An Approach towards Component Based Software Measurement

Ratneshwer¹ and Divya Gupta²

¹Department of computer Science, MMV, Banaras Hindu,University,Varanasi,India.
²Department of Computer Science Engineering,MMMEC, Gorakhpur, India
¹ratneshwer@gmail.com and ²guptadivya28@gmail.com

ABSTRACT
Conventional software measurement processes proposed in software engineering literature focus on non-CBSE (Component Based Software Engineering) based development of software. It is quite clear that all the software measurement, concerning CBSE, should include parallel measurement; one for the component development process and the other for the component based software development process. An approach for software measurement would be more useful if one designs it keeping in mind the goals of the organization. The conventional measurement models do not consider CBSE in its considerations of measurement and metrics. It is therefore necessary to consider a approach that is based on peculiarities and importance of CBSE and hence a measurement approach based on Goal Question Metric (GQM) paradigm for this purpose is being proposed herein. This initial proposition of such a approach may be purposefully employed by the professionals and the corresponding useful feedback may be analyzed. It calls for further extensive research oriented studies, by all concerned, for perfection of details of the proposed approach.

Categories and Subject Descriptors
D.2.8 [Metrics]: Process Metrics

General Terms
Measurement

KEYWORDS
Component Based Software, Software Process, Measurement, GQM, Requirement Analysis

1. INTRODUCTION
Component-Based Development, an approach to develop a software system with help of reusable software components, has its own challenges, because its parallel development processes, coding style, system life cycle etc., are required to be different from that of conventional software development. Conventional software processes proposed in software engineering literature focus on non-CBSE based development of software. It is quite clear that all the software processes, concerning CBSE, should include parallel processes; one for the development for components and the other for the development with components. Keeping this parallel development in mind, all software processes (development process, management process, software configuration management process and ‘software process management’ process) should be redefined [1]. One can say that the result of CBSE is seen not only in development of software but components also as products of the development process. This quantitative change in nature of the development process and its outcomes would necessary generate various issues related to the measurement in CBSE. Software measurement must be examined to consider problems and possibilities of their applicability in CBSE. It is quite clear that all the software measurement, concerning CBSE, should include parallel measurement; one for the measuring component development process and the other for the component based software development process. A approach for software measurement of CBSE would be more useful if one designs it keeping in mind the goals of the organization. This paper proposes an investigative measurement approach for CBSE based on Goal-Question-Metric (GQM) paradigm. For the discussion here, requirement analysis phase of CBSE has been considered but this approach is applicable for all phases of component based software life cycle. This also presents the activities concerning the characterization of the measurement approach. The aim of this work is to helping process analysts to find useful indications regarding the process performance, process improvement and quality of final product i.e. software components/component based software.

The rest of the paper is organized as follows. In section 2, we briefly describe some GQM applications on CBSE. Section 3 examines the conventional measurement process in context of CBSE. In section 4, we explain the proposed measurement approach for CBSE and activities involved in the approach. In section 5, an example illustration of the proposed approach has been shown. Finally we conclude in section 6.

2. RELATED WORK
An important research topic in software measurement is GQM based measurement. In fact, GQM have been recognized as a leading concept in the software measurement and this area has received increasing attention over the last few years. Various approaches [2], [3], [4] & [5] have been proposed in the software engineering literature to apply GQM approach for conventional software measurement. The software industry later discovered various technologies that aim at identification of repeatable design experiences and items under the name reuse and component based software engineering. Gill et al [6] have discussed the research agenda regarding the measurement of component-based software and to suggest some component-based metrics, which can provide baselines for quality and productivity improvement programs within organizations adopting CBSE. Ali & Ghafoor [7] have given a metrics guided quality management approach for component based software systems. Gill & Grover [8] have suggested few necessary guidelines.
for CBD measurement and proposed some relevant metrics applicable to CBD systems. Very few works have been observed regarding application of GQM approach for component based software and most of them have covered only requirement engineering phase. Alves and Finkelstein [9] have argued that a goal-oriented approach can support an effective balancing between requirements and COTS feature during the decision-making. Carvallo et al [10] have proposed a goal based method named RECSS (Requirements Engineering for COTS-based Software Systems) which aimed at supporting requirements elicitation and analysis in the context of COTS-based software systems. Alves [11] has proposed a goal-oriented approach to identify mismatches in COTS based systems and deal with conflicts. Chung et al [12] have presented the CARE (COTS-Aware Requirements Engineering) Approach, a process-oriented approach which assists the requirements engineer (RE) with the challenging tasks of defining goals, matching, ranking, and selecting potential COTS components, and negotiating changes to the components and/or the SUD. Most of the above efforts cover the requirement engineering related issues; but they do not cover the component based software measurement as a whole. This paper extends the above contributions further by proposing a GQM based measurement approach for the CBSE specifically.

