A Requirements Analysis Framework for Human Activity Systems (HAS): The Case of Online Learning

PHILIP O AYOO and JUDE T LUBEGA*
Faculty of Computing and Information Technology
Makerere University

Abstract

The task of designing information systems is clearly interdisciplinary, since it requires domain knowledge in business process development within the social environment, and the processes management of technological applications. This paper explains e-learning as a human activity system, which requires soft methodologies that deal with the analysis of evolving and ill-defined needs, as well as traditional hard approaches to the design of physical solutions to meet those needs. Consequently, a requirements analysis framework is proposed for constructivist online learning systems in which soft systems thinking is integrated as the essential strategy of requirements elicitation and analysis. This framework is capable of capturing both the formal and informal, as well as the hard and soft aspects of the requirements within a social environment.

Keywords: systems thinking, information systems, requirements analysis, soft systems methodology, human activity systems, e-learning requirements analysis.

IJCIR Reference Format:
Philip O. Ayoo and Jude T. Lubega. A Requirements Analysis Framework for Human Activity Systems (HAS): the Case of Online Learning. International Journal of Computing and ICT Research, Special Issue Vol. 3, No. 1, pp. 12-25. http://www.ijcir.org/Special-Issuevolume3-numbe1/article2.pdf.

1. INTRODUCTION

Requirements drive almost every activity, task, and deliverable in a software development project [McEwen, 2004], since accurate requirements form the most essential part of the formula for successful software product development. On surveys of failed projects, the one cause of failure that shows up most often is having missed or poorly defined requirements. Consequently, poor requirements are the biggest problem that end up increasing the budgeted engineering cost and not matching with the expected quality. Without good quality of specifications the people charged with developments will have no firm idea of the needs of the would-be users of the systems (Liu, 1992).

According to Rose [2002], the software engineering approach to information systems development (ISD), in association with structured methods, and the more recent generation of object-oriented development methods, remain the most significant influence on contemporary systems development. Many other influences (prototyping, RAD, extreme programming) share the same primarily technical standpoint. However, given the nature of the IS discipline, a technical standpoint is reasonable and inevitable, but many commentators have pointed out that it tends to exclude or minimise many other important factors:

* Author's Address: Philip O. Ayoo and Jude T. Lubega, Faculty of Computing and Information Technology, Makerere University, P.O. Box 7062, Kampala, Uganda payoo@iucea.org and jlubega@cit.mak.ac.ug

International Journal of Computing and ICT Research, ISSN 1818-1139 (Print), ISSN 1996-1065 (Online), Special Issue Vol.3, No.1 pp.12-25, October 2009

social, cultural, political, semantic, managerial and so on [Rose, 2002]. For this reason, a lot of research is currently on-going to better understand the social aspects of IS development.

2. PURPOSE OF REQUIREMENTS ANALYSIS

Systematic requirements analysis is also known as *requirements engineering*, and is sometimes referred to loosely by names such as *requirements gathering*, *requirements capture*, or *requirements specification* [McConnell, 1996]. The term can also be applied specifically to the analysis proper (as opposed to elicitation or documentation of the requirements).

A requirement describes what a system should do. According to *Technology Blueprint*, requirements are enumerated specifications that list characteristics that identify the accomplishment levels needed to achieve specific objectives for a given set of conditions. Requirements include business goals, objectives, processes, and all other business and system requirements whose purpose is to alter the "*as is*" view of the world in some way [Bleistein, *et al.*, 2006]. A requirement is a vital tool to help clarify client needs, the business objectives, and methods required to fulfilling that need. However, requirements must be actionable, measurable, testable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design.

Some people separate "requirements analysis" from "analysis" [The Object Agency, 1995]. Within this categorization, "requirements analysis" involves the establishment of system and/or project requirements, and "analysis" (e.g. structured analysis and object-oriented analysis) involves the proposal of a solution (going no deeper than the user interface) that satisfies the established requirements. According to *Merriam-Webster Dictionary*, analysis is the separation of a thing into the parts or elements of which it is composed; it is the examination of a thing to determine its parts or elements, including a statement showing the results of such an examination. Analysis often begins with the recognition of a need/want, and goes no deeper (in detail) than the "user interface" of the delivered product [The Object Agency, 1995]. Within this perspective, therefore, requirements analysis is the investigation of a problem that focuses on what functionality is required but not on how to provide that functionality. Beyond this (i.e. providing functionality) is design, which starts at the "user interface," and defines the internal software architecture for the delivered product.

Sudhakar [2005] identifies the types of requirements needed for capturing all aspects of software. They are:

1. *Business Vision and Business Requirements*: These are described as the "why" requirements. They capture the high level objectives of the organization or customer requesting the system or the product. They can be considered as "why we are doing this" requirements and should capture the fundamental reason for product existence.
2. *Functional Requirements*: Functional requirements can be considered as the "what" requirements. These capture the functionality that must be built into the system to satisfy business requirements. Use-cases are a great way to document functional requirements.
3. *Non-Functional Requirements*: They are the "how" or "how well" requirements. They capture the technology specific and "*-ility*" requirements of the system like compatibility, usability, availability, performance, reliability, etc.
4. *UI Prototypes*: UI prototyping helps in reducing risk, decreasing system size and complexity, and reducing requirements change. The UI prototypes (or screenshots) are the most effective way of communicating requirements to the development team, as they provide a great visualization tool for both development and test engineers to understand the exact software flow.

The requirements analysis for the construction of requirements specifications is the key activity for successful developments of systems [Liu, 1992; Wiegers, 2003; Vat, 2005]. Without good quality of specifications the people charged with developments will have no firm idea of the needs of the would-be users of the systems. Naik [2004] observes that getting the requirements right in the first place costs 50 to 200 times less than correcting code. According to this observation, each hour spent on review avoids an average of 33 hours of maintenance. Ideally, therefore, any project over two weeks duration should necessarily include the requirements analysis process [Naik, 2004].

The ultimate aim of understanding end-users and their requirements is to develop a product that is both useful and usable, the combination of which determine the usability of a system [Löwgren 1995, cited in Neale *et al*., 2007]. According to the International Organisation for Standardisation (ISO), usability is the extent to which a product can be used by specified users to achieve specified goals with effectiveness,

efficiency and satisfaction in a specified context of use. Indeed, users have been shown to be more satisfied with system design arising from thorough requirements analysis [Neale *et al*., 2007], and lack of user input has been cited as one of the major reasons for the failure of system development (Standish Group, 1994). To prevent this disaster, the clients have to be involved in the whole process of the developments of systems for consultation by the developers [Liu, 1992].

