Project Number 260041
SUPPORTING ACTION

EnRiMa
Energy Efficiency and Risk Management in Public Buildings

Deliverable 4.1a: Requirement Analysis of the Decision Support System Engine

Supplement to deliverable D4.1: Requirement Analysis

Start date of the project: October 1, 2010

Duration: 42 months

Organisation name of lead contractor for this deliverable: IIASA

Lead authors: IIASA, URJC, SINTEF, and Tecnalia

Revision: final, August 10, 2012

Project funded by the European Commission within the Seventh Framework Programme (2007-2013)		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Contents

List of Figures

List of Tables

List of Acronyms

AJAX	Asynchronous JavaScript and XML
API	Application Programing Interface
BMS	Building Management System
D4.1	Deliverable D4.1 (Requirement Analysis, October 2011)
D4.1a	This deliverable
DBMS	Data-Base Management System
DoW	Description of work, Annex 1 to the EnRiMa Grant Agreement
DTO	Data Transfer Object
DSS	Decision Support System
DW	Data Warehouse
HTTP	The Hypertext Transfer Protocol
ICT	Information and Communication Technology
JAXB	Java Architecture for XML Binding
JAX-WS	Java Architecture for Web Services
OOP	Object-Oriented Programing
OXM	Object--XML Mapping
RA	Requirement Analysis
SM	Structured Modeling
SMS	Symbolic Model Specification
SMT	Structured Modeling Technology
SOAP	Simple Object Access Protocol
UI	User Interface
WPx	x-th Work-Package of EnRiMa
WSDL	Web Service Definition Language
WS	Web-Service
XML	Extensible Markup Language

Executive summary

This deliverable (labelled D4.1a) serves as the revision of deliverable D.4.1 (Requirement Analysis, IIASA et al, 2011), as well as provides information about the Decision Support System Engine (DSSE) that is useful for the forthcoming activities. The main part of D4.1a presents the DSSE components, namely the DSS Kernel, Solver Manager, and Scenario Generator, and their services to be provided to the EnRiMa DSS. The second part describes the role of the Kubik facility. D4.1a is complementary to the following earlier deliverables: D4.1 (Requirement analysis, October 2011), D1.1 (Requirement assessment, December 2011), and D5.1 (Draft specifications for services and tools, June 2012).

This report starts with an overview of the DSSE components and a summary of the services each of them provides; the overview also explains how the Symbolic Model Specification (SMS) supports consistency of services provided by all EnRiMa DSS components that share (either provide or modify or use) various types of information, as well as outlines how the SMS facilitates effectiveness of development and maintenance of the DSS components. Next, each of the three DSSE components is presented. The DSS Kernel description deals with the following topics: Web-services, data-warehouse and the corresponding data services, and the data privacy; moreover, we present the Kernel architecture, as well as the approach to, and corresponding tools for, development of the DSS components that consume Web-services. The other two DSSE components (Solver Manager and Scenario Generator) are then described. This is followed by a summary of the interface to and between the DSSE components. We conclude the part dealing with the DSSE components by illustrating their roles in execution of workflows specified in selected use cases of the EnRiMa DSS developed in D4.1. The second part of D4.1a consists of a description of the Kubik laboratory building and its services for development and verification of the EnRiMa models.

In addition to fulfilling the EC request for the D4.1 revision, this deliverable contributes to the on-going and forth-coming EnRiMa activities. Most of the D4.1a content has been known to EnRiMa partners from many discussions and the corresponding background notes, results of tests performed at Kubik, as well as from experiments with the Web-services testing-package. However, the request for the D4.1 revision triggered the organization of this dispersed information into a consistent document; it will guide the EnRiMa partners in the effective use of possibilities offered by the available methodology and technology (especially the Symbolic Model Specification and the Web-services), as well as by the capabilities of the Kubik facility. The scope of Web-services and data services, although well defined in the DoW, is now specified in more detail, and illustrated by the presentation of services that they provide to workflows defined in the use cases developed in the Requirement Analysis (IIASA et al., October 2011).

1 Introduction

This deliverable (further on referred to as D4.1a) has been developed in response to one of the requests the Commission specified in the letter received on 13.7.2012, namely, "to take into consideration the comments under "remarks" for both D4.1 and D8.3 and to submit the revised deliverables within one month of reception of this letter." This deliverable concerns only the request related to D4.1; the D8.3 revision will be submitted separately.

The scope of D4.1a covers not only the above mentioned remarks, but also issues raised during the review meeting on June 5th, 2012, as well as the more detailed recommendations included in the Technical Review Report for the EnRiMa project, dated 20.6.2012, in particular, the following two elements of item 1.b (page 3):

- "Deliverable D4.1 covers the requirement analysis for the entire DSS without an overview of the system functional components underlining the ones to be developed in the project. Deliverable D4.1 has to be revised and resubmitted in order to have clear conclusions linking the requirement analysis with design elements and software architecture."
- "The KUBIK facility is not considered, neither in the Sankey diagram evaluation neither for Requirement Analysis (in deliverable 4.1). In this context it is not clear at which level of testing the KUBIK facility will be used. At this stage of the project, the added value of using the KUBIK facility is not evident and the impression is that the ENRIMA project could run adequately without the involvement of the KUBIK facility. A clear justification regarding the need of involving the KUBIK facility into the ENRIMA project is required."

Therefore, we have extended the scope of the requested revision in order to not only respond to the EC request but also to report on the related work done after the D4.1 delivery in October 2011. For justification of this deliverable content it is desired to summarize the following issues:

- The D4.1 submitted in October 2011 was the first EnRiMa deliverable. Its committed scope was the Requirement Analysis (RA) of the EnRiMa DSS Engine (DSSE). However, a RA of the DSSE had to be based on the RA of the whole DSS. Therefore, the consortium decided to extend the scope of D4.1. The delivery time of D4.1 did not enable a presentation of the description of the DSS components and the architecture; these both critically important elements have been designed later.
- The EnRiMa partners agreed already during the review meeting on June 5th, 2012, that it would be useful to develop a description of the DSSE components and their roles in fulfilling the RA, as well as present the DSS architecture. Following this discussion, the deliverable D5.1 (submitted in June, 2012) includes the corresponding descriptions of the DSSE components, and of the DSS Kernel architecture. These elements have been extended and included in D4.1a (this deliverable) together with several new elements that we consider useful for the D4.1 revision.
- D5.1 (submitted in June, 2012) describes the overall architecture of the EnRiMa DSS, its main software modules, and their relations; hence, it partly addresses the above quoted EC recommendations that were formulated before D5.1 was submitted. Therefore, this report focuses on the DSSE components, which was the originally planned scope of D4.1, and presents the topics that could not be included in D4.1.

- D4.1 provided RA for the public buildings as sites for which the EnRiMa DSS will be implemented. Kubik is a laboratory building designed for diverse tests (including simulations of given situations, e.g., different occupancies, energy loads, and energy flows); therefore, it was not rational to prepare a DSS RA for Kubik. However, Kubik is critically important for proper development and verification of models developed for the EnRiMa DSS. In order to justify this statement, we describe in D4.1a the Kubik role, and illustrate it by summarizing the tests to be made at Kubik for development and verification of the EnRiMa models.
- The D4.1 is, due to its wide scope, a rather long (84 pages) document with the content that is needed and remains valid, but does not meet the EC revision recommendations summarized above; in order to fulfill the EC request for D4.1 revision, rather extensions by new parts than modifications of the old ones are needed. Therefore, a rational way appears to be the development of a self-contained supplement. Thus, we have taken this approach, and developed this deliverable (named D4.1a) as a supplement to D4.1.

This deliverable consists of two parts, each addressing the corresponding (above cited) element of the request for the D4.1 resubmission. The first part presents the DSSE components, namely the DSS Kernel, Solver Manager, and Scenario Generator, and their services to be provided to the EnRiMa DSS; it presents their design, software architecture, shows how each component contributes to meeting the DSS requirements specified in D4.1, as well as how they shall work together. The second part describes the role of the Kubik facility; it explains the tests to be carried out at Kubik, and their role in calibration and verification of EnRiMa DSS models; it also presents the interactions between the EnRiMa DSS components and the Energy Management System of Kubik, which is essential for verifying the integration of EnRiMa DSS with the ICT infrastructure of actual Building Management Systems (BMSs).

The remaining part of D4.1a contains the following elements. Section 2 provides an overview of the DSSE components; it also includes, in response to the discussion during the review meeting, a summary of the role the Symbolic Model Specification (SMS) plays in supporting consistency of services provided by the DSS components. The next three Sections provide descriptions of each of the DSSE components, i.e., the DSS Kernel, Scenario Generator, and Solver Manager, and the services each of them provides to the EnRiMa DSS. Following the discussion during the review meeting, the Kernel description includes a presentation of its architecture, as well as comments on the data privacy issues. Section 6 outlines the interface between the DSSE components. The services provided by the DSSE components are illustrated by their roles in the execution of workflows specified in selected use cases developed in D4.1. The Kubik laboratory building and its services for the development and verification of the EnRiMa models are described in Section 8. The conclusion is presented in Section 9.

2 Overview of the Decision Support System Engine (DSSE)

The DSS Engine (DSSE) is the backbone of the EnRiMa DSS providing to its users three types of state-less services, each through one of the DSS components, the DSS Kernel, Solver Manager, and Scenario Generator. Diverse types of actual DSS users and stakeholders are characterized in Section 3 of deliverable D4.1 (IIASA et al. 2011). They will use the DSSE services in a transparent way, i.e., consume them through the User Interface (UI) being developed by WP5, and documented by the corresponding deliverables. Thus, diverse DSSE services will be used whenever requested by users through the UI. The UI will organize workflows controlled by the users, and composed of tasks needed for achieving the corresponding DSS functionality.

