Chapter 9
A Web-Based Software System for Model Integration in Impact Assessments of Agricultural and Environmental Policies

Jan-Erik Wien, Andrea Emilio Rizzoli, Rob Knapen, Ioannis Athanasiadis, Sander Janssen, Lorenzo Ruinelli, Ferdinando Villa, Mats Svensson, Patrik Wallman, Benny Jonsson, and Martin van Ittersum

Introduction

Policy makers are confronted with complex, socially relevant problems (Van Ittersum et al., 2008). When handling these problems, there is a general tendency to increase participation of citizens and other stakeholders. In this way policy makers

J-.E. Wien, R. Knapen (✉), and S. Janssen
Center for Geo-Information, Alterra Wageningen UR,
P.O. Box 47, 6700 AA, Wageningen, The Netherlands
e-mail: jan-erik.wien@wur.nl; sander.janssen@wur.nl

A.E. Rizzoli and I. Athanasiadis
IDSIA – Istituto Dalle Molle di Studi sull'Intelligenza Artificiale,
Galleria 2, CH-6928, Manno, Switzerland
e-mail: andrea@idsia.ch; ioannis@idsia.ch

R. Knapen
Center for Geo-Information, Alterra Wageningen UR, P.O. Box 47, 6700 AA,
Wageningen, The Netherlands
e-mail: rob.knapen@wur.nl

L. Ruinelli
AntOptima, Via Fusoni 4, 6900, Lugano, Switzerland
e-mail: lorenzo.ruinelli@antoptima.ch

F. Villa
Gund Institute for Ecological Economics, University of Vermont,
617 Main Street, Burlington, Vermont VT, 05405, USA
e-mail: ferdinando.villa@uvm.edu

M. Svensson, P. Wallman, and B. Jonsson
Centre for Sustainability Studies, Lund University, P.O. Box 170, SE-221 00, Lund, Sweden
e-mail: mats.svensson@lucsus.lu.se; patrik.wallman@lucsus.lu.se; benny.jonsson@lucsus.lu.se

M. van Ittersum and S. Janssen
Plant Production Systems Group, Wageningen University, P.O. Box 430, 6700 AK,
Wageningen, The Netherlands
e-mail: martin.vanittersum@wur.nl

F.M. Brouwer and M. van Ittersum (eds.), *Environmental and Agricultural Modelling: Integrated Approaches for Policy Impact Assessment,* DOI 10.1007/978-90-481-3619-3_9,

want to improve the quality of their policies and obtain a broader support for and understanding of the proposed policy measures. In this new governance concept, the policy making process is the product of complex interactions between governmental and non-governmental organizations, each seeking to influence the collectively binding decisions that have consequences for their interest. Policy making is more and more a process of cooperation and participation in which the policy maker becomes a facilitator of the process.

To account for this new governance, policy measures need to be assessed in an integrated context. Rotmans and van Asselt (1996) defined Integrated Assessment (IA) as "an interdisciplinary and participatory process combining, interpreting and communicating knowledge from diverse scientific disciplines to allow a better understanding of complex phenomena". In this interdisciplinary and participatory process, information needs to be accessible in the way that all different types of stakeholders achieve a mutual understanding of the problems, objectives and solutions. But this mutual understanding across disciplines is often hindered by jargon, language, past experiences and presumptions of what constitutes persuasive argument, and different outlooks across disciplines or experts of what makes knowledge or information salient for policy makers or policy assessments (Cash et al. 2003). This mutual understanding is essential in integrated assessment studies in which modelling frameworks are built from multiple models, data sources and indicators, which have to be coupled meaningfully. The software architecture of modelling frameworks should be such that it reflects and enables the mutual understanding of the group of researchers.

This chapter addresses problems and offers some solutions concerning knowledge integration in integrated assessment studies, first by analysing the theoretical perspective and then discussing the key role played by ontologies to support model integration in the context of the SEAMLESS project. Finally, a web-based software architecture for implementing modelling frameworks in integrated assessment studies is presented and we demonstrate its use to improve mutual understanding within the researchers' team. Along the way, the paper details the challenges and the processes used to manage and tame the complexity of a large European integrated project (e.g. SEAMLESS) and the use of an ontology to support (linking of) projects, models, indicators and raw data. This chapter concludes this paper by describing the major functionalities offered by SEAMLESS-IF and its expected future developments.

Challenges in Integrated Modelling

Interoperability from a Theoretical Perspective

Integrated modelling requires interoperability, which is the ability of two or more systems or components to exchange information and to use the information that has been exchanged (IEE 1990). Sølvberg (1998) distinguished between syntactic, structural and semantic interoperability. Syntactic interoperability is defined

as the ability of two or more systems/components to exchange and share information by marking up data in a similar fashion (e.g. using the extensible markup language, XML). Structural interoperability means that the systems/components share semantic schemas (data models) that enable them to exchange and structure information (e.g. using the resource description framework, RDF). Semantic interoperability is the ability of systems/components to share and understand information at the level of formally defined and mutually accepted domain concepts. Semantic interoperability requires the correct interpretation and mutual understanding of all exchanged information. In order to obtain mutual understanding of exchanged information, the actors have to share a model of what the data stand for. Semantic interoperability is about how to achieve such mutual understanding (Sølvberg 1998).

One of the earliest theories dealing with understanding and remedies for misunderstanding is Richards's "Meaning of Meaning" theory (Ogden and Richards 1923). Instead of focusing on the information that is communicated, Richards wanted to study the meaning of the words. He felt that understanding is the main goal of communication and communication problems result from misunderstanding. One of the ideas behind the Meaning of Meaning Theory is "The Proper Meaning Superstition" (Ogden and Richards 1923). This is the belief that every word has an exact meaning. Richards says that the Proper Meaning Superstition is false because words mean different things to different people in different situations. This misunderstanding causes problems when two people believe they are talking about the same thing, but actually talk about different things.

Another concept that Richards uses is the idea of signs and symbols in communication. Words are examples of such symbols. Symbols have no natural connection with the things they describe (Griffen 1997). Words are symbols of something because they have been given meaning. But very often words mean one thing in a certain context and mean another thing in a different context. This is why it is so important to study the context to get a better understanding of the meaning.

To come to this better understanding, Richards invented the Semantic Triangle (Fig. 9.1). This triangle shows the relationship between symbols and their referent.

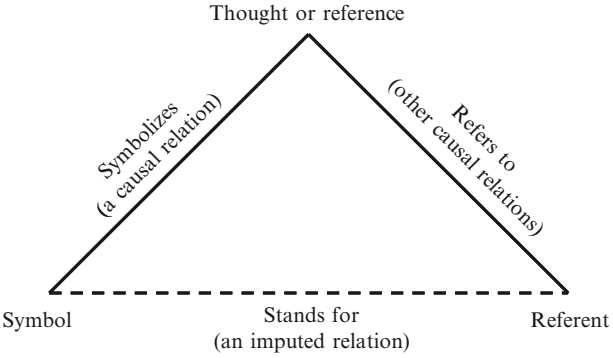

Fig. 9.1 Semantic triangle by Richards (Ogden and Richards 1923)

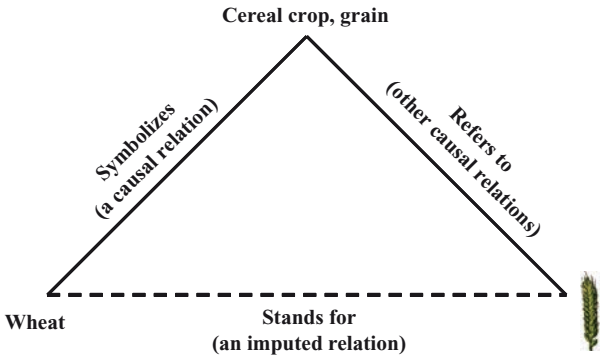

Fig. 9.2 Semantic triangle for wheat

One part of the triangle is the symbol, or the word. Another part of the triangle is the thought or reference. These are the words that one would use to describe the referent. The referent, the last part, is the thing that one would picture in his mind.

An example in integrated agricultural modelling (Fig. 9.2) would be "wheat". The words to describe the referent can be for example "cereal crop, grain, flour". The referent, the last part, is the thing that one would picture in his mind.

Understanding that people mean different things when they say the same word is an important finding that is very relevant for the process of integrated modelling, more so when it comes to associate the proper meaning with variables to be exchanged between two models (Wien et al. 2007).

