Trade-offs in the design of cross-disciplinary software systems

[1]T. van der Wal, [1]R. Knapen, [2]M. Svensson, [3]I. Athanasiadis and [3]A.E. Rizzoli,

[1]Alterra, [2]LUCSUS - Lunds Universitet, [3]IDSIA, E-Mail: Tamme.vanderWal@wur.nl

***Keywords**: integrated assessment frameworks, software design, ontologies, modeling and simulation.*

EXTENDED ABSTRACT

As researchers we are often faced with the difficult and demanding task of preparing models, and their computer implementations, for decision making, or, more recently, for integrated assessment. Such assessment often involves large scale problems, where the decisions to be made can deeply affect the environment, the social context and the economic background of regions and even nations.

Yet, we face the grim reality that a model is a focused representation of the world, and it is always a result of several compromises in terms of details and structure, leading to trade-offs in terms of complexity, flexibility and performance. This trade-off becomes an essential design property. We often wish our models to be as simple as possible, balancing transparency, understandability and level of detail.

Now, we are involved in the SEAMLESS project, an EU FP6 Integrated Project, aims at generating an integrated framework of computer models. This framework can be used for assessment of how future alternative agricultural and environmental polices affect sustainable development in Europe. Thus, we are designing a cross disciplinary software system to deal with different simulation domains. In this, we need to take care of many differences between the different modeling societies.

We decided to apply an architecture centric development method and evaluated this with stakeholders based on a so-called Architecture Trade-off Analysis method. When prioritizing the requirements we used a cost-benefit analysis as a weighting factor for deciding what to do first. Requirements were grouped in user-roles, that appeal to differences in user-interface options.

The resulting software architecture identified the necessity to identify two major blocks: the modeling environment, to be used by a number of user roles, mostly modelers and coders, and the processing environment, which is oriented towards the needs of those user roles more focused on system analysis, rather than design and implementation. Another key factor of our architecture is the knowledge base, which provides a common repository for all knowledge, data, model sources which are shared by the two environments.

When moving on from architecture to design and implementation, we tried to steer clear of the risk of inventing another modelling framework, and therefore in our prototype we use different existing frameworks for different tasks in the overall design. This means that we discussed the view that ‘one tool fixes everything’, and we chose to rely on specific frameworks for specific needs. We chose a modelling framework with a track record in crop modeling, to target our biophysical modeling needs, and we selected a de facto standard framework for economic modeling to solve agri-economic modelling problems. All of this comes at a price, that is the extra effort required to integrate different frameworks. We chose therefore to develop an evolution of the OpenMI integration framework to target this issue.

In this article we describe all the risks we have identified as associated to our architecture centric approach and how we dealt with them. This article describes the design of the modeling framework for SEAMLESS. A first prototype is ready in January 2006.

1. INTRODUCTION

Free trade, globalization, the opening of markets pose a whole set of new questions to agricultural policy makers. The World Trade Organisation (WTO) negotiation rounds define the regime of tariffs which consequently define the market price for agricultural goods at the global scale. Yet, agricultural production is the result of plant growth processes in fields, which are managed by farmers at a local scale. This means that effects of policy changes at the macro level are governed by and have an impact on the micro level. Integrated analyses and assessment are then required to make the ends meet, in order to be able to answer fundamental questions on how the environment through agricultural production will react to a new regime of subsides and, at the same time, what social impacts will be measured in terms of changes in the employment rate, and how supply will change in response to the new economic conditions.

Such integrated assessments have to deal with both the global context as well as its microscale effects and they call for a new modeling arrangement. The SEAMLESS project (www.seamless-ip.org) aims at providing a framework able to deal with such demands, based on the joint knowledge and experiences of 32 highly valued partners in agri-environmental research. This framework provides structure and tools for building and deploying models and corresponding datasets.