The general assumption of the EnRiMa architecture is that all DSSE services used by end-users will be accessed through the UI, and provided through Web-services (WSs). This approach has several important advantages; we mention only three of them. First, it enables robust integration of heterogeneous (i.e., developed and run on diverse hardware and software platforms, possibly at distant locations) components into a DSS fitting particular needs. Second, it supports efficient development and modifications of the DSS components, with minimum interdependences during the whole cycle of software development use, and maintenance. Third, it facilitates software reusability.

The DSSE has modular structure illustrated in Figure 2-1, and is designed to make it reusable for other buildings without substantial modifications. Actually, the modifications are limited to adaptation of the Symbolic Model Specification to the new building, and provision of the corresponding data.

The three DSSE components, each described in a subsequent section, have the following main functions. The DSS Kernel provides the set of harmonized WSs that enables integration of heterogeneous components into the DSS. The Scenario Generator provides realizations for the stochastic parameters for a scenario tree, which is then used for stochastic optimization. The Solver Manager provides solutions of the stochastic optimization problems. These descriptions are preceded by presentation of the role the SMS plays in integration of heterogeneous software components into the EnRiMa DSS.

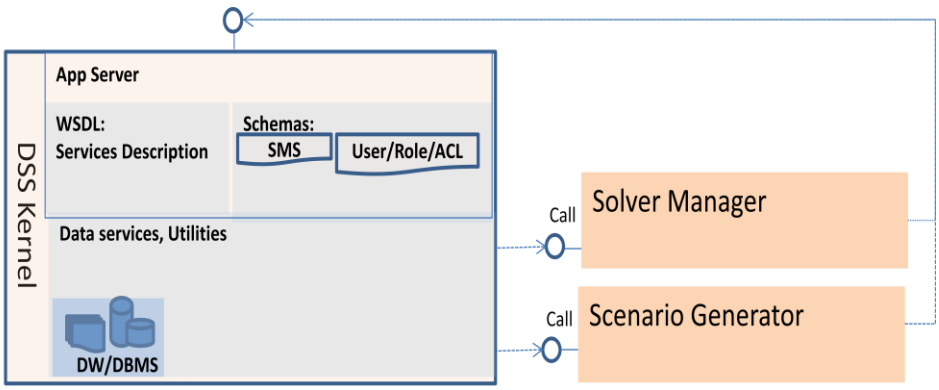

Figure 2-1 Overview of the DSS Engine architecture.

2.1 Role of the Symbolic Model Specification (SMS)

2.1.1 Overview

Semantic consistency of objects used in different DSS components is a key necessary condition for the DSS usability. In the EnRiMa DSS it is achieved by adopting the Structured Modeling Technology (SMT) approach (see e.g., Makowski, 2005), which uses the Symbolic Model Specification (SMS) as the common (for all involved DSS components) source of specification of model entities. The SMSs of operational and strategic EnRiMa models are maintained by the DSSE; therefore we present here the role of SMS in the DSS development and maintenance.

Models used for model-based DSS are composed of three types of compound entities, i.e., coefficients, variables, and relations, as well as sets (including subsets, optionally indexed by members of other sets) of indices. The indexing structure, implied by the sets of indices, is used for populating the compound entities into the corresponding sets of actual entities (coefficient, variables, and relations); thus, the sets serve as dictionaries of eligible (for each specific version of data) values of the corresponding indices. This approach implies consistency of the model components between applications that deal with different phases of the model development. We comment here on only key elements of the corresponding modeling cycle:

- The SMSs (for operational and strategic models, respectively) provide an adequate (for the EnRiMa DSS) specification of parameters, variables and relations, as well as the corresponding indexing structure; thus corresponding model instances support analysis of the decisions and the representation of consequences of their implementation. Hence, the SMS is the right basis for assuring consistency between interdependent processes of the data (to be used as model parameters) management, and the corresponding model generation and analysis.
- The data will come from different sources, and may be provided by different users at different times. The data management processes will be done by the DSS users within the workflows supported by the UI; the SMS may include additional data attributes that support the organization of these processes. Data management includes versioning, which is a very useful approach for data sets in which small parts are modified, e.g., for comparative analysis, or for periodic data updates. The authorized users will decide which data versions can be used for defining model instances; only the corresponding model instances will be used for model analysis.
- The consistency of data with the SMS will be checked, during the data management processes, through WSs provided by the DSS Kernel. Therefore the model parameters will be consistent with the SMS.
- All DSS components will therefore operate on consistent sets of data, in particular model parameters defining model instances, parameters used for scenario generation and stochastic optimization, and the results of the corresponding analysis. Using the SMS will also assure that the generated model instances will be consistent with the model documentation (both generated from the same SMS).

2.1.2 Elements of the SMS

As already mentioned, the EnRiMa models, like any non-trivial model, are composed of compound (generic) entities (parameters, variables, and relations) that are populated into

actual entities through indices that take values from the corresponding sets. Depending on the needs of modelers and users, specification of the model entities optionally includes additional (to those needed by mathematical programming representation) attributes. To illustrate this approach, we provide examples of the necessary and optional attributes.

2.1.2.1 Common attributes

Here we list attributes that are common for all types of entities, i.e., sets, parameters, variables, and relations. At the minimum, each entity has two attributes:

- Label (symbol): a (very) short string; used as the entity id, e.g., in relations, data services, etc. The symbol has to be unique within a model (or a name-space, if the latter is used; see below).
- Iterator: a vector (empty for scalar entities) composed of indexing sets; members of indexing sets are used for populating a compound (generic) entity (a set, parameter, variable, or relations) into the corresponding collection of entities. In other words, a number (defined by all combinations of members of the corresponding indexing sets) of entities are created as instances of one compound entity.

The following optional attributes are also worth considering, either for all or some types of entities, because they provide information useful for other (not just those belonging to DSSE) components:

- Short name: a string (preferably short) having an easy association with the meaning of the entity, used e.g., in listings, tables, etc.
- Description: a 1-2 lines of text, used e.g., in UI-hints, documentation, etc.
- Measurement unit.
- Name-space (can be used for either keeping the corresponding entity *private* within a model, or sharing its specification with other models).

The above list contains the most commonly used attributes; other attributes can be added, if desired.

2.1.2.2 Sets

Here, a set means a container for indices of a specific type. Sets are often organized in hierarchical structure. The root (base) set contains all indices of a specific type. Subsets can optionally be indexed. If subsets are used, then each needs to have, in addition to the common attributes, the following attribute:

- Parent set: indicates that the set is a subset of the corresponding parent set; e.g., if S is a set of energy sources, then $Sren \in S$ might be a subset of renewable sources, and $S_t \in S$ might be a subset of sources available at time $t \in T$, where T is a set of time periods. Note that S_t is an indexed subset, and $Sren$ is a not-indexed subset.

Members of an indexing set are used as indices for instantiating entities, some of them may have an iterator composed of several indices. Therefore it is practical to associate with each member (index) two descriptors (ids):

- Symbol: a (very) short string; used as the index id, especially for internal purposes (e.g., handling of data and results) e.g., for a country index: at, be, es, no, se, uk; and
- A word, used in the communication with (less advanced) users; e.g., for a country index: Austria, Belgium, etc.

2.1.2.3 Parameters

In addition to the common attributes, the following attributes of parameters should be used for parameters:

- Type: math-type, e.g., integer, float, double; also deterministic or stochastic; implementation of the latter depends on the agreed representations of stochastic parameters.
- Lower and upper bounds (either constants or other parameters).
- Source: specification of the data source (optional, but very useful for organizing data management processes).
- Group: for grouping parameters (optional, but very useful for management of the data access rights).

2.1.2.4 Variables

In addition to the common attributes, the following attributes should be used for variables:

- Type: math-type (same as specified above for parameters).
- Lower and upper bounds (either constants or selected parameters).
- Role (optional, but very useful for analysis of large models):
    - decisions (inputs), controlled by the DSS user;
    - external decisions (inputs), exogenously given;
    - outcomes (outputs), used for measuring the consequences of the implementation of inputs; to be used as criteria (objectives); auxiliary variables introduced for various reasons (e.g., to simplify model specification, or to allow for easier computational tasks).

2.1.2.5 Relations

Relations (in mathematical programming often called constraints or functions) should have, in addition to the common attributes, the following attributes:

- Lower and upper bounds.
- Group: (optional, for grouping relations).
- Activity: (optional) binary indicator to enable excluding the relation from the model.
- Role: (optional, for presenting relations in diversified ways, e.g., as assignments, definitions, or constraints).

2.1.3 Representations of the SMS

EnRiMa DSS shall have three SMS representations of each model (with optional versions), to be developed, and be available for all DSS components:

- The DW representation, transparent for developers of other DSS components, to be handled by the Kernel.
- XML-based representation, containing information requested by a specific DSS component; it will be provided by the Kernel through the corresponding Web-services.
- Human readable representations (HTML and PDF), to be generated from the XML representation.

2.1.4 Summary

Structured Modeling (SM), originally proposed by Geoffrion (1987), provides a proven framework supporting the whole modeling cycle, which is especially effective for non-trivial

models developed by collaborating teams. Separation of model specification from management of data used for model instances was a break-through in modeling technology, and it is now considered as a key element of good modeling practice. The SMT (Makowski, 2005) extends the original SM framework by offering additional functionality, in particular optional attributes of the SMS, which facilitate development and maintenance of the UI, as well as separating the modeling activities into two parts. First, handled by users through a UI; second, provided by a server (DSS Kernel) that maintains all persistent elements of the model and the corresponding modeling process. SMT also extends the SM approach by providing the Data Warehouse (DW) that handles all data of the whole modeling process in a way transparent to the model users. The SMS plays a key role in assuring consistency of these data. The Kernel provides Web-services through which all needed modeling information is exchanged. Thus, the Kernel is model-independent, and therefore can be re-used for other models.

