Educational Technology & Society 5(2) 2002
ISSN 1436-4522

How to Annotate Educational Multimedia with Non-Functional Requirements

Giovanna Avellis
TECNOPOLIS CSATA
Human Resources Innovation and Training
Str. Prov. Per Casamassima Km.3,70100, Valenzano, Bari, Italy
Tel: +39-080-4670 374
g.avellis@tno.it

Anthony Finkelstein
University College London
Dept. of Computer Science
Gower Street, London WC1E 6BT, UK
Tel: + 44 (0) 20 7679 7293
a.finkelstein@cs.ucl.ac.uk

ABSTRACT

We develop a scheme for representing critical non-functional requirements (NFRs), and apply it to the domain of multimedia educational software (MES) to validate it. Our approach extends the model for representing design rationale by making explicit evaluation goals, providing the means to improve the quality of MES. Further research issues will be discussed including the need to relate NFRs to the architectures and a set of architectures to an application domain.

Keywords

Architecture, Design rationale, Educational software, Multimedia, Non-functional requirements

Introduction

It is widely recognised that non-functional requirements (NFRs) are crucial in software development and that different architectural choices can have a substantial impact on the quality or service provided by the resulting system (Arango & Prieto-Diaz, 1991; Avellis, 2000; Devanbu et al., 1991). However, there is a gap in the way current software development methods build and keep track of the links between requirements, especially NFRs, and architectures in constructing and evolving complex systems.

We provide an explicit map or links or traceability relationships between the NFRs and multimedia educational software (MES) systems and use the map, respectively

- to reason about the value of a MES system; and
- to incrementally evaluate the NFRs during software development.

In this paper, we focus on analysis and reasoning about the process of building a value model of a software system, by explicitly representing NFRs. The techniques and representations in the paper are then demonstrated using an application from the domain. MES systems represent a broad class of software systems with complex characteristics that tend to make evaluation difficult. The educational potential of multimedia, both as a learning and teaching tool, is widely acknowledged, and various initiatives undertaken encourage the integration of educational multimedia resources in school practice (Avellis & Capurso, 1999a).

This paper defines the background and discusses the main issues of MES evaluation and the problem of annotating NFRs to MES. Section 2 describes the context of the problem. Section 3 identifies the features of the software domain and points out the needs in evaluating MESs. The evaluation criteria developed in the framework of ERMES (EuRopean Multimedia Educational Software network) ESPRIT project (Avellis & Ulloa, 1997) are discussed. Section 4 introduces the annotation scheme for representing NFRs. It introduces the selected scheme of the NFR representation, which is a process-oriented as opposed to a product-oriented representation. The conclusions identifies further research issues in building links between NFRs and architectures.

Context of the Problem

Main developments in software engineering have been centred on the functional and object-oriented perspective. The functional viewpoint is not the only design dictum in engineering. A refutation of the design dictum "Form follows function" (Petroski, 1994) applies to software systems as well as any complex systems. The functionality of the system offers an explicit level of representation of system capabilities and the object-oriented representations provide a suitable basis for understanding the application concepts as the represented objects can be easily mapped with the real world objects.

The object-oriented perspective provides the means to localise the effect of functional changes in system architecture. It also restricts the impact and propagation of changes such that the changes in an aspect of the system are mapped to the changes to other aspects of the system (Avellis, 1992; Avellis et al., 1991).

By 'aspect' or 'view' of a system we mean a set of abstractions that provides us with one of (many possible) characterisations of a software system. A 'model of view' captures the semantics used by that view (Avellis, 1990; Avellis & Borzacchini, 1992). In the literature on reverse engineering, a 'view' is often a structural view that contains information about the structure of the product. For instance, the Software Re-engineering Environment (SRE) of CSTaR-Arthur Andersen (Kozaczynsky & Ning, 1989) stops at the level of identifying generic programming plans well before identifying application-specific knowledge. One of the consequences of not having application-specific views of the system is that the maintainer has to compute on his own a complex mapping between the description of change and the part of the system to be changed, being the majority of changes expressed in terms of the vocabulary of application domains (Arango & Prieto-Diaz, 1991). The understandability issue (Corbi, 1989) - the problem to grasp the relationships among different views of a software system and their interconnections - relates to our built-in limitations, as humans to deal with large-scale complex objects. The phenomenon of "invisibility" of a software system (i.e., the structure of software, unlike those of buildings or automobiles, is hidden and the only external evidence we have of software is related to its behaviour) has been highly emphasised for many systems in literature (Devanbu et al., 1991).

There is a need to develop richer models for capturing and analysing NFRs in Software Engineering. However this is not a simple enterprise.

Examples of difficult tasks include:

- choosing an architecture to satisfy some NFRs;
- evaluating the impact of a change of NFRs in the system structure;
- modifying the architecture; and,
- evaluating NFRs during system development.

