Classification of quality attributes for predictability in component-based systems

Ivica Crnkovic[1], Magnus Larsson[2]
[1]*Mälardalen University, Department of Computer Science and Engineering*
Box 883, 721 23 Västerås, Sweden, ivica.crnkovic@mdh.se, http://www.idt.mdh.se/~icc
[2]*ABB Corporate Research, Västerås, Sweden, magnus.larsson@mdh.se*

Abstract

One of the main objectives of developing component-based software systems is to enable integration of components which are perceived as black boxes. While the construction part of the integration using component interfaces is a standard part of all component models, the prediction of the quality attributes of the component compositions is not fully developed. This decreases significantly the value of the component-based approach to building dependable systems. This paper classifies different types of relations between the quality attributes of components and those of their compositions. The types of relations are classified according to the possibility of predicting the attributes of compositions from the attributes of the components and according to the impacts of other factors such as system environment or software architecture. Such a classification can indicate the efforts which would be required to predict the system attributes that are essential for system dependability and in this way, the feasibility of the component-based approach in developing dependable systems.

1. Introduction

Component-based development (CBD) is of great interest to the software engineering community and has achieved considerable success in many engineering domains. CBD, has been extensively used for several years in desktop environments, office applications, e-business and in general in Internet- and web-based distributed applications. In many other domains, for example dependable systems, CBD is utilized to a lesser degree for a number of different reasons. An important reason is the inability of component-based technologies to deal with quality attributes as required in these domains.

In CBD one requirement is that components should be selected and integrated in an automatic and efficient way. This goal is achieved for the functional part; components are selected and integrated through their interfaces. The question is if a similar approach can be applied to quality attributes. For component-based systems crucial questions in relation to quality attributes are the following:

- Given the system attributes, which attributes are required of the components concerned?
- Given a set of component attributes, which system attributes are predictable?
- How can the quality attributes of a system be accurately predicted from the attributes of components which are determined with certain accuracy.
- To which extent, and under which constraints are the emerging system attributes (i.e. the system attributes non-existent on the component level) determined by the component attributes?

These and similar questions have been addressed at a series of CBSE workshops [3], and particular models of certain attributes have been analyzed [5,9], but so far very little work has been done in the systematization and classification of quality attributes in accordance with the questions above.

In this paper, our intention is to analyze the different methodologies which can be used for predicting system behavior from the attributes of the components involved. These attributes can be classified according to the ability of component-based technologies to specify them and provide methods for expressing their compositions, i.e. the ability to predict the attributes of component assemblies. Such a classification indicates the feasibility of the component-based approach for building dependable systems.

2. A Classification Framework

The quality model defined in ISO/EIC 9126-1 “Software engineering - product quality” standard classifies quality attributes as external, visible on the system level, and internal, properties of subsystems and components. Relation between internal and external quality attributes is not unambiguous; an internal quality attribute may have impact on different external quality attributes and of course an external quality attribute is a result of combination of internal attributes.

The classification we consider here is related to composability, i.e. propagation and impact of internal quality attributes to the external. We classify attributes according to the principles applied in deriving the system attributes from the attributes of the components involved. Instead of the term "system", we use a generic term *Assembly (A)* which simply denotes a set of interacting components. Such an assembly can be a part of a software system (for example a functional unit, or a subsystem), or the entire system. The only characteristic we want to relate to an assembly is a set of integrated components. Some attributes, however, cannot be related only to an assembly, but are explicitly related to the entire system and its interaction with the environment. In such cases we refer to a *System (S)*.

According to composition principles we can distinguish the following types of attributes:

a. *Directly composable attributes*. An attribute of an assembly which is a function of, and only of the same attribute of the components involved.
b. *Architecture-related attributes*. An attribute of an assembly which is a function of the same attribute of the components and of the software architecture.
c. *Derived attributes*. An attribute of an assembly which depends on several different attributes of the components.
d. *Usage-depended attributes*. An attribute of an assembly which is determined by its usage profile.
e. *System environment context attributes*. An attribute which is determined by other attributes and by the state of the system environment.

