Software Component Evaluation: A Theoretical Study on Component Selection and Certification

Alexandre Alvaro[1], Rikard Land[2], Ivica Crnkovic[2],
[1]*Federal University of Pernambuco and C.E.S.A.R – Recife Center for Advanced Studies and Systems, Brazil*
[2]*Mälardalen University, Department of Computer Science and Electronics, Sweden*
alexandre.alvaro@cesar.org.br, {rikard.land, ivica.crnkovic}@mdh.se

Abstract

Software components need to be evaluated at several points during their life cycle, by different actors and for different purposes. Besides the quality assurance performed by component developers, there are two main activities which include evaluation of components: component selection (i.e. evaluation performed by the system developer in order to select the best fit component to use in a system) and an envisioned component certification (i.e. evaluation made by an independent actor in order to increase the trust in the component). This paper examines the fundamental similarities and differences between these two types of component evaluations and elaborates how these fit in the overall process views of component-based development for both COTS-based development and software product line development.

1. Introduction

Component-based software development has emerged as a viable and economic alternative to the traditional software development process [35]. The ability to build complete system solutions by interconnecting components through public interfaces independently created and deployed components is the driving force behind Component-Based Software Engineering (CBSE). However, organizations reusing existing software components can only achieve the improvements related to software reuse if the selected software components have a certain quality degree.

A major problem when building software systems from components is the unknown quality of the components and the unknown side-effects of their integration. In our view, component evaluation has to be performed at different stages in the component life cycle, by different actors, for different reasons. During *component development*, the component vendor assures the quality of the components developed before made publicly available for reuse. *Component certification* means that an independent actor performs an evaluation according to standardized procedures, so that an issued certificate is seen as a quality stamp which increases the trust in the component. During *system development*, components are evaluated with respect to system requirements, and selected in a *component selection process* in order to select the components that best fit the target system.

Apart from the quality assurance performed by the component vendor (as is done for any product developed by any vendor), in the scenario outlined above components are evaluated both in order to *select a component* to use in a system and in order to *certify a component* (i.e. assuring some properties of the component). This paper examines the fundamental similarities and differences between these two types of component evaluations. The contributions of this paper are: 1) a review of previous literature studies of software component certification and selection, 2) an examination of to what extent similar or identical methods and practices can be used in the two processes, 3) a discussion about the fundamental theoretical limits of what can be evaluated, and 4) an elaboration of how the two processes of *component selection* and *component certification* should be designed to interact with the processes of *component development* and *system development*.

The remainder of this paper is organized as follows: Section 2 describes the work related to this study. Section 3 describes the envisioned overall process view of component-based development, and Section 4 examines the similarities and differences between the component evaluation performed during component selection and component certification. Section 5 presents the main challenges related mainly to the establishment of component certification, and Section 6 presents the concluding remarks and directions for future work.

2. Current State-of the-Art and Sate-of-the Practice of Component Evaluation

The focus of this paper is an overall CBSE process view with focus on component evaluation during component certification and component selection, and there are three types of related works:

1. Overall CBSE processes, especially if they combine component certification and selection (section 2.1).
2. Summaries of research in component selection and component certification respectively (section 2.2).
3. Actual works detailing out principles and methods of component selection and component certification respectively (section 2.3 and 2.4).

2.1. Overall processes for Component-based Development

To our knowledge, there are no publications combining the current knowledge of component selection and component certification as an integrated part of component-based software development (CBD) process. Several works discuss the lifecycle process divided into (i) *component development*, (ii) *component selection* and (iii) *system development* processes [3, 11, 21, 23, 28] in which component certification is treated implicitly or not at all. There are some works that relate component-based development processes to a certification process. In [36] the cooperation between component development and component certification are considered, and in [34] it is discussed whether a certification process is an (un)necessary part of component-based development. However, our claim is the necessity of recognizing *component certification* as a separate process, but integrated into the overall CBD process, and our aim is to compose these four processes in order to provide a cooperative process that work together in order to provide quality aspects around the component's and system's development activities.

2.2. Reviews/Surveys of Component Selection and Component Certification

There was a workshop in 1997 organized by the SEI, where many of the principles of component selection were first put in print [30]. We have made a survey of published methods for component selection [20], contributing with a meta-model covering all the essential parts of the existing methods (this survey is further related in section 2.3) Apart from our own survey of component selection methods, there are only a few publications including some very limited surveys [3, 27, 32] (limited in the number of methods surveyed, and in the dimensions compared).

