Jun 18th, 10:40 AM - 12:00 PM

Environmental Software Development with UML

Peter A. Khaiter
York University, pkhaiter@yorku.ca

Marina G. Erechtchoukova
York University, marina@yorku.ca

International Environmental Modelling and Software Society (iEMSs)
7th Int.Congress on Env. Modelling and Software, San Diego, CA, USA,
Daniel.P. Ames, Nigel W.T. Quinn and Andrea E. Rizzoli (Eds.)
http://www.iemss.org/society/index.php/iemss-2014-proceedings

Environmental Software Development with UML

Peter A. Khaiter[a] **and Marina G. Erechtchoukova**[a]
[a]*School of Information Technology, Faculty of Liberal Arts and Professional Studies, York University, Toronto, Ontario M3J 1P3, Canada*
pkhaiter@yorku.ca, marina@yorku.ca

Abstract: The issues of rational use of the environmental objects continue to attract public attention world-wide. There are even growing concerns on the part of various stakeholders and general public in relation to the on-going global climate change and a common demand for a harmonized type of relationships between the societal development and the environment crystalized in the concept of environmentally-friendly sustainable development. The interplay of a broad range of fields, such as economics, ecology, psychology, sociology, hydrology and agronomy, makes the area of sustainability inherently complex. Decision-making in the field requires a due support by the sophisticated environmental information systems (EISs). In this paper, we present an approach to the environmental software development on the basis of the Unified Modeling Language (UML). The main focus is placed upon the multi-layered architectural logical designs of the EISs and the dynamic or behaviour aspects of an information system being built implementing the tasks of sustainable environmental management as the problems of optimal control or multi-objective optimization.

Keywords: Environmental information system; software development; UML; actor; use case; multi-layered architecture.

1 INTRODUCTION

The problem of decision-making in, and management of, environmental systems is intrinsically complex. In order to address this complexity, environmental managers need information systems on the basis of software engineering methods and computing tools.

In this paper, we investigate the role of the Unified Modeling Language (UML) in the development of environmental information systems. The UML is a standardized graphical language being used in object-oriented software engineering for specifying, documenting and visual modeling of an IT project's artifacts or components within a wide range of applications.

With the three groups of graphical models (i.e., structure diagrams, behavior diagrams and interaction diagrams), the UML is aimed to provide a standard notation and describe different aspects of a software project. We demonstrate the ways, in which the UML can be applied in the software development for the needs of environmental management and protection.

A framework for sustainable environmental management on the basis of ecosystem services is considered as a case study. The framework is structured on the basis of the multi-layered architectural logical design. The UML graphical models are used to present the functional aspects of software blocks implementing constituting layers of the framework.

Given the fact that the UML is supported by multiple commercial and free CASE-tools, it is reasonable to expect its growing application in the field as a tool facilitating automated construction and synthesis of environmental information systems.

2 METHODS

2.1 The SDLC framework

For many years, the IT industry applies the Systems Development Life Cycle (SDLC) as a common methodology to manage a software project. The framework of SDLC specifies the stages in producing an information system. There are multiple variations on the overall SDLC (e.g., waterfall development, phased development, parallel development, evolutionary prototyping, throwaway prototyping, agile development, etc.), however, any of them has a similar set of four fundamental phases: planning, analysis, design and implementation (Dennis et al., 2005). Each phase is a container for relevant activities (Figure1).

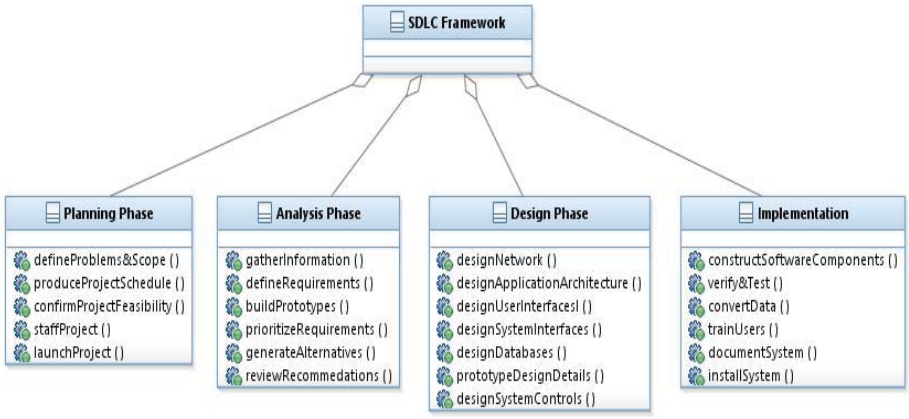

Figure 1. The SDLC framework shown as a UML class diagram using aggregation type of relationships and operations to represent activities of the corresponding phase.

