Component-Based Software Engineering Analysis

Yusuf Altunel1 ve Mehmet R. Tolun2

y.altunel@iku.edu.tr
2Prof. Dr., Doğu Akdeniz Üniversitesi, Bilgisayar Müh. Bölümü, KKTC.
mehmet.tolun@emu.edu.tr

Anahtar Kelimeler: Component’e Dayalı Yazılım Mühendisliğinde Çözümleme Süreci.
İlgili Oturum Konusu: Bilgi Teknolojileri

I. ABSTRACT

Component-based software engineering (CBSE) revolutionize the way of developing software. It suggests development by assembling reusable software components rather than developing every line of code from scratch. Although the new approach has its own problems to deal with, the possible advantages are defined as improving the time-to-market and software quality, reduce the effort and costs of software development, and providing a relatively easy maintenance.

Our study has shown us that a methodology in component-based development (CBD) must be recursive in nature. The new approach to software development requires all processes of software development life cycle be investigated under the new requirements of CBD. In this paper we studied the software analysis process from the component-based software development viewpoint. We investigated the classical software analysis activities and evaluated them considering the special requirements of component-based software development. This study has shown us that, there are some new activities and roles that must be defined and some of the classical activities should be modified to satisfy the unique characteristics of component-based development.

II. INTRODUCTION

Software engineering provides no generally accepted methodologies, tools, and approaches to ensure the quality and requirements satisfaction of the final product. Traditionally developed software lacks the quality, developed with too high costs and insufficient measurement, and is hard to be maintained. The CBSE has certain assertions to change the software engineering activities to mimic the way that other engineering disciplines construct their products. CBSE suggests construction of software by assembling components which are reusable, high in quality, easy to integrate, as done in other engineering disciplines in development of complex devices from simple but ready to integrate small-scale reusable devices.

Component-base software engineering activities, similar to the traditional engineering practices provide procedures to be applied to derive productions from the specified requirements2. However, CBSE has its own unique requirements and the software
development style is totally different when compared with structural and object-oriented software development methodologies. In this paper, the analysis activities of component based software development are studied and the necessary tasks to perform in analysing the components and component-based systems are described. The main aim is to provide a clean picture of all component-based analysis activities.

III. SOFTWARE ANALYSIS METHODOLOGIES

The main aim of analysis is to understand and formulate the business characteristics and constraints, identify the environment where the software will be plugged in, the requirements and the functionality of the software system. This is a complex and hard job, requires certain activities, knowledge and experience and therefore the process cannot fully be automated with the current technology. The systems engineering, requirement elicitation, and functional description are the main tasks executed at the analysis. All these activities require high level of communication among the stakeholders.

There are industry-accepted methodologies defined to overcome the difficulties of the analysis process in software engineering. Structured Systems Analysis is an information-oriented methodology starts with understanding the flow of information within the organization. The activities involve defining all externally observable data objects, evaluate the flow and content of the information, define and elaborate software functions, understand the software behaviour, establish system interface characteristics, uncover design constraints.

Object Oriented Analysis (OOA) analysis enables the analyst to represent a software problem by means of objects, attributes, and operations as the primary modelling components. The methods in the area all propose to represent classes and class hierarchies, creating object-relationship models, and deriving object-behaviour models. For example, in UML to formulate the requirements certain models are defined like business process models, use case models, activity models, specialized dictionary, CRC cards, mind maps, business class models.

Comparing with classical input-processing-output (information flow) model the objective of OOA is to develop a set of object-based models contains information, function, and behaviour characteristics to satisfy user requirements with essential principles of object-orientation: encapsulation, object classification, and inheritance. On the other side, structured analysis takes a view of input-process-output approach to satisfy requirements. Data is separated from the processes, transforming the data and behavioural characteristics are second in importance whence the portioning of data flow diagrams and functional decomposition is used heavily.
IV. ANALYSIS IN COMPONENT-BASED DEVELOPMENT

The created analysis model will be used in different levels of component-based development. The design will heavily be based on the results obtained from the analysis activities. The analysis will further affect the architectural design, technical restrictions on implementation of component-based application\(^4\) and in detection of possible reusable components\(^13\). Additionally, the testing activities will use the outcomes obtained from the analysis activities. A careful analysis considering the quality and maintenance requirements will certainly decrease the huge costs and big amount of time required to maintain the software systems and individual components.

Quality and reusability are the main focus of any component-based software development activity. Therefore, the analysis study should include corresponding activities related with reusability and the quality of both the software process and the products that will be developed at the end of software construction. To identify the domain dependent reusable software components, domain analysis must be performed. In this study the domain experts should be employed. Requirement analysis will provide the business and user dependent characteristics expected from the whole component-based system and individual components. In functional analysis the system will be identified in the means of functionality without going deep into design and implementation decisions. Functional analysis is the step where the main components and their sub-components are identified. The defined components are modelled and formally specified by adding interface characteristics. These specifications will be used in component evaluation and market study. The specified components must be searched in the market to find out if purchasing is possible and at which level. The market study ends when a buy-or-develop decision is taken. Market study also will help in deciding whether individual components will be developed for in-house usage only, or market sold; will they be special usage or generic.