In addition, we also have provided a survey about component certification [1] that covers the most relevant work in this area. Although there are no other component certification surveys, there is some work related to component certification process [7, 36] where the main idea is to provide a well-defined component certification process. Besides, these works are in their beginning and there is a need for more comprehensive evaluation (in both academia and industry). Also, the certification process was not developed within the component development and/or system development contexts, which means they may be difficult to apply in a real process scenario.

2.3. Component Selection

Since many software systems are built using components, evidently components are in practice evaluated and selected. This section summarizes the main findings from our previous survey of published systematic methods for component selection [20].

The first method to be published (in 1995) is OTSO (Off-The-Shelf Option) [18], which introduced the fundamental idea of progressive filtering, evaluated not only functional and non-functional properties of the component, but also strategic considerations and architecture compatibility, and suggested AHP (Analytical Hierarchy Process) for comparison of candidates. In PORE (Procurement-Oriented Requirements Engineering, 1998) [23, 28], components are evaluated while the system requirements are developed, which is also true for RCPEP (Requirements-driven COTS Product Evaluation Process, 2001) [21], CRE (COTS-Based Requirements Engineering, 2002) [3], and CARE [5] (COTS-Aware Requirements Engineering, 2004). STACE (Socio-Technical Approach to COTS Evaluation, 1999) [19] stressed the importance of non-technical factors to evaluate.

CSCC (Combined Selection of COTS Components, 2002) [32] considers the total cost for a system rather than specifying in advance the individual costs for different components. *CCCS* (Compatible COTS Component Selection, 2005) [4] consider sets of complementary component as candidates, focusing on how well components will fit together.

i-MATE [22] focuses on middleware selection, and the main contribution is the description of reusable requirements for that domain.

PECA (Plan, Establish, Collect, Analyze, from the SEI) [7] can be noted for its flexible structure of activities, and *RDR* (Requirements and Design Reviews, developed and used at NASA Goddard) [25] describes well the relation between acquired components and system parts being built in-house.

Typically, the suggested processes can be described as a progressive filtering [30], where all found components are first evaluated at a high level, using easily accessible information about components, for example from vendors' marketing material. For convenience, we label this phase *high-level evaluation*. Also, other criteria such as costs in the short and long term, vendor stability are evaluated. After this phase, some components are

discarded and the remaining are evaluated more thoroughly by experimenting and prototyping using the real components (*prototyping evaluation*), after which one or a few components are recommended or selected. This is true although some methods describe this in terms of fixed phases, others as progressively more detailed iterations or as a flexible or opportunistic structure of activities.

The main distinction between *high-level evaluation* and *prototyping evaluation* is whether the component needs to be available during the evaluation or not, which have a big impact on the cost and time (and skills) required for the evaluation and consequently impacts the number of components that can practically be evaluated. In *prototyping evaluation*, the acquisition of a component may come with a (high) cost and can introduce (long) delays in the evaluation process, and the evaluation itself requires learning the component and systematically setting up, executing, and documenting many tests thoroughly.

The published methods differ in how the actual selection of a component is performed. There are different opinions: some processes are based on a formal method for comparison and ranking, such as AHP (Analytical Hierarchy Process) or weighted scores [3, 18, 19, 21, 23, 28]. Others suggest that a formal comparison runs the risk of not catching the intent of the comparison, and emphasize discussions and reasoning [7, 23, 28]. Gap analysis is a tool which helps focusing on the total cost, time, and risk of each alternative [20, 29].

2.4. Component Certification

Component certification is a method to ensure that software components conform to well-defined standards; based on this certification, trusted assemblies of components can be constructed [8]. However, this task seems to be very difficult because the software engineering community has expressed many and often divergent properties to evaluate software components [33].

In addition, existing literature is not that rich in reports related to practical software component certification experience, but some relevant research explores the theory of component certification in academic scenarios. The component certification history can be "divided" into two periods [1]: from 1993 to 2001, where the focus was mainly on mathematical and test-based models and, after 2001, where the focus was on techniques and models based in predicting quality requirements.

The first period was focused on methods of component certification using modeling techniques, making possible not only to certify components but to certify the system containing the components as well [40, 41].

Another technique was presented by Voas [37], where he defined a certification methodology using automated technologies, such as black-box testing and fault injection to determine whether the component fits into a specific scenario.

The state of the art, up to around 1999, was that components were being evaluated only with the results of the tests performed to the components. However, such testing had no well-defined way to measure the efficiency of the results. In 2000, Voas & Payne [38] defined some dependability metrics in order to measure the reliability of the components, and proposed a methodology for systematically increasing dependability scores by performing additional test activities.