Historically, we can distinguish two very general approaches to the SDLC: the traditional paradigm (combines the structured approach and the information engineering approach) and the object-oriented paradigm (Satzinger et al., 2012). As noted by Dennis et al. (2005), the primary difference between the two paradigms is how the problem is decomposed – in traditional paradigm, the problem decomposition process is either process-centric (the structured approach) or data-centric (the information engineering approach) whereas in the object-oriented paradigm, it is focused on objects that contain both data and processes.

Modern information systems are inherently more complex than, perhaps, any other human construct (Brooks, 1987). Booch et al. (2007) observe that this inherent complexity derives from four elements: the complexity of the problem domain, the difficulty of managing the development process, the flexibility possible through software and the problems of characterizing the behaviour of discrete systems. The object-oriented paradigm is better suitable to address the issues of complexity in software development due to its specific characteristics including classes, objects, methods, messages, encapsulation, information hiding, inheritance, polymorphism and dynamic binding (Dennis et al., 2005). The object approach to systems development follows the iterative and incremental process (Maciaszek, 2007) as the means to cope with the complexity.

Any approach to the SDLC involves a modeling exercise aimed at representing the components of an information system being built in a set of graphical diagrams. For example, a data flow diagram, entity-relationship diagram and structure chart are typical graphical models applied in the traditional paradigm. The key graphical set used in the object-oriented paradigm is the unified modeling language (UML).

2.2 The UML

The UML is a general-purpose visual modeling language that is used to specify, visualize, construct, and document the artifacts of a software system (Rumbaugh et al., 2005). The UML was developed in the mid-1990s with an idea in view to unify the best features of earlier methods and notations. In

November 1997, the UML was adopted by the Object Management Group (OMG), an international consortium that creates and maintains standards for the IT industry (Booch, 2007; Vanderperren et al., 2008).

The goal of UML is to provide a standard notation that can be used by all object-oriented methods and to select and integrate the best elements of precursor notations (Bruegge and Dutoit, 2010). At the same time, the UML is independent of any software development process as long as it supports the object-oriented approach to software production (Maciaszek, 2007).

The UML provides graphical notation to model the artifacts of an information system being built. The complexity of the modern software development cannot be captured by a single diagram. The UML offers different types of diagrams to address all the aspects of the system. These diagrams are classified into structure diagrams and behaviour diagrams. The latter group also includes a subset of interaction diagrams (Figure 2).

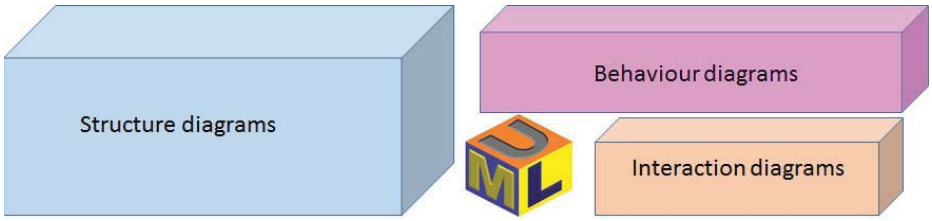

Figure 2. UML taxonomy and official logo.

The most recent version of the language, UML2.x, totals 14 graphical models subdivided into seven structure diagrams, three behaviour diagrams and four interaction diagrams. A summary of the UML diagrams and their characteristics is given in Table 1 (Booch, 2007; Dennis et al., 2005; Fowler, 2004).

Table 1. Types of UML diagrams and their description.