Interoperability in Integrated Modelling

If the integrated model, composed by a number of different models pertaining to domains as diverse as economy, agronomy, social sciences, ecology etc., is developed by a single scientist, or even by a closely connected group of scientists, the potential ambiguities and misunderstandings can be solved by internal discussions and interactions. The problem is that, given the scale and the ambition of such integrated models, it is unrealistic to even think that a single group can encompass all the required knowledge to develop such a model. Therefore, different models are developed by different groups, and as a consequence the need arises for structured and organized ways to solve ambiguities and misunderstandings regarding the meaning of the variables to be shared and exchanged by the models.

In such a complex integration task, different types of misunderstandings around the meaning of concepts can occur:

1. As the same concept might be used for different meanings, assuming unspecified information, for example "crop yield" in a model as a single value and "crop yield" in a database as a series of values over space and time;
2. As different concepts might be used, which have the same meaning, for example "derivative" and "rate";
3. As concepts might be used with an ambiguous meaning, for example the word "scenario" (Schoemaker 1993);
4. As relationships between concepts might be understood in a different way, for example the spatial relationship of inclusion of farm within an "agri-environmental zone", and the relationship between the same farm and an administrative area.

The challenge in integrated modelling is the conceptual integration. To achieve this, we need explicit semantics and a shared conceptualization. For this we need to tackle the different perceptions and interpretations of people involved. Different modelling approaches, different formalisms and last but certainly not least, the different integration requirements and ambitions need to be taken into account.

To enable semantic interoperability in integrated modelling, the problem of semantic conflicts or semantic heterogeneity needs to be solved.

The Need for a Common Ontology

One of the many definitions of ontology is from Neches et al. (1991) "an ontology defines the basic terms and relations comprising the vocabulary of a topic area as well as the rules for combining terms and relations to define extensions to the vocabulary". According to Gruber (1993), an ontology is "an explicit and formal specification of a conceptualization". A conceptualization is explained as an abstract model of a phenomenon, identifying the relevant attributes. A conceptualization is an abstract, simplified view of the world that we wish to represent. Every knowledge base or knowledge-based system is committed to some conceptualization (Gruber 1995). Formal specification refers to the fact that the language semantics are machine readable, and follow a mathematical framework for logical reasoning. Often this is done by use of W3C OWL (Ontology Web Language, Patel-Schneider et al. 2004). A formal specification helps to communicate the definition of terms in a context independent way and formal language semantics allows some automated consistency checks.

An ontology helps to formalize the knowledge captured in and/or between models, in order to subsequently facilitate model development, testing and documentation (Scholten et al. 2007) and increase model re-usability and exchangeability (Rizzoli et al. 2008; Villa et al. 2009). Besides this it separates knowledge captured in the model from the actual implementation in a modelling language or in software e.g. Java, FORTRAN, Matlab, STATA, etc. (Gruber 1993; Villa et al. 2006) or from the data in a database (Zander and Kächele 1999).

The development of a common ontology by a group of researchers is a complex, challenging and time-consuming task (Farquhar et al. 1995; Gruber 1993), and it still remains a scientific challenge. To achieve ontological commitment, i.e. the agreement by multiple parties to adhere to a common ontology, when these parties do not have the same experiences and theories (Holsapple and Joshi 2002), a collaborative approach is needed (Pennington et al. 2007). Other approaches for ontology development are the inspirational approach, the inductive approach, the deductive approach and the synthetic approach (Holsapple and Joshi 2002). A collaborative approach has the advantages that researchers from different disciplines are diverse in their contributions, which avoids blind spots and which has more chances of getting a wide acceptance (Holsapple and Joshi 2002).

The Role of Ontologies in SEAMLESS

The Structure of SEAMLESS-IF Framework

The SEAMLESS consortium develops a computerized and integrated framework (SEAMLESS-IF) to assess the impacts on environmental and economic sustainability of a wide range of policies and technological improvements across a number of scales (Van Ittersum et al. 2008). In the SEAMLESS-IF approach different type of models and indicators are linked into a type of scientific workflows (van der Aalst and van der Hee 2002) named *model chains*, where each model uses the outputs of another model as its inputs and ultimately indicators are calculated. With respect to the models (Fig. 9.3), macro-level economic partial equilibrium models (GTAP and CAPRI; Britz et al. 2007) are linked to micro-level farm optimization models (FSSIM-MP and FSSIM-AM; see chapter 5 of this volume) and field crop growth models (APES; cf. Van Ittersum and Donatelli 2003 - see Chapter 4 of this Volume), using micro-macro up scaling methods (EXPAMOD; Bezlepkina et al. 2007). The macro-level economic partial equilibrium models simulate markets for agricultural commodities and trade between the European Union and other world trading blocks and the micro-level farm optimization problems allocate the farm area to different agricultural activities based on farmer objectives and available farm resources. Examples of agricultural activities are growing a wheat–sugar beet rotation with intensive management or keeping dairy cattle for the production of milk. The agricultural activities for cropping systems can be evaluated by the field crop growth models on their yield and environmental effects (e.g. nitrate leaching, pesticide leaching, soil erosion). Finally the micro-macro up-scaling methods extrapolate on the basis of the micro-level behaviour of the optimization model FSSIM-MP price-elasticities that are an input to the macro level partial equilibrium model CAPRI. These models provide, through their outputs, the basis for the calculation of indicators of interest to the user. Each of these models are derived from different disciplines, operate on different time and spatial scales, are programmed in different programming languages and have a different implementation of scenarios.

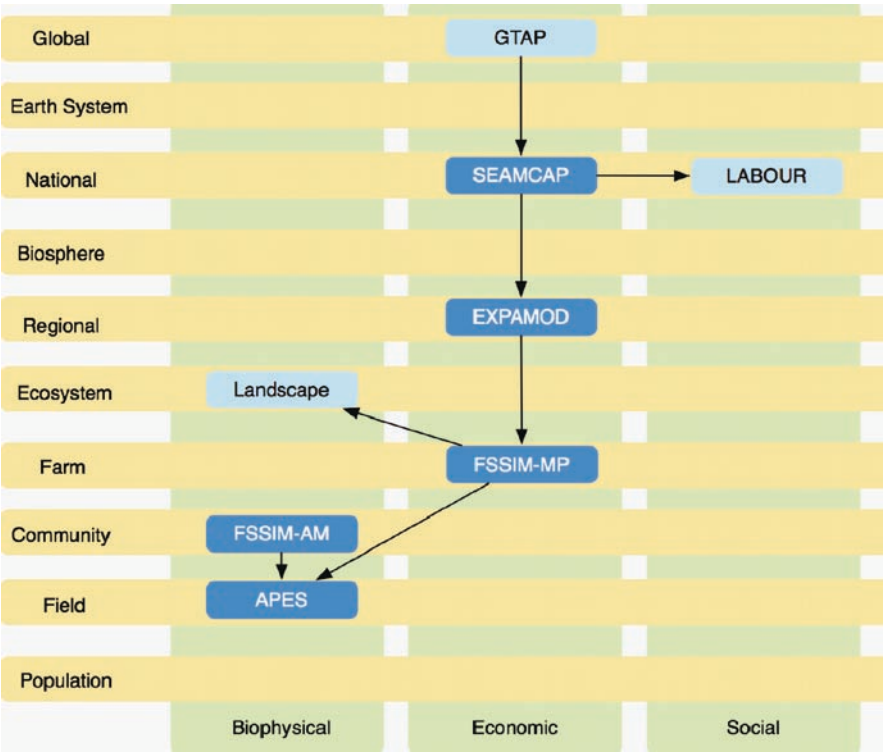

Fig. 9.3 Overview of SEAMLESS-IF models. The arrows indicate the dependencies among models in the chain. The components in lighter colour are not presently integrated in SEAMLESS-IF

Knowledge Management Technologies in SEAMLESS

Knowledge management technologies are employed, such as semantic modelling and ontologies for specifying data, models, projects and their relationships. All the commonly shared data types in SEAMLESS were declared in ontology (starting from projects, up to the model exchange items). This is a very important shift in everyday practice that SEAMLESS has achieved: modellers specify the data requirements of their models in a higher level, i.e. that of an ontology. This ontology is then automatically transformed into a relational database model, to which "data collecting" activities need to comply with. Data may originate from third-party, pre-existing databases or may have been directly collected (i.e. through surveys). Either of the two is the case, data collection activities need to facilitate the filling-up of the generated database, which is fit to the model data requirements.

The process used in the SEAMLESS project for creating a common ontology for models, indicators and raw data was based on a participatory and collaborative approach. A dedicated group of scientists was formed with participants from all parts of the project. This cross-disciplinary group, named DOT.force (after Data and Ontology Task force), aimed to develop a common ontology that represents a shared conceptualization between the different databases, models and indicators, involved in the SEAMLESS-IF process. The DOT.force listed among its members, knowledge engineers and domain experts. Knowledge engineers have the technical skills and relevant experience in ontology design and conceptual modelling. The domain experts hold knowledge about a specific domain, like a certain model or dataset, a set of indicators, a particular case study or scenario, or even the SEAMLESS-IF process. Domain experts interacted with knowledge engineers for specifying the conceptualizations involved in their niches, and ultimately facilitated the development of specific parts of the ontology (Janssen et al. 2009).