3. CLASSIFYING AND DOCUMENTING REQUIREMENTS

Requirements analysis can be a long and arduous process during which many delicate psychological skills are involved [Wikipedia]. Conceptually, requirements analysis includes three types of activity:

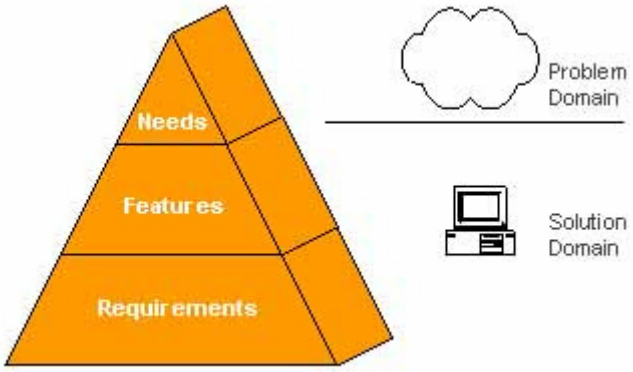
1. *Eliciting requirements*: the task of communicating with customers and users to determine what their requirements are. This is sometimes also called requirements gathering;
2. *Analyzing requirements*: determining whether the stated requirements are unclear, incomplete, ambiguous, or contradictory, and then resolving these issues;
3. *Recording requirements*: Requirements may be documented in various forms, such as natural-language documents, use cases, user stories, or process specifications.

As observed by McEwen [2004], requirements are not requirements unless they are written down. Sudhakar [2005] states that the most critical factor of successful software product development is the right requirements documentation. Taking the time at the beginning of a project to define and document needs, features, and requirements enables one to establish traceability to ensure that software requirements specification aligns with business objectives and continues to do so throughout the project [McEwen, 2004]. Poor requirements are the biggest problem that end up increasing the budgeted engineering cost and not matching with the expected quality.

According to Pohl [1993] (cited in Williams and Kennedy, 2006), the term *requirements engineering* is used to describe a systematic process of developing requirements through an iterative co-operative process of analysing the problem, documenting the resulting observations in a variety of representation formats, and checking the accuracy of the understanding gained. McEwen [2004] explains three separate documents within which requirements are usually captured (as shown in **figure 1**):

1. Stakeholder needs
2. Software features
3. Software requirements specification

Figure 1: Requirements Categories [McEwen, 2004]



Stakeholder Needs

Stakeholder needs, which are part of the problem domain, describe what stakeholders require for a successful project. In other words, *needs describe what the application should do to help improve or lower the cost of a business process, increase revenue, or meet regulatory or other obligations*. Documenting stakeholder needs involves identifying, understanding, and representing different viewpoints. Often, users and stakeholders do not know how to solve the entire problem but are experts at explaining what they need to do their job better. Each stakeholder sees the problem from a different perspective. Therefore, one must understand the needs of *all* stakeholders in order to understand the entire problem domain.

Software Features

After defining stakeholder needs, they must be translated into a set of distinct system features. Needs do not indicate a particular solution; they simply describe the business need. It is perfectly fine for stakeholders to express themselves in any way they wish; often, an analyst will want to ask additional questions to clearly understand both needs and features. Needs are part of the *problem domain*, and features are part of the *solution domain*. It is critically important to fully understand the problem domain before deciding on a solution; often, an analyst will find opportunities to generalize the solution once they fully understand the problem.

Software Requirements Specification

A requirement is a software capability that must be met or possessed by a system or a system component to satisfy a contract, standard, or desired feature. Requirements may be classified into functional and non-functional requirements. Functional requirements present a *complete* description of how the system will function from the user's perspective. They should allow both business stakeholders and technical people to walk through the system and see every aspect of how it should work before it is built. Non-functional requirements, in contrast, dictate properties and impose constraints on the project or system. They specify *attributes* of the system, rather than what the system will do.

4. TRADITIONAL REQUIREMENTS ANALYSIS TECHNIQUES

During requirements analysis energy should be directed towards ensuring that the final system or product conforms to client needs rather than attempting to mould user expectations to fit the requirements [Liu, 1992]. A trained software practitioner called the "requirements analyst" communicates with knowledgeable user(s) to understand what the requirements are. Analysts can employ several techniques to elicit the requirements from the customer. Historically, this has included such things as holding interviews, or holding focus groups (sometimes referred to as requirements workshops) and creating requirements lists [Wikipedia]. More modern techniques include prototyping, and use-cases. Where necessary, the analyst will employ a combination of these methods to establish the exact requirements of the stakeholders, so that a system that meets the business needs is produced.

Noting that the quality of requirements specifications directly affects the quality of the developed systems, Liu [1992] and Urquhart [2001] outline some of the more commonly used techniques in requirements analysis:

4.1 Structured System Analysis Methodology

Two different sorts of methods for system analysis have been developed under the name of structured system analysis methodology: *informal methods* and *formal methods*. The specifications produced using *informal methods* (such as Structured Analysis Method) are comprehensible due to the notations such as diagrams, graphics and natural language, etc, but lack the formality. *Formal methods* try to overcome this weakness by adopting mathematical notations. The specifications produced using formal methods are concise and precise in general, but lack comprehensibility. Therefore, there is a great need for developing a language and a method for producing comprehensible and precise requirements specifications.

4.2 DeMarco's Approach

Tom DeMarco first proposed the model of data flow diagrams to be a method and language for requirements analysis. A data flow diagram shows the connections between processes, data flows and files. DeMarco's data flow diagram provides the hierarchy mechanism to fit into complex requirements analysis. The hierarchical structure in data flow diagrams is constructed by decomposing each upper level bubble into a sub data flow diagram which reflects the details of how to transform their input data flows into their output data flows. The function of each bottom level process is usually specified using structured English. In addition to the data flow diagrams, a data dictionary is usually designed for defining data flows, components of data flows, files and processes occurring in the data flow diagrams.

4.3 JSD

JSD, short for Jackson Software Development, is a systematic method for specifying and implementing computer systems, especially for information systems. JSD starts with the description of the model of the real world and then specifies the functions of the system to be developed based on the model of the real world. The implementation step is therefore centrally concerned with transforming the specification to make it convenient to execute. To be clear and understandable, JSD provides a series of diagrams for the development steps.