Our research aims at mapping the relationships between NFRs and architectures in order to analyse the impact of changing NFRs on the architecture.

Some open issues concern the understanding of how prioritisation and evolution of NFRs impact the requirement traceability problem and software architectural choices. Requirements traceability (Finkelstein, 1991) refers to the ability to describe and follow the life of a requirement, both forwards and backwards. A lack of common definition of requirements traceability (purpose-driven, vs. solution-driven, vs. information-driven, vs. direction-driven) has been detected where requirements traceability problem was perceived not to be uniform due to diverse definitions and a number of fundamental conflicts have been found.

The need for improved requirements specification traceability is evident from the literature (Harandi & Ning, 1988). NFRs have yet to be incorporated in the core of specification, design, implementation techniques and tools. So progress in this regard had been limited.

Evaluation of Multimedia Educational Software

Many national and international activities in MES are currently partially funded by the European Commission, involving private and public sector organisations (Avellis & Fresa, 1999). In this context, the need for educational multimedia for vocational training purposes is widely recognised. However, users of educational multimedia cannot appraise educational resources because they are not able to evaluate their characteristics, potentialities, and limits (Avellis & Capurso, 1999a).

The reason why it is not so easy to carry out a critical evaluation of educational multimedia lies in the problem that these resources are relatively new compared to traditional print-based learning materials. Most people are still not used to handling them, nor they are aware of the educational potential. Finally, educational multimedia also has an intrinsic complexity for it encompasses two aspects: it is a software running on a computer and an educational resource. Evaluating both these aspects is very different from evaluating, for example, a book or any traditional educational resource.

The distinction between being software and supporting learning is blurred because the way the application runs affects educational effectiveness, but educational purpose underlies the design of the software. Therefore, both aspects must be carefully considered during the evaluation. However, it is very difficult to develop a pre-defined set of standards against which the educational value of the software can be determined, because it is not possible to define a unique and general instructional approach. The educational value of a piece of software is something very difficult to define in practice (Avellis & Capurso, 1999b). The evaluation methodology adopted in the ESPRIT project ERMES (Avellis & Ulloa, 1997) consists of identifying aspects of the object, and then defining the quality indicators in relation to these aspects. Defining the object of evaluation is a key step, because it suggests evaluation criteria. We group the characteristics of MES under these four evaluation categories:

- educational features of the software;
- technical features;
- aspects relating to the ease of use (usability); and,
- aspects relating to the content.

Each one of these categories has been further grouped into sub-categories. For example, educational features are divided into target users, pedagogical characteristics, instructional support materials, and so on. That means that when evaluating the educational features of a multimedia product, the aspects relating to the target users, the pedagogical characteristics, the instructional support materials, and so on have to be taken into account.

MES is a computer program, which performs a specific educational task. The multimedia component can be identified in the use of a variety of media to deliver instruction or support the learning activities. MES is also characterised by the presence of interactive components, which should enable the user to control the learning environment.

Features of MES includes (Avellis & Capurso, 1999a):

- the *content* which is to be taught;
- the *delivery media* used to provide information;
- the *user interface*, that is the way the educational software presents itself to the user;
- the *interaction* devices, by which the user interacts with the computer, making choices, answering questions or performing activities, and is provided with *feedback* to each response; and,
- the *instructional strategy* adopted.

A Scheme for Critical NFR Representation

There is a need to develop techniques to express NFRs, which include quality requirements (Finkelstein, 1994). We place at the centre of the development process *the generation of a value model* such as in classical engineering disciplines (Finkelstein & Finkelstein, 1983). That is, a key component of the system development process is achieving a model of *what is valued* in the resulting system. In this view, quality characteristics are not externally imposed on a development process but constructed within it.

The scheme developed to express NFRs is based on the work done by (Kunz & Rittel, 1970), particularly in the area of design rationale.

We also take into account (Conklin & Begeman, 1988) the *issue-position-arguments* model, where in our scheme an 'issue', that is a problem to solve, is a *NFR or quality characteristics/sub-characteristics to evaluate*, an 'argument', that is a supporting justification is a *procedure* which helps to determine which design alternative to choose to implement the related non-functional requirements. Finally, 'position', that is a solution to the problem, is either a *statement* of NFR, which gives a quality goal to be supported by the final design, or *design alternatives*. Statement is an ascertainable property (possibly measurable characterising a NFR). The set of links is given in the figure 1.