The knowledge engineers in the DOT.force worked on a number of actions that lead to a complete and consistent ontology. These actions included:

1. Integrating the different database schemas into a single SEAMLESS database schema;
2. Clarifying the model interface data structures (domain models), while adding relevant metadata;
3. Associating indicators with model result sets or other static data collections;
4. Supplementing the ontology with meta-data on the concepts it holds, like textual descriptions, data sources, other documentation references, units and value ranges;
5. Developing an upper ontology to cover concepts relevant with the SEAMLESS-IF process and execution.

Different methods were used to construct the ontology for the different actions. For actions one and two on databases and models, dedicated meetings were organized to develop the ontology, while for action three on indicators a proposal was made by the knowledge engineers, which was then evaluated by relevant domain experts. Action four on metadata was carried out independently by domain experts, once agreement on the common ontology was reached between domain members and knowledge engineers. For action five on project and scenario definition an iterative process was used to develop a document, which was later synchronized with the project ontology after each iteration.

The database design is directly mandated by the object structure of the entities and properties described in the ontology and the data can be stored in the database based on an adjunct persistence XML (eXtensible Markup Language) document provided along with each class implementing the ontology structure.

Other working groups within the project have been populating the database following the schema generated by the ontology that the DOT.force concluded to. To explain more details about the knowledge management technology, we need first to give an overview of the SEAMLESS-IF architecture.

An Architecture for Knowledge Integration in Integrated Assessment Studies

The SEAMLESS-IF Architecture

A keystone of the SEAMLESS project is the software architecture enabling and supporting the development of integrated assessment studies. The global architecture of SEAMLESS-IF is typical for a layered (multi-tier) web based application. The diagram in Fig. 9.4 illustrates this architecture.

The principle of dividing a software system into layers reaches back to the early days of computer science (Dahl et al. 1968). Originally the idea was used for the design of operating systems but it applies equally well to other types of software. In the layered software architecture the system's functionality is usually separated in a persistence layer for storage of domain object state, a domain layer for the domain model and domain logic, a services layer that controls transactions and contains the business logic, an application layer for use-case workflows, syntactic validation and interaction with the services layer and finally a presentation layer for the user interfaces. As a general rule each layer has only dependencies on those below it, not above, limiting the effect of changes and increasing maintainability. In a way each layer acts as a client to the tier below and as a server to the tier above. All layers can be located on a single computer, or they can be divided amongst a number of systems. The latter is the case for SEAMLESS-IF.

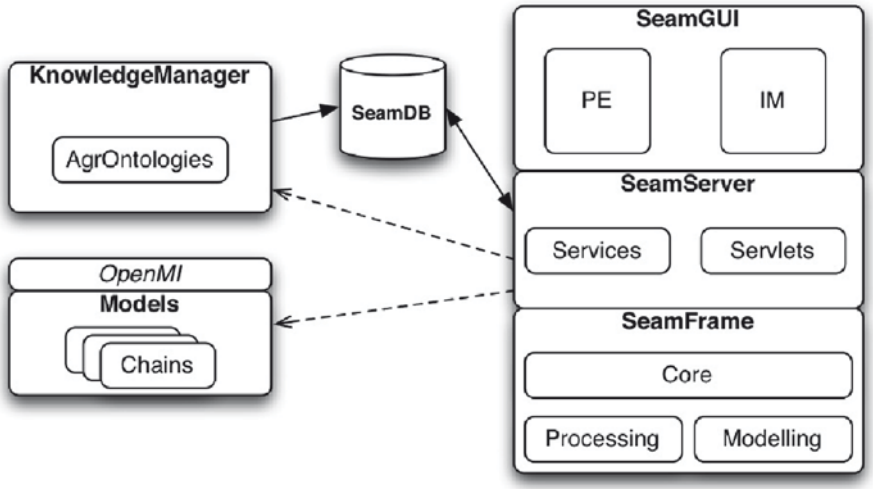

Fig. 9.4 The SEAMLESS-IF architecture on the right is composed by three layers (SeamFrame, the core framework; SeamServer, the server side services; SeamGUI the client-based graphical user interface) and by external modules such as OpenMI compliant models, a set of ontologies managed by the Knowledge Manager and the SEAMLESS database

The end user interacts with the system by means of *SeamGUI*, a browser based application providing a Graphical User Interface (GUI) running as client of the web (server) application. The client-server architecture of SEAMLESS-IF allows for future clients to be developed and linked to the existing server, in order to cater for specific needs of different user groups. The currently available SeamGUI client includes two task specific user interfaces:

- *PE GUI*, the policy expert GUI: through this interface an expert can evaluate the impact of alternative agricultural policies from the different aspects of sustainability;
- *IM GUI*, the integrative modeller GUI: the module that guides the end-user to manage projects and request the execution of model chains, in order to produce results to be later used by policy experts.

SeamGUI requires a user to log in to the system first, and then decides based on the user permissions which task user interface to show. It then interacts with the software services provided by the *SeamServer*, the web-based application framework built on the core classes provided by *SeamFrame*. The latter is a modelling framework purposely designed to develop integrated assessment tools, thanks to:

- The *Modelling Environment*, which is a programming framework that offers a series of facilities to encapsulate and wrap existing models for execution by the processing environment. It allows to deliver model components wrapped by a SeamFrame specific interface, compliant with the Open Modelling Interface (OpenMI) standard (www.openmi.org) (Gijsbers and Gregersen 2005, Gijsbers et al. 2006), so that it can be executed by the Processing Environment (see below). Future versions of the Modelling Environment could incorporate interactive modelling facilities.
- The *Processing Environment* is both a programming framework and a software application that retrieves requests for the execution of chains of model-components from a queue. This queue is represented by a table in the database, and experiment processing requests can be entered through the SeamGUI. The Processing Environment enables model composition and execution. The actual exchange of data among the models in a chain is based on the OpenMI that provides a standardized interface to define, describe and transfer data between software components that run sequentially. This choice was made based on technical and functional requirements and the possibility offered to re-use legacy models. The use of the central database to store the queue allows for multiple instances of the Processing Environment to run in parallel.

External to the layered architecture, but fundamental for SEAMLESS-IF, is the *Knowledge Manager*, that provides access to and manages persistency of data in the databases. SeamFrame uses the domain model and classes generated for it from the SEAMLESS ontology by the Knowledge Manager. These classes become part of the Modelling Environment. Through a Hibernate-based object relational mapping (www.hibernate.org) the domain model is stored in databases.

The Role of Software Engineering in the Design of Integrated Assessment Tools

Integrated assessment systems tend to be studied and built as part of (large) research projects where the approach is mostly bottom-up: starting from simple models building a larger and more complex model, accounting for the interactions among the simpler processes (Knapen et al. 2007). This emphasizes the linking of the models and perhaps the development or re-using of a framework to support it. Possibilities of the models and the created links between them, and the framework's flexibility are then put behind a most of the time rather technical looking user interface.

The same system could also be thought of from a top-down perspective: start from the larger problem, and try to detail its subcomponents. This is more of a User Centred Design (UCD) (Raskin 2000) approach where usability and business requirements drive the features and technical development. Tell tailing would be that the user interface would be conceptualised first, and its usefulness studied with the intended customers or target customer groups. The bottom-up view serves mostly the modellers and framework builders, and the top-down all the other customers that most likely pay (in some way or another) for the development of the system and only care to a limited level about the intricate internal machinery of connected interdisciplinary models. To them, in essence it is a data intensive software system that has to provide usable information at the right moment, timely for the decision taking process. Data intensive software systems are well known to software engineering, for example consider the large banking and insurance systems. These are commonly called (Business) Information Systems, or Enterprise Applications. If we regard integrated modelling systems from this perspective, would it make sense to apply to them some of the same software design rules – or software architecture, as used for other enterprise applications? The answer is logically positive, and we have thus adopted advanced software development methodologies, especially relying on *design patterns* to assist our design process, as they provide a mechanism for providing software design advice in a reference format (Gamma et al. 1994).