Let us discuss these cases and give examples in the following subsections.

a. Directly composable attributes

Definition: *A directly composable attribute of an assembly is a function of, and only of the same attribute of the components.*

$$P = \text{attribute},\ A = \text{assembly},\ c = \text{component}$$
$$A = \{c_i | 1 \le i \le n\} \qquad (1)$$
$$P(A) = f(P(c_1), P(c_2), \ldots, P(c_n))$$

Note that the attribute of the assembly is the same as the component attribute. Further, the component technology is not explicitly specified in the relation (1). However it is obvious that the function f itself is dependent on the technology since the mechanisms to assemble components is provided by the component technology.

An example of an attribute of this type is the static memory size of a component or an assembly, this is also known as the memory footprint. The simplest composition model is the calculation of the static memory of an assembly as the sum of the memories used by each component. A more complicated model can be found in the Koala component model [10], in which additional parameters, such as size of glue code, interface parameterization and diversity are taken into account (i.e. the parameters determined by the component technology used).

The attributes of this type can be calculated directly from the component attributes and the particular technology. There are no other assumptions and therefore these attributes are the easiest to specify and calculate. This does not mean that the composition functions are easy or even possible to express formally. However the fact that the attribute is visible on component and assembly level, and that the assembly attribute is dependent only on the component attributes simplifies the prediction procedure.

b. Architecture-related Attributes

Definition: *An architecture-related attribute of an assembly is a function of the same attribute of the components and of the software architecture.*

$$A = \{c_i | 1 \le i \le n\}$$
$$P(A) = f(P(c_1), P(c_2), \ldots, P(c_n), SA) \qquad (2)$$
$$SA = \text{software architecture}$$

In this case the assembly attributes depend not only on the component attributes but on the architectural structure. The software architecture is often used as a means for improving particular attributes without changing the component attributes. These types of attributes can be tuned by different architectural solutions or variations.

An example of such an attribute is a performance predictability model for J2EE (Java 2 Platform, Enterprise Edition) application [11]. A typical application implemented in this technology is a distributed web-based application in which the variability in scalability is achieved by it being possible to add new clients and new computational components to the server. To achieve concurrency the components are executed in different threads. A possible extension variation of this architecture is the possibility to include several nodes with web servers and business applications. The performance of the system is related to the number of clients and the number of server components. A typical requirement for such applications is the performance and scalability, i.e. the dependencies between the performance and number of clients and active business components. The form of the relation in [11] shows that it is possible to calculate the optimal number of threads in relation to the number of clients to achieve a minimum respond time per transaction.

c. Derived Attributes

Definition: *A derived attribute of an assembly is an attribute that depends on several different attributes of the components.*

$$A = \{c_i | 1 \le i \le n\}$$

$$P(A) = f\begin{pmatrix} P_1(c_1), P_1(c_2), \cdots, P_1(c_n), \\ P_2(c_1), P_2(c_2), \cdots, P_2(c_n), \\ \vdots \\ P_k(c_1), P_k(c_2), \cdots, P_k(c_n) \end{pmatrix} \qquad (3)$$

P = assembly attribute

$P_1 ... P_k$ = component attributes

In the same way that a function of an assembly is more than the sum of the component functions, there are attributes that are the result of the composition of different component attributes.

An example of such an attribute in a real-time system is the end-to-end deadline (a maximal response time) that is a function of different component attributes, such as worst case execution time (WCET) and execution period

Emerging attributes, i.e. attributes that are pertinent on a system (or an assembly) level but are not visible on the component level are of special interest in this category. For such attributes the major challenge is to identify the attributes of the components that have impact on them.

d. Usage-dependent Attributes

Definition: *A Usage-dependent attribute of an assembly is an attribute which is determined by its usage profile.*

$$P(A, U_k) = f(P(c_i, U'_{i,k})): \; i,k \in N \qquad (4)$$

P = attribute for a particular usage profile

U_k = assembly usage profile

$U'_{i,k}$ = component usage profile

The behavior of an assembly and consequently of a system depends not only on the internal attributes of the components and their composition but also on the particular use of the system. A usage profile U_k which determines a particular attribute P_k must be transformed to the usage profile $U'_{i,k}$ to determine the attributes of the components.

Attributes of this type introduce particular problems as they depend on the use of the system. This means that the component developers must predict as far as possible the use of the component in different systems – which may not yet exist. A second problem is the transfer of the usage profile from the assembly (or from the system) to the component. Even if the usage profile on the assembly level is specified, the usage profile for the components is not easily determined especially when the assembly is not known.