3. MEASUREMENT IN CONTEXT OF CBSE
Measurement is important because a scale is needed to manage the efforts required to accomplish or impose something [13]. The research question of this study is to how effectively one can measure component based software development. Most of the traditional software measurements are applicable to monolithic applications whereas component based measurement should depend mainly on the granularity and interoperability aspects of software components [6]. Traditional measures can be used for CBSSs, provided that the notions of various terms are suitably modified in context of CBSE [14]. The existing software measurements hardly address the adoptability issues, interface complexity issues, interdependence, separate component testing and composability issues that are essentials for component based measurement. As it can be noticed that traditional measures are not specified considering the component oriented concepts and hence software measurement for CBSE should have some different approach. Keeping this parallel measurement in mind, all software measures should be redefined in context of CBSE. The activities of component development and component based software development are very different hence there should be different measures required for components and component based software. For example, the set of activities and there order relations for requirement phase (for software components and component based software) are given below.

3.1 Set of Activities and their Order Relation for Commercial Component Development
Requirement Phase
1. Domain analysis
2. Requirement identification
3. Requirement categorization
4. Requirement communication (requirements negotiation)
5. Description of components
6. Requirement validation
Output: Requirement specification document of a commercial component (having specified requirements and constraints)

3.2 Set of Activities and their Ordering Relationship for CBS Development
Requirement Phase
1. Problem analysis
2. Requirement identification including identification of components brought from outside etc
3. Requirement communication with the client
4. Requirement elicitation
5. Description of identified modules
6. Requirement verification and validation
Output: Specified requirements and constraints of the CBS (Software requirement specification) along with identification of components from component repositories.

It can be inferred from the above two set of activities that activities of component development are different from activities of component based software development hence use of parallel measurement will be more effective in case of CBSE.

4. THE PROPOSED MEASUREMENT APPROACH FOR CBSE
A approach for software measurement would be more useful if one designs it keeping the goals of the organization [15]. Commonly, measurement frameworks are ambitious undertakings that require large data collection and analysis efforts. Unfortunately, such frameworks usually become too extensive, resulting in graveyards with a lot of data never being analyzed and used. One well known way for defining measurements that tries to focus on the most important ones is the Goal Question Metrics approach. GQM focuses on eliciting goals and questions as derivers for finding metrics necessary to collect [15]. Going from goals to metrics has proven successful to assure relevant and used measurement.

As organizations adopt component based software engineering, it becomes necessary to clearly and outline its characteristics, benefits and organizational implications [6]. Measurement frameworks for CBSE should address following questions: how should I size the components, how detailed should I get, how often I should measure and what should I measure [16]. In CBSE, there are two types of organizations; one that develop/manage software components only (for commercial purpose) and the other that develop software systems/products with help of reusable components. Correct measurement of component based software depends on correct measurement of the software components. A CBSE measurement framework should consider this interdependency and also component measurement should be considered a separate activity. The construction of an effective measurement framework has to be based on the identification of a wide variety of metrics and on the definition of consequent approach to collect them.

It can be noticed from the above discussion that the traditional measurement is not specified considering the component oriented concepts of development and hence a measurement framework for CBSE should have some different approach. We hereby propose a measurement framework for CBSE that has integration of two measurement models and the both may work in parallel. The proposed measurement framework has two dimensional approaches. In the first dimension, we define the measurement for software components; the second dimension includes measurement of Component based software.
The set of activities of the proposed framework are as follows. The strategy for deciding the activities is based on a clear vision for improving the software processes and the quality of the product.

1. Goal identification for chosen software development phase
2. Goal classification into software components related goals and component based software related goals.
3. Goal observation from different stakeholder’s view (user and developer)
4. Goal conflict analysis
5. Identification of the questions that must be answered in order to achieve a particular goal.
6. Investigation of the metrics that help to answer the questions adequately.
7. Experimental verification/validation of selected metrics.

See Figure 1: A Generic Measurement Framework for CBSE
The above figure shows the proposed steps for the measurement in CBSE. The steps have to be applied in a disciplined manner in order to elaborate goals and their corresponding questions. The proposed framework can be customized for a specific domain and for a particular software life cycle phase.

The description of the steps in the above framework is as follows.