We have been inspired by design patterns in some of our key design choices:

- The *layered software architecture*, an architectural design pattern (Fowler 2007), with its multi-tier levels, helped us to organise the complexity of software, by clearly delegating different roles to components in the different layers.
- *Data Transfer Objects* (DTOs) helped us to optimise remote method calls. A data transfer object is an object that holds all the data required for a call to a remote interface. This pattern has proven itself very useful as the amount of data to be transferred between the client and the server can become rather sizeable in integrated assessment studies, and optimisation in this area brought sensible improvements in the tool usability.
- The *Service Layer* and the *Data Access Object design pattern* were useful in the definition of SeamServer boundary and its set of available operations from the perspective of interfacing client layers. It encapsulates the business logic, controlling transactions and coordinating responses in the implementation of its operations.

– The *Message Service* pattern was fundamental in the integration and the exchange of information between the client and the server applications. It provides a publish/subscribe infrastructure that allows client(s) and the server the exchange messages in real time. There are two key components: a message service running in the application server and a client-side API (Application Programming Interface, a set of routines, data structures, object classes and/or protocols provided by libraries and/or operating system services in order to support the building of applications). The message service manages a set of destinations and handles the asynchronous messaging to them. Unlike synchronous messaging this does not rely on direct connections, the sender of the message does not have to wait for a response from the recipient because it can rely on the middleware to ensure delivery of the request and eventually the response. Representational State Transfer (REST; the architectural style of e.g. HTTP) and web services are examples of asynchronous messaging and a good strategy for integration of enterprise applications. They promote a loosely coupled system of components and also encourages design of components with high cohesion (local processing) and low adhesion (remote work).
– The *CRUD* pattern (create, retrieve, update, delete) has been placed at the base of our data persistence strategy. Almost all applications include some form of persistence storage and have to perform CRUD operations on it (Kilov 1990).

SEAMLESS Server Technologies

As described, SEAMLESS-IF is based on a layered, client-server architecture. The processing environment facilities, in particular the model chain executor, are deployed on the SEAMLESS server. The current software stack for the server configuration consists of a web application server (Tomcat[1]), data storage (PostgreSQL[2]), a data access layer (Hibernate) and the SeamServer software (Fig. 9.5).

The ontology is used to maintain the consistency across the different data perspectives: it is used to generate the database structure, object-relational mapping files for Hibernate and Java Beans to access the data.

The business logic, where the preparation and management of experiments to be run by the processing environment happens, is based on the Spring Framework,[3] a Java[4] based solution delivering a full-stack Java/JEE application framework.

[1] http://tomcat.apache.org/

[2] http://www.postgresql.org/

[3] http://www.springsource.org/

[4] http://java.sun.com/

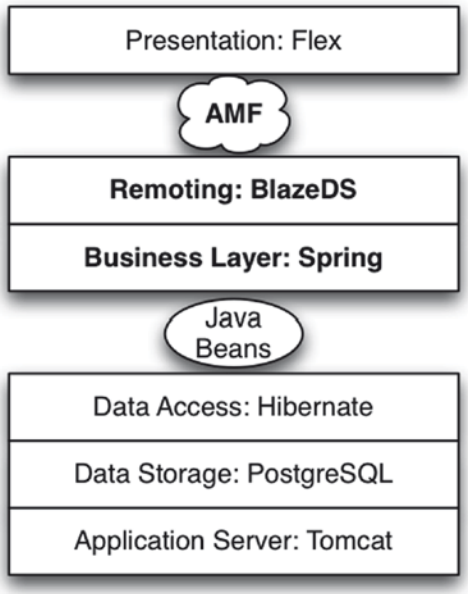

Fig. 9.5 The server technologies adopted for the SEAMLESS server. At the lowest level, we have the Tomcat application server, and the PostgreSQL database. Access to data is made possible through Java Beans using Hibernate ORM (Object Relational Mapping). The business logic is implemented using the Spring framework and remoting based on BlazeDS. The end-user GUI is implemented in Flex, it runs on the client's web-browser (using the Flash Player) and dialogues with the remoting level

Finally, the remoting technology we adopted is BlazeDS, a server-based Java remoting and web messaging technology that allows the connection to Adobe® Flex® and Adobe AIR™ applications for delivering rich Internet application (RIA). Some of the advantages of this approach are:

- Maintainability: less custom source code to maintain, instead leveraging of popular open source frameworks;
- Stability and security: replacing custom coding with well-known and tested open source frameworks;
- Performance: replacing XML with the binary AMF (Action Message Format). Tests shows that this is at least four times faster;
- Flexibility: use of well-known design patterns to enhance the flexibility, such as dependency injection.[5]

[5] http://martinfowler.com/articles/injection.html#InversionOfControl

SEAMLESS Client Technologies

The client-server model is a popular design for large and complex applications, not only for mainframe systems but also for remote systems that provide services over the Internet. Think of e-mail clients using mail storage servers or applications running inside a web browser. Confusingly client is used for software as well as hardware. In general clients are classified as fat clients, thin clients, or hybrid clients (Table 9.1).

A fat client (also known as a thick client or rich client) is a client that performs the bulk of any data processing operations itself, and does not necessarily rely on the server. A thin client is a minimal sort of client, its functionality being limited to the presentation layer. Thin clients use the resources of the host computer. A thin client's job is generally just to graphically display pictures provided by an application server, which performs the bulk of any required data processing. A hybrid client is a mixture of the above two client models. Similar to a fat client, it processes locally, but relies on the server for storage data. These are also known as rich clients and implement both presentation and application layers. In designing a multi-tier architecture, there is a decision to be made as to which parts of the task should be done on the client, and which on the server. This decision can crucially affect the cost of clients and servers, the robustness and security of the application as a whole, and the flexibility of the design for later modification, porting and reuse.

The SeamServer server application can be accessed by SEAMLESS clients. Basically we can consider two types of clients: system-to-system and human-to-system.

A human-to-system client is a graphical user interface that allows using SeamServer functionalities, e.g. to define a project or to visualize results. An initial client (SeamGUI) is developed as part of SEAMLESS-IF. SeamGUI is a hybrid client, it has no local storage but does provide local processing. For example in the case of its SeamPRES component that retrieves calculation result data from SeamServer and processes it locally to create visualisations (tables, charts, map, and so on). A hybrid client currently is mostly referred to as a RIA, software applications running on the user's computer that rely on a server for core functionality but also have some own logic for an enhanced user experience.

A system-to-system client is a programming interface that exposes the SeamServer functionality. In this case, there is no user interaction, rather there are just computers exchanging information (actually the SeamServer application server provides services that the second computer is consuming). The chosen SEAMLESS-IF architecture, using open standards whenever possible, is capable of supporting system-to-system clients. However, this is not a priority within our work, and we do not envision deploying such clients during the duration of the project.

Table 9.1 Types of clients

	Local storage	Local processing
Fat client	Yes	Yes
Hybrid client	No	Yes
Thin client	No	No

For building SeamGUI and SeamPRES, the two flagship client-applications of SEAMLESS-IF, Adobe Flex[6] has been selected as the most usable technology. It offers declarative user interface programming with a rich library of professional and functional components. Code is compiled and runs in a Flash virtual machine (inside a web browser). Protocols for data exchange (pull, server push and data binding) with a server are included. Since a Flex application basically is deployed as a Flash application running within a web browser it is easy to create GUI mock-ups and distribute them to get feedback. This process helped a lot in clarifying the end-user requirements. End-users could easily access the client application and provide with their feedback on design and usability issues.

Finally, it is worth mentioning that web based clients use extensively the Cascading Style Sheets (CSS) standard whenever possible to allow future synchronization (or changes) of look and feel.

The SEAMLESS Knowledge Manager

The SEAMLESS Knowledge Manager has been developed as part of the Thinklab platform, as an open source project[7] for accessing and managing ontologies. It is built upon the Protege-OWL libraries, and has been designed using the Java Plug-in Framework approach (Villa 2005).

The SEAMLESS Domain Manager, a plug-in part of the Knowledge Manager, is a tool for delivering Java objects that are used by models and tools for exchanging information. The SEAMLESS Domain Manager uses Thinklab core infrastructure for accessing ontologies and following the domain class principle, it may:

1. Generate domain classes from ontology;
2. Access domain objects at runtime (by using EJB/Hibernate).

The Domain Manager plug-in of the Knowledge Manager (Villa 2005; Rizzoli et al. 2007a, b; Athanasiadis and Janssen 2008) provides in this way the service for persistently storing generated classes using the EJB/Hibernate technology (www.hibernate.org). Through Hibernate a relational database may store (and retrieve) populated Java objects. In this sense, the Domain Manager is the 'knowledge processing' component of SeamFrame, providing access to and modification of SEAMLESS data, through the ontology-specified interfaces (Fig. 9.6).

