Jul 1st, 12:00 AM

Classification of Quality Attributes For Software Systems in the Domain of Integrated Environmental Modelling

Naeem Muhammad

Stijn Van Hoey

Piet Seuntjens

Wesley Boenne

Viaene Peter

International Environmental Modelling and Software Society (iEMSs)
2012 International Congress on Environmental Modelling and Software
Managing Resources of a Limited Planet, Sixth Biennial Meeting, Leipzig, Germany
R. Seppelt, A.A. Voinov, S. Lange, D. Bankamp (Eds.)
http://www.iemss.org/society/index.php/iemss-2012-proceedings

Classification of Quality Attributes For Software Systems in the Domain of Integrated Environmental Modelling

Naeem Muhammad[1], Stijn Van Hoey[12], Piet Seuntjens[123], Wesley Boenne[1], Viaene Peter[1]
naeem.muhammad@vito.be[1], stijn.vanhoey@vito.be[12], piet.seuntjens@vito.be[123], wesley.boenne@vito.be[1], peter.viaene@vito.be[1]

1 VITO, Environmental Modelling, Belgium
2 University of Ghent, Belgium
3 Antwerp University, Belgium

Abstract: Quality Attributes (QA) which are also known as non-functional requirements, are characteristics of a software system that define its quality. Performance, availability, usability are typical examples of QAs. Performance for example defines how efficiently a system works with the given resources. QAs' significant impact on the overall quality of a system is widely recognized. They play a critical role in the success of the system, therefore it is imperative to identify the right QAs and validate them to realize a good system. The importance further increases for those software systems which are involved in developing integrated environmental models, mainly because of their complex nature. Such complexity leaves models less trustable by the users. This trust can be retained by developing high quality software systems, which requires a profound understanding about the QAs of the system.
We find that the role of QAs for such systems is not well addressed. This paper attempts to understand that role. We identify a list of QAs which are specific to the integrated environmental modelling. We further discuss how QAs are linked with the environmental models and how they affect the quality of those models. For these purposes, we use hydrological models as a case study.

***Keywords**: Environmental Modelling; Software Quality Attributes; Hydrological Models*

1 INTRODUCTION

The development of a software system starts with collecting its requirements. These requirements can be classified into two categories, functional and non-functional requirements. Functional requirements describe the functional behaviour of the system whereas non-functional requirements express how better a system should work. These non-functional requirements are also known as Quality Attributes (QA) of the system. Some of the commonly used QAs are given in Table 1 [TRUDY 2008]. QAs play a vital role in the success of a software system. Their impact on the success of a software system is widespread [BOSCH and LUNDBERG 2003, ZHANG and GODDARD 2005]. They provide the means for measuring the quality of a system, which according to IEEE Standard 1061-1992 is defined as: "The degree to which software possesses a desired combination of quality attributes" [IEEE 1998].

Table 1 Quality Attributes

Performance	The quality of a system that determines how better or fast a system performs its functions.
Reliability	The ability of a system to perform desired behavior

	under previously specified circumstances, and recover from undesired states if occurred.
Safety	The ability of a system to avoid potential hazards to itself, its users and the environment in which it is used.
Security	The ability of a system to resist any unauthorized use.
Usability	The system must be easy to use, operate and handle.

A better understanding about the QAs of the software systems used in the domain of environmental modelling is of great importance. We find that currently such an understanding needs to be highlighted in a better way. The environmental models are becoming more and more sophisticated, that has raised their level of complexity. Such complexity can reduce the understandability of the models, hence affecting the model trustability [BECK 2011]. This trustability of models can be restored by understanding the importance of QAs and paying the required attention to those QAs while developing software systems. To achieve high quality software systems for environmental modelling, while developing those system focus should be on:

1. Identifying the right QAs of the system.
2. Prioritizing the identified QAs.
3. Evaluating the correctness of the QAs.

To identify a list of right QAs, we first outline a development life cycle of a typical environmental model. Subsequently, we discuss software needs at different phases of the life cycle. To exemplify our discussion we use hydrological models. We also describe some commonly used approaches for prioritizing and evaluating QAs.

