Issues of CBD Product Quality and Process Quality

Mark Woodman, Oddur Benediktsson
School of Computing Science
Middlesex University
Trent Park
London N14 4YZ, UK
m.woodman@mdx.ac.uk,
o.benediktsson@mdx.ac.uk

Bruno Lefever
Computer Associates
Bvd. de le Woluwe, 34
1200 Brussels,
Belgium
bruno.lefever@ca.com

Friedrich Stallinger
Systems Engineering & Automation
Kepler University Linz
Altenbergerstraße 69
4040 Linz, Austria
fs@sea.uni-linz.ac.at

Abstract

This position paper presents arguments for including the properties of processes involved in various approaches to component-based software development in predicting system properties. It discusses how processes impact on system properties and relates the issues raised to standards that already address process and product quality. Although many standards still apply, CBD changes interpretations and emphases.

1. Introduction

The starting point of this position paper is the long-established domain of software process quality and capability assessment. The paper aims to contribute to the discourse on predicting properties of component-based systems from the properties of their parts, recognising that in many engineering contexts the maxim applies that process quality is neither necessary nor sufficient for product quality. We argue that many of the processes involved in component-based software development affect the quality of the systems produced: predicting the properties of such systems is complicated by these newly emphasised software development activities and several questions must be addressed before using component properties as predictors for system properties. Here we enumerate several of the CBD issues we have confronted and suggest how the particular processes impact on the prediction of component-based system properties.

The work described here is part of the EU-funded project, OOSPICE [11], whose goals include the specification and validation of a generic model for CBD, the specification, tooling and user trials of a conforming CBD method, and the extension of the ISO 15504 international suite of standards on software process assessment [8] to cover object-oriented and component-based software development. The OOSPICE partnership[i] includes European and Australian representatives of academe, the software industry and users of components and process improvement schemes.

2. Quality of CBD systems

There is a large variety of software development approaches that can justifiably claim to be component-based. To develop rules for predicting the properties of a system developed from components necessitates looking at the variety of approaches and variety of ways that components appear in systems. For example, the SEI vision of CBD, as expressed by Bachman *et al.* [3], states that the (certified) properties of components *and* component frameworks be used to predict system properties. However, nothing is said about predicting system properties when components are compositions or adaptations of other components. Furthermore, nothing is said about predicting the properties of frameworks that are likely to be composed from a variety of pieces, or indeed components. Bachman *et al.* also require a component model for component-based systems but say nothing about how its properties may be used in the prediction of system properties. We argue that these and similar omissions by others are evidence of the implicit impact of development processes which dictate such practices as component composition, adaptation, etc. So prediction of system properties must involve determining process properties.

In fact the processes involved in CBD can change the meaning of traditionally defined quality attributes. For example, Allen [1] tabulates ten CBD-related "ilities" for which he suggests IT and business benefits. They are: accuracy, adaptability, <u>clarity</u>, <u>flexibility</u>, interoperability, maintainability, performance, replaceability, <u>reusability</u>, <u>scalability</u>. The underlined items lie outside the list of quality characteristics and subcharacteristics as listed in the ISO 9126 standard [9], which is the basis for many standard approaches relating to software quality. The terms above are subject to many interpretations. To make progress with predicting system properties requires unambiguous semantics and a more extensive list. However, Allen does expose the difference CBD makes. His characterizations are tailored to the context of CBD and they reflect a shift in what the terms traditionally mean, for example, as listed in ISO 9126. Moreover, new business goals are introduced by CBD that have contributed to its popularity. For example, "time-to-market" may be an overriding business goal implying that the reusability attribute is at the top of the list.

A particular software process, or every actual enactment of such a process, may influence the way component properties are interpreted (e.g. by applying different weighting to a quality attribute). Alternatively, a process may change the focus for predicting a system quality attribute, for example, from the system's components to the producers of its components. In that case, the *capability* of the main processes employed to produce the component or even the *maturity* of the organization (e.g. as per ISO 9001 certification) that produced the components may be of great importance for predicting system properties.