SEAMLESS incorporates disciplines from a whole range of domains, including agricultural policy evaluation, agro-economics, rural sociology, farm management and business administration, agronomy, crop and soil modeling, ecology etc. To integrate all these disciplines in a coherent simulation framework involves three main challenges:

1. Each discipline has developed it's own language and definitions, which is mirrored in their modeling and data. Cross-discipline integration requires careful analysis and interpretation of the jargon;

2. Each discipline has its own breed in modeling, which is due to the nature of the subject studied and governed strongly by the application and the focus of the 'customer' of the modeling exercise. We must integrate numerical simulation, used to model biophysical systems, with mathematical programming and expert rule systems, used to solve farm-economical management problems;

3. The diversity of the user community stretches from policy makers to scientific software engineers. We need to integrate the requirements from all user *roles* in the framework and decide how different roles are expressed in user interfaces.

Dealing with these issues will increase the complexity of all current systems. It has been noted this is subject to the trade-off between complexity, flexibility and performance (Wal et al., 2003). This means an increase in complexity must result in a decrease in flexibility or performance or both. It has been shown however that depending on the purpose of an integrated system, simplification in modelling, or even a game, can be very successful in addressing the complexity of a system (Erisman et al, 2002). This clearly requires a complete redesign of your modeling system.

This paper discusses the requirements for architecture and design of the SEAMLESS framework resulting from experiences, literature and stakeholder interviews. We address the trade-offs mentioned before.

2. DESIGN TRADE-OFFS FOR INTEGRATION

The software engineering community has suffered from severe miscommunication between developers and customers, leading to an IT reputation of never delivering on time, never delivering within budget and never delivering what is wanted. The 2001 update of the CHAOS report (Standish Group, 2001) shows that in 2000 only 28% of all IT projects manage to deliver what is promised in time and within budget. It also mentions that the top 4 factors for project success are all related to communicating with and managing the stakeholders of the project.

The Carnegie Mellon Software Engineering Institute (SEI) has developed the Architecture Tradeoff Analysis Method (ATAM) to help a system's stakeholder community understand the consequences of architectural decisions with respect to the system's quality attribute requirements and business goals (Nord et al., 2003). This is part of the architecture centric development method characterized by the following steps (amongst other, Nord et al, 2003):

1. defining the business case for the system;

2. understanding the requirements;

3. selecting, refining or creating the software architecture;

4. documenting and communicating the software architecture;

5. analyzing or evaluating the software architecture;

6. implementing the system based on the software architecture;

7. ensuring that the implementation conforms to the software architecture.

In SEAMLESS we have applied the architecture centric development to allow the stakeholders to participate in the design and decision process of building the SEAMFRAME architecture. Besides stakeholder interviews and collecting requirements in a user-centered design method, derived from agile programming methods, we have also created a dummy system, the so-called *animated narrative-demo* (AND), to show to stakeholders how the system might look like and it can be used. The AND serves as an electronic discussion document and has proved is value in communication and requirements gathering.

Based on ATAM we created a list of risks leading to management decisions for dealing with these risks. We found that the main risks can be summarized as: unrealistic user expectations, reinventing the wheel, and premature obsolescence of the framework. Let's see them in detail.

User expectations, expressed as business goals as well as requirements, are very much based on the Aladdin's lamp assumption. The software engineers are able to fulfill every wish, and they deliver a dreamed system that solves all problems. The truth is that we already have enough challenges trying to integrate the current situation and make it work. Stepwise improvement of the system must gradually introduce new, wanted, requirements in addition to the business goals. Managing user expectations is a major task.

Reinventing the wheel is a serious danger since there are considerable efforts done already in the agricultural and adjacent domains with regard to integrated modeling and framework architectures. We do not want to create a completely new framework but try to connect to these other initiatives and apply relevant parts to our system. The Yet Another Modeling Framework (YAMF) syndrome has very well landed within the SEAMLESS community. The main risk is however that we incorporate architectural aspects in our design that do not comply to our business case, just because they are there already. On the other hand, there are lots of legacy systems and dealing with them and their authors is a challenge in its self.

Even when we deliver the final framework, we might find that it is already obsolete. While lifespan assumptions and life cycle analysis in software engineering suggests a longer life time of the framework in comparison to its components, it is however our experience so far that in the science community the life time of the component exceeds the (expected) life time of the framework. This life cycle inversion directs our efforts in respecting the individual value of the component above the value of the framework. We have taken good notice of this in the design and architecture.

