Evaluating Non Functional Requirements in Mobile Learning Contents and Multimedia Educational Software

Gianna Avellis, Antonio Scaramuzzi
TECNOPOLIS CSATA
SP.per Casamassima Km.3 70100
Valenzano/Bari/Italy
Email:g.avellis@tno.it,
a.scaramuzzi@tno.it

Anthony Finkelstein
University College London, Dept. Of Computer Science
Gower Street, London WC1E 6BT, UK
Email: a. finkelstein@cs.ucl.ac.uk

Abstract

We developed a scheme for representing critical non functional requirements (NFRs), and apply it to the domains of mobile e-learning contents and Multimedia Educational Software 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 e-learning contents, especially m-learning contents. 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: Non Functional Requirements (NFRs), Mobile Learning Contents (MLC), Multimedia Educational Software (MES).

1. Introduction

It is widely recognised that non functional requirements (NFRs) are crucial in software development and that different architectural choices can have different impact on the quality of the final system (Arango & Prieto-Diaz, 1991), (Devanbu, Brachman et al., 1991), (Avellis, 2000). 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 identifying explicit links between the NFRs and m-learning systems and use the map, respectively:

- to reason on the "value" of the 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 by using an application from the domain. M-learning systems represent a broad class of software systems with complex characteristics that tend to make evaluation difficult. The educational potential of m-learning contents, 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 m-learning contents evaluation and the problem of annotating NFRs to them. Section 2 describes the context of the problem. Section 3 identifies the features of the software domain and points out the needs in evaluating m-learning contents. 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 to represent 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.

2. Background

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. Main developments in software engineering have been centred on the functional and object-oriented perspective, mainly because 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.

This perspective has been pursued for so long as one of the main gain is to provide 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, Iacobbe 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), i.e. 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, Brachman & al., 1991).

There is a need to develop richer models for capturing and analysing NFRs in Software Engineering. However this 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.

One open problem of our research is to map NFRs to architectures in order to analyse the impact of changing NFRs in the architecture.

Other 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 in (Gotel & Finkelstein, 1996) 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 literatures (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.

3. Evaluation of Mobile Learning Contents and Multimedia Educational Software

Mobile e-Learning is so new that we are only beginning to see the potential of mobile devices in training and performance support.

Mobile devices are small, portable and compact. They can often fit in a pocket or purse. Unlike laptop computers, which are expensive, heavy and power hungry, mobile devices are relatively low-cost, lightweight, and some work a very long time on a charge or a couple of standard disposable or rechargeable batteries.
The small screen size of mobile devices (a Non Functional Requirements) makes some people questions their worth as e-Learning delivery tools. The truth is, some of these devices also have good audio capability, allowing students to listen to a narrated lecture, rather than read material on a small screen. Some critics also point to the restricted input capabilities (another NFR) of some of these devices, questioning students' ability to enter large amounts of text into a device to take notes or answer an essay-type question. Many of these devices, however, are extremely adaptable (a NFR) and can be attached to a full-size folding keyboard that makes entering large amounts of information every bit as fast (a NFR) as with a conventional computer.

3.1 Mobile e-learning in Practice

Mobile e-Learning is in its infancy. Although many experts in the field see a great potential in using mobile devices in e-Learning, there are presently few implementations on which to base a study of best practices.
With mobile e-Learning in its infancy, and some mobile devices similar in functionality to conventional computers, it's only natural that the first generation of mobile e-Learning content will closely resemble conventional e-Learning, only presented on a smaller screen.
As mobile devices evolve and people discover new ways in which mobile devices functionalities can be applied to training, mobile e-Learning will likely become something increasingly different from conventional e-Learning. Mobile e-Learning will no longer be a miniaturised version of conventional e-Learning. Internet phones may be applied to mentoring and they may be used to register students in courses and pay fees, as well as present training content through audio.
Content development tools may appear that will provide the ability to publish content adaptively to a wide range of mobile devices. In addition, the student may well have control over reading or listening to the content using voice-synthesised XML technologies.
Since mobile e-Learning is so young, there are presently more possibilities to what can be done with this technology than concrete examples. But, with the number of mobile devices predicted to surpass the number of conventional computers for Web access in the near future, and with bandwidth for mobile devices slated to increase dramatically in the short term, mobile e-Learning is certain to become an important part of training.

Many national and international activities in Mobile Learning Contents (MLC) and in Multimedia Educational Software (MES) in general, are currently partially funded by the European Commission, involving private and public sector organisations (Avellis & Fresa, 1999). In this context, the need of 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).

3.2 Evaluation issues

The reason why it is not so easy to carry out a critical evaluation of mobile 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 has also 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 those when 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 defined, because it is not possible to define a unique and general instructional approach. The mobile 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 in 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 (Ulloa, 1998). 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 MLC or MES, 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. Multimedia educational software is also characterised by the presence of interactive components, which should enable the user to control the learning environment.