In 2001, Morris et al. [26] proposed an entirely different model for software component certification. The model was based on the tests that developers supply in a standard portable form. So, the purchasers can determine the quality and suitability of purchased software.

Besides these contributions, the main advance achieved in this period was the fact that component certification began to attract attention and started to be discussed in the main CBSE workshops [9, 10].

After a long time considering only tests to assure component reliability levels, around 2000, the research started to change focus and other issues began to be considered in component certification, such as reuse level degree, reliability degree, component predictability properties, among other aspects.

In 2001, Stafford & Wallnau [34] developed a model for the component marketplaces that supports prediction of system properties prior to component selection. Other two works extended this one: **(i)** in 2003, Hissam et al. [15] introduced Prediction-Enabled Component Technology (PECT) as a means of packaging predictable assembly as a deployable product. PECT is meant to be the integration of a given component technology with one or more analysis technologies that support the prediction of assembly properties and the identification required component properties and their possible certifiable descriptions; and **(ii)** during 2003, a CMU/SEI's report [39] extended Hissam et al. work [15], describing how component technology can be extended in order to achieve Predictable Assembly from Certifiable Components (PACC). This initiative is developing technology that will predict the runtime behavior of assemblies of software components.

In 2003, Meyer [24] highlighted the main concepts behind a trusted component along two complementary directions: a "*low road*", leading to certification of existing components (e.g. defining a component quality model), and a "*high road*", aimed at the production of components with fully proved correctness properties. However, these two directions are still ongoing research.

As we can note, the efforts to develop a component certification standard is only in its beginning. In our currently proposal (Figure 1 and Figure 2) for component certification we have three steps: **(i)** Data Collection; **(ii)** Define, Design and Plan; and **(iii)** Evaluation. Generally, for component quality assurance, typically, it is needed a set of information about the software component and the environment (i.e. a *Data Collection* step) in order to evaluate its quality using a set of techniques and methods. After that, the definition, design and plan of the evaluation should be done in order to provide information as complete as the evaluation team can about the certification activity (i.e. *Define, Design and Plan* step). Finally, with the certification plan in hand, the evaluation activity is executed, data are collected and recommendations are given (i.e. *Evaluation* step). We considered that these three steps are essential in an efficient component certification activity.

3. The Overall Component Based Processes

As said earlier, there are four processes with different purposes: *component development*, *component certification*, *system development* and *component selection*. These processes differ significantly in different organizational and business settings. We distinguish different types of component-based development [11]:

- **COTS-based development**, where components are developed by some organizations to be marketed as such, and systems are built by other organizations using components available on the marketplace, with no direct influence on the component providers
- **Product-line development**, where a single organization develops components for internal reuse in several products, and
- **Architecture-driven component development**, where components are developed as the result of a top-down system design process and not for reuse, but where the component-based paradigm is adopted e.g. to enforce a well-modularized the system and to be able to use the benefits of standard service of a component technology.

The need for both component certification and a systematic selection process is largest for COTS-based development and product-line development, and smallest for architecture-driven development. For this reason we will therefore outline the process for COTS-based development and then describe how the picture changes for product-line development.

3.1 Commercial-Off-The-Self (COTS)-Based Development

The COTS-based development is based on the idea that the COTS provider develops components and makes them available on the market, while product developers search for them and use them. Here we have a clear distinction between component providers and component users. The component development process is separated from the system development process, and they are connected by component certification and component selection processes.

The entire process is shown on Figure 1 by using the simplest sequential model. Of course any type of development process (sequential or evolutionary) can be used in development of both systems and components. The component development process **(1)** starts from requirements elicitation of the software components, passing from its design and implementation using some architectural standards, and considering some environment constrains, until the verification & validation of this component for further deployment. The process continues with the certification process **(2)**: After fully tested internally the component is delivered to a certification organization. Based on the component information (i.e. documents, source code, tutorials, examples, and so on) and on the target domain, the certification organization performs a set of activities to verify the component properties and to issue the certificate. The techniques, methods and tools are selected to evaluate the component according the certification plan defined during the certification process.

The components are stored in the organization's private repository **(3).** Then, the component development organization can also make some products public **(4)**. It may decide to publish a combination of earlier, certified versions of a component as well as newer, not yet certified versions. With "make public" we mean not necessary making itself immediately available, but in many cases make public the information about the component.

Once made public, customers – i.e. system developers – can find these components for evaluation during its component selection process **(5)**. This selection (i.e. *Find* step) is made by the system developer using search mechanism available in those system repositories (the search can be executed through keywords and queries).