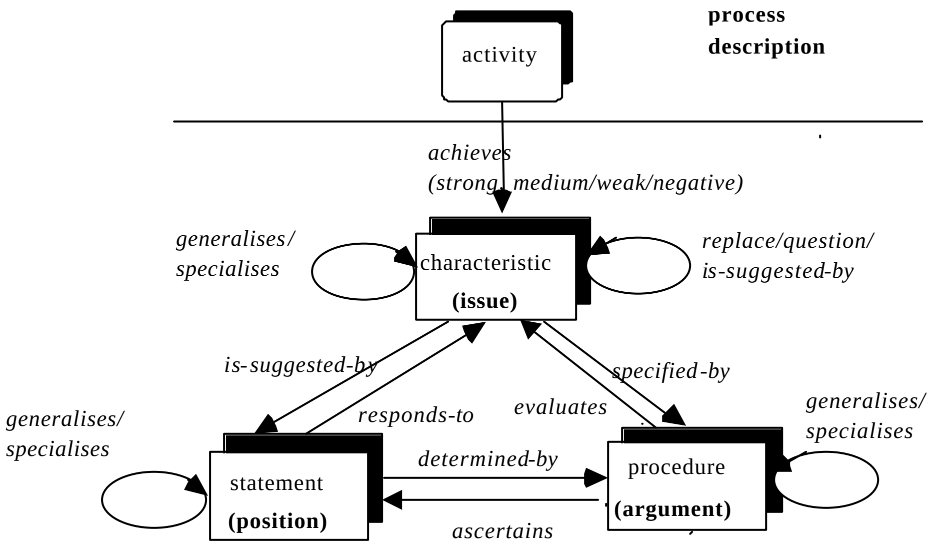

Figure 1. Non-functional Requirements Representation scheme

It is important to underline that the statement contains measurable elements by which non-functional requirements can be constructed in the software system by a procedure which applies to different architectural choices.

We enhance the representation of NFR with Quality Function Deployment (QFD) features, a new systematic method of design-oriented approach to assure that the customer needs drive the product design and production process since the late 1960's. The method is called Quality Deployment and/or Quality Function Deployment (QD/QFD). We enhance the scheme of NFR representation above by introducing the following.

In order to be assured that we will achieve a particular software quality characteristic, it is helpful to associate it with some activities within the software evaluation and development process. *Activity* is the evaluation and/or implementation activity of the quality characteristic, which provides the context of evaluation. A quality characteristic obtained as a result of performing an activity is *strong/medium/weak/negatively*.

In QFD style, we attach weights - strong/medium/weak/negative - to this link, in order to let the end users (teacher, trainers, students, administrators) to assign a weighted value to the characteristic of the system under evaluation.

Although a quality characteristic can be constructed independently of the description of the development process of a product, it is useful to link product and process descriptions to the quality characteristics. Avellis (2000) provides insights on how to relate this process view to a product view, by introducing the role played by the architecture of a software system and relating it to the NFRs.

One application of the scheme is that a NFR related to MES can be that *the MES package should fit the subject/topics and learning objectives of my course*. The activity related to this example is to *evaluate the educational aim of the MES package*, which satisfies the *educational features* evaluation criterion.

The e*ducational features* criterion has several sub-characteristics to be taken into account, such as instructional characteristics, which is suggested by the requirement statement *appropriateness of learning objectives suitable for age and competence of target users* and is measured by the procedure *verify that content and learning objectives are consistent with the national curricula requirements.*

Another application example is the NFR is that *the MES package should be easy to operate*. The activity related to the third example is *understanding the usage of a MES package*, which achieves in *medium* form the quality characteristics of "usability". This in turn, can be further specialised into the sub-characteristics 'ease of use', which is suggested by the requirement statement *the way software operates* and several procedures to measure usability, including these questions: What are the IT skills required to operate the software? Is on-screen help available? Are directions clear and accurate? Are directions available at all times? Is management of assessment instruments easy?

Conclusions and Further Research

In this paper, we presented work in progress towards the design and development of the *Educational Multimedia Evaluation Tool* based on the framework of ESPRIT project ERMES. The key issue is how to improve the current ERMES Evaluation tool by a scheme to annotate Non-functional Requirements (NFRs) to Educational Multimedia. Further research is needed in this context, such as how to annotate NFRs to architectures.

A system quality attribute (i.e., NFR) is largely permitted or precluded by its architecture . The motivation for software architecture is to have a basis for understanding and standardise systems and their components.

Software has yet to achieve the level of reuse realised by hardware disciplines. Although software is easy to reproduce, software variations are much more difficult to standardise, identify and control. While a universal reuse solution remains elusive, great improvement have been made by focusing on well-defined areas of knowledge or activity (domains) (Arango & Prieto-Diaz, 1991). Architectures provide a means for structuring knowledge of the system within a domain, including requirements. The possibilities for reuse are greatest where specifications are least constrained at the architectural level. Reuse is normally considered only at the implementation phase. This practice limits reuse to fine-grained modules at best and fails to allow for broader utilisation of assets at a subsystem or higher level, by neglecting to plan at early stages of development.

