Software Architectures for Distributed Environmental Modeling

Ari Jolma[1] and Kostas Karatzas[2]

[1] Department of Civil Engineering, Aalto University,
Niemenkatu 73, 15140 Lahti, Finland
[2] Department of Mechanical Engineering, Aristotle University of Thessaloniki,
54124 Thessaloniki, Greece
ari.jolma@aalto.fi, kkara@eng.auth.gr

Abstract. Environmental modeling is increasingly more integrative and collaborative work. The Internet and Web hold a promise of a shared environmental information infrastructure, which supports modeling. While the traditional (enterprise) information system architectures have not been much employed in the environmental domain, new standards-based Web technologies create an opportunity. At the same time the modeling workflows are being investigated, and this work may provide a fruitful starting point for new kind of top-down services that support modeling within environmental information infrastructures.

Keywords: Environmental modeling, Distributed information system, Software architecture, Workflow.

1 Introduction

Developing purposeful and credible environmental models that consider all relevant aspects of the problem and integrate heterogeneous information is an acknowledged goal [1], [2]. Environmental modeling is collaborative work involving teams of experts, stakeholders and model users. Interconnecting data and tools in information systems is a traditional method for solving information processing problems. During the last 15 years information systems technology has been embracing the Internet and the Web [3], changing the paradigm of information management to information exchange [4]. Internet is a natural platform for information infrastructures that link information systems. “Shared Environmental Information System” is a concept and initiative put forward by the European Commission and European Environment Agency [5]. Information infrastructure that supports environmental modeling allows interconnecting environmental models with one another and with data. Collaboration that is based on setting up and using information services promises to improve and help modeling.

Attaining the perceived goals of a shared environmental information infrastructure naturally assumes that useful data and services exist on the Web. This is indeed already the case to some degree but often the data and services are still oriented to humans and not programs. To use such services directly in applications is possible but

J. Hřebíček, G. Schimak, and R. Denzer (Eds.): ISESS 2011, IFIP AICT 359, pp. 255–260, 2011.

error-prone and often includes cumbersome reverse-engineering. Web services are applications, which allow direct interactions with other applications using standards-based protocols and interfaces [3]. Many datasets and real-time data are available or becoming such as web services [6], [7]. Using models in the Web by human users and over the Web by other applications is in production use (our observation) and attempts to more standards-based approaches have been reported since early 2000's [8], [9], [10], [11].

Modeling is much more than just preparing and connecting datasets to various models and running the models. It is very difficult to define modeling and its boundaries to development of model-based information systems and to model-based integrated assessment. For the purposes of this paper we consider modeling to be a process where data is used and processed and various software tools are used and developed to solve problems, answer questions, and provide support for planning and management. Important elements of modeling include objectives, conceptualization, assumptions, validation, consultation, and transparency [1]. A shared information infrastructure for modeling needs to include tools for working and spelling out these.

In this paper we review some key concepts of distributed information system architectures in the light of environmental modeling and cases. Based on the review we discuss possible ways forward in developing shared information infrastructures for environmental modeling.

2 Fundamental Architectural Patterns

Distributed information systems traditionally have a three layer architecture comprising a resource layer, an application logic layer, and a presentation layer. In addition, there may be external components, which request information from or provide data to the system. The three layers can be split in various ways to tiers, which are basically separate systems that communicate in agreed, often standardized, ways. The client – server pattern is used between the tiers and external components, the best-known case being web browser clients requesting information asynchronously and already in presentation form from web servers.

The service-oriented architecture is based on the fundamental concept of a service. Services can be simple, for example obtaining a specific resource from a resource layer, or complex, for example using an application. For using complex services one needs coordination help, which basically means instructions. This is because the same service may need to be contacted several times in order to complete the requested service. Services are described to users based on their interfaces, and a service, which has a very simple interface, may actually trigger a very complex procedure within the information system.

Fundamental architectural patterns are generic designs that employ concepts like services, events, tasks and messages. An event is an occurrence of something at a particular point in time. A task is a set of actions that need to be completed. Order of completing, control flow from one action to another, timing, cancellation of actions and other things make up workflow patterns [12]. A message is an object of communication. Fundamental patterns reflect common or effective ways to deal with situations and they help define generic tools, for example programming languages, event

dispatchers, service catalogs, and message parsers. Typically information system designers and developers wish to use generic tools as much as possible in order to develop robust and maintainable systems.

Event-driven architecture is based on the concept of events happening at random. New events are taken up by an event dispatcher, which is responsible for routing the event to an appropriate handler or handlers. Events are handled by services. In some architectures event-handler can be added and removed at will.