Fetching data from the Knowledge Base actually means to retrieve data from a relational database (the SEAMLESS database), using information provided by Hibernate.[8]

[6] http://www.adobe.com/products/flex/

[7] http://www.integratedmodelling.org. Terms of use are defined by the General Public Licence (http://www.gnu.org/gpl).

[8] http://www.hibernate.org/

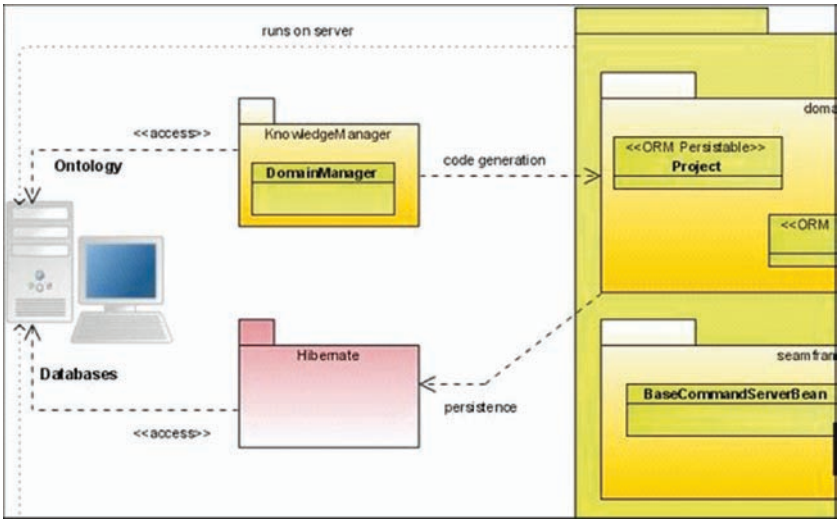

Fig. 9.6 The Domain Manager component. The Domain Manager is used to automatically generate Java Beans which, using the Hibernate platform, can persist objects used to exchange data among models and the database

The Knowledge Manager only deals with metadata (i.e. information on data structure). Data are delivered by the SEAMLESS database and are accessible by using the generated Java Beans through Hibernate.

In the generated source code, Java annotations are used for linking Java Beans with the ontology, like in this example:

```
@ConceptURI("http://ontologies.seamless-ip.org/crop.owl#Crop")
public class Crop implements Serializable
{
...
…@PropertyURI("http://ontologies.seamless-ip.org/crop.owl#hasCropSoil
Requirements")
…public CropSoilRequirements getCropSoilRequirements(){
...
  }
}
```

Since the annotations are accessible at runtime using Java reflection the ontology information can be used in the future for reasoning, e.g. to validate whether an output of a model can be used as an input for another model.

In all practicality it is a fully automatically generated persistence layer for the SEAMLESS system. Higher layers of the system, and the models (or model wrappers) themselves, use it to retrieve, work with and store instances of the concepts as defined in the ontology.

OpenMI and Model Linking

Although the SeamFrame application server operates in Java environment, the model components can be implemented using other languages as long as they can be integrated. This requirement has been translated into the fact that model components should be OpenMI[9] compliant, by implementing the OpenMI interfaces and allow linking to other components (for data exchange). The model component can take care of this by itself or a wrapper or bridge can be programmed.

The Open Modelling Interface and Environment (www.openmi.org) has been developed from the need to answer questions related to integrated hydrological management of catchments within the EU fifth framework program project HarmonIT (www.harmonit.org). OpenMI provides a standardized interface for data exchange between software components that run sequentially, based on a *pipes and filters architecture* (Gregersen et al. 2007). Since the release of the OpenMI in early 2006 the environmental domain adopted the OpenMI within several European projects. This introduced new requirements, which were implemented, resulting in a new OpenMI version (Verweij et al. 2007). Examples of these projects are:

– SEAMLESS – assess agricultural and agro-environmental policies;
– SENSOR – assess sustainability impacts of land use related policies;
– NitroEurope – assess the effects of reactive nitrogen in the environment;
– EFORWOOD – assess sustainability impacts of European forest wood chains.

These projects have a strong integrated character: environmental, social and economic dimensions are taken into account to perform an ex-ante integrated assessment. An IA can be defined as an interdisciplinary process of combining, interpreting and communicating knowledge from diverse scientific disciplines in such a way that the whole cause–effect chain of a problem can be evaluated from a synoptic perspective with two characteristics:

- It should have added value compared to single disciplinary assessment.
- It should provide useful information to decision makers (Rotmans and Dowlatabadi 1998).

OpenMI provides a standardized interface to define, describe and transfer (numerical) data between software components. The data definition concerns what the data is about (*quantity*) and where (*element set*) and when (*time*) it applies. Each component (*LinkableComponent*) has a meta data description of its exchangeable data in terms of a *quantity* and an *element set*. Each unique exchangeable *quantity* is registered and published in a so-called *ExchangeItem*. Connections between *ExchangeItems* of *LinkableComponents* are defined by a *Link* and exist as a separate entity (Fig. 9.7). For more information see http://www.openmi.org.

OpenMI is a pull-based system (a pipes and filters architecture). Figure 9.8 shows how the chain of *LinkableComponents* is triggered (step one) by a successive

[9] http://www.openmi.org/

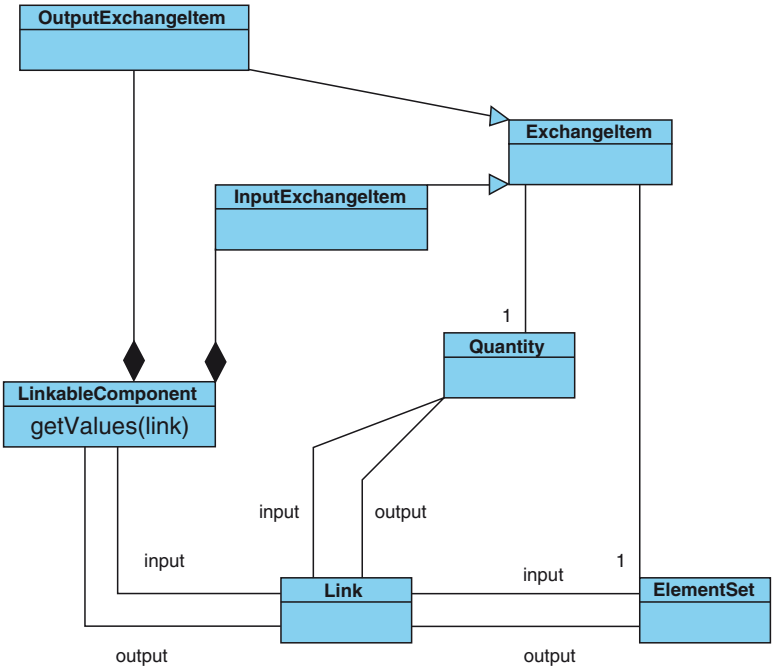

Fig. 9.7 Simplified class diagram of LinkableComponents and Links

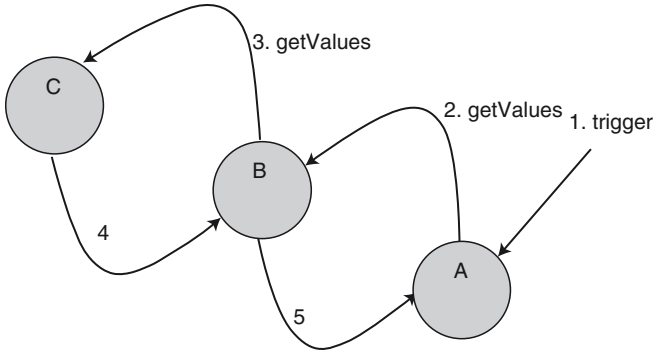

Fig. 9.8 Call chain of LinkableComponents. A, B and C are linkable components

call of the *LinkableComponent* method *getValues()* (step two and step three) after which values are returned to the original caller (step four and step five).

The number of *links* between two models can vary from one up to hundreds. A *link* describes one semantic connection concerning what variable is exchanged between the models and where the exchange takes place (Gregersen et al. 2007). To get any number of variables from a model, it needs to be called for each of these

variables. It is up to the internal intelligence of the model to avoid unnecessary (re) calculations (e.g. by using some caching system).

Allowing a *link* to concern multiple variables simplifies the use of that *link* within a call for variable values. Links could also be given more functionality, but there should stay a clear distinction between functionality of a *link* versus that of a *LinkableComponent*.

Using SEAMLESS-IF for Integrated Assessment Studies

In this section we describe how SEAMLESS-IF can be used in an integrated assessment study. First we focus on the pre-modelling phase and the definition of the boundary conditions of the problem. Then, we enter the modelling phase, where the appropriate model chains are selected, their parameters are defined and sets of simulations experiments are executed. Finally we describe the post-modelling stage, where results are analysed and interactively examined by the end user, an activity that may lead to a redefinition of the pre-modelling or modelling stages. The process is iterated until the end user has gained a sufficient insight on the problem and she has obtained a satisfactory result in the analysis process.