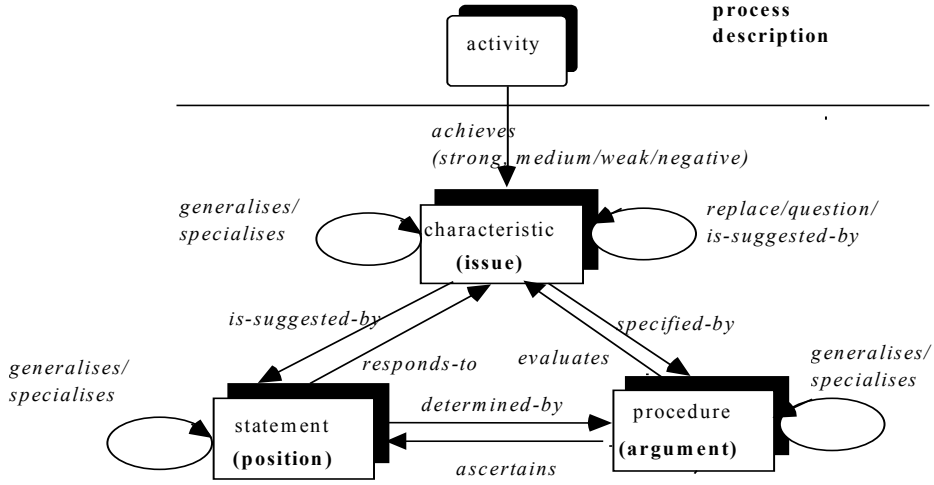
Features of mobile multimedia educational software includes (Avellis & Capurso, 1999a), (UWA Consortium, 2002) :

- 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
- *access* which refers to the navigational paths available to the user to reach the needed content
- *navigation* allowing the user to go from one piece of content to another
- *presentation* which can provides guidelines for defining the visual communication strategies or presenting the content, navigation strategies and operation to the user
- *user operation* are those operations that are visible to the users and the only ones the user must be aware of
- *system operation* that are not visible to the users, but are essential in building user operation

4. A Scheme for Critical NFR Representation

There is a need to develop techniques to express NFRs, which include quality requirements (Finkelstein, 1994). This underlines 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 (Potts & Bruns, 1988). We also take into account (Conklin & Begeman, 1988) "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 following:

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.

(Mizuno & Akao, 1978) have established 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 developed 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 characteristics 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 characteristics is *strong/medium/weak/negatively* obtained as a result of performing an activity.

In a Quality-Function-Deployment (QFD) style we attach some 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 characteristics 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. In (Avellis, 2000), we provide 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.

In the following we give some examples of the application of the scheme above to MLC and MES.

A NFR related to MLC can be "*the MLC should fit the subject/topics and learning objectives of my course*".

The activity related to this example is "*evaluate the educational aim of the MLC package*", which *strongly* achieves the quality characteristics "*educational features*" .

"*Educational features*" quality characteristics 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*".

The last example is the NFR "*the MES package should be easy to operate*".

The activity related to the second 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 "*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?"*.

5. Conclusions and Further Research .

In this paper, we presented work in progress towards the design and development of the *Evaluation Tool* based on the framework of ESPRIT project ERMES for MLC and MES in general. 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 MLC and 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 not only MES users but also MLC users 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.

A further aim is to research and demonstrate innovative mobile contents for addressing training in IT and education sectors and to evaluate the requirements, especially non functional requirements, of e-learning modules for mobile applications and services. The wireless e-Learning solution will focus on mobile learning objects representation which can suit to mobile delivery media and to methodologies for adapting Multimedia Educational Software to m-Learning environments.

Constructing, evaluating and evolving wireless and e-Learning contents to meet users' requirements is one of the flaws of current e-learning and mobile learning systems. Key users requirements are Non Functional Requirements (NFRs). Functional requirements set out services expected by the system user. NFRs, on the contrary, set out constraints on the system and the product and process standards to be followed. As such, they play a central role in evaluating the quality of wireless and e-Learning modules.

Software Quality is attracting more and more attention in Software Engineering for two reasons: on the technical side, it is usually not clear to those involved in the development how to measure the various quality criteria on a day-to-day basis (i.e. formative analysis), nor how to achieve them and measure it on completion (summative analysis). On the customer's side, the issue is simply not knowing what to ask for. To this end, a distinction has been underlined between Basic Quality Factors, such as functionality, reliability, ease of use, economy, safety, and Extra Quality Factors, such as flexibility, repairability, adaptability, understandability, documentation, enhanceability. These latter are quality factors related to the external, or observable quality of a piece of software and are particularly important in the world of eLearning where technical strategies are

emerging in parallel with educational and pedagogical strategies, as well as in the framework of mLearning where the constraints of mobile devices and the supported software are very important for delivering effective contents, in addition to Mobile Quality Factors identified so far, such as accessibility, navigation, presentation, and system user operation.