In this paper we make the following contributions:

- Identify a list of QAs that are specific to the software systems used in developing environmental models.
- Elaborate the role of QAs in achieving trustable environmental models.

Structure of the paper: In section 2 we describe the development life cycle of environmental modelling. In section 3 we discuss the relationship of QAs with models, by identifying QAs of each phase of the development life cycle. We outline methodologies for prioritizing and evaluating QAs in section 4. Finally, in section 5 we draw conclusions.

2 ENVIRONMENTAL MODELLING DEVELOPMENT LIFE CYCLE PROCESS

An environmental model evolves through various phases during its development. Although there is no standard process specifying the development life cycle of environmental models, we use one proposed by Gupta et al. [2008]. The process is shown in Figure 1. We find that it is a generic process which is applicable to a wide range of models. Following we will discuss different phases of this process and describe the software needs of each phase.

To provide a comprehensive understanding about the above mentioned process, we divide it into the following major phases.

Model Data:
Analysis of the data is always the first step of the modelling phase. It gives the first insight in the system behaviour and helps in identifying the major tasks involved. When applying complex environmental models the amount of data can be large, therefore flexible methods are needed to quickly visualize and compare data to allow interpretation of the available information. Methods to compute derivative information (e.g. peak over threshold information) based on raw data are also preferred.

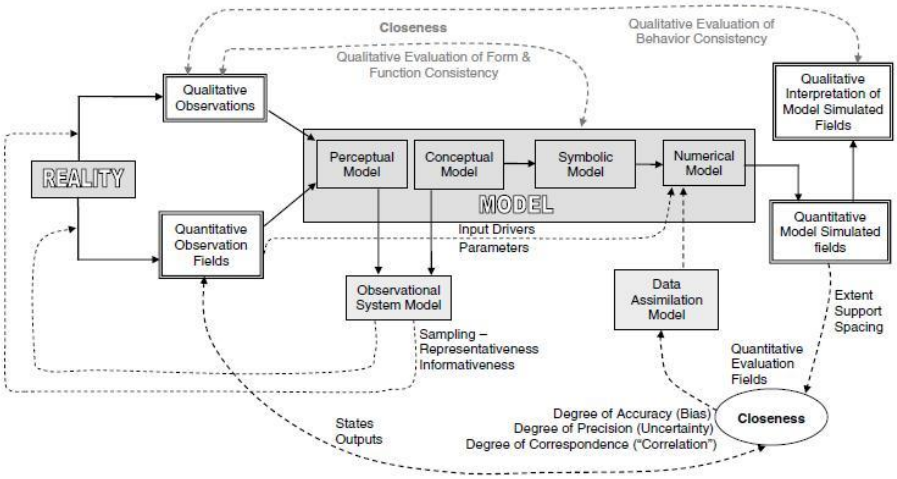

Figure 1 Conceptual description of the model building and evaluation process

Furthermore, data manipulation is mostly needed to convert data into a format that is usable within the modelling itself. Facile import from a wide range of data formats is necessary together with the methods to convert the data in the preferred or needed format. Compatibility with different open-source standards is also getting necessary.

Since failure of measurements equipment is possible, missing values and errors in data are frequently observed, therefore appropriate methods to deal with such a situation are required to ensure correct data handling.

Typical kinds of data used in environmental models are time series of variables and geographical data (maps, design schemes) with spatial attributes (raster format or vector format), where also these spatial attributes can have varying values in time. Environmental software systems need to be flexible to handle these different formats.

Perceptual and Conceptual Model:

The perceptual model phase is the first translation of the data into an understanding of what is important in the model. When this is more clearly specified and the relevant inputs, system boundaries, state variables and outputs are defined, this perceptual model is translated in a conceptual model representing the current state of knowledge about the working of the system.

Since most of the environmental software systems already represent a set of conceptual models with associated assumptions and conceptualisations of the system, a clear communication of the adopted concepts is essential for the user. This enables the user to make the right decisions and to evaluate if the adopted assumptions are in correspondence to his personal perceptions.