Starting a Project in SEAMLESS-IF (Pre-modelling Phase)

Through SeamGUI, the user may build up a project that specifies an integrated assessment exercise. A *project* (Fig. 9.9) is characterized by the definition of the problem it tries to solve or study, and it incorporates at least one experiment configuration, that is, the configuration of the models to be executed during the analysis. An *experiment*, in turn, is associated with a single model chain and is parameterized by the specification of an outlook, a context and a policy option. Also the *indicators* are associated with a problem; they are used for quantifying the analysis results. Through a single project, there are several alternatives that can be investigated. Based on the results of the computation, the calculated indicators become available in the framework and can be used for visualizing results in SeamPRES, the visualization client. The project definition is a result of the DOT. Force[10] team (Janssen et al. 2007), and has been specified as an ontology in OWL.

Through the SeamGUI client application, the end user may configure a SEAMLESS-IF project, by specifying the narrative descriptions of problem and experiments, context, outlook and policy options, but also, by selecting indicators applicable for the exercise at hand. Figure 9.10 presents the screenshot of the project definition panel.

[10]Seamless Data-Ontology Task Force.

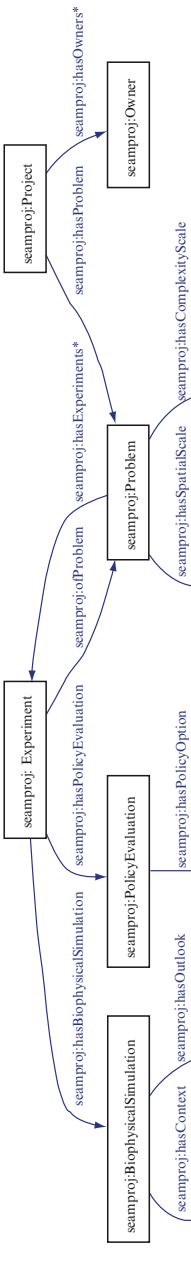

Fig. 9.9 Partial view of the Project Ontology

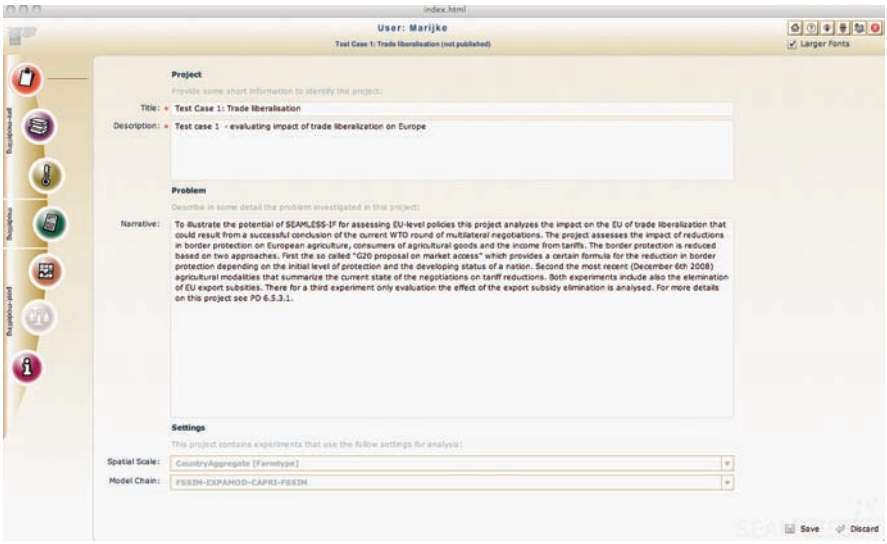

Fig. 9.10 Screenshot of the project definition panels in SeamGUI

Narrative Experiments

Another important aspect in the pre-modelling phase is the narrative specification of the experiments. The narrative specification is required, so that the narrative of the policy expert can be translated to the integrative modeller (and the specialists of each of the model component, i.e. the agricultural modellers), who will materialize them in turn, as project parameters through the Experiment configuration in the modelling phase. There is a one-to-one relationship between the narrative and the configuration: each context, outlook and policy option has a narrative description that will be accessible to the integrative modeller while detailing the configuration.

Indicator Selector and Fact Sheet Viewer

The pre-modelling phase concludes with a choice of indicators for the project. The indicator manager allows the selection of indicators based on the Goal Oriented Framework or directly from the library of available indicators. The aim of using an indicator framework in selecting indicators for an impact assessment is to assist the user in selecting a balanced set of indicators that can help to get a clear picture of how and in what way the assessed future policy may contribute or not to a more sustainable agriculture and society as a whole. The indicator framework can consequently be seen as an indicator "sorting" tool that helps its user to see the impacts of the future policy from different perspectives. Figure 9.11 illustrates the panels for managing the indicators.

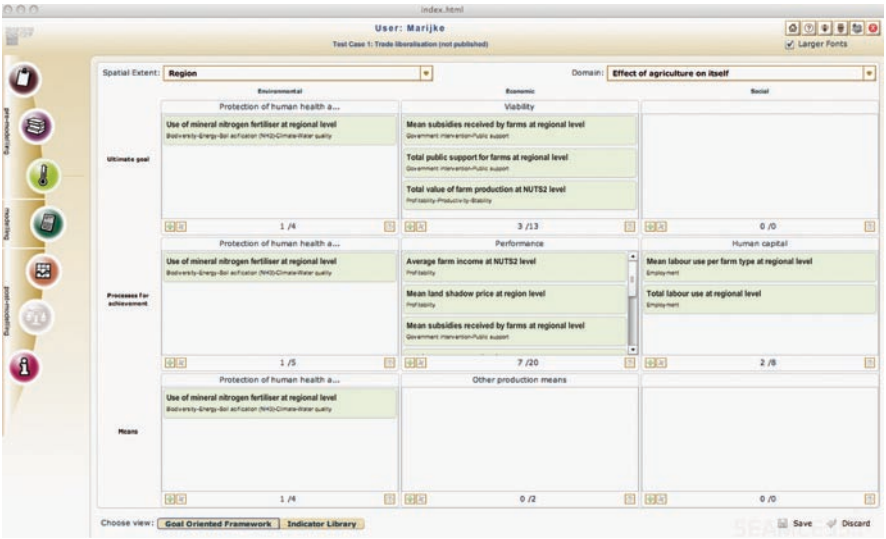

Fig. 9.11 Indicator manager screenshot

Two types of indicators are available in SEAMLESS-IF. First, the so called endorsed indicators which are fully described by a group of indicator experts and documented with PDF documents that can be reviewed from within SeamGUI. The other type is called model variables, basically the data exchanged between models on the OpenMI level, and documented on a model basis.

The selection of indicators for a project triggers the model chain calculation, and if supported by a model it can optimise its calculations based on the requested results. The visualization of results in SeamGUI is also based on this selection of indicators.

Constrained Choice of the Model Chain by Scale Selection (Modelling Phase)

The scales that are relevant when performing an integrated assessment are the scales of the problem definition, including policy options, contexts, outlooks and indicators. The scales (spatial and temporal extents/resolution) of the problem are defined by the integrative modeller jointly with the policy expert. The scale of a model is defined by the modeller, taking into account the scales of the available data. The interpretation of the problem scale into the model scale is made by the modeller when the model is integrated in SEAMLESS-IF.

The selection of the extent and the resolution of the problem scale in SEAMLESS (in the pre-modelling phase) implies the scale of the models and also the model chain to be used. Specifically, in the final release of SEAMLESS-IF the combinations

Table 9.2 Problem scales and model chains in SEAMLESS-IF

Extent	Resolution	Model chain[11]
Regional	AEnZ	FSSIM-AM/APES/FSSIM-MP
EU25	Farmtype	CAPRI/EXPAMOD/FSSIM-MP/FSSIM-AM
Farmtype	AEnZ	FSSIM-AM/APES

reported in Table 9.2 are foreseen, but more can be added, thanks to the extensibility of the proposed approach.

Detailed Specification of Experiments

A project may encapsulate several experiments. Each experiment corresponds to a single run of the model chain. All experiments in the same project refer to the same model chain, as the latter is implied by the scales of the problem. An experiment consists of three parts: the *Outlook*, that defines the trends of the envisioned future, the *Context* that specifies the boundaries of the problem and is specific to the biophysical subchain, and the *Policy option* that defines the conditions for the policy assessment subchain. In Fig. 9.12 we display, as example, the policy option panel.

