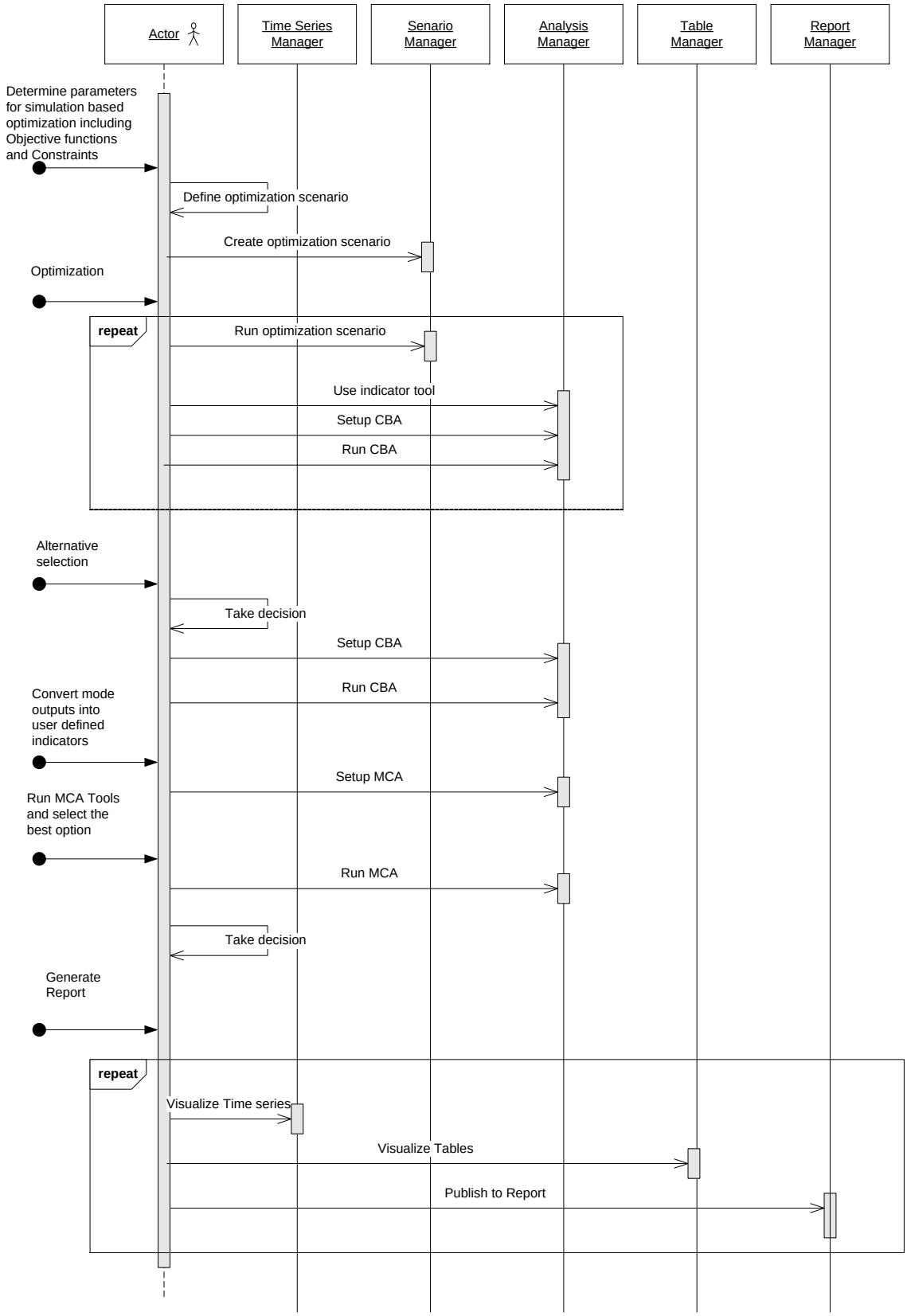

Figure 5.43 Optimization (UML)

Note from the figure that

- Defining a scenario lies with the user.

- Creating a scenario will (likely) involve also setting up and calibrating a model to run, but may use models already existing in the NB DSS

- Creation of an optimization scenario is like defining other scenarios, only rules for optimisation are additional specifications.

- Running an optimization scenario to the user is similar to running any other scenario. The logic being executed will hold the optimizing evaluation and parameter adjustment (to be conducted via the adapter)

- The actor “System” is used in a number of steps, which may imply some activities are “within the system” or “batch driven”. This is not clarified yet.

- Decision taking is an activity with the user

- Setting up and running CBA and MCA analyses as shown will involve many components for indicator definition as well as calculation (not shown here)

- Report generation will involve different tools in the NB DSS

6 VIEWPOINT: SYSTEM USE

This view point addresses the system seen from a user perspective. The aspects addressed include user profiles and the DSS Front-end user interface (UI).

6.1 User profiles and permissions

The NB DSS will be accessed and used by many – different – users. They will access the system with different purposes and performing different actions. A system for determining their level of access and permissions on the system for accessing functionality and data will be devised.

The system will have the following characteristics:

- Be simple to administer
- Ensure people get access to what they are supposed to – no more, no less
- Group people in logical/organisational groups with common access rights and usage of the system
- Allow sharing of data across institutions and countries, hence rules must apply how to transfer permissions
- Allow users to cooperate and work together on studies sharing data and results. Handling of data in a study is described in more details in Section 5.1.4.3.

6.1.1 User groups

In order to avoid detailed maintenance of user permissions the access system will use groups of users – which share common rules of permissions to functionality and data. Each user will be member of one or more groups.

A minimum set of user groups will exist by default in all installations, but others may be created locally according to needs. The default groups are:

- Everybody

  All users are member of this group by default. This group cannot be removed. The users of this group can do simple things like view data and make reports, but not add data or alter the database content.

- Data Owner

  Users in this group can administer data at the global level.

- Administrators

  The administrator can do everything, but also perform certain system tasks such as adding new user, create new groups and create new studies.

When a new user is added to the system he is by default assigned to the "Everybody" group, getting the associated permissions. When a new group is defined it can be defined as a copy of an existing group as a starting point and then from there on modified with respect to assigned resources and actions.

6.1.2 Study groups

A user that has been assigned the role as Study Owner can administer the permissions to functionality and data within the study in 3 groups:

- Study Reviewer

  A study reviewer has permission to view all data in the study.

- Study Member

  A study member can manipulate all data within the study

- Study Owner

  The study owner can delegate other users to become administrator of the study as well as assign users to the Study Users and Study Viewers groups. Only Study Owners can remove a study.

6.1.3 Functionality permissions

A user is member of one or more groups. Members of a group are allowed access to perform an action on a resource. If permission is not given to any groups of which a user is member a user does not have access to the action and resource. If different levels of access are given a user through group memberships the highest level prevails.

Resources and associated actions are defined by the DSS Front-end modules providing the functionality. Each module will update the list of resources and actions as part of the installation and define permissions to the 4 default groups following the general guidelines above. The set of permissions can be modified by the administrator.

Examples of some resources and some actions are:

- Time series
  - Create time series
  - Import time series
  - Import time series data
  - View time series
- Study
  - Create study
  - Add Study Administrator

- Remove Study Administrator
- View Study

- Users
  - Create user
  - Edit user
  - Delete user
  - Assign user to a group
  - Remove user from a group

- Model Setups
  - Create model setup
  - Import model setup
  - Edit model setup
  - Delete model setup
  - Promote model setup to common data

6.1.4 Data permissions

All data entities (such as time series, models, scenarios etc.) will be created in the system with as set of defined permissions. These permissions are set according to a set of simple policies:

- When created in the Global data area the permissions are:
  - Read for Everybody
  - Read, Update, Delete for Administrators

  Only Administrators have permissions to create data in the Global data area.

- When created inside the framework of a study - manually or through a process - permissions are created for the study user groups.

6.2 DSS Front-end UI

This Chapter describes the overall design of the DSS Front-end user interface (UI). It is the Shell component that is responsible for displaying the UI. For a detailed technical description of the Shell component, see Section 5.1.2.2

The DSS Front-end is a Windows application. It has an IDE-style user interface where all windows reside under a single parent window. The IDE-style UI comprises for ex-

ample dockable[1] and collapsible child windows, tabbed windows and splitters for resizing of child windows. Generally, the UI is inspired by other well-known IDE-style UI's like for example Microsoft Outlook and Microsoft Visual Studio.

Conceptually, the UI appears as illustrated in Figure 6.1, with a main window (the Shell) hosting a number of child windows of different types.