4.4 SSADM

Structured Systems Analysis and Design Methodology (SSADM) breaks down the work into phases, which are then divided into stages. The three phases of SSADM are: feasibility, analysis, and design. The feasibility phase consists of two stages: problem definition and project definition. These stages assess the scale of the problem and the costs and likelihood of improving the situation. The analysis phase includes three stages: analysis of the current system, specification of the required system, and select service level for new system. The design phase consists of three stages: detailed data design, detailed process design, and physical design control. Initially design is done at a logical level, and then converted to a physical design. In a SSADM project, data flow diagrams (DFDs) are used to describe the existing physical system, the logical equivalent of the existing system, and the required system, while logical data structuring technique (LDST) and entity life histories (ELHs) diagrams are employed to describe the data requirements.

4.5 Prototyping

In the mid-1980s, *prototyping* was seen as the solution to the requirements analysis problem. Prototypes are mock-ups of an application that allow users to visualize an application that has not yet been constructed. The technology of rapid prototyping presents an effective approach of portraying functionality. But its power is limited as rapid prototypes are produced based on the informal requirements specifications of the proposed systems. The possible misunderstanding of the specifications during the rapid prototyping often affects the quality of the rapid prototypes and therefore a lot of modifications for the rapid prototypes are required in order to fit clients' true requirements. Eventually the rapid prototyping is not rapid and the cost is not so low.

4.6 Use Cases

The "use case" is introduced as a more formal method for requirements capture and initial planning. Use cases typically avoid technical jargon, preferring instead the language of the end user or *domain expert*. According to McEwen [2004], organizations are rapidly adopting use cases as a means to communicate requirements because they:

1. are easier to create, read, and understand than traditional functional specifications;
2. show how the system will work from the users' perspective rather than the system's perspective;
3. force us to think about the end-game (what is the user trying to accomplish by using the system?);
4. require us to define how the system should work, step-by-step;
5. Provide an excellent basis for building test cases and helping to ensure that these are built before the code is written;
6. provide a common requirements "language" that is easy for stakeholders, users, analysts, architects, programmers, and testers to understand.

Fundamentally, use cases provide a way of depicting and structuring the interaction of a real world Actor (person, organisation or external system) with the application being studied. Use cases do not describe any internal workings of the system, nor do they explain how that system will be implemented, but simply show the steps that a user follows to perform a task. Use cases are often co-authored by requirements engineers and stakeholders.

4.7 Object-Oriented Requirements Analysis

Object-oriented requirements are requirements where the information (content) is localized around, and expressed in terms of, objects and the interactions and interrelationships among these objects [The Object Agency, 1995]. Object-oriented requirements analysis is a process whereby two important activities occur: one, we study user (customer) wants/needs to arrive at an object-oriented definition of the implicit or explicit criteria that must, should, or might be met by any delivered solution to these wants/needs; and then we define project needs, priorities, and constraints (preferably in an object-oriented manner), and define an object-oriented solution to the user (customer) wants/needs that, preferably, goes no deeper in detail than the user interface(s).

Object-oriented analysis aims to overcome some of the problems of the structured methodologies (such as SSADM), which can become a rigid straitjacket on systems development [Urquhart, 2001]. The high degree of consistency within object-oriented approaches, coupled with the recursive/parallel life-cycle, greatly blurs the distinction between what has been traditionally thought of as "analysis," and what has been traditionally thought of as "design."

4.8 CRC Cards

One similar structured approach to user involvement in the process of object-oriented development is the use of Class, Responsibility and Collaboration (CRC) cards. The technique uses small teams, including users who are "domain experts", with the aim of discovering the objects, the classes that the system comprises, their responsibilities (what they do) and their collaboration, how they interact with other objects

in the system. Brainstorming is used to derive provisional classes and behaviour and role-playing is then used to work through some scenarios, "living the system". The technique is aimed at a better, deeper understanding of the problem situation so that the information system developed provides a proper solution.

5. PROBLEMS WITH TRADITIONAL RA TECHNIQUES

The software crisis of the 1960s embodied a growing realisation that systems took too long to build, cost too much and did not work too well (Lycett, 1999). According to a survey by the *Standish Group* [McEwen, 2004], of more than 352 companies reporting on more than 8,000 software projects: over 40% to 60% of defects in software are caused by poor requirements definition; 31% of all software projects are cancelled before they are completed (a waste of $81 billion); customers do not use 20% of their product features; while 53% of projects cost 189% of their original estimate. The survey also revealed that in large companies, only 9% of projects are on time and within budget; in small companies, 16% of projects are on time and within budget; while the average time overrun is 222% of the original estimate [Williams and Kennedy, 2006].

According to Williams and Kennedy [2006], systems failure has been blamed on poor requirements engineering process, resulting to poor understanding of domain knowledge and poor use of methods, techniques and tools. Most requirements analysis writings either advise general guidelines which are short of any concrete operational aspects, or they are not simply at the scale of complex open systems construction [Bastani, 2007]. The Standish Group survey also asked respondents to identify the causes of these failures. **Table 1** below shows the top three reasons why projects are "impaired."

Project impairment factors	% of Response
Lack of user input	12.8%
Incomplete requirements and specifications	12.3%
Changing requirements and specifications	11.8%

Table 1: Top Three Project Impairment Factors (Standish Group)

As this table shows, poor requirements are the biggest problem. McEwen [2004] observes:

> If it isn't clear what you are supposed to build, how can you estimate the cost of building it? How can you create a project plan, assign resources, design system components, or create work orders? You need accurate requirements to perform these activities. Of course, requirements evolve as a project proceeds, but carefully worded basic requirements provide a starting point. Then, as the project progresses, you can fill in details and update planning documents as the requirements evolve.

In their study, Williams and Kennedy [2006] established that errors in information systems have the following distribution: incomplete requirements 56%, design 27%, coding 7% and other 10%. The high error percentage due to incomplete requirements is attributed to the poor methods used to elicit and analyse requirements. Further evidence that errors due to incomplete requirements analysis take a disproportionate larger effort share has been provided: incomplete requirements 82%, design 13%, coding 1% and others 4% (Williams and Kennedy, 2006).