Chain Execution, Server-Side Queuing Model Execution and Client-Side Monitoring

When the modeller has completed the configuration of a model chain (that is, of the experiment), s/he can start its execution. This adds the newly configured model chain to the queue of the model chains to be executed on the server. SeamServer provides the queuing facility. The end-user may monitor the queue on his/her client and see which model chain is next to be executed. When an experiment arrives to the top of the queue, it is retrieved by the (one of the) processing environment(s) that executes it on a server. The model chain type is retrieved from the experiment, instantiated, initialized with the experiment and executed. While models of the chain are finishing, intermediate results and indicator values are linked to the experiment. When the chain is finished, it links the final results to the experiment. As the results are attached to the experiment, they can be visualized with SeamPRES, a component of the SeamGUI application.

[11] Model name abbreviations: FSSIM – Farm System Simulator, FSSIM-AM – the Agricultural Management part of FSSIM; FSSIM-MP – the Mathematical Programming part of FSSIM, SEAMCAP – a SEAMLESS version of the 'Common Agricultural Policy Regionalised Impact analysis' model, APES – Agricultural Production and Externalities Simulator.

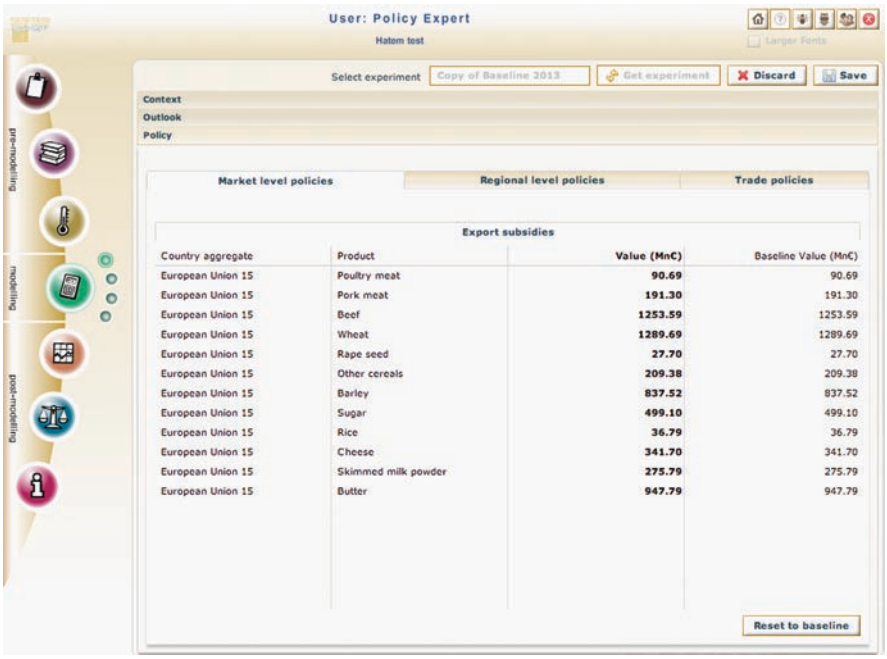

Fig. 9.12 Detailed experiment configuration in the SeamGUI. The example gives the export subsidies for different commodities in a policy proposal

At present, only one model chain can be executed at a time, but the current design also allows for parallel execution of more model chains at once, by installing multiple (virtual) servers with all required models and a Processing Environment.

The Visualization Environment (Post-modelling Phase)

SeamPRES is the visualization tool in SEAMLESS-IF, implemented as a component of the SeamGUI application. It enables the users of SEAMLESS-IF to interactively display impacts, indicators and model outputs and provides clipboard copy and paste integration with other applications. For example it is possibly to copy table data from SeamPRES to the systems clipboard and paste it into another application like Microsoft Excel for further processing. SeamPRES is a tool that can digest and visually display SEAMLESS model outputs in various ways, to improve the analysis and the dissemination of the model results. These model outputs are either available directly, processed (or copied) into indicator results or, when compared based on experiments and expected changes, as impacts.

The initial version of SeamPRES can retrieve calculated indicator values and display those in three major ways: tables, graphs and maps. As an integrated component of

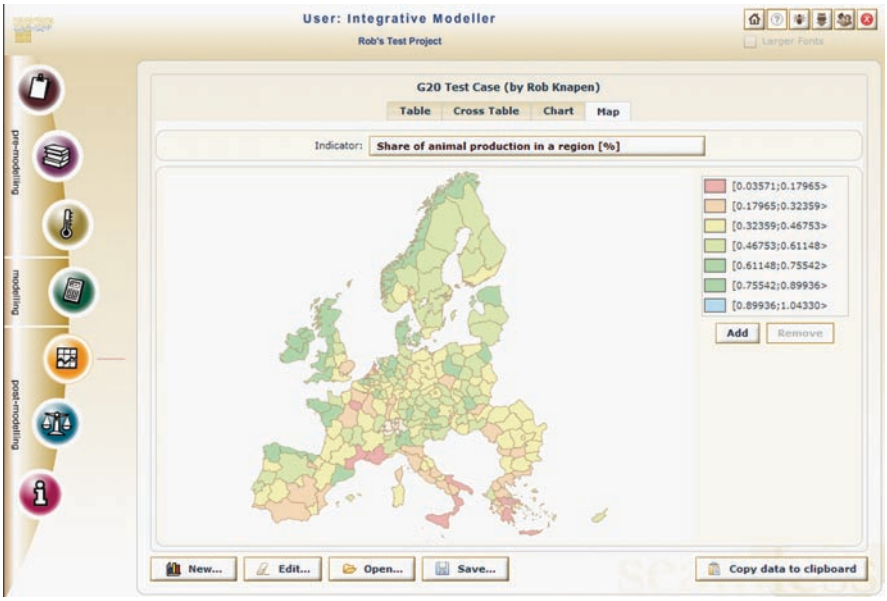

Fig. 9.13 SeamPRES map viewer displaying indicator results per NUTS2 regions

SeamGUI SeamPRES uses all information and functionality from the other components to perform its specific tasks. For example the selection of indicators is used in SeamPRES to decide what values are supposed to be visualized and possible compared to each other, avoiding some of the possible illogical choices that a user could make. SeamPRES uses the same functionality as the rest of SeamGUI to communicate with the SeamServer to retrieve the calculation results. Some specific optimisations have been made to cope with the large amount of data and variety of types of indicators that have to be processed (see also Fig. 9.13).

Conclusions

The challenge in integrated modelling is the conceptual integration. To achieve this, we need explicit semantics and a shared conceptualization. A participatory and collaborative approach is a key success factor for the creation of a common ontology for models, indicators and raw data. SEAMLESS has achieved a very important shift in integrated modelling practice, i.e. modellers specify the data requirements of their models in a higher level, i.e. that of an ontology. This ontology is then automatically transformed into a relational database model, to which "data collecting" activities need to comply with. This ensures the match between data collection and data requirement for running the model-chain. It also provides the required semantic

interoperability. The use of ontology has proved to be very useful if not essential both for the technical integration of knowledge in the SEAMLESS Integrated Framework and in understanding the meaning of communicated words of the diversity of people within the project.

The client and server technologies that were adopted in SEAMLESS-IF have an advantage in the field of stability and security by replacing custom coding with well-known and tested open source frameworks, use of well-known design patterns, and component-oriented programming. By replacing XML with the binary AMF format for exchange of information between clients and server, we improved performance. Tests show that this is at least four times faster. Another advantage and an important requirement for the system is in the field of maintainability and flexibility. There is less custom source code to maintain because of use of popular open source frameworks and standards, and part of the source code is generated from a higher level ontology.

References

Athanasiadis, I. N., & Janssen, S. (2008). Semantic mediation for environmental model components integration. *Information Technologies in Environmental Engineering, 1*, 3–11.

Bezlepkina, I., Domínguez, I.P., Heckelei, T., Romstad, E., & Oude Lansink, A.G.J.M. (2007). *EXPAMOD: Component to statistically extrapolate from FSSIM models to other farm types and regions including aggregation to NUTS2: Motivation, description and prototype*. PD3.6.11.2, SEAMLESS Integrated Project, EU 6th Framework Programme, contract no. 010036-2 (p. 28), from www.SEAMLESS-IP.org

Britz, W., Pérez, I., Zimmermann, A., & Heckelei, T. (2007). *Definition of the CAPRI core modelling system and interfaces with other components of SEAMLESS-IF*. SEAMLESS Rep. No. 26, SEAMLESS Integrated Project, EU 6th Framework Programme, contract no. 010036-2 (p. 116), from www.SEAMLESS-IP.org

Cash, D. W., Clark, W. C., Alcock, F., Dickson, N. M., Eckley, N., Guston, D. H., et al. (2003). Science and technology for sustainable development special feature: Knowledge systems for sustainable development. *PNAS, 100*(14), 8086–8091.

