A THREE-DIMENSIONAL REQUIREMENTS ELICITATION AND MANAGEMENT DECISION-MAKING SCHEME FOR THE DEVELOPMENT OF NEW SOFTWARE COMPONENTS

Andreas S. Andreou, Andreas C. Zographos and George A. Papadopoulos
Department of Computer Science, University of Cyprus, 75 Kallipoleos Str., P.O.Box 20537, CY1678, Nicosia, Cyprus
Email: aandreou@cs.ucy.ac.cy, a.z.zsoft@cytanet.com.cy, george@cs.ucy.ac.cy

Keywords: Software components development, Requirements elicitation, Management decision-making, Reusability

Abstract: Requirements analysis and general management issues within the development process of new software components are addressed in this paper, focusing on factors that result from requirements elicitation and significantly affect management decisions and development activities. A new methodology performing a certain form of requirements identification and collection prior to developing new software components is proposed and demonstrated, the essence of which lays on a three-entity model that describes the relationship between different types of component stakeholders: Developers, reusers and end-users. The model is supported by a set of critical factors analysed in the context of three main directions that orient the production of a new component, that is, the generality of the services offered, the management approach and the characteristics of the targeted market. The investigation of the three directions produces critical success factors that are closely connected and interdependent. Further analysis of the significance of each factor according to the priorities set by component developers can provide a detail picture of potential management implications during the development process and more importantly can support management decisions related to if and how development should proceed.

1 INTRODUCTION

Quality in software engineering is defined as the level to which a certain software product meets its specifications (Schach, 1999). Specifications are the formal or technical transformation of corresponding requirements collected during a preceding analysis stage. Therefore, quality is highly dependent on the success a requirements engineering process exhibits on identifying the right needs that a software product must fulfil and serve. Component-based software engineering makes use of the notion of components, independent software parts that can be reused in a way that their combination in various fields or applications results in the creation of a full software product dedicated to serve a specific working domain. A software component should aim primarily at meeting its requirements/specifications for increasing its quality. In addition, components must be of high quality due to customers insisting on such quality, and this can be achieved only if the requirements are clearly specified (Szyperski, 1997). In a demanding and dynamically evolving component market, developers are often facing a difficult management problem (profit-wise): Should they proceed in developing a new software component or not? Are market conditions in favour of the former or the latter? Developers' target of course is to produce a component that will be successful, will have many uses (purchases) and will result financial profits. The critical question for developers to answer is the following: How can they orient component success factors prior to development, or what management decisions should they take to achieve high quality and sufficient reuses? The present paper moves along the lines of these management and quality issues attempting to provide a framework for collecting factors that on one hand will assist the analysis and specification of new components, and on the other will offer the means for the management to decide if conditions support the development of a particular component, and if so what other critical aspects of the development process should be examined.

Little research effort has so far been reported in the international literature that was devoted not to the process of reusing existing components but to the study of requirements elicitation and specification as a means for achieving high quality or assisting the management activities when developing new ones. Baum and Becker (2000) introduce the fundamental concepts of generic components for addressing a broad spectrum of application requirements, while Grundy (1999, 2000) proposes the alternative of aspect-oriented requirements engineering for component-based software systems development. Moreover, specifications are used in Mili et al. (1994) to address one of the open problems in the field of component-based software engineering, namely the storing and retrieving of components, by proposing a scheme using formal representations to model abstract specifications. Sitaraman et al. (2001) introduce a compositional approach to address performance specification problems in an effort to assist the choosing and assembling of components that best fit their time and space requirements. In Sugumaran et al. (1999) a new methodology is proposed and evaluated, which reuses domain knowledge artifacts for systems analysis and detailed object/code artifacts for systems development. Requirements engineering for component-based systems is also reported in a number of research studies: Barber (1999) introduces the Systems Engineering Process Activities (SEPA) methodology, which addresses the issue of requirements reuse and evolution in component-based software development from the perspective of separating domain-level requirements from application-level ones. Ncube and Maiden (1999) propose the use of the PORE, an iterative requirements engineering process, which gives emphasis on requirements acquisition, definition and validation, for the selection of components during the development of component-based systems. Finally, Baum et al. (2000) describe Design Spaces as a way of mapping requirements to reusable components by representing requirements in a cross-produce fashion; this cross-produce can then be correlated against specific criteria to derive the most suitable components for building new software systems.