3. MODULARITY FOR INTEGRATION

All of above has led us to propose a global design, based on 3 coherent building blocks that can be deployed in an integrated fashion to deliver the required functionality. The main blocks are the Modelling Environment, the Processing Environment, and the Knowledge Base (figure 1). The former grants access to *Sources* (data, software components, results, etc.) by means of a set of *ontologies*, which conceptualise the structure and organization of the sources.

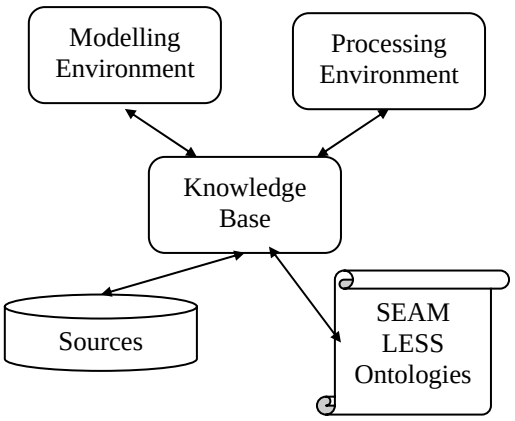

Figure 1. The SEAMLESS building blocks.

The *Modeling Environment*, which targets the needs of the different kind of modelers, from biophysical modelers, to economic modelers. We will implement different modeling environment variants depending on the type of modeling required; e.g. the modeling environment for economic models will be build around the GAMS

language and engine. Any Modelling Environment delivers models, which are packaged as software components (Szyperski, 2002). A software component therefore implements a model that can be simulated or optimized, according to the computational engine it embeds. The modeling environment reflects the need to build new models in an integrated fashion. Modelers can easily develop their ideas into software and deploy it in the standardized environment. The modeling environment is a real productivity tool and should decrease the time-to-market of new models considerably.

The *Processing Environment* allows the system analysts to access the work of the modelers. It allows analysts to define and execute scientific workflows, which are sequences of operations on data and models. The scientific workflow defines how to connect systems together to perform a certain analysis. The processing environment finally, comprises the high-level integration that connects all sources into a executable system. This includes user-interface elements such as user dialogues and presentation of scientific data, in graphs, tables and maps.

All relevant components for integrated modeling, such as models, databases or datasets, expert rules and all other formally and digitally captured knowledge are stored in the *Sources.* Our sources will be primarily existing models and databases, just conveniently wrapped in a fashion to be deployed in the processing environment. Existing sources can be in any language and can also stem from other frameworks. The modeling environment integrates these sources into components.

Access to the sources is mediated by the *Knowledge Base* that includes functionality and rules to use the ontologies to semantically mark up the sources. For instance, the ontology can be used to store concrete information on purpose and application of a source, and it also informs the user about scale, scope and restrictions in use.

Thanks to the SEAMLESS *ontologies*, the knowledge base includes functionality to negotiate between sources when the user wants to connect them and to automatically add converters in the scientific workflow (Rizzoli *et al*., 2005).

4. SOFTWARE REUSE FOR INTEGRATION

This block system has been elaborated in several sessions involving all stakeholders (developers and users) during the first phase of the project. We have used diverse technologies for this, ranging from writing papers and documents, discussions at meetings and developing software parts to provide some hands-on examples to the user community. During these 'negotiations' we have analysed the outcomes based on the Cost Benefit Analysis Method (CBAM) (amongst other, Nord et al., 2003) in order to direct the project in a manageable fashion towards available budget and resources. Besides prioritization this has also led to trade-offs with regards to so-called 'build-or-buy' decisions, which in the open source segment better is defined by 'build-or-borrow'. In our steps 6 and 7 (implementation and verification) of the ATAM architecture-centric development we have chosen to develop a prototype, which will be the basis for further developments. This prototype uses existing parts to a maximum, while it also focuses on integration of the current state-of-the-art, and does not impose too many enhancements right away.

We had to select the building blocks on which to start developing the Modelling and the Processing Environment, and the Knowledge Base.