Type of diagram	Description	Diagrams
Structure diagrams	Show static structure of elements in the system; may depict architectural organization of the system, physical elements of the system, its runtime configuration and domain-specific elements	• Package diagram • Class diagram • Component diagram • Composite structure diagram • Deployment diagram • Object diagram • Profile diagram
Behaviour diagrams	Depict dynamic features used to describe the functionality of software systems or to model the functional requirements of an evolving information system	• Activity diagram • State machine diagram • Use case diagram
Interaction diagrams	A subset of behavior diagrams which emphasize the flow of control and data among the components of the system	• Communication diagram • Sequence diagram • Interaction overview diagram • Timing diagrams

In this paper, we present an approach to the software development of environmental information systems for sustainable environmental management on the basis of UML diagrams.

3 RESULT

3.1 Multi-layered logical architecture

It has been widely acknowledged that effective environmental management calls for a systematic integration of knowledge developed across a broad range of fields, such as economics, ecology, psychology, sociology, hydrology and agronomy (Kelly et al., 2013), as well as corresponding models and tools facilitating their interactions (Carnevalle et al., 2012; Jakeman and Letcher, 2003; Voinov and Bousquet, 2010). This creates the complexity of the problem domain and results in the structural complexity of the environmental information systems supporting decision-making in the field.

For many years, multi-layered logical architectures have been used in software development as a means to reduce the complexity by introducing layers of the objects and constraining intercommunications between non-neighbouring layers (Maciaszek, 2007). In addition, the hierarchical structures allow to apply standard solutions (or patterns) in software design, such as high cohesiveness and loose coupling of layers, broker pattern, generator pattern, reactor pattern, façade and mediator patterns, adapter pattern, proxy pattern, factory pattern, singleton pattern, *etc*. (see, e.g., Fox, 2007).

It has been argued that a framework for sustainable environmental management should, at the minimum, consist of five layers (`Ecosystem`, `Monitoring`, `Modeling`, `Valuation` and `Management`) (Khaiter and Erechtchoukova, 2012) as shown in Figure 3.

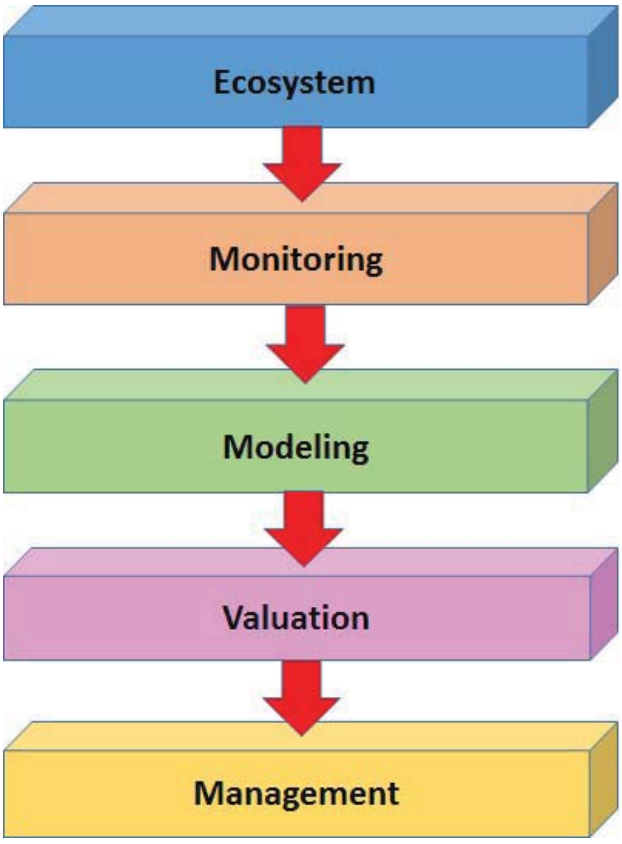

Figure 3. Layers of a framework for sustainable environmental management.

In the subsequent sections, we suggest the steps in software development of EISs and for describing internal contents of each layer on the basis of UML.

3.2 Actors

One of the possible strategies in modeling of software functions with UML is to begin it with identifying the actors, i.e., persons outside the system that interact with the system (Booch, 2007). In UML, actors are not specific users, but rather roles they play. Compared with the projects in business or technological domains, EISs are characterized by unique diversity of stakeholders and actors, including government authorities, researchers, IT specialists, environmental agencies, NGOs, general public and private sector.