In *Software Project Survival Guide,* Steve McConnell [McConnell, 1996] comments that “fixing upstream defects downstream can cost 50 to 200 times as much as fixing them upstream”. This is compelling motivation for taking the time to plan, and is a pretty logical statement when one considers that, during requirements analysis (an upstream activity), a feature is described using a handful of words. Once that handful of words makes it to code (a downstream activity), it can be hundreds or thousands of lines of program code.

Although substantial progress has been made in terms of methods, techniques and tools used within the requirements engineering (RE) phase of systems development, little attention has been paid to the understanding of the RE process effectiveness itself [Williams and Kennedy, 2006]. The designers of information systems (IS) and programmers often begin designing and programming the incumbent system too early, before they actually understand the users’ or stakeholders’ requirements.

In recognition of this defective methodology, the Garmisch Conference [1968] recommended the application of a systematic, disciplined and quantifiable approach to the development, operation and maintenance of software (the discipline of software engineering). This provided a somewhat technical

orientation to information systems development, which also later demonstrated the inadequacy of computer science to address socially based problems associated with the use of computers in an organization or business context [Lycett, 1999]). For the future system to be effective it has to have a balance between the technical worldview of designers and programmers, and the social worldview of users and customers [Urquhart, 2001; Castro, *et al.*, 2002; Williams and Kennedy, 2006].

Checkland and Holwell [1998] argue that information systems are more than "data manipulation" systems and that the study of the human activity systems is vital to an understanding of what an effective information system might look like. A human activity system (HAS) is an assembly of people and other resources organized into a whole in order to accomplish a purpose [Nardi, 1996; Banathy, undated]. The people in the system are affected by being in the system and, by their participation in the system, they affect the system. According to Vat [2005], IS developers must therefore understand the intertwined regularities and idiosyncrasies of human activities and create systems that support the right moves and degrees of agility at the right times and places and for specific purposes.

6. SYSTEMS THINKING

The design of IS must therefore be examined more closely from a systems perspective. Checkland [1981] draws attention to two alternative paradigms to explain the nature and significance of systems thinking. In the first paradigm, the world is considered to be systemic and is studied systematically; while in the second paradigm, the world is problematic, that is, it admits to many different interpretations and we study it systemically. Following this continuum, the first paradigm reflects the notion of hard systems thinking, while the second paradigm reflects the notion of soft systems thinking.

Hard systems thinking refers to systems engineering thinking where a systematic process of problem solving is followed [Geode, 2005]. Within this thinking, information systems development is traditionally seen as a hard approach, where stages of a lifecycle can be identified to simplify the development process. As an example, the system development lifecycle (SDLC) for traditional information systems can be classified as a hard systems approach to systems development. Typical phases of the SDLC according to the 'waterfall' model [Royce, 1970] include requirements analysis, system and software design, implementation and unit testing, systems testing, and operation and maintenance. In this mode, user participation is normally restricted to the first phase and testing is done according to user specifications.

During the analysis phase of a software development effort, however, the analyst(s) must accomplish two goals: first, achieve an adequate, complete, and verifiable understanding of the problem at hand, i.e., the problem that the new/modified system is supposed to solve (about 60% of the analysis effort according to researchers); and second, define a solution to this problem, i.e., describe how the new/modified system will appear to the user(s). Hard systems rely more on the technical view in the design of computer-based information systems, and this approach usually leads to failure in such projects [Goede, 2005]. Jackson [1997] [in Cropley and Cook, 2005] identifies, among the reasons for failure of information technology systems, the fact that most of them give inadequate attention to the human and organizational factors which are vital in determining the ultimate effectiveness of new systems. Cropley and Cook [2005] emphasize that the lack of success to date, results from the limited scope of systems engineering methodology as currently applied and the inability of hard systems engineering, by itself, to adequately deal with this complex systems domain.

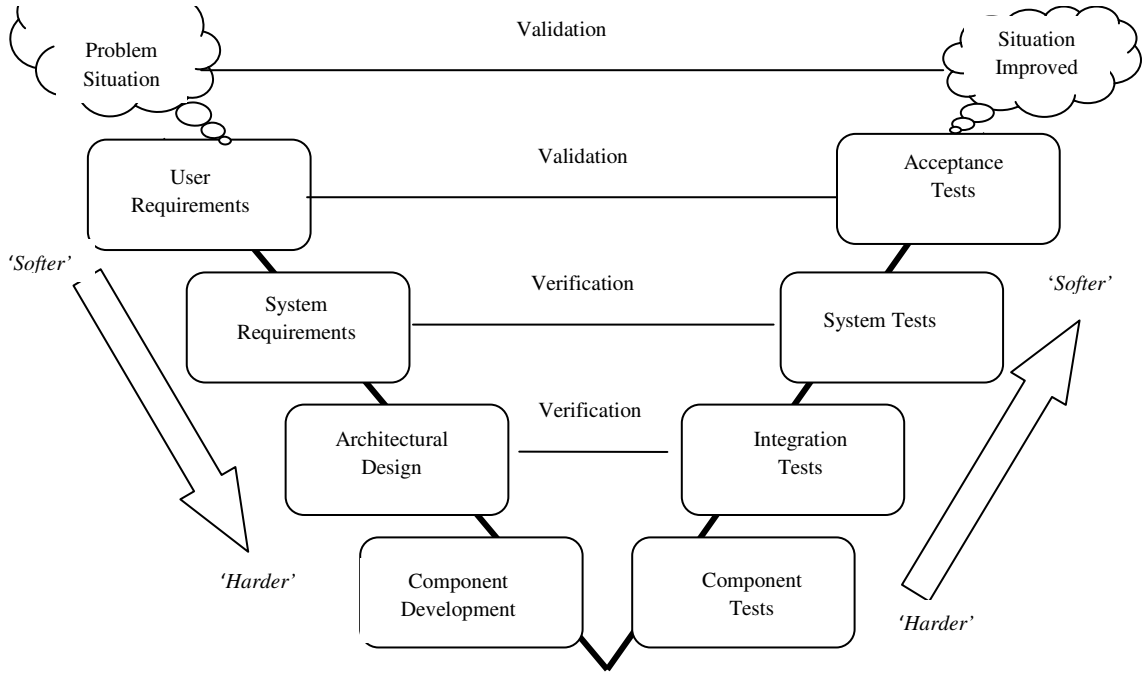
Action research carried out at Lancaster revealed that traditional approaches from systems engineering were not suitable to all environments, particularly those experiencing some kind of "problem" which could not easily be defined [Bond and Kirkham, 1999]. Since the purpose of information systems is to facilitate and mediate social interaction, the social aspects are a crucial element in their development and use [Lycett, 1999]. Traditional hard techniques are therefore of little use in the initial analysis of a human activity system, which is likely to have problem areas that are not clearly defined, and which are unstructured [Lehaney and Paul, 1996; Williams and Kennedy, 2006].