Symbolic/Numeric Model:

The translation of the conceptual model into a mathematical set of equations is typically done by building a numerical model approximation, solving a set of Ordinary Differential Equations (ODE) or Partial Differential Equations (PDE), representing the processes involved. The ability of solving the set of equations by numerical approximation is also subject to approximations. Clear communication about the consequences of the numerical implementation is preferred.

Since the used solver techniques are determining the stability and speed of the solution, the ability to choose between different solution schemes helps in optimizing the model application. Furthermore it can also influence the capacity of further model analysis [Kavetski and Clark 2011].

Model analysis:

Model analysis is a very important step in the model development, since it represents the phase where the implementation is tested and evaluated. In most model applications major attention is given to the model analysis part, since it helps in identifying shortcomings of the conceptual model (modelling as hypothesis testing, [Fenicia et al. 2008] or numerical model.

The evaluation of the numerical and mathematical calculation, also referred as model verification in comparison to model validation, is required to be automated.

Besides, a typical problem in current applications is the indistinctness of error messages when failures occur.
Inverse modelling also known as calibration or parameter optimization has vital importance in model analysis and the evaluation of conceptual models. In inverse modelling certain model parameters need to be estimated based on available data, since parameters are not always directly related with measurable data [Beven and Binley 1992]. A facile handling in confronting measured values with simulation outputs both visually and by means of performance criteria is required to facilitate this. Furthermore, the easy extraction of model outputs at intermediate steps is essential to do a clear evaluation of the model outcomes.
Since models are always a simplified version of the reality, simplifications made to describe the system under study are inherently allied to the model implementation. Uncertainty and sensitivity analyses are a necessity for model evaluation are increasingly applied. Most of these uncertainty and sensitivity analyses are requiring a large number of model evaluations. Software that gives the possibility to maximise the use of computer resources or cluster infrastructure is required.

3 QUALITY ATTRIBUTES AND THE MODEL

Table 2 contains a list of all the requirements that we identified in section 2. This list can be extended, but is limited here to support the proof of concept of deriving and prioritizing QA. To clarify the different requirements, examples relevant for distributed hydrological models are included, although providing a complete set of requirements is without the scope of this paper.

Table 2: Overview of software requirements and corresponding QA

	Requirements	**QA**	**Distributed Hydrologic model example**
Model Data	Easiness of exchange with different external data standards and formats with changing temporal and spatial properties.	Compatibility	GIS shape files and vector formats delivering input data.
	Dealing with anomalies and missing values in data.	Usability	Different rainfall gauges giving data of different time periods.
	Converting data internally to the proper spatial and temporal resolution.	Flexibility	Rainfall data at point scale towards spatial input for catchment.
	Quickly visualize and compare data.	Usability	Stream flow data in different sections of the river.
	Computation of derivative information out of ‘raw’ data.	Flexibility	Flow duration curves (FDC).
Perceptual/Conceptual model	Clear communication of the adopted conceptualisations and assumptions.	Transparency	Kinematic wave approximation for overland flow.
	Adapt or extend the model structure.	Extensibility	Adding case-specific interception model component.
Symbolic/ Numeric Model	Information about the numerical approximations.	Transparency	Connection groundwater and soil water.
	Decide about optimal numerical	Flexibility	Implicit or explicit solver.

	solver.		
Model Analysis	Easy extraction of intermediate information.	Accessibility	Measured and modelled groundwater levels at different locations in the catchment
	Confronting measured values with simulation outputs.	Usability	Nash-sutcliff efficiency to evaluate model prediction [Nash and Sutcliff 1970]
General	Maximise the use of computer resources or cluster infrastructure.	Performance	
	Clear descriptions of errors and failures, and recovery from them.	Reliability	

Model Data

Rainfall is a driving force in hydrological processes and rainfall data is an essential input for hydrological models. Rainfall data can be both point measurements (coming from rain gauges) and spatial information (radar measurements). The first is mostly delivered in time series formats together with the GPS coordinates of the gauge, the latter in GIS format, a stack of raster maps. Compatibility of the software means providing the functionalities of reading in the range of data standards existing in the GIS scenery to minimize external pre-processing by the user.