For the purpose of the framework for sustainable environmental management, we determine the following categories of actors: `Ecologist`, `Modeler`, `Governmental authority`, `Policy maker`, `Environmental agency`, `NGO`, `Monitoring specialist`, `Lab technician`, `IT specialist`, `DB specialist`, `Environmental economist`.

3.3 Use cases and use case diagrams

A behaviour aspect of an information system is depicted by the use cases. Each use case is a complete piece of functionality, from the actor's point of view, that provides value to the actor (Booch, 2007). Once all the actors are identified, the next logical step in software development is to determine respective use cases for each of them.

A use case diagram is a UML graphical model to associate the actors with their use cases and to document the communication that takes place between the use case and the actor when requesting the use case execution (Dennis et al., 2005). It also shows relationships between the use cases: `<<include>>`, `<<extend>>` and generalization.

We apply the notation of a UML use case diagram to present internal functional steps of each architectural layer. For example, Figure 4 demonstrates internal steps of the `Ecosystem` layer structured within a UML component.

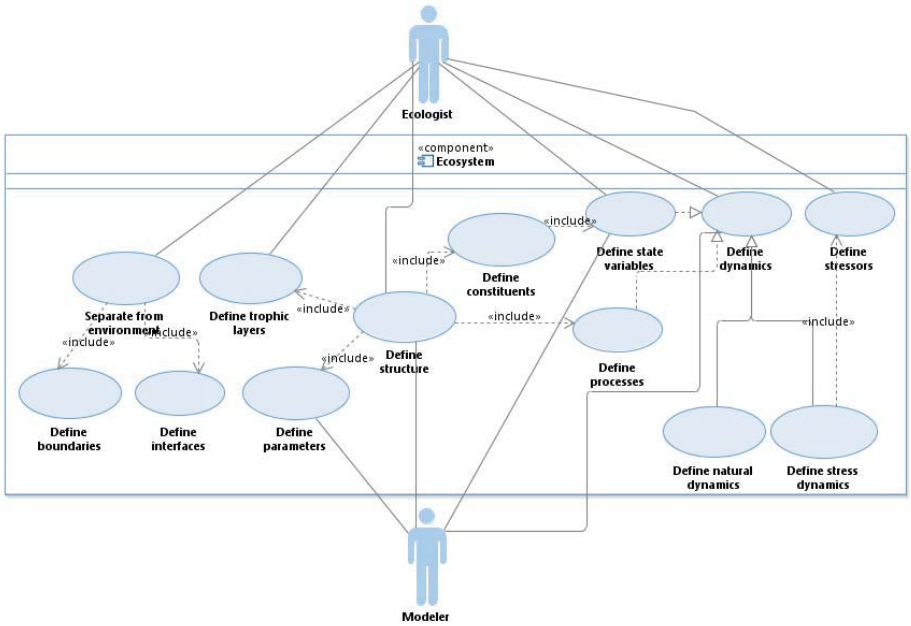

Figure. 4. Internal steps of the `Ecosystem` layer shown as a UML use case diagram.

3.4 Activity diagrams

Activity diagram provide a visual depiction of the flow of activities, whether in a system, business, workflow, or other process. These diagrams focus on the activities that are performed and who (or what) is responsible for the performance of those activities (Booch, 2007). Activity diagrams can be used to model everything from a high-level business workflow that involve many different use cases, to the details of an individual use case, all the way down to the specific details of an individual methods (Dennis et al., 2005).