3. The wide spectrum of CBD

There is a plethora of software development experience that is claimed to be component-based. Considerable work is being carried out to find abstract representations of the methods used in CBD and OOD as is exemplified in the work of Henderson-Sellers and Unhelkar [7] and others. A simple taxonomy is not possible. The variety of CBD approaches can be viewed as space of instantiations derived from a common (meta)model. Somewhere in that space are those approaches that package objects (e.g. as advocated by D'Souza and Wills [6]).

Elsewhere in that space may be an e-business strategy [1]: "e-Business improvement planning is an incremental and continuous process that recognizes that IT is now integral to the success of the business strategy." Some approaches are centered on the use of COTS/MOTS/GOTS but these may occupy quite different spaces; cf. the phase-ordered view summarized by Cai *et al.* [4] with the view of long term involvement of component vendors represented by Morisio *et al.* [10]. Other CBD approaches emphasize corporate re-use policies (e.g. the Select-Perspective approach). Not only do the high-level characterizations of CBD influence process properties and hence product properties, the way individual projects enact a particular approach matters.

Some experts do more clearly acknowledge the impact of processes. Bosch [2] takes a product-line approach, which emphasizes system architecture over components *per se*; he explicitly draws attention to the tension between what might be seen as a purist view of CBD and a pragmatic view. (He characterizes these as "academic" and "industrial" views; his terms may be tongue-in-cheek but serve to illustrate different but valid interpretations.) The distinction is important in the context of predicting properties of a system from the properties of its components. A view of components as "black boxes" that are accurately specified both in terms of behavior and service level together with certification rules promises a tractable calculus for determining properties of compositions and adaptations (i.e. extensions or constraints) of third-party components. Hence a quality assessment approach like ISO 15504 can be extended to this end. However, a view that admits components to be "open boxes" requires stringent assessment of the development processes for such a calculus to work.

Furthermore, the requirement of Bachman *et al.* [3] that component frameworks be taken into account is crucial. In his summary of the "industrial" view of CBD, Bosch characterizes the approach to frameworks and glue as ad hoc. Our discussions with industry make us prefer the term "pragmatic". Many companies are both constrained and empowered by what is immediately available, by legacy systems and by their expertise in what may be called "application integration". This expertise often results in framework reuse at the *specification level*; a particular system may consequently have many parts with many technologies, e.g. CORBA, EJB, MQSeries. Companies see a high potential of reuse in the application integration approach. This is independent of the underlying component framework. For example, an ERP package may be used with, say, HP Unix, or IBM's CICS; these are different component frameworks for component implementations offering identical, or at least similar, component specifications. Any combination of, for example, an EJB-framework component specification and the chosen ERP one would be reusable. This is because there are several component implementations (each running on their own framework) available for the same specification. In other words, the middleware layer in the application integration world offers flexibility regarding new combinations of smaller-grained component frameworks into bigger ones via generic interfaces. The pragmatic engineering approach taken by many companies is still very much component-based but includes complex, though well practised, processes. This is further evidence that process properties must be factored into the prediction of component-based system properties.

4. Relating component quality to product quality

Following Allen [1] business goals translate into product quality attributes. For example the goal "short time-to-market" is highly dependent on the reusability attribute of a component-based product. Similarly the goal "reduced maintenance cost" depends on the maintainability attribute of the product.

In general we suppose that the "quality-in-use" attributes of a product (e.g. desired functionality) are influenced by external quality attributes (e.g. requirements quality) that in turn are influenced by internal quality attributes (e.g. design quality) [9]. Internal quality attributes are influenced by the software process quality (e.g. verification activities).