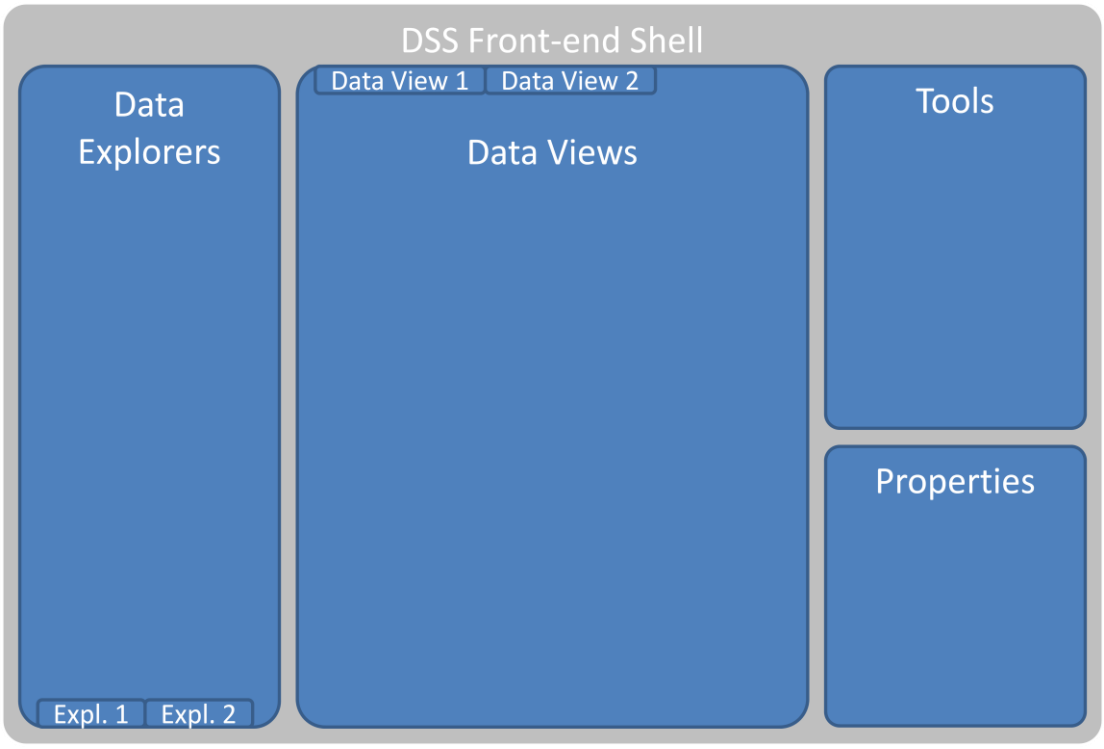

Figure 6.1 The DSS Front-end Shell UI

The different types of windows in the Shell are:

- Data Explorers (one or more)
- Data Views (zero or more)
- Tools (one)
- Properties (one)

The Data Explorers window and the Data Views window are tabbed windows acting as containers for Data Explorer windows and Data View windows. For a detailed technical description of the modules, see Section 5.1.2.3.

In the following sections, each of these window types will be described.

[1] The window can be "glued" to another window

6.2.1 Data Explorers

Data Explorers are used to retrieve a subset of the data that is contained in the database. Data may be retrieved according to a user defined search/filter criterion – for example filtering by attributes - or by choosing a criterion from a list of previously saved criteria.

Data Explorers can be of more general character, such as e.g. the Timeseries Explorer and GIS Explorer, or tailored to more specific tasks, such as e.g. the Scenario Explorer. Although the different Data Explorers may differ in appearance, they will normally consist of a set of controls that are used to build the query, and a section that lists the retrieved data – typically in a tree view.

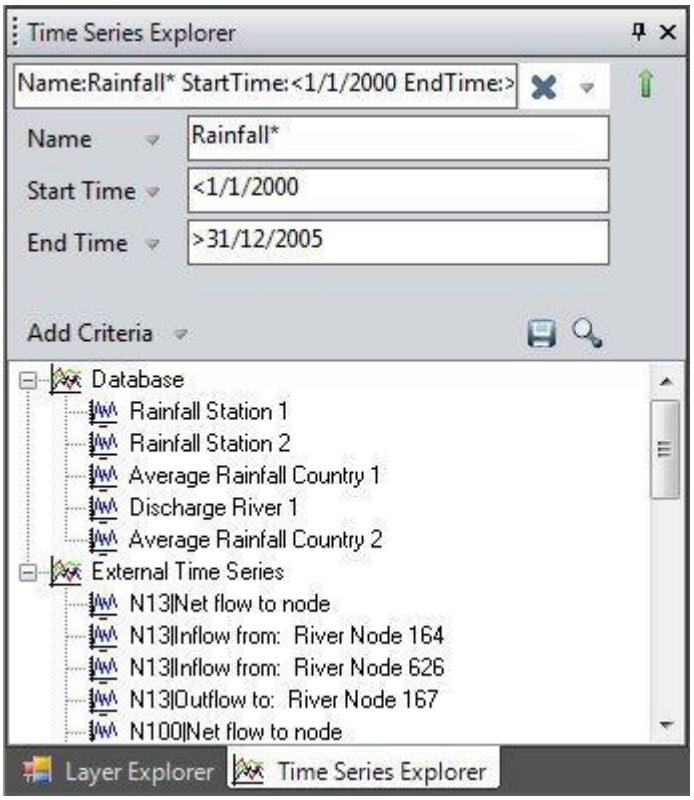

Figure 6.2 Example of Data Explorer window

6.2.2 Data Views

Data Views are used to present the data. Normally they include UI controls such as charts, tables or maps. In Figure 6.3 below, an example of a data view window implemented by the Timeseries Manager is shown. This data view utilises a chart control.

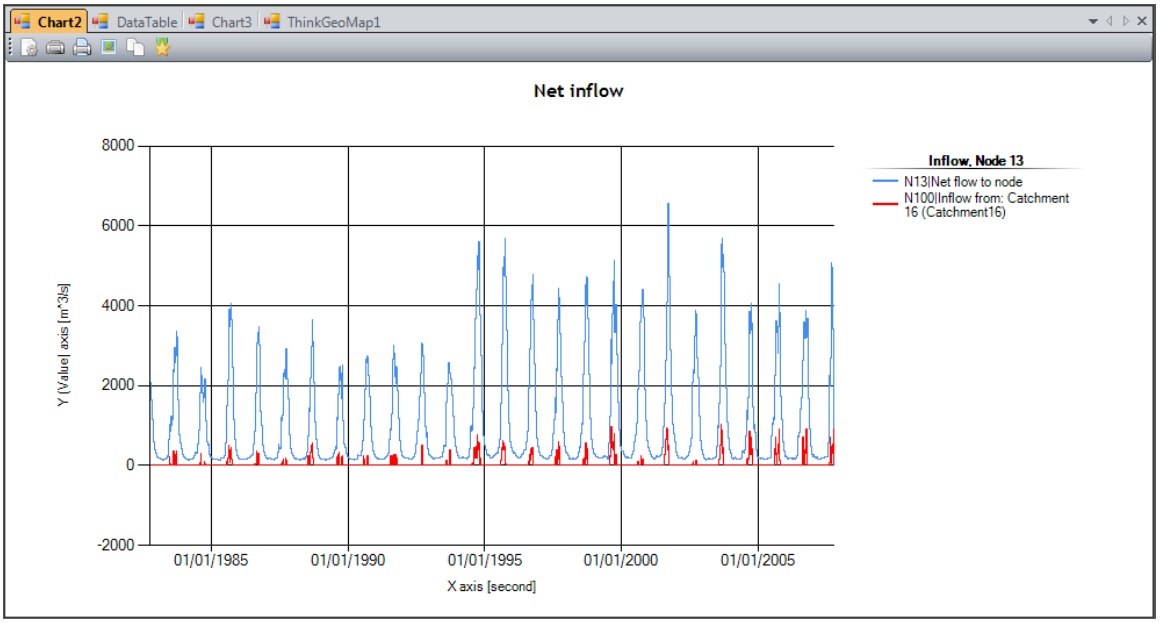

Figure 6.3 Example of a Data View window

6.2.3 Properties

Whenever a UI object[1] is selected in the DSS Front-end - normally by clicking on it - the Properties window displays a tree of available property categories.

When a property category is selected, the corresponding properties are shown, and can be edited. For a chart data series, the property categories include for example M*arker Lines*, *Appearance Settings*, *Stripline Settings* and *Chart Axes Settings*. If for example the Appearance Settings category is selected, the Chart Style, Color Settings, Line Settings and Marker Settings can be set in the Properties window as shown in Figure 6.4.

[1] A UI object can be e.g. one or more time series selected in the Timeseries Explorer, a y-axis in a time series plot, a table column, or one or more map features.