1. The first step is to identify the goals of interest for the organization at hand. To identify the goals to be achieved, one has to consider the results that an organization wants to achieve and the suggestions of the organization’s management team.
2. In CBSE, component development is considered as separate activity along with Component based software development. The development & management process and objectives of both activities are very different. Hence, goals should be separately identified for component development and component based software development respectively. Categorizing the goals according to this way makes it possible to include goals for specific dimensions and minimizing the risk of goals to be very general covering all dimension.
3. The interpretation of a goal depends on the role of the people associated with it. In process of identification of goals, it is necessary to look the goals from the user’s and developer’s point of view. Multiple views are required because it gives the possibility to prioritize the goals.
4. Some conflicts may arise from user’s and developer’s viewpoints and concerns. These conflicts must be detected and resolved. Based on the conflict analysis, a subset of goals should be selected to be implemented in the measurement framework.
5. This step creates questions, and answers which will help achieve these goals. In GQM paradigm, questions are created to refine and characterize goals. Questions should be designed in way that their answer will characterize a special aspect of their assigned goal. The collected answers of all questions assigned to a goal should provide a clear and usable overview of the goals [17].
6. The next step is to identify metrics for the chosen goals and questions. Metrics describe a distinctive class of data representing a given property of the object to be measured. Because GQM is a goal-driven approach, only those metrics are chosen for a measurement program that actually helps fulfilling the stated goal.
7. In order to understand which metrics can be successfully used to evaluate software components/component based software, an empirical study is required. An empirical study permits the accumulation of the knowledge needed to extract significant conclusions that can be applied in practice.

5. EXAMPLE ILLUSTRATION: REQUIREMENT ENGINEERING MEASUREMENT APPROACH FOR CBSE
To show the example illustration of the proposed framework, we choose requirement engineering phase of component based software life cycle. It can be seen from section 2, that requirement engineering activities of software component development and component based software development are not similar and hence goals are identified separately for requirement engineering of software components and component based software.

Requirement definition must say why a system is needed, based on current or foreseen conditions, which may be internal applications or external market [18]. The advantages of applying the proposed approach on the requirement engineering process are: achieving requirement, completeness, structuring complex requirements, managing requirement conflicts.

5.1 Requirement Engineering Goals for Software Components
Requirement engineering for software component development consists of four phases: requirement identification, requirement refinement, requirement specification and requirement validation. Development of software components for future reuse is an important task in Component-Based software engineering. An appropriate Domain Engineering helps in exploration of common requirements among similar applications for the purpose of identification of reusable components. Refinement deals with elaboration of component’s internal characteristics (function, data and behavior) and non functional criteria that a component should meet. Specification deals with formally specify the component’s interface definition, internal structure, pre and post conditions and other performance criteria. It is important to understand a component in a system context and to validate the scope of software component in that context. In software component development on has to consider both views i.e. the component developer and the future end user. Here goals are identified keeping both views.

(a) Goals of Requirement Identification Phase
User’s View
- Component should meet the business requirements.
- Component should be cost effective.

Developer’s View
- Component should be reusable in similar type of applications.
- Component should be able to work with multiple interfaces.

(b) Goals of Requirement Refinement Phase
User’s View
- Component’s pre-requisite regarding resources and environment should be properly defined.

Developer’s View
• All externally observable data, functions, information flow, design constraints should be explored.

(c) Goals of Requirement Specification Phase

User’s View
- Component Requirement Specification should meet end user desire.
- Component Requirement specification should be easy to understand.

Developer’s View
- Component requirement Specification should be unambiguous and complete.

(d) Goals of Requirement Validation Phase

User’s View
- User and developer should have the same perception of the system.

Developer’s View
- Specification should be complete, consistent and accurate in context of all pre and post conditions, constraints and behavioral domain.

5.2 Goals of Requirement Engineering for CBS

Requirement Engineering for CBS consists of four phases: Component Identification Phase, Requirement Communication (negotiation with client) Phase, Requirement Acquisition Phase and Requirement Validation Phase. Component identification phase deals with identification of suitable components from various repositories. Components advertise the services they offer and can be stored in a repository. This process is driven by the evaluation criteria which take as input high level requirements. The evaluation criteria may include functional and non-functional requirements. Requirement negotiation phase deals with negotiation with client regarding selected components on the basic of cost and quality factors. One of the key aspects of the COTS evaluation is the assessment of how well the COTS alternatives satisfy stakeholder requirements. It is possible that available COTS features are not matched perfectly to organization’s specific needs. Therefore, a main challenge of the selection process is how to handle mismatches between what is desired by the stakeholders and what is possible to achieve with the COTS product [19, 20]. In order to successfully manage mismatches, stakeholders have to prioritize and negotiate unsatisfied requirements. The lack of methods and software tools is all the more surprising given the new opportunities that component-based software engineering offers the requirements acquisition process. For example, stakeholders often have prior knowledge of candidate products during the requirements acquisition phase, so acquisition can focus on the requirements that can be used to best discriminate between competing COTS products, and products can be rejected as relevant new requirement are acquired [21]. It is an important concern in requirement engineering of a COTS based system is to test end user domain requirements vs. the supplier specifications i.e. how exactly are the suppliers’ component specifications compared to the end-users’ domain validation scenarios? The starting point of requirement validation is the observation that also for validation of higher-order domain functionality, only a small subset of the full domain is actually relevant for the end users’ intended automation with distinct effects on utilized system behavior [22]. In software component development one has to consider both views i.e. the component developer and the future end user. Here end user means client of final component based software. Here goals are identified keeping both views.