Soft systems methodology (SSM) was therefore developed to provide a structured methodology, which could be used to explore such problems, initially as a precursor to information systems design. SSM is a general problem structuring approach that seeks to incorporate multiple stakeholder views in the analysis of a given problem [Moores and Gregory, 2000]. When applied to IS development, the method requires negotiation and debate between the stakeholders when exploring the feasibility of developing an information system.

Effective requirements engineering, therefore, is most effectively seen as a form of heterogeneous engineering in which technical, social, economic and institutional factors are brought together in a current solution space that provides the baseline for construction of proposed new solution spaces [Bergman, *et al*., 2002]. The rationale for the suggested methodological pluralism is to maximize our understanding in elicitation of domain knowledge that can be specified and validated to improve the effectiveness of the RE process.

Thus, according to Hutton [2006], it is sensible that a soft systems analysis precedes the definition of the user requirements and the transition into the hard world of solutions. In this respect, Hutton [2006] recommends the introduction of an additional layer to the familiar 'V' model of systems development, with a transition from soft to hard thinking as the problem situation is refined into a statement of requirements, and then from hard to soft again as we ensure the system delivers the requirements within the operational context and does improve the situation. This is illustrated on figure 6 below.

Figure 2: Adding another Layer to the Systems Engineering V-Model [Adapted from Arnold *et al.*, 1998]



This model shows that SSM can be applied as an additional layer to a systems engineering framework. The root definitions and conceptual modeling encourage stakeholder involvement and buy-in during the early stages of a project, and provide an abstract representation of what is needed to develop and justify the user requirements. Maintaining the rich picture and the conceptual models provides a vehicle to demonstrate that the system has improved the problem situation, hence another layer to the 'V'. Eva [2004] believes that the end result in SSM is an understanding of the problem domain so that a hard study can then be applied to specify a solution.

7. E-LEARNING REQUIREMENTS ANALYSIS

Learning involves real life situations, and real life involves ill-formulated and ill-structured problems and conditions: real life problems have context, depth, complexity and duration; involve cooperative situations and shared consequences; and are worth solving and can provide benefits when solved [Fitzsimmons, 2001]. In the analysis of any learning environment, whether small or large, one observes within it a set of human activities related to each other so they can be viewed as a whole [Checkland, 1981].

The analysis of learners' requirements is vital for identifying what they want and what they expect from a learning environment [Sun *et al*., 2003]. A learning environment can be seen as a complex socio-technical system, since it can be idealised as an open system that depends on the technology, the sentiments

of the members, and the organisational environment [Checkland, 1981]. It is also essential for the instructional designer to know learners' characteristics in order to build a system that maps learners' preferences.

Although the system is organised to focus on a primary task (the learning) this cannot be separated from the environment and the social factors, including cultural ones [Slay, 2002]. For active, authentic learning, then, activities should be ill-defined, comprise a single complex task to be investigated over a sustained period of time, provide the opportunity to examine the task from different perspectives, promote reflection to enable abstractions to be formed, promote articulation to enable tacit knowledge to be made explicit, and provide coaching and scaffolding by faculty at critical times [Fitzsimmons, 2001].

The design of traditional e-learning environments has been criticized for being technology-driven rather than problem-driven or learner-driven [Tynjala and Hakkinen, 2005]. In designing e-learning systems, however, von Brevern (2004) argues that one needs first to define the psychological (i.e. cognitive) and social rationales, and then incorporate them into any instructional design model. Indeed, von Brevern believes that applying the pedagogical context of e-learning (cognitive, behavioural or social) is key to understanding and planning educational technology systems that are interrelated to "subject heavy domains" or "context specific" environments.

The situated aspects of learning, according to Bell [2006], could be explored better if learning was regarded as taking place within a human activity system (HAS), and therefore, rather than focusing on 'hard system' thinking, Peter Checkland's soft systems methodology (SSM) has been found to be quite appropriate during the initial phases in developing e-learning artefacts. Hutton [2006] argues that SSM would encourage the analyst to explore the problem situation rigorously, rather than to develop a solution to a perceived problem, or even to define a problem that fits the preferred solution.