All the above references, nevertheless, address their research targets based on software components that have already been developed and are available for reuse. The present paper stands on a totally different ground than the reported research work, with its main contribution being the proposition of a new methodology that can assist the analysis phase and guide the management tasks when developing a new software component. Specifically, it is involved with the issue of achieving high quality when originally developing software components by looking at the problem from the angle of identifying and addressing proper component requirements and specification prior to development. The attention is drawn upon: (i) constructing a simple but robust methodology for guiding developers in defining a set of key features, both functional as well as non-functional, that the component should exhibit to enhance its quality and promote its attractiveness to potential reusers, (ii) offering a way of tackling potential management problems prior to development.

2 SOFTWARE COMPONENTS QUALITY, REQUIREMENTS ANALYSIS AND MANAGEMENT

Software components quality can in general be described in a variety of dimensions: Level of performance, resource utilization and efficiency, robustness, value-for-money, fault isolation, safety, ease-of-use, ease-of-modification, interoperability, adaptability, level of independence, viability, etc. These dimensions, though, are highly dependant on the application domain, the specific functionality the component offers and the environment, both technical and business, in which it will be integrated. Unfortunately, these are never known prior to the actual process of reusing components to achieve a specific goal. Thus, information about the critical factors that guide the success of a component as part of a larger group of other interacting components can become available only when the three axons, application domain, required functionality and existing technical and business environment, are known.

The inherent problem with the effort to elicit proper requirements upon development of components to be reused is actually the inability to do so until these requirements become available at a later stage, that is, during the actual process of reuse. Our proposition is a new methodology that tackles this inherent problem via a scheme incorporating hypothetical requirements. These requirements are identified through the recording of a number of critical factors that will potentially affect components' reuse. The critical factors are then analysed in terms of interdependencies and conclusions are drawn upon potential development implications. Based on these conclusions, management decisions can be taken regarding the production of a software component or the abortion of the process. In the former case, requirements elicitation and collection can be performed based on specific stakeholders' informational templates.

The methodology is analytically described through a series of three steps as follows:

Step1: Identify critical success factors within the following three axons: "Market", "Management" and "Generality"

Critical success factors can be categorised into three main dimensions: The target-market in which the component aims at entering, the component's management, and its generality (figure 1). The first dimension determines whether the component is developed to create a new market and possibly to set a new standard, or to compete in an existing market of components aspiring at winning competition or gaining a high percentage of customers. The dimension of management attempts to set an upper bound to the price estimated for a component's future purchase by potential customers, according to the effort, time and resources put in the process of its development, as well as according to the advantages offered upon its reuse. The generality dimension refers to the functionality feature of a component for general problem solving capabilities in contrast to the provision of a specialized solution for a particular problem. All three dimensions are highly correlated and the first dimension to be addressed (entry point) will affect decision making on the other two. For example, if the component will create a new market, then the management may decide to accept higher development costs and increased number of human resources than usual, aiming at shortening the release time and be benefited by the prospective monopoly.

The selection of these particular dimensions was not arbitrary. Wallnau et al. (2002) report that the nature of the system architecture changes under the market regime to respond to marketplace dynamics (availability of component features, interfaces that components support, which components thrive, etc.). This directly influences the selection of components that best support the functionality of the various structural parts of the system and dictate the kind of interaction between these components. In practice, the "Market" axon is more likely to be used as the starting entry-point and this is depicted in figure 1 with a larger circle. The "Generality" axon represents the viewpoint of software component manufacturers. In a recent study (Seppanen et al., 2001) the authors presented a survey indicating that component development may pay off if components are "big" enough in functional terms, i.e. they come close to being software products. Thus, under this perspective the generality of services provided may be regarded as an aspect of the at most significance for component developers. Finally, the "Management" axon is necessary to complete the scenery by integrating the information provided by the other two dimensions and by taking control over the planning and organizing activities during the development process.

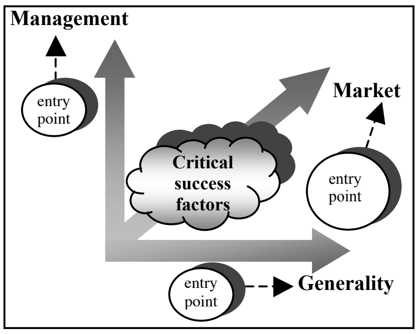

Figure 1: Main dimensions for identifying critical factors that will potentially affect components' reuse. Dimensions are highly correlated: The first dimension to be analysed (entry point) will determine the critical factors that belong to the other two

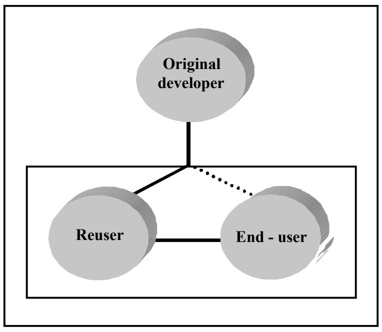

Figure 2: A three-entity relationship model between different stakeholders. The solid line indicates direct connection. The dashed line corresponds to indirect type of connection. The inner rectangle indicates that reuser and end-user can be seen as one entity