Proceedings of the 5th National Conference; INDIACom-2011

(a) Goals of Component Identification Phase

User’s View
- Components should be cost effective.
- Components should be efficient.

Developer’s View
- Components should meet the required functionality.
- Components should be reusable.

(b) Goals of Component Negotiation Phase

User’s View
- Component description should be unbiased means it should reveals all risk regarding its uses.

Developer’s View
- Better understanding and quantifying risks early

(c) Goals of Component Acquisition Phase

User’s View
- Chosen components should be maintainable.

Developer’s View
- Requirements should be as measurable as possible to enable effective product selection

(d) Goals for Requirement Validation

User’s View
- Component based software should satisfy the end user’s desire.

Developer’s View
- To validate the suppliers’ component for reuse and control.

The above defined goals are identified based on the understanding and literature study. These goals could be more effective if they are explicitly stated by stakeholders (users and developers) by keeping objectives of the organization. An Organization may have some specific objectives for example real time/embedded software systems may have some additional objectives. Hence goals of such an organization would be different and need special attention in the process of goal identification.

5.3 Conflict Analysis

A conflict link between two goals is introduced when the satisfaction of one of them may prevent the other from being satisfied [23]. Managing conflicts among user's and developer's viewpoints is an important concern of the approach. Conflict analysis helps to decide the priority of goals. Conflicts among goals should be detected before deriving questions and metrics. If during requirement engineering process all the possible conflicts and inconsistencies of goals can properly be analyzed then probability of its performance failure can be minimized. An Application Developer selects a component by matching his requirements, with the Specification Document of that component. Some requirements in Specification Document of the component may have conflicts with the requirements of the Application Developer. It may mean, for example, two terms are used in different contexts to mean the same thing or one and the same term is used at two places for two different meanings. These requirement conflicts may be of the following types: business requirement conflicts, resource related conflicts, architectural requirement conflicts, technology level conflicts, temporal level conflicts etc.

5.4 Deriving Questions from Goals
Once a preliminary set of goals and requirements is obtained and validated with stakeholders, many other goals can be identified by refinement and abstraction, just by asking HOW and WHY questions about the goals/requirements already available, respectively [24,25]. These goals can be refined to questions which help to solve the goals. Questions should be designed in way that their answer will characterize a special aspect of their assigned goal. The collected answers of all questions assigned to a goal should provide a clear and usable overview of the goals. The important criteria while deciding questions for a specific implementation is how good the calculated value semantically represents the fulfillment degree of the goal. So, for example, it does generally not make sense to just summarize all questions’ values and average them somehow as there may be many questions which are just designed to understand the problems domain better and don’t give interesting information concerning the goal fulfillment [17]. Some questions may create new questions in order to understand and fulfill the goals. Questions may be informative types, subjective measure but that also need to be taken care of.

5.5 Deriving Metrics from Questions

The next step is to identify metrics for the chosen goals and questions. In software engineering, metrics usually have three different primary characteristics which can be classified as follows: The object they measure (product or process), the objectivity (objective or subjective) and the directness (direct or indirect) [17]. Because GQM is a goal-driven approach, only those metrics are chosen for a measurement program that actually help fulfilling the stated goal. Every question cannot be quantified into metrics. Some questions may result in subjective measure but that also need to be taken care of.

5.6 Validate Metrics through Experiments

To demonstrate the practical utility of the metrics derived from goals and questions, an experimental study is required. Experimental study will help to obtain some conclusions about the influence of the metrics. Empirically validation of the proposed metrics will prove helpful.

6. CONCLUSION

This approach is applicable for many types of organizations like organizations that develop components only, organizations that develop component-based software or organizations that develop components along with component-based software. Here we do not cover the legal, market and social aspects of CBSE but in latter phases these can be included. The proposed approach may use as a standard for measurement of component-based software and software components.

REFERENCES


Fig. 1: A Generic Measurement Framework for CBSE