However, it is important to grasp the internal quality of a system. Ultimately, the external quality of a system depends upon the internal quality. For example, the enhanceability of a system is directly related to how well structured the internal design is, i.e. the size, definition and relationships between modules and subsystems. Internal quality factors includes completeness, consistency, parsimony, traceability, rationality, structure, paradigm, and quality of algorithms and representations, as well as understandability and documentation.

The nature of these factors is not well understood.

This is why we propose to investigate some research issues on how to evaluate quality factors in wireless and e-Learning modules, and apply it to several domains/scenarios to validate the scheme.

This will result in an integrated set of m-learning training modules and an analysis and assessment of evaluation criteria to grasp their requirements for advanced mobile and wireless technologies. To this end, it will collaborate with current standardization working groups, especially the evaluation and assessment of non functional requirements of m-learning and e-learning modules.

Actually, the current industry standards (AICC, IMS, IEEE/LTSC, etc.) have already addressed the problem of metadata tagging of educational resources to allow an easier access and retrieval through e-learning systems.

Further improvements in standardisation could be achieved by extending to the NFRs (e.g. target delivery device) the set of characters currently adopted to describe and classify learning modules. This will result in an increased capability of the user to assess the suitability of a selected educational material for a specific application environment (e.g. mobile learning).

Acknowledgements

This work has been partly financially supported by the European Commission under Human Capital and Mobility scheme, and the ESPRIT projects ERMES (EuRopean Multimedia Educational Software) and MACS (Maintenance Assistance Capabilities for Software) (Avellis, 1990), and finally the ETI action MOBITECH (European SMEs Challenge in Mobile Communication Technologies).

6. References

Arango, G. & Prieto-Diaz, R. (1991). *Domain Analysis and Software System Modelling*, California: IEEE CS Press.

Avellis, G. (1990). MACS ESPRIT project and the Software Evolution Expert System, In *Annual Conference of Associazione Italiana per l'Informatica e il Calcolo Automatico*, Bari: Laterza Publishing cCompany, (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*, Montreal: IEEE CS Press, 26-73.

Avellis, G. (2000). Annotating Multimedia Educational Software with Non Functional Requirements – An Approach to Evaluate Educational Multimedia, *UCL Internal Report.*

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*, Capri: IEEE CS Press, 325-332.

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,* Hameenlinna:12-19.

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,* Stockholm: IOS Press, 520-527.

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, Stockholm: IOS Press, 401-407.

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*, Sorrento: IEEE CS Press, 220-227.

Avellis, G. & Ulloa, A.S. (1997). ERMES Technical Annex, *ESPRIT 24111 Project*, ERMES Consortium (Eds.).

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 *IEE Colloquium on Tools and Techniques for Maintaining Traceability During Design, Computing and Control Division, Professional Group C1, Digest No. 1991/180*, London: IEE.

Finkelstein, A. (1994). Quality Arguments: Product and Process Interaction, In *Proceedings of Software Quality Management '94*.

Finkelstein, A & Finkelstein, L. (1983). Review of Design Methodology, *IEE*, 212-222.

Gotel, O. & Finkelstein, A., (1996). An Analysis of the Requirements Traceability Problem, In Arnold, R. & Bohner, S. (Eds.), *Software Change Impact Analysis, IEEE CS Press*.

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

Kozaczynsky, W & Ning, J. (1989). SRE: A Knowledge Based Environment for Large Scale Software Re-engineering Activities, *Communications of the ACM.*

Kunz, W. & Rittel, H.(1970). Issues as Elements of Information Systems, *Technical Report S-78-2, Institut fur Grundlagen der Plammung*, Universitat Stuttgart,.

Mizuno, S. & Akao, Y.(1978). Quality Function Deployment, *Nikkagiren-Syuppan*, Tokio.

Petroski, E. (1994). Design Paradigms: Case Histories of Error and Judgement in Engineering, Press Syndacate of the Univ. of Cambridge (Eds.).

Potts, C. & Bruns, G. (1988). Recording the Reasons for Design Decisions, In *Proceedings of the Tenth International Conference on Software Engineering*, IEEE CS Press (Eds.).

Ulloa, A. (1998). ERMES Evaluation Guidelines for MES, ERMES ESPRIT Project Del.5.1, Valenzano, Bari, Italy.

UWA Consortium (2002). The UWA Approach to Modeling Ubiquitous Web Applications, In Proceedings of Conference "IST Mobile & Wireless Telecommunications Summit 2002, owards Ubiquitous Communications", Thessaloniki, Greece.