In order to illustrate the scope of advantages offered by a more complete (than the minimum necessary for a model specification) definition of SMS, we outline some of them:

- Defining and using optional attributes of model parameters (such as data source or data group) substantially ease organization and maintenance of data management processes by the UI.
- Information about available data, and its completeness, can be structured according to the needs of handling specific data subsets, and thus make the data management processes through UI much easier to implement and maintain.
- Dynamic generation of forms (e.g., driven by specifications of subsets of parameters downloaded from SMS by a Web-service) makes form maintenance much easier than forms with hard-coded field specifications.
- Use of optional attributes enables validation of form's fields by the client, which is by far more efficient than data validation by the server, and subsequent handling of possible errors. Validation may include not only the data types, but also conformity to the corresponding range of feasible values (also downloaded by a Web-service).
- The UI forms handling indexed data may use choice lists composed of downloaded values of indices, thus eliminate errors caused by specification of undefined values of indices.
- Optional attributes of variables facilitate specification of preferences (for optimization-based integrated model analysis) and diverse analyses of results.
- Maintenance of user data-access rights is much easier to be handled by the UI.

3 The DSS Kernel

3.1 Overview

The DSS Kernel provides state-less services requested by the DSS users through the User Interface (UI) or directly through other DSS components, including applications integrated with the ICT infrastructure of the corresponding building. The Kernel provides the needed functionality through the corresponding Web Services (WSs). The latter requires also "back-office" applications needed for actual implementation of WSs, such as checking consistency of the provided data with the SMS, organizing the data provided as Data Transfer Objects (DTOs) into structures suitable for effective handling by the Data Warehouse (DW) implemented in a DBMS; organizing data available from the DW into DTOs used by applications consuming the WSs; user handling; access control; user's on-line problem reporting and comments; preparing data for diverse on-line reports to be provided by the UI for developers and users.

Thus, the DSS Kernel consists of:

- WSs for interfacing the DSS components that need to exchange data through the Kernel.
- Data Warehouse (DW) supporting handling of the DSS data that needs to be stored.
- Data services providing access to the DW.
- Back-office applications supporting implementation of the WSs.

These elements are presented in the corresponding Sections below. Additionally, the following topics are presented in the subsequent Sections:

- The Kernel architecture.
- Data privacy issues.
- Consuming the WSs within the DSS components.

3.2 Web-Services (WS)

3.2.1 Overview

A Web-Service (WS) is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with WSs in a manner prescribed by its description, i.e., using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards. The main advantages of using WSs are as follows:

- Flexibility: WSs are only concerned with the transfer of semi-structured data between software components. This gives flexibility of implementation, allowing systems to adapt to changes of requirements, technology etc. without directly affecting users.

- Interoperability: WSs support exchange of data between heterogeneous (i.e., developed and run on diverse hardware and software platforms) software components that can run at distant locations.
- State-less: WSs are independent, self-contained requests; they do not require information from, or state of, other requests; they are also independent of the content or state of other WSs.

Thus the clients (applications that consume WSs) can build various business processes that typically need exchange of data available from different sources and in diverse formats.

3.2.2 Functionality of the WSs

The Kernel shall provide WSs supporting the following top-level functionality:

- Interfacing the DSSE with the UI developed in WP5.
- Interfacing the Kernel with the scenario generator and solver manager.
- Other functionality needed for a DSS such as user handling, access control, user's on-line problem reporting and comments, providing data for diverse reports for users and developers, backups, etc.

In order to meet these requirements, the WSs provided by the Kernel shall support:

- Storing various types and sets of the provided data (parameters of the model, specification of model analysis tasks, results of model analysis, data needed for administration of the DSS).
- Checking consistency of the provided data with the SMS, and with the specified sets of indices.
- Management of data versioning and acceptance.
- Definitions of model instances, analysis tasks.
- Providing information about the status of the solver tasks, and availability of solutions.
- Providing the stored data to the DSS components in diverse (corresponding the needs of the client applications) DTOs.
- Management of the DSS users, including authentication, specification of roles, access rights.
- Triggering execution of the scenario generator and solver manager.

3.2.3 Summary of WSs specification

From the client (applications that will use Web-Services provided by the Kernel) perspective, a WS consists of four components:

- Envelope containing data needed for identification of the context within which the WS is used. The envelope includes elements such as: id of the user and his/her credentials, model id, data set and version, etc.
- Request (specification of the service). It identifies a specific WS, and provides the corresponding parameters.
- Response (specification of the response to the service). The message sent by WS server that may include one or more of the following: HTTP status, SOAP body contains the data of the response, error messages, and warnings.
- DTO (Data Transfer Object) containing structured data needed for processing the request or the response. DTO optionally contains specification of objects transferred

as attachments for requests of storing/retrieving objects (files) that are not processed by the Kernel.

3.3 Data-Warehouse (DW)

The DW shall be designed and implemented within a suitable DBMS, using the technology available from the SMT, thus be also efficient for handling huge amounts of data with complex indexing structures. The corresponding DBMS schema should be independent of the data specification, and be transparent for the client applications. The data services used for accessing the DW should assure consistency of all persistent elements of the modeling process, and support control of access to the data. However, the data services shall be hidden from the client applications; they shall access the data only through WSs using the DTOs suitable for their data structures.

The DSS Kernel shall maintain the following data types:

- SMS.
- Model data, including sets, parameters, and results of analyses (all consistent with the SMS).
- Model instances.
- Model analysis tasks (specifications and results of the tasks corresponding to the requested runs of the scenario generator and solver manager).
- Data of authorized users of the DSS, together with their roles as users of the EnRiMa DSS.
- Model journal, i.e., automatically gathered information about modeling activities passing through the DSSE.

3.4 Data services

The data services shall be developed for two different purposes, therefore we present them separately.

3.4.1 Internal Kernel data services

The internal Kernel data services will provide functionality needed by Kernel components. These services will be hidden from the client applications, and independent of a specific data structure implied by the SMS. They will support management of the data provided to, and requested from, the Kernel through WSs and the corresponding DTOs:

- Convert data provided in diverse DTOs into structures suitable for the DW.
- Organize data from DWs into diverse DTOs.
- Organize and maintain data structures for handling indexing structures for compound model entities, including support for indexed subsets of indices.
- Read/store data from/to DBMS-based DW.
- Check consistency of the uploaded/requested data with the SMS.

- Support data persistency and versioning.
- Provide diagnostics (completeness regarding the SMS) of a selected data version.
- Manage access rights to the DSSE and the (subsets of) data.
- Maintain the modeling journal (automatic documentation of the modeling process).

3.4.2 Interim data provision from test sites

The DSSE shall host a small dedicated activity that will provide data from the test sites in the period starting with the termination of WP1 until the prerelease version of the UI (to be developed by WP5) will be available; this will be combined with testing the WSs. This activity does not contribute to the Kernel development; it shall use the results of WP1, in particular tasks 1.2 and 1.3, and provide results to the corresponding activities of WP5.

3.5 Back-office applications

In addition to the internal Kernel data services, diverse applications shall be developed for providing services such as user handling, access control, feedback from users, data for on-line reports to be provided through the UI, backups, etc. The needed functionality shall be provided by a harmonized collection of applications, complementary to the internal Kernel data services.

3.6 DSS Kernel architecture

The Kernel, in order to effectively process the wide scope of WSs and be reusable, has modular structure illustrated in Figure 3-1. We first summarize the communication with and within the Kernel. The Kernel is accessed by the external components through WSs; the Kernel components developed in the Java programming language communicate through Java methods. Next, we outline functionality of each Kernel component.

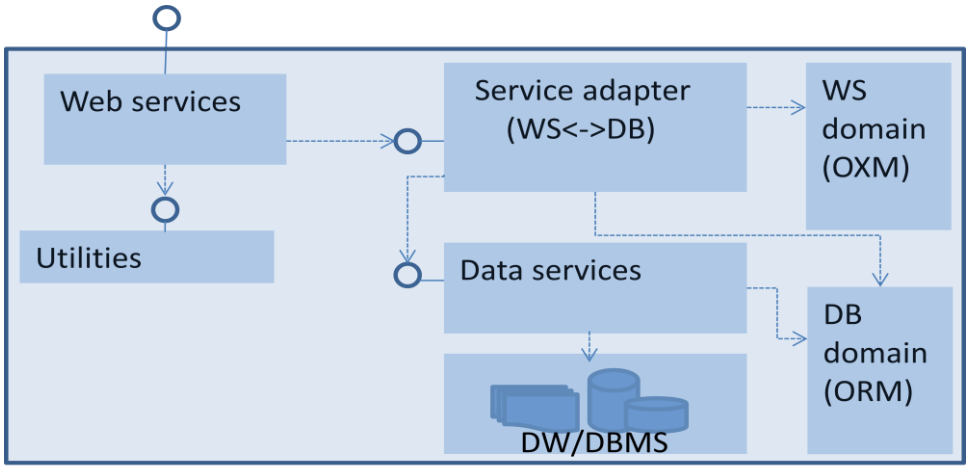

Figure 3-1 The DSS kernel components.

The function and characteristics of each Kernel component are as follows:

1) Web services: publish the contract (as the WSDL file) through lightweight application service (Tomcat) and the endpoints implement the Web-services through calling the service adapter.
2) Service adapter: provides link between WSs and the internal data services; transform formats between Web services domain objects and database domain objects; makes the data services transparent (a black box) for components that do not belong to the Kernel.
3) Web services domain: the DTOs based on the contract (WSDL file) are used for generating objects through an Object/XML mapping (OXM) tool (e.g., JAX-WS, JIBX, XStream, gSoap). Such tools also generate classes for applications consuming WSs in other DSS components (UI, Solver Manager, Scenario Generator, data wrappers and other applications integrated with the ICT infrastructure of each building); more details are provided in Section 3.8.
4) Database domain: the Java Persistence API (JPA) entity objects generated using the database schema through object-relational mapping tools, such as Hibernate, Eclipse link, Toplink, etc.
5) Data services: transactional business logic, read/store data from/to DBMS through JPA which will handle conversion of DTOs into the Data-Warehouse schema that will be independent of the DTOs thus remaining transparent for clients and stable, i.e., not requiring modifications when DTOs will be modified.
6) Data Warehouse: dedicated data structures implemented within a DBMS, accessible to external (to the Kernel) clients only through the WSs provided by the Kernel. Therefore neither a DBMS choice nor the corresponding DBMS schema influences other (than Kernel) DSS components. Currently, PostgreSQL is used for prototyping, and will be used for the final Kernel version.
7) Utilities: a container of diverse back-office applications that support various functions needed by the Kernel.