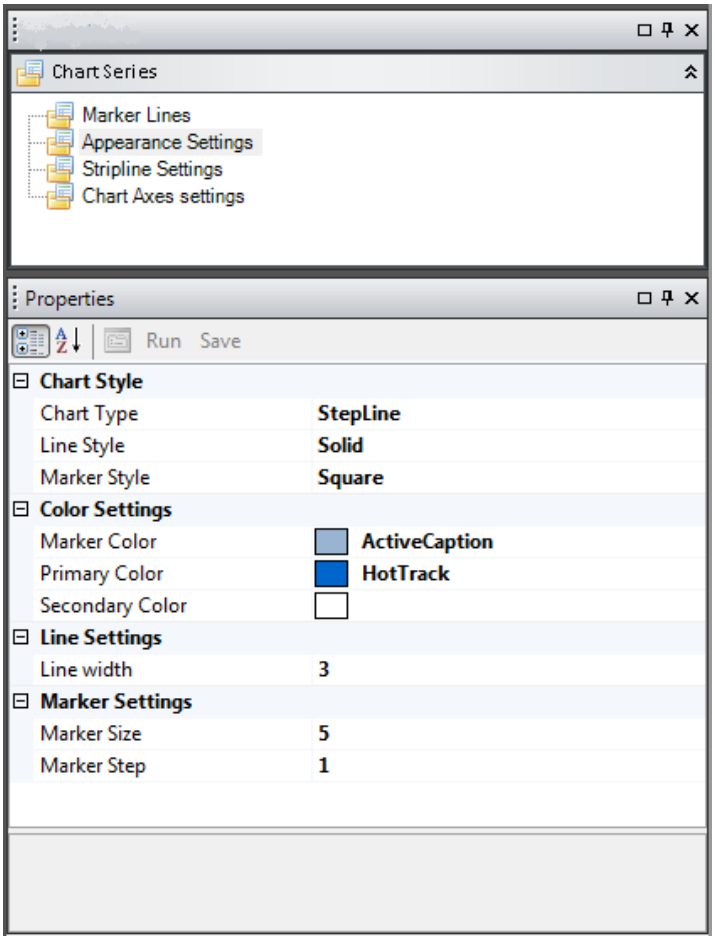

Figure 6.4 The Properties window

6.2.4 Tools

Similar to the Properties window, whenever one or more UI objects are selected in the DSS Front-end, the Tools window displays a tree of tools that are registered to work on the selected object(s).

Hence, if e.g. a time series has been selected, the Tools window will include tools such as *Extract Period*, *Resample*, *Fit to Normal Distribution* etc. as shown in Figure 6.5.

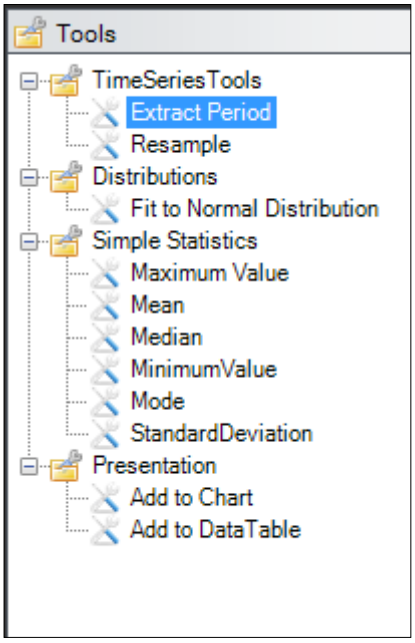

Figure 6.5 The Tools window

When a tool is selected in the Tools tree, it can be configured in the Properties window and executed. Moreover, a tool configuration can be saved, and saved configurations will appear as child nodes to the original tools in the Tools tree.

For a detailed technical description of the Tool components, see Section 5.1.2.4.

6.2.5 Notifications

This section describes how to use messages and exceptions within the NB DSS components, as well as the underlying architecture.

Messages in this context are dialogs displaying Information, Warnings, Confirmations, User- and Program Errors to the user.

6.2.5.1 Message types – description

The different types of messages that are supported by the NB DSS Message Framework are described in the following sections, along with guidelines on when to use the different types.

6.2.5.2 Informational messages

This message type shall be used when the system needs to present useful and relevant, but not critical, information to the user. The information messages do not require immediate user actions and users can freely ignore them. The Information dialog is shown as a transitory dialog that, if the user does not click the dialog, disappears after 5 seconds.

Example: *When a user adds a time series to an existing chart that contains more than one chart area, it may not be apparent which chart area the series was actually added to. The task of the message framework would in this case be to provide the information to the message control that would allow the system to inform the user about which chart area the series was added to.*

6.2.5.3 Warnings

Warnings are used to inform the user of a condition that might cause a problem in the future. A warning could also be appropriate if the user is about to perform an action that has significant consequences or cannot easily be undone.

Example: *The user is about to close the application without saving changes to the project file.*

The Warning Control is a modal message box, having Yes/No buttons.

6.2.5.4 Confirmations

Confirmations are used to confirm actions that have significant or unintended consequences.

Example: *The user may be asked to confirm the deletion of data.*

The Confirmation Control is a modal message box, having Yes/No buttons. The difference between a confirmation and a warning is the icon displayed on the message box.

The standard confirmation dialog is provided with Yes/No buttons. If required, a customizable confirmation dialog can be invoked providing more action options to the user.

6.2.5.5 User input errors

A User Error Control is used to inform the user of a problem caused by invalid user input. It should present the user with guidelines on how to correct the error, so the task can continue.

6.2.5.6 System errors

A Program Error alerts users of a problem that has already occurred. Program errors are a result of a bug or insufficient error handling in the code, and needs to be reported to the Application responsible within NBI, so the problem can be fixed. System errors are always logged to the Windows Event log with a stack trace showing where in the application, the error occurred.

6.3 Using the application

6.3.1 Start-up and login

During the login process the user connects to a DSS database (details provided by the system administrator or similar) and specifies a set of valid credentials (username and password) for the DSS database. The user can also select the Study he wants to log on to (see Section 6.1.2).

The user needs to have a valid account in order to logon. If he logons on to a specific Study he also needs to be associated with that study.

6.3.2 Navigation

Within a Study the user is able to access the global as well as local (to the Study) data for manipulation.

Typical activities are:

- The user can browse the catalogue of data, e.g. time series and map layers.

- The user can create maps by adding map layers to it
- The user can plot time series by selecting the time series in the Data Explorer and chose the Plot command. The plot layout is configurable.
- The user can manipulate time-series via the analysis tools or define a sequence of analysis tools to be executed (with associated tool parameters and settings). The configuration (parameters, sequences etc) of these data manipulations may be stored for later re-use.
- The user can register a model (already prepared in a modelling tool) with the system, linking the model setup time series to time series in the system and creating system representations of key features of the model configuration in the database as part of the model description.

The available set of functionalities is determined by:

- The installed modules (the system is expandable)

  The different modules provide different functionality (Time series Management, Scenario Management, etc). The system is expendable if new functionality is required this can be added via new modules.

- The user level (permissions)

  See further details on user profiles and permission in section 6.1.

- The data

  Some parts of the functionality relates to certain data and data types, e.g. plots of time series. The functionality becomes available to the user when working with the relevant type of data for that functionality, for instance as new options in a pop-up menu or appearance of certain tools in a tool box.

6.4 Scripting

The UI is not the only way to interact with the DSS Front-end. It is also possible to interact with the system programmatically through scripting. As explained in Chapter /5/, all of the modules follow a layered architecture having a data layer and a business layer..

Most functionality provided by the business service layers are based on documented public interfaces. This makes it possible to programmatically access the business services and thereby automate interactions with the system. Scripting is just another way than the UI to interact with the system; it does not provide additional functionality in itself

Model tools providing .NET based or COM based public interfaces can be scripted from the script component

Figure 6.6 below – copied from Chapter 5 – is key to understand how scripting is implemented with the DSS Front-end.

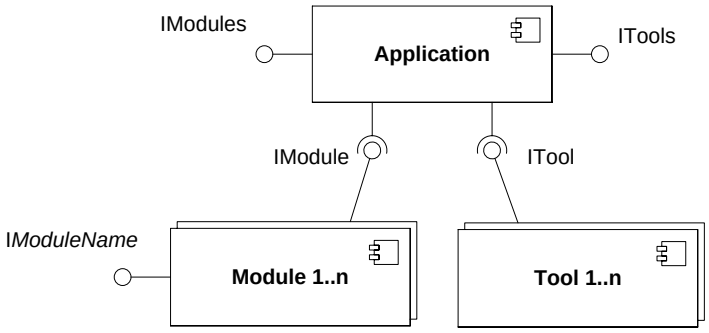

Figure 6.6 Application, Modules and Tools components (UML)

The Application component is the component that glues all the other components (Modules and Tools) together and forms a single application. In itself the Application component does not provide any user interface. This is provided by the Shell component. This is discussed in more details in Section 5.1.2.1.

When a user wants to script some functionality in the DSS Front-end, the Application component provides the root of the object model (the entry point to the scripting functionality). Access to Modules and Tools are provided through the Application object. The text box below shows in pseudo-code how to use the Application object, the Timeseries Manager and a Tool to calculate the mean value of a time series.

```
// Instantiate the Application object
app = CreateObject(”DSS.Application”)

// Get a timeseries from the database
tm = app.GetModule(”TimeseriesManager”)
ts = tm.GetTimeseriesById(27)

// Load and execute the Mean tool
tool = app.GetTool(“Mean”)
tool.SetData(ts)
mean = tool.Execute()
```

Figure 6.7 Sample script

The business service layer is exposed as Microsoft COM components which make it possible to use all script languages that can work with COM interfaces. Examples are VBScript, Python, and Perl.

The DSS Front-end will support scripting by having a code editor, the Script Manager that can be used to write and execute scripts and store them within the DSS Database.