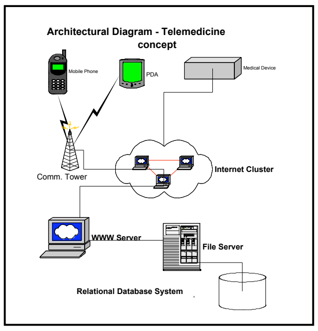

Figure 3: Diagrammatic representation of the component-based telemedicine system in the context of which the proposed methodology was utilized during the development of components

The identification of certain critical factors for each of the proposed model's axons is based on the focus questions depicted in tables 1, 2 and 3. Focus questions may be divided into primary and secondary, while the answers to these questions will enable the collection of associated factors reflecting various aspects of each of the three axons analysed. Thus, a general informational platform is constructed, reflecting on one hand the targets served and the advantages offered by a certain component, and on the other hand the issues related to the actual process of development and marketing.

After completing the gathering of the critical factors, the component developer assigns a level of importance to each factor. Thus, he prioritizes the requirements that result from the factors identified, giving greater emphasis on those he regards more significant. The level of impact that this prioritization may have on the development process is analyzed and examined in step 2, resulting to insights of where and how development procedures will be affected. For example, if the component will join a current market offering significant advantages compared to its competitors, then the average cost of competitive components will have little or no interest at all and thus it will be given less attention. Further to that, if the time-to-market estimation dictates quick development in order to bid the market at the right time when a need is present, then the number and/or the skills of the development team's members will be given higher priority than the development costs.

Table 1: Focus questions for the "Market" axon

Primary Focus Questions	**Secondary Focus Questions**
1.1 Is the targeted market a new market (functional-wise) → YES	1.1.1 What are the potentials for success (advantages: first solution? New functional characteristics?)?
	1.1.2 What new services will be offered?
	1.1.3 Will the process of setting a new standard be attempted?
1.2 Is the targeted market an existing market (functional-wise) → YES	1.2.1 What are the principal competitors (vendor-wise)?
	1.2.2 What are the competitive components and what are the similarities and differences of their functionality compared to the new component?
	1.2.3 What is the average purchase cost of competitive components?

Table 2: Focus questions for the "Management" axon

Focus Questions
2.1 What is the estimated time needed to develop the component?
2.2 What is the estimation about the appropriated time to release the component in the targeted market (time-to-market)?
2.3 What is the estimated number of people required for development?
2.4 What are the necessary skills for the staff to be involved?
2.5 What is the estimated development cost?
2.6 What is the estimated purchase cost?

Table 3: Focus questions for the "Generality" axon

Primary Focus Questions	**Secondary Focus Questions**
3.1 Will the component be of a general-purpose functionality or a specific problem solver?	3.1.1 What services will be offered?
	3.1.2 How will these services fit and survive in the targeted market?
	3.1.3 What application domain is addressed?
	3.1.4 What is the problem dealt and what is the solution offered?
	3.1.5 How does the solution tackle the specific problem?

Table 4. Factor weights abbreviations for the "Market" and "Management" axons

Market	**Management**
New : MR1	Time to develop : MN1
Functional advantages : MR1,1	Time to market : MN2
New services : MR1,2	Human resources required : MN3
New standard : MR1,3	Level of skills required : MN4
Existing : MR2	Development cost : MN5
Competitors' position in the market : MR2,1	Purchase cost : MN6
Similarities with competitive components : MR2,2	
Average purchase cost of competitive components: MR2,3	

Step 2: Define critical factors' level of significance according to developers' priorities (management decisions). Analyse dependencies and orient the impact of management decisions on the less significant factors

Managing the development process of new software components is a difficult task comprising a set of decisions that, no doubt, will have a determinative effect on the success the component will achieve entering the components market arena. For example, if the management chooses to proceed in a slower development rate than the one suggested by market analysis for the release of a new component, and attempts to reduce development costs by using only existing personnel and not recruiting new people, then market conditions may change in the meantime, with a competitor component-vendor succeeding in releasing his component first and overtaking the market share. Thus, making the right decision the right time can be the key to the lock for newly created components to enjoy a rich number of reuses. This step will introduce a management decision scheme, which offers a general picture of how certain factor priorities set by managers/developers can affect critical aspects of the development process. Recalling that the requirements elicitation model proposed comprises three main axons, which are then decomposed into a number of critical factors through a set of focus questions, we will proceed in analysing factor priorities and dependencies between factors. Table 4 presents the abbreviations of the factors that describe each component, which will be used to reflect the relative significance that may be assigned by component developers. We will concentrate on factors within the first two axons ("Market", "Management"), due to the fact that usually the factors under the "Generality" axon do not have a major influence on the management decisions taken prior to the development process. By this we mean that developers rarely set the generality or specialization of a component's services as a priority target with which everything else must align.