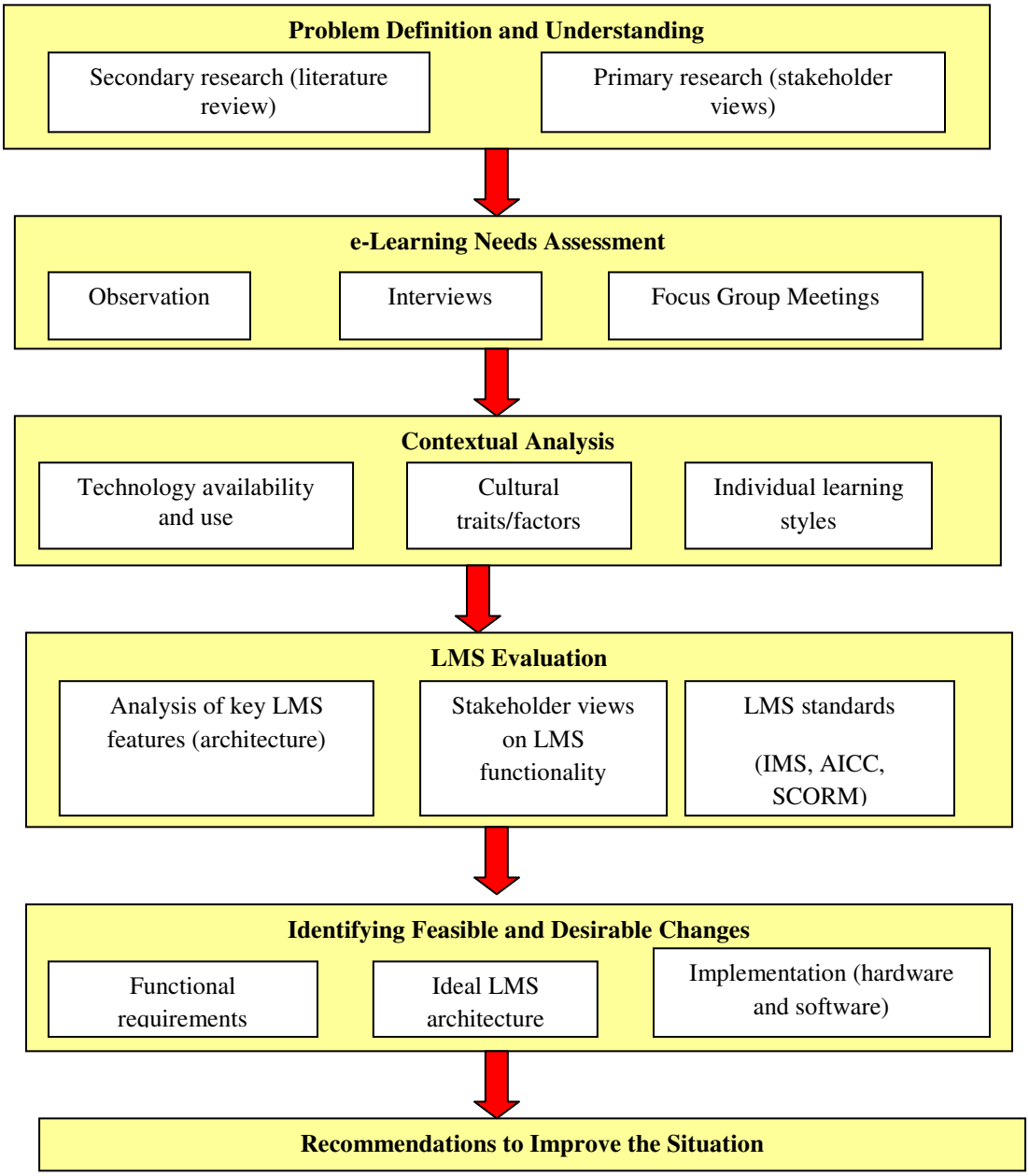
For this reason, e-learning solutions must include features related both to software design issues, and pedagogical aspects [Tynjala and Hakkinen, 2005; Epstein, 2006]. Because software design and pedagogical solutions are inter-depended, the design process itself has to be carried out as a collaborative process between experts in learning and experts in software design.

8. PROPOSED REQUIREMENTS ANALYSIS FRAMEWORK

From the foregoing discussion, a framework may now be proposed for analysing requirements for a constructivist online learning system. Carey and Carlson [2002] advise that in order to develop a framework that addresses a set of applications in a domain (or domains), one needs to have requirements from each of these applications (for each domain). Requirements are best gained from experts in the domain, for example, from business analysts for business applications (or education/e-learning experts for e-learning requirements). These domain experts do not have to be technically savvy—that is the technical team's job—but they need to know the applications (or more specifically, the business processes) in the domain (or domains) very well.

A six-step framework has been proposed for this analysis (figure 3), and the process is explained in the sections that follow.

Figure 3: Proposed Requirements Analysis Framework for Constructivist Online Learning System



8.1 Problem Definition and Understanding

Preceding requirements analysis, a comprehensive structured approach is applied in understating the challenges faced and the gaps that currently exist in the provision of personalised e-learning. As observed by Talavera *et al.* [2001], successful technology-based environments design must start with a compelling vision statement where the stakeholders of the project reach an agreement on what problem need to be solved and which are the boundaries and most important features of the system. Ideas from Checkland's 7-stage soft systems methodology (SSM) will be applied in developing this understanding.

First, the researcher must seek to develop a proper understanding of the e-learning environment with respect to the learning environment that they seek to address. Apart from literature review, as many perceptions of the problem situation as possible are captured from a wide range of people to ensure a balanced view of the situation. This is done through consultations (in the form of structured interviews) with education/e-learning experts, LMS system administrators, and learners, especially those with some experience in using the web as a teaching and learning resource. More preferably, this sample will be

selected from instructors who have at least shared an online version of a syllabus, posted an instructor profile, or reviewed and critiqued online resources on the web.

8.2 (e-)Learning Needs Assessment

The exact needs of learners in this environment are established, and then the main e-learning platforms that are currently being used to address these needs are identified. Respondents here include learners, education/e-learning experts, instructors, content developers, and LMS system administrators.

A case study approach is employed in developing an understanding of the e-learning needs, and the three East African Partner States of Kenya, Tanzania and Uganda are included. Data is collected from three universities in this region, which are currently experimenting with online learning in a meaningful way – Makerere University (Uganda), University of Dar es Salaam (Tanzania), and University of Nairobi (Kenya). Sampling from the three countries in the region is expected to give a relatively diverse set of views based on the experience of users.

Various strategies are employed when collecting facts, ranging from very informal, unstructured approaches to very formal, structured tools employed in traditional systems analysis. Participant observation, for instance, is used to identify tasks performed (what learning activities are undertaken), identify tools employed (the e-learning systems/platforms used), and establish interactions between people/systems (the extent to which users appreciate the systems). A consensus workshop (in the lines of focus group meetings) is conducted within each of the three universities to examine the various models and attempt to reach an accommodation between the viewpoints reflected. All relevant stakeholders – learners, instructors, education/e-learning experts, content developers, LMS system administrators, etc – are participants in these meetings.

8.3 Contextual Analysis

There must be a deliberate attempt to analyse the environmental and situational contexts within which learning occurs, and which may have some impact on learning – including technology use and challenges (e.g. access to Internet (bandwidth), cultural traits that influence learning (e.g. reading culture), and individual predisposition towards learning (e.g. learning styles). The target group for this category is mainly the learners themselves, supplemented by instructors and LMS system administrators. Over a period of time, a database of the user characteristics and the corresponding preferences would allow for the automatic generation and recommendations of the design features of a constructivist framework. Personalisation rules are established based on the database of the relationship between user characteristics and their preferred design features.

8.4 LMS Evaluation

Before these systems are considered effective, the user experience is studied and analyzed to provide the optimum solution to meet pedagogical needs of both faculty and students. This not only by studying the systems themselves, but also seeking views of users – including learners, instructors, education/e-learning experts, etc. These platforms are studied in detail in order to understand their core features, and how they implement the various learning theories/models as informed by the identified needs. In another study [Ayoo, 2008], it was established that East African universities use mainly an open source system called Moodle for their e-learning activities, while popular commercial products are Blackboard and WebCT. Still others (used occasionally/minimally) include WEDUSOFT, KEWL and A-Tutor.