In following sub-sections we will discuss the sub-processes that must be identified in analysis of component-based systems.

DOMAIN ANALYSIS

Domain engineering consists of activities to identify the common points of the application domain. This study will help in developing similar applications to achieve the successful software reuse. The possible alternative architectures could be analysed and presented. Additionally, the domain specific generic components are identified and standardized\(^9\). The domain experts must be employed both to analyse the domain characteristics of the problem, and participate to the development in identification and resolving trade-offs between the possible solutions\(^9\).

The activities of the domain analysis would be defined as abstracting the results and providing solutions to the outcomes produced by the requirement analysis. Sorting and structuring the requirements, identifying and describing development units are other activities. The processes and actors interaction are obtained from the use case description and each of the interaction is further divided into sub-activities. Activity
diagrams corresponding to use cases are created to describe the dependencies, parallel processes, decisions and branching points. The activity models would be used later to tailor the individual components for different contexts. Similar or identical activities and sub-activities from different use cases could be recognized and eliminated to prevent redundancies\textsuperscript{19}.

**REQUIREMENTS ANALYSIS**

Requirement analysis is defined as the process of discovering systems, collecting data and identifying the expected software characteristics in the means of functionality, performance, quality and other non-functional characteristics. For component-based development, it is critical to identify the requirements of a component before applying a formal evaluation process. The component requirements will depend on usage context of the component. Quality attribute requirements must exist within the requirements specification in both abstract and concrete forms. Having too concrete specifications might increase the maintenance costs, on the other side, having too abstract specifications would increase the difficulty of objectively measuring the quality\textsuperscript{3}.

**FUNCTIONAL ANALYSIS AND MODELLING**

Functional analysis is the step of defining the software system’s functionality to satisfy the requirements stated previously. A method similar to “Class-Responsibility-Collaborator” (CRC) modeling in object-oriented methodology could be used to identify the possible components and their sub-components. In such a model the followings could be included: component name, component type, component characteristics, responsibilities, data items, subcomponents, collaborated components, and events fired.

Structural characteristics (primitive data items, functions, events, sub-components, interface characteristics—human-computer interfaces, hardware interfaces, and interfaces to other components and connectors--, collaborated components), and behavioural characteristics of each component (states, event-actions, interactions with actors and other sub-systems) and component-based systems must be identified at the functional analysis step. Functional analysis is accepted as an iterative task, so the analysis and models generated must continuously be refined and detailed until the system is found to be satisfactory by the analyst and possible users or customers.

Modelling activities are required to graphically describe the functional, non-functional and behavioural characteristics of components and component based systems to remove possible complications at early steps of the development. The model becomes an important concept used as a means of communication between the stakeholders, and used to forward the collected knowledge to other development teams. The analysis models are further used to generate detailed design models and producing test cases later.

The model developed for component-based software construction will not provide any means of class hierarchies, since components are defined as context-free pieces of
software. So, there will be no generalization-specification relationships between the individual components of component-based analysis model.

NON-FUNCTIONAL DESCRIPTION

Software requirements must be defined in terms of measurable quality characteristics\(^6\). Hence, it is necessary to define a methodology to specify the software characteristics and some metrics to measure the quality objectively. Some quality models are available comprising a definition of software quality and some measurable quality characteristics, but they must be adapted to satisfy the unique requirements of CBSE. For example in ISO 9126 a software quality model is defined, which could be adapted to CBSE. The quality characteristics of the model are defined as: the *functionality*, *reliability*, *usability*, *efficiency*, *maintainability*, and *portability*\(^6\).

To increase the *quality* of the product, at each step of development the necessary quality assurance activities must be applied\(^7\) and analysis is not an exception. Quality assurance will focus on product. To evaluate the component quality some quality metrics should be defined and proven to show that there is a statistical relationship between the quality and metrics.

Functionally decomposing the software system into components and sub-components would help to achieve the *modifiability*\(^3\). Interconnection topology is the set of all interconnections in a system used to exchange information. It is a structural dependency and simplifying interconnection topology helps simplifying the component reconfiguration and allows the system to be modified with minimum impact\(^22\).

The component technology is built on the idea of the systematic reusability. A cost-benefits analysis is required to define the level of the reusability of individual component. The possible future reuse of the component, the availability of the component in market, the marketing possibilities must be investigated. The extra costs required developing components as reusable must be defined and corresponding resource planning must be done. The gains of adding reusability to the component must be worth of it.

*Performance* characteristics of the component and component-based systems are specified in early steps of the development. The performance would be related with response time, efficiency, and hardware resource requirements like memory, CPU time, and other devices.