Factor priorities can be classified into two categories: (i) Mutually excluded priorities, that is, priorities that cannot be set simultaneously. This category includes factors that belong to the MR1 and MR2 groups. (ii) Coexisting priorities, which indicate that the importance of some factors is simultaneously taken into consideration. This category contains in general all the factors that do not belong to category (i). Our suggestion is the creation of a Factor Interdependencies Table (FIT), which will represent the impact of each factor on the rest of the set. Specifically, given that a certain factor was assigned higher priority than others, the FIT determines what other factors will be influenced. The FIT is constructed as a (n□n) table containing the abbreviated factor descriptions of table 4 in the first row/column. The dependencies are depicted in the table with a mark from a row-factor (significant) to various column-factors (affected).

The construction of the FIT assumes that the management/development team has already determined, during step 1, the factors it regards more critical and significant prior to development. Dependencies are then marked and a level of impact that priority factors put on the affected ones is identified. Thus, a general picture is formed reflecting potential implications that decision-making will cause and giving insights on how cost, time and human resources will eventually be shifted or shaped by setting certain priorities. Following this management decision-making scheme, the managers can organize and plan better the development process due to the fact that the FIT analysis transforms certain unknown risks into expected risks. The preceding factors significance analysis can be used as a management tool for deciding whether it will be profitable to proceed and develop a certain component or not. Furthermore, in the case in which the decision is in favour of a component's production, the management will be aware of critical issues related to the development process, such as level of costs, required human resources, deadlines to be met etc., and thus will have the benefit of organizing and monitoring related activities for minimizing or eliminating expected risks.

Step3: Based on the factors identified and studied in steps 1 and 2 proceed to collect the required informational features for the new software component or cancel the development attempt

The requirements elicitation process for a new software component, as stated earlier, has to rely on assumptions and estimations that can, to the most accurate extent possible, reflect requirements that will be set during its reuse and its full operation in an integrated application. In this context a new requirements elicitation process will be proposed, the essence of which rests on a three-entity model as shown in figure 2. The entities participating in the model are the "Original developer" of a software component, the "Reuser" assembling the component with other components to form an application, and the "End-user" who actually works with the application formed.

The model suggests two types of connections/relationships between entities: First, the direct connection (solid lines) describing the relationships between original developer and reuser on one hand, and reuser and end-user on another. The term "direct" is used here to denote that the original developer when creating a component should always bear in mind that its success relies on the number of uses it will have provided that it exhibits all those key characteristics that could attract potential reusers. In align with this, a reuser must always take into consideration the specific needs and requirements of a group of end-users prior to assembling and integrating components. Second, the type of indirect connection (dashed line) reflects the kind of collateral impact that design and implementation decisions on behalf of the original developer may have on the final recipients of the services offered by the particular component. Due to the fact that the reuser represents the end-user with respect to his requirements on quality issues, this type of connection can also be seen as a subset of the developer-reuser relationship (inner rectangle area).

The general form of the model covers also the situation in which reuser and end-user is the same entity. In such a case, an end-user, that need not be an expert programmer, reuses components to develop a software application framework to support his everyday working activities. This case is handled in the proposed model considering the entities "Reuser" and "End-user" as merging into a single entity directly related to the entity "Original developer" with corresponding merger of the direct and indirect types of connection into a single general connection including both types. For simplicity's sake let us perform the following abbreviations for the three connections/relationships taking place in the model of figure 2: OD-R will denote the connection between original developer and reuser, R-EU will refer to the connection between reuser and end-user, and OD-EU will correspond to the connection between original developer and end-user.

As previously mentioned this step will be carried out only in the case in which the result of the FIT analysis favoured the production of the component and its outcome will be specific informational templates for each of the relationships participating in the three-entity model:

OD-R: The template is depicted in table 5 and comprises three main informational sections, namely the "services", "criteria" and "eligibility information" section. These sections contain a significant part of the information required for providing cataloguing and reusing facilities, as well as for revealing human and social factors that contribute or contrast to the reuse of a particular component.

R-EU: This kind of relationship can be addressed using classic requirements engineering processes. Thus, we can follow the suggestion in the work of Kotonya and Sommerville (1996) and Sommerville (2000) and use the Viewpoint Oriented Design (VORD) type of template to collect viewpoint (stakeholder) and service information, as shown in table 6.