The publish–find–bind pattern depicts how service providers publish services, how those needing services find them, and how the services are bound to applications or other services. This patterns implicitly assumes a shared service description language and services for publishing services. Linking of services that publish services is recursively possible by publishing them with each other.

A message broker receives messages, possibly validates and transforms them, and routes them to subscribers. An example of a message is a request for a specific service.

Architectural patterns can be exploited within a single application and they can be used as a model for collaboration among humans.

3 Process and Workflow Management

Environmental modeling is itself a process and it comprises several processes and tasks that deal with new measurements, parameter estimation and other things. These processes and tasks may be very specific and deal with established routines or they may be conceptually high-level considerations of things like what is the appropriate spatial aspect of the problem.

A workflow is a sequence of tasks, which is completed by an orchestrated use of applications or by humans. Workflow management systems originate historically in office automation [3]. A typical workflow application today is an administrative document processing system, for example a business travel procurement system. Programming languages have been developed for workflow systems [12]. The biggest difference between normal programming languages and workflow languages is that workflows may take days or longer to complete [3]. To cope with this workflow languages support complex recovery and exception handling routines. Workflow systems typically store the execution state into persistent storage.

Below we present two architectures, which consider processes and workflows.

3.1 Architecture for Linking Processes and Events

A process can be thought of as anything that produces a result set at a certain point in time. A process may be passive, i.e., it waits for an operator to push data into it and run it, or it may be active, i.e. it listens to events and acts under suitable conditions. The result sets of a processes are stored into a resource layer with appropriate meta data. Each addition of a result set triggers an event, which conveys information about what kind of new information is available.

Consider for example a modeling effort, which considers storm water within a municipality. The modeling proceeds as new data is obtained. The primary processes are

measurements and changes to the sensor network, engineering structures and such. Secondary processes are statistical, spatial, data cleaning and such operations, which reduce the amount of data but add to the value of new individual result sets. Tertiary processes may be models, which are run off-line but using data directly from the resource layer. Each model run produces a result set, which may be rather large but also contains appropriate meta data. Another kind of tertiary processes may be specifically designed for detecting anomalies and their results may cause events, which trigger emails or sms's.

This kind of architecture is good in the sense that it can be designed to maintain meta data that allows a high level of provenance for the modeling. The downside is that it would require rather considerable changes to current modeling practices and tools, which are not suited to such information exchange.

3.2 Architecture for Formal Modeling Workflows

The workflow that is described in [1] could possibly be coded into a workflow language. A necessary prerequisite would be an ontology for modeling tasks. The ontology should make it possible for formal treatment of stakeholders, temporal and spatial scope of the model, conceptualization, etc. Formal treatment is make possible for example by having these concepts as classes in the system. The middleware should then make it possible for storing stakeholder, conceptualization, and other such objects into the resource layer. The workflow management system for the modeling should support storing the state of the modeling project. Applications need to exist, which can be used, e.g., to create conceptualizations. The modeling workflow comprises calling such applications as unit operations. Executing the ten step modeling workflow in a specific information system would require a resource layer which can manage rather complete data about the environmental system, stakeholders and other aspects of modeling.

4 Collaboration Support

A shared information infrastructure in the Web implies collaboration among modeling teams, stakeholders, data providers and others. The technical implementation is greatly helped by adhering to technical standards, which also may make it easier for new parties to join. There are several architectural possibilities for developing collaborative information infrastructure, each having its specific model for collaboration. We can distinguish between information management, which in collaborative context implies shared information, and information exchange, which in collaborative context implies shared ontologies. A shared information infrastructure can employ both architectural patterns. In the first case there is a single resource layer in the space into which parties contribute, in the second case each party publishes resources into the infrastructure. Of course mixtures can exist too.

In collaborative modeling parties work on a shared deliverable and bring their own specific contribution into the collaboration. In collaborative software development one common architecture is a revision control system. There are two forms of the revision control system, which reflect the two forms of shared information

infrastructure described above. In the first form a single common repository is set up and parties check out versions from that and commit changes into that. In the second form each party has its own repository, where it imports and merges changes from other repositories. Collaborative modeling can mean working on a shared model (source code) in a repository.

Social networks and various toolkit for including, e.g., geospatial applets within them have recently become very popular. The technological platform is JavaScript or other programming language embedded in a web browser. Perhaps the most advanced project employing this technology and paradigm (also called Web 2.0) is iemHUB.org. IemHUB.org is a website for developing and sharing knowledge and tools for environmental systems analysis [13] started by US EPA.