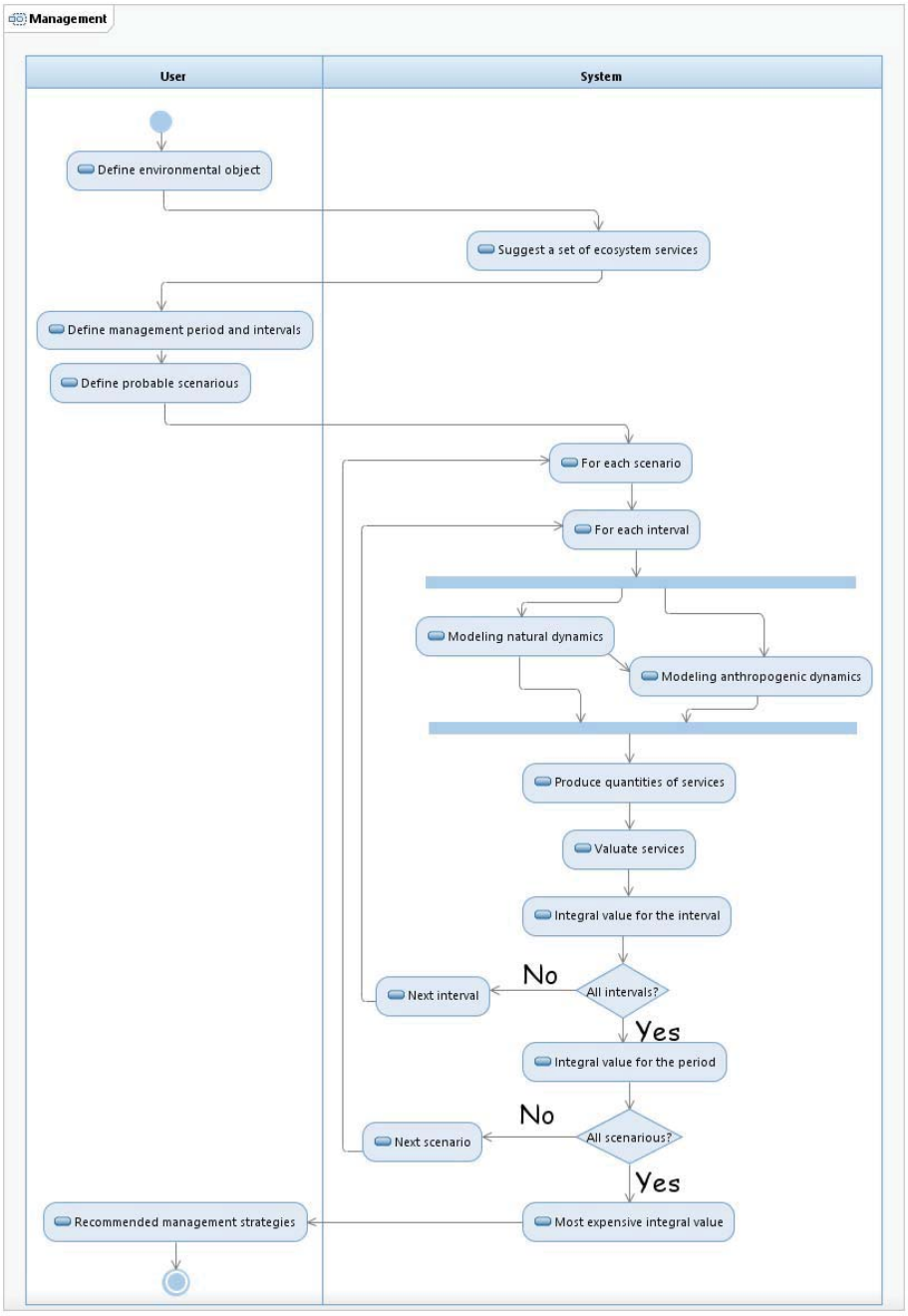

Figure 5. Logic of the `Management` layer shown as a UML activity diagram.

We suggest the interpretation of the tasks of sustainable environmental management as the problems of optimal control or multi-objective optimization. In such a case, the most favourable strategy of development can be chosen from the criterion of maximum net environmental value (MNEV) delivered by the set of complementary ecosystem goods and services. The Management component implements. Figure 5 demonstrates the flow of operations within the Management layer of the framework using the notation of a UML activity diagram.

4 CONCLUSION AND DISCUSSIONS

In this paper, we demonstrated an approach to the development of information systems for the needs of environmental sustainable management. The advantages of multi-layered architectural designs have been discussed. As a key modeling tool, we applied the UML diagrams. The main emphasis in the paper has been made on depicting the dynamic or behaviour aspects of an information system being built.