Table 5. Informational template for collecting the component developer - reuser relationship factors

OD-R template
Services: A set of available services for reusers to select a component from a list: ▪ Categorization: The type of component categorization followed ▪ Sorting: The kind of sorting used ▪ Retrieving: The way a reuser can retrieve a component ▪ Testing: The available testing privileges offered **Criteria:** A set of criteria to be used by reusers for identifying the right component from a list: ▪ Functionality: The functionality offered by a component ▪ Operating System: The required OS platform for a component to execute ▪ Programming language: The language used for developing a component ▪ Application domain: A description of the field a component best fits and serves ▪ Functional characteristics: A set of significant functional characteristics describing a component **Eligibility information:** Information to be used for testing the eligibility and appropriateness of a component selected from a list: ▪ Source code: A full source code listing including in-line and prologue comments, along with information related to the programming language, compiler and operating system used for developing the component (when modifications are allowed) ▪ Diagrammatic representations: Graphical presentation of data and control flow, logic of functionality and decision trees (when modifications are allowed) ▪ Test cases: A set of test cases, both normal and abnormal, and the corresponding responses ▪ Failure Data: Information describing failure characteristics, such as mean-time-to-failure, mean-time-between-failures, criticality of failures dealt and how they were resolved

Table 6. Informational template based on the VORD standard forms for collecting the reuser – end-user relationship factors

R-EU template	
Viewpoint template	**Service template**
Reference: The viewpoint name **Attributes:** Attributes providing viewpoint information **Events:** Set of event scenarios describing how the system reacts to viewpoint events **Services:** Set of service descriptions **Sub-VPs:** The names of sub-viewpoints	**Reference:** The service name **Rationale:** Reason why the service is provided **Specification:** List of service specifications **Viewpoints:** List of viewpoints receiving the service **Non-functional requirements:** A set of non-functional requirements that constrain the service

Table 7. Informational template for collecting the component developer – end-user relationship factors

OD-EU template
Cost: Purchase cost that may inhibit the selection of a component for reuse according to budget available from client/user of the final product. **Market:** Market or application area, which a component best supports and executes. **Generality:** Description of the generality, in functionality terms, of a component's services. **Functional characteristics:** A set of functional characteristics a component exhibits that target a specific user-group. **Non-functional characteristics:** A set of non-functional component characteristics that aim at serving best a targeted user-group and contribute to its selection

OD-EU: The template that provides informational factors of this type of relationship is presented in table 7. The factors here are highly correlated with those identified in step 1, and are being utilized to form the required documented "acquaintance" between reuser and component for assisting the process of selecting the most suitable component that meets end-users' requirements.

The three-step methodology described so far offers developers a complete framework to address managerial problems of the development process and to identify a set of critical requirements, both functional and non-functional, for producing qualitative and commercially successful software components. A detailed demonstration of the proposed methodology is presented in the next section.

3 METHODOLOGY DEMONSTRATION

The proposed methodology was applied in the context of an EU funded project (MEDICATE, 2000) that aims at developing a component-based telemedicine system utilizing mobile information devices (phones, palmtops) used by doctors and medical devices connected to patients, and assuring reliable real-time communication between all participants (figure 3). The system comprises: (i) A main file server, a relational database management system and a way for supplying patient information into a database, (ii) An Internet Information Server (IIS) responsible for retrieving data from the RDBMS, translate it to the appropriate XML or WML form, and provide a two-way communication means between the actual data server and all mobile or non-mobile devices through the WWW, (iii) Mobile or non-mobile devices capable of connecting through the WWW to the corresponding IIS. The specific system was fully described via three different component families, based on the ACME Architecture Definition Language (Garlan et al., 1997): Mobile components, standard components and communication components.

Although the target of the project was to develop various software component types for the purposes of the specific telemedicine system, the developers decided to investigate the possibility of building certain commercial components that could be offered to potential reusers at a certain cost and be reused outside the context of the specific telemedicine project in the future. A management/development team was formed, which decided to follow the proposed methodology in order to investigate the management implications of developing such components and perform requirements analysis. For demonstration and validation purposes we will demonstrate the steps of the methodology for two components named C1 and C2 that belong to the mobile component family. C1 is responsible for retrieving information from a RDBMS using XML-like queries and translate the result to various output forms, while C2 receives information from C1 in a proper form and transmits it to the terminal.