The multi-layered representation of the Kernel is illustrated in Figure 3-2. The top layer contains the WSs published within the WS-domain in the WSDL format, and endpoints indicating a location for accessing the service. This domain is shared by all components of the EnRiMa DSS. The middle layer contains the service adapter which maps the requested operations into the domain data services. Data services layer provides stateless transactional business logic, the data access layer provides interface to the data warehouse built on a DBMS. The data access and services layers share cross-cutting applications, e.g., for handling logging, transaction management and security.

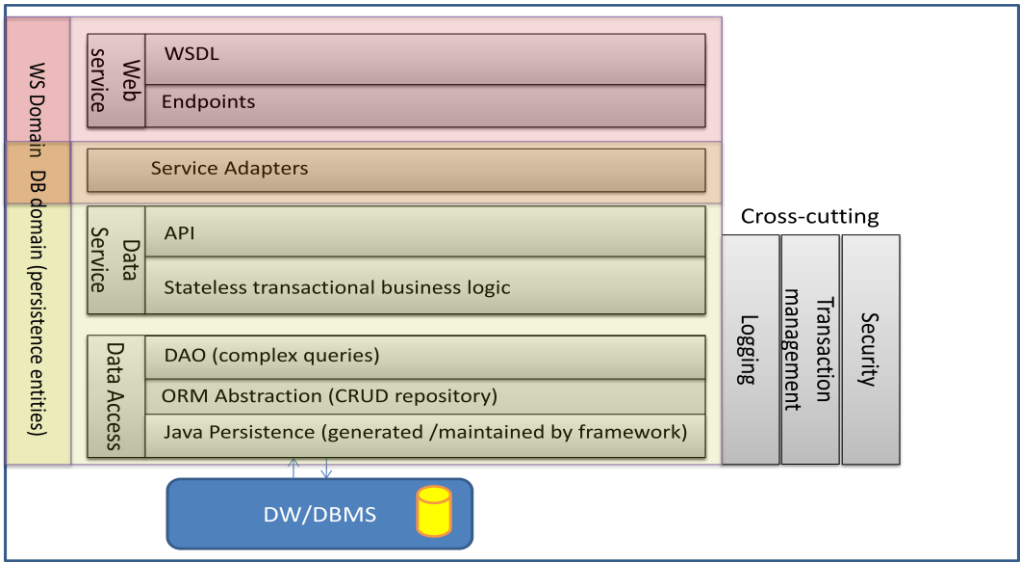

Figure 3-2 DSS Kernel layers.

3.7 Data privacy

The data privacy policy is decided by the EnRiMa consortium, and conforms to the corresponding standard rules. The Kernel shall provide mechanisms to implement this policy. We comment here on key issues only. The data handled by the EnRiMa DSS is of two types:

- Personal data of the DSS users: it will include a minimum set needed for the user authorization, assigning and verifying her/his roles in using the DSS, and email address for communication. All of this data will be kept confidential, and will be provided only to the DSS applications that actually need them.
- Data used for the DSS models, and resulting from their analysis. The ownership of this data belongs to the entity that provides it. An authorized person of this entity (either the EnRiMa partner or the site manager) will decide the data access rights. These rights will be managed by the Kernel.

The data access rights of each user will be decided during the DSS configuration, and implemented in the standard way. The DSS administrator will grant to a person designated by each site the role of administrator of access rights to the corresponding site-related data. The site administrator will then manage the access rights to this data. A similar approach will be used for access rights to the data developed by the EnRiMa partners.

The Kernel shall implement the standard approach to the data protection. Access to the data will be through WSs consumed in the UI and in the authorized applications integrated with the ICT infrastructure of each test site. Each WS request will contain credentials of the user running the corresponding application. These will be checked with the access rights defined by the authorized persons. Moreover, the Kernel shall maintain the model journal, which will support the monitoring of the data access, and thus help to detect possible abuse of the access rights.

The above outlined mechanism will be used during the project duration. The EnRiMa consortium will decide what shall happen with the data after the project termination. Depending on this decision, the data will either be made available to the specified

institution(s), or will be permanently removed from the DBMS and the corresponding back-up files.

3.8 Consuming Web-services in the DSS components

Figure 3-3 illustrates the philosophy of developing and using the WSs. One starts with the requirements from clients, i.e., other DSS components (UI, Solver Manager, Scenario Generator, Data Wrappers and possibly other applications to be integrated with the ICT of the sites that will require WSs), and develop the XML schema and the WSDL file that meet the requirements. After this is completed, the clients can use the WSs directly. Marshalling (serialization) the WS-domain objects (DTOs) to XML, unmarshalling XML to WS-domain objects, and translations between the WS-domains and DBMS-domain are totally transparent for the clients.

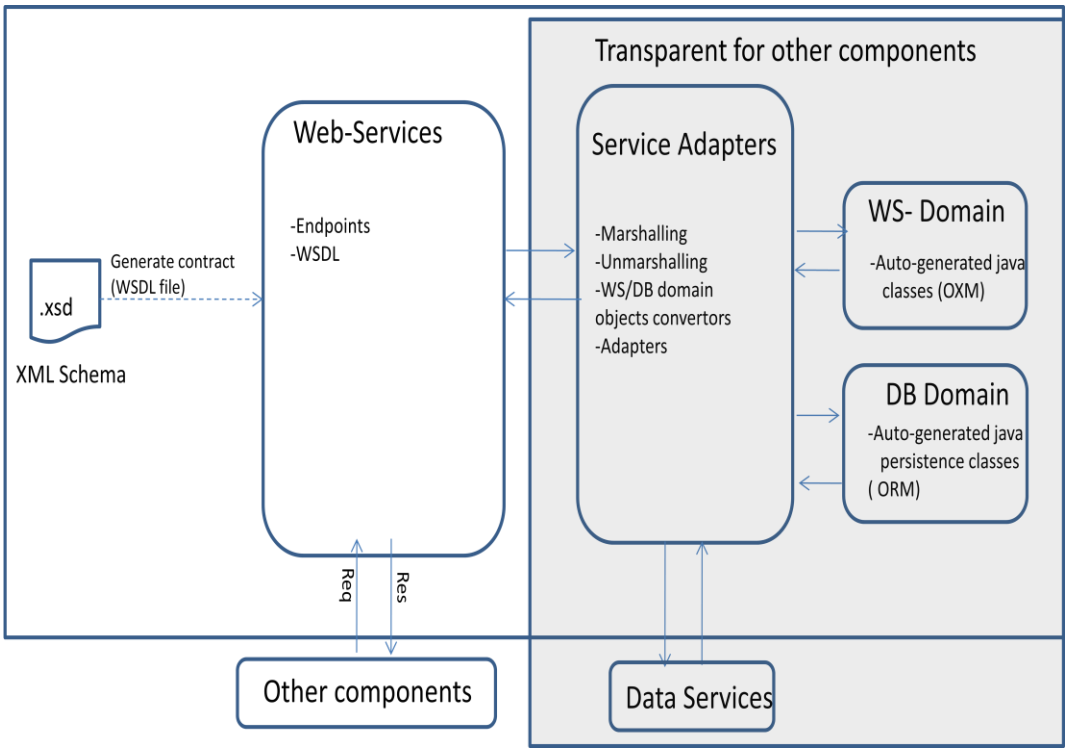

Figure 3-3 DSS Kernel Web-services.

Moreover, there are tools for generating definitions of objects (e.g., classes) directly from the XML schema for all widely used programming languages and tools. A selection of such tools is illustrated in Figure 3-4. Therefore, following the philosophy outlined here, and using the corresponding tools, substantially reduces the resources needed for the development of applications in diverse components, enables parallel development, supports consistency between these applications, and thus contributes to the effectiveness of the software development and to the reliability of the applications, as well as of maintenance and reusability of the DSS components.

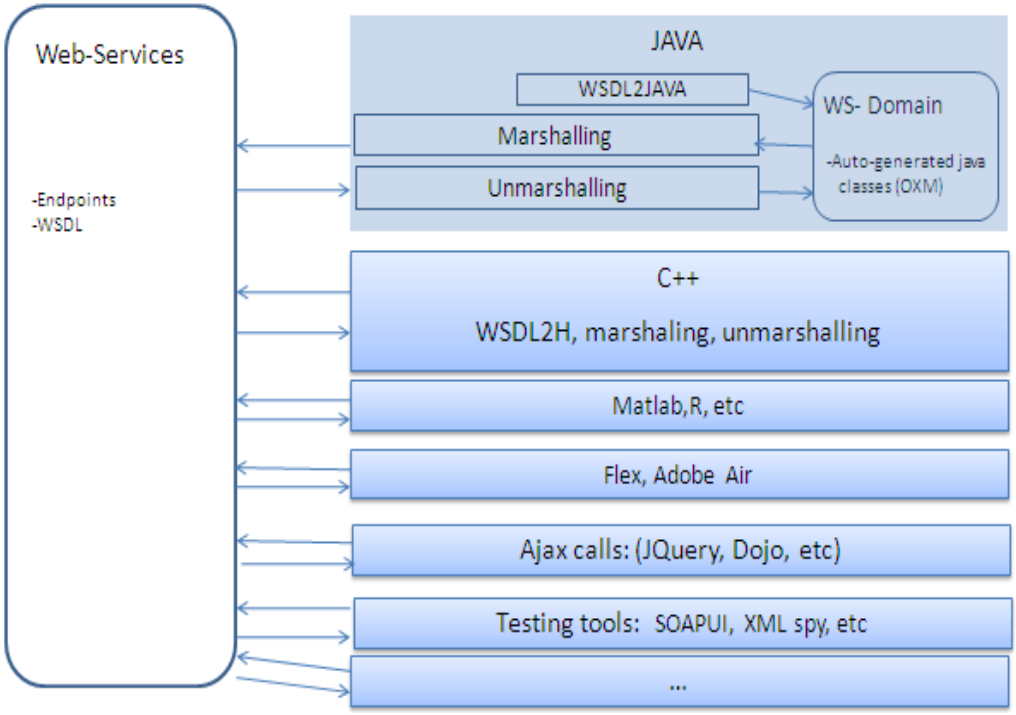

Figure 3-4 Examples of technology for the DSS Kernel clients.