A particular problem with this type of attribute is the limited possibility of reusing measured and derived attributes. If the usage profile is changed, the attributes must be re-calculated or re-measured. An example of such an attribute is reliability which is based on a usage profile. The question arising here is the possibility of reusing previous specifications of the attribute [2].

e. System Environment Context Attributes

Definition: *A system environment context attribute is an attribute which is determined by other attributes and by the state of the system environment.*

$$P(A, U_k) = f(P(c_i, U'_{i,k})): \; i,k \in N \qquad (5)$$

P = attribute for a particular usage profile

U_k = assembly usage profile

$U'_{i,k}$ = component usage profile

The attribute depends not only on the system attribute determined by the usage profile, but also the environment in which the system is used. An example of such an attribute is safety. As the safety attribute is related to the potential catastrophe, it is obvious that in different circumstances, the same attribute may have different degrees of safety even for the same usage profile. We can argue that these attributes are out of the scope of the predictable assembly, but as such attributes are also dependent on component attributes, this relation is important. The analysis approach for such attributes is opposite to the composition; the system environment and the system attributes define the requirements for component attributes.

3. Predictability of dependability attributes

We use the definition of dependability from [1] in which it is defined as the ability of a system to deliver service that can be trusted and the ability of a system to avoid failures that are more severe and frequent than are acceptable to the users. We discuss the dependability attributes [1], namely reliability, availability, safety, confidentiality, integrity and maintainability.

Reliability

The definition of reliability originates from the probability that a system will fail within a given period of time. The probability of failure is directly dependent on the usage profile and context of the module under consideration. One possible approach to the calculation of the reliability of an assembly is to use the following elements [7,8]:

- Reliability of the components – Information that has been obtained by testing and analysis of the component given a context and usage profile
- Usage paths – Information that includes usage profile and the assembly structure. Combined, it can give a probability of execution of each component, for example by using Markov chains.

A model based on these assumptions needs the means for calculating or measuring component reliability and an architecture which permits analysis of the execution path. Component models that specify provided and required interface, or implement a port-based interface make it possible to develop a model for specifying the usage paths. This is an example in which the definition of the component model facilitates the procedure of dealing with the quality attribute. The system reliability can be analyzed by (re)using the reliability information of the assemblies and components (which can be derived or measured).

Availability

Availability is defined as the probability of a module being available when needed.

In the same way as reliability, availability can be obtained by measurements through the usage profiles.

The difference between reliability and availability is that availability is not only dependent of the system but on the repair process, which implies that the availability of an assembly cannot be derived from the availability of the components in the way that its reliability can be derived from the reliability of its components. If availability is treated in a larger context, non run-time attributes must be taken into a consideration. Availability is related to the maintenance and support of the components constituting the assembly.

Safety

Safety is an attribute involving the interaction of a system with the environment and the possible consequences of the system failure. I is a system attribute, neither a component nor an assembly attribute. Its safety depends on where and how the system is deployed. Since safety is a system attribute that is dependent on the system's environment, a means for analyzing safety is a top-down approach, a decomposition rather than composition. In the analysis process, the components' attributes are used as selection criteria or are identified as demands that should be met. For this reason a component-based approach might not have the apparent advantage – on the contrary, if the starting idea is a reuse of existing components, the components' attributes cause new constraints and in this way might decrease the system safety. However, when the constraints are identified and unambiguously related to the constraints on the system level, the system safety can increase. Also, some attributes, such as reliability, might improve the accuracy of the system safety prediction, especially if known or measured when used in other applications.

Confidentiality and Integrity

Security aspects, confidentiality and integrity, defined as follows apply to dependable systems.

- Confidentiality is defined as a measure of the absence of unauthorized disclosure of information;
- Integrity is defined as the absence of improper system state alterations.

From the definitions it is apparent that these attributes are not directly measurable and composable, and this is the main obstacle to the development of a theory for their prediction.

Confidentiality and integrity are emerging system attributes that can be tested and analyzed on the system and architectural level but not on the component level. Usage profiles can be used for testing and analysis, but it is impossible to automatically derive these attributes from the component attributes.