Since rainfall gauges can break down, long-term measurements mostly have lacking or erroneous periods in the data. When one gauge has missing values, data from nearby gauges can be used to fill in the measurement gaps. Other strategies are possible and software systems need to provide support for such manipulations.

After importing the data into the software, functionalities are needed to manipulate the rainfall data internally to the proper spatial and temporal resolution for the specific modelling exercise without losing information. When only rainfall data coming from discrete points in space is available and the model needs spatial inputs, the conversion from point to area average inputs is needed. Another example is the changing temporal scale between different model exercises (hourly, daily, monthly) with required adaptation of the time series.

Stream flow is another required source of information in hydrological modelling and mainly used to calibrate the models. When comparing the hydrographs in different sections of the river, a first impression of potential areas where the water is flowing over the river banks, during storm events can be obtained (flattened peak flow periods). Extra information can be picked up by calculating derivative information like Flow Duration Curves (FDC), showing the percentage of time flow is above a certain value.

Perceptual/conceptual model

The routing of water over land can be modelled by a range of different conceptualizations. The Saint-Venant equations, derived from the Navier-stokes equations can be simplified itself to the diffusive wave and kinematic wave and the applicability of each dependent from the specific conditions. The conditions of these approximations must be satisfied. Clear communication towards the user is needed in order to avoid irrelevant model structural choices.

Interception is the process of rainfall water that is captured by vegetation. The implementation is typically very empirical. Extending or editing this implementation could be beneficial for the model performance.

Symbolic/Numeric Model

When solving the system of model equations in a distributed hydrological model, the connection between groundwater and soil (subsurface) water is not straightforward. Different applications are using different implementations to relate both components [Ewen et al. 2000, Panday and Huyakorn 2004] with related consequences on model behaviour. Clear explanation towards the user of is essential to correctly interpret the model outcome which is of increasing importance when things appear irrational.

Model Analysis

Most hydrological models are based on some kind of water balance stating the different loss and gain. This is also a major step in verifying the model implementation, since a diversion from this balance is directly related errors in the implementation or by numerical instability. Since the system boundaries working with are part of an open system, mass balances alone are no guarantee for correctness. However, software support can help in achieving such correctness.
The inverse modelling process to estimate specific parameters can be supported by different types of data, groundwater levels can be used to check if the groundwater component is estimating reasonable behaviour. Software systems need to provide support to extract different output information easily and to compare it with the measured data.

General

An optimal use of computer resources is essential when working with complex distributed hydrological models. Certainly when extra model evaluations are required for optimization, sensitivity or uncertainty analyses, calculation time should be reduced to a minimum. Moreover, the compatibility with cluster computing technology is of increasing importance.

4 PRIORITIZING AND EVALUATING QUALITY ATTRIBUTES

A system should work correctly according to its functional requirements. But, it must also achieve its quality goals set in the form of QAs [Svensson et al. 2011, Bass 2006, Svensson et al. 2010]. A system needs to be evaluated in order to know if it has achieved its required quality. Failing to meet the quality goals generally leads to the failure of the system. Determining what, when and how to evaluate a system's quality is a difficult task [Yang et al. 2009]. Possible risks to QAs should be identified and fixed at early stages in the software development life cycle. Any unresolved issue with QAs may lead to a major refactoring, increased cost and delays.

Architectural decisions are among the earliest made in the development life cycle. A high quality system can be built by preventing errors with those decisions. It has long been acknowledged that architecture level analysis of QAs is a cost-effective approach [Clements et al. 2002, Malek et al. 2007]. To perform such an analysis, well defined techniques are required to prioritize and evaluate QAs for a software system [France et al. 2007]

Considering the importance of QAs, it is vital to ensure up-front, that right architectural choices have been made to meet the required quality of the QAs [Taylor et al. 2007]. Therefore, the architecture phase of the development life cycle of software systems is the most suitable place for evaluating the quality of QAs, which is a proven cost-effective approach [Malek et al. 2007]. Architecture level analysis helps in assessing the quality of the system before its real implementation.

According to Bass et al [2003] "The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them."