Table 8. Critical factors identified for components C1 and C2 according to the “Market”, “Management”, “Generality” axons

Component C1	**Component C2**
1.1 No 1.2 Yes 1.2.1 There are few vendors offering components with the same functionality (six were identified, the major two were analyzed) 1.2.2 *Microsoft:* Complete ActiveX Data Objects (ADO) and Extensible Markup Language (XML) framework within the recent releases of ADO 2.6 and SQL Server 2000. Full Functional, expensive solution, need to have good knowledge of SQL Server *GA Express DOM programming:* ActiveX control that makes transformation of XML-like to queries to Document Object Model tree like structure 1.2.3 Costs range from \$400 to \$1200	1.1 Yes 1.1.1 It’s a totally new solution offering new functional characteristics: Few parameters, quick response 1.1.2 Regardless of the mobile device type this component will produce similar results. This terminal device independency will be achieved through the production of HTML and textual output 1.1.3 Yes, a new standard for sending information to remote devices (PDAs and 2,5th generation mobile phones) 1.2 No
2.1 Approximately one working week or 20 man hours 2.2 Three weeks from today (setting this at the time of the methodology application) 2.3 One person 2.4 Advanced design and programming skills, good knowledge of the DCOM model and other Microsoft development tools 2.5 Approximately 115€ (5,75€ per working hour) plus the cost for the development tools 2.6 Approximately 18€ (or 15% of the initial human resources cost)	1.3 Approximately two working weeks or 40 man hours 1.4 One month from today (setting this at the time of the methodology application) 1.5 Two persons 1.6 Advanced design and programming skills, good knowledge of the DCOM model and other Microsoft development tools 1.7 Approximately 230€ (5,75€ per working hour) plus the cost for the development tools 1.8 Approximately 35€ (or 15% of the initial human resources cost)
3.1 General purpose functionality 3.1.1 Communication using XML schema streams 3.1.2 Independence of the RDBMS engine based on the use of the emerging mobile technology 3.1.3 Mobile communications 3.1.4 The problem is to develop a software standard for mobile data communications and the solution will be the new component itself. 3.1.5 Reusable code, mobile device independency	3.1 Specific problem solver 3.1.1 Provides XML streams to PDAs and text to 2,5th generation mobile phones 3.1.2 First solution, with anticipated high level of effectiveness and reliability 3.1.3 Mobile communications 3.1.4 The problem is to combine the Internet technology with wireless devices. The component will offer the solution to this problem 3.1.5 Offers a reliable wireless communication scheme over the WWW

Developers decided to develop both components using Microsoft tools and specifically C1 using Microsoft Visual Basic 6.0 and C2 using Microsoft Visual FoxPro 6.0, due to the fact that they were highly experienced with those programming tools.

The steps of the proposed methodology for these specific components were performed as follows:

Step1: The critical factors gathered are presented in table 8 for each of the two components previously mentioned (answers follow the questions’ numbering of tables 1 to 3). Summarizing the key points of this table, C1 is a general-purpose component targeting an existing market and requiring minimum human and financial resources, while C2 is a specific problem solver that aspires to set a new standard and form a new market, doubling the requirements in human and financial resources compared to C1. Developers set the following priorities for each component: Since C1 will be entering an existing market, greater emphasis was given to competitive components’ similarities (MR2,2) and their average cost (MR2,3), aiming at offering a new component with at least the same functional features but in a lower and more attractive price. For C2 their attention was drawn on factors describing the creation of a new market (MR1,2 – MR1,3) offering specific problem solving capabilities, with emphasis on quick release (MN2) in order to a timely bid of the market.

Step 2: The FIT for C1 and C2 is given in table 9, marking with “1” the dependencies of the prioritized factors in the C1 case, and with “2” the corresponding factors in the C2 case.

In the C1 case two factors were given the highest priority, the similarities of existing competitive components (MR2,2) and their average cost (MR2,3). The MR2,2 dependencies analysis resulted that this factor primarily influenced the level of skills of the developer's personnel which should be high enough to identify the functionalities of the competition and produce the same services at minimum or more, preserving a low development cost in order for the component's future purchase cost to be lower or equal to that of the competition. The MR2,3 analysis proved that this priority was affecting all MN factors: The average cost of competitive components left no room for producing a new one with higher purchase cost provided that it would offer the same, more or less, functionalities with competition. Thus, development costs should align with this, something that was translated to either reduced human resources with higher level of skills (experienced) but more expensive, or to recruitment of new personnel with lower skills level but lower man-hour payment. Each case, though, would have an undesired effect on the time to develop the component, which would be increased.