In this paper, we focussed on setting a process in which we can argument on the quality of MES on the basis of identified NFRs and developed some case studies to critically evaluate the process. A follow-up research result is the development of an evaluation tool to help MES users (teachers, trainers, education specialists, school administrators) to choose educational software, of high quality, suitable to their needs and valuable as educational resource to integrate in their own courses/current curriculum based on the selection of NFRs.

Acknowledgements

This work has been partly financially supported by the European Commission under Human Capital and Mobility scheme, and the ESPRIT projects and MACS (Maintenance Assistance Capabilities for Software).

References

Arango, G., & Prieto-Diaz, R. (1991). *Domain analysis and software system modelling*. Los Alamitos, CA: IEEE Computer Society Press.

Avellis, G. (1990). MACS ESPRIT project and the software evolution expert system. *Annual Conference of Associazione Italiana per l'Informatica e il Calcolo Automatico, 1*, 3-7.

Avellis, G. (1992). CASE support for software evolution: A dependency approach to control the change process. In Forte, G., Madhavji, N., & Muller, H.A. (Eds.), *Proceedings of the Fifth International Workshop on Computer-Aided Software Engineering* (26-73). Montreal: IEEE CS Press.

Avellis, G. (2000). *Annotating multimedia educational software with Non-functional Requirements – An approach to evaluate educational multimedia*. [Internal Technical Report]. London, UK: University College - London.

Avellis, G., & Borzacchini, L.(1994). A blackboard model to design integrated intelligent software maintenance environment. In *Proceedings of the Fourth International Conference on Software Engineering and Knowledge Engineering* (pp. 325-332). Capri, Italy: IEEE CS Press.

Avellis, G., & Capurso, M. (1999a). ERMES evaluation methodology to support teachers in skills development. In Tuomi, E., Salonen, M., Saarinen P., & Sinko, M. (Eds.). *Proceedings of IFIP WG3.1 and 3.5 Open Conference on Communications and Networking in Education: Learning in a Networking society* (pp. 12-19). Hameenlinna, Finland: Painopaikka Yliopistopaino.

Avellis, G., & Capurso, M. (1999b). ERMES services for education. In Roger, J., Stanford-Smidth, B., & Kidd, P.T. (Eds). *Business and Work in the Information Society: New Technologies and Applications* (pp. 520-527). Stockholm: IOS Press.

Avellis, G., & Fresa, A. (1999), Education and training: A challenge between industries and cultures. In Roger, J., Stanford-Smidth, B., & Kidd, P. T. (Eds). *Business and Work in the Information Society: New Technologies and Applications* (pp. 401-407). Stockholm: IOS Press.

Avellis, G., Iacobbe, A., Palmisano, D., Semeraro, G., & Tinelli, C. (1991). An analysis of incremental capabilities of a software evolution expert system. In *Proceedings of Conference on Software Maintenance-1991* (pp. 220-227). Sorrento: IEEE CS Press.

Avellis, G., & Ulloa, A. S. (1997). *ESPRIT 24111 Project* [ERMES Technical Annex]. Bari, Italy : ERMES Consortium, TECNOPOLIS CSATA.

Conklin, J., & Begeman, M. L. (1988). gIBIS: A hypertext tool for exploratory policy discussion, *ACM Transactions on Office Information Systems*, *6* (4), 303-331.

Corbi, T., (1989). Program understanding: challenge for the 1990s. *IBM System Journal*, 28 (2).

Devanbu, P., Brachman, R., Selfridge, P., & Ballard, B. (1991). LaSSIE: A knowledge-based software information system. *Communications of the ACM*, *34* (5), 34-49.

Finkelstein, A. (1991). Tracing back from requirements. In IEEE Colloquium on Tools and Techniques for Maintaining Traceability During Design, Computing and Control Division, Professional Group C1, Digest No. 1991/180. London: IEEE.

Finkelstein, A. (Ed.) (2000). *The Future of Software Engineering*. ACM Press.

Finkelstein, A., & Finkelstein, L. (1995). Review of design methodology. In Collen, A., & Gasparski, W. (Eds.), *Design and Systems Praxiology*, The International Annual of Practical Philosophy and Methodology, Vol.3 (pp. 95-122). New Brunswick, NJ: Transaction Publishers.

Harandi, M., & Ning, J. (1988). PAT: A knowledge-based program analysis tool. In *Proceedings of Conference on Software Maintenance 88* (pp. 312-319). Phoenix: IEEE CS Press.

Kunz, W., & Rittel, H.(1970). Issues as elements of information systems. *Technical report S-78-2, Institut fur Grundlagen der Plammung*. Stuttgart, Germany: Universitat Stuttgart.

Petroski, E. (1994). Design paradigms: Case histories of error and judgement in engineering, Cambridge, UK: Cambridge University Press,.