Dahl, O.-J., Dijkstra, E. W., & Hoare, C. A. R. (1968). *Structured programming*. London: Academic.

Farquhar, A., Fikes, R., Pratt, W., & Rice, J. (1995). *Collaborative ontology construction for information integration* (Tech. Rep. No. KSL-95-63). Stanford, CA: Knowledge Systems Laboratory, Department of Computer Science, Stanford University.

Fowler, M. (2007). *Patterns of enterprise application architecture*. Boston, MA: Addison-Wesley.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. M. (1994). *Design patterns: Elements of reusable object-oriented software*. Upper Saddle River, NJ: Pearson Education.

Gijsbers, P.J.A., & Gregersen, J.B. (2005). OpenMI: Glue for model integration. In A. Zerger & R.M. Argent (Eds.), *MODSIM 2005 International Congress on Modelling and Simulation* (pp. 648–654). Australia/New Zealand: Modelling and Simulation Society.

Gijsbers, P.J.A., Wien, J.E., Verweij, P., & Knapen, R. (2006). Advances in the OpenMI. In *Proceedings of 7th International Conference on Hydroinformatics, HIC 2006* (pp. 72–81).

Gregersen, J. B., Gijsbers, P. J. A., & Westen, S. J. P. (2007). OpenMI: Open modelling interface. *Journal of Hydroinformatics, 9*(3), 175–191.

Griffen, E. M. (1997). *A first look at communication theory*. New York: McGraw-Hill.

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition, 5*, 199–220.

Gruber, T. R. (1995). Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human and Computer Studies, 43*(5/6), 907–928.

Holsapple, C. W., & Joshi, K. D. (2002). A collaborative approach to ontology design. *Communications of the ACM, 45*(2), 42–47.

IEE (Institute of Electrical and Electronics Engineers). (1990). *IEEE standard computer dictionary: A compilation of IEEE standard computer glossaries*. New York: IEEE.

Janssen, S. J. C., Andersen, E., Athanasiadis, I. N., & van Ittersum, M. K. (2009). A database for integrated assessment of European agricultural systems. *Environmental Science and Policy 12*(5), 573–587.

Janssen, S.J.C., Wien, J.J.F., Li, H., Athanasiadis, I.N., Ewert, F., Knapen, M.J.R., Huber, D., Thérond, O., Rizzoli, A., Belhouchette, H., Svensson, M., & van Ittersum, M.K. (2007). Defining projects and scenarios for integrated assessment modelling using ontology. In L. Oxley & D. Kulasiri (Eds.), *MODSIM 2007 International Congress on Modelling and Simulation* (pp. 2055–2061). Australia/New Zealand: Modelling and Simulation Society.

Kilov, H. (1990). From semantic to object-oriented data modeling. In *First International Conference on System Integration* (pp. 385–393).

Knapen, M.J.R., Verweij, P.J.F.M., & Wien, J.J.F. (2007). Applying enterprise application architectures in integrated modelling. In L. Oxley & D. Kulasiri (Eds.), *MODSIM 2007 International Congress on Modelling and Simulation* (pp. 798–804). Australia/New Zealand: Modelling and Simulation Society.

Neches, R., Fikes, R., Finin, T., Gruber, T., Patil, R., Senator, T., et al. (1991). Enabling technology for knowledge sharing. *AI Magazine, 12*(3), 36–56.

Ogden, C. K., & Richards, I. A. (1923). *The meaning of meaning* (8th ed.). New York: Harcourt, Brace & World.

Patel-Schneider, P.F., Hayes, P., & Horrocks, I. (2004). *OWL web ontology language semantics and abstract syntax*. W3C Recommendation. Retrieved from http://www.w3.org/TR/owl-semantics/

Pennington, D.D., Madin, J., Villa, F., & Athanasiadis, I.N. (2007). Computer-supported collaborative knowledge modeling in ecology. Workshop on social and collaborative construction of structured knowledge. In *16th International World Wide Web Conference (WWW2007). CEUR Workshop Proceedings, 273*. BANFF, Canada published online.

Raskin, J. (2000). *The humane interface – new directions for designing interactive systems*. New York: ACM Press.

Rizzoli, A.E., Athanasiadis, I.N., & Villa, F. (2007a, September). Delivering environmental knowledge: A semantic approach. In O. Hryniewicz, J. Studzinski, & M. Romaniuk (Eds.), *Proceedings of 21st International Conference on Informatics for Environmental Protection: EnviroInfo 2007* (Vol. 1, pp. 43–50). Germany: Shaker Verlag.

Rizzoli, A. E., Donatelli, M., Athanasiadis, I. N., Villa, F., & Huber, D. (2008). Semantic links in integrated modelling frameworks. *Mathematics and Computers in Simulation, 78*(2–3), 412–423.

Rizzoli, A., Li, H., Athanasiadis, I.N., & Marechal, F. (2007b). Semantically rich interfaces for simulation interoperability. *2007 European Simulation Interoperability Workshop* (p. 60), Simulation Interoperability Standards Organization, Genoa, Italy.

Rotmans, J., & Dowlatabadi, H. (1998). Integrated assessment modeling. In S. Rayner & E.L. Malone (Eds.), *Human choice and climate change: Tools for policy analysis* (Vol. 3, pp. 291–377). Columbus, OH: Batelle Press.

Rotmans, J., & van Asselt, M. (1996). Integrated assessment: A growing child on its way to maturity. *Climatic Change, 34*(3–4), 327–336.

Schoemaker, P. J. H. (1993). Multiple scenario development: Its conceptual and behavioral foundation. *Strategic Management Journal, 14*(3), 193.

Scholten, H., Kassahun, A., Refsgaard, J. C., Kargas, T., Gavardinas, C., & Beulens, A. J. M. (2007). A methodology to support multidisciplinary model-based water management. *Environmental Modelling and Software, 22*(5), 743–759.

Sølvberg, A. (1998). Data and what they refer to. In P. P. Chen (Ed.), *Concept modeling: Historical perspectives and future trends*. Berlin: Springer.

van der Aalst, W. M. P., & van der Hee, K. M. (2002). *Workflow management: Models, methods and systems*. Cambridge: MIT Press.

Van Ittersum, M. K., & Donatelli, M. (2003). Modelling cropping systems [Special issue]. *European Journal of Agronomy, 18*(3–4), 187–394.

Van Ittersum, M. K., Ewert, F., Heckelei, T., Wery, J., Alkan Olsson, J., Andersen, E., et al. (2008). Integrated assessment of agricultural systems - a component based framework for the European Union (SEAMLESS). *Agricultural Systems, 96*, 150–165.

Verweij, P.J.F.M., Knapen, M.J.R., & Wien, J.J.F. (2007). The use of OpenMI in model based integrated assessments. In L. Oxley & D. Kulasiri (Eds.), *MODSIM 2007 International Congress on Modelling and Simulation* (pp. 1166–1171). Australia/New Zealand: Modelling and Simulation Society.

Villa, F. (2005). A semantic model of computation for natural system modelling. In A. Zerger & R.M. Argent (Eds.), *MODSIM 2005 International Congress on Modelling and Simulation* (pp. 751–757). Australia/New Zealand: Modelling and Simulation Society.

Villa, F., Athanasiadis, I. N., & Rizzoli, A. E. (2009). Modelling with knowledge: A review of emerging semantic approaches to environmental modelling. *Environmental Modelling and Software, 24*(5), 577–674.

Villa, F., Donatelli, M., Rizzoli, A.E., Krause, P., Kralisch, S., & van Evert, F.K. (2006). Declarative modelling for architecture independence and data/model integration: A case study. In A. Voinov, A. Jakeman, & A.E. Rizzoli (Eds.), *Proceedings of the iEMSs Third Biennial Meeting, "Summit on Environmental Modelling and Software"*. Burlington, VT: The International Environmental Modelling and Software Society.

Wien, J. J. F., Knapen, M. J. R., Janssen, S. J. C., Verweij, P. J. F. M., Athanasiadis, I. N., Li, H., et al. (2007). Using ontology to harmonize knowledge concepts in data and models. In L. Oxley & D. Kulasiri (Eds.), *MODSIM 2007 International Congress on Modelling and Simulation* (pp. 1959–1965). Australia/New Zealand: Modelling and Simulation Society.

Zander, P., & Kächele, H. (1999). Modelling multiple objectives of land use for sustainable development. *Agricultural Systems, 59*, 311–325.