The ISO 15504 standard provides a process assessment model: "Part 2 ... defines a two dimensional reference model for describing processes and process capability used in a process assessment. The reference model defines a set of processes, defined in terms of their purpose and outcomes, and a framework [sic] for evaluating the capability of the processes through assessment of process attributes structured into capability levels" [8]. In the standard the requirements are given for performing an assessment in a consistent way. An assessment is performed on a given process instances. We designate the outcome of an assessment for a particular process the *process capability profile*. The capability profile will summarize the ability of the process to perform on the defined capability levels. There are six capability levels enumerated 0 to 5 and named incomplete process, performed process, managed process, established process, predicted process, and optimizing process.

Let us suppose that a component-based product is to be constructed mostly from ready-made components together with the needed glue-ware. Some non-functional quality attributes of the product are stated such as maintainability, performance, and reliability. Can the system quality attributes really be predicted from such attributes of the constituent components? The answer depends on what we know about the constituent components. If we know, for example, reliability measures for individual components and the component usage pattern then the overall reliability can be computed [14]. The same holds for performance. An attribute such as maintainability is a different story. Maintainability attributes such as changeability and stability can be quantified but it may be difficult to deduce overall maintainability for a system from that of the individual components.

In some cases we may want to obtain knowledge about the software process capability of the component vendor. Suppose that a software developer is working with a component vendor on a long-term basis and that the delivered components are of critical importance. The developer might then do well in asking the vendor for process assessment of some of the pertinent processes such as the development process, maintenance process, configuration management process, and quality assurance process.

To conclude this abbreviated argument, we briefly examine eleven potential CBD quality attributes – Allen's ten [1] plus reliability – and postulate the effect a CBD view has on "classical" software processes. The ISO 15504 process reference model will be used as a baseline (a version is published in [15]). The considerations are made with a component user (i.e. the component-based system designer) in mind. Putative CBD process names are be italicized.

1. **Reusability.** Increased reuse has long been seen as a major business goal as is reflected by the *reuse process* in ISO 15504. CBD elevates reuse to new heights. This will change activities in all process categories for example *acquisition* and *requirement elicitation* in the customer-supplier process category, *analysis and design* in the engineering process category in the supporting life cycle process category, and *risk management* in the organizational life cycle process category.

2. **Maintainability.** Reduced maintenance costs is seen as a major benefit of CBD. Ideally, "defect-free" components are available and are distributed by vendors that enhance functionality as needed. In a distributed component-based system the *system and software maintenance process* changes because of the involvement of the component vendor. So will some other processes such as *configuration management.*

3. **Accuracy.** The capability of the component to provide the agreed effects and with needed precision must be designed into the component. Here *component certification* [4] comes into play. It is a new process in relation to the current ISO 15504 reference model [8].

4. **Clarity.** Rigorous and clear interface specification is mandatory for successful CBD. Again *component certification* is called for to ascertain the clarity as seen from the component user.

5. **Replaceability.** A component needs to be easily replaceable with another component having different implementation code. Accuracy and clarity of the interface is a key issue here as well as the way the component utilizes the underlying component framework. Again *component certification* is called for.

6. **Interoperability.** The capability of a component to interact with other components and the component framework. From the component usage point of view interoperability is an attribute of highest importance. Component interoperability is concluded from the specifications. The main concern of the component user is its adherence to the specifications. The component user (i.e. the acquirer) may need to get involved in the specification activity and again *component certification* is called for.

7. **Scalability.** The property of scaling-up smoothly is well known in many domains, e.g. database systems. In distributed CBD systems unexpected bottle-necks may show up. Some new activities may need to be added to *testing* to validate needed scalability.

8. **Performance.** The component needs to provide appropriate performance under peak load relative to the amount of allocated resources. Again *component certification* is called for to validate stated performance attributes.

9. **Flexibility.** The component needs to be liberated from "one-track" development and deployment. Some new activities may need to be added to *testing* to validate needed flexibility.

10. **Adaptability. "**Components can be reused and extended in a "plug and play" fashion in different business contexts." [1] Some new activities may need to be added to *testing* to validate needed adaptability.