3.9 Summary

It is worth stressing that the EnRiMa architecture based on the WSs and SMT enables effective development of DSS components that can be combined into a robust model-based DSS. The WSs provided by the Kernel enable efficient interface between components that consume WSs, and use of the SMS assures data consistency across all components accessing data through WSs. Moreover, the DTOs can be specified according to the needs of each component, and the public-domain tools support the automatic generation of the corresponding classes that can be directly embedded into the client applications. The DBMS schema used for implementation of the DW is independent of data models used by clients; moreover the use of the DBMS is transparent for the client applications. The client applications can be developed on heterogeneous hardware and software platforms, and run at distant locations. There are public-domain tools supporting use of WSs within all programming languages and software tools commonly used for modeling tasks. The SMT has been successfully used for collaborative interdisciplinary research on the development of large and complex models.

All these arguments justify the statement that the EnRiMa architecture is based on efficient and robust modeling methodology and technology that supports efficient development and implementation of the DSS, as well as its use, maintenance, and reusability.

4 Solver Manager

4.1 Overview

The Solver manager will be the module within the DSS Engine which will provide the solution of the stochastic optimization problems. This module will be independent and will work in coordination with the rest of the DSS Engine modules, namely: the Scenario Generation tool and the Kernel. As a low-level tool (that is, without direct interaction of the end user), it will take the models and parameters provided by other modules (e.g. database or UI), and after preparing data and applying the appropriate algorithms and computations a solution will be returned. This solution will be used by other modules, mainly to store and present the results to the user (eventually the decision maker).

4.2 Functionality of the Solver Manager

The features that will be included in the solver manager will be:

- Capability to consume the Web services provided by the Kernel.
- Capability to generate different file formats (e.g. MPS, GAMS, AMPL, etc.).
- Capability to communicate with different third party software, both stand-alone solvers and optimization software.
- Specific solvers and optimization software including stochastic optimization capabilities.

The communication with other modules of the DSS will be made through the WSs of the DSS Kernel described in Section 3. Thus, the operations inside the Solver Manager will be transparent (black box) from the perspective of the rest of the modules.

4.3 Inputs and outputs

The inputs of the solver manager will be:

- The model, from the Symbolic Model Specification.
- The problem instance, including the parameters of the model and the actual sets to be considered in the problem.
- The scenario tree, from the scenario generation tool.
- Building configuration, options and decision maker's preferences.

The outputs of the solver manager will be:

- The optimal values for the variables.
- Further analysis, depending on the solver used, e.g. sensitivity analysis.

4.4 Solver Manager Components

The components of the Solver Manager are represented in Figure 4-1. The interface will consume the WSs in the kernel, and provide the results once the problem is solved. The data preparation (both input and output) modules will interface with the solver using the suitable protocols for it. The solver is called from the Data preparation programs.

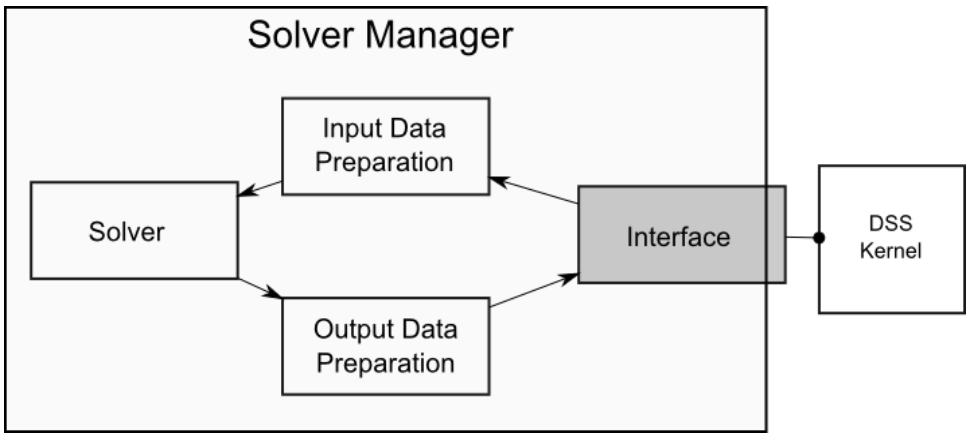

Figure 4-1 Solver Manager scheme.

For the data preparation modules, R programs will be used. The data provided by the Interface, will be prepared and translated into the solver required format. Then, a call to the solver is done, requesting the appropriate output. Finally, the output is processed and sent to the kernel through the Interface.

As the selection solver is a task out of the scope of this deliverable, it is not defined here which solver will be used (it will be done in Task 4.5). In any case, as it is an independent part of the solver manager, any solver can be supported. For the prototypes, calls to the GAMS software and to R solvers' APIs are being used.

There are several ways to interface R with XML services, directly using packages such as XML or using Python interfaces. The final solution for the interface will be decided while developing the data preparation modules, in close cooperation with the kernel developers.

5 Scenario Generation Tool

The scenario generation tool provides realizations for the stochastic parameters for all nodes in a scenario tree, based on input data about the parameters with their statistical properties and about the structure of the scenario tree. This means that it transforms the information the user has about these parameters to information which can be used by the optimization model. In this sense, the tool does not generate new information; it merely processes all available information in a best possible way. The tool itself is completely embedded within the DSSE, making it (normally) invisible to the average user. For a more advanced user, input and output values may be visible as they are communicated to and from the tool, but beyond this, the tool will appear as a "black box" also to these users. A detailed description of the tool is given in deliverable D3.2, “Scenario generation software tool” (SINTEF, 2012).

The scenario generation tool will have three main inputs; module type, parameter data and scenario tree:

- Module type (operational or strategic) - specified by the user. It might also be locked at installation time in the case where the operational and strategic modules are used by different users.
- Data for the stochastic parameters (for more details, see deliverable D3.2):
    - in case of the strategic module
        - weather prediction; automatic download from some weather-service provider - the choice of provider might be country-specific, or even site specific.
        - long-term trends of energy prices - provided by local EnRiMa partner; country or site specific.
        - technology development: efficiency and prices - provided by the EnRiMa team; common for all installations.
        - government subsidies and regulations: provided by local EnRiMa partner; country specific.
    - in case of the operational module
        - historical weather data; either from a weather-service provider, or from own database (site-operated weather station).
        - electricity and heating prices - in case of long-term contracts, the values are stored in a database, with possibility of manual updating; in case of real-time pricing for electricity, we will need a tool to download the prices for the next period from the electricity provider (automatically).

- Scenario-tree structure: time periods, number of stages, number of branchings. This will likely be set at the installation time. Could be made changeable in case of an advanced user.

In case of missing or insufficient data for some of the parameters, we can either consider the parameter as deterministic (i.e., it has the same value in all the scenarios), or we can use an expert opinion to estimate its statistical properties. Since this would mean some adjustments to the tool, it would have to be done at installation time.

The outputs of the tool are tables with values of the stochastic parameters for each node in the scenario tree, together with information about the tree structure (parent/predecessor node, stage and probability). This basically consists of a set of realizations of each stochastic

parameter for each scenario tree node – and "directions" how to put them in the multistage stochastic model to be built. The main purpose of these tables is to serve as an input for the optimization modules, though we will also provide a visualization tool so the user can examine the scenario-tree values before using them.

6 Interface to and between the DSSE components

The WSs provide a simple, flexible, robust, and effective interface between heterogeneous DSS components. As described in Section 3, tools are available for consuming the WSs provided by the Kernel. Here we comment on two issues. First, the execution of a stand-alone DSS-component can be triggered. Second, how downloading diverse data-structures can be organized. Both issues are illustrated by the interface between the Kernel and the Solver Manager illustrated in Figure 6-1.

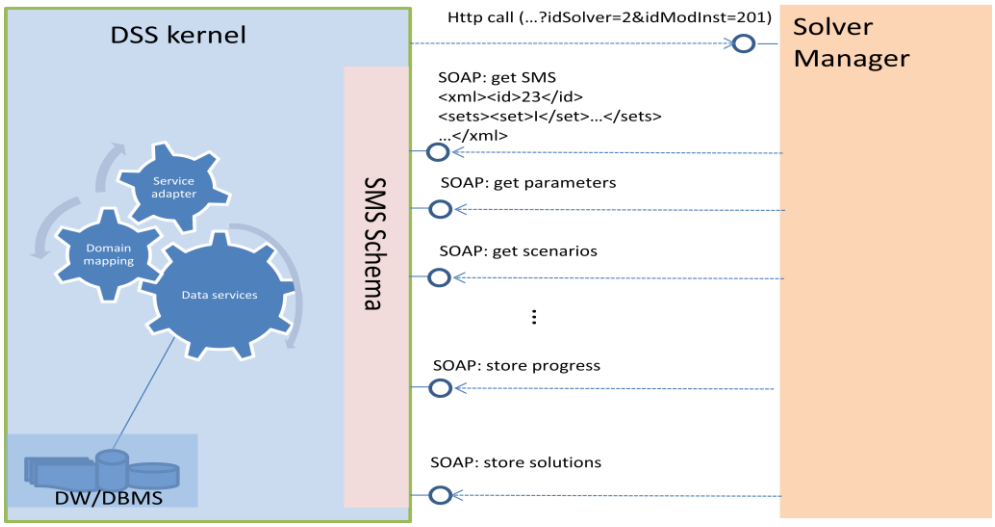

Figure 6-1 Interface between DSS Kernel and solver manager.