Constructivist evaluation offers a very explicit methodological framework, grounded in the epistemology of social constructivism, with clear points of convergence and useful pointers to appropriate validation criteria for research [Levy, 2003]. Constructivist evaluation is employed to consider *claims* (assertions favourable to what is being evaluated); *concerns* (unfavourable assertions); and *issues* (items on which reasonable people might disagree). This type of evaluation aims to reach consensus between different stakeholders and the evaluators in organising the negotiation.

Based on the evaluation criterion, both structured and semi-structured questionnaires are used to capture the views of users on the present online learning environments. Various features are taken into account when evaluating the online learning platforms, starting from the function and usability of the overall learning system in the context of the human, social and cultural contexts where they are used, and developing an understanding of how the various features are integrated to facilitate personalised learning. Both pedagogical and technological aspects are carefully examined.

When evaluating learning management systems, one hears a lot of terms that end in "*-ity*": high availability, usability, scalability, interoperability, stability and security. Hall [2003] provides a framework within which each of these issues may be examined to establish why they are critical to the function of any enterprise management system.

1. *High availability*: The LMS must be robust enough to serve the diverse needs of thousands of learners, administrators, content builders and instructors simultaneously.
2. *Scalability*: The infrastructure should be able to expand—or "scale"—to meet future growth, both in terms of the volume of instruction and the size of the student body.
3. *Usability*: To support a host of automated and personalized services, such as self-paced and role-specific learning, the access, delivery and presentation of material must be easy-to-use and highly intuitive—like surfing on the Web or shopping on Amazon.com.
4. *Interoperability*: To support content from different sources and multiple vendors' hardware/software solutions, the LMS should be based on open industry standards for Web deployments (XML, SOAP or AQ) and support the major learning standards (AICC, SCORM, IMS and IEEE).
5. *Stability*: The LMS infrastructure can reliably and effectively manage a large enterprise implementation running 24x7.
6. *Security*: As with any outward-facing collaborative solution, the LMS can selectively limit and control access to online content, resources and back-end functions, both internally and externally, for its diverse user community.

From an operational point of view, the extent to which the LMS and its key components are web-deployable – including content management, user administration and system administration – is established. Further, the evaluation targets the extent to which the LMS is built on an open architecture that supports the emerging learning standards – IMS, AICC and SCORM. These industry-standard groups are creating technical specifications to enable and support a unified, standardized content model for web-based learning.

8.5 Identifying Feasible and Desirable Changes

The focus at this stage is to seek ways of improving the situation, by identifying possible changes in the system structure, system procedures and attitudes of the users. Here, care is taken to ensure that the changes are systemically desirable, that they are as a result of the insight gained from selection of root definitions and conceptual model building, and that they are culturally feasible given the characteristics of the situation, the people in it, their shared experiences and their prejudices.

Stakeholder views are sought on what their expectations are from an ideal/improved environment. The aim is to collect worldviews on the question "What is your view of a pedagogically effective online learning environment?" In applying a systems engineering perspective to e-learning, a number of issues are considered:

1. What are all the functional requirements an e-learning system must fulfill?
2. What should an ideal e-learning architecture look like?
3. How does one pull together all the hardware and software pieces to get the best solution for the identified needs?

8.6 Recommendations to Improve the Situation

The job at this stage is to implement the changes identified and put them into action. This is the stage at which a hard methodology now takes over in ensuring the development of a system to address the problem that will now have been identified and structured.

7. CONCLUSION

The emphasis in information systems development must be a collaborative effort between technical experts and those who truly understand the purposeful action served. Models of purposeful human activities can be used as scenarios to initiate and structure sensible discussion about information support for the people undertaking the real-world problem situations.

Hard systems engineering methods return to the foreground when the ill-structured problem of the human activity system has been refined to give satisfactory requirements for the realization of the components, in an evolving, complex environment. The central feature of a system methodology for real-time information systems, therefore, calls for the judicious application of hard systems methods to well-defined needs, in the global framework of a soft systems approach. Hard systems engineering must take place in, and be guided by, the nature of the overarching human activity system. As a result, the requirement for a complete systems approach, using both hard and soft concepts, is recommended for e-learning systems.

8. REFERENCES

AYOO, P. O. 2008. Are East African universities ready for e-learning? *Proceedings of the1st Regional Conference on e-Learning*, Kenyatta University, Nairobi, November 18 – 20 2008.

BANATHY, B. website. A taste of systemics: characteristics of a human activity system. In Mandel, T. (ed.) *ISSS Integrated Systemic Inquiry Primer Project (ISIPP).* http://www.isss.org/primer/bela6.html (retrieved 05/10/2008)

BASTANI, B. 2007. A requirements analysis framework for open systems requirements engineering. In *ACM SIGSOFT Software Engineering Notes*. Vol. 32, Issue 2, pp. 1 – 19. http://portal.acm.org/citation.cfm?id=1234753 (retrieved 30/07/2008)

BENJAMIN, L. K. 1998. Practical software requirements: a manual of content and style. Greenwich, CT: Manning Publications Co.

BERGMAN, M., KING, J. L. AND LYYTINEN, K. 2002. Large-scale requirements analysis revisited: the need for understanding the political ecology of requirements engineering. *Requirements Engineering Journal*, Springer London. Vol. 7, No. 3. http://www.springerlink.com/content/79xlamrhc5691ex8/ (retrieved 14/05/2008).

BLEISTEIN, S., *et al.* 2006. B-SCP: A requirements analysis framework for validating strategic alignment of organizational IT based on strategy, context, and process. *Information and Software Technology*, 48 (2006), pp. 846–868.

CAREY, J. AND CARLSON, B. 2002. Introduction to developing object-oriented frameworks. http://www.awprofessional.com/articles/article.asp?p=27123&rl=1 (retrieved 27/07/07)

CASTRO, J., KOLP, M. AND MYLOPOULOS, J. 2002. Towards requirements-driven information systems engineering: The Tropos Project. http://citeseer.ist.psu.edu/castro02towards.html (retrieved 30/07/2008)

CHECKLAND, P. 1981. *Systems thinking, systems practice*. Chichester, UK: Wiley.