The system developers need to evaluate which component best fit the system requirements in order to be integrated into it **(6)**. During *high-level evaluation*, the main goal is to evaluate a set of documentations, the component constraints and the component reputation based on its owner. The idea is to refine a pool of selected components to be input to the next activity; and during *prototyping evaluation*, where the system developers create

prototypes with the components selected previously and selects the superior component to compose the system, based on the system requirements.

When a component is selected and integrated into the system, the system development organization enters a mode of maintaining and evolving the system which may include integrating newer versions of the component in the future **(7)**. After that, the assets generated by the system development process (including acquired components) are stored in the repository system of this organization to allow future reuse in other projects.

In addition to this basic flow between activities, there are several other loose interconnections (not shown in the figure). The component requirements are affected by system requirements, either through a close business relationship with some system developer(s), or by following trends in the domain of the component. And conversely, system requirements may be influenced by requirements (or capabilities) of existing components. It could be noted that for software components, many of these requirements are closely related to design and system integration (e.g. what platform and component technology the component is designed for).

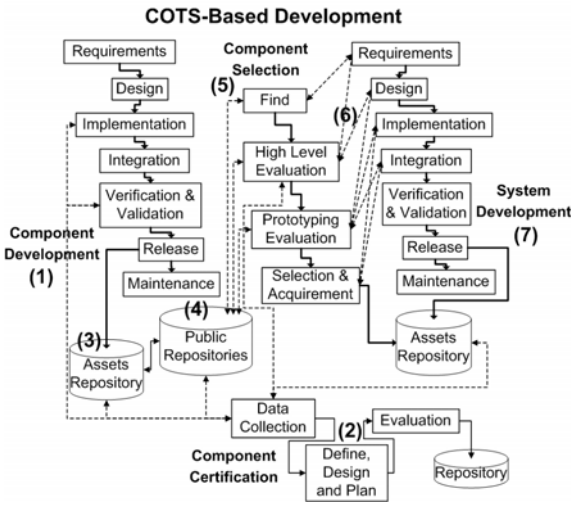

Figure 1. COTS-Based Development.

3.2 Software Product Line (SPL)-Based Development

According to Clements & Northrop [6], a Software Product Line (SPL) is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way. Similar to COTS components, components developed in a SPL development organization are intended to be reusable in several (possibly unknown) systems. The difference from COTS components is that SPL components are developed in-house to address the needs of the product development, and are not aimed to be marketed publicly. Also, since all processes take place within the same organization they can be better planned and performed more efficiently.

In a SPL organization (Figure 2) we have the following activities: **(1)** the system development can impact component development much more directly (i.e. during the development of new versions or branches in a product line, the system requirements can be forwarded directly to the component development). The same thing happens with the component selection process **(2)** because by having access to a repository system, a component can be selected during its development. Also, new component requirements derived from the system requirements can be added to the existing component requirements before the component release; in this way the system process interferes in the component development.

The certification process **(3)** and goals may be different for SPL development compared to COTS-based development because of the following reasons:

- **(i)** The certification organization (which performs the certification) is not necessary external. There are several reasons for this: a) to keep organization's business/technical goals secret; b) to acquire knowledge from the development organization; and c) to have the same overall goals as the development organization. At the same time, keeping certification within the development organization may compromise objectivity. This could possibly be solved by outsourcing or subcontracting an external certification organization.
- **(ii)** While in COTS-based development the goal of certification is to certify the component in general (e.g. some standard measures of performance), in a SPL-based organization some particular properties or usage contexts might be of special interest (e.g. throughput and latency for a certain common input and number of users, etc.).
- **(iii)** The certification methods can include tools and methods used in the development process (i.e. another type of reuse which increases overall development efficiency).
- **(iv)** A SPL organization has only one repository to store, search and use the components **(5)**. Thus, finding software assets from previous developments is easier and faster.

We can conclude that organizations to COTS-based certification and SPL-based certification may be more efficient and more accurate, but less independent; a certification of a component may be seen as a standard verification of the component.

Different from COTS-Based Organization, if it is necessary more precise evaluation of some properties during the component selection (*prototyping evaluation)*, the component certification is called to do this task **(4)**. This happens when more than one component is a good solution to be integrated into the system, and the component certification process is thus used to evaluate the particular properties of interest for the target system and aid the software engineer in his/her decision. Since in a SPL organization the selection and certification processes occur inside the same organization it is possible to integrate the processes and efficiently exchange information.