6.5 Scheduling and batch

The NB DSS will offer possibilities for scheduling and batch processing. Likely scenarios for running tasks in batch are:

- A long running task which does not require user interaction
- Repeatable tasks
- A sequence of operations to be started automatically

While scenarios for using a scheduling mechanism include

- Automatic transfer of data to and from the NB DSS
- A wish to execute a batch task at a given time

Furthermore the batch and scheduling functionality is linked to the need for load-spreading, i.e. dividing execution of long running and CPU demanding tasks across different machines.

Scheduling will require a service to provide the trigger mechanism to start a process - either by event or by schedule. Different possibilities exist for this:

- Using the built-in Scheduled Tasks service in Windows. Starting from Windows Vista the scheduler allows definition of multiple steps in one task. It can execute any executable using windows credentials.

  Benefits from using the Windows service include:

  - It does not require special installation
  - It will be available for all NB DS clients
  - It can execute also non DSS executables

  Potential drawbacks are:

  - It is a local service which starts tasks in the same machine as the scheduler, thus load-spreading of batch jobs is a manual process requiring access to all the machines involved
  - Cross-computer access for the task schedule is possible, but will require local permissions for a domain account

- Using the scheduler on the database server. As the server is either Linux or Windows the immediate options are: Cron for Linux and Scheduled Tasks for Windows.

  For both situations the benefits of using the database server include:

  - One common place to define triggers

- All triggers use the same clock

With respect to distributed execution of the jobs having the scheduling on the server in both instances suffer from the problem of providing credentials as well as starting processes on remote machines.

The Cron task scheduler with Linux "only" triggers one command hence using the server trigger will involve different logic in Corporate and Professional version of the software.

In either case the NB DSS will need an editor for configuration of the commands to execute as well as supply (a number of) executables which can run common operations in batch.

Scripting (see section 6.4) is a likely vehicle for creation of batch operations to be performed, but stand-alone executables and batch files are also possibilities with the external schedulers. Direct use of the multi-step tasks in the Windows Task Scheduler will enable sequencing of actions, but it is also a possibility that this may be supplied by the NB DSS application as a separate software component.

Logically it is tied to the DSS proxy, which in deployment context is a NB DSS representative (without the GUI) capable of performing tasks such as running scenarios and generating reports in back-end computers. The NB DSS scheduling will make use of the DSS Proxy to control execution of unsupervised tasks.

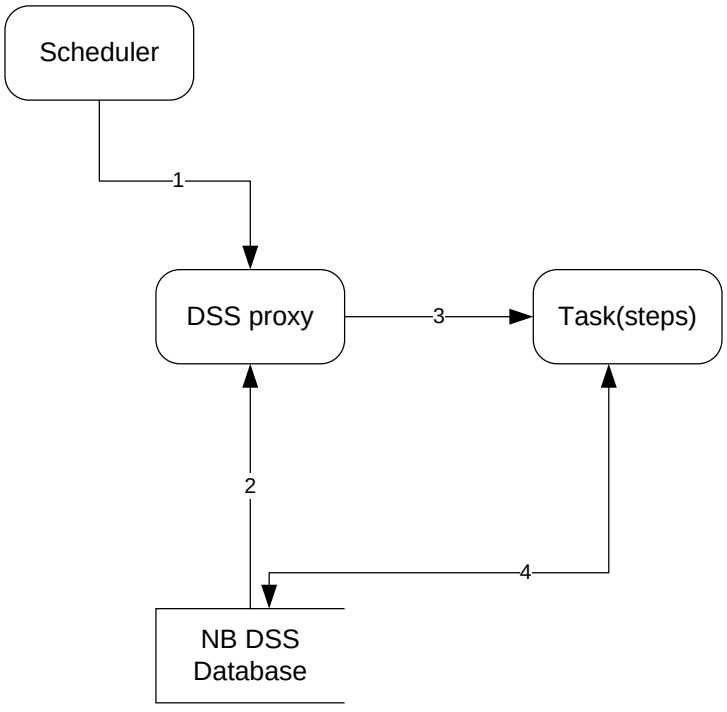

Figure 6.8 Concept of scheduling and batch

Figure 6.7 illustrates a likely scheduling scenario. The scheduler triggers execution of a task – and alerts the DSS Proxy (1). The DSS Proxy reads the definition of tasks (and steps) in the database (2) and executes them (3). Each task (and step) may interact with the database also (4).

Scheduler, DSS proxy and DSS Database may be located on different machines. The scheduled trigger must have permissions to talk to the DSS Proxy, which in turn must have permissions to communicate with the database (be configured to use a dedicated NB DSS login account) and each of the tasks (steps) must also include configuration of credentials to use for database communication, as well as credentials for accessing other resources required to execute the task. The configuration of these external components shall de designed during the detailed design stages.

6.6 Database Reports

The requested functionality for supporting database reports as defined and discussed in the ToR and "4+1" use cases is not detailed at a level that provides for a precise architectural design. However, indicate that the likely components involved in the reporting functionality are:

- A query part
- A report template definition and selection part
- A report part

The query definition part deals with definition, selection and execution of queries. One likely design for this part is based on parameterised queries – either as SQL strings or expressions at a higher abstraction level. I.e. the user can when generating a query select a parameterised query, provide the parameters, select attributes to be included in the output and execute the query. A super-user or system administrator might be allowed to create new parameterised queries.

In connection with running the query, the user can select a template that defines the format of the report. I.e. the generated report will be formatted according to the report template.

The report part deals with how to format the report according to the selected template, how to store the report within the database and possible how to export it.

The Consultant feels a need to further clarify what shall be the functionality of reporting.

7 VIEWPOINT: IMPLEMENTATION

This part of the document describes the system as seen from a software developer point of view, i.e. its implementation. Focus for this section is the GIS integration, Internationalization and data types and unit handling.

7.1 GIS Integration

The basic GIS functionality provided by the DSS Front-end is not based on custom developed code but on integration with 2 3rd party GIS components. These are:

- PostGIS for storage
- ThinkGeo for visualization and simple UI-related GIS processing like unions and intersects

This section describes the integration between the GIS components and the DSS Front-end.

Any GIS system basically consists of 3 parts – storage, processing and visualization of spatial data. The description of the GIS integration architecture is organized accordingly, i.e.

1. How is GIS data stored within the DSS Front-end
2. Where and how in the architectural framework does GIS processing take place
3. Display of GIS data

7.1.1 Overall GIS Architecture

The 3 GIS functionality areas – storage, processing and visualization – are mapped on the DSS architecture as depicted in Figure 7.1 below.