The logical sequence of steps starts with the identifying the actors followed by the study of their interactions with the system expressed in terms of respective use cases. These modeling components are represented in the notation of a UML use case diagram.

The notation of a UML activity diagram is used to show the overall flow of control and describe procedural logic of the tasks of sustainable environmental management.

The two other categories of the UML models, namely, structure diagrams and interaction diagrams are also important for a successful implementation of an EIS, however the discussion of their role was not within the scope of this paper.

The UML is being used in software development practice of business and technological projects for many years. However, its applications within the problem domain of environmental information systems remain limited and lacking a common methodology. The latter is especially important in view of a growing interest on the part of environmental modeling society towards a broader use of the UML in their projects.

ACKNOWLEDGMENTS

The authors are thankful to the anonymous reviewers for their helpful suggestions and comments on the early version of the manuscript. Part of this work was supported by the Faculty of Liberal Arts and Professional Studies (FLA&PS) through individual grants to each co-author. Personal thanks to Dr. Martin Singer, Dean of FLA&PS, for his support of this project. All UML diagrams have been prepared with IBM® Rational® Software Architect 9.0.

REFERENCES

Booch, G., Maksimchuk, A., Engle, M.W., Young, B.J., Conallen, J., Houston, K.A., 2007. Object-oriented analysis and design with applications, third ed., Pearson, Boston.

Brooks, F.P., 1987. No silver bullet: essence and accidents of software engineering. IEEE Computer 20 (4), 10–19.

Bruegge, B., Dutoit, A.H., 2010. Object-oriented software engineering using UML, patterns, and Java, third ed., Pearson, Upper Saddle River, NJ.

Carnevalle, C., Finzi, G., Pisoni, E., Volta, M., Guariso, G., Gianfreda, R., Maffeis, G., Thunis, P., White, L., Triacchini, G., 2012. An integrated assessment tool to define effective air quality policies at regional scale. Environ. Model. Softw. 38, 306-315.

Dennis, A., Wixom, B.H., Tegarden, D., 2005. Systems analysis and design with UML vesion 2.0, second ed., Wiley, Hoboken, NJ.

Fowler, M., 2004. UML distilled: a brief guide to the standard object modeling language, third ed., Pearson, Boston.

Fox, C., 2007. Introduction to software engineering design: processes, principles, and patterns with UML2. Addison-Wesley, Boston.

Jakeman A.J., Letcher, R.A., 2003. Integrated assessment and modelling: features, principles and examples for catchment management. Environ. Model. Softw. 18 (6), 491-501.

Kelly (Letcher), R.A., Jakeman, A.J., Barreteau, O., Borsuk, M., ElSawah, S., Hamilton, S. H., Henriksen, H.J., Kuikka, S., Maier, H., Rizolli, A.E., van Delden, H., Voinov, A.A., 2013. Selecting among five common modelling approaches for integrated environmental assessment and management. Environ. Model. Softw. 47 (4), 159-181.

Khaiter, P.A., Erechtchoukova, M.G., 2012. Quantitative assessment of natural and anthropogenic factors in forest carbon sequestration. In: Seppelt, R., Voinov, A.A., Lange, S. Bankamp, D. (Eds.), International Congress on Environmental Modelling and Software (iEMSs 2012). International Environmental Modelling and Software Society, Leipzig, Germany, pp. 2075-2082.

Maciaszek, L.A., 2007. Requirements analysis and system design, third ed., Pearson, Harlow, England.

Rumbaugh, J., Jacobson, I., Booch, G., 2005. The Unified Modeling Language reference manual, second ed., Addison-Wesley, Boston.

Satzinger, J.W., Jackson, R.B., Burd, S.D., 2012. Systems analysis and design in a changing world, sixth ed., Course Technology, Boston.

Vanderperren, Y., Mueller, W., Dehaene, W., 2008. UML for electronic systems design: a comprehensive overview. Des Autom. Embed. Syst. 12, 261-292, doi 10.1007/s10617-008-9028-9.

Voinov, A., Bousquet, F., 2010. Modelling with stakeholders. Environ. Model. Softw. 25, 1268-1281.