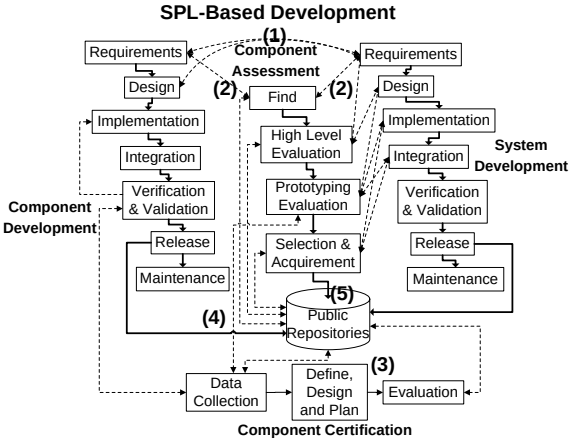

Figure 2. SPL-Based Development.

4. Component Selection *vs.* Component Certification

This section outlines the main characteristics of the component evaluation made during component selection and component certification. In many cases, the actual methods employed to evaluate a component could be identical (e.g. a stress test under some simulated circumstances, or a test of functional correctness using a very large input set), but at the same time there are some fundamental differences that greatly affects how the evaluation can be carried out.

During *component selection*, the decision how many components to evaluate, how thoroughly to test them, what properties of the components to evaluate, etc. is dependent on the goals and objectives of the system development organization concerning the envisioned system itself and the characteristics of the development process, such as acceptable cost and risk. The goal is to select a component that best meets the requirements and constraints among many candidates. The process can be characterized as a gradual filtering, from many potential components to fewer which – with some confidence – are believed to suit the system requirements best. As described earlier, the evaluation can be divided into a high-level evaluation (*high-level evaluation*) which only evaluates information about components (gathered e.g. from the Internet or from the vendors themselves) and an evaluation where the actual component is a tested and prototyped (*prototyping evaluation*).

During *component certification*, it is the component vendor who orders a certification of the component. Typically, the certification concerns the technical characteristics of the component itself and the outcome is information about the component. The input is one single component (not many). The properties of interest are those which the component vendor believes will pay back in larger incomes of their component, by charging a higher price and/or selling more components since the certification is a quality stamp (it could also be noted that there may be standards and regulations mandating certain evaluations).

Figure 3 visualizes the evaluations made during the two different processes.

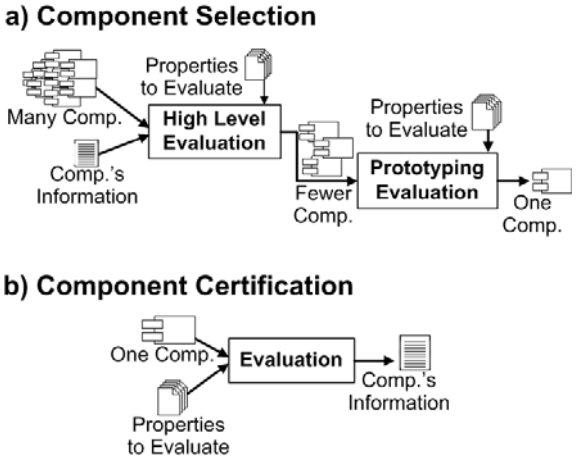

Figure 3. The inputs and outputs of the Component Selection and Component Certification activities.

While there is a main similarly between certification and evaluation is in the fact that in both cases a component is evaluated against some set of properties of interest, there a number specific differences:

- As already said, the properties to be evaluated come from different sources (in the case of component selection properties are derived from system requirements, while properties to be certified are demanded by the component vendor and/or standards and regulations);
- As already said, during component selection, evaluation is performed in order to select the best fit component (among several) for a system, while component certification is performed in order to make assertions about certain properties for a specific component;
- During component selection, evaluation only needs to last until the evaluator has enough confidence to make a selection (i.e. less critical requirements and/or a lower accepted degree of

confidence and/or a larger differences between components' capabilities would allow a less rigorous evaluation);
- During component selection, some evaluation can be done with only information about a component (including information about the vendor, etc), while certification always means evaluation of an actual component (documentation, examples of use, source-code in many cases, etc); and
- During component certification, the documentation of the evaluation is the most important outcome (perhaps in the very condensed form of a "certificate"), while during component selection the most important result is the decision to use a certain component or not.

From this list it follows that the actual methods to evaluate a certain property of interest could be identical (i.e. the same benchmark performance test). However, from a process point of view the methods would be carried out differently. Perhaps the most important difference is, when the goal is to select the component that fits best for a specific system; the evaluation can be very flexible and opportunistic, which is noticeable in e.g. the available evaluation methods PECA [7] and PORE [23, 28] (i.e. at each point in time one may ask whether enough information is gathered to be able to make the decision to select a component with enough confidence, or if not what is the most important property to evaluate next and how).