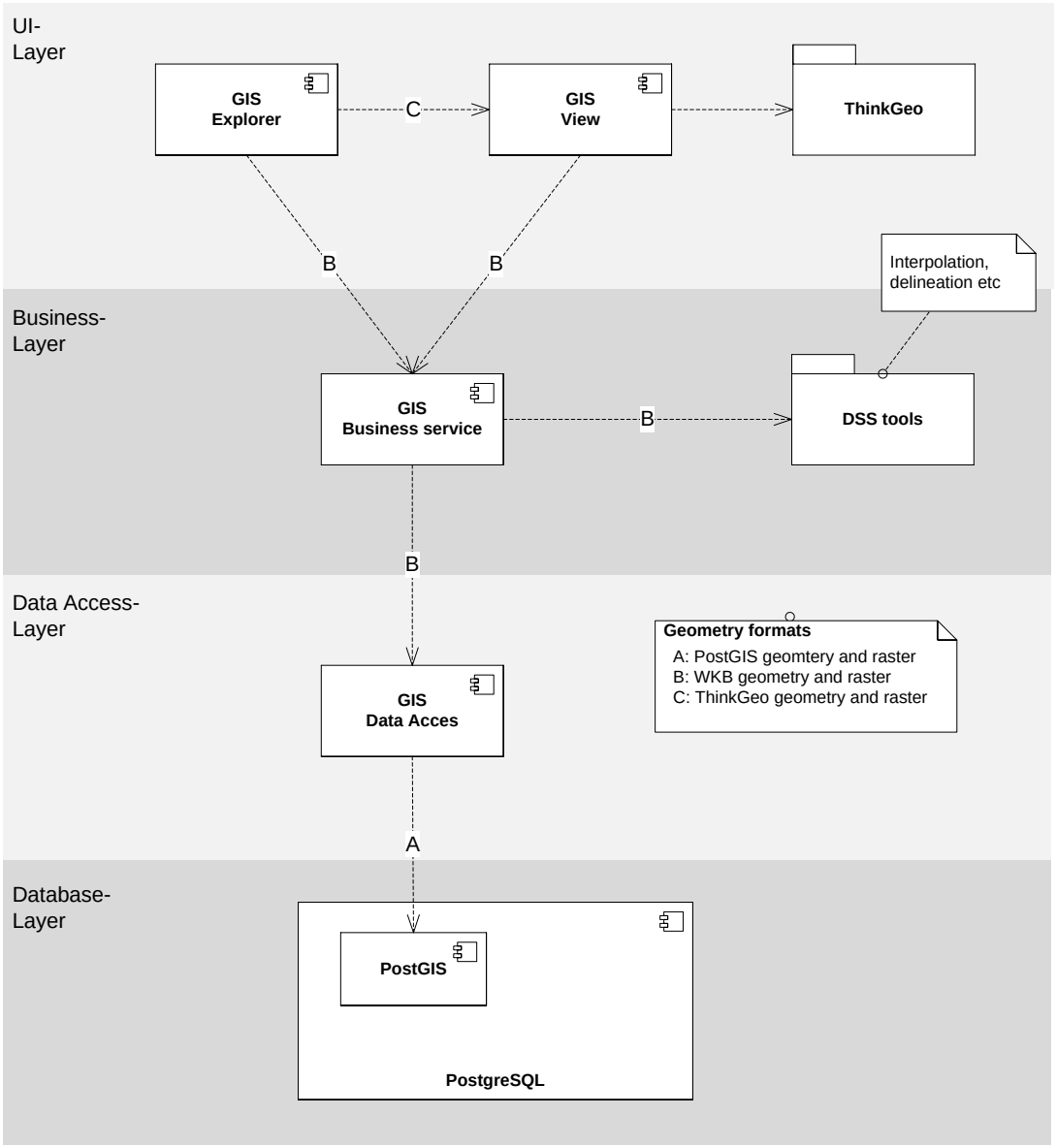

Figure 7.1 Location of GIS functionality on the baseline architecture

Note from the figure that:

- PostGIS and PostgreSQL is together responsible for storage of GIS data.
- GEO processing capabilities is performed by DSS Tools which logically belongs to the business layer and for simple UI-related functionality (zooming, panning, etc) directly in the GIS view.
- Visualization is done solely by the GIS view component which depends on ThinkGeo for rendering and interacting with the view.
- GIS data is converted from PostGIS geometry based data to WKB based data at the data access layer and to the ThinkGeo based format at UI-layer. In this way the influence of the 3rd party ThinkGeo component on the whole system is kept to a minimum. Note also that ThinkGeo resides only at the UI-layer level.

7.1.2 GIS Storage

PostGIS is a spatial extension for PostgreSQL and it complies with the OpenGIS Consortium (OGC) "Simple Features for SQL" specification, and provides high-performance spatial SQL access to GIS objects in the database. I.e. it provides the data types necessary for storing and querying spatial data. This section describes the architectural pattern used for the overall system design. Once the PostGIS component is installed successfully, a spatial database is created in the PostgreSQL database to support all the spatial functionalities.

The OGC compliance comprises the following with respect to storage:

- It supports all the OpenGIS object types
- It supports operations and the SQL schema definitions to insert, query, manipulate and delete spatial objects.
- It supports the OpenGIS WKT and WKB representations of geometries.
- It uses OpenGIS SRTEXT representations alongside PROJ4 representations to provide coordinate system capabilities.

Note: At the time of writing raster data is not supported by PostGIS; but an open source project – WKTRASTER – works on extending PostGIS with raster support. The aim of the project is to support raster formats by adding a new data type – called raster – to PostGIS. It is the ultimate intention to have WKTRASTER embedded with PostGIS. A temporary solution is to store the raster data as BLOB's in the database and have the GIS Manager's data access layer and business layer processing the raster files. ThinkGeo has strong support for raster data.

The DSS Front-end data access layer (DL) as shown in Figure 7.2, also accommodates for spatial data.

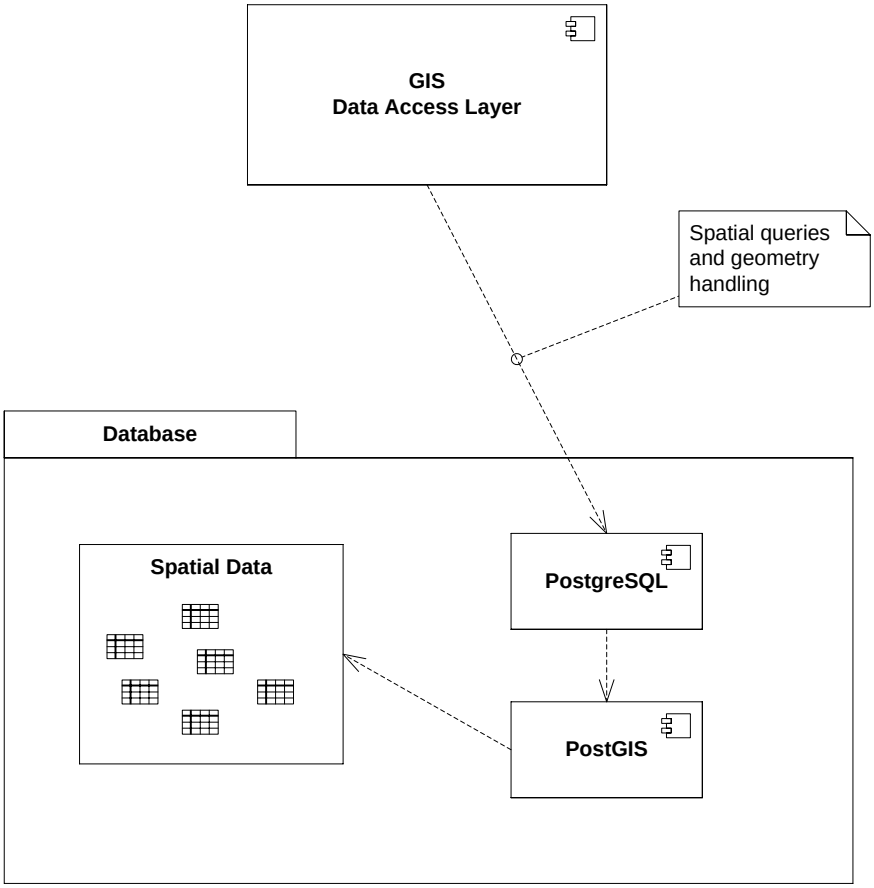

Figure 7.2 Query for spatial data

The DL support for spatial data is based on data type conversion functionality provided by the PostGIS component. PostGIS stores spatial data in the PostgreSQL database using the geometry type – geometry is a specific PostGIS data type added to PostgreSQL through PostGIS. Traditional programming languages like C# do not provide a corresponding data type, i.e. the data type needs to be mapped.

When the DL performs a query – that being a read, insert or create statement - against the PostgreSQL database involving tables with geometry data types, PostGIS will come in action and – and if instructed so in the SQL statement - convert the geometry data type to or from a text string (WKT).

The DL leverages this functionality in order to do pass spatial data to the DSS Front-end layer for processing.

7.1.3 Geo-processing

Geo-processing involves manipulating spatial data for various purposes, e.g. interpolating spatial data. This is implemented with the DSS Tools component that logically resides at the business layer level, see Figure 7.1

Different options exist for implementing the Geo-processing tools:

1. Use of PostGIS geo-processing capabilities. PostGIS geo-processing is made through SQL functions embedded with the SQL queries.

2. Use of ThinkGeo geo-processing capabilities. ThinkGeo geo-processing is made through dedicated geo-processing component.

3. Custom developing the functionality based on standard algorithms.

During the detailed analysis and design stages the preferred way to implement geo-processing will be taken.

7.1.4 GIS Visualization

All GIS visualization and UI interactions are performed through the ThinkGeo component. This component includes all GIS visualization and user interface interaction features required by the DSS. Comprising:

- Pan, zoom, select features
- Forwarding of click and key press events to the DSS Front-end

Additionally ThinkGeo includes a large set of basic GIS functionalities and possibilities for directly interacting with a number of standard database systems – including PostgreSQL and PostGIS. The ThinkGeo control is also capable of reading data from ESRI geo-databases.

7.2 Internationalization

NBI has a specific requirement for support of English and French languages in the application user interface and help files. This section describes the solution with respect to internationalization, i.e. the ability for the DSS Front-end to present itself to the user in different languages.

7.2.1 GUI components – labels, texts

The application will build upon the facilities in and preferred way of the Microsoft .NET Framework. This includes:

- Separation of all localizable texts and string from the code by placing them in resources files
- Translate for preferred languages (English, French) and include the resources with the application. English is the default language of the application
- Use of the .Net Resource Manager to access these resources at runtime

7.2.2 Regional settings – date time formats, decimal numbers

The UI of the DSS Front-end will adjust to regional settings of the client workstation on which it is running in the following way:

- Determine the language used by Windows
- Choose the same language for the application if available. Otherwise choose the default language
- Use regional settings with respect to

- Formats of data in input fields: date, time, numbers and decimal points
- Use date and time in the user specified time zone

- If the user wants to override the language chose by the method above allow an override, for instance via a command-line argument.

- Online help opens in the language matching that of the application

The above is only applicable to the DSS Front-end. 3rd party tools and programs may have limitations with respect to localization.

The SRS discusses how data should be presented to users with respect to unit and suggest that these shall follow the choice of the individual users. In Windows users use the Regional and Language Options control panel applet for defining the format of often used data types like number, data and currency. A similar scheme will be implemented for the NB DSS allowing the individual user to specify the preferred unit for different data types, e.g. preferred units for water levels, rain fall etc.