A structure of a software system represents its static and dynamic arrangements of architectural elements. The former portrays design-time arrangement whereas the later defines runtime arrangement. The most important aspect of software architecture is concerned with properties of the system, which can be categorized as externally visible properties and quality properties [Rozanski et al. 2005]. The first category deals with interactions between the system and its environment. Whereas the second category represents QAs.

The main purpose of documenting an architecture of a software system has shifted, from merely a communication artefact, to its use for system validation. An architecture can be analyzed for prioritizing and evaluating the correctness of QAs of a software system. Performing such analyses has become a cost-effective approach [Galster et al. 2008]. It helps in developing a system conforming precisely to its QAs. There are many architecture level analysis methods available.

Hereunder we list a few commonly used methods [Clements et al. 2002, Dobrica and Niemela 2002, Babar and Gorton 2004]:

The Architecture Trade-Off Analysis Method (ATAM) [Kazman et al. 1998]: The main purpose of ATAM is to understand the correct priorities of QAs for a system by performing a trade-off among them.

Scenario-Based Architecture Analysis Method (SAAM) [Kazman et al. 1994]: SAAM is used to analyze an architecture for modifiability Quality Attribute (QA). In addition, it also provides support for analyzing other QAs such as maintainability and flexibility.

Software Architecture Analysis Method for Evolution and Reusability (SAAMER) [Horing et al. 1997]: SAAMER is mainly used to assess an architecture from the point of view of evolution and reusability of the system.

Cost Benefit Analysis Method (CBAM) [Ionita et al. 2003]: This method can be used to analyze a system for cost benefit analysis and to understand potential risks to the project.

5 CONCLUSIONS

In this paper we highlighted the role of QAs in achieving high quality environmental models. For this purpose, we first outline a development life cycle of environmental modelling, followed by a description of the software needs of each phase of the life cycle. We also discuss few methods that can be used to prioritize and evaluate QAs to develop software systems fulfilling the right needs of environmental modelling. We use hydrological models to exemplify our discussion.

Now a days, software systems are considered a fundamental part of the environmental modelling. However, we find that the role of software systems in achieving high quality models is not recognized yet. A quality software system, with right and accurate QAs can lead to a good model.

A wide range of evaluation methods are available, which can be used to evaluate QAs while developing software systems for environmental modeling. In addition, a considerable attention should be given to prepare a prioritized list of QAs to focus on the quality aspects of the system that are actually need.

We have outlined a preliminary list of QAs for software systems used in the domain of environmental modeling. Although we only used hydrological models for this purpose, we believe that the list contains QAs that are applicable for software systems used for other environmental models. As a future work we plan to identify a similar kind of QA list for environmental models from other fields.

REFERENCES

Beck, MB. Grand challenges for environmental modeling. Environmental Modeling and Software. 25(4), 611–612, 2010.

Bosch, J., Lundberg, L., Software architecture – engineering quality attributes. Journal of Systems and Software, 66(3), 183-186, 2003.

Trudy, S., Quality attributes for embedded systems. In Advances in Computer and Information Sciences and Engineering, T. Sobh, Ed. Springer Netherlands, 536–539, 2008.

Zhang, S., Goddard, S., XSADL an architecture description language to specify component-based systems. In International Conference on Information Technology: Coding and Computing (ITCC 2005), vol. 2, 443 – 448, 2005.

IEEE. IEEE standard for a software quality metrics methodology. Tech. Rep. 1061-1998, IEEE, 1998.

Gupta, H.V., Wagener, T. and Liu, Y. Reconciling theory with observations: Elements of a diagnostic approach to model evaluation. Hydrological Processes, 22(18), 3802-3813, 2008.

Beven, K. and A.M. Binley. The Future of Distributed Models: Model calibration and uncertainty prediction. Hydrological Processes 6:279-298, 1992.

Ewen, B.J., G. Parkin, P. Enda, and P.E. O'Connell. Shetran: Distributed river b asin f low modeling system. Journal of Hydrologic Engineering, 2000.