6.1 Triggering execution of stand-alone DSSE components

The execution of a stand-alone component (such as Solver Manager or Scenario Generator) can be started by a simple http call that passes a small number of arguments identifying the requested service, and optionally used for the caller identification. Such a call can be generated directly by the UI, or requested through a corresponding WS to be executed by the Kernel. Embedding the required functionality into the application is easy: there are standard solutions enabling same binaries to be executed through either an http-call or a command line statement. Both types of calls use the same set of the call arguments that can be processed by one C++ or Java class. Such an approach also greatly simplifies the development of the application; the same binaries can be tested using the command-line calls, and made available as a Web-enabled application.

6.2 Consuming WSs for downloading data and uploading results

Some stand-alone applications operate on complex data structures organized into classes according to the OOP paradigms. We use the Solver Manager (here simply referred to as the solver) to outline how the WSs combined with the SMS support effective download and

upload of the corresponding data. The approach is illustrated by the sequence of the selected solver actions:

- Solver is started by the http call with arguments identifying the computational task.
- After processing the call arguments, the solver uses a WS to get the data fully identifying the task. These data will contain id's of all needed components, i.e., SMS, model instance, type of analysis, parameters of the analysis, scenarios prepared by the scenario generator, etc.
- Each of these data components can be handled by objects of the corresponding class, e.g., C++ or Java or R. The corresponding data structures define the DTOs which can be used for WSs supporting the download of the corresponding data identified by the elements of the response of the first WS used by the solver.
- The solver can use the WSs to download each needed data components directly to the objects of the corresponding classes. The order of downloads shall correspond to the solver needs; from the Kernel perspective any order can be used because WSs are not interdependent.
- During the optimization process, the solver can use a WS to report the optimization progress.
- After the optimization is completed, the solver uploads the results to the Kernel using the WSs with DTOs corresponding to the structures used for optimization results.

Figure 6-1 outlines just an example of the WSs used to downloading the needed data components. Actually, the real solver implementation may use another composition of DTOs, if it corresponds better to the solver data structures. Moreover, we point out that the consistency of data with the SMS is checked during the data commitment; therefore the downloaded data will be consistent with the SMS.

7 DSSE services supporting workflows defined by use cases

In order to present the roles of the DSSE components and their interactions with the UI, we show here how each of them will contribute to achieving the required DSS functionality defined by workflows specified by the four use cases selected from Section 9 of deliverable D4.1 (IIASA et al., 2011). The use cases were selected with the aim to illustrate the key functionalities of the DSSE components. Therefore, the presentation focuses on the services provided by the DSSE. The roles of the UI and of the applications to be integrated with the ICT of each building are only outlined because they will be presented in the forthcoming deliverables of WP5 and WP6.

7.1 Initial data set

This use case (Section 9.5.2, p. 74 of the D4.1) is representative for using the WSs during the model development, which is being done by the EnRiMa partners. The initial data sets for both EnRiMa models (operational and strategic) will be prepared by the model developers, and provide the basis for subsequent data modifications (versioning). We summarize the WSs functionality supporting each step (copied below in the *italics font*) specified in this use case:

1. *Run automatic generation of the data space for model parameters and results*. The data space will be made available by the Kernel for each submitted SMS. It includes not only the model parameters and results of the model analysis, but also other data needed for supporting the whole modeling cycle. Therefore, this step of the workflow is performed during the DW initialization.

2. *Import available data from the data warehouse.* WSs will provide information about the available data in DTOs corresponding to the needs of the UI. Also either the whole data set or parts of it can be requested by WSs, and will be provided as the response to the corresponding request. Thus, the data will be provided in structures suitable for handling through the UI.

3. *Receive or develop the missing data.* WSs will handle the upload of data provided in DTOs consistent with the SMS. The model developers will use these WSs either by inputting the data to the forms provided by the UI (with associated actions executing the corresponding WS) or by dedicated applications that consume the needed WSs.

4. *Verify the data completeness.* Data completeness can be verified in several complementary stages that shall be controlled by the UI. We mention here only those most commonly used. First, the SMS offers the possibility of specifying data types and ranges. Second, a WS will provide this information to be used for the standard form validation procedures by UI. Third, WSs will provide the lists of valid values of indices corresponding to the data items being processed; such a list can be used as a choice list within the corresponding form (or application) to support using only appropriate values of indices. Fourth, the WS used for uploading the data (from either a form or an application) will check the consistency of uploaded data with the SMS. Fifth, WSs will be available for providing information

about data items (and groups of data, if defined in the SMS) already uploaded to the DW.

5. *Commit data.* Each data subset uploaded through a WS will be stored in the DW, if the provided data is consistent with the SMS, and with the specified values of indices. An authorized user can change any data until the corresponding data (sub-) set is committed. A WS with the request for committing data will lock it, and define the corresponding data version, which will be available for specification of a model instance. The locked data cannot be changed; however, the committed data set can be used as a basis for further data versions. This approach to data handling is necessary for meeting three requirements: first, the data persistency and replicability of results; second, efficient handling of data modifications; third, providing the committed data subsets for reuse as parts of data of another model, or another instance of the same model.
6. *Send information about the data availability*. WSs will provide information about both committed and stored (but not yet committed) data (sub-) sets. This information therefore will be available either through the UI or dedicated application, both using the available WSs.

7.2 Data for operational planning

This use case (Section 9.2.1, p. 68 of the D4.1) is representative for data management processes, in which the users shall be supported by the workflows organized by the UI. In this case the workflows should support consistent execution of all nine steps; the corresponding WSs will provide the requested Kernel functionality supporting this process. Below we comment on the roles of the DSS components in each of the steps specified in this use case:

1. *Select a version of symbolic model specification.* A WS handling the request for providing the list of SMSs of a selected model will return the corresponding list. The list will contain the information provided with the SMS submission (including the user-id, date, optional description and notes). The list can be presented by the UI as an annotated choice list, and the user will select one of its items.
2. *Select a base data set.* Support of this step will be the same as for Step 1.
3. *Receive all needed data updates from external sources.* The user (or users responsible for specific sub-sets of data) will check the corresponding sources. This can be supported by the site-specific applications. WSs can be consumed in these applications (as explained in Step 3 of Section 7.1). Data conversions (from the formats provided by the sources to the DTOs corresponding to the data objects used by the applications processing data) shall be done by site-specific data wrappers.
4. *Receive all needed data updates from local sources.* The approach shall be similar to that summarized in Step 3. The BMS may upload data to the site-specific data wrapper module. Alternatively, the wrapper may download the data from the building ICT, if requested through the UI, or a site-based scheduler triggering tasks

based on either the status of the available data or time intervals. Each of these applications can consume a corresponding WS, for either to get the needed information from the Kernel, or upload the data to the Kernel. Moreover, a WS will support uploading unstructured data that will be available for later downloading and further processing by a dedicated application.

5. *Process/adapt data.* Data processing and adaptation shall be done interactively (through the UI), optionally supported by the underlying dedicated applications, including site-specific data wrappers. The WSs shall be used in the same way as described in Step 3. Thus, the user shall review, change, and approve all parameters to be used in the analysis through the corresponding forms of the UI. This can be done in several stages, each dealing with a subset of data. At each stage, a WS can be used for querying the current state of the corresponding data subsets.
6. *Optionally, run the scenario generation (see the corresponding use-case).* Support for this Step is presented in Section 7.3
7. *Check data completeness.* Support for this step is presented in Step 4 of Section 7.1.
8. *Commit data.* Support for this step is presented in Step 5 of Section 7.1.
9. *Define the corresponding model instance*. This shall be done by a simple UI form providing optional fields for the model instances description and notes. A WS will be available for handling the corresponding data submission.

7.3 Scenario generation

The workflow of this use case is specified in Section 9.4.3, p. 72 of D4.1 (IIASA et all, 2011). It consists of the following steps:

1. *Run the scenario generator.* Preconditions for running the scenario generator are selections of (1) a parent model, and (2) a data set for the scenarios. Both will be supported in a way similar to that described in Step 1 of Section 7.2. After the selections are completed, execution of the scenario generator will be triggered by a simple https call, either directly by the corresponding UI form, or through a dedicated WS that will execute such a call. After the scenario generator is started, another WS can be used for getting information about computation status.
2. *Analyze the diagnostics from the generator*. The results of the scenario generation will be provided by WSs, and therefore available for analysis through the UI. Another WS can be used by the Scenario Generator for providing users with optional information supporting analysis of the diagnostics, and/or characteristics of the generated scenarios.
3. *Upload the scenarios into the DW.* The scenarios will be uploaded through WSs used directly by the scenario generator. The user however, shall decide about committing them to a specific version of data that will then be available for operational and strategic planning.

4. *Send the notification on the availability of the scenarios.* This will be supported in the way described in Step 6 of Section 7.1.

7.4 Operational planning

The workflow of this use case is specified in Section 9.2.2., p. 69 of D4.1 (IIASA et all, 2011). It consists of the following steps:

1. *Select a model instance.* Same support as described in Step 1 of Section 7.2
2. *Select a type of analysis.* Same as Step 1. Additionally, we note that different types of analysis can be supported by the solver manager.
3. *Select parameters of analysis.* Same as Step 1. Additionally, we note that some parameters of the analysis may be predefined, and therefore their values will be available through a WS to be displayed in a dialog provided by the UI.
4. *Start the computations, and wait for notification of availability of results.* Similar to the support presented in Step 1 of Section 7.3.
5. *Analyze the results.* Similar to the support presented in Step 2 of Section 7.3.
6. *Get approval, if appropriate.* This is the site-specific procedure, which can be supported by a dedicated workflow implemented by UI.
7. *Implement the HVAC control parameters.* Also this is the site-specific procedure, which can be supported by a dedicated workflow implemented by UI, which may involve dedicated applications integrated with the ICT infrastructure of the building.