This scheme will be implemented through the use of the following design decisions:

- Data in the database will always be stored in system preferred units, e.g. water levels in meters.

- Data will - when fetched for display - be converted to the user's preferred unit.

- Data entities in the database will include an attribute defining the data type.

- Data being imported shall have their data type and unit specified.

And supported by the following components:

- The DHI EUM component for engineering unit management which is described in the SRS (see /2/).

- The *SystemUnits* table in the Global data compartment defining the system preferred units

- The *SystemUserUnits* table defining the preferred units on a per user basis

8 VIEWPOINT: IT INFRASTRUCTURE

This chapter describes the system from an IT infrastructure and deployment point of view. The focus for the description is the database, the deployment architecture and requirements to the use of operating systems, network etc.

8.1 Database

This section describes the infrastructure aspects of the database solution that has been selected for the NB DSS. The logical use of the database is discussed in Section 5.1.3.5.

The database solution covers not only the database itself but also other aspects closely related to the selected database. The aspects discussed below comprise the following:

- Description of database and its role in the NB DSS
- Storage of GIS objects
- The data solution technology stack
- Database administration

8.1.1 The Database

PostgreSQL will be used by the NB DSS for all data storage. This database is a very widely used and well-known database and has been so for the at least last 10-15 years. The database has a reputation of possessing good enterprise abilities, being fairly easy to manage, provide comprehensive supporting functionality and – not least – be available for a wide variety of operating systems. The latter supports the requirement that the database part shall be available for both Windows and Linux.

8.1.2 GIS Functionality

PostgreSQL does not provide possibilities for storing and processing spatial data. The necessary data type for storing geometries and the functionality for working with spatial data does not exist. However, the well-known PostGIS extension provides the geometry data type and thus the spatial functionality.

PostGIS is a spatial extension for PostgreSQL which complies with the OpenGIS "Simple Features for SQL" specification, and provides high-performance spatial SQL access to GIS objects in the database. I.e., it provides the data types necessary for storing and querying spatial data.

8.1.3 The Data Solution Technology Stack

The NB DSS design is based on a standard 3-layer application architecture pattern but typically is deployed on 2 tiers – the database tier and the application tier. This is depicted in Figure 8.1 below.

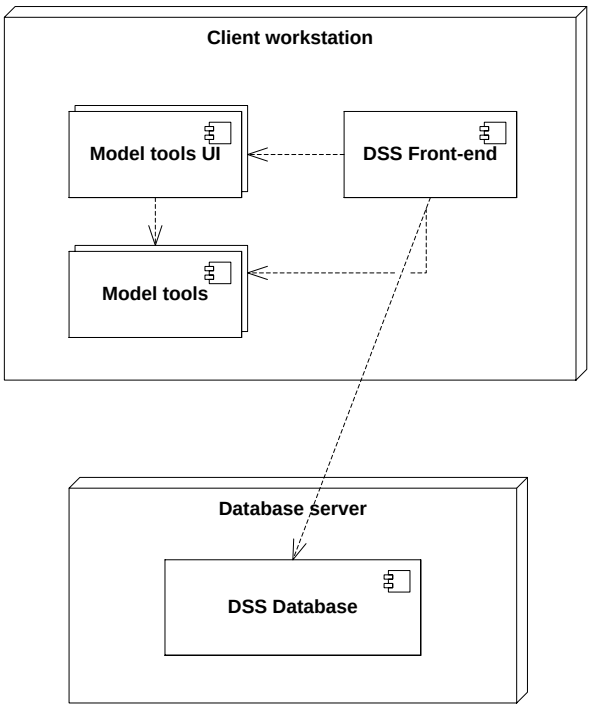

Figure 8.1 NB DSS tiers (simplified) (UML)

The technology stack deployed on the Database Server (database node) and on the Client Workstation (workstation node) enable the DSS Front-end to interface with the database is depicted in Figure 8.2 below.

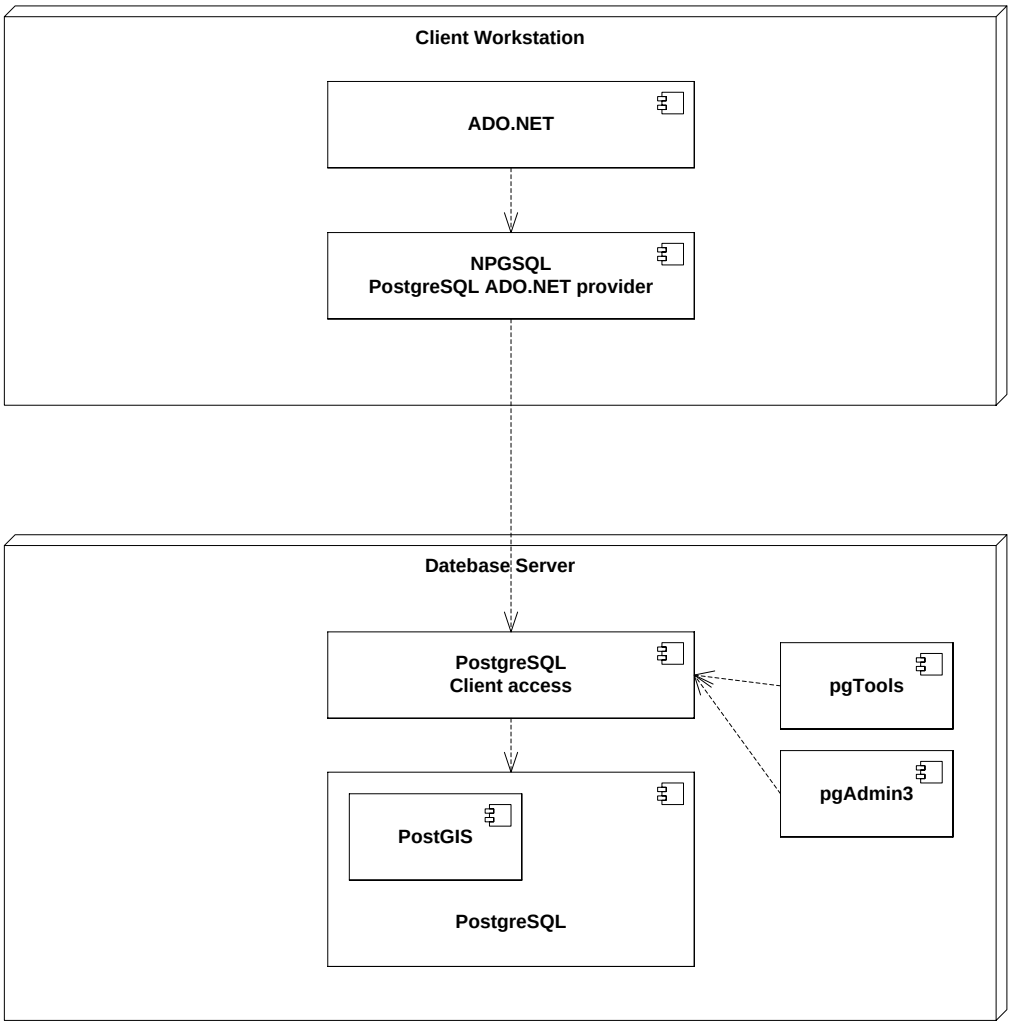

Figure 8.2 Technology stack (UML)

The NB DSS uses Microsoft ADO.NET to communicate with the database. However, because Microsoft does not provide an ADO.NET driver for PostgreSQL, a third data party driver, NPGSQL from pgFoundry, must be installed on workstations running the DSS Front-end.

The anticipated versions of the technology components are as follows:

- PostgreSQL – 8.4
- PostGIS – 1.4
- NPGSQL
- PostgreSQL tools – corresponding to the selected version of PostgreSQL

Changes to these selected versions might occur according to new releases.

The NPGSQL Data Provider communicates on behalf of ADO.NET with the PostgreSQL client access component for interacting with the database.

PgAdmin 3 will be used as system administration console for the PostgreSQL database and a collection of database tools, pgTools, will likely be used for providing backup and restore functionality.

8.1.4 Database Administration

Most database administration tasks can be performed from within NB DSS – e.g., user management, data import and data export. However, there are a few situations where a systems administrator must interact directly with the database, i.e. from outside the NB DSS. An example of this is backup and restore.

The NB DSS system provides description on how to perform such tasks. These descriptions will be based on the use of standard PostgreSQL system administration tools like pgAdmin III, pg_dump, pg_dumpall and pgsql.

8.2 Deployment

This section describes deployment related aspects of NB DSS. The aspects covered are:

- Client-server architecture – a description of the basic pattern used for the application design
- Configurations – a discussion on how the major components of the system can be deployed
- Processes – a description of the processes – and their communication – within the system
- Hardware – a description of the required hardware elements
- Installation – a discussion of the system can be installed

8.2.1 Client-server architecture

A user of NB DSS will frequently perform complex analyses on large data sets, often different analyses on the same data sets (e.g. to generate an optimal set of synthetic time series as preparation for ensemble modelling).