Analyzing point 1.2.2 of table 8 for C1, developers characterized Microsoft's solution as a fully functional component that can perform XML parsing as well as tight integration with SQL server database engine. A detailed market-wise view showed that the development of a similar component would undertake a high risk due on one hand to the very successful position Microsoft's component currently holds in the components market, and on the other to the increased costs required for a new component to achieve functional outperformance. The management/developing team after having this clear picture of development implications and risks decided not to proceed with the production of a marketable component, since its purchase costs was estimated to be greater compared to that of the competition, especially in cases in which similar, but with slightly reduced functionality components, were offered for free and the possibility of a reuser choosing a particular component from a pool of similar ones with a certain purchase cost was estimated as minimal. Thus, this component was developed only to serve the specific telemedicine project described earlier and not for commercial purposes.

The C2 case was totally different. The primary concern of developers was the production of a component, which would bring a new service and create a new standard in the mobile communications software components market. The MR1,2 and MR1,3 dependencies analysis revealed strong interaction with the required human resources and their level of skills based on the reasoning that since this would be a totally new solution with nothing similar in the market for using it as a starting point, then probably a significant number of people with high analysis, design and developing skills would have to be hired to undertake these difficult and demanding tasks. Consequently, this would have a significant impact on the development and purchase costs, which were directly related to the quantity and quality of human resources. In addition, factor MR1,3 dictated that a careful business analysis should determine the appropriate time to release the component in the market for setting this new standard. This time-to-market was directly linked to the time bounds of the development process. Since the time-to-market was also regarded as a high priority factor (MN2), then developers had to compromise with any constraint the market analysis would pose as a result of the time-to-market estimation.

Table 9. Functional Interdependencies Table (FIT) for components C1 and C2 marked with "1" and "2" respectively

	MR1,1	**MR1,2**	**MR1,3**	**MR2,1**	**MR2,2**	**MR3,3**	**MN1**	**MN2**	**MN3**	**MN4**	**MN5**	**MN6**
MR1,1	▨			▨	▨	▨						
MR1,2		▨		▨	▨	▨			2	2	2	2
MR1,3			▨	▨	▨	▨	2	2	2	2	2	2
MR2,1	▨	▨	▨	▨								
MR2,2	▨	▨	▨		▨					1	1	1
MR2,3	▨	▨	▨			▨	1	1	1	1	1	1
MN1							▨					
MN2	2	2	2				2	▨	2	2	2	2
MN3									▨			
MN4										▨		
MN5											▨	
MN6												▨

Therefore, at the end of the day this factor determined all the rest, that is, the kind of new services the component would offer and the type of standard it would set, as well as all the rest factors that belong to the "Management" axon, that is, when must development end, how many people should be involved with development and with what skills, and as a consequence of these latter two factors the costs of development and future purchase. Having all these in mind the management/development team reached the conclusion that market conditions were in favor of developing such a component and that potential implications or development risks would be manageable. Hence, the development of the C2 component was launched, starting with the analysis part based on step 3 of the proposed methodology.

Step3: Collection of informational features of C2

OD-R _ developers decided to provide all the services listed in table 5 via a special component storage-retrieving system, which is currently under development. This system is based on an encoded scheme of components' features stored in a relational database and an intelligent retrieving mechanism based on genetic algorithms. Upon retrieving a component the reuser will have access to source code and diagrams, as well as to test cases and failure data.

RE-EU _ The VORD methodology presented in table 6 was utilized to analyze the required services using as viewpoints the preceding component feeding -, and the subsequent component being fed by -, the component under development, seen as a chain of functional components. These two viewpoints were regarded as stakeholder entities and services were defined according to their needs.

OD-EU _ factors were covered through the OD-R relationship informational template.

The application of the proposed methodology proved simple and straightforward in practice, giving a new perspective as regards the way that components can be conceived, analyzed, developed and finally become commercially available to potential reusers. Following the suggested steps the components' developers were able to: (i) investigate the feasibility level of producing a profitable commercial component from the management's point of view and analyze the implications of the development process before the process started, and (ii) identify and incorporate a set of functional and non-functional characteristics in a systematic and controlled way, in the case a decision for proceeding with development of a specific component was taken. In addition, they felt confident that they touched upon significant marketing and management issues and that the component they produced enclosed all prerequisites for being a success in the application domain addressed.

4 CONCLUSIONS – FUTURE WORK

Seeking for quality in software will always be the ultimate goal for a computer scientist or software engineer, no matter what the technical environment for development or the application domain may be. Radical changes in the way we produce software as we move to more efficient, automated and modern development tools and techniques, necessitate a certain transition in the methodology used for measuring and assuring quality. One of these modern ways to produce software applications nowadays is through reusing existing software components assembled to achieve the desired functionality and purpose.