Fenicia, F., H.H.G. Savenije, P. Matgen, and L. Pfister. Understanding catchment behavior through stepwise model concept improvement. Water Resources Research 44:13, 2008.

Kavetski, D. and M.P. Clark. Numerical troubles in conceptual hydrology: approximations, absurdities and impact on hypothesis testing. Hydrological Processes 25:661-670, 2011.

Nash, J.E. and J. Sutcliffe. River flow forecasting through conceptual models part i — a discussion of principles. Journal of Hydrology 10:282-290, 1970.

Panday, S. and P.S. Huyakorn. A Fully coupled physically-based spatially-distributed model for evaluating surface / subsurface flow. Advances in Water Resources 27:361-382, 2004.

Svensson, R. B., Gorschek, T., Regnell, B., Torkar, R., Shahrokni, A., and Feldt, R. Quality requirements in industrial practice; an extended interview study at eleven companies. IEEE Transactions on Software Engineering 99, PrePrints, 2011.

Svensson, R. B., Host, M., and Regnell, B. Managing quality requirements: A systematic review. Software Engineering and Advanced Applications, Euromicro Conference , 261–268, 2010.

Bass, L. Principles for designing software architecture to achieve quality attribute requirements. In Proceedings of the Fourth International Conference on Software Engineering Research, Management and Applications (Washington, DC, USA), IEEE Computer Society, p. 2, 2006.

Yang, J., Huang, G., Zhu, W., Cui, X., and Mei, H. Quality attribute trade off through adaptive architectures at runtime. Journal of Sys. and Soft.82,2,319–332, 2009.

Malek, S., Edwards, G., Brun, Y., Tajalli, H., Garcia, J., Krka, I., Medvidovic, N., Mikic-Rakic, M., and S., S. G. An architecture driven software mobility framework. J. Syst. Softw. 83, 972–989, 2010.

Clements, P., Kazman, R., and Klein, M. Evaluating software architecture: methods and case studies. Addison-Wesley Longman Publishing Co., Inc., 2002.

France, R., and Rumpe, B. Model-driven development of complex software: A research roadmap. In 2007, Future of Software Engineering (Washington, DC, USA), FOSE '07, IEEE Computer Society, pp. 37–54, 2007.

Taylor, R. N., and van der Hoek, A. Software design and architecture the once and future focus of software engineering. In 2007 Future of Software Engineering (Washington, DC, USA, 2007), FOSE '07, IEEE Computer Society, pp. 226–243.

Bass, L., Clements, P., and Kazman, R. Software Architecture in Practice, second ed. SEI Series in Software Engineering. Addison-Wesley Professional, 2003.

Rozanski, N., and Woods, E. Software systems architecture: working with stakeholders using viewpoints and perspectives. Addison-Wesley Professional, Apr. 2005.

Galster, M., Eberlein, A., and Moussavi, M. Early assessment of software architecture qualities. In Second International Conference on Research Challenges in Information Science, 2008.

Dobrica, L., and Niemela, E. A survey on software architecture analysis methods. Software Engineering, IEEE Transactions on 28, 7, 2002.

Babar, M. A., and Gorton, I. Comparison of scenario based software architecture evaluation methods. In 11th asia pacific software engineering conference, 600, p. 607, 2004.

Kazman, R., Bass, L., Webb, M., and Abowd, G. Saam: a method for analyzing the properties of software architectures. In Proceedings of the 16th international conference on Software engineering (Los Alamitos, CA, USA), IEEE Computer Society Press, pp. 81–90, 1994.

Kazman, R., Klein, M., Barbacci, M., Longstaff, T., Lipson, H., and Carriere, J. The architecture tradeoff analysis method. In International Conference on Engineering of Complex Computer Systems, pp. 68–78, 1998.

Ionita, M., Hammer, D., and Obbink, H. Scenario based software architecture evaluation methods an overview. In Workshop on Methods and Techniques for Software Architecture Review and Assessment at the International Conference on Software Engineering, 2002.

Horng Lung, C., Bot, S., Kalaichelvan, K., and Kazman, R. An approach to software architecture analysis for evolution and reusability. In Proceedings of CASCON 97, pp. 144–154. 1997.