This implies that substantial amounts of data must be transferred over a network. In order to improve the response time of the system (and hence the experience of working with the system) the dependency on network capacity is minimized by deploying the over a 2-tier client-server decomposition pattern. This 2-tier decomposition of NB DSS is depicted in Figure 8.3 below.

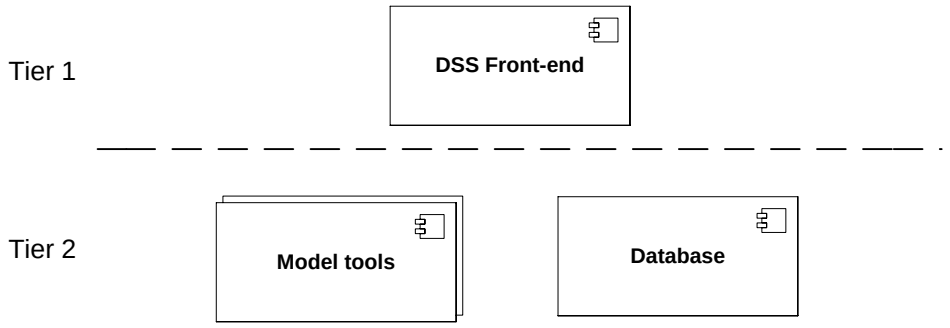

Figure 8.3 Overall NB DSS de-composition (UML)

The DSS Front-end holds all the DSS processing functionality, e.g., time series management and scenario management, and is installed on tier 1. Tier 2 hosts the database and the model tools. This approach has the following advantages compared to a 3-tier application where all or a part of the processing capabilities are centralized on an application server:

- The DSS Front-end can load data into memory and process them without the need for network operations
- Data does not have to transmitted between more than 2 computers in the system

8.2.2 Configurations

The NB DSS can be deployed in a number of different ways or configurations – each having its own advantages and disadvantages.

8.2.2.1 Professional Edition

In the simplest form all the 4 main components are – as depicted below - installed on the same computer (client workstation). This configuration is called the professional edition.

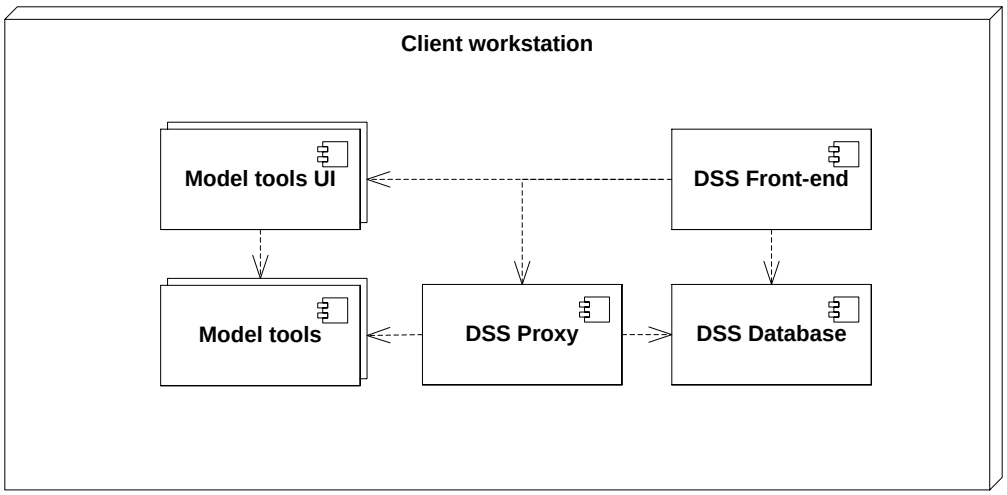

Figure 8.4 Professional edition (UML)

The primary aim of the professional edition is to service organisation or individuals with limited DSS needs or where only one person has to work with the system. The primary limitation of the professional edition is however that it only serves one person.

It will be difficult to apply professional editions in an environment where work groups have to collaborate on e.g. studies. Other limitations to the professional version prevail:

- It requires model tools to be installed locally
- It typically does not participate in an institutionalized data synchronization

8.2.2.2 Corporate Edition

Organisations that have more substantial needs or where groups of people must share data should opt for a Corporate edition. A Corporate edition is composed of the same components as the professional edition, but is deployed differently. A number of configurations are possible. Two of them are shown below in Figure 8.5 and Figure 8.6.

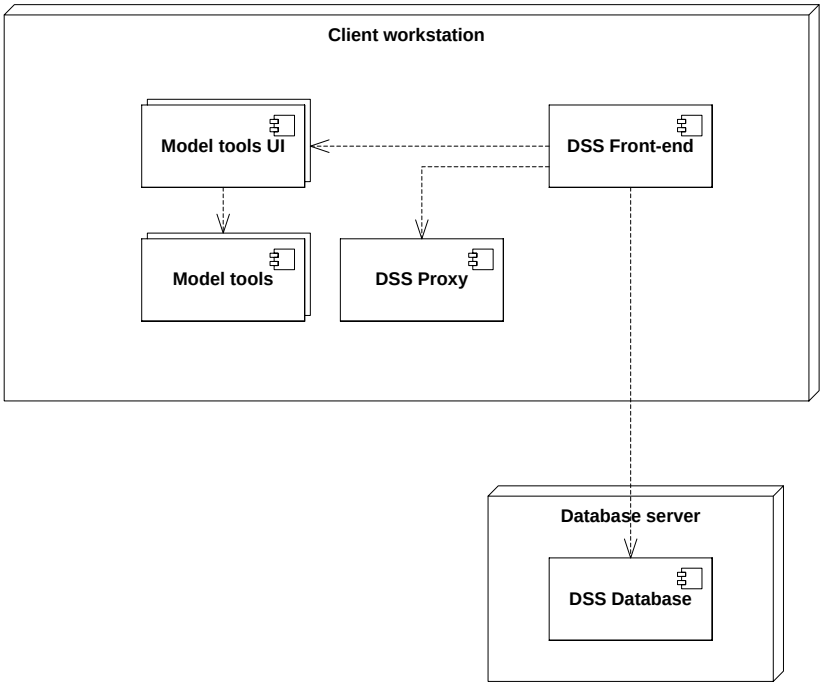

Figure 8.5 Corporate edition - deployment type A (UML)

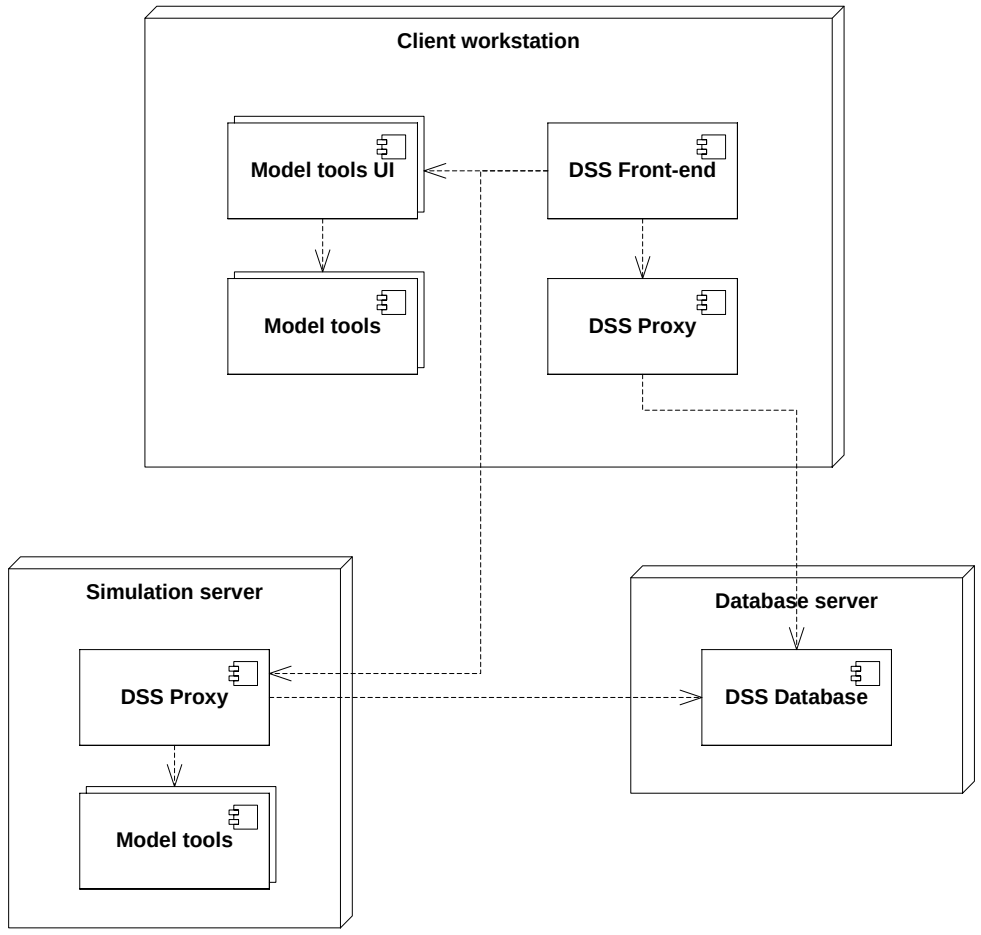

Figure 8.6 Corporate edition - deployment type B (UML)

Figure 8.5 and Figure 8.6 illustrate the following configurations:

- A: A configuration where the system is deployed over 2 computers
- B: A configuration where the system is deployed over 3 computers[12]

The difference between the two options is the installation of the model tools. In option A, these are installed together with the model tool user interface and DSS Front-end on the client workstation, while in Option B, the model tools are installed on a central simulation server.