The choice for the modeling environment was not easy, given the requirement of being able to model across a wide scope of disciplines and spatial and temporal scales. We therefore decided to target the development of two modeling environments, one for biophysical modeling and another one for farm-economical modeling. For the former we have chosen for MODCOM (Hillyer *et al.* 2003) while we have evaluated several alternatives (see also Evert *et al.* 2005, in these proceedings). The latter modeling environment is based on GAMS (http://www.gams.com), which is a *de facto* standard for agricultural-economic modeling.

For the processing environment we decided to rely on the OpenMI framework (http://www.openmi.org). Although primarily designed and build in the hydrological domain, the OpenMI framework offers functionality and interfaces that suits our need nicely. Most important, it is open source (as MODCOM is) and this allows us to contribute to its development.

For the knowledge base we rely primarily on existing ontologies and existing tools. We are inspired by projects like KEPLER (http://www.kepler-project.org) and SWEET, the Semantic Web for Earth and Environmental terminology (http://sweet.jpl.nasa.gov/ontology/) where similar, but more ambitious efforts are undertaken. The software we are developing is based on the Java libraries provided by the Jena (http://jena.sourceforge.net) and Protégé (http://protege.stanford.edu) projects.

In order to integrate all these building blocks, we have designed the Seamless OpenMI+ Framework Architecture (SOFA), as a comfortable and soft landing of all these systems. SOFA is a SEAMLESS specific implementation of OpenMI. The '+' in OpenMI refers to the Requests For Change (RFCs) we have already implemented and we have offered to the OpenMI consortium for adoption in the standard. These refer mainly to adaptations of OpenMI to a more generic modeling scene than just hydrology.

SOFA's main responsibility is to integrate components (sources, tools, environments) using the mediation of the knowledge base.

5. USERS AND FRAMEWORKS

On top of the architecture and using the frameworks, software applications are built to satisfy the needs of different users. For the sake of analysis we have categorized the different user roles that can be distinguished within our user community:

- Coder: creates new data structures, new models and new applications, using the SEAMFRAME environment as a productivity tool for better time-to-market;
- Linker: builds scientific workflows, dedicated to specific research questions. A Linker combines existing models and databases that suit his purpose.
- Provider: brings in legacy software components and needs functionality to describe and formalize the components for further reuse;
- Runner: executes a chain of components in a scientific workflow by changing parameters and evaluating the results;
- Player: plays around with prepared experiments and explores alternatives. The Player differs also from a Runner in its attitude such as impatience, attracted by GUI elements and a desire to test the system capabilities;
- Viewer: looks at prepared results and their interpretation, as presented in Reference Book kind of application (Wien et al., 2005).

Through mixing and matching of components different applications will be assembled to support tasks of different user roles. All applications are from the same breed, which is shown in their deployment of the same framework and the re-use of software components.

The architecture centric development allows to abstract from the desired functionality to a base set of requirements for the whole system. The framework does not prevent extensions and adaptations; on the contrary, thanks to its component based structure, it provides the necessary elements to build upon the framework and its components. This has been made possible by defining framework concepts for specific parts and separating the modeling environment from the linking environment acknowledging the essential difference between the two: a modeling environment is basically meant for coders. The processing environment is used by all user roles as it is the backbone of end-user applications. In this analogy, the knowledge base is the spinal cord. The explicit separation of processing and modeling environment allows us to integrate different modeling environments in our system. In SEAMLESS we will have one modeling environment for equilibrium models using multiple goal linear programming tools like GAMS. Another modeling environment will be build upon MODCOM. The third modeling environment concerns a declarative modeling tool, that provides a sound way of documentation and knowledge transfer of model (-equations). When a model is declaratively expressed, the equations are clearly separated from the imperative code that performs the numerical integration and data preparation. The model is therefore written using a text based declarative language, which states facts that are true about the model and that can be processed by an inference engine. Think of a declarative model as a Prolog program. A key advantage of using a declarative language to store models is the capability to export models according to different implementation requirements and even platforms (Muetzelfeldt, 2004).

The SOFA takes care of the integration of the different frameworks and enforces the use of the knowledge base in retrieving and deploying components.

6. CONCLUSIONS

Large scientific software projects, like in SEAMLESS benefit greatly from an architecture centered development method. The lack of a clearly defined customer with clearly specified requirements could be well tackled with the trade-off analysis where stakeholders could clearly indicate what restrictions were and were not acceptable.