11. **Reliability.** The component needs to provide a specified level of capability to avoid failure. In safety critical systems needed reliability levels are stated in terms of frequency of failure. CBD of highly reliable systems calls in the first place for components of designated reliability and secondly that the resulting system meets the reliability requirements. Again *component certification* is called for to validate the stated reliability attributes of components and component framework. In addition the system builder must make sure that the needed glue-ware and adaptations preserve the reliability of the constituent components and component framework. This naturally leads to the requirement that the deployed processes need to be of "adequate" capability to provide for effective software development of high quality software.

5. Conclusion

Supported by observations that particular processes involved in CBD can change the meaning of traditionally defined quality attributes or alter the focus for predicting a system quality attribute, we claim that a series of questions must be addressed before using the respective component properties as predictors for system quality properties. Furthermore, we argue that this prediction must involve determining *process* properties. The development of rules for predicting the properties of a system developed from components based on this hypothesis necessitates looking at the variety of approaches of software development claimed to be component-based as well as the variety of ways that components appear in systems. In short: we must account for actual CBD processes in predicting a component-based system's properties.

6. References

1. P. Allen, *Realizing e-Business with Components*, Addison-Wesley, 2001.
2. J. Bosch, *Design and Use of Software Architectures*, ACM Press, Addison-Wesley, 2000.
3. F. Bachman, L. Bass, S. Commela-Dorda, F. Long, J. Robert, R. Seacord, K. Wallnau, Volume II: *Technical Concepts of Component-Based Software Engineering*, Software Engineering Institute, CMU/SEI-20000TR-008, May 2000. http://www.sei.cmu.edu/pub/documents/00.reports/pdf/00tr008.pdf
4. X. Cai, M.R. Lyu, K. Wong, Component-Based Software Engineering: Technologies, Development Frameworks, and Quality Assurance Schemes, *Proceedings APSEC 2000, Seventh Asia-Pacific Software Engineering Conference*, Singapore, December 2000, pp 372–379.
5. J. Cheesman, and J. Daniels, *UML Components*, Addison-Wesley, 2000. (www.umlcomponents.com)
6. D.F. D'Souza, A.C. Wills, *Objects, Components, and Frameworks with UML*, Addison-Wesley, 1998.
7. B. Henderson-Sellers, B. Unhelkar, *OPEN modeling with UML*, Addison-Wesley, 2000.
8. ISO/IEC TR 15504–1:1998(E) *Information technology – Software process assessment*.
9. ISO/IEC JTC1/SC7 N2228, FDIS 9126–1 *Software Engineering – Product quality – Part 1: Quality model*.
10. M. Morisio, C.B. Seaman, A.T. Parra, V.R. Basili, S.E. Kraft, S.E. Condon, Investigating and Improving COTS-Based Software Development Process, *Procs. ICSE 2000, 22nd International Conference on Software Engineering*, Limerick, June 2000, pp 32–41.
11. OOSPICE Web site at http://www.oospice.com/
12. SEL, *COTS Study, Phase 1 Initial Characterization*, Software Engineering Laboratory Series, SEL-98-001. http://sel.gsfc.nasa.gov/website/documents/online-doc/98-001.pdf
13. C. Szyperski, *Component Software*, ACM Press, Addison-Wesley, 1999.
14. S. Yacoub, B. Cukic, and H. H. Ammar, Scenario-based Reliability Analysis of Component-Based Software, *Procs. of the 10th International Symposium on Software Reliability Engineering, ISSRE 99*, Boca Raton, Florida USA, November 1–4 1999, pp 22–31.
15. S. Zahran, *Software Process Improvement*, Addison-Wesley, 1997.

[i] The partners in OOSPICE are the Department of Systems Engineering and Automation, Kepler University Linz, Austria; Middlesex University, London; University of Boras, Sweden; Center for Object Technology and Application Research, University of Sydney; Griffith University, Australia; Computer Associates, Brussels; Dataservice Informatik Ges.m.b.H, Austria; Huber Computer Datenverarbeitung GmbH, Austria; and Volvo Information Technology, Sweden.