This paper studied issues of analysis and management during the development of new software components, emphasizing on quality features and critical characteristics that contribute to the production of successful commercial components. Our proposition was a simple and effective step-by-step methodology that provides the means for investigating market and management aspects prior to developing components on one hand, and a systematic way for performing requirements analysis of a new software component on another. The proposed methodology begins with defining a critical, raw-basis informational platform for a particular component, based on three axons: Market, management and generality. Factors within these axons are identified and analysed from the management's point of view, addressing potential implications during the development phase and assessing the prospect of commercial success. This analysis reveals the level of dependency between costs, time and human resources according to market conditions and supports developers to decide whether to proceed to developing a component or not. In the case of proceeding to development the methodology suggests a new requirements elicitation and gathering process, with the factors' informational platform already defined serving as the foundation of a model describing the relationships between three entities participating in component-based software development: The developer of a component, its reuser, and the end-user of the software application in which the component is integrated. The proposed methodology analyses each of these relationships to define a set of requirements features, both functional and non-functional, as well as the necessary documenting information, that a component must provide to become more attractive than its competitors and to achieve a significant number of uses.

The methodology was demonstrated and validated through a telemedicine software application based on components. The development of two components was approached through the steps proposed and the most significant result was the management's decision to produce only one of the two components for commercial purposes, due to the fact that the factors' analysis of the abandoned one (first component) suggested that market conditions did not justify the expected development costs/resources risks. The development process of the second component was based on the requirements elicitation scheme proposed and developers expressed their satisfaction having a systematic way of collecting components' functional and non-functional features and their confidence that the software component produced via this methodology reached high levels of quality.

Our future work will concentrate on three directions: First, on completing the components' storing and retrieving software system currently under development. Second, on investigating the determination of the weighting structure that can be used to describe the correlation between each of the axons, as well as the factors defined within each axon. Finally, on employing the proposed methodology in several other case studies, to compare its practical implementation in different application domains.

REFERENCES

Barber, K.S., 1999. Increasing opportunities for reuse through tool and methodology support for enterprise-wide requirements reuse and evolution. *9th Annual Workshop on Software Reuse (WISR'9).*

Baum, L., Becker, M., Geyer, L., Molter, G., 2000. Mapping requirements to reusable components using Design Spaces. *IEEE International Conference on Requirements Engineering (ICRE2000).*

Baum, L., Becker, M., 2000. Generic components to foster reuse. *37th International Conference on Technology of Object-Oriented Languages and Systems (TOOLS-PACIFIC 2000).*

Garlan, D., Monroe, R. and Wile, D., 1997. ACME: An Architecture Description Interchange Language. *CASCON'97.*

Grundy, J., 1999. Aspect-oriented requirements engineering for component-based software systems. *4th IEEE International Conference on Requirements Engineering (RE 1999).*

Grundy, J., 2000. Multi-perspective specification, design and implementation of software components using aspects. *International Journal of Software Engineering and Knowledge Engineering 10(6).*

Kotonya, G., Sommerville, I., 1996. Requirements Engineering with viewpoints. *BCS/IEE Software Engineering J., 11(1), 5-18.*

MEDICATE, 2000. The control, identification and delivery of prescribed medication. *FP5-IST project, IST-2000-27618.*

Mili, A., Mili, R., Mittermeir, R., 1994. Storing and retrieving software components. *16th International Conference on Software Engineering.* IEEE Computer Society Press.

Ncube, C., Maiden, N.A.M., 1999. PORE: Procurement - Oriented Requirements Engineering method for the component-based systems engineering development paradigm. *2nd international workshop on Component-Based Software Engineering.*

Seppanen, V., Helander N., Niemela, E., Komi-Sirvio, S., 2001. Original software component manufacturing: Survey of the state-of-the-practice. *Euromicro Conference (CBSE2001).*

Schach, S.R., (1999). *Classical and Object-Oriented Software Engineering.* McGraw-Hill, London, 4th edition.

Sitaraman, M., Kulczycki, G., Krone, J., Ogden, W.F., Reddy, A.L.N., 2001. Performance specification of software components. *ACM SIGSOFT Software Engineering Notes.*

Sommerville, I., 2000. *Software Engineering*, Addison-Wesley, 6th edition.

Sugumaran, V., Tanniru, M, Storey, V.C., 1999. Identifying software components from process requirements using domain model and object libraries. *20th International Conference on Information Systems.* Association for Information Systems.

Szyperski, C., 1997. *Component Software. Beyond Object-Oriented Programming.* Addison-Wesley.

Wallnau, K.C., Hissam, S.A., Seacord, R.C., 2002. *Building systems from Commercial Components.* Addison-Wesley.