Advantages and disadvantages with the 2 solutions include:

- Installing the model tools on a dedicated simulation server will make it possible for an organisation to share a common powerful workstation for scenario simulations, thus optimizing the utilization of hardware resources.
- An organisation that undertakes studies likely must have the modelling tools (also) deployed on the client workstation. Partly in order to integrate new models into the NB DSS and partly because most modelling systems will not be able

[12] The DSS Proxy component is a service that on behalf of the DSS Front-end manages simulations.

relay simulations to other computers in the network, such organisations would benefit from a centralized deployment of model tools

- License management of modelling packages might become easier – and potential cheaper – with a single centralized simulation server. Having modelling packages scattered on the network will require a queuing system to ensure license availability (i.e., a dedicated license server).

- A centralized simulation server will make it possible – or at least – simpler to manage a system where the DSS database links to simulation output instead of embedding it.

An organisation deploying a corporate edition must carefully consider these aspects before finally selecting the deployment configuration.

8.2.3 Processes

Each of the 4 components identified in the above paragraph, i.e. DSS Front-end, DSS Proxy, Model Tools (including UI) and the DSS Database, consist of one or more processes. The picture below depicts the processes associated with the components.

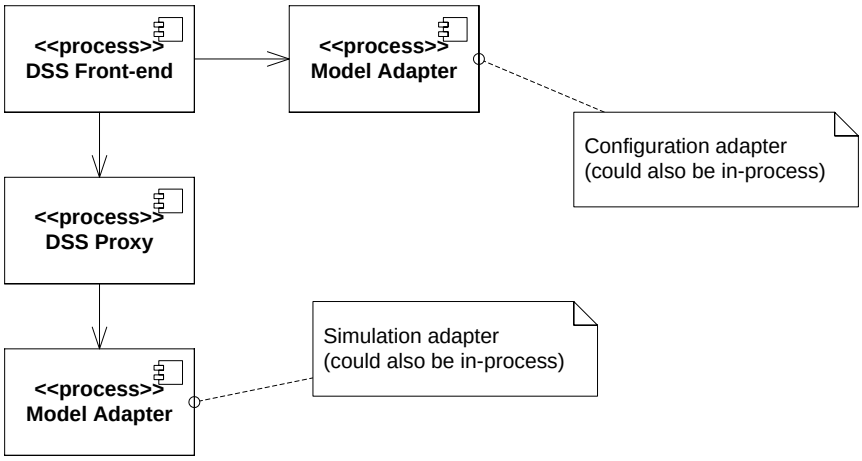

Figure 8.7 DSS Front-end Processes (UML)

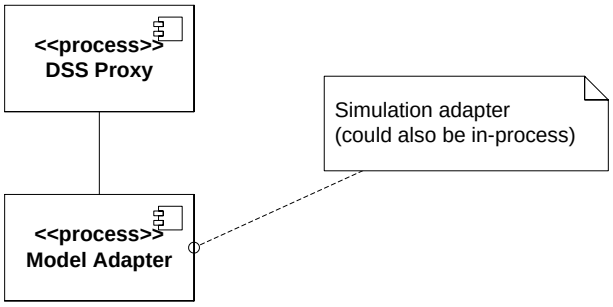

Figure 8.8 DSS Proxy component process (UML)

Figure 8.9 DSS Database processes (UML)

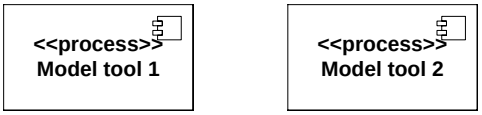

Figure 8.10 Model Tools processes (UML)

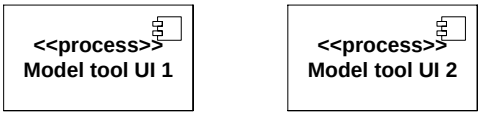

Figure 8.11 Model Tools UI processes (UML)

Note the following from the above figures:

- The DSS Front-end process uses the DSS Proxy process to initiate and monitor model simulations and the Model Adapter process to import and export models
- The DSS Proxy uses the Model Adapter to insert simulation output in the DSS database
- Model Adapters, Model Tool UI and Model Tools come in triples; i.e. for each model tool one model adapter and (likely) a corresponding model tool UI must exist.

Each of the above process is briefly characterized in the table below.

Table 8.1 Processes

Process	Description
DSS Front-end process	This is a process created when the user starts the DSS Front-end on his workstation. It is a standard Windows process running managed code.
DSS Proxy process	This is a Windows service process that is started when the computer boots on which the service is installed. The purpose of the DSS Proxy process is to launch a model simulation without invoking the DSS Front-end process. This is necessary to: • Be able to run a simulation on another computer than the one executing the DSS Front-end process. • Allow the user to exit the DSS Front-end without halting the simulation. • Allowing the user to suspend a simulation (e.g., if the simulation takes all the computer resources, thus making interactive work difficult).

Process	Description
Model Adapter	This is a standard Windows process with the purpose of bridging between the DSS Front-end and the model tools. The adapter will understand the format of the model tools' input and output data.
PostgreSQL service	This is a Windows service or Linux daemon application. It runs the PostgreSQL RDBMS database server.
Model tool	This is a standard Windows process running model simulation.
Model tool UI	This is a standard Windows process running the native user interface to the corresponding model tool. Some modelling systems might combine the simulation part and the user interface into a single executable or might not all provide a user interface at all.

8.2.4 Operating Systems

The various hardware nodes in a NB DSS deployment require different operating systems.

- The Database server can be installed on either Windows or Linux. At the time of writing the exact version of these two supported operating systems are not decided. Likely candidates are one of latest Windows Server operating systems, the latest release of the Windows client operating system and the latest Ubuntu (Linux) Server version.

- The required operating system for the Simulation server varies from modelling system to modelling system. The DHI supplied modelling systems are generally supported by the two most recent versions of the Windows client operating systems – currently Windows Vista Business Edition and Windows XP.

- The DSS Front-end will be supported on the latest Windows client operating system.

Note that if an organisation decides to deploy a Professional edition of the NB DSS system or consolidate the model tools with the client workstation, that organisation must carefully select an operating system that is supported by both the DSS Front-end and the employed model tools.

8.2.5 Hardware

Just as the supported operating systems – as described above – can vary between hardware nodes, so does the requirements for the hardware.

- The Database server should – when deployed in an organisation where the DSS system is mission critical – be installed on proper server hardware. This comprises a system with characteristics as shown below:
    - Disks with high throughput
    - RAID 1 disks for the operating system and database server installation
    - RAID 5 disks for the database files
    - Redundant power supply

- Dual core (or better) processor

- Simulation server – should be a high performance workstation, i.e., processor speed is typically more important than disk speed. The characteristics of such a server comprise:

  - Dual core processor (or better). If the simulation engines installed on the workstation can utilize multiple cores[13], the number of cores should be scaled accordingly. Similarly, the number of processor or processor cores should be sized according to an estimated load – the number of simulations running in parallel.

  - 4 GB of memory or more

  - Sufficient hard disk space to accommodate for the simulation output

- Client workstation – a standard modern workstation or laptop should suffice. Characteristics comprise:

  - 4 GB of memory

  - Dual core processor (or better)

  - High resolution graphics adapter (e.g. 1680x1440) and corresponding monitor

If an organisation chooses to consolidate the deployment on 1 or 2 hardware nodes, the hardware characteristics should be selected carefully.

Additionally hardware requirements:

- The Local Area Network (LAN) connecting the nodes should accommodate for at least 10 Mbps between the client workstations and the simulation server and at least 100 Mbps between the database server and the simulation server.

- The installation – at least the database server and simulation server – should utilize an uninterruptible power supply.

- An organisation could consider duplicating the database server in a cluster to increase redundancy.

8.2.6 Installation and setup Configuration

Installation of a NB DSS system comprises a number of steps:

1. Installation and configuration of the database – this is performed using the PostgreSQL standard installer maintained by EnterpriseDB – see www.enterprisedb.com. After the installation, the database must be configured according to NB DSS supplied installation configurations.

[13] At the time of writing only few of DHI's MIKE products can leverage multiple cores (currently only the flexible mesh version of MIKE 21).

2. Installation and possible configuration of the model tool software – this is performed with model tool native installers. If the model tools are installed on a dedicated simulation server, the DSS Proxy and the model tool's adapter must be installed on the said server.

3. Installation and configuration of the DSS Front-end – this is performed using the installer supplied with the DSS Front-end.