Maintainability

Maintainability is related to the activities of people and not of the system itself. Component technologies might provide support for dynamic upgrading/deployment of components which can improve the maintainability of a system. In this case the maintainability is much a matter of component technology, and not of the component itself. The system architecture thus has an impact on maintenance. There are many parameters that can be measured and then used to estimate the maintainability of a code (for example McCabe Metrics for complexity [19]). These parameters can be identified for each component. It is however not clear how these parameters can be defined on the assembly level. One possibility is to define a mean value of all components normalized per lines of code.

4. Current State of the Work

To demonstrate the classifications we have provided an extensive list of quality attributes and evaluated the classification by performing a survey on twelve researchers [4]. In 67% of the answers we have the same classification as the researchers. Our classification is performed under careful consideration with the classification definitions in mind. The researchers did not have the same background information. They were not overly confident in their answers; around 60% answers they have confidence on the low side, mainly because of vague definitions of quality attributes.

We have also demonstrated procedures for certain attributes (latency and configuration consistency) [4].

For dependable systems the questions are focused on which attributes should be part of components specifications that can be used for compositional reasoning and also for a top-down analysis. In particular we focus on automotive domain in which we build predictable enabled component model [6]. In the current phase the model is being developed, in which the safety and reliability quality attributes are being concerned. The purpose of the research is to examine the ability of component-based approach to be used for safety-critical systems in the automotive domain.

5. Conclusion and Future Work

A full advantage of component-based approach will be achieved when not only the functional parts are reused, but also when this approach leads to easier and more accurate predictability of the system behavior. In component-based approach, many system attributes can be derived from the component attributes, this being more accurate if a support for defining and measurements of the attributes are built in the component technologies and if there are well defined restriction rules n using these technologies.

The quality attributes can be classified with respect to types of composition, in which each type is characterized by the required input for obtaining predictability on the system level. Some types show clear composable characteristics, while others are not directly related to compositions.

In spite of diversity of attributes, technologies, and theories, it should be possible to create reference frameworks that by identifying type of composability of attributes can help in estimation of accuracy and efforts required for building component-based systems in a predictable way.

6. References

[1] Avižienis A., Laprie J.-C., and Randell B., "Fundamental Concepts of Computer System Dependability", In *Proceedings of IARP/IEEE-RAS Workshop on Robot Dependability: Technological Challenge of Dependable, Robots in Human Environments*, 2001.

[2] Crnkovic I. and Larsson M., *Building Reliable Component-Based Software Systems*, ISBN 1-58053-327-2, Artech House, 2002.

[3] Crnkovic I., Schmidt H., Stafford J., and Wallnau K. C., "5th Workshop on Component-Based Software Engineering: Benchmarks for Predictable Assembly", In *Software Engineering Notes*, volume 27, issue 5, 2002.

[4] Magnus Larsson, *Predicting Quality Attributes in Component-based Software Systems*, Mälardalen University, 2003.

[5] Moreno G. A., Hissam S. A., and Wallnau K. C., "Statistical Models for Empirical Component Properties and Assembly-Level Property Predictions: Toward Standard Labeling", In *Proceedings of 5th Workshop on component based software engineering*, 2002.

[6] SAVE, *SAVE, Component-based design for Safety-critical vehicular systems,* 2004., http://www.mrtc.mdh.se/SAVE/

[7] Schmidt H., "Trustworthy components: compositionality and prediction", *Journal of Systems & Software*, volume 65, issue 3, pp. 215-225, 2003.

[8] Schmidt H. and Reussner R. H., " Parametrized Comtracts and Adapter Synthesis", In *Proceedings of 5th ICSE workshop on CBSE*, 2001.

[9] Stafford J. and mcGregor J., "Issues in Predicting the Reliability of Composed Components", In *Proceedings of 5th workshop on component based software engineering*, 2002.

[10] van Ommering R., "The Koala Component Model", in Crnkovic I. and Larsson M. (editors): *Building Reliable Component-Based Software Systems*, ISBN 1-58053-327-2, Artech House, 2002.

[11] Yan L., Gorton I., Liu A., and Chen S., "Evaluating the scalability of enterprise javabeans technology", In *Proceedings of 9th Asia-Pacific Software Engineering Conference*, pp. 74-83, IEEE, 2002.