CHECKLAND, P. AND HOLWELL, S. 1998. *Information, systems and information systems: making sense of the field*. Chichester: Wiley.

DE CROOCK M. *et al.*, 2002. Active learning for adaptive Internet (Alfanet IST-2001-33288). ftp://ftp.cordis.europa.eu/pub/ist/docs/ka3/eat/alFANET.pdf (retrieved 11/09/2007)

GREGORY, F. H. 1993. SSM to information systems: a Wittgensteinian approach. *Journal of Information Systems* 3, 149 - 168.

GREGORY, F. H. 1995. Soft systems models for knowledge elicitation and representation. *J. Opl Res. Soc.,* 46, pp. 562 - 578.

GREGORY, F. H. 1993. Cause, effect, efficiency and soft systems models. *J. Opl Res. Soc.,* 44, pp. 333 - 344.

HALL, J. 2003. Assessing learning management systems. http://www.clomedia.com/content/templates/clo_feature.asp?articleid=91 (retrieved 14/05/2008)

IEEE 1998. IEEE recommended practice for software requirements specifications. *IEEE STD-830*.

LIU, S. 1992. Formal requirements analysis method based on data flow analysis and rapid prototyping. A thesis submitted to the University of Manchester for the degree of Doctor of Philosophy in the Faculty of Science.

LYCETT, M. 1999. The development of component-based evolutionary information systems. A dissertation submitted for the degree of Doctor of Philosophy. Department of Information Systems and Computing at St John's Brunel University.

MACHADO, M. AND TAO, E. 2007. Blackboard vs. moodle: comparing user experience of learning management systems. *Frontiers in Education Conference - Global Engineering: Knowledge without borders, opportunities without passports, 2007 FIE '07 37th annual***.** http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?tp=&arnumber=4417910&isnumber=4417795 (retrieved 14/05/2008).

MERRIAM-WEBSTER DICTIONARY, 1995.

MCCONNELL, S. 1996. Rapid development: taming wild software schedules, 1st ed., Redmond, WA: Microsoft Press.

MCEWEN, S. 2004. Requirements: an introduction. http://www.ibm.com/developerworks/rational/library/4166.html?S_TACT=105AGX78&S_CMP=HP (retrieved 30/07/2008).

MERCKER, J. 2007 A requirements analysis primer. http://www.techlinks.net/CommunityPublishing/tabid/92/articleType/ArticleView/articleId/3849/A-Requirements-Analysis-Primer.aspx (retrieved 11/09/2007).

MÖDRITSCHER, F. *et al.,* 2006. The first AdeLE prototype at a glance. In *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2006*. pp. 791-798. http://coronet.iicm.tugraz.at/denis/pubs/edmedia06b.pdf.

MOORES, T. T. AND GREGORY, F. H. 2000. Cultural problems in applying SSM for IS development. *Journal of Global Information Management*, vol. 8, no. 1. http://www.geocities.com/Athens/Oracle/6925/Cultprobs_ssm.htm (retrieved 31/07/2008).

NG, PAN-WEI 2003. Adopting use cases, part 1: understanding types of use cases and artifacts. http://www.ibm.com/developerworks/rational/library/1809.html?S_TACT=105AGX78&S_CMP=HP (retrieved 30/07/2008).

NAIK, S. 2004. Software requirements analysis. http://www.devpapers.com/article/233 (retrieved 11/09/2007).

NARDI, B., (ed.) 1996. Context and consciousness: activity theory and human-computer interaction. Cambridge, MA: MIT Press.

NEALE, S. H. *et al*. 2007. Online biodiversity resources – principles for usability. *Biodiversity Informatics*, Vol. 4, pp. 27-36.

PROTEANS SOFTWARE SOLUTIONS 2005. Best practices of requirements collection. In *Agile RUP for Product Development*. http://www.proteans.com/CS/Web/blogs/agilerup/archive/2005/07/06/19.aspx (retrieved 11/09/2007).

RICHARD, V. 1997. Stakeholders, soft systems and technology: separation and mediation in the analysis of information system requirements. *Information Systems Journal*. 7 (1) , pp. 21–46.

SLAY, J. 2002. Human activity systems: a theoretical framework for designing learning for multicultural settings. *Educational Technology & Society*. 5 (1). http://www.ifets.info/journals/5_1/slay.html (retrieved 05/10/2008).

SMITH, R. website. How does it all work: playing with technology? http://www.edtechleaders.org/chat_series/presentations/smith.ppt. (Retrieved 14/05/2008).

SUDHAKAR 2005. Best practices of requirements collection. In *Agile RUP for Product Development*. http://www.proteans.com/CS/Web/blogs/agilerup/archive/2005/07/06/19.aspx (retrieved 14/05/2008)

SUN, L., WILLIAMS, S., OUSMANOU, K. AND LUBEGA, J. 2003. Building personalised functions into dynamic content packaging to support individual learners. In: *Roy Technology Blueprint FAQs*. http://www.jcapitaltech.com/faqs.php (retrieved 11/09/2007).

TECHNOLOGY BLUEPRINT FAQS http://www.jcapitaltech.com/faqs.php (retrieved 11/09/2007).

THE OBJECT AGENCY, INC. 1995. Defining OO requirements analysis. http://www.toa.com/pub/00DFOORA.PDF (retrieved 11/09/2007).

THE STANDISH GROUP 1994. *The CHAOS Report*.

URQUHART, C. 2001. Bridging information requirements and information needs assessment: do scenarios and vignettes provide a link? *Information Research*, 6(2). http://InformationR.net/ir/6-2/paper102.html (retrieved 05/10/2008).

VAT, K. H. 2005. Modeling human activity systems for collaborative project work: an IS development perspective. *Issues in Informing Science and Information Technology*. http://proceedings.informingscience.org/InSITE2005/I05f65Vat.pdf (retrieved 05/10/2008).

WIEGERS, K. E. 2003. Software requirements 2: practical techniques for gathering and managing requirements throughout the product development cycle, 2nd ed. Redmond: Microsoft Press.

WILLIAMS, D. AND KENNEDY, M 2006. A framework for improving the requirements engineering process effectiveness.

WILLIAMS, S. (ed.) 2003. *Proceedings of the 2nd European Conference on e-Learning*, Glasgow Caledonian University Glasgow, Scotland 6-7 November 2003.