We note that the strategic planning use case contains a very similar workflow, and thus the corresponding support by the WSs will be similar.

7.5 Summary

The presented DSSE support for the workflows belonging to different use cases shows also the flexibility and reusability of the WS-based approach, as well as advantages of the modular approach, which provides the users and the client-applications with all the necessary support without involving clients in details of the pretty complex underlying processes handled by the multi-layer Kernel architecture described in Section 3. The client applications can request and submit the data in DTOs that best fits the way each application processes the data, and is independent of the way data is maintained in a DBMS. This is a qualitative advantage in comparison to the traditional approach, in which the applications had to adapt to a specific DBMS schema, which moreover had to be changed whenever the data structure was changed.

8 Kubik laboratory building

The aim of this section is to explain the purpose of the tests to be carried out in Kubik, the usefulness of these tests for the EnRiMa project and DSS operational module and the potential of Kubik laboratory building.

Secondly, the interactions between EnRiMa's DSS (Decision Support System) architecture specification and the existing Energy Management System in Kubik will be analysed.

8.1 DSS operational module tests in Kubik

Within the frame of the operational module of the DSS being developed in EnRiMa, Kubik plays a key role.

It has been mentioned in previous chapters of D4.1 that one of the objectives of the DSS operational module is to support the building operator in the decision making for operating the HVAC system.

8.1.1 Objectives of the tests to be performed in Kubik

The principal goal is to test the DSS operational module for its analysis and characterization as well as possible improvements suggestions definition.

Particular objectives are:

- to test the building envelope and HVAC model behaviour,
- to test the energy demand calculation in a "controlled environment".

This means testing the key equations of the operational module in what refers the HVAC system behavior, defined in D2.2, with the least uncertainties, hence the use of Tecnalia´s laboratory building Kubik.

In Figure 8-1 below a graphical representation of the links between Kubik tests and other tasks is shown.

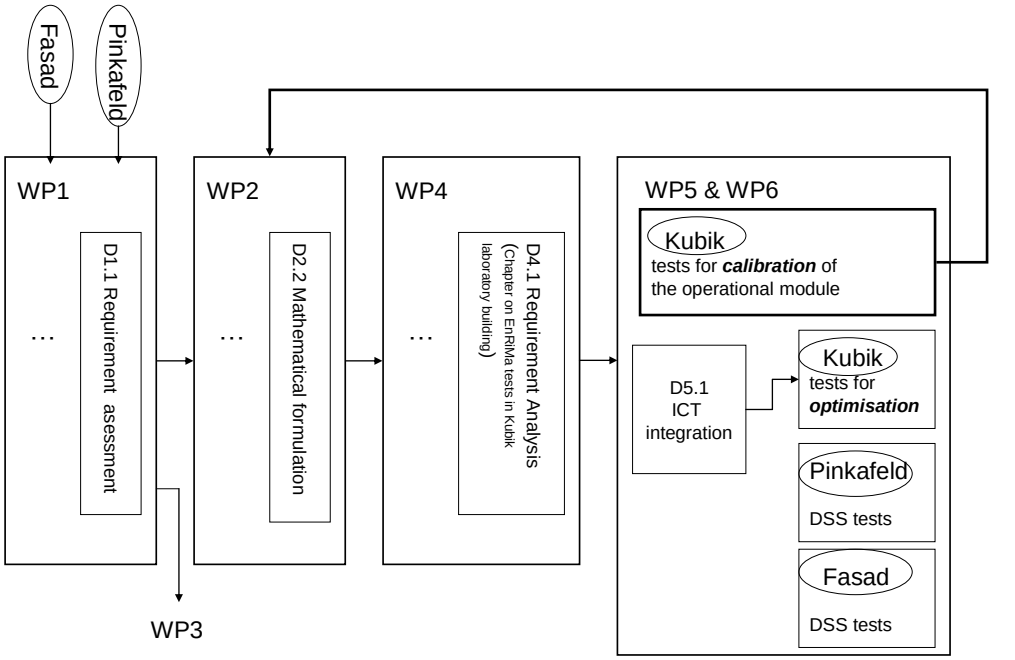

Figure 8-1 Workflow illustrating the role of tests performed at Kubik.

It can be seen that Kubik aim is to assist the development and optimisation of the DSS operational tool which will be particularly developed for Fasad and Pinkafeld sites.

Furthermore, Kubik tests are also required for the optimisation part of the DSS operational module.

8.1.2 Operational model calibration and validation

By building envelope behavior we mean the evolution of the internal space temperature as a function of the building fabric characteristics and the supply air temperature to the space. Internal air temperature is also determined by the building solar gains, internal gains (people, lights and equipment), as well as infiltration and fabric heat gains/losses.

It is important to calibrate the building in terms of these parameters which are inherent to a real building. While supply air temperature and internal air temperature are parameters that the building operator can modify/adjust, heat gains/losses that come from solar, external temperature and internal gains, are not possible to modify, therefore those need to be determined under different scenarios. In Kubik, internal gains from people could be neglected, as this is test facility where no people are found during the tests. However, it is anticipated that a few lights and some equipment would be in the rooms where the test are to be performed.

These behaviors are characterized by equation 11 in D2.2.

In line with the same principles, Kubik HVAC system is able to adapt in order to perform the DSS operational module tests. The HVAC system in Kubik currently supplies a constant amount of air at a constant temperature. The final room temperature is generally achieved by the room fan coil units which add heat or cool to the air by recirculation in order to meet the room set point temperature.

The tests that are required to calibrate the HVAC system's behavior modeled by the equations defined in D2.2 equation 19 are limited to the AHU operation and therefore, the current Kubik operation must be modified in order to calibrate the equations.

Similarly, the heating demand can be calculated in Kubik to test the validity of equation 20 in D2.2.

8.1.3 Tests procedure definition

8.1.3.1 Tests performed at the Kubik facility

With regards to the cell or groups of cells that are available to perform the tests, two floors have been identified as potential areas to carry out the tests. Table 3-27 of D1.1 shows the current Kubik floor layouts.

Preferably the tests shall be carried in room K on the first floor or another one with similar conditions. This area comprises a sufficient volume of air and has a considerable external glazed area, facing South and West.

The air supplied to this space come from the main AHU and passes through a final distribution element named MCU (Measurement Control Units). A MCU consists of a variable air volume fan, an electrical heater, a flow measurement device and a temperature sensor. All of the parameters of the MCU are controllable

8.1.3.2 Parameters in Kubik and data from the information model

A set of tests will be carried out, by continuously monitoring certain parameters. In particular, for each test the data that will be collected in a given time interval (usually 15 minutes or one hour) are the following:

Table 8-1 Summary of tested parameters.

Parameters	**Source**		Nomenclature
External ambient temperature	Meteo forecasts (Euskalmet)		χ
Solar irradiance	Meteo forecast (Euskalmet)		σ
Internal ambient temperature in the space	BMS	Room temperature sensor	Λ
Air flow rate supplied into space	BMS	MCU air flow rate	Ω_{vent}
Constant internal gains in the space: IT equipment and lighting	BMS	Estimated	λ
Supply Air Temperature (SAT) into the space	BMS	MCU temperature sensor	γ

Additionally the following parameters although not critical for the tests, might be useful for the purpose of identifying sources of errors. Therefore it is worth logging the following parameters:

- AHU supply temperature
- External ambient relative humidity
- Internal ambient relative humidity in the space
- Heating and cooling consumption in AHU coils
- AHU set point temperature
- AHU return temperature
- AHU external temperature sensor
- Volume of incoming fresh air
- Volume of extracted air
- AHU Supply fan velocity

After the data gathering exercise the data will need to be refined to ensure its suitability.

8.1.3.3 Defining the tests

The following is a description of the works that intent to assist in the development of the DSS operational module.

Kubik is a modular and flexible laboratory where tests can be carried out simultaneously. However, due to the nature of the tests (where multiple aspects are being considered) it must be ensured that no other experiment will interact with the DSS operational tests. Therefore, the following are the conditions under which the tests are to be carried out.

- AHU unit must be operated as required by EnRiMa team.
- All space fan coil units must be switched off.
- Supply air into space only from AHU, all windows and doors must be closed, in order to minimise uncontrolled air flows.
- Tests will need to be carried out in at least two HVAC modes: heating and cooling periods. It is expected that in November Kubik will be available to carry out the heating mode tests and possibly the cooling mode tests will be performed in May.

The tests that are envisaged at this stage are described below. The convenience of generating more scenarios will be continuously evaluated.

In addition, for each of the tests performed a Sankey diagram will be generated and provided via a Web service along with other graphs that will provide relevant information.

- Scenario 1: Evolution of internal temperature without HVAC system. Aim: Calibration of building fabric heat/losses of equation 11 in D2.2.

Table 8-2 Characteristics of Tests 1.

	Test 1.a	Test 1.b
Mode	Heating	Cooling
HVAC on/off	Off	Off
Duration	Two consecutive days: Saturday and Sunday	
Frequency	Four weekends	

- Scenario 2: Evolution of internal temperature with HVAC system on and fixed supply air temperature into the space. Aim: Calibration of the HVAC system parameters in equation 11 in D2.2 and behaviour of the HVAC system according to equation 19 in D2.2.

Table 8-3 Characteristics of Tests 2.