Component selection is an established process in many organizations, but for component certification there is neither any well-defined standard adopted by the software industry [26, 38], nor is there an established component quality model which is a required foundation for component certification with objective and widely understood results. This fact may be due also to the relatively novelty of this area [14]. For this reason further research is needed in order to develop processes, methods, techniques, and tools aiming to obtain well-defined standards for component certification [2].

To illustrate the difference of evaluation context between selection and certification processes we are outlining three quality properties I namely accuracy, performance and safety.

An important difference between the evaluations performed is that the evaluation is done with a system context during component selection and without a system context during component certification. Thus, the component properties can be evaluated through different perspectives. In this way, the properties described next are based on two standards: the SQuaRE project [16] and IEC 61508 [17]. The SQuaRE project [16] has been created specifically to make two standards converge, trying to eliminate the gaps, conflicts, and ambiguities that they present. These two standards are the **(i)** ISO/IEC 9126, which defines a quality model for software product, and **(ii)** ISO/IEC 14598, which defines a software product evaluation process, based on the ISO/IEC 9126. Thus, the quality model contains 6 characteristics and 27 sub-characteristics, it is necessary to discuss each one with or without context. The IEC 61508 [17] outline the only the safety property and its related characteristics.

Next we present these three properties and the difference between evaluation contexts:

- *Accuracy and suitability.* Although accuracy, defined as a sub-characteristic *functional* characteristic can be verified (according to some standardized means, and provided that there are exact descriptions of a component's semantics in a standardized format), *suitability* sub-characteristic, also a sub-characteristic of functional characteristic) cannot be evaluated without a system context [13];
- *Performance ("efficiency" in SQuaRE project)* characteristics varies with the platform chosen (slight differences in e.g. schedulers, memory managers, or in general computer architecture may result in very different runtime characteristics) but it may be possible to package assertions or test results either with information about the test environment attached (in a standardized format, or even better a standardized environment is used) or by parameterize the results (see our previous work where an analysis method for component worst-case execution time supports partial evaluation without system context which is packaged parameterized on input ranges which – if standardized and widely used – would be one possible solution [13]); and
- *Safety* is a system property which depends on the external environment context [17, 39, 42] and it is not possible to certify the "safety of a software component". It is however possible to verify and certify different properties that may have impact on safety.

The examples above show some limitations of component certification and the necessity of evaluating certified components also in the selection process. However the existence of certified components would enable an efficient filtering of component candidates during the selection process.

5. Challenges

Component selection happens already today in a large scale (evidently, since there is a component marketplace and many systems are built using components). However, component certification is neither completely understood nor established in industrial praxis. The examination of the

fundamental differences between component selection and component certification unveil some important challenges to be addressed for software component certification to be established:

- **Standardization:** standards are needed so that test and analysis results as well as issued certificates are universally recognized and have an agreed-upon meaning. To this end, and also useful for component selection, analysis and test methods need to be designed enabling a meaningful division and packaging of tests and analyses between component vendors, certifiers, certification institutes, system developers and software organizations;
- **Costs:** It must be considered who will pay for the cost of component certification, and if it is worth the cost. For component selection the cost is intrinsic to the software project and must be included in the project plan;
- **Liability & Warranties:** Which kind of warranties does the customer have from a component certified in case the component fails during operation? Presumably, if the evaluation context is specified in enough detail it becomes possible to limit the risks for the certifier.

In summary, certification promises to be useful only if it already is established in a large scale. It will require very large efforts and hence incentives to create the necessary standards and establish certification organizations, and to pay for each component certification. These incentives exist within certain large SPL organization [11], but we can well ask whether these incentives exist for the COTS marketplace?

Another open question is the relation between component certification and certification of professionals and of organizations (which exist today, for example, ISSO 9000, CMMI, MS, Cisco and Java certification). If certified professionals in a certified organization develop software according to accredited procedures, what would be the extra value of certifying the actual software?

6. Concluding Remarks and Future Directions

This paper presented two main processes to assure the quality of the software components developed: selection process and certification process. We carefully collected a set of relevant works in order to present a brief review of these areas. After that, we discussed the application of the two processes in two kinds of software organizations: product line and COTS-based development organizations. The main differences and commonalities of these two processes were also presented.