4.1 An Example of a Collaborative Modeling Website

Within the CLIME project [14] the first author was involved in a collaborative modeling effort, where several teams across Europe were using the same models but each on their own catchment and/or lake. To help this effort an interactive website was developed for uploading parameter sets for comparison. The website was developed upon a rather comprehensive project database, which contains for example used models, modeled catchments and users as classes. In the CLIME database a model object contains (links to) parameters, which are again objects, and a modeled catchment contains (links to) subcatchment objects. Although the database was not formally developed as a resource layer for environmental modeling with a unifying interface more than a standard relational database, the collaborative modeling tool was relatively easy to develop as an interactive tool exploiting the database.

The collaborative modeling web page included, besides the parameter set exchange and comparison tool, a file upload tool to share model configuration files and a simple discussion tool. No web mapping tool was included although other applications based on the modeling database were developed that included one [14].

5 Discussion

In environmental information systems that support modeling, the resource layer contains measurement data, terrain data and data about civil infrastructure, conceptual data about the environmental system, management problem related data, and so on. In the application logic layer there are several applications for managing the data: importing new data, preparing it for analytical tools, various models, visualization tools, etc. In enterprise systems the support for, integration of, and management of these applications is a task for middleware and workflow management systems [3]. Traditionally in the environmental domain integrated modeling systems and geographic information systems (which are increasingly being integrated into modeling systems) have played a dominant role. For example the use of relational databases as independent resource layers in environmental modeling has been very limited.

The workflow language BPEL has received some attention and it has been used in environmental modeling to program a model use [10]. A lot is happening in the standards-based distributed data servicing due to the work carried out within the OGC.

OGC has also defined a standard for a processing service (Web Processing Service, WPS), which has recently gained interest. Mainly the interest is in publishing fundamental geospatial methods as services.

The dangers in blind use of technical solutions in modeling have been said and told in numerous papers and presentations. Our findings as reported in this paper suggest that seeking for ways to publish generic modeling knowledge as services is a possibly rewarding way forward and beneficial for modeling projects. As shortly described above in chapter 3.2 the ten step procedure from [1] is a potential starting point for such an exercise. At the same time possibilities for further developing standards-based interfaces for environmental modeling databases (resource layers) and modeling middleware should be investigated.

References

1. Jakeman, A.J., Letcher, R.A., Norton, J.P.: Ten iterative steps in development and evaluation of environmental models. Environmental Modelling & Software 21, 602–614 (2006)
2. Denzer, R.: Generic integration of environmental decision support systems – state-of-the-art. Environmental Modelling & Software 20, 1217–1223 (2005)
3. Alonso, G., Casati, F., Kuno, H., Machiraju, V.: Web Services: Concepts, Architectures and Applications. Springer, Heidelberg (2004)
4. Aberer, K.: Emergent Semantics. Invited talk at the ICSWN 2004 Conference, Paris (2004), http://lsirpeople.epfl.ch/aberer/Talks/EmergentSemantics%20ICSNW.pdf
5. Shared environmental information system, http://ec.europa.eu/environment/seis/index.htm
6. Doyle, A., Reed, C.: Introduction to OGC Web Services. OGC White Paper (2001)
7. Horsburgh, J.S., Tarboton, D.G., Piasecki, M., Maidment, D.R., Zaslavsky, I., Valentine, D., Whitenack, T.: An integrated system for publishing environmental observations data. Environmental Modelling & Software 24, 879–888 (2009)
8. Kokkonen, T., Jolma, A., Koivusalo, H.: Interfacing environmental simulation models and databases using XML. Environmental Modelling & Software 18, 463–471 (2003)
9. Eder, W.J., Zipf, A.: Towards interoperable atmospheric (air flow) models in Spatial Data Infrastructures using OGC Web Services – state of the art and research questions. In: EnviroInfo 2009 (Berlin) Environmental Informatics and Industrial Environmental Protection: Concepts, Methods and Tools, pp. 403–412. Shaker Verlag (2009)
10. Theisselmann, F., Dransch, D., Haubrock, S.: Service-oriented Architecture for Environmental Modelling – The Case of a Distributed Dike Breach Information System. In: 18th World IMACS / MODSIM Congress, Cairns, Australia, pp. 938–944 (2009)
11. Granell, C., Díaz, L., Gould, M.: Service-oriented applications for environmental models: Reusable geospatial services. Environmental Modelling & Software 25, 182–198 (2010)
12. van Der Aalst, W., Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P.: Workflow Patterns. Distributed and Parallel Databases 14(1), 5–51 (2003)
13. Integrated Environmental Modeling, http://iemhub.org
14. George, G. (ed.): The Impact of Climate Change on European Lakes. Aquatic Ecology Series, vol. 4, p. 507. Springer, Heidelberg (2010)