When retrieving user requirements after all, we have made use of software parts, presented as mock-ups, that show (potential) users the vision we have with the software. These software parts were very useful in focusing on specific functionality and made the abstract discussions

much more concrete. Besides that, our animated narrative demo made sure all stakeholders keep their minds also with the greater picture or the overall function of the system.

When converting user requirements into planning and prioritize the different requirements we found a cost-benefit analysis very worthy. The costs were mainly assessed by software engineers and expressed in terms of man-hours work.

The separation of the linking function of the framework from the modeling function has been a very useful line of work. Besides a risk-mitigation effect it also provides another hook for variation and extension.

The choice for the existing frameworks to be included in the overall SEAMFRAME environment is not undisputed. The fact that we have made a choice in the first place was one of the best things that happened in the project, because it allowed us to move forward towards realization. Whether these frameworks are good enough will be assessed after completing and using the SEAMLESS prototype. This is due in June 2006. With a life expectation of components that exceeds the life expectation of the framework, we anticipate for reusing the components in other frameworks, either in parallel in other projects or as a successor of the current framework.

With the SEAMFRAME design and architecture that has been described here, we think we can cope with the main challenges mentioned in the introduction: dealing with different languages and domains; dealing with different modeling paradigms and dealing with different user roles.

7. ACKNOWLEDGMENTS

This publication has been partially funded under the SEAMLESS integrated project, EU 6th Framework Programme for Research, Technological Development and Demonstration, Priority 1.1.6.3. Global Change and Ecosystems (European Commission, DG Research, contract no. 010036-2). The authors wish to thank all our partners in this project for their fine collaboration; their names cannot be listed here, but you can find more information on the project website: http://www.seamless-ip.org.

8. REFERENCES

Erisman, J.W., A. Hensen, W. de Vries, J. Kros, T. van der Wal, W. de Winter, J. E. Wien, M.van Elswijk, M Maat and K. Sanders (2002): 'NitroGenius: A nitrogen decision support system. A game to develop the optimal policy to solve the Dutch nitrogen pollution problem'. *Ambio 31* (2): 190-196.

Evert, F. van, D. Holzworth, R. Muetzelfeldt, A.E. Rizzoli, F. Villa. 2005. Convergence in integrated modeling frameworks. *MODSIM 05*, 11-15 December, Melbourne, Australia.

Hillyer, C., J. Bolte, F, van Evert, A, Lamaker. 2003. The ModCom modular simulation system. *European Journal of Agronomy*, 18 (3-4).

Muetzelfeldt, R.I. 2004. Declarative Modelling in Ecological and Environmental Research. European Commission Directorate-General for Research, Position Paper no. EUR 20918. European Commission, Brussels, B.

Nord, R.L., M.R. Barbacci, P. Clements, R. Kazman, M. Klein, L. O'Brien, J.E. Tomayko. (2003). Integrating the Architecture Tradeoff Analysis Method (ATAM) with the Cost Benefit Analysis Method (CBAM). *Technical Note, CMU/SEI-2003-TN-038*, Carnegie Mellon Software Engineering Institute, Pittsburgh, PA.

Rizzoli, A.E., M. Donatelli, I. Athanasiadis, F. Villa, R. Muetzelfeldt, D. Huber. (2005). Semantic links in integrated modeling frameworks. *MODSIM 2005*, 11-15 December, Melbourne, Australia.

Standish Group International Inc., (2001), Extreme Chaos, www.standishgroup.com.

Szyperski, C., Gruntz, D., Murer, S. 2002. *Component Software – Beyond Object-Oriented Programming, Second Edition*. ACM Press, New York, NY.

Wal, van der T., J.J.F. Wien, and A.J. Otjens (2003), The application of frameworks increases the efficiency of knowledge systems, proceedings of the 4th EFITA conference on ICT in agriculture.

Wien, J.J.F., M. Blind, and T. van der Wal, (2005) The AQUASTRESS integrated Solutions Support System, *MODSIM 2005*, 11-15 December, Melbourne, Australia.