	Test 2.a	Test 2.b
Mode	Heating	Cooling
HVAC on/off	On	On
Room set point	Constant at 21ºC/23ºC/25ºC	
Supply Air temperature	Fixed and determined by AHU controls according with equation 19 in D2.2.	
Duration	One week each temperature, 3 weeks in total	
Frequency	One in each mode	

- Scenario 3: Evolution of internal temperature with HVAC system on and variable supply air temperature into the space. Aim: Optimisation of HVAC system operation. Note that the extent of the optimisation tests is subject to the progress of WP 2 and 4 at the time of the tests.

  It is anticipated that Test 3.a (November 2012) is likely to be run in an offline mode at best. The optimization tool should be developed within WP3 and WP4 for then. However, it is potentially possible that the optimisation tool will be ready to test online (following the implementation of the specifications defined in D5.1) in May 2013 (Test 3.b).

Table 8-4 Characteristics of Tests 3.

	Test 3.a	Test 3.b
Mode	Heating	Cooling
HVAC on/off	On	On
Room set point	Variable depending on external conditions	
Supply Air temperature	Determined by AHU controls	
Duration of tests	One week	
Frequency of tests	One in each period	

8.2 Tests of the ICT requirement analysis

The previous chapter has defined:

- what is the purpose of the tests to be developed in Kubik,
- the suitability of the building for performing this kind of tests,
- and the need within the EnRiMa project for using this building as a test facility.

This chapter will try to define the borders and interactions between the EnRiMa's DSS (Decision Support System) architecture specification, defined in D5.1, and the existing Energy Management System in Kubik, described in D1.1. Figures 2-8 and Figure 3-1 of D5.1 are the reference ones in what concerns the EnRiMa's DSS architecture specification.

The architecture defined here will be used for the final tests in which the EnRiMa's DSS operational module's full capabilities will be tested. That is the optimization functions and solutions applied to the HVAC system operation. However, it is not necessary to be used for the off-line operational module's calibration tests.

The Figure 8-2 links the existing architectures for the different services provision: the Kubik Energy Management System (BMS), the weather forecast provided by Tecnalia Meteo within the Energy and Environment Division, and the EnRiMa's DSS.

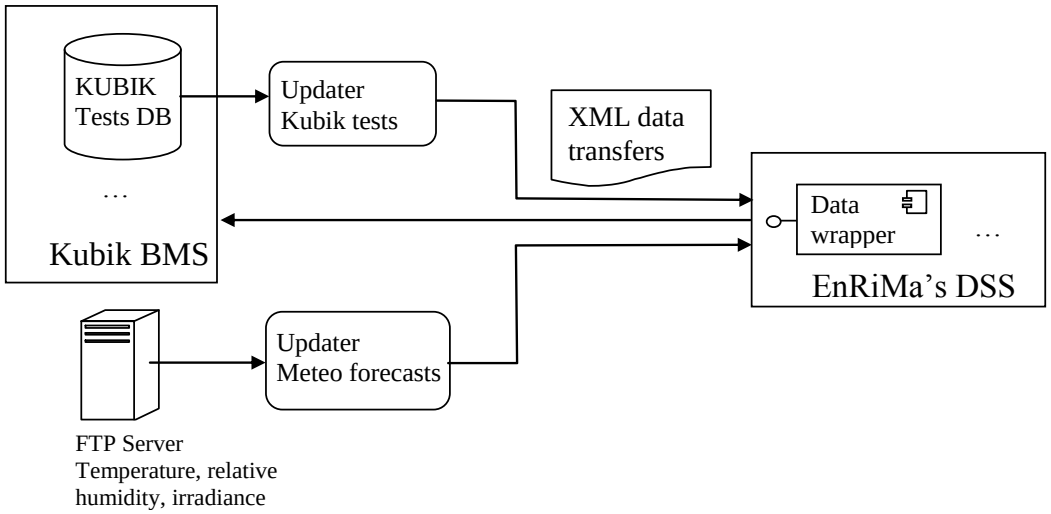

Figure 8-2 Integration of different ICT services involving Kubik.

In the following points the different elements of this integration will be described:

- **EnRiMa's DSS**: Decision Support System being developed in the frame of the EnRiMa project. It is useful for different stakeholders who take decisions at strategic and operational level in what refers to the use of energy and investments in the building.
- **EnRiMa's DSS data wrapper to external data sources**: The data wrapper module is responsible for communicating with external data sources, and converting relevant data into a suitable format for storage by the DSS Kernel. The data wrapper module can be triggered by an FTP file upload from an external source (push), or being set to fetch information from external sources on a regular interval (pull).
- **Kubik BMS**: The different energy subsystems in Kubik are controlled by a Building Energy Management System, described in D1.1. The different control orders to these subsystems can be automatized through XML files by means of the Updater Kubik tests.
- **Kubik Tests Data Base**: In this database the different energy subsystems regulation, control parameters and measurement values obtained in the tests are stored.
- **Updater Kubik Tests through XML files**: This module will be developed by Tecnalia from specifications defined within WP5. This module facilitates:
    - The reading of certain measured values from the existing BMS in Kubik. For example one of these values could be the actual cell's zone temperature value.
    - The transfer of the constructive data or test parameters affecting the operational module.
    - The transfer of the measured values constructive data and/or test parametres to the DSS's data wrapper through XML files.
- **FTP Server dedicated for meteo forecasts**: Tecnalia's Meteo service provides a file containing temperature, relative humidity and sun irradiance forecasted values for the following 24 hours in one hour time intervals via an FTP server.
- **Updater Meteo forecasts through XML files**: To be developed by Tecnalia from specifications defined within WP5. This module facilitates the transfer of the weather forecasted data to the DSS's data wraper through XML files.

Concerning the input and output data to the EnRiMa's DSS and taking into account the information model defined within EnRiMa, figure 2-8 in deliverable D5.1, at least the following set of information would be considered within the tests in EnRiMa:

- Building,
- InternalBuildingP t,
- BuildingTarget,
- EnvironmentP t,
- HVACsystemP,
- EMarketPrice t,k,n
- Pollutionrate k,l,n
- TechControl.

9 Conclusion

The EnRiMa DSS has been developed to support decision-making aimed at improving energy-efficiency of buildings. It will be tested on selected buildings with the aim to be reusable, i.e., easily adaptable to other buildings or facilities. Therefore, a modular structure of the DSS is a key condition to achieve both goals. Any DSS has to meet specific needs of the users for which it is provided. In order to meet these needs efficiently, it is rational to design the DSS architecture in such a way that possibly many components can be reused without substantial modifications; and the needed modifications should be easy to implement. Moreover, since the building ICT infrastructures use diverse hardware and software platforms, interoperability of the DSS components is necessary for efficiency of reusability.

The EnRiMa DSS architecture was designed to meet the above summarized needs; it also supports efficient software development and maintenance through the following key features: modular structure, clearly defined responsibilities of each of the top-level components (UI, DSSE, applications to be integrated with the ICT of each building), as well as WSs used for robust and effective communication between heterogeneous DSS components. The DSSE components can be very easily reused; its only one building-specific DSSE element, namely the SMS, has been designed to be easily adaptable for other buildings. The UI needs to be modified to meet specific requirements of each building, but it can be designed and implemented in such a way that the modifications will be relatively easy. Finally, the applications that need to be integrated with, or interface the building ICT infrastructure will have to be modified to fit the ICT of each building.

In addition to fulfilling the EC request for D4.1 revision, this deliverable contributes to the on-going and forth-coming EnRiMa activities. Most of the D4.1a content has been known to EnRiMa partners from many discussions and the corresponding background notes, results of tests performed at Kubik, as well as from experiments with the Web-services testing-package. However, the request for the D4.1 revision triggered organization of this dispersed information into a consistent document. This will guide the partners in the effective use of possibilities offered by the available methodology and technology (especially the Symbolic Model Specification and the Web-services), as well as by the capabilities of the Kubik facility. The scope of Web-services and data services, although well defined in the DoW, is now specified in more detail, and illustrated by presentation of services that they provide to workflows defined in the use cases developed in the Requirement Analysis (IIASA et al., October 2011).

Acknowledgements

This report has been prepared at a very short notice, and it is therefore composed of sections prepared by partners who lead the corresponding activities, and also take responsibility for the content of corresponding section. We point out extremely short time available for comments by other EnRiMa partners. However, although the time constraints enforced segmentation of work on this deliverable, the underlying work results from collaborative activities started in October 2010.

Contributions to this deliverable are as follows. Eugenio Perea (Tecnalia) has developed the section on the Kubik facility. Sections on the Solver Manager and Scenario Generation Tool have been prepared by Emilio Lopez Cano and Javier Martinez Moguerza (URJC), and by Michal Kaut and Adrian Werner (Sintef), respectively. All other parts of this report have been written by Marek Makowski, Hongtao Ren and Janusz Granat (IIASA), who also compiled the whole document.

Finally, the authors note that parts of contributions the respective partners provided for deliverable D5.1 (SU et al, 2012) have been adapted for this report, and gratefully acknowledge the editorial-type comments provided by Markus Groissboeck (CET).

References

Geoffrion A., An Introduction to Structured Modeling (1987), Management Science, vol 33, no 5, pp 547-588.

HCE, IIASA, SU, UCL, URJC, SINTEF, CET, and Tecnalia (2011). Requirement Assessment. EnRiMa Deliverable D1.1, European Commission FP7 Project Number 260041.

IIASA, SU, UCL, URJC, SINTEF, CET, HCE, and Tecnalia (2011). Requirement Analysis. EnRiMa Deliverable D4.1, European Commission FP7 Project Number 260041.

Makowski M., (2005), A Structured Modeling Technology, European Journal of Operational Research, vol. 166, no 3, pp 615-648.

SU, IIASA, SINTEF, URJC, and CET (2012). Draft Specification for Services and Tools. EnRiMa Deliverable D5.1, European Commission FP7 Project Number 260041.

SINTEF (2012). Scenario Generation Software Tool – Documentation for the Software Tool, part of EnRiMa Deliverable D3.2, European Commission FP7 Project Number 260041.