*Maintainability* cannot be built after the fact. It must be considered during the early stages of analysis and design. In CBSE Component evaluation and selection will directly impact the maintainability of used components. In selecting the reusable components, the components must satisfy the criteria for maintainability and evolvability: Openness of components, extensibility, ease of integration, adherence to standards, availability of source code, open source licensing, interoperability with many different vendors, ease of tailoring to meet local requirements, and run-time support for managers\(^22\).
Component Specification

At the very early steps of development the component definition is ambiguous, defected with over-specification, and insufficient descriptions. Generally the textual descriptions are used and they are sometimes outdated. Formal methods would be useful to locate these defects and clarify them. The improve in component specification and modelling will help to meet specific set of functional, quality and performance objectives.

The formal methods use mathematically defined syntax, and normally, they are not accepted as suitable for software systems in engineering practices. However, the specification would at least be partially coded into formal systems and used in refinement. The functional and behavioral characteristics, timing properties, performance requirements, real-time constraints, security policies, and internal structures would be defined using the formal methods.

Component Evaluation

Evaluation is defined as a new task to maintain the advantages of component-based development to support the reusability. An effective evaluation process should use the decision theory to make the choices quantifiable and strictly control the identical evaluations with directly comparable results.

Evaluation process takes place in two forms. The first one is to evaluate a typical third party product according to the satisfaction of requirements defined in specification activities. All knowledge about the product, human-computer interface, product capabilities; functional and non-functional characteristics must be evaluated.

Second type of evaluation activities is related with buy-or-develop decision and the result will depend on the evaluation. Normally, purchasing must be preferred, since component development, testing, and maintenance costs would easily exceed the costs of purchasing. However, the final decision must carefully be taken due to some risks related with the producer’s stopping the component support in future. Additionally, finding a component to satisfy all criteria is almost impossible. Component licensing and substitution costs related with the future version releases must also be carefully considered in the final decision.

Component Interface Definition

The interfaces of the components, including component-to-component, hardware, operating system, and man-machine interfaces, must all be defined within the analysis process. The interface definitions of individual components will affect the design and architecture of the final system, and used in development of test cases. A component might have more than one interface so that the component would be plugged into several applications. These interfaces must be specified free of component internals to prevent affecting the component usage when the design and development characteristics are changed in future. Adapters or wrappers would be used to separate the interface from the implementation. With this view, the interface is treated as a
separate object and wrapper remains under the control of integrator. As a result, replacing the component will not affect the collaborated components²².

Market Study

Market study is two-fold in component-based analysis. The first one is related with identifying the third-party components in market that would be purchased. In a typical matured market, there will be too many components in different granularity and functionality with different versions and capabilities. Generic components, context or domain specific components will be available and selecting the most appropriate one among them is harder than it is expected. The market study must provide concise, accurate, and up-to-date information about the components available in market. Considering the high costs due to the continuous market study, some brokerage organizations would be employed to maintain the information about the market components².

The second type of market study is necessary for component suppliers, since detailed knowledge of the market is a vital precondition in economical success. A component-based software development that specialized in developing reusable software components to the market must have detailed knowledge on the market. Such a development usually starts with a market study, scanning the market for similar components and analysing the real requirements of potential customers. Depending on the results, the supplier estimates the potential income and the strategic value of the planned component. This assessment will in turn be used to further refine the features of the component in order to find an optimum cost/value ratio¹.

V. RECURSIVE CBD ANALYSIS

Two types of development are defined under component-based approach so far: development with software components and developing components. Our investigation has shown us that both types of development could be unified. The components would be treated as complex software systems and they could further be constructed out of other sub-components. Consequently any software component can be developed applying the same rules that has been applied to developing software systems and the component-based software development task becomes a job of applying the same processes recursively. In this way, the whole system would recursively be constructed by decomposing the software system into the main components and later, for each individual component, accepting the component as a stand-alone new software system that must be developed individually. At each level of granularity, the components must be analysed and decomposed further into their corresponding sub-components until primitive level components are reached. So, after evaluating the alternatives and corresponding risks, the components will be specified and with a careful market study, a decision will be taken to purchase the component, get it as a free-source code, or develop it from scratch (and continue the recursive development). Therefore the defined analysis activities will be applied to each component at each level of component granularity until the constructed software analysis model will satisfy both the development team, and possible customers and users.
VI. CONCLUSIONS

Component-based software development is a new approach to solve some problems of software development by decreasing the costs, increasing the reusability and software quality and shorten the time-to-market. Our investigation has shown us that, a component-based system development and developing reusable software components would be unified using a recursive approach. The methodology helps to increase the number of reusable software components as by-products and would be marketed individually.

We studied the analysis process of component-based software development and identified the basic sub-activities. Later the defined sub-activities are investigated from the point of view of a unified, recursivecomponent-based software development methodology.


19 Bernd Oestereich; *Developing Software with UML, Object-Oriented Analysis and Design in Practice*; Addison-Wesley, 1999.


21 John Speed, Bill Councill, George T. Heineman; *Component-Based Software Engineering As a Unique Engineering Discipline*; in *Component-Based Software Engineering Putting the Pieces Together*; Editors: George T. Heineman, William T. Councill; Addison-Wesley, 2001.