As future work we intend to perform a case study in order to validate the ideas provided here. Some specific elements have been studied [42] but it remains to validate the overall processes. The current theoretical results presented in the present paper contribute in understanding these processes and can be used as a basis for further empirical studies.

Acknowledgements

This work is partly funded by the Swedish Foundation for Strategic Research (SSF). The authors would like to thank our colleagues at Mälardalen University (Department of Computer Science and Electronics), for their reviews and suggestions during this work.

References

1. A. Alvaro, E.S. Almeida and S.R.L. Meira, “Software Component Certification: A Survey”, In: *The 31st IEEE EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), Component-Based Software Engineering (CBSE) Track*, Porto, Portugal, 2005.
2. A. Alvaro, E.S. Almeida and S.R.L. Meira, “Component Quality Assurance: Towards a Software Component Certification Process”, *In the IEEE International Conference on Information Reuse and Integration (IRI)*, Las Vegas, USA. 2007.
3. C. Alves, J. Castro, "CRE: a systematic method for COTS components Selection", *XV Brazilian Symposium on Software Engineering (SBES)*, Rio de Janeiro, 2001.
4. J. Bhuta, B. Boehm, “A Method for Compatible COTS Component Selection”, 4th International Conference on COTS-Based Software Systems, Spain, Springer, LNCS 3412, 2005.
5. L. Chung and K. Cooper, “Defining Goals in a COTS-Aware Requirements Engineering Approach”, *Systems Engineering*, Vol 7, No. 01, pp. 61-83, 2004.
6. P. Clements and L. Northrop, “Software Product Line: Practices and Patterns”, *SEI Series in Software Engineering*, Addison Wesley, USA, 2001.
7. S. Comella-Dorda, J. Dean, E. Morris, and P. Oberndorf, “A Process for COTS Software Product Evaluation”, *Proceedings of the 1st International Conference on COTS-Based Software System* (ICCBSS), Orlando, Lecture Notes in Computer Science (LNCS), pp. 4-6, 2002.
8. B. Councill, “Third-Party Certification and Its Required Elements”, *The 4th Workshop on Component-Based Software Engineering (CBSE)*, Lecture Notes in Computer Science (LNCS), Springer-Verlag, Canada, May, 2001.
9. I. Crnkovic, H. Schmidt, J. Stafford and K. C. Wallnau, *Proceedings of the 5th Workshop on Component-Based Software Engineering(CBSE): Benchmarks for Predictable Assembly, In: The Software Engineering Notes*, Vol. 27, No. 05, 2002.

10. I. Crnkovic, H. Schmidt, J. Stafford and K. C. Wallnau, *Proceedings of the 4th Workshop on Component-Based Software Engineering(CBSE): Component Certification and System Prediction, In: The Software Engineering Notes*, Vol 26, No. 06, 2001.

11. I. Crnkovic, M. Chaudron and S. Larsson, "Component-Based Development Process and Component Lifecycle", *In International Conference on Software Engineering Advances (ICSEA)*, 2006.

12. I. Crnkovic and M. Larsson (eds), *Building Reliable Component-Based Software Systems*, Artech, 2002.

13. J. Fredriksson, R. Land, "Reusable Component Analysis for Component-Based Embedded Real-Time Systems", 29th International Conference on Information Technology Interfaces (ITI), Cavtat, Croatia, IEEE, 2007

14. M. Goulao and F. Brito e Abreu, "The Quest for Software Components Quality", *The 26th IEEE Annual International Computer Software and Applications Conference (COMPSAC)*, England, pp. 313-318, 2002.

15. S. A. Hissam, G. A. Moreno, J. Stafford and K.C. Wallnau, "Enabling Predictable Assembly", *Journal of Systems and Software*, Vol. 65, No. 03, pp. 185-198, 2003.

16. ISO/IEC 25000, *Software engineering – Software product quality requirements and evaluation (SQuaRE)*, Guide to SQuaRE, International Standard Organization, July, 2005.

17. IEEE 61508, International Electrotechnical Commission (IEC), *IEC 61508, Functional safety of E/E/PE safety-related systems*, 1998.

18. J. Kontio, "OTSO: A Systematic Process for Reusable Software Component Selection", *University of Maryland*, CS-TR-3478, UMIACS-TR-95-63, December 1995.

19. D. Kunda, L. Brooks, "Applying Social-Technical Approach For Cots Selection", Proceedings of 4th UKAIS Conference, University of York, McGraw Hill. April 1999.

20. R. Land, L. Blankers, "Classifying and Consolidating Software Component Selection Methods", MRTC report ISSN 1404-3041 ISRN MDH-MRTC-218/2007-1-SE, 2007.

21. Lawlis et al, "A Formal Process for Evaluating COTS Software Products", Computer, vol 34, no 5, 2001.

22. A. Liu, I. Gorton, "Accelerating COTS Middleware Acquisition: The i-Mate Process", IEEE Software, Volume 20, Issue 2, Pages: 72-79, March 2003.

23. N. A. Maiden, C. Ncube, "Acquiring COTS Software Selection Requirements", *IEEE Software*, Vol. 15, No. 02, pp. 46–56, 1998.

24. B. Meyer, "The Grand Challenge of Trusted Components", *The 25th IEEE International Conference on Software Engineering (ICSE)*, USA, pp. 660–667, 2003.

25. M. Morizio, C. B. Seaman, V.R. Basili, A.T. Parra, S. E. Kraft and S.E. Condon, "COTS-based software development: Processes and open issues", *Journal of Systems and Software*, Vol. 61, pp. 189-199, 2002.

26. J. Morris, G. Lee, K. Parker, G.A. Bundell and C.P. Lam, "Software Component Certification", *IEEE Computer*, Vol. 34, No. 09, pp. 30-36, 2001.

27. F. Navarrete, P. Botella and X. Franch, "How Agile COTS Selection Methods are (and can be)?", *Proceedings of the 31st IEEE EUROMICRO Conference on Software Engineering and Advanced Application (SEAA), CBSE Tack*, pp. 160–167, 2005.

28. C. Ncube and N.A.M. Maiden, "PORE: Procurement-Oriented Requirements Engineering Method for the Component-Based Systems Engineering Development Paradigm", *In the 2th International Workshop on Component-Based Software Engineering*, Los Angeles, USA, 1999.

29. C. Ncube and J.C. Dean , "The Limitations of Current Decision-Making Techniques in the Procurement of COTS Software Components", *Proceedings of the First International Conference on COTS-Based Software Systems (ICCBSS)*, Lecture Notes in Computer Science (LNCS), pp. 176-187, 2002.

30. P. Oberndorf, L. Brownsword, E. Morris, C. Sledge, "Workshop on COTS-Based Systems", Special report CMU/SEI-97-SR-019, Software Engineering Institute, 1997

31. J. Poore, H. Mills and D. Mutchler, "Planning and Certifying Software System Reliability", *IEEE Computer*, Vol. 10, No. 01, pp. 88-99, 1993.

32. G. Ruhe, "Intelligent Support for Selection of COTS Products", *Web-Services, and Database Systems: NODe, Web- and Database-Related Workshops*, Erfurt, Germany, Lecture Notes In Computer Science (LNCS), 2003.

33. H. Schmidt, "Trustworthy components: compositionality and prediction", *Journal of Systems and Software*, Vol. 65, No. 03, pp. 215-225, 2003.

34. J. Stafford and K.C. Wallnau, "Is Third Party Certification Necessary?", *The 4th Workshop on Component-Based Software Engineering (CBSE)*, Lecture Notes in Computer Science (LNCS) Springer-Verlag, Canada, 2001.

35. C. Szyperski, *Component Software: Beyond Object-Oriented Programming*. Addison-Wesley, USA, 2002.

36. M. Tziakouris, A. S. Andreou, "A quality framework for developing and evaluating original software components", *Information & Software Technology*, Vol. 49, No. 2, pp. 122-141, 2007.

37. J. M. Voas, "Certifying Off-the-Shelf Software Components", *IEEE Computer*, Vol. 31, No. 06, pp. 53-59, 1998.

38. J. M. Voas and J. Payne, "Dependability Certification of Software Components", *Journal of Systems and Software*, Vol. 52, No. 2-3 , pp. 165-172, 2000.

39. K. C. Wallnau, “Volume III: A Technology for Predictable Assembly from Certifiable Components”, *Software Engineering Institute (SEI), Technical Report*, Vol. III, April, 2003.

40. C. Wohlin, P. Runeson, M. Höst, C. Ohlsson, B. Regnell and A. Wesslén, “Experimentation in Software Engineering: an Introduction”, *Kluver Academic Publishers*, Norwell, 2000.

41. C. Wohlin, and B. Regnell, “Reliability Certification of Software Components”, *In The 5th IEEE International Conference on Software Reuse (ICSR)*, Canada, pp. 56-65, 1998.

42. I. Crnkovic, M. Larsson, O. Preiss, Concerning Predictability in Dependable Component-Based Systems: Classification of Quality Attributes, Architecting Dependable Systems III,, p pp. 257 